Lecturas de Cátedra

Modelos lineales generales en R

Aplicaciones en ciencias agronómicas y ambientales

Facundo José Oddi
Lucas Alejandro Garibaldi

EDITORIAL UNRN

Modelos lineales generales en R

Lecturas de Cátedra

Modelos lineales generales en R

Aplicaciones en ciencias agronómicas y ambientales

Facundo José Oddi
Lucas Alejandro Garibaldi

EDITORIAL
UNRN

Índice

Prefacio

En las ciencias agronómicas y ambientales a menudo es necesario resolver problemas en contextos de incertidumbre. Las razones son el carácter aleatorio de los fenómenos que estudian y el trabajo con información parcial (una muestra), lo que provoca la consecuencia de no tener total certeza de arribar a conclusiones correctas. Dado que esto es casi ineludible, es necesario que los profesionales de estas ciencias adquieran herramientas para tomar decisiones basadas en datos y la capacidad de manejar, cuantificar y transmitir la incertidumbre asociada a la toma de decisiones. La estadística provee el marco conceptual y metodológico para hacerlo y, dentro de esta, los modelos lineales generales (MLG) son una de las herramientas más útiles.

Esta obra aborda los MLG desde una perspectiva aplicada, a partir de la implementación (uso/aplicación) del programa estadístico R. En este abordaje, como estrategia didáctica, los contenidos de los MLG se integran mediante la misma problemática agroecológica: hábitat natural y rendimiento de cultivos. Creemos que esta estrategia es una característica diferencial de este libro. La mayoría de los textos de estadística, en general, y modelos estadísticos, en particular, recurren a múltiples ejemplos para tratar los diferentes contenidos que abarcan. En este sentido, no los reemplaza, sino que los complementa. El objetivo es brindar una visión integrada de los MLG como apoyo a la resolución de problemáticas de índole profesional. En la práctica, la resolución se hace usando las herramientas que provee la estadística y sus modelos, desde la exploración de los datos y la formulación del modelo hasta la toma de decisiones y la realización de predicciones. Sin entrar en detalles, un modelo estadístico permite representar la dependencia de una variable que denominamos *dependiente* con respecto a una o más variables *independientes*.

Las capacidades de R son brevemente desarrolladas en el capítulo 1 como apoyo a los códigos aplicados en los capítulos subsiguientes en los que se introducen modelos lineales (la variable y es predicha por una suma de parámetros) de complejidad creciente: en el capítulo 2, la regresión lineal simple (un predictor cuantitativo); en el capítulo 3, el modelo unifactorial (un predictor categórico); en el capítulo 4, los modelos multifactoriales (dos o más predictores categóricos); en el capítulo 5, la regresión lineal múltiple (dos o más predictores cuantitativos), y en el capítulo 6, las regresiones con variables categóricas (al menos un predictor cuantitativo y al menos un predictor categórico). Estas etiquetas responden a cómo se los encuentra en los textos de estadística más tradicionales.

Finalmente, el capítulo 7 tiene como propósito mostrar que estos modelos, tradicionalmente segmentados, pertenecen a un mismo marco conceptual: el de los MLG. Si bien cada capítulo tiene sus particularidades de acuerdo a los conceptos que introducen, todos (a excepción de la sección dedicada a R) siguen una misma estructura: se presenta el problema y se exploran los datos, se especifica y discute el modelo estadístico, se estiman sus parámetros y se lo evalúa. La figura 1 muestra la estructura del libro.

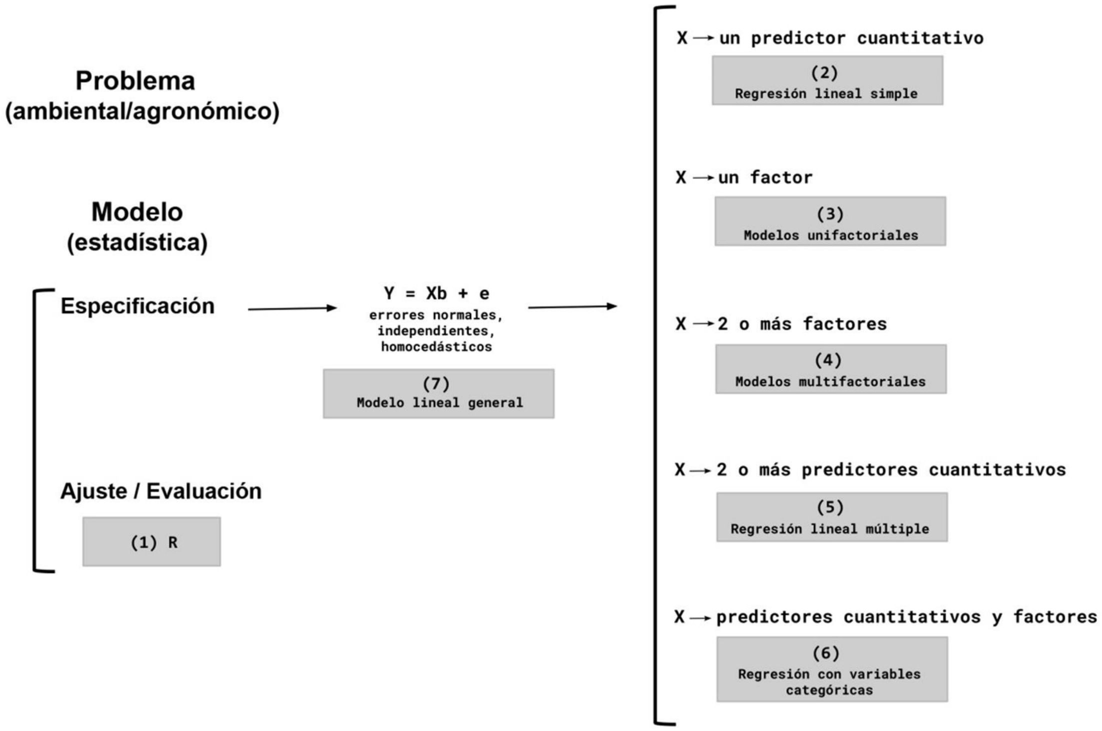

Figura 1. Estructura del libro y recorrido a lo largo de los capítulos

El lector debe poseer el conocimiento de los contenidos de un curso introductorio en probabilidad y estadística (nociones de estadística descriptiva, variables aleatorias, distribuciones de probabilidad, distribuciones por muestreo, estimación puntual y por intervalos, pruebas de hipótesis, comparación de medias, correlación). Si bien en algunos capítulos se abordan conceptos de diseño de muestreo, no se discute sobre aspectos relacionados a la causalidad de las relaciones. En este sentido, la base de datos que utilizamos proviene de un estudio observacional y,[1] como tal, solo nos permite probar la asociación estadística (más específicamente, modelar la dependencia) entre la variable de interés y el conjunto de predictores. Siendo un texto de orientación aplicada, no se presentan demostraciones formales sobre los métodos utilizados. Para la profundización en los fundamentos de los modelos lineales generales, en R y en conceptos relacionados al finalizar la sección se presenta una lista bibliográfica recomendada. Esperamos que el lector encuentre esta obra de utilidad para el abordaje de sus propios análisis.

[1] Los experimentos se pueden clasificar en *manipulativos* y *mensurativos* (u observacionales). En los primeros, a las unidades experimentales seleccionadas se les asigna aleatoriamente un tratamiento con lo cual las respuestas observadas son necesariamente debidas al factor que se evalúa y se puede demostrar causalidad. Por el contrario, en los experimentos mensurativos las unidades experimentales seleccionadas en la muestra ya vienen con el tratamiento (solo se las observa) de modo que la respuesta podría ser causada por un factor que varía conjuntamente con aquel que se evalúa (efecto confundido).

Hábitat natural y rendimiento de cultivos

Uno de los mayores desafíos de la agricultura extensiva es mantener o aumentar la producción sin deteriorar el medioambiente. Algunos de los paradigmas vigentes, como la agroecología y la intensificación ecológica, proponen mantener y promover los hábitats naturales en paisajes productivos para favorecer las funciones que estos proveen a los cultivos (por ejemplo: polinización, control biológico). Esto, además de proveer hábitat para distintos organismos, podría disminuir la utilización de agroquímicos reduciendo el costo ambiental y económico del proceso productivo.

En función de estas ideas, la Asociación Argentina de Consorcios Regionales de Experimentación Agrícola (AACREA), que nuclea productores agropecuarios de toda la Argentina, junto con investigadores del CONICET, explora la relación entre producción y configuración del paisaje. Los datos presentados, publicados por Goldenberg y otros (2023), corresponden a una campaña reciente en la región pampeana y fueron generados por la herramienta DAT (Datos Agrícolas Trazados) de CREA (https://herramientas.redcrea.org.ar/dat-crea/). Con DAT se extrae información a nivel de lote (unidad experimental) sobre manejo, tenencia de la tierra, tipo de ambiente y configuración espacial, además de indicadores productivos. La base de datos de trabajo (tabla 1) constituye una submuestra del conjunto original y corresponde a 105 lotes en los que se sembró girasol. Asumiremos que estos lotes representan una muestra aleatoria e independiente de la población de lotes agrícolas presente en el área de estudio. Las variables de configuración espacial evaluadas son la densidad de borde de hábitat natural en un radio de 1,5 km, expresada en m ha^{-1}, y el tamaño del lote (ha). La cantidad de fertilizante nitrogenado aplicado (kg ha^{-1}), la densidad (n° semillas ha^{-1}) y la fecha de siembra, y el cultivo previo representan variables de manejo. También se informa la región en la que se localiza el lote (existen cuatro regiones codificadas como «ra», «rba», «rc» y «rd»). El indicador productivo, y la variable de interés, es el rendimiento del cultivo (kg ha^{-1}):

Tabla 1. Base de datos utilizada en el libro

lote	rendimiento	borde	tamaño	N	densidad siembra	fecha siembra	cultivo previo	región
1	*2881*	*30,0*	*67*	*6,71*	*5000*	*16/10/2018*	*gram. anual*	*ra*
2	*2339*	*15,3*	*60*	*6,71*	*45000*	*20/10/2018*	*gram. anual*	*ra*
3	*2192*	*34,8*	*62*	*6,71*	*43000*	*19/10/2018*	*gram. anual*	*ra*
4	*2584*	*41,2*	*67*	*6,71*	*45000*	*26/10/2018*	*gram. anual*	*ra*
5	*2312*	*54,5*	*66*	*4,40*	*57000*	*17/10/2018*	*gram. anual*	*rb*
6	*3103*	*63,3*	*15*	*41,05*	*57000*	*23/10/2018*	*maíz*	*ra*
7	*2273*	*21,4*	*50*	*6,71*	*47000*	*24/10/2018*	*maíz*	*ra*
8	*3150*	*34,2*	*72*	*55,80*	*54000*	*20/10/2018*	*maíz*	*ra*
…	…	…	…	…	…	…	…	
105	*2031*	*21,6*	*27,0*	*5,5*	*60000*	*26/10/2018*	*verdeo*	*rc*

Aspectos generales de los modelos estadísticos

Los modelos estadísticos se formulan en lenguaje matemático y presentan dos componentes: una denominada *determinística* y otra que llamamos *aleatoria*.[2] La suma de ambas componentes resulta en el valor de la variable dependiente y. Más específicamente, la componente determinística modela la tendencia promedio de y en función de un conjunto de predictores x que opera con un conjunto de parámetros θ. La componente aleatoria ε expresa la variabilidad en torno al promedio y se modela mediante una distribución de probabilidades d definida por otro conjunto de parámetros ω. Esto quiere decir que el modelo asume que ε se distribuye como «~» d. De manera general, un modelo estadístico puede formularse como:

$$\underbrace{y}_{\textbf{variable}} = \underbrace{f(x;\,\theta)}_{\substack{\textbf{componente}\\ \textbf{determinística}}} + \underbrace{\varepsilon}_{\substack{\textbf{componente}\\ \textbf{aleatoria}}}$$

$$\underbrace{\varepsilon \sim d(\omega)}_{\substack{\textbf{modelo de la}\\ \textbf{componente}\\ \textbf{aleatoria}}}$$

Conceptualmente, aunque dependiendo de la naturaleza de y, cualquier función f puede ser usada en la componente determinística, y lo mismo sucede con las distribuciones de probabilidad de la componente aleatoria. Los parámetros que definen el modelo son constantes poblacionales desconocidas y se denomina *parametrización* a la manera en que se disponen en ambas componentes. A lo largo del libro utilizamos la variable *respuesta* o *de interés* como sinónimos de variable dependiente. De igual manera, nos referimos a las variables independientes como *predictores* o *covariables*.[3]

Veamos un ejemplo concreto. La ecuación USLE expresa que, bajo determinadas condiciones climáticas, edáficas y de manejo, la pérdida de suelo por erosión se incrementa linealmente con el largo de la pendiente del terreno l.[4] Por lo tanto, podríamos describir la pérdida de suelo p como una multiplicación entre l y un coeficiente b de pérdida por unidad de distancia:

$$p = b \times l$$

Esto se denomina *modelo matemático* o *determinístico*, ya que a cada valor de la variable independiente l le corresponde un único valor de la variable dependiente p. En relación a la

---

[2] También se la conoce como componente *estocástica* o *término de error*.

[3] El término *covariable* en general hace referencia a variables independientes cuantitativas que no son de interés principal.

[4] La ecuación Universal de Pérdida de Suelo está formulada como $p = r \times k \times l \times s \times c \times p$. Donde p es la perdida de suelo; r cuantifica la erosividad de la lluvia; k la erodabilidad del suelo; s la inclinación del terreno; c el uso y manejo del suelo; la mecánica de laboreo. El término l considera el efecto de la longitud de la pendiente del terreno (la distancia que el agua de lluvia recorre por gravedad) y se expresa en relación a la pérdida de suelo cada 22,1 m.

formulación general, $b \times l$ sería lo que indicamos como $f(x; \theta)$, con «b» conformando el conjunto de parámetros poblacionales θ y l la variable independiente x.

Al incluir una componente aleatoria, los modelos estadísticos reconocen explícitamente que existe variabilidad y, por lo tanto, la realidad no puede ser descripta de manera perfecta por una ecuación matemática. Aplicado a nuestro ejemplo, esto quiere decir que la pérdida de suelo predicha por la componente determinística del modelo difiere de la real en una magnitud ε:

$$p = b \times l + \varepsilon$$

La diferencia entre esta ecuación y la anterior es conceptualmente relevante. En primer lugar, ahora estamos indicando que la pérdida de suelo se debe en parte a la longitud de pendiente ($b \times l$) y en parte a otras causas (ε). Además, el término aleatorio implica que para una misma longitud de pendiente existen múltiples valores de p. Ello se relaciona a la segunda ecuación del modelo estadístico planteado anteriormente de forma general, que impone una distribución de probabilidades a ε, y en consecuencia también a p. Por ejemplo, podríamos asumir que ε sigue una distribución normal con parámetros media μ y varianza σ^2. Con esto, podemos definir el modelo estadístico de la siguiente forma:

$$p = b \times l + \varepsilon$$
$$\varepsilon \sim \mathcal{N}(\mu; \sigma^2)$$

Considerando que si $\varepsilon = 0$ entonces $p = b \times l$, asumir que ε sigue una distribución normal con $\mu = 0$ implica que, para una longitud de pendiente dada, las pérdidas de suelo se distribuyen normalmente con media igual a $b \times l$. En otras palabras, conocer el parámetro b y el largo de la pendiente nos permite predecir la pérdida de suelo promedio. Es necesario notar que es sumamente importante interpretar los parámetros en términos del problema. En este caso, b nos informa el cambio en la pérdida de suelo que se espera al aumentar una unidad la longitud de la pendiente. En cambio, si parametrizamos el modelo como $p = 1/ b \times l$, el parámetro b indica cuánto debe modificarse la longitud de la pendiente para que la pérdida de suelo aumente una unidad. Por su lado, el parámetro σ^2 de la componente aleatoria nos sirve para conocer cuán probable es observar determinada pérdida de suelo dada una cierta longitud de pendiente.

Una característica distintiva de los modelos estadísticos es su vínculo con los datos. Los datos son utilizados para estimar los parámetros del modelo y evaluar su comportamiento. En nuestro modelo, por ejemplo, debemos asignar valores a b y a σ^2. Para ello necesitamos un conjunto de pares de datos p, l (para los que podríamos seleccionar sitios con diferentes largos de pendiente y en cada uno de ellos registrar la pérdida de suelo) y utilizar algún criterio estadístico con el cual encontrar los mejores valores de b y σ^2, de acuerdo al criterio seleccionado. Esto es lo que llamamos *el ajuste del modelo* y se basa en la teoría de la estimación estadística. Una vez ajustado el modelo, usamos los datos para evaluar si el modelo los describe razonablemente bien y si cumple con sus supuestos, por ejemplo, la distribución de ε.

R

El objetivo de este libro no es enseñar a programar en R. Para ello existe una amplia bibliografía (ver por ejemplo https://www.bigbookofr.com/), la cual está en continua expansión, con diversos énfasis y niveles de profundidad. La mayoría de los textos están escritos en inglés como los de Spector (2008), Matloff (2011), Crawley (2013), Davies (2016), Cotton (2017), Lander (2017). Entre las referencias más actuales se pueden consultar Venables y Smith (2021), Wickham (2021), Gagolewski (2023), o las últimas ediciones del libro de Kabacoff (2022) y del popular R *for Data Science* de Wickham y otros (2023). También existen obras en castellano que se pueden explorar desde la página de R (https://cran.r-project.org/other-docs.html#nenglish), y están disponibles las traducciones de los libros *R for Begginers* de Paradis (2003) y la del mencionado libro de Wickham y otros, en este caso a través de un proyecto colaborativo de la comunidad de R de Latinoamérica para que sea más accesible en la región. Muchos libros se escriben con el paquete `bookdown` (ver https://bookdown.org/) y sus contenidos se encuentran en línea.

De suma utilidad son también los foros de consulta de los usuarios de R, como por ejemplo:

- https://stackoverflow.com/
- https://www.r-bloggers.com/
- https://rpubs.com/
- https://community.rstudio.com/
- https://stats.stackexchange.com/questions/tagged/r
- https://www.reddit.com/r/RStudio/

Nuestro foco está puesto en la resolución de problemas ambientales mediante modelos lineales generales, lo cual se lleva a cabo con R. En otras palabras, R es una herramienta y no un fin. No obstante, hemos incluido un capítulo introductorio sobre R base y en el resto del libro, al aplicar funciones, brindamos información de modo que el lector comprenda su uso. Este lenguaje es sumamente versátil por lo que existen muchas formas de realizar lo mismo, y se generan constantemente nuevos paquetes para tareas específicas. En este sentido, el libro provee funciones comúnmente usadas para este tipo de análisis.

Códigos y sintaxis

Los comandos y las salidas de R han sido creados con el paquete `rmarkookdown`. Estos aparecen separados del texto y el lector los puede identificar porque se utiliza la tipografía `Consolas` sombreada en gris. Todo lo que es código se muestra en rojo. Los comentarios a los códigos se indican con violeta y siempre precedidos por el símbolo de numeral. Cuando reproducimos la ejecución de un código, los resultados se muestran en color negro y también son precedidos por el sigo numeral. Esto difiere de la salida en la consola de R (en donde el resultado se muestra sin el numeral) y permite que no interfieran en caso que el usuario copie el código y lo reproduzca en R. El siguiente es un ejemplo de un código comentado y su salida:

```
dat <- c(2,4,5,5,11) # creamos el objeto dat
dat

## [1]  2  4  5  5 11
```

Cuando en el texto nos referimos a algún comando, objeto o función de R lo indicamos de igual manera que con el código, como por ejemplo cuando indicamos que utilizamos el paquete `rmarkookdown`. En el caso de la referencia a una función, esta se presenta seguida de un paréntesis, como por ejemplo `read.table()`.

Es recomedable seguir los capítulos frente a una computadora. Para esto es necesario copiar los códigos directamente o utilizar los scripts que acompañan la publicación de este libro. El archivo de datos con el que trabajamos (*lotes.txt*) está guardado como texto delimitado por tabulador y con coma como separador decimal. Los scripts y los datos se encuentran disponibles en el siguiente enlace:

https://bit.ly/49XnCOu

Lista de referencias bibliográficas

Anderson, David R., Dennis J. Sweeney y Thomas A. Williams. (2012). *Estadística para administración y economía*. Cengage Learning.

Bowerman, Bruce L., Richard T. O'Connell y Anne B. Koehler. (2007). *Pronósticos, series de tiempo y regresión: un enfoque aplicado*. Ed. Cengage Learning.

Bretz Frank, Torsten Hothorn y Peter Westfall. (2010). *Multiple comparisons using R*. CRC Press.

Bolker, Ben M. (2008). *Ecological models and data in R*. Princeton University Press.

Burnham, Kenneth P., David R. Anderson y Kathryn P. Huyvaert. (2011). AIC model selection and multimodel inference in behavioral ecology: some background, observations, and comparisons. *Behavioral Ecology and Sociobiology*, *65*(1), pp.23-35.

Cotton, Richard. (2013). *Learning R*. O'Reilly Media.

Christensen, Ronald. (2011). *Plane answers to complex questions. The theory of linear models*. Springer.

Davies, Tilman M. (2016). *The book of R*. No Starch Press.

Faraway, Julian J. (2005). *Linear models in R*. Chapman & Hall/CRC.

Fox, Gordon A., Simoneta Negrete-Yankelevich y Vinicio J. Sosa. (2015). *Ecological statistics: contemporary theory and application*. Oxford University Press.

Fox, John y Sandford Weisberg. (2011). *An R companion to applied regression*. Sage Publications.

Gagolewski, Marek. (2023). *Deep R programming*. Zenodo. https://doi.org/10.5281/zenodo.7490464

Garibaldi, Lucas A., Francisco J. Aristimuño, Facundo J. Oddi y Florencia Tiribelli. (2017). Inferencia multimodelo en ciencias sociales y ambientales. *Ecología Austral*, *27*(3), pp. 348-363.

Garibaldi, Lucas A., Facundo J. Oddi, Francisco J. Aristimuño y Aliosha N. Behnisch. (2019). *Modelos estadísticos en lenguaje R*. Editorial UNRN.

Gelman, Andrew y Jennifer Hill. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press.

Goldenberg Matías G., Fernanda A. Santibañez Ossa, Alfred Burian, Ralf Seppelt, Emilio H. Satorre, Gustavo D. Martini y Lucas A. Garibaldi. (2023). Landscape configuration is an important predictor of sunflower yield in the argentinean pampas region. *Ecología Austral*, *33*(1), pp. 170-177.

Hsu, Jason C. (1996). *Multiple comparisons. Theory and methods*. Chapman & Hall.

James, Gareth, Daniela Witten, Trevor Hastie y Robert Tibshirani. (2013). *An introduction to statistical learning with applications in R*. Springer.

Jones, Elinor, Simon Harden y Michael J. Crawley. (2022). *The R book*. Wiley.

Kabacoff, Robert I. (2022). *R in action*. Manning Publications.

Matloff, Norman S. (2011). *The art of R programming. A tour of statistical software design.* No Starch Press.

Murtaugh, Paul A. (2014). *In defense of P values.* Ecology, *95*(3), pp. 611-617.

Lander, Jared. (2017). *R for everyone: advanced analytics and graphics*. Addison-Wesley Data & Analytics Series.

Logan, Murray. (2010). *Biostatistical design and analysis using R*. A practical guide. Wiley.

Paradis, Emmanuel. (2003). *R para principiantes*. https://cran.r-project.org/doc/contrib/rdebuts_es.pdf

Perelman, Susana B., Lucas A. Garibaldi y Pedro M. Tognetti. (2019). *Experimentación y modelos estadísticos*. Editorial Facultad de Agronomía, UBA.

Pinheiro, José C. y Douglas M. Bates. (2000). *Mixed-effects models in S and S-plus*. EdSpringer.

Schad, Daniel J., Shravan Vasishth, Sven Hohenstein y Reinhold Kliegl. (2020). How to capitalize on a priori contrasts in linear (mixed) models: a tutorial. *Journal of Memory and Language*, *110*,art. 104038.

Seber, George A. F. (2015). *The linear model and hypothesis*. Springer.

Spector, Phil. (2008). *Data manipulation with R*. Springer.

Venables, William. N., David M. Smith, y The R Core Team. (2021). *An introduction to R*. https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf

Webster, Allen L. (2000). *Estadística aplicada a los negocios y la economía*. McGraw-Hill.

Wickham, Hadley. (2021). *Mastering shiny: build interactive apps, reports, and dashboards powered by R*. O'Reilly Media. https://mastering-shiny.org/

Wickham, Hadley, Mine Çetinkaya-Rundel y Garret Grolemund. (2023). *R for Data Science*. O'Reilly Media.

Wickham, Hadley y Garret Grolemund. (2016). *R para ciencia de datos*. https://es.r4ds.hadley.nz/
Zuur, Alain F., Elena N. Ieno, Neil Walker, Anatoly A Saveliev. y Graham M. Smith. (2009). *Mixed effects models and extensions in ecology with R*. Springer.

Capítulo 1. Introducción al lenguaje R

1. 1. R y Rstudio

R es un entorno informático y un lenguaje de programación para el análisis de datos. Es un lenguaje porque las instrucciones para el análisis de datos se realizan mediante códigos, pero además provee el entorno computacional que los interpreta. En otras palabras, en R comunicamos lo que queremos hacer en forma directa, sin la intermediación de botones. Esto representa una ventaja para la transparencia y replicabilidad del análisis, además de que permite la automatización de tareas, aunque requiere un esfuerzo del usuario para aprender la sintaxis del lenguaje.

1. 1. 1. Descarga e instalación de R

El programa R se descarga desde «CRAN» (The Comprehensive R Archive Network), la red de R que almacena todas las versiones base y los paquetes (https://cran.r-project.org/). Conviene siempre descargar la última versión (se actualiza todos los años). En este libro usamos la versión 4.2.1. Para la instalación y ejecución es necesario tener en cuenta el sistema operativo (Windows, OSX, Linux) y la arquitectura del procesador (32 bites, 64 bites). En versiones anteriores, en Windows por default se descargan e instalan dos versiones ejecutables de R, cuyos iconos en el escritorio son *i386* (32-bit) y *x64* (64-bit), y en este caso se debe elegir la correspondiente al sistema operativo.

1. 1. 2. Scripts

Antes dijimos que en R damos instrucciones mediante códigos o comandos. A los códigos podemos escribirlos cada vez que necesitamos dar una instrucción o, de manera más eficiente, ya tenerlos escritos y acceder desde R. Para esto usamos *scripts*, que son secuencias de comandos contenidas en documentos de texto con extensión «.R». Los scripts pueden leerse desde cualquier editor de texto y tienen la particularidad de que R puede ejecutar los códigos que contienen. La gran ventaja de utilizar scripts es que los códigos quedan guardados, lo cual permite ejecutarlos posteriormente y compartirlos con otros usuarios. Es una buena práctica que los scripts puedan ser leídos, ejecutados y entendidos por otras personas, o incluso por nosotros mismos luego de un tiempo de haberlos escrito. Para ello, es necesario ser prolijos y ordenados agregando comentarios claros y concisos (ver la sección «Ejecución de comandos»).

1. 1. 3. RStudio

Si bien podemos usar R en forma directa, es conveniente ejecutar los scripts desde un entorno integrado de desarrollo (IDE, por sus siglas en ingles). Los IDE facilitan la escritura y revisión de los códigos de los scripts, y hacen más eficiente la gestión del entorno de trabajo. Uno de los IDE más utilizados es «RStudio», el cual se descarga desde el sitio https://www.rstudio.com/products/rstudio/download/.

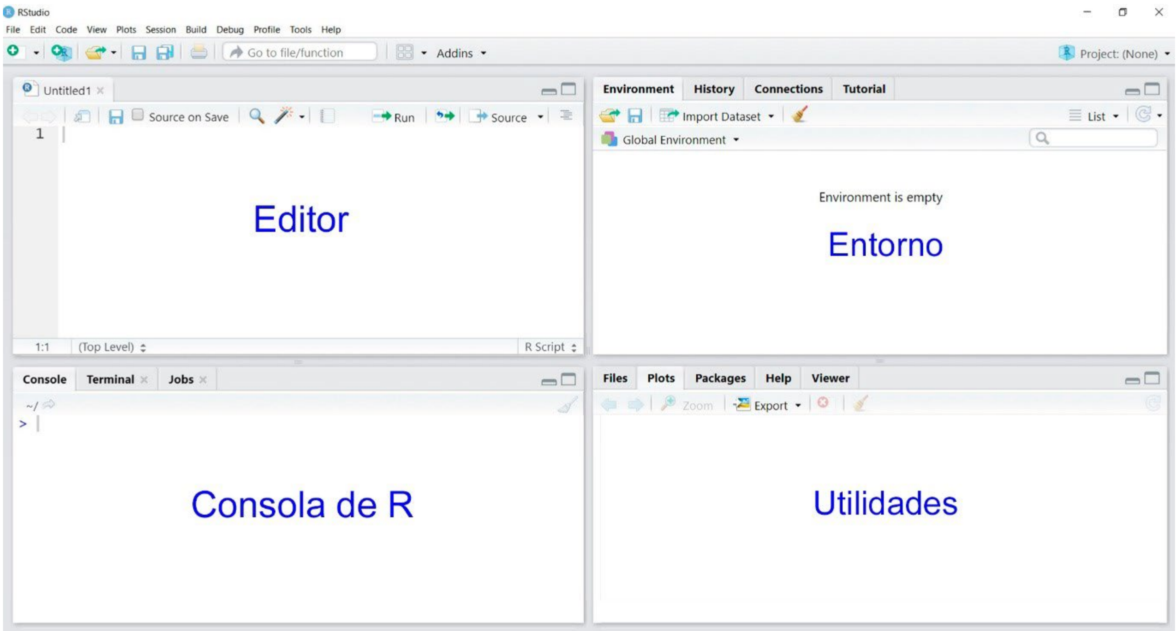

Figura 1. 1. Interfaz de RStudio

1. 2. Lo básico para trabajar con R (y RStudio)

Una vez instalados R y RStudio podemos comenzar a trabajar. RStudio tiene, por defecto, una interfaz que se organiza en cuatro paneles en una estructura de 2 x 2 (figura 1. 1):

1. Izquierda inferior, la consola de R: se muestran los códigos ejecutados y las salidas no gráficas. Podemos escribir y ejecutar los códigos directamente.
2. Izquierda superior, el editor de códigos: se abren y visualizan los scripts. Su apariencia facilita la edición y revisión. Los objetos, funciones y argumentos se muestran con diferentes colores y se indica cuándo hay errores de sintaxis en el código. Podemos comentar los scripts y dar una estructura de secciones para ordenarlos. Decimos que corremos un script cuando desde este panel damos la orden de ejecutar las líneas de código en la consola de R.
3. Derecha superior, el entorno: muestra los comandos ejecutados y los objetos creados durante la sesión. Podemos importar datos y crear espacios de trabajo.
4. Derecha inferior, utilidades: cuenta con varios subpaneles organizados en pestañas. Dentro de estas, desde «Files» accedemos al disco duro, «Plots» almacena las salidas gráficas, «Help» muestra las ayudas que solicitamos y «Packages» es el gestor de paquetes.

1. 2. 1. Codificación

Utilizar el sistema «UTF 8» garantiza que se puedan abrir los scripts tanto en Linux como en Windows sin errores de codificación de caracteres. Esto lo podemos hacer desde RStudio realizando lo siguiente:

1. Ir a «Tools».
2. Luego a «Options».
3. Cambiar «Default texting encoding» a «UTF-8».

1.2.2. Ejecución de comandos

Desde el editor de códigos de RStudio (panel izquierdo superior) podemos enviar, o no, las sentencias a la consola de R (panel izquierdo inferior). Para ser enviadas a R, nos ubicamos con el cursor sobre la línea del código a ejecutar y presionamos «Ctrl» + «Enter» o el botón «Run» que se encuentra arriba a la derecha del panel de edición. Por ejemplo, al escribir `"hola"` en el editor y ejecutar el comando, en la consola vemos:

```
"hola"

## [1] "hola"
```

Veremos este mismo resultado si escribimos `"hola"` en la consola de R y apretamos la tecla «Enter». La única diferencia fue que ahora ejecutamos el comando directamente desde R sin pasar por el editor. En la consola de R, el cursor es el símbolo `>` e indica que R está listo para recibir un comando (el término inglés en la jerga de los lenguajes de programación es «prompt»). Todo texto que tenga un signo `#` (numeral) delante no interfiere con la sentencia, es decir, R lo ignorará. Esto es sumamente útil dado que podemos realizar comentarios de ayuda en los scripts:

```
"hola" # palabra de la lengua española usada como saludo

## [1] "hola"
```

También podemos realizar operaciones matemáticas:

```
1+2*5

## [1] 11
```

Si escribimos la operación entre comillas, estamos indicando a R que se trata de una secuencia de caracteres. Por lo tanto, no realiza la operación y procede en forma similar que al escribir `"hola"`:

```
"1+2*5"

## [1] "1+2*5"
```

Como vemos, las sentencias que ejecutamos aparecen en la consola junto con el resultado seguido del `>`, que indica que la sentencia fue ejecutada y que R está listo para un nuevo comando. Si hay un error en la sentencia, R lo indica en la salida con una leyenda que dice cuál fue el problema.

```
"a" + 3

## Error in "a" + 3 : argumento no-numérico para operador binario

a + 3

## Error: objeto 'a' no encontrado
```

En el primer caso, R nos indica que no es válido realizar una operación matemática con un carácter y en el segundo que `a` aún no fue definido. R es estricto en su sintaxis y los errores de código son cotidianos para sus usuarios. Por lo tanto, quienes comienzan a programar no deben frustrarse cuando el código no corre ya que es algo con lo que convivimos al usar R.

1. 2. 3. Directorio de trabajo

Para simplificar la sintaxis posterior es necesario definir el director de trabajo desde donde se leen los datos y en donde se guardan los scripts y gráficos que realizamos. La función `getwd()` nos muestra el directorio de trabajo actual y con la función `setwd()` podemos definir aquel en donde queremos trabajar, por ejemplo, con el comando:

```
setwd("D:\\Mis documentos")
```

indicamos que el directorio de trabajo está en la carpeta «Mis documentos» del disco «D:\\». En Windows podemos usar una barra derecha en lugar de las dos barras invertidas:

```
setwd("D:/Mis documentos")
```

En RStudio podemos indicar el directorio de trabajo desde la solapa «Files» del panel de utilidades (derecha inferior):

1. Ubicamos el cursor sobre los tres puntos «…» (Go to directory) y al dar clic se abre el explorador desde el cual se ubica la carpeta que deseamos utilizar como «Working directory» (directorio de trabajo).
2. Luego, «More».
3. Finalmente, «Set as working directory».

1. 2. 4. Carga de datos

Podemos ingresar los datos en forma directa concatenándolos mediante `c()`:

```
dat <- c(2,4,5,5,11) # creamos el objeto dat
dat

## [1]  2  4  5  5 11
```

Además, con `seq()` y `rep()` podemos generar secuencias de elementos con determinados patrones. Por ejemplo:

```
seq(from=0, to=24, by=3) # de 0 a 24 cada 3

## [1]  0  3  6  9 12 15 18 21 24

rep(c(1,2,5), times=2) # repetir el vector dos veces

## [1] 1 2 5 1 2 5

rep(c(1,2,5), each=2) # repetir dos veces cada elemento del vector

## [1] 1 1 2 2 5 5
```

Si los datos se encuentran en un archivo almacenado en nuestro ordenador, podemos leerlos usando `read.table()`. Por ejemplo, si queremos cargar el archivo *lotes.txt*:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
dat.rend
```

Al ejecutar la línea de `dat.rend`, que es el objeto que creamos, se imprime su resultado mostrando los datos en un formato poco ameno. Si solo queremos ver las primeras filas:

```
head(dat.rend)

##    lote rendimiento borde tamanio     N dsiembra   fsiembra    cprevio region
## 1     1        2881 30.03      67  6.71    50000 16/10/2018 gram_anual     ra
## 2     2        2339 15.33      60  6.71    45000 20/10/2018 gram_anual     ra
## 3     3        2192 34.83      62  6.71    43000 19/10/2018 gram_anual     ra
## 4     4        2584 41.22      67  6.71    45000 26/10/2018 gram_anual     ra
## 5     5        2312 54.54      66  4.40    57000 17/10/2018 gram_anual     rb
## 6     6        3103 63.28      15 41.05    57000 23/10/2018       maiz     ra
```

La función `View()` permite visualizar la tabla de datos con mejor legibilidad:

```
View(dat.rend)
```

Para que R pueda leer el archivo *lotes.txt*, este debe estar en el directorio de trabajo. Existen muchas otras posibilidades para la carga de datos, y estas dependen del formato del archivo donde están almacenados (texto delimitado por comas, hoja de cálculo, formatos de otros programas como SPSS, SAS o Stata) y de donde se encuentra (ordenador, nube, sitio de internet, etcétera). En RStudio existe un atajo para la importación de datos desde la opción «ImportDataset» de la pestaña «Environment» del panel del entorno (derecha superior).

1. 2. 5. Objetos

R es un lenguaje orientado a objetos. Esto significa que los objetos creados se almacenan en la memoria activa del ordenador (no se usan archivos temporales) con un nombre específico, aspecto que otorga a R gran flexibilidad y simpleza en la programación. Un objeto encapsula información y puede contener un valor, un carácter, conjuntos de datos con diferentes estructuras como vectores o matrices, e incluso un objeto puede ser una función o un resultado. Por ejemplo, `dat` es un objeto que consta de cinco elementos. Podemos revisar los objetos que hemos creado durante la sesión ejecutando:

```
objects() # objetos creados

## [1] "dat"      "dat.rend"
```

Como muestra el resultado, hasta el momento solo hemos creado dos objetos. Tengamos presente que `objects()` informa solamente los objetos que fueron creados durante la sesión. En el paquete base hay más de mil objetos almacenados que se cargan cada vez que abrimos R. Los objetos se crean mediante el operador «asignar», el cual se denota por `<-` (el signo menor seguido del signo menos), aunque también se puede utilizar el signo `=` (igual), y sus nombres pueden contener letras (R discrimina entre mayúsculas y minúsculas), números, puntos o iones, pero no pueden comenzar con un número. Si ejecutamos un comando sin asignarlo a un objeto, entonces R imprime su resultado y la sentencia se pierde. Por ejemplo al ejecutar el siguiente código imprime los cinco números:

```
c(1,3,4,8,10)

## [1]  1  3  4  8 10
```

Si los asignamos a un objeto, estos números son guardados en la memoria de R. Algo importante a tener en cuenta es que podemos reescribir objetos, en cuyo caso aquel que se creó originalmente desaparece de la memoria activa:

```
dat <- c(1,3,4,8,10)  # re-escribimos el objeto dat
```

Para imprimirlo (visualizar su salida) debemos ejecutarlo:

```
dat # lo imprimimos

## [1]  1  3  4  8 10
```

Si deseamos imprimir el resultado del objeto al momento que lo creamos, escribimos el código entre paréntesis:

```
(dat <- c(1,3,4,8,10)) # para crear objetos e imprimirlos a la vez

## [1]  1  3  4  8 10
```

En R todo objeto tiene una clase. Esta determina los atributos del objeto y la manera que trabaja dentro de R. Por ejemplo, `dat` es una secuencia de números que resulta en un objeto de clase numérica. El código con el que creamos `dat.rend` generó un `data.frame`, una clase de objeto de mayor complejidad que la anterior. La clase de un objeto puede consultarse usando `class()`:

```
class(dat)

## [1] "numeric"

class(dat.rend)

## [1] "data.frame"
```

1. 2. 6. Tipos y estructuras de datos

La anterior salida es útil para clarificar dos aspectos de los datos: de qué tipo son y cómo se estructuran. Lo primero hace referencia a las características de los elementos que contienen los conjuntos de datos. Entre los tipos de datos encontramos números, enteros (`integer`), reales (`numeric`) o complejos (`complex`), y cadenas de caracteres (`character`). También existen datos con valores lógicos (`logical`) y otros tipos especiales como los valores ausentes (`NA`) o nulos (`NULL`). La estructura se relaciona con la organización de los datos y en R existen cuatro estructuras básicas: los vectores, las listas, las matrices y los arreglos. Los vectores (`vector`) o secuencias de un mismo tipo de dato conforman el corazón de las estructuras de datos en R, incluso un número individual es considerado un vector de un solo elemento (el resto de las tres estructuras son en sí mismas vectores con atributos adicionales). Las listas (`list`), que también son vectores, permiten diferentes tipos de datos para lo cual su estructura permite almacenar subvectores con los elementos del mismo tipo. Como tercera estructura están las matrices (`matrix`), que contienen datos de un mismo tipo organizados en forma rectangular (filas y columnas). Y finalmente, están los arreglos (`array`) que amplían la dimensión de las matrices y, como las listas, permiten diferentes tipos de datos. También existen estructuras de datos que se construyen a partir de estructuras básicas, por ejemplo `data.frame` o `table`, y que son útiles a diferentes propósitos.

El objeto `dat` es un vector y las clases de los vectores, en este caso `numeric`, se asocian al tipo de dato que contienen. Para verificar que `dat` es un vector:

```
is.vector(dat) # chequeamos que dat tiene estructura de vector

## [1] TRUE
```

Los objetos de clase `data.frame` tienen una estructura interna de lista y son convenientes para trabajar con tablas de datos. Si queremos explorar la estructura de un objeto de esta clase y los tipos de datos que contiene, podemos usar la función `str()`:

```
str(dat.rend) # tipo de datos que contiene el objeto dat.rend

## 'data.frame':    105 obs. of  9 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 ...
##  $ borde      : num  30 15.3 34.8 41.2 54.5 ...
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 ...
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 ...
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 ...
##  $ fsiembra   : chr  "16/10/2018" "20/10/2018" "19/10/2018" "26/10/2018" ...
##  $ cprevio    : chr  "gram_anual" "gram_anual" "gram_anual" "gram_anual" ...
##  $ region     : chr  "ra" "ra" "ra" "ra" ...
```

Lo que informa la salida anterior es que `dat.rend` es un `data.frame` con nueve (sub)vectores de 105 elementos (filas) cada uno. Tres de los vectores corresponden a cadenas de caracteres, tres a números enteros (`integer`) y tres tienen números reales (`numeric`). Verificamos las propiedades `dat.rend`:

```
is.vector(dat.rend) # dat.rend no tiene estructura de vector

## [1] FALSE

is.data.frame(dat.rend) # dat.rend es un data frame

## [1] TRUE

is.list(dat.rend) # es decir, una lista

## [1] TRUE

is.vector(dat.rend$region) # conteniendo vectores

## [1] TRUE

is.character(dat.rend$region) # de diferentes tipos

## [1] TRUE

is.numeric(dat.rend$borde)

## [1] TRUE
```

Una pregunta que surge es por qué los datos del vector `dat` son de tipo numérico, mientras que los de `dat.rend$rendimiento` son de tipo entero, si ambos contienen números sin decimales. La razón es que, al ejecutar un número, R asumirá que es de tipo `numeric` a menos que se lo indiquemos. Para indicar explícitamente que un número entero es de tipo `integer`, debe estar seguido de la letra `L`:

```
class(2) # dato de tipo numérico

## [1] "numeric"

class(2L) # dato de tipo entero

## [1] "integer"

2L # el resultado de 2L es 2 como entero

## [1] 2
```

Sin embargo, lo que comúnmente hacemos es una coerción de datos mediante `as.`:

```
dat <- as.integer(dat) # coerción de "numeric" a "integer" usando as.
class(dat)

## [1] "integer"
```

Ahora bien, al importar datos mediante `read.table()`, los vectores que contengan solo números enteros son, por defecto, convertidos a `integer` (lo mismo ocurre si creamos vectores con el operador `:`). No obstante, para la mayoría de las aplicaciones es irrelevante si un conjunto de números enteros está codificado como datos de tipo numérico. La información respecto a la región a la que pertenece el lote, el cultivo previo y la fecha de siembra fue guardada como cadena de caracteres. Estas son variables categóricas y para su análisis estadístico en general conviene realizar una coerción a datos de tipo `factor`. Para esto podemos proceder como antes:

```
dat.rend$region <- as.factor(dat.rend$region) # coerción de "chr" a "factor"
class(dat.rend$region)

## [1] "factor"
```

En el caso de la fecha de siembra, si bien podemos tratarla como una categoría, se trata de una variable asociada al tiempo y, como tal, de naturaleza cuantitativa (entre dos fechas transcurre una determinada cantidad de días). A pesar de que las fechas estén almacenadas en el formato correcto, si no lo indicamos explícitamente, R las leerá como cadena de caracteres. La forma de indicarlo es mediante `as.Date()`:

```
dat.rend$fsiembra <- as.Date(dat.rend$fsiembra, format="%d/%m/%Y") # formato fecha
str(dat.rend$fsiembra)

##  Date[1:105], format: "2018-10-16" "2018-10-20" "2018-10-19" "2018-10-26" "2018-10-17
" ...
```

Con `format="%d/%m/%Y"` indicamos que las fechas están almacenadas como dd/mm/yyyy (día/mes/año) y no en el formato que R lee por defecto, que es yyyy-mm-dd (año-mes-día). Si queremos asegurarnos que `read.table()` importe los datos cualitativos como factores, ejecutamos:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",",
                       stringsAsFactors = TRUE)
str(dat.rend)

## 'data.frame':    105 obs. of  9 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 ...
##  $ borde      : num  30 15.3 34.8 41.2 54.5 ...
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 ...
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 ...
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 ...
##  $ fsiembra   : Factor w/ 28 levels "10/10/2018","11/10/2018",..: 7 12 10 16 8 13 14
12 21 2 ...
##  $ cprevio    : Factor w/ 7 levels "gram_anual","maiz",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ region     : Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
```

1. 2. 7. Funciones y operaciones

R tiene funciones predefinidas que, como vimos, son objetos. Por ejemplo, `c()` y `read.table()` son funciones (ver sección 1. 3 para un resumen de algunas comúnmente aplicadas al manejo de datos). Las funciones son seguidas de paréntesis, dentro de los cuales se definen sus argumentos, si los tuviera. Así, al usar la función `read.table()` cuando creamos el objeto `dat.rend` desde los argumentos `header=TRUE` y `dec=","` le indicamos que el archivo a leer (`"lotes.txt"`) tiene encabezado (es decir, que la primera fila corresponde a los nombres de las variables y no es un dato) y que la puntuación decimal es la coma, respectivamente.

Algunas funciones de R son genéricas ya que aplican a diferentes clases de objetos. El resultado de estas funciones depende de la clase del objeto sobre la cual la aplicamos y como ejemplos típicos tenemos `summary()`, `str()` o `plot()`. Por ejemplo, cuando ejecutamos `summary()` a un vector de tipo numérico, la función devuelve medidas descriptivas de la distribución empírica de los datos como la media, la mediana y los percentiles. Cuando se aplica sobre un vector de datos categóricos (cualitativos), en cambio, el resultado es una tabla de frecuencias. Además de funciones, sobre los objetos podemos realizar operaciones. Primero ejecutamos operaciones sobre números (R como calculadora):

```
4 + 8

## [1] 12

4 + 8*2

## [1] 20

(4 + 8)*2

## [1] 24
```

Se pueden combinar funciones con operaciones:

```r
log(4 + 8) # logarítmo natural
## [1] 2.484907
log10(4 + 8) # logarítmo en base 10
## [1] 1.079181
log((4 + 8)*2)
## [1] 3.178054
log(4 + 8) + log(2)
## [1] 3.178054
```

Ahora operamos sobre vectores:

```r
dat
## [1]  1  3  4  8 10
dat + 10
## [1] 11 13 14 18 20
```

Esta operación implicó que sumamos 10 a cada elemento del vector `dat` (se debe notar que no hemos creado ningún objeto nuevo). En cambio, al ejecutar la sentencia siguiente obtenemos como resultado un único valor:

```r
sum(dat) + 10
## [1] 36
```

En este caso, 10 es sumado a 26, el resultado de la sumatoria de los elementos de `dat`:

```r
sum(dat) # sumatoria de los elementos de dat
## [1] 26
```

Si lo que sumamos son vectores, R realizará sumas entre elementos según el orden en que aparecen. Por ejemplo:

```r
dat + c(5,5,10,20,30) # suma de dat con un vector de 5 elementos
## [1]  6  8 14 28 40
```

En efecto, en el primer caso, al sumar el valor 10 a un vector de cinco elementos, lo que hizo implícitamente R es una suma entre vectores de cinco elementos, `dat`, por un lado, y una secuencia repitiendo cinco veces 10, por el otro:

```
dat + c(10,10,10,10,10) # similar a dat + 10

## [1] 11 13 14 18 20

dat + 10

## [1] 11 13 14 18 20
```

Así como `+` y `*`, R contiene operadores aritméticos, lógicos y relacionales, y de la misma manera que `sum()` y `log()`, existen múltiples funciones para realizar operaciones matemáticas y estadísticas (ver sección 1. 3). Si por ejemplo queremos calcular la media aritmética de un conjunto de datos como `dat`, debemos sumar todos sus elementos y dividir por la cantidad de elementos, en este caso 5:

```
(1+3+4+8+10)/5 # media aritmética de los elementos de dat

## [1] 5.2
```

Para no tener que ingresar los valores manualmente, podemos hacer uso de las funciones `sum()` para la sumatoria y `length()` para obtener la cantidad de elementos:

```
sum(dat)/length(dat) # media aritmética obtenida usando las funciones sum() y length()

## [1] 5.2
```

o directamente usar la función `mean()`:

```
mean(dat) # media aritmética obtenida usando la función mean()

## [1] 5.2
```

Finalmente, podemos escribir una función nosotros mismos para calcular la media de cualquier vector. Veamos:

```
# Ejemplo de una función para calcular la media aritmética
media.aritmetica <- function(x){
    suma <- sum(x) # sumatoria
    n <- length(x) # cantidad de elementos
    return(suma/n)
}

media.aritmetica(dat) # media aritmética de dat obtenida usando la función creada

## [1] 5.2
```

R nos permite crear funciones desde `function()`, con la cual creamos una función que tiene un único argumento (`x`). Al ingresar un vector numérico (`dat` en este caso), internamente genera un objeto con la suma de sus elementos (`suma <- sum(x)`), establece su tamaño (`n <- length(x)`) y devuelve la división de ambas cantidades (`return(suma/n)`), es decir, el promedio. Cuando la función incluye múltiples expresiones, como en este caso (la sumatoria, la cantidad de datos, y la división), necesitamos agruparlas entre llaves (`{ }`). Si simplificamos nuestra función usando un solo paso, no es necesario el uso de las llaves. Veamos:

```
media.aritmetica2 <- function(x) sum(x)/length(x)
media.aritmetica2(dat)

## [1] 5.2
```

A diferencia de `sum()`, `length()` o `mean()`, las funciones creadas por los usuarios no son parte de la librería base de R, de modo que es necesario ejecutar su código para que puedan usarse dentro de la sesión. Algo similar ocurre con las funciones incluidas en los paquetes (ver subsección 1. 2. 11), las cuales, si bien están predefinidas, la librería a la que pertenecen debe ser cargada en cada sesión para que estas funciones estén disponibles para ser usadas. En resumen, en R hay tres tipos de funciones:

- las incluidas dentro de la librería base: solo deben invocarse para ejecutarlas,
- las incluidas dentro de paquetes: para usarlas es necesario llamar a (cargar) la librería a la que pertenecen,
- las creadas por el usuario (no predefinidas): antes de usarse debe ejecutarse su código.

1. 2. 8. Operaciones con matrices

Las matrices cumplen un rol esencial en los modelos estadísticos y R dispone de gran flexibilidad para operar con ellas. Para generar matrices usamos la función `matrix()`, a la cual debemos pasarle los elementos de la matriz y definir el número de filas y columnas que deseamos:

```
dat.mat <- c(3, 3, 2, 2, 6,
             5, 4, 6, 9, 1,
             5, 3, 9, 9, 9,
             1, 1, 1, 2, 1,
             7, 8, 9, 5, 2) # entramos 25 números
dat.mat

##  [1] 3 3 2 2 6 5 4 6 9 1 5 3 9 9 9 1 1 1 2 1 7 8 9 5 2

M1 <- matrix(dat.mat, ncol=5, byrow=TRUE) # matriz de 5 x 5
M1

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    3    3    2    2    6
## [2,]    5    4    6    9    1
## [3,]    5    3    9    9    9
## [4,]    1    1    1    2    1
## [5,]    7    8    9    5    2
```

Con el argumento `byrow` indicamos si los números se van introduciendo por filas (`byrow=TRUE`) o por columnas (`byrow=FALSE`). En ocasiones es útil identificar las filas y columnas:

```
dimnames(M1) <- list(c("f1", "f2", "f3", "f4", "f5"),
                     c("c1", "c2", "c3", "c4", "c5"))
M1

##    c1 c2 c3 c4 c5
## f1  3  3  2  2  6
## f2  5  4  6  9  1
## f3  5  3  9  9  9
## f4  1  1  1  2  1
## f5  7  8  9  5  2
```

A continuación, exploramos algunas operaciones algebraicas:

```
2 * M1 # multiplicación de cada elemento de M1 por 2

##    c1 c2 c3 c4 c5
## f1  6  6  4  4 12
## f2 10  8 12 18  2
## f3 10  6 18 18 18
## f4  2  2  2  4  2
## f5 14 16 18 10  4
```

Lo que hemos hecho es simplemente una multiplicación de cada elemento de `M1` por 2, de manera similar a lo visto antes con la suma de un número y un vector. Lo comprobamos repitiendo 25 veces el valor 2 y multiplicando la secuencia con `M1`:

```
rep(2, 25) * M1

##    c1 c2 c3 c4 c5
## f1  6  6  4  4 12
## f2 10  8 12 18  2
## f3 10  6 18 18 18
## f4  2  2  2  4  2
## f5 14 16 18 10  4
```

Con el operador `*` hemos realizado una multiplicación elemento a elemento y, como observamos, el resultado es el mismo. La misma lógica aplica a la siguiente operación:

```
dat * M1

##    c1 c2 c3 c4 c5
## f1  3  3  2  2  6
## f2 15 12 18 27  3
## f3 20 12 36 36 36
## f4  8  8  8 16  8
## f5 70 80 90 50 20
```

En este caso, `dat` es un vector de cinco elementos y la multiplicación elemento a elemento ocurre entre `dat` y cada columna de `M1`. Lo comprobamos realizando la multiplicación columna por columna:

```
dat * M1[, 1]

## f1 f2 f3 f4 f5
##  3 15 20  8 70

dat * M1[, 2]

## f1 f2 f3 f4 f5
##  3 12 12  8 80

dat * M1[, 3]

## f1 f2 f3 f4 f5
##  2 18 36  8 90

dat * M1[, 4]

## f1 f2 f3 f4 f5
##  2 27 36 16 50

dat * M1[, 5]

## f1 f2 f3 f4 f5
##  6  3 36  8 20
```

Las salidas de cada operación coinciden con las columnas de la matriz que resulta de `dat * M1` (se debe advertir que para extraer las columnas de `M1` hemos utilizado `[]`; exploraremos la gran utilidad de esto en la sección «Indexación y filtros»). Todas las multiplicaciones que hemos realizado hasta aquí son escalares (elemento a elemento) y raramente se utilizan en el contexto de los modelos estadísticos. Para realizar una multiplicación de matrices debemos ejecutar `%*%`:

```
dat %*% M1

##       c1  c2  c3  c4 c5
## [1,] 116 115 154 131 73
```

Ahora sí, el resultado de multiplicar una matriz de 5 (filas) x 5 (columnas):

```
nrow(M1) # nº filas de M1

## [1] 5

ncol(M1) # nº columnas de M1

## [1] 5

dim(M1) # filas x columnas

## [1] 5 5
```

con un vector de cinco elementos, que en esencia es una matriz de 5 x 1:

```
dim(as.matrix(dat)) # dimensión de dat como matriz

## [1] 5 1
```

es una matriz de 1 x 5.

```
dim(dat %*% M1) # dimensión de la matriz resultante

## [1] 1 5
```

Lógicamente, si multiplicamos `M1` (5 x 5) por una matriz 4 x 1 obtenemos un error ya que la operación no es posible:

```
r4 <- seq(1:4) # vector de 4 elementos
dim(as.matrix(r4)) # o matriz 4 x 1

## [1] 4 1

M1 %*% r4

## Error in M1 %*% r4 : non-conformable arguments
```

Veamos algunas funciones de R que son propias de las matrices:

```
cov(M1, y=M1, use="all.obs") # matriz de varianzas y covarianzas

##     c1    c2    c3    c4    c5
## c1 5.2  5.30  7.90  4.90  0.80
## c2 5.3  6.70  6.85  2.10 -1.55
## c3 7.9  6.85 14.30 10.05  3.85
## c4 4.9  2.10 10.05 12.30  2.85
## c5 0.8 -1.55  3.85  2.85 12.70

cor(M1, y=M1, use="all.obs") # matriz de correlación

##            c1         c2        c3        c4          c5
## c1 1.00000000  0.8979182 0.9161306 0.6126915  0.09844337
## c2 0.89791825  1.0000000 0.6998181 0.2313286 -0.16803217
## c3 0.91613062  0.6998181 1.0000000 0.7577841  0.28568724
## c4 0.61269150  0.2313286 0.7577841 1.0000000  0.22802919
## c5 0.09844337 -0.1680322 0.2856872 0.2280292  1.00000000

diag(M1) # diagonal

## [1] 3 4 9 2 2

diag(diag(M1)) # creamos una matriz diagonal con la diagonal de M1

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    3    0    0    0    0
## [2,]    0    4    0    0    0
## [3,]    0    0    9    0    0
## [4,]    0    0    0    2    0
## [5,]    0    0    0    0    2

det(M1) # determinante

## [1] 396
```

```
t(M1) # transpuesta

##    f1 f2 f3 f4 f5
## c1  3  5  5  1  7
## c2  3  4  3  1  8
## c3  2  6  9  1  9
## c4  2  9  9  2  5
## c5  6  1  9  1  2

scale(M1, center=TRUE, scale=TRUE) # estandarización (por columna)

##            c1          c2         c3         c4         c5
## f1 -0.5262348 -0.30906696 -0.8991060 -0.9694521  0.6173349
## f2  0.3508232  0.07726674  0.1586658  1.0264787 -0.7856989
## f3  0.3508232 -0.30906696  0.9519946  1.0264787  1.4591552
## f4 -1.4032928 -1.08173437 -1.1635489 -0.9694521 -0.7856989
## f5  1.2278812  1.62260156  0.9519946 -0.1140532 -0.5050922
## attr(,"scaled:center")
##  c1  c2  c3  c4  c5
## 4.2 3.8 5.4 5.4 3.8
## attr(,"scaled:scale")
##       c1       c2       c3       c4       c5
## 2.280351 2.588436 3.781534 3.507136 3.563706

eigen(M1) # valores propios

## eigen() decomposition
## $values
## [1] 22.2893009+0.000000i -2.7109273+1.713004i -2.7109273-1.713004i
## [4]  2.4180790+0.000000i  0.7144747+0.000000i
##
## $vectors
##                [,1]                    [,2]                    [,3]
## [1,] -0.3133323+0i  0.62674884+0.00000000i  0.62674884+0.00000000i
## [2,] -0.3773366+0i -0.59239752-0.04755900i -0.59239752+0.04755900i
## [3,] -0.6527185+0i  0.16270246-0.16315089i  0.16270246+0.16315089i
## [4,] -0.0942883+0i  0.03480401+0.00302196i  0.03480401-0.00302196i
## [5,] -0.5696562+0i -0.36618957+0.25609307i -0.36618957-0.25609307i
##                 [,4]            [,5]
## [1,]  0.541307040+0i  0.85078829+0i
## [2,]  0.293350480+0i -0.45820728+0i
## [3,] -0.771267913+0i -0.13759008+0i
## [4,]  0.161434684+0i -0.21621154+0i
## [5,]  0.004102851+0i  0.02295448+0i

solve(M1) # inversión

##             f1          f2            f3         f4          f5
## c1  0.87121212  1.06818182 -1.666667e-01 -4.2651515 -0.26515152
## c2 -0.37626263 -0.64393939 -5.555556e-02  2.9116162  0.24494949
## c3 -0.28030303 -0.20454545  1.666667e-01  0.1287879  0.12878788
## c4 -0.13636364 -0.04545455  3.819167e-17  0.9545455 -0.04545455
## c5  0.05808081 -0.12878788  5.555556e-02  0.3156566 -0.01767677
```

1. 2. 9. Datos faltantes

«NA» es la forma en que ingresamos los datos faltantes en el archivo de datos para que R los lea como tales (NA proviene de la abreviación de *not available*). Los NA son datos de tipo lógico y su tratamiento es importante ya que los resultados que obtenemos muchas veces dependen de ello. A continuación, creamos un vector con datos faltantes para explorar algunas funciones útiles:

```
dat2 <- c(4, 3, 5, 7, NA, 8, 8, 12, NA, 2, 3) # creamos un vector con NAs
anyNA(dat2) # verificamos si hay algún NA
## [1] TRUE
which(is.na(dat2))  # extraemos la posición de los NA
## [1] 5 9
dat2[c(5,9)] # ubicamos los NAs por indexación
## [1] NA NA
```

En general, cuando realizamos una operación o aplicamos una función sobre datos que contienen `NA`, el resultado es `NA`:

```
mean(dat2) # promedio calculado con NA
## [1] NA
```

Para evitarlo, debemos omitir, excluir o remover los datos faltantes, lo cual podemos hacerlo aplicando funciones sobre los datos:

```
mean(na.omit(dat2)) # omitimos los NA con una función
## [1] 5.777778
```

o también indicarlo desde los argumentos de las funciones que usamos para la operación:

```
mean(dat2, na.rm=TRUE) # omitimos los NA con un argumento
## [1] 5.777778
```

1. 2. 10. Indexación y filtros

En R, la posición de los elementos de los vectores, así como la de las tablas de datos, matrices, listas y arreglos, están indexados y para acceder a ellos usamos corchetes. Por ejemplo, para extraer el segundo elemento del vector `dat` ejecutamos:

```
dat
## [1]  1  3  4  8 10
dat[2] # segundo elemento de dat
## [1] 3
```

En el caso de las estructuras de datos que presentan dos dimensiones, tales como las matrices y las tablas de datos

```
dim(dat.rend) # 105 filas x 9 columnas
## [1] 105   9
```

necesitamos indicar la fila y la columna. Por ejemplo, para extraer la densidad de siembra (sexta columna de `dat.rend`) del segundo lote de la muestra (fila 2), ejecutamos:

```
dat.rend[2,6] # fila 2 columna 6 del data.frame dat.rend

## [1] 45000
```

Esto es lo mismo que extraer la posición 2 de `dat.rend$dsiembra`; usando la función `names()` accedemos a los nombres de los vectores de `dat.rend`:

```
names(dat.rend)

## [1] "lote"        "rendimiento" "borde"       "tamanio"     "N"
## [6] "dsiembra"    "fsiembra"    "cprevio"     "region"

dat.rend$dsiembra[2] # segundo elemento del vector "dsiembra"

## [1] 45000
```

El operador `$` se asocia a las listas y lo usamos para extraer la columna que nos interesa. Si lo que queremos es toda la información del segundo lote (fila) entonces debemos dejar vacío el espacio que corresponde a las columnas, es decir, luego de la coma dentro de los corchetes:

```
dat.rend[2, ] # segunda fila de dat.rend

##   lote rendimiento borde tamanio    N dsiembra   fsiembra    cprevio region
## 2    2        2339 15.33      60 6.71    45000 20/10/2018 gram_anual     ra
```

La indexación es sumamente útil para operar en R. Por ejemplo, podemos filtrar datos de acuerdo a ciertas condiciones:

```
dat.rend$rendimiento[dat.rend$region == "ra"] # rendimiento de los lotes en la región a

##  [1] 2881 2339 2192 2584 3103 2273 3150 1972 3496 3449 3650 2353 3500 2745 2720
## [16] 1770 1985 2417 2512 1925 3567 3468 3588

dat.rend$rendimiento[dat.rend$rendimiento > 3000] # filtramos los lotes con rendimientos
mayores a 3000 kg/ha

##  [1] 3103 3150 3496 3027 3050 3477 3592 3449 3650 3500 3067 3381 3100 3567 3468
## [16] 3588
```

Con el primer código extraemos rendimientos de los lotes de la región «a», y con el segundo los de aquellos lotes que superaron los 3000 kg/ha. Dos funciones útiles para filtrar datos son `which()` y `subset()`. Con la primera obtenemos la posición de los elementos que cumplen cierta condición: Por ejemplo, al ejecutar:

```
which(dat.rend$rendimiento > 3000)

##  [1]   6   8  45  50  54  55  56  71  72  74  85  86  90 100 101 102
```

se obtiene la posición en la que están los elementos del vector que cumplen con la condición de ser mayores a 3000. En cambio, `subset()` muestra los valores, tal como cuando filtramos por indexación:

```
subset(dat.rend$rendimiento, dat.rend$rendimiento > 3000)

##  [1] 3103 3150 3496 3027 3050 3477 3592 3449 3650 3500 3067 3381 3100 3567 3468
## [16] 3588
```

Si el filtro lo aplicamos sobre `dat.rend`, y dentro de los corchetes dejamos vacío el espacio luego de la coma, lo que obtenemos es la información completa de cada lote con rendimientos superiores a 3000 kg/ha:

```
dat.rend[dat.rend$rendimiento > 3000, ] # subconjunto de dat.rend  con el vector "rendim
iento" > 3000

##     lote rendimiento  borde tamanio     N dsiembra   fsiembra  cprevio region
## 6      6        3103  63.28    15.0 41.05    57000 23/10/2018     maiz     ra
## 8      8        3150  34.21    72.0 55.80    54000 20/10/2018     maiz     ra
## 45    45        3496  22.95    60.0 31.81    57000 11/10/2018 soja_1ra     ra
## 50    50        3027  57.71    75.0  3.85    59000 16/10/2018 soja_1ra     rb
## 54    54        3050 102.24     8.0  3.30    42400 15/10/2018 soja_1ra     rc
## 55    55        3477 102.24    39.0  4.40    58200 15/10/2018 soja_1ra     rc
## 56    56        3592 102.24    13.0  4.40    60600 15/10/2018 soja_1ra     rc
## 71    71        3449  22.95    62.0 31.81    57000 20/10/2018 soja_1ra     ra
## 72    72        3650  63.28    55.0 33.49    57000 23/10/2018 soja_1ra     ra
## 74    74        3500  32.96    18.0  7.20    60000 15/10/2018 soja_2da     ra
## 85    85        3067  79.99    35.0  3.30    54600 17/10/2018 soja_2da     rc
## 86    86        3381  79.99    27.0  5.50    60600 17/10/2018 soja_2da     rc
## 90    90        3100  22.65    85.0 64.00    60000 23/10/2018 soja_2da     rd
## 100  100        3567  21.18    37.1 78.00    60000 15/10/2018   verdeo     ra
## 101  101        3468  21.18    17.4 78.00    60000 15/10/2018   verdeo     ra
## 102  102        3588  21.18    18.4 78.00    60000 15/10/2018   verdeo     ra
```

En el espacio luego de la coma podemos especificar qué información asociada a estos lotes queremos. Por ejemplo, si solo nos interesa saber a qué regiones pertenecen los lotes que rindieron más de 3000, debemos indicar la columna 9:

```
dat.rend[dat.rend$rendimiento > 3000, 9] # region de los lotes con rendimientos mayores
a 3000

##  [1] ra ra ra rb rc rc rc ra ra ra rc rc rd ra ra ra
## Levels: ra rb rc rd
```

Si además quisiéramos saber la superficie de esos lotes, también especificamos la columna 4:

```
dat.rend[dat.rend$rendimiento > 3000, c(9,4)] # incluimos el área de estos lotes
```

```
##     region tamanio
## 6       ra    15.0
## 8       ra    72.0
## 45      ra    60.0
## 50      rb    75.0
## 54      rc     8.0
## 55      rc    39.0
## 56      rc    13.0
## 71      ra    62.0
## 72      ra    55.0
## 74      ra    18.0
## 85      rc    35.0
## 86      rc    27.0
## 90      rd    85.0
## 100     ra    37.1
## 101     ra    17.4
## 102     ra    18.4
```

Como antes, podemos obtener la misma información usando la función `subset()`:

```
subset(dat.rend, rendimiento > 3000)
```

```
##     lote rendimiento  borde tamanio     N dsiembra   fsiembra  cprevio region
## 6      6        3103  63.28    15.0 41.05    57000 23/10/2018     maiz     ra
## 8      8        3150  34.21    72.0 55.80    54000 20/10/2018     maiz     ra
## 45    45        3496  22.95    60.0 31.81    57000 11/10/2018 soja_1ra     ra
## 50    50        3027  57.71    75.0  3.85    59000 16/10/2018 soja_1ra     rb
## 54    54        3050 102.24     8.0  3.30    42400 15/10/2018 soja_1ra     rc
## 55    55        3477 102.24    39.0  4.40    58200 15/10/2018 soja_1ra     rc
## 56    56        3592 102.24    13.0  4.40    60600 15/10/2018 soja_1ra     rc
## 71    71        3449  22.95    62.0 31.81    57000 20/10/2018 soja_1ra     ra
## 72    72        3650  63.28    55.0 33.49    57000 23/10/2018 soja_1ra     ra
## 74    74        3500  32.96    18.0  7.20    60000 15/10/2018 soja_2da     ra
## 85    85        3067  79.99    35.0  3.30    54600 17/10/2018 soja_2da     rc
## 86    86        3381  79.99    27.0  5.50    60600 17/10/2018 soja_2da     rc
## 90    90        3100  22.65    85.0 64.00    60000 23/10/2018 soja_2da     rd
## 100  100        3567  21.18    37.1 78.00    60000 15/10/2018   verdeo     ra
## 101  101        3468  21.18    17.4 78.00    60000 15/10/2018   verdeo     ra
## 102  102        3588  21.18    18.4 78.00    60000 15/10/2018   verdeo     ra
```

```
subset(dat.rend[,9], dat.rend$rendimiento > 3000)
```

```
##  [1] ra ra ra rb rc rc rc ra ra ra rc rc rd ra ra ra
## Levels: ra rb rc rd
```

```
subset(dat.rend[,c(9,4)], dat.rend$rendimiento > 3000)
```

```
##     region tamanio
## 6       ra    15.0
## 8       ra    72.0
## 45      ra    60.0
## 50      rb    75.0
## 54      rc     8.0
## 55      rc    39.0
## 56      rc    13.0
## 71      ra    62.0
## 72      ra    55.0
## 74      ra    18.0
## 85      rc    35.0
## 86      rc    27.0
## 90      rd    85.0
## 100     ra    37.1
## 101     ra    17.4
## 102     ra    18.4
```

Muchas veces es útil utilizar criterios dicotómicos. Esto ocurre en variables cuantitativas que superan o no superan cierto umbral, o la pertenencia o no a cierta categoría en variables cualitativas de más de dos niveles. En estos casos, la función `ifelse()` permite crear una nueva variable que informa si se cumple el criterio. Por ejemplo, podemos establecer que 5000 semillas por hectárea es el umbral a partir del cual la densidad de siembra se considera alta:

```
dat.rend$dsiembra2 <- ifelse(dat.rend$dsiembra > 50000, "alta", "baja")
str(dat.rend)

## 'data.frame':    105 obs. of  10 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 ...
##  $ borde      : num  30 15.3 34.8 41.2 54.5 ...
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 ...
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 ...
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 ...
##  $ fsiembra   : Factor w/ 28 levels "10/10/2018","11/10/2018",..: 7 12 10 16 8 13 14
12 21 2 ...
##  $ cprevio    : Factor w/ 7 levels "gram_anual","maiz",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ region     : Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
##  $ dsiembra2  : chr  "baja" "baja" "baja" "baja" ...
```

Acabamos de agregar una nueva columna a `dat.rend`, la cual informa si el lote se sembró a baja o alta densidad de semillas. Lo que hicimos con `ifelse()` es indicar que si la densidad es mayor a 50000 (`dat.rend$densidad > 50000`), entonces queremos que se asigne la clase `"alta"`, y `"baja"` en caso contrario. La columna fue codificada como de cadena de caracteres, pero desde un punto de vista estadístico la variable es un factor de naturaleza ordinal. Para que así sea considerada debemos realizar una coerción de datos indicando que el nivel «baja» está por debajo del nivel «alta»:

```
dat.rend$dsiembra2 <- ordered(dat.rend$dsiembra2, levels=c("baja","alta"))
str(dat.rend$dsiembra2)

##  Ord.factor w/ 2 levels "baja"<"alta": 1 1 1 1 2 2 1 2 1 2 ...
```

Si luego deseamos removerla de la base de datos, indexamos la última columna (10) con el signo menos:

```
dat.rend <- dat.rend[, -10]
str(dat.rend)

## 'data.frame':    105 obs. of  9 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 ...
##  $ borde      : num  30 15.3 34.8 41.2 54.5 ...
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 ...
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 ...
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 ...
##  $ fsiembra   : Factor w/ 28 levels "10/10/2018","11/10/2018",..: 7 12 10 16 8 13 14
12 21 2 ...
##  $ cprevio    : Factor w/ 7 levels "gram_anual","maiz",..: 1 1 1 1 1 2 2 2 2 2 ...
##  $ region     : Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
```

1. 2. 11 Paquetes (librerías)

Los paquetes contienen conjuntos de funciones acompañados por la documentación que las describe. R se instala con un paquete base que se carga automáticamente al comenzar una sesión. Las capacidades del paquete base pueden ser extendidas mediante otros desarrollados para fines específicos. Por ejemplo, existen paquetes para facilitar la realización de gráficos, visualizar y modelar datos espaciales y series temporales, realizar minería de datos, vincular R a otros programas, y muchísimas aplicaciones más. Estos paquetes complementarios están alojados en el CRAN y son controlados por los desarrolladores de R. Para poder usar las funciones de un paquete, primero es necesario instalarlo (este paso se realiza una única vez) y luego cargarlo en cada sesión que lo necesitamos. La instalación de los paquetes la hacemos con la función `install.packages()`, y su carga con la función `library()`. Usando como ejemplo el paquete `car`:

```
install.packages(car) # instalación del paquete car
library(car) # carga del paquete car

## Loading required package: carData
```

Si luego que lo cargamos queremos removerlo de la memoria escribimos:

```
detach("package:car", unload=TRUE)
```

Una forma de acceder a las funciones de los paquetes sin antes cargarlo es mediante el doble operador `::`. Para esto debemos escribir el nombre del paquete, el operador y luego la función. Por ejemplo, para acceder a la función `fractions()`, que es parte del paquete `MASS`, y expresar números como fracciones, ejecutamos:

```
MASS::fractions(c(0.1, 0.11, 0.2, 1.75, 1.723))

## [1]      1/10     11/100       1/5       7/4 1723/1000
```

Lógicamente, el paquete debe estar previamente instalado.

1. 2. 12 Gráficos

R es un lenguaje que permite producir gráficos de una gran calidad adecuados para publicaciones. Siempre que abrimos R se cargan un conjunto de funciones gráficas base, las cuales están contenidas en el paquete `graphics`. Otros paquetes también cuentan con funciones gráficas específicas e incluso existen paquetes que están diseñados para la creación de gráficos, como por ejemplo `ggplot2`, que han desplazado a las funciones base. No obstante, estas son muy útiles y recurrimos a ellas a lo largo de todo el libro. Alternativamente, mostramos algunos ejemplos de gráficos creados con `ggplot2`.

Las funciones gráficas del paquete base se dividen en aquellas que generan el gráfico, denominadas de *alto nivel*, y aquellas que permiten agregar elementos a un gráfico existente, a las que se las llama de *bajo nivel*. Entre las funciones del primer grupo, una de las más utilizadas es `plot()`, con la cual podemos generar diferentes tipos de gráficos de acuerdo a los datos de entrada. Por ejemplo, permite realizar un gráfico de dispersión entre dos variables cuantitativas como el rendimiento y el tamaño del lote (figura 1. 2):

```
plot(x=dat.rend$tamanio, y=dat.rend$rendimiento) # figura 1.2
```

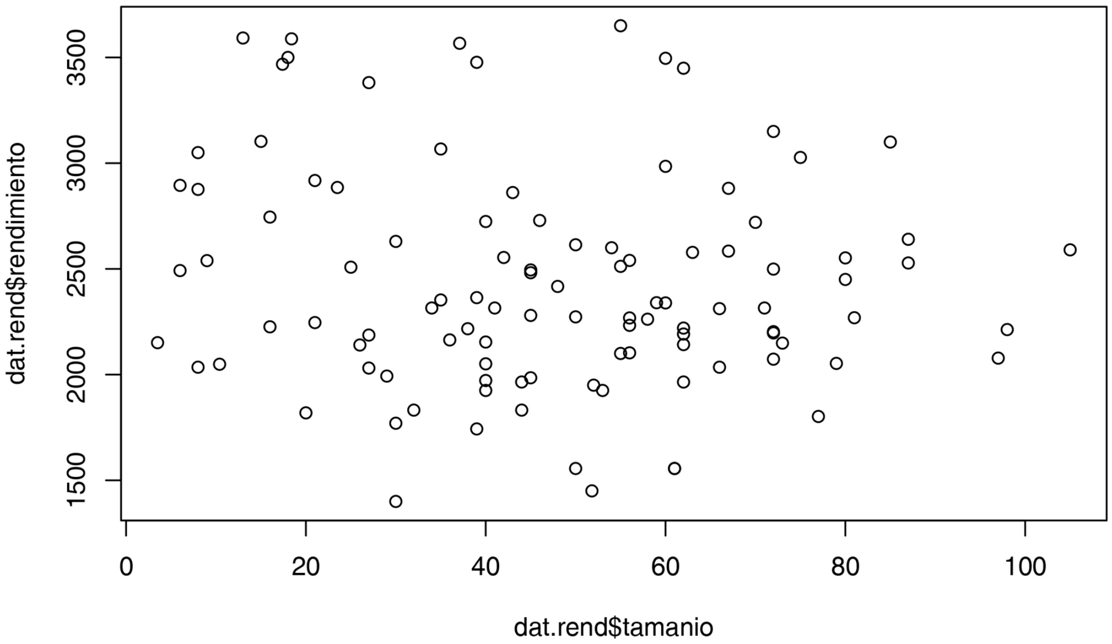

Figura 1. 2. Gráfico de dispersión obtenido al ejecutar la función `plot()` sobre dos variables cuantitativas

Al aplicar esta función sobre una variable categórica como la región se obtiene un gráfico de frecuencias (figura 1. 3):

```
plot(x=dat.rend$region) # figura 1.3
```

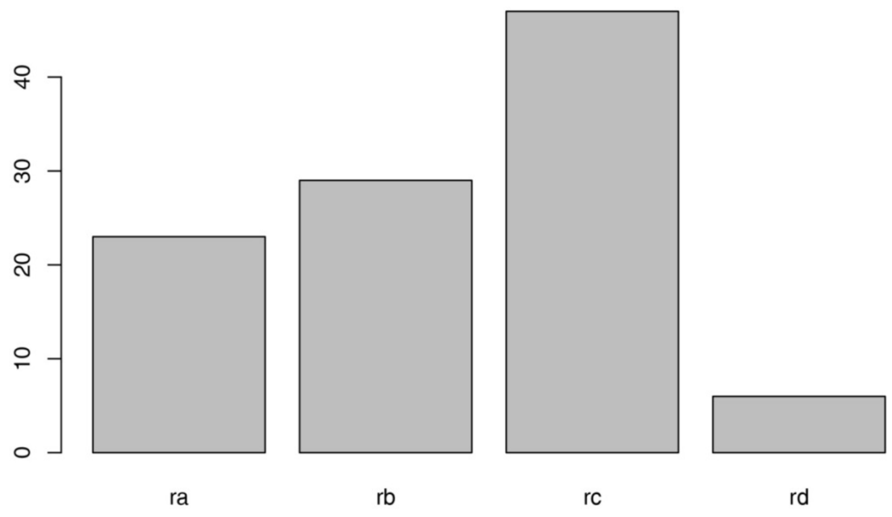

Figura 1. 3. Distribución de frecuencias obtenida al ejecutar la función `plot()` sobre una variable categórica

Vinculando una variable cuantitativa con una variable categórica se generan gráficos de caja y bigotes de la primera para cada nivel de la segunda (figura 1. 4):

```
plot(dat.rend$rendimiento ~ dat.rend$region) # figura 1.4
```

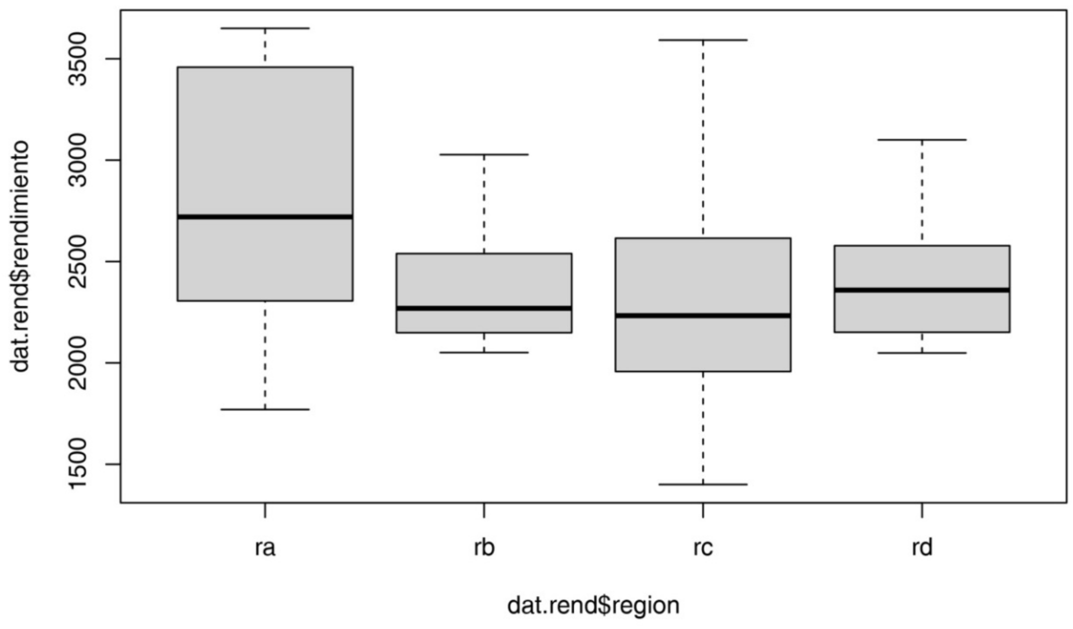

Figura 1. 4. Gráficos de caja y bigotes obtenidos al ejecutar la función `plot()` en una variable cuantitativa vinculada a (en función de) una categórica

La función `par()` permite configurar diferentes parámetros gráficos. Por ejemplo, se puede mostrar los tres gráficos en una misma salida indicándolo desde el argumento `mfrow` (figura 1. 5):

```
# figura 1.5
par(mfrow=c(1,3)) # 1 fila x 3 columnas
plot(x=dat.rend$tamanio, y=dat.rend$rendimiento)
plot(x=dat.rend$region)
plot(dat.rend$rendimiento ~ dat.rend$region)
```

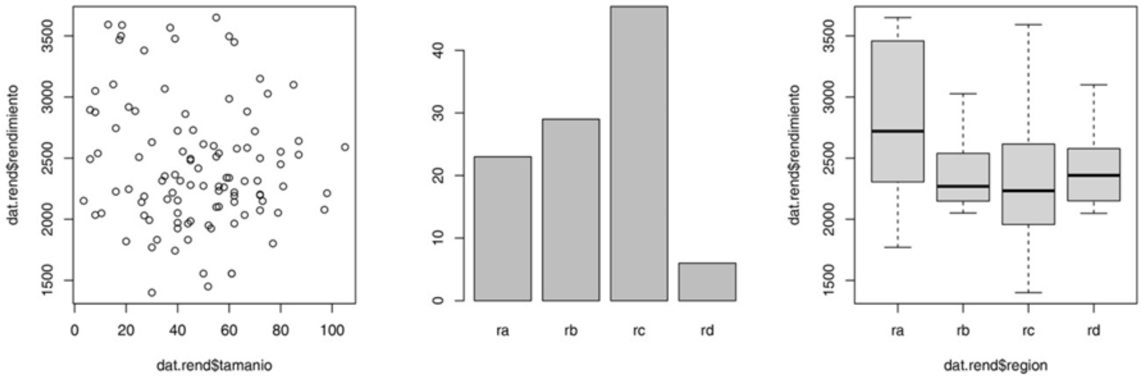

Figura 1. 5. Obtención de tres gráficos en una misma salida configurando el argumento `mfrow` de la función `par()`.

Algunos parámetros gráficos debemos definirlos desde los argumentos de la función grafica. Por ejemplo, es posible etiquetar los ejes y modificar sus rangos, e incluir un título en cada gráfico (figura 1. 6):

```
# figura 1.6
par(mfrow=c(3,1)) # 3 filas x 1 columna
plot(x=dat.rend$tamanio, y=dat.rend$rendimiento,
     xlim=c(0,100), ylim=c(500,4500),
     xlab="Tamaño del lote (ha)", ylab="Rendimiento (kg/ha)",
     pch=16, main="Grafico de dispersión")

abline(h=mean(dat.rend$rendimiento), v=mean(dat.rend$tamanio),
       lty=2, lwd=1, col="red") # agregamos una línea vertical y otra horizontal
text(x=46, y=675, labels="Tamaño medio", adj=1) # agregamos texto
text(x=103, y=2700, labels="Rend. medio", adj=1) # agregamos texto

plot(x=dat.rend$region, col=c("red", "green", "darkgrey", "yellow"),
     xlab="Región", ylab="Frecuencia absoluta", ylim=c(0,60),
     main="Distribución de frecuencias")
plot(dat.rend$rendimiento ~ dat.rend$region,
     col=c("red", "green", "darkgrey", "yellow"),
     xlab="Región", ylab="Rendimiento (kg/ha)",
     ylim=c(500,4500), main="Gráficos de caja y bigote")
```

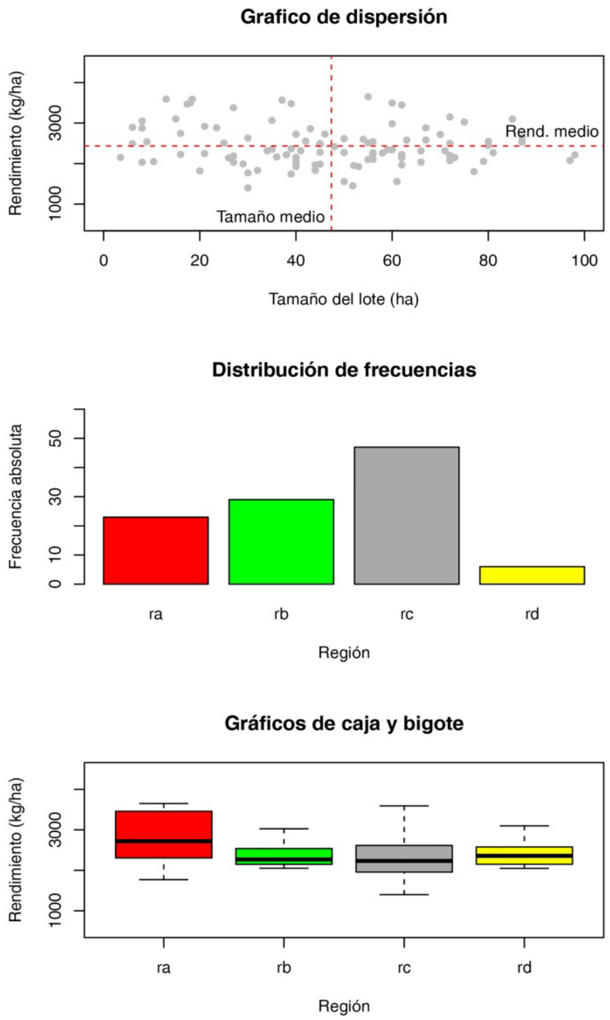

Figura 1. 6. Personalización de gráficos combinando la función `par()`, los argumentos de una función de alto nivel y funciones de bajo nivel

El argumento `type` nos permite modificar la representación del gráfico de dispersión:

`type="p"`: nube de puntos (opción por defecto),
`type="l"`: líneas,
`type="b"`: puntos y líneas,
`type="n"`: gráfico vacío.

Otras funciones de alto nivel comúnmente usadas son `hist()`, `boxplot()` y `barplot()`. Los siguientes son algunos de los argumentos comunes a las funciones gráficas de alto nivel:

`xlim`; `ylim`: control del rango de los ejes *x* e *y*, respectivamente,
`xlab`; `ylab`: etiquetas de los ejes *x* e *y*, respectivamente,
`main`; `sub`: título y subtítulo, respectivamente,
`lty`; `lwd`: control del tipo y ancho de línea, respectivamente,
`col`: control del color,
`font`: control de la fuente del texto.

Las funciones de bajo nivel nos sirven para enriquecer los gráficos, por ejemplo, agregándoles puntos, líneas, texto y más. Por ejemplo, en el gráfico anterior usamos las funciones `abline()` debajo del gráfico de dispersión para agregar una línea perpendicular al eje *y*, la cual indica el valor del rendimiento promedio, junto a una línea perpendicular al eje *x*, en este caso para indicar el tamaño de lote promedio. Además, añadimos texto al gráfico mediante la función y `text()`. Otras funciones de bajo nivel comúnmente usadas son `points()`, `legend()` y `curve()`.

1. 2. 13. La ayuda en R

Siempre podemos consultar a R sobre las especificaciones de una función escribiendo su nombre dentro de `help()` o luego de un signo de preguntas:

```
help(read.table)

## starting httpd help server ... done

?read.table
```

Para que R nos muestre la ayuda de la función de un paquete, este debe estar previamente cargado (las funciones del paquete base están siempre disponibles porque se carga al abrir R).

```
?Anova

## No documentation for 'Anova' in specified packages and libraries:
## you could try '??Anova'

library(car)
?Anova
```

Una de las ventajas de consultar la ayuda de R es que devuelve documentos con una estructura estandarizada (utilidad de la función, paquete en el que se encuentra, argumentos, sintaxis). En RStudio, además, la tecla «tab» nos ayuda con las especificaciones de la función.

1. 3. Resumen de R en tablas

Las siguientes tablas resumen el manejo básico de R. En el caso de las funciones, todas se encuentran en el paquete base. No se trata de una lista exhaustiva y la separación en grupos es arbitraria. Muchas de ellas podrían estar incluidas en varias tablas.

Tabla 1. 1. Tipos de datos usuales en R

Tipo	Ejemplo	Descripción
`character`	`"dos"`	cadenas de texto
`complex`	`2 + 0i`	números complejos
`factor`	`dos`	categorías (niveles de un factor)
`integer`	`2L`	números enteros
`logical`	`TRUE` ; `FALSE`	valores lógicos
`na`	`NA`	elemento faltante (se desconoce)
`null`	`NULL`	elemento nulo (no existe)
`numeric`	`2.0`	números reales

Tabla 1. 2. Estructuras de datos en R

Estructura	Descripción
`arrays`	variables indexadas en un arreglo n-dimensional de elementos de un mismo tipo
`list`	elementos no necesariamente del mismo tipo
`matrix`	elementos de un mismo tipo organizados en forma rectangular
`vector`	elementos de un mismo tipo

Tabla 1. 3. Operadores en R

Operador	Tipo	Descripción
`+`	aritmético	suma
`-`	aritmético	resta
`*`	aritmético	multiplicación
`/`	aritmético	división
`%/%`	aritmético	división entera
`^`	aritmético	potencia
`==`	relacional (*)	«igual que»
`¡=`	relacional	«distinto de»
`<`	relacional	«menor a»
`<=`	relacional	«menor igual a»
`>`	relacional	«mayor a»
`>=`	relacional	«mayor igual a»
`&`	lógico (*)	«y»
`|`	lógico	«o»
`¡`	lógico	«no es»
`xor`	lógico	«o uno o el otro»

Nota: (*) El resultado de aplicar operadores relacionales y lógicos es `TRUE` o `FALSE`.

Tabla 1. 4. Algunas funciones de intérprete (tipo «shell») y de entorno en R

Función	Descripción
`attach()`	acceso a las variables de incluidas en un `data.frame`
`class()`	clase de un objeto
`detach()`	remoción del acceso dado por `attach()`
`dir()`	lista de los archivos alojados en el directorio de trabajo
`getwd()`	directorio de trabajo actual
`install.packages()`	instalación de paquetes
`library()`	carga de paquetes
`ls()`	similar a `objects()`
`objects()`	lista de objetos dentro del espacio de trabajo
`options()`	opciones globales para cómputo y visualización
`rm()`	remoción de un objeto del espacio de trabajo
`rm(list = ls())`	remoción de todos los objetos del espacio de trabajo
`setwd()`	configuración del directorio de trabajo
`str()`	estructura de un objeto

Tabla 1. 5. Algunas funciones útiles para el manejo de datos en R

Función	Descripción
`as.data.frame()`	coerción a `data.frame`
`as.Date()`	coerción a `Date`
`as.factor()`	coerción a `factor`
`as.integer()`	coerción a `integer`
`as.matrix()`	coerción a `matrix`
`as.numeric()`	coerción a `numeric`
`as.table()`	coerción a `table`
`as.vector()`	coerción a `vector`
`dim()`	dimensiones de una matriz o `data.frame`
`head()`	visualización de las primeras filas de un `data.frame`
`length()`	largo de un objeto (cantidad de elementos)
`merge()`	combinación/fusión de dos `data.frame`
`na.omit()`	remoción de datos faltantes
`names()`	nombres de un `data.frame`
`ncol()`	número de columnas de una matriz o `data.frame`
`nrow()`	número de filas de una matriz o `data.frame`
`order()`	arregla un `data.frame` según el orden de sus vectores
`read.csv2()`	importación de datos desde archivo separado por comas
`read.table()`	importación de datos desde archivo
`sort()`	ordena un vector en forma creciente/decreciente
`split()`	partición de datos
`subset()`	selección de un subconjunto de datos
`View()`	visualización de los datos

Tabla 1. 6. Algunas funciones estadísticas en R

Función	Descripción
`cor()`	coeficiente de correlación (Pearson, Kendall, Spearman)
`cov()`	covarianza
`IQR()`	rango intercuartílico
`max()`	máximo
`mean()`	media aritmética
`median()`	mediana
`min()`	mínimo
`quantile()`	cuantiles
`range()`	rango
`sd()`	desvío estándar
`var()`	varianza
`weighted.mean()`	media ponderada

Tabla 1. 7. Algunas funciones matemáticas en R

Función	Descripción
`abs()`	valor absoluto
`acos()`, `asin()`, `atan()`	arcocoseno, arcoseno, arcotangente
`cos()`, `sin()`, `tan()`	coseno, seno, tangente
`cumsum()`	suma acumulativa
`det()`	determinante de una matriz
`diag()`	diagonal de una matriz
`eigen()`	descomposición de una matriz
`exp()`	exponencial
`factorial()`	factorial
`floor()`	redondeo al entero menor
`log()`	logaritmo natural
`log2()`	logaritmo en base 2
`log10()`	logaritmo en base 10
`round()`	redondeo a un decimal especificado
`sqrt()`	raíz cuadrada
`sum()`	sumatoria
`sqrt()`	raíz cuadrada
`t()`	transpuesta de una matriz

Tabla 1. 8. Funciones para operar con distribuciones de probabilidad en R

Función	Tipo	Ejemplos*
`d + abreviación`	densidad de probabilidad/probabilidad	`dnorm(), dt(), df()`
`p + abreviación`	probabilidad acumulada	`pnorm(), pt(), pf()`
`q + abreviación`	cuantiles	`qnorm(), qt(), qf()`
`r + abreviación`	muestra aleatoria	`rnorm(), rt(), rf()`

Nota. (*) Los ejemplos dados corresponden a tres distribuciones claves en los modelos lineales generales: la distribución normal, cuya abreviación en R es `norm`, la distribución *t* de Student, cuya abreviación es `t`; y la distribución F, cuya abreviación es `f`.

Tabla 1. 9. Funciones para gráficos básicos en R

Función	Descripción
`barplot()`	gráficos de barras
`boxplot()`	gráficos de caja y bigotes
`hist()`	histogramas
`mosaicplot()`	gráficos en mosaico
`pie()`	gráficos de tortas
`plot()`	gráficos de dispersión y otros

Capítulo 2. Regresión lineal simple

2. 1. Introducción

Este capítulo presenta el modelo de regresión lineal simple, a partir del cual una variable dependiente cuantitativa es relacionada a una única variable independiente también cuantitativa. Siendo el primero de los modelos, sirve de base para mostrar el flujo de trabajo que seguimos en el resto de los capítulos (en la práctica el orden no es rígido y el proceso es más bien iterativo):

- planteo del problema,
- carga y exploración de datos,
- planteo del modelo estadístico para resolver el problema,
- estimación de los parámetros del modelo, enfatizando la importancia de interpretar tales parámetros en términos del problema,
- evaluación de la incertidumbre en la estimación de los parámetros,
- bondad de ajuste,
- validación de supuestos,
- contraste de hipótesis,
- predicciones sobre la variable dependiente.

Como aspecto central, introducimos la función `lm()` desde la cual discutimos: error estándar, intervalos de confianza y pruebas de hipótesis sobre los parámetros del modelo como medidas de incertidumbre e inferencia; intervalos de confianza y de predicción de la variable dependiente como medidas de capacidad predictiva; cuadrado medio del error y coeficiente de determinación como medidas de bondad de ajuste; y la validación de los supuestos de linealidad, normalidad, homogeneidad de varianzas e independencia en base a los residuos del modelo, haciendo énfasis en la variabilidad propia del muestreo y en el tratamiento de los valores extremos.

2. 2. Problema

El propósito es evaluar el aumento del rendimiento de girasol en función de la cantidad de fertilizante nitrogenado que se aplica en el lote. Los fertilizantes son un insumo utilizado en la agricultura convencional para aumentar el rendimiento del cultivo. También se busca determinar la magnitud del aumento, ya que es relevante para la predicción del rendimiento y la rentabilidad económica.

2. 3. Datos

Cargamos los datos recordando definir el directorio de trabajo previamente:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
```

Y los exploramos:

```
names(dat.rend) # nombre de las columnas

## [1] "lote"        "rendimiento" "borde"       "tamanio"     "N"
## [6] "dsiembra"    "fsiembra"    "cprevio"     "region"

length(dat.rend$id) # cantidad de lotes relevados

## [1] 105

summary(dat.rend) # medidas resumen

##       lote       rendimiento        borde           tamanio             N
##  Min.   :  1   Min.   :1400   Min.   :  6.55   Min.   :  3.50   Min.   : 3.30
##  1st Qu.: 27   1st Qu.:2078   1st Qu.: 32.96   1st Qu.: 30.00   1st Qu.: 4.40
##  Median : 53   Median :2315   Median : 48.39   Median : 46.00   Median : 5.50
##  Mean   : 53   Mean   :2432   Mean   : 48.14   Mean   : 47.37   Mean   :15.04
##  3rd Qu.: 79   3rd Qu.:2720   3rd Qu.: 61.72   3rd Qu.: 62.00   3rd Qu.: 9.00
##  Max.   :105   Max.   :3650   Max.   :102.24   Max.   :105.00   Max.   :83.40
##     dsiembra        fsiembra            cprevio              region
##  Min.   :31000   Length:105         Length:105         Length:105
##  1st Qu.:54600   Class :character   Class :character   Class :character
##  Median :57600   Mode  :character   Mode  :character   Mode  :character
##  Mean   :55742
##  3rd Qu.:60000
##  Max.   :67000
```

Nuestro marco conceptual predice un aumento del rendimiento con la aplicación de fertilizantes y queremos estimar la magnitud del mismo. Exploramos esta relación mediante un gráfico de dispersión (figura 2. 1):

```
# figura 2.1
plot(dat.rend$N, dat.rend$rendimiento, xlab="N (kg/ha)", ylab="Rendimiento (kg/ha)",
ylim=c(500, 4500))
```

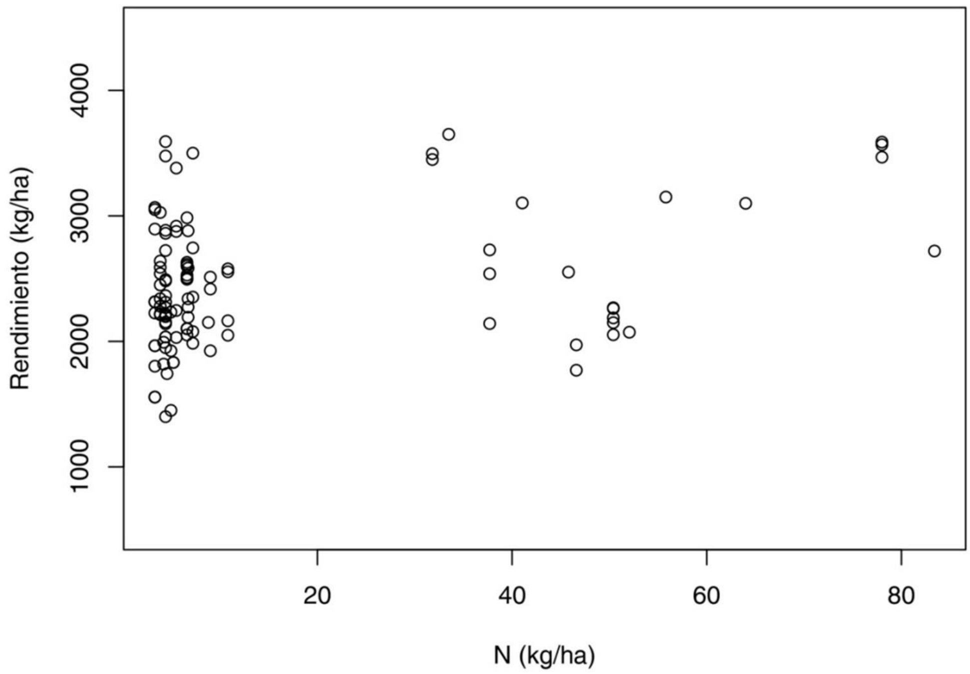

Figura 2. 1. Gráfico de dispersión entre rendimiento y fertilización nitrogenada.

Parece observarse un aumento del rendimiento con la aplicación de fertilizante.

2. 4. Modelo y ajuste

Debemos traducir nuestras ideas conceptuales al lenguaje de la estadística. Bajo nuestro marco de modelado planteamos una regresión lineal simple para estudiar cómo varía el rendimiento de girasol en función de la cantidad de fertilizante aplicado:

$$r_i = \beta_0 + \beta_1 \times n_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0;\, \sigma^2)_{independientes}$$

donde

r_i = rendimiento (kg ha^{-1}) del lote *i* (*i* = 1, 2, … N)
n_i = cantidad de fertilizante nitrogenado aplicado en el lote *i* (kg ha^{-1})
ε_i = término de error (kg ha^{-1})

El modelo tiene tres parámetros a estimar: β_0, β_1 y σ^2. Nuestro interés se centra en la pendiente (β_1) que expresa la variación promedio en el rendimiento del lote por cada unidad de fertilizante adicionada. En este problema, la ordenada al origen o intercepto (β_0) también es de interés práctico ya que expresa el rendimiento promedio de los lotes que no aplican fertilizante. La varianza (σ^2) nos provee una medida de variabilidad del rendimiento entre lotes que aplican la misma cantidad de fertilizante.

2. 4. 1. Ajuste del modelo

Estos tres parámetros son constantes poblacionales (caracterizan el total de lotes de la región de interés) que estimamos a partir de nuestros datos (los 105 lotes relevados). El modelo estimado (muestra) es:

$$r_i \;=\; b_0 \;+\; b_1 \times n_i \;+\; e_i$$

$$e_i \sim \mathcal{N}(0;\, s^2)_{independientes}$$

donde b_0, b_1 y s^2 son, respectivamente, los estimadores de β_0, β_1 y σ^2. En R, para estimar los parámetros de una regresión lineal simple usamos la función `lm()` («lm» por *linear model*), que implementa el «Método de Mínimos Cuadrados Ordinarios» (MMCO).

```
?lm # función para ajustar modelos lineales por MMCO
```

Lo que debemos hacer es indicar la variable dependiente seguido del operador `~`, la independiente, y el `data.frame` en el cual se encuentran ambos vectores:

```
m.rls <- lm(rendimiento ~ N, data=dat.rend) # ajuste del modelo
```

`m.rls` es el objeto en el que guardamos el modelo ajustado (`rendimiento ~ N` se lee como rendimiento en función de fertilizante). Para ver el ajuste a los datos podemos usar la función `abline()` sobre el gráfico de dispersión (figura 2. 2):

```
abline(m.rls, lwd=3, col="blue") # insertamos la recta (figura 2.2)
```

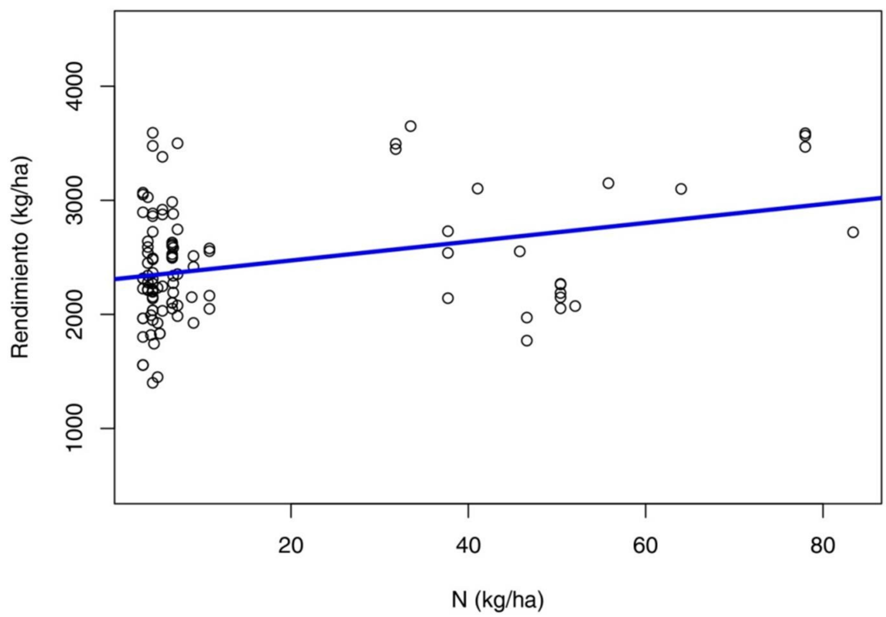

Figura 2. 2. Gráfico del modelo ajustado (línea azul)

Imprimimos los resultados del modelo ajustado:

```
m.rls # imprimimos el resultado del ajuste

##
## Call:
## lm(formula = rendimiento ~ N, data = dat.rend)
##
## Coefficients:
## (Intercept)            N
##    2307.908        8.231
```

La salida nos muestra las estimaciones puntuales del intercepto `(Intercept)` y de la pendiente (`N`), desde las cuales informamos la ecuación de regresión lineal simple que predice el rendimiento de girasol promedio:

$$\hat{r} = 2307{,}9 + 8{,}2 \times n$$

El sombrero en *r* denota que es su estimación. En términos del problema, inferimos un aumento promedio en el rendimiento de 8,2 kg ha^{-1} por cada kg ha^{-1} que adicionamos de fertilizante (interpretación de la pendiente, cabe señalar que las unidades se cancelan y el parámetro es adimensional) y que los lotes que no fertilizan tienen un rinde promedio de 2308,9 kg ha^{-1} (el intercepto se expresa en kg ha^{-1}, las unidades de la variable respuesta). La función `summary()` nos permite extraer más información del modelo ajustado:

```
summary(m.rls)

##
## Call:
## lm(formula = rendimiento ~ N, data = dat.rend)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -944.1 -347.8  -59.6  265.3 1247.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2307.908     59.825  38.577  < 2e-16 ***
## N              8.231      2.385   3.452 0.000809 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 490.6 on 103 degrees of freedom
## Multiple R-squared:  0.1037, Adjusted R-squared:  0.09498
## F-statistic: 11.91 on 1 and 103 DF,  p-value: 0.0008092
```

La estimación de *s* es 490,6 kg ha^{-1} y en esta salida se informa como `Residual standard error`. Por lo tanto, la estimación de la varianza del modelo es 240688,4 kg^{2} ha^{-2} (490,6^{2}).

2. 4. 2 Incertidumbre sobre los parámetros estimados

Los valores reportados en la ecuación de regresión lineal simple son las estimaciones puntales de b_0 y b_1. Como sabemos, una estimación puntual es útil en la medida que sea acompañada por medidas de incertidumbre. Como medidas de incertidumbre podemos usar el error estándar del estadístico o su intervalo de confianza. En la salida anterior, las estimaciones puntuales se informan como `Estimate` y los errores estándar de b_0 y b_1 como `Std. Error`. Si solo nos interesa la información relacionada a los coeficientes estimados:

```
summary(m.rls)$coef

##                Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2307.907572  59.825357 38.577414 4.794763e-63
## N              8.230847   2.384574  3.451705 8.091990e-04
```

Una forma de obtener los intervalos de confianza de b_0 y b_1 es mediante la función `confint()`:

```
confint(m.rls, level=0.95) # IC para el intercepto y la pendiente

##                   2.5 %     97.5 %
## (Intercept) 2189.258086 2426.55706
## N              3.501607   12.96009
```

Desde el argumento `level` hemos pedido intervalos de confianza del 95 % (0.95 es el valor por defecto por lo que podríamos haber omitido el argumento).

2. 5. Bondad de ajuste

La bondad de ajuste nos informa sobre cuán cerca están las predicciones del modelo ajustado ($\hat{r}_i$) de los datos observados (r_i). Existen varias alternativas para su evaluación y todas involucran ambas cantidades. Veamos algunas de ellas.

2. 5. 1. Cuadrado medio del error

El cuadrado medio del error (CME) es la estimación de σ^2, es decir, la varianza residual del modelo:

$$\mathrm{CME} = \frac{\sum_{i=1}^{n}(r_i - \hat{r}_i)^2}{n-2}$$

A diferencia de la varianza clásica, se divide por $n - 2$ ya que en el modelo de regresión lineal simple la estimación del rendimiento promedio involucra dos parámetros (el intercepto y la pendiente). El CME constituye un indicador de lo que no es explicado por el modelo, de manera que cuando mayor es su valor, menor es la bondad de ajuste. Podemos calcularlo desde los residuos:

```r
residuos <- resid(m.rls) # residuos del modelo
cme <- (sum(residuos^2))/(length(residuos)-2) # ∑(residuos)²/(n-2)
cme # CME

## [1] 240695.9
```

O, en forma más simple, desde el `summary`:

```r
summary(m.rls)$sigma^2 # CME desde el summary() del modelo ajustado

## [1] 240695.9
```

2. 5. 2. Error estándar residual

El error estándar residual es la raíz cuadrada de la varianza residual de manera que se trata de un desvío estándar:

```r
sqrt(cme) # raíz cuadrada del CME

## [1] 490.6077

summary(m.rls)$sigma # informado en el summary() del modelo ajustado

## [1] 490.6077
```

Su denominación puede confundir ya que no cuantifica la variabilidad de un estadístico (como un error estándar) sino la variabilidad de los datos en torno a las predicciones del modelo. En el caso de un modelo de regresión lineal simple, este indicador mide cuánto se alejan en promedio los datos de la recta.[1] Por lo tanto, a mayor error estándar residual menor bondad de ajuste. Una de sus ventajas es que se expresa en las mismas unidades que la variable dependiente.

2. 5. 3. Coeficiente de determinación (R^2)

El coeficiente de determinación (R^2) se calcula como:

$$R^2 = \frac{\text{SCT} - \text{SCE}}{\text{SCT}}$$

donde SCT y SCE son las sumas de cuadrados total y del error, respectivamente:

$$\text{SCT} = \textstyle\sum_{i=1}^{n}(r_i - \bar{r})^2$$

$$\text{SCE} = \sum_{i=1}^{n}(r_i - \hat{r}_i)^2$$

[1] Formalmente no es un promedio porque su denominador es $n - p$ (siendo p el número de parámetros de la componente determinística).

El coeficiente de determinación representa la proporción (o porcentaje) de la variabilidad de la variable dependiente que es explicada por el modelo así que a mayor R^2, mayor bondad de ajuste. Tiene un rango entre cero (el modelo no explica nada de la variabilidad) y uno o 100 si está expresado en porcentaje (ajuste perfecto). El R^2 es informado como `Multiple R-squared` en la salida del `summary` del modelo y podemos extraerlo directamente:

```
summary(m.rls)$r.squared # R² informado en el summary()

## [1] 0.1036796
```

o calcularlo con los residuos del modelo y los rendimientos observados:

```
sce <-sum(residuos^2) # SCE
sct <- sum((dat.rend$rendimiento - (mean(dat.rend$rendimiento)))^2) # SCT
R2 <- (sct - sce)/sct # R²
R2

## [1] 0.1036796
```

El valor de `0.1036796` indica que el 10 % de la variación en el rinde de girasol de los lotes es explicado por una relación lineal con la cantidad de fertilizante nitrogenado que se aplica. No debemos confundir el R^2 con el coeficiente de correlación de Pearson (r_p), el cual mide la asociación lineal entre dos variables:

```
rp <- cor(dat.rend$rendimiento, dat.rend$N) # coeficiente de correlación de Pearson
rp

## [1] 0.3219932
```

De todos modos, en la regresión lineal simple ambos coeficientes están estrechamente relacionados ya que $\sqrt{R^2} = r_p$:

```
sqrt(R2) # raíz cuadrada de R²

## [1] 0.3219932
```

Sin embargo, la raíz cuadrada del coeficiente de determinación no nos indica el tipo de asociación. Cuando la relación es negativa, $\sqrt{R^2}$ igual será positivo y tenemos que multiplicar por -1 para obtener r_p. Con la función `cor.test()` obtenemos el valor p bajo una hipótesis nula de que no existe asociación lineal ($r_p = 0$):

```
cor.test(dat.rend$rendimiento, dat.rend$N) # prueba de hipótesis correlación lineal

## 
##  Pearson's product-moment correlation
## 
## data:  dat.rend$rendimiento and dat.rend$N
## t = 3.4517, df = 103, p-value = 0.0008092
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1389002 0.4838007
## sample estimates:
##       cor 
## 0.3219932
```

y encontramos que es el mismo que aquel obtenido en la prueba *t* a dos colas.

2. 5. 4. Gráfico de observados vs. predichos

Si el ajuste fuera perfecto, cada valor predicho debería ser igual al observado. Mientras más discordantes los valores, menor bondad de ajuste. Por lo tanto, un gráfico interesante es el de observados en función de predichos. En este, la línea uno a uno sirve como referencia al ajuste perfecto (figura 2. 3):

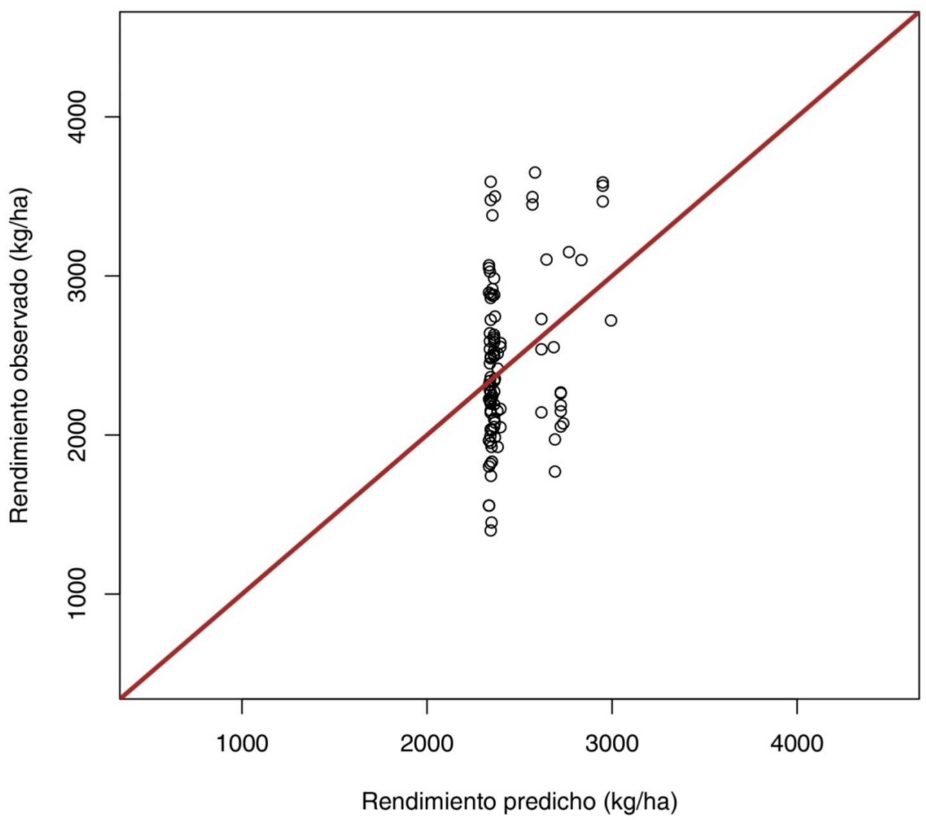

Figura 2. 3. Gráfico de observados *vs* predichos para evaluar la bondad de ajuste del modelo estimado
Nota. La línea representa una recta 1 a 1 sobre la cual las observaciones deberían localizarse si las predicciones fueran perfectas.

```
predichos <- fitted(m.rls) # rendimientos predichos
# figura 2.3
plot(predichos, dat.rend$rendimiento,
     xlab="Rendimiento predicho (kg/ha)", ylab="Rendimiento observado (kg/ha)",
     xlim=c(500, 4500), ylim=c(500, 4500))
abline(a=0,b=1,lwd=3, col="brown") # recta uno a uno
```

2. 6. Validación de supuestos

Recordemos que la componente aleatoria del modelo:

$$\varepsilon_i \sim \mathcal{N}(0;\, \sigma^2)_{independientes}$$

implica asumir que los residuos son:

- normales,
- homocedásticos (igual varianza en toda la recta),
- su promedio es cero (la relación es lineal),
- independientes (los residuos no están correlacionados).

Si estos supuestos no se cumplen, las inferencias y predicciones del modelo no serán válidas. La validación se puede realizar gráficamente, aunque también contamos con indicadores numéricos y pruebas de hipótesis.

2. 6. 1. Homocedasticidad, linealidad e independencia

La homocedasticidad, la linealidad y la independencia son supuestos que los evaluamos sobre el gráfico de residuos en función de predichos (figura 2.4), en el que no debemos encontrar ningún patrón. Complementariamente, podemos evaluar el gráfico de residuos en función de la variable independiente:

```
# figura 2.4
par(mfcol=c(1,2))

# - Gráfico de residuos vs. predichos
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

# - Gráfico de residuos vs. variable independiente
plot(dat.rend$N, residuos, xlab="N (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
```

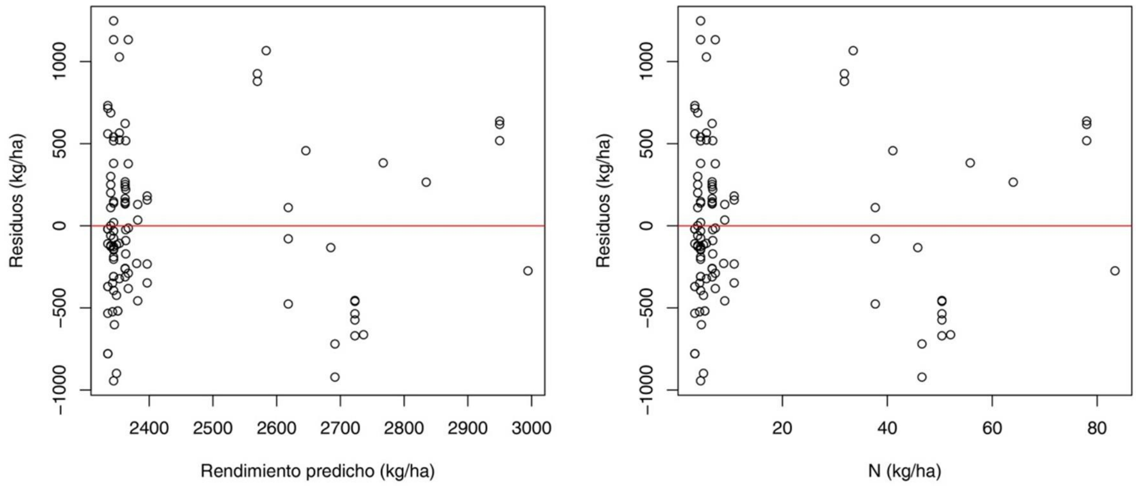

Figura 2. 4. Gráfico de residuales en función de: predichos (izquierda) y variable independiente (derecha)
Nota. La línea horizontal (residual = 0) sirve de referencia para evaluar la existencia de patrones en los residuales con los cuales detectar heterocedasticidad o falta de independencia por no linealidad.

El paquete `car` presenta una función útil para evaluar los residuos (figura 2. 5):

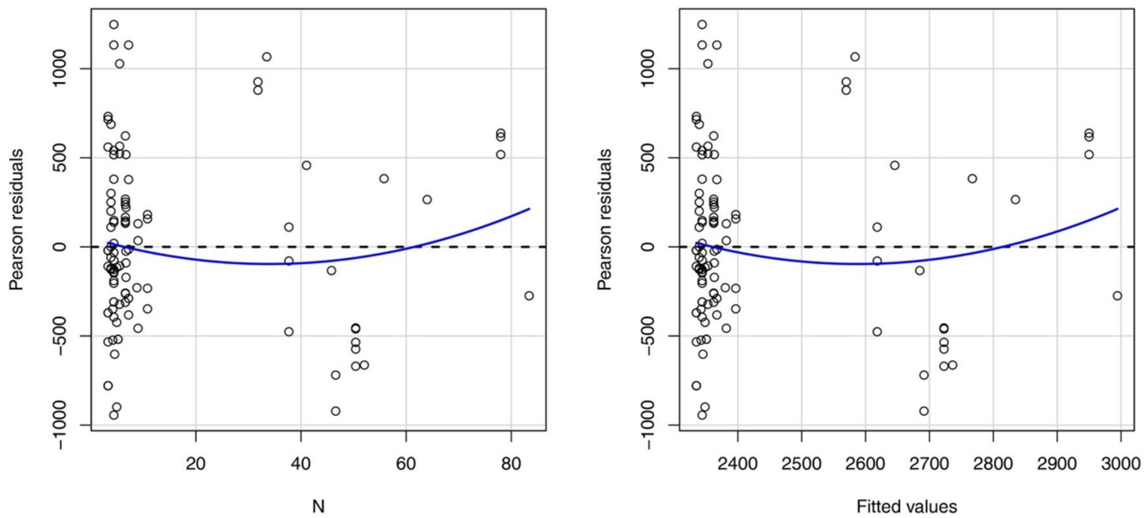

Figura 2. 5. Gráfico de residuales obtenido con la función `residualPlots()` del paquete `car`

```
library(car) # cargamos el paquete (debe estar instalado)

## Loading required package: carData

residualPlots(m.rls) # figura 2.5
```

Las figuras 2. 4 y 2. 5 indican que no se observan problemas de linealidad y existe una leve tendencia de menor varianza a mayores rendimientos.

2. 6. 2. Normalidad

La curva normal es simétrica y con forma de campana, y esto es lo que deberíamos encontrar al observar la distribución de los residuos del modelo mediante un histograma y un gráfico de caja y bigotes. De igual forma, los cuantiles de los residuos deberían coincidir con los cuantiles de la distribución normal, lo cual podemos evaluar desde un gráfico cuantil-cuantil («Q-Q plot»).

```
# figura 2.6
par(mfrow=c(1,3))
hist(residuos, main="", xlab="Residuos (kg/ha)", ylab="Frecuencia") # histograma
boxplot(residuos, horizontal=T, xlab="Residuos (kg/ha)", range=1.5) # caja y bigotes
qqnorm(residuos, main="", xlab="Cuantiles teóricos",
       ylab="Cuantiles muestrales") # gráfico cuantil-cuantil (QQ plot)
qqline(residuos, col="red") # línea cuantil-cuantil
```

De acuerdo a la figura 2. 6, los residuos parecen seguir una distribución normal. Podemos corroborarlo aplicando alguno de los test de normalidad. Estos se basan en la diferencia entre las frecuencias observadas (la de los residuos) y las probabilidades teóricas (la distribución de probabilidad asumida) y, en este caso, tienen como hipótesis nula que los residuos siguen una distribución normal. Entre los test de normalidad más usados se encuentran el de Kolmogorov-Smirnov, el de Lilliefors y el de Shapiro-Wilk. Veamos cómo aplicar el primero de ellos:

```
ks.test(residuos,"pnorm", mean(residuos), sd(residuos)) # - Test Kolmogorov-Smirnov

## Warning in ks.test.default(residuos, "pnorm", mean(residuos), sd(residuos)):
## ties should not be present for the Kolmogorov-Smirnov test

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuos
## D = 0.073481, p-value = 0.6222
## alternative hypothesis: two-sided
```

El valor *p* obtenido es superior a 0,05, sugiriendo no rechazar la hipótesis nula del test que afirma normalidad de los residuos.

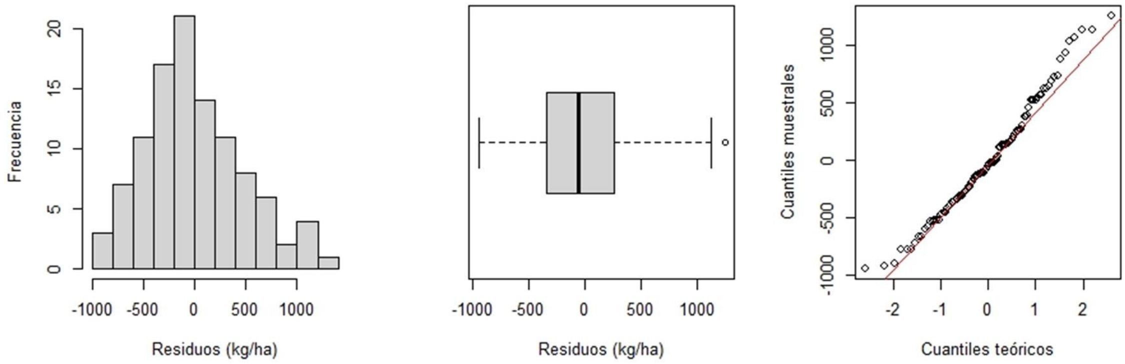

Figura 2. 6. Evaluación gráfica del supuesto de normalidad de residuales mediante un histograma (izquierda), un gráfico de caja y bigotes (medio) y gráficos cuantil-cuantil (derecha)

2. 6. 3. Los supuestos y el muestreo

Al evaluar los residuos debemos ser conscientes del proceso de muestreo. Trabajar con una muestra implica que no observaremos un patrón perfecto a pesar de que los residuos provengan del modelo teórico (normales, homocedásticos, lineales e independientes), y esto es más acentuado cuanto menor es el tamaño muestral. Utilicemos la simulación para para comparar los residuos con 105 datos (nuestro tamaño muestral) extraídos de una distribución normal con media cero y varianza igual a la varianza muestral:

```
set.seed(432) # para reproducir el resultado
normal <- rnorm(mean=0, sd=summary(m.rls)$sigma, n=length(summary(m.rls)$residuals))
```

La función `rnorm()` genera *n* valores (pseudo)aleatorios y por lo tanto cada vez que la ejecutemos obtendremos diferentes resultados. Para que esto no ocurra, debemos fijar un valor en la función `set.seed()` y correr el comando antes de cada ejecución de `rnorm()` (y lo mismo vale para reproducir cualquier simulación estocástica).

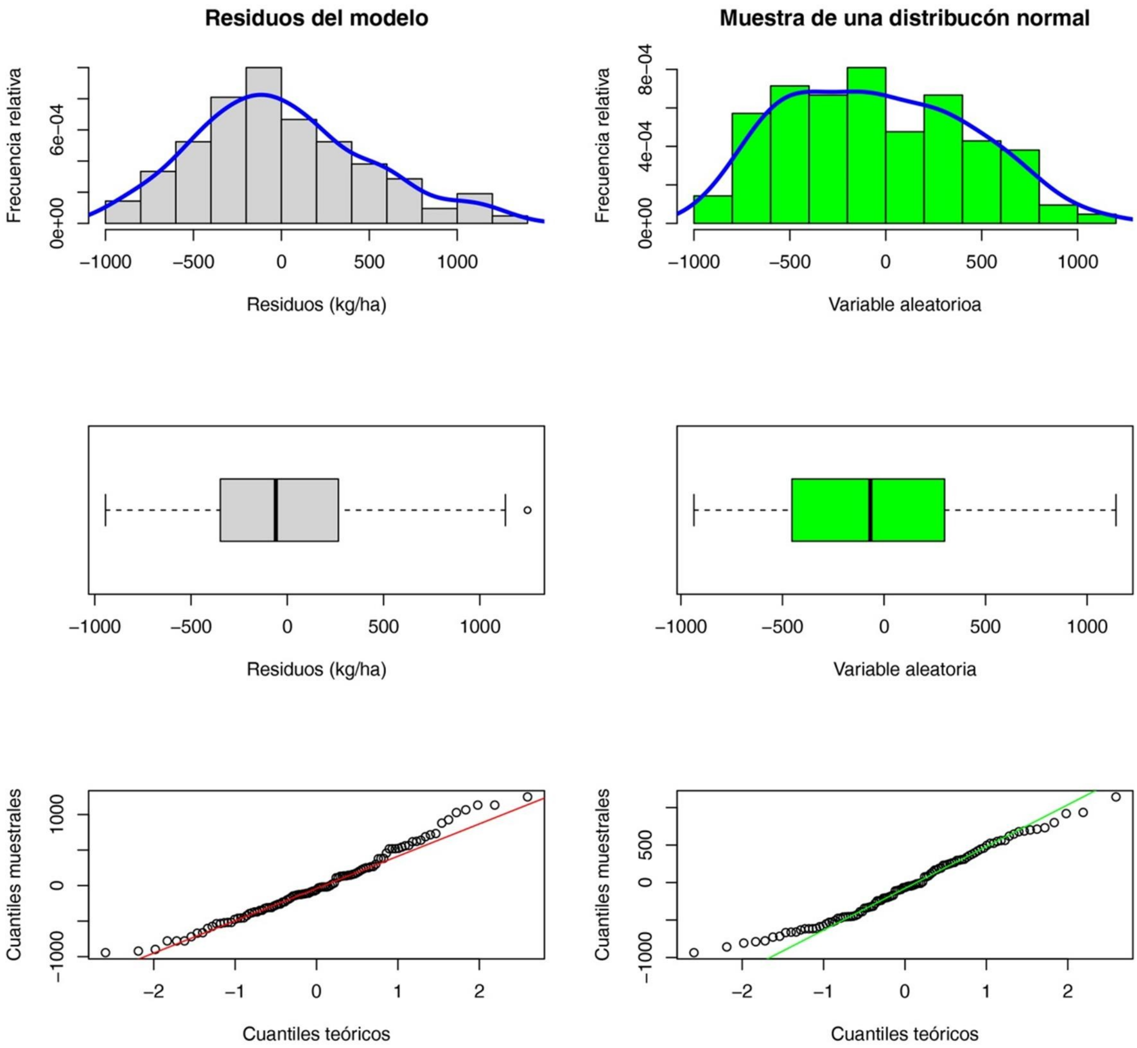

Figura 2. 7. Comparación de la distribución de residuales (izquierda) frente a una muestra del mismo tamaño extraída de una distribución normal con media igual a cero y desvío estándar igual al error estándar residual del modelo ajustado (derecha)

```
# figura 2.7
par(mfrow=c(3,2))
hist(residuos, main="Residuos del modelo", freq=F,
     xlab="Residuos (kg/ha)", ylab="Frecuencia relativa")
lines(density(residuos), col="blue", lwd=3)

hist(normal, col="green", freq=F,
     main="Muestra de una distribucón normal",
     xlab="Variable aleatoria", ylab="Frecuencia relativa")
lines(density(normal), col="blue", lwd=3)

boxplot(residuos, bty="l", range=1.5, horizontal=T, xlab="Residuos (kg/ha)")
boxplot(normal, bty="l", range=1.5, col="green", horizontal=T,
        xlab="Variable aleatoria")

qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")

qqnorm(normal, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(normal, col="green")
```

En una distribución normal, la asimetría (calculada como el tercer momento) es igual a 0 y la curtosis (calculada como el cuarto momento) a 3. Comparamos los residuos con la muestra extraída de una distribución normal:

```
library(moments) # instalar antes el paquete
skewness(residuos) # asimetría de los residuos

## [1] 0.4173252

kurtosis(residuos) # curtosis de los residuos

## [1] 2.797251

skewness(normal) # asimetría de la muestra normal (el valor esperado es 0)

## [1] 0.2515261

kurtosis(normal) # curtosis de la muestra normal (el valor esperado es 3)

## [1] 2.227031
```

Simulamos 105 datos que provienen de una distribución *t* (mayor Curtosis que la normal) y 105 datos de una distribución uniforme (menor Curtosis que la normal) y comparamos gráficamente:

```
# figura 2.8
par(mfcol=c(2,4))

# Residuos del modelo
hist(residuos, main="Residuos del modelo",
     xlab="Residuos (kg/ha)", ylab="Frecuencia")
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")

# Muestra de una distirbución normal estándar
```

```
set.seed(432)
normal <- rnorm(n=105, mean=0, sd=1)
hist(normal, main="Distribución normal", col="green",
     xlab="Variable aleatoria", ylab="Frecuencia")
qqnorm(normal, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(normal, col="green")

# Muestra de una distribución t (mayor curtosis que la normal)
set.seed(432)
t.student <- rt(n=105, df=2)
hist(t.student, main="Distribución t (mayor curtosis)", col="orange",
     xlab="Variable aleatoria", ylab="Frecuencia")
qqnorm(t.student, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(t.student, col="orange")

# Muestra de una distribución uniforme (menor curtosis que la normal)
set.seed(432)
uniforme <- runif(n=105, min=0, max=1)
hist(uniforme,main="Distribución uniforme (menor curtosis)", col="blue",
     xlab="Variable aleatoria", ylab="Frecuencia")
qqnorm(uniforme, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(uniforme, col="blue")
```

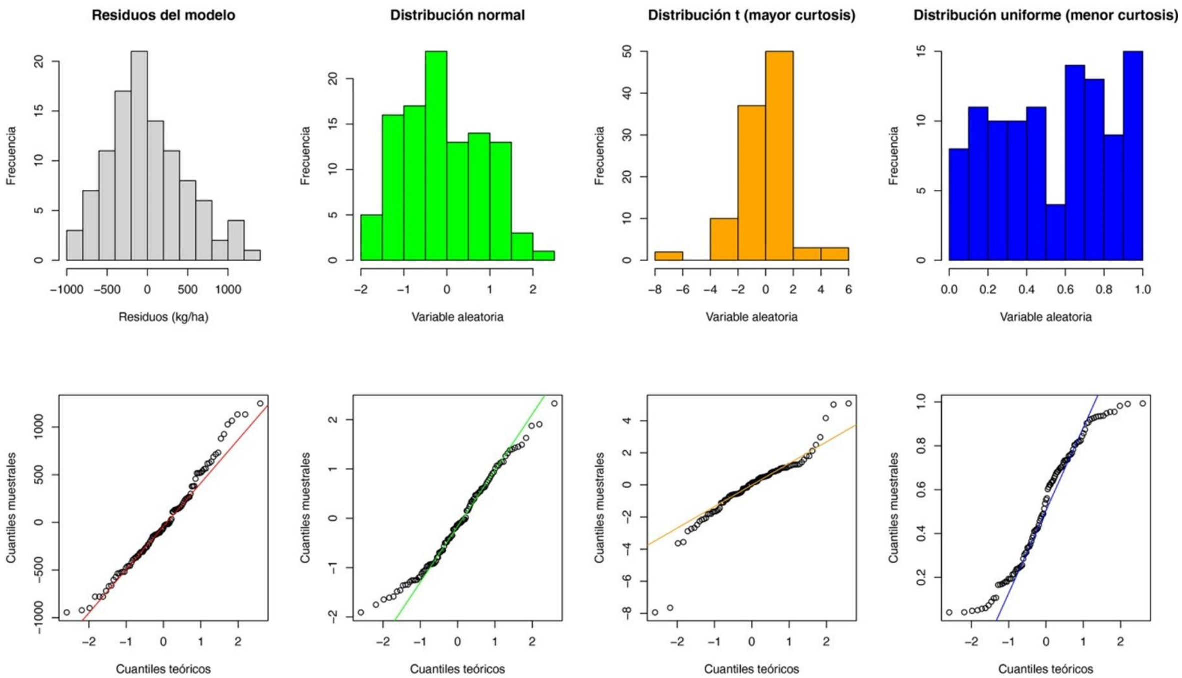

Figura 2. 8. Comparación de la distribución de residuales (histograma y gráfico cuantil-cuantil a la izquierda) con muestras del mismo tamaño obtenidas de distribuciones con diferentes niveles de curtosis: normal, *t* (mayor curtosis que una normal) y uniforme (menor curtosis que una normal)

2. 7. Predicciones

Podemos utilizar nuestro modelo para predecir rendimiento de girasol a niveles de fertilización de interés. Por ejemplo, de acuerdo a nuestro modelo, el rendimiento de girasol (promedio) que obtenemos por aplicar 40 kg ha^{-1} de fertilizante es de 2641,1 kg ha^{-1}:

```
2307.908 + 8.231 * 40

## [1] 2637.148

# alternativamente
m.rls$coefficients[1] + m.rls$coefficients[2] * 40

## (Intercept)
##    2637.141

# utilizando la función "predict.lm"
predict.lm(m.rls, newdata=data.frame(N=40))

##        1
## 2637.141
```

En la función `predict.lm()` ingresamos el valor de la variable independiente como `data.frame` porque el argumento `newdata` necesita un objeto de esta clase. El valor que muestra la salida es la estimación de la media de una distribución normal (la distribución de r_i condicional al valor de n_i). Por lo tanto, tiene su intervalo de confianza y podemos obtenerlo, si así lo indicamos, con la función que usamos para la predicción:

```
predict.lm(m.rls, newdata=data.frame(N=40), interval="confidence")

##        fit      lwr      upr
## 1 2637.141 2485.658 2788.625
```

Y de igual forma el intervalo de predicción:

```
predict.lm(m.rls, newdata=data.frame(N=40), interval="prediction")

##        fit      lwr      upr
## 1 2637.141 1652.415 3621.867
```

con el cual evaluamos probabilidades para unidades experimentales (el rendimiento de un lote), a diferencia del intervalo de confianza que aplica a medias muestrales (el rendimiento promedio de los lotes). Podemos también predecir el rendimiento condicional a la cantidad de nitrógeno aplicado en cada lote de la muestra y extraer los intervalos de confianza y predicción. Para esto, debemos omitir el argumento `newdata`:

```
head(predict.lm(m.rls, interval="confidence"))

##        fit      lwr      upr
## 1 2363.137 2260.328 2465.945
## 2 2363.137 2260.328 2465.945
```

```
## 3 2363.137 2260.328 2465.945
## 4 2363.137 2260.328 2465.945
## 5 2344.123 2236.652 2451.594
## 6 2645.784 2490.401 2801.167
```

```
head(predict.lm(m.rls, interval="prediction"))
```

```
## Warning in predict.lm(m.rls, interval = "prediction"): predictions on current data re
fer to _future_ responses
```

```
##        fit      lwr      upr
## 1 2363.137 1384.716 3341.558
## 2 2363.137 1384.716 3341.558
## 3 2363.137 1384.716 3341.558
## 4 2363.137 1384.716 3341.558
## 5 2344.123 1365.201 3323.045
## 6 2645.784 1660.450 3631.117
```

Una forma útil de comunicar la incertidumbre de un modelo de regresión lineal simple es agregando los intervalos de confianza y de predicción sobre la recta estimada (figura 2. 9). Para esto debemos trabajar un poco. Veamos:

```
# figura 2.9
nx <- seq(from=min(dat.rend$N), to=max(dat.rend$N), by=0.05)
ci <- predict(m.rls, newdata=data.frame(N=nx), interval="confidence")
pi <- predict(m.rls, newdata=data.frame(N=nx), interval="prediction")
intervalos <- data.frame(ci, pi[,c(2,3)])
colnames(intervalos)=c("media", "ICinf", "ICsup", "IPinf", "IPsup")
round(head(intervalos), 1)
```

```
##    media  ICinf  ICsup  IPinf  IPsup
## 1 2335.1 2225.1 2445.1 1355.9 3314.3
## 2 2335.5 2225.6 2445.4 1356.3 3314.7
## 3 2335.9 2226.1 2445.7 1356.7 3315.1
## 4 2336.3 2226.7 2446.0 1357.1 3315.5
## 5 2336.7 2227.2 2446.2 1357.6 3315.9
## 6 2337.1 2227.7 2446.5 1358.0 3316.3
```

```
plot(dat.rend$N, dat.rend$rendimiento,
     xlab="N (kg/ha)", ylab="Rendimiento (kg/ha)",
     ylim=c(500, 4500))
lines(nx, intervalos[,1], lwd=3) # rendimientos predichos
lines(nx, intervalos[,2], col="red", lwd=2, lty=2) # límite inferior IC
lines(nx, intervalos[,3], col="red", lwd=2, lty=2) # límite superior IC
lines(nx, intervalos[,4], col="orange", lty=2) # límite inferior IP
lines(nx, intervalos[,5], col="orange", lty=2) # límite superior IP
```

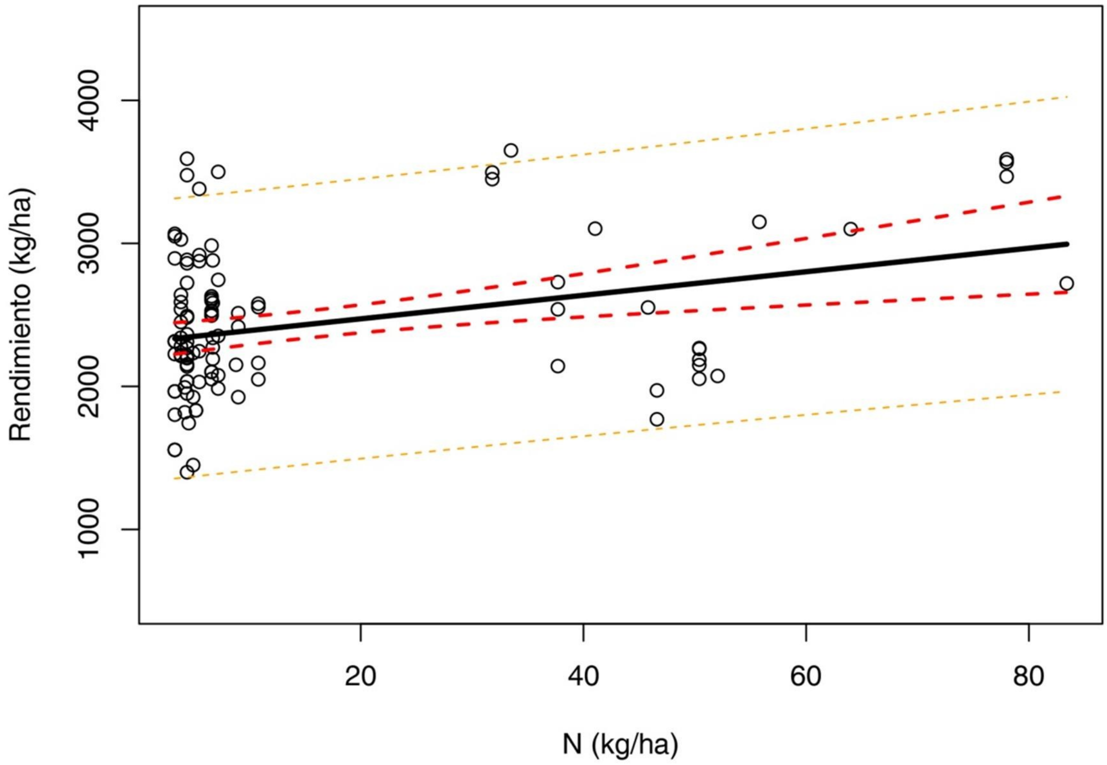

Figura 2. 9. Intervalos de confianza (líneas punteadas de tono oscuro) y predicción (líneas punteadas en tono claro) del modelo ajustado (línea sólida).

El paquete `ggplot2` simplifica la generación de gráficos como estos, aunque demanda algo de tiempo comprender su sintaxis de gráficos en capas. La figura 2. 10 muestra el gráfico obtenido con la función `ggplot()`:

```
library(ggplot2) # instalar antes el paquete

?ggplot

# figura 2.10
df <- data.frame(dat.rend, predict.lm(m.rls, interval="prediction"))

## Warning in predict.lm(m.rls, interval = "prediction"): predictions on current data re
fer to _future_ responses

names(df)

##  [1] "lote"         "rendimiento" "borde"       "tamanio"     "N"
##  [6] "dsiembra"     "fsiembra"    "cprevio"     "region"      "fit"
## [11] "lwr"          "upr"

ggplot(df, aes(N, rendimiento)) +
  geom_line(aes(y=lwr), color="orange", linetype="dashed", lwd=1) +
  geom_line(aes(y=upr), color = "orange", linetype="dashed", lwd=1)+
  geom_smooth(method=lm, se=TRUE, color="black") +
  geom_point(size=2) +
  ylim(500, 4500) + xlab("N (kg/ha)") + ylab("Rendimiento (kg/ha)")

## `geom_smooth()` using formula 'y ~ x'
```

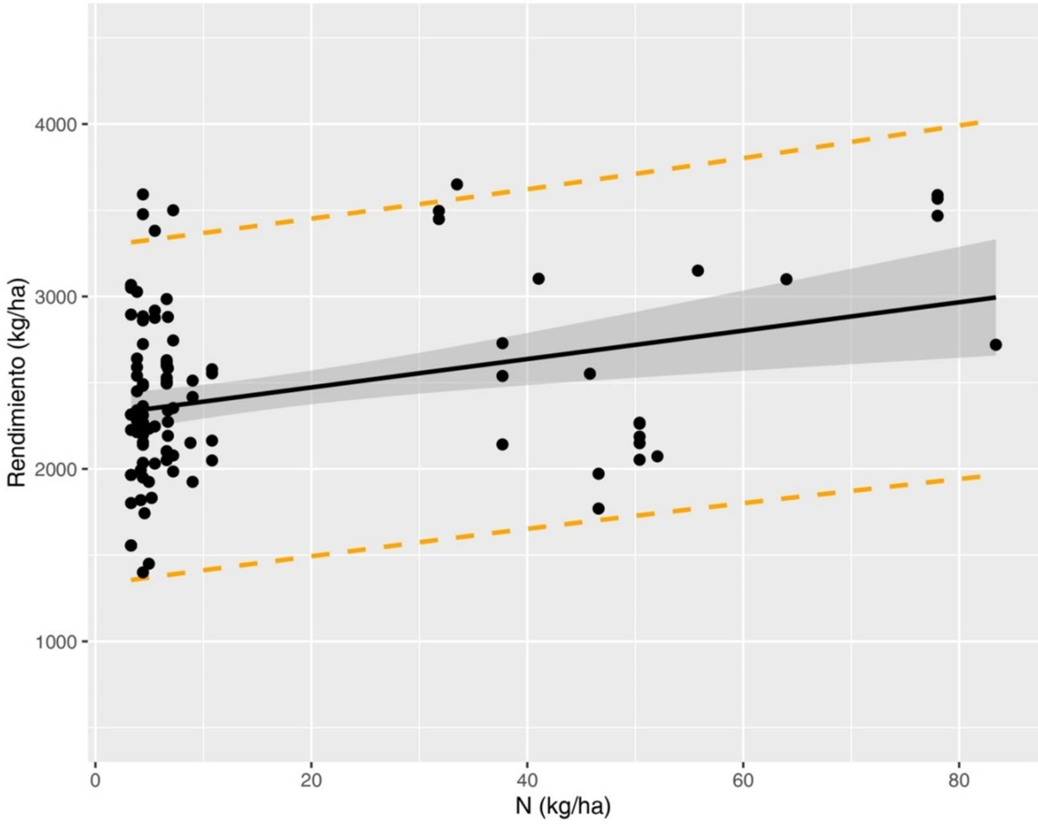

Figura 2. 10. Intervalos de confianza (área sombreada) y predicción (líneas punteadas) del modelo ajustado (línea sólida) graficados con `ggplot()`

2. 8. Hipótesis sobre los parámetros

En general, el interés en un modelo de regresión lineal simple está en la pendiente. En nuestra problemática estamos interesados en cómo varía el rendimiento del cultivo con la fertilización nitrogenada.
Al ajustar el modelo estimamos que, en promedio, el rendimiento aumenta 8,2 kg ha^{-1} por cada kilo (1 kg ha^{-1}) de fertilizante. Ahora bien, ¿este resultado es producto del nitrógeno u ocurrió por azar? Una de las herramientas para concluir al respecto es realizar un contraste de hipótesis sobre la pendiente suponiendo que no existe relación entre variables (hipótesis nula):

$$H_0: \ \beta_1 = 0$$

y calcular el valor *p*, el cual expresa la probabilidad de observar el estadístico (3,452 en este caso), o uno más extremo, en caso de que no haya efecto. El valor *p*, que lo encontramos en la salida del `summary` del modelo, es de 0,000809 e indica que el resultado encontrado es muy poco probable si H_0 fuera cierta. En otras palabras, nos sugiere concluir que existe efecto del nitrógeno. Veamos de donde surge el valor *p* (para esto debemos tener presente que bajo la hipótesis nula la distribución por muestreo de b_1 es una *t* con *n* - 2 grados de libertad):

```
b1 <- summary(m.rls)$coefficients[2,1]
b1 # Estimate N (b1)

## [1] 8.230847

es.b1 <- summary(m.rls)$coefficients[2,2]
es.b1 # Std. Error N (error estándar de b1)

## [1] 2.384574

tobs <- b1/es.b1
tobs # t value N (b1 en escala t)

## [1] 3.451705

pt(q=tobs, df=103, lower.tail=FALSE)*2 # Pr(>|t|)

## [1] 0.000809199
```

Todos los valores *p* que devuelve R para los parámetros de los modelos ajustados son obtenidos bajo H_0: $\beta_j = 0$. Sin embargo, nuestra predicción teórica (aumento del rendimiento con la fertilización) requiere de una prueba de hipótesis a una cola:

$$H_0: \ \beta_1 \leq 0$$

El valor *p* de esta prueba lo obtenemos como:

```
pt(q=tobs, df=103, lower.tail=FALSE)

## [1] 0.0004045995
```

que es lo mismo que dividir por 2 el valor *p* del `summary` del modelo

```
summary(m.rls)$coefficients[2,4]/2

## [1] 0.0004045995
```

Análogamente, si queremos comparar el *t* observado contra el *t* crítico:

```
qt(p=0.05, df=105-2, lower.tail=FALSE) # "t crítico" pruea a una cola

## [1] 1.659782

Tobs # t observado

## [1] 3.451705
```

Como vemos, el *t* observado es más extremo que el *t* crítico, lo que implica el rechazo de la hipótesis nula. La función `anova()` aplicada sobre el modelo ajustado nos devuelve la tabla de ANOVA de la regresión en la que la existencia de relación se pone a prueba indirectamente mediante un análisis de varianza. Profundizaremos sobre este análisis en el capítulo siguiente.

```
anova(m.rls) # tabla de ANOVA del modelo

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716  11.914 0.0008092 ***
## Residuals 103 24791681  240696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

2. 9. Transformaciones

Cuando no se cumplen los supuestos, podemos usar transformaciones (logaritmos, raíces, etcétera). Esto implica transformar la escala de la variable independiente, la dependiente, o ambas. No obstante, los parámetros se estimarán de acuerdo a la escala transformada, dificultando su interpretación en términos del problema. Antes habíamos detectado un posible problema de homogeneidad de varianzas, probemos algunas transformaciones:

```
# figura 2.11
par(mfcol=c(3,3))

# Escala original
plot(dat.rend$N, dat.rend$rendimiento,
     xlab="N (kg/ha)", ylab="Rendimiento (kg/ha)",
     main="Escala original")
abline(m.rls, col="blue", lwd=2)

plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="blue")

hist(residuos, main="",
     xlab="Residuos (kg/ha)", ylab="Frecuencia")

# Transformación log(x)
m.rls.t1 <- lm(rendimiento ~ log10(N), data=dat.rend)
summary(m.rls.t1)

##
## Call:
## lm(formula = rendimiento ~ log10(N), data = dat.rend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -946.00 -332.32  -57.32  282.14 1262.29
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2087.2      114.1  18.290  < 2e-16 ***
## log10(N)       376.9      113.2   3.329  0.00121 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 492.4 on 103 degrees of freedom
## Multiple R-squared:  0.09712,   Adjusted R-squared:  0.08836
## F-statistic: 11.08 on 1 and 103 DF,  p-value: 0.001212
```

```
res.t1 <- resid(m.rls.t1)
pred.t1 <- fitted(m.rls.t1)

plot(log10(dat.rend$N), dat.rend$rendimiento,
     xlab="log 10 (N)", ylab="Rendimiento (kg/ha)",
     main="Transformación log 10 (x)")
abline(m.rls.t1, col="blue", lwd=2)

plot(pred.t1, res.t1,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="blue") # parece haber una mejora

hist(res.t1, main="",
     xlab="Residuos (kg/ha)", ylab="Frecuencia")

# Transformación log(y)
m.rls.t2 <- lm(log10(rendimiento) ~ N, data=dat.rend)
summary(m.rls.t2)

##
## Call:
## lm(formula = log10(rendimiento) ~ N, data = dat.rend)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.215957 -0.059190 -0.002413  0.052248  0.193251
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.3561404  0.0106267  315.82  < 2e-16 ***
## N           0.0013511  0.0004236    3.19  0.00189 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08715 on 103 degrees of freedom
## Multiple R-squared:  0.0899, Adjusted R-squared:  0.08106
## F-statistic: 10.17 on 1 and 103 DF,  p-value: 0.001887

res.t2 <- resid(m.rls.t2)
pred.t2 <- fitted(m.rls.t2)

plot(dat.rend$N, log10(dat.rend$rendimiento),
     xlab="N (kg/ha)", ylab="log 10 (rendimiento)",
     main="Transformación log 10 (y)")
abline(m.rls.t2, col="blue", lwd=2)

plot(pred.t2, res.t2,
     xlab="Valores predichos", ylab="Residuos")
abline(a=0, b=0, col="blue") # la transformación empeora la situación

hist(res.t2, main="",
     xlab="Residuos", ylab="Frecuencia")
```

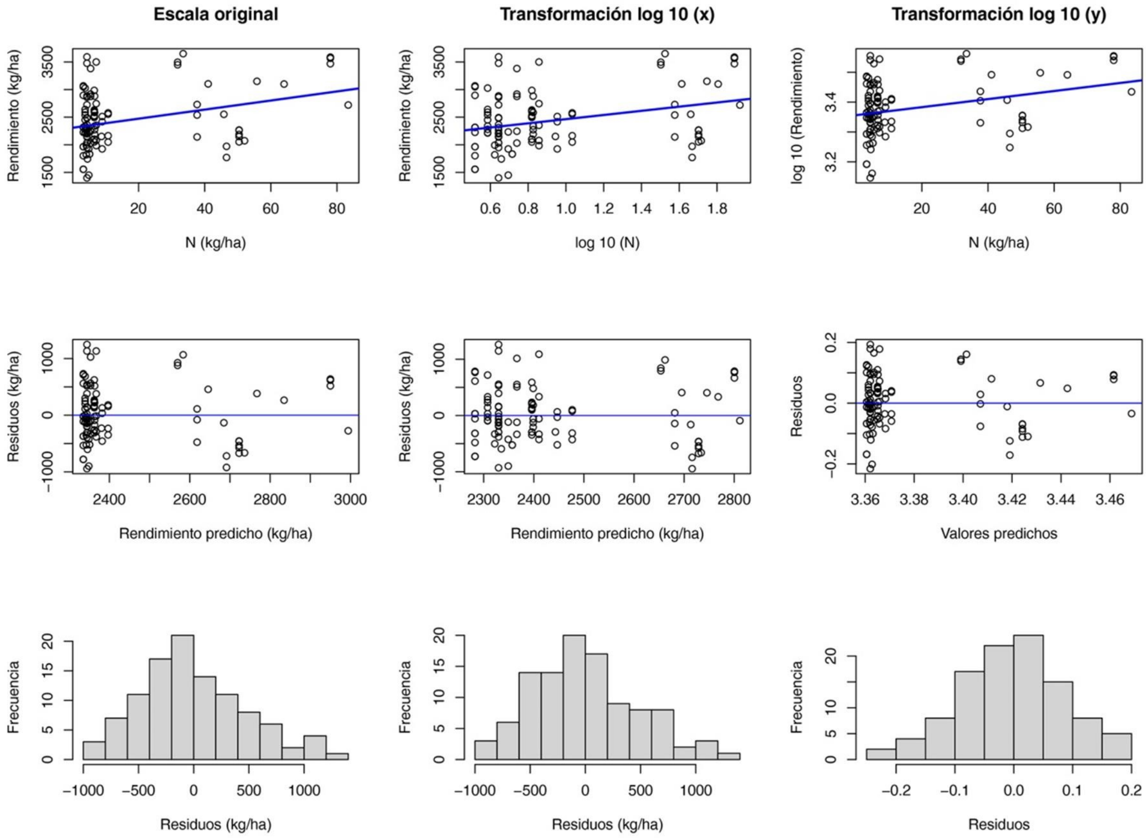

Figura 2. 11. Comparación del modelo ajustado en la escala original (gráficos a la izquierda) respecto a dos opciones de transformación mediante el gráfico de dispersión con la recta de regresión estimada (panel superior), el gráfico de residuos *vs* predichos (panel medio) y el histograma de los residuos (panel inferior)

Transformar el eje *x* a la escala logarítmica parece mejorar la distribución de los residuos (figura 2. 11). Si optamos por esta alternativa, la pendiente estimada es 376,9. Esto implica que el rendimiento aumenta 376,9 kg ha^{-1} por cada aumento de un orden de magnitud (0,2 a 1, 1 a 10, 10 a 10, etcétera) en la cantidad de fertilizante aplicado. Por lo tanto, este modelo indica que la relación entre *x* e *y* en la escala original es no lineal. Como puede intuirse, las transformaciones son una alternativa para linealizar relaciones.

2. 10. Valores extremos

La presencia de valores extremos muchas veces es la causa de la violación a los supuestos del modelo. Existen observaciones:

- atípicas (*ouliers*): alejadas/extremas en el eje *y*,
- de gran *leverage*: alejadas/extremas en el *x*.

Las observaciones atípicas y a la vez de gran *leverage* se consideran influyentes para el análisis. Cuando se las detecta, no deben ser removidas de forma automática (según Steve Hawkins, las observaciones que se alejan mucho del resto son sospechosas de haber sido generadas por un mecanismo diferente, imprimiéndoles gran importancia como para descartarlas). La influencia de una observación puede ser medida por la Distancia de Cook o *Cook's distance*. La función `plot()` tiene seis gráficos para evaluar varios supuestos. De estos, los gráficos 4 y 5 permiten el análisis de los valores extremos y sugieren que no hay observaciones influyentes (figura 2. 12):

```
par(mfcol=c(1,2))
plot(m.rls, which = c(4,5)) # figura 2.12
```

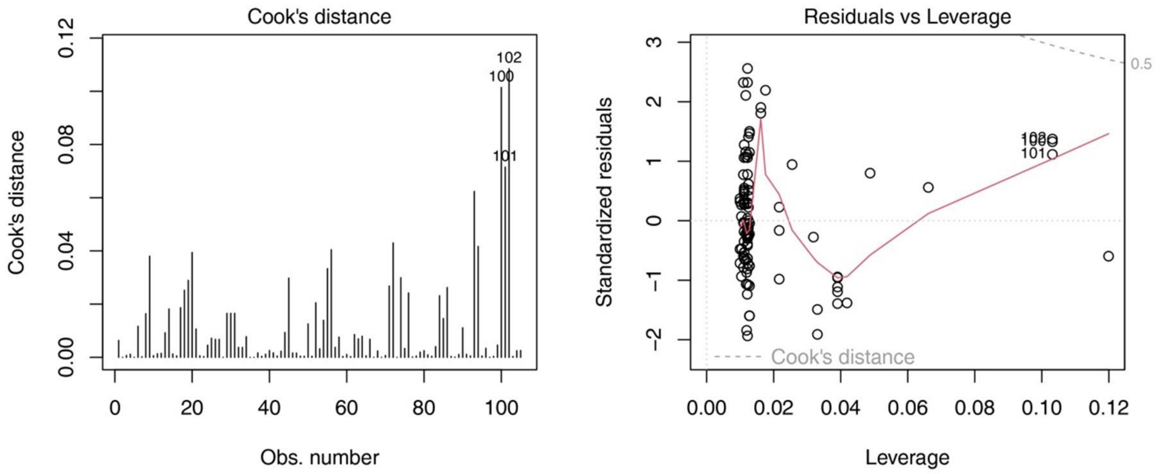

Figura 2. 12. Evaluación de observaciones atípicas y de influencia en base a la Distancia de Cook (izquierda) y el Leverage (derecha).

Una observación de alta influencia tiene gran peso en el ajuste del modelo. En otras palabras, el valor de los parámetros estimados cambia de manera importante al removerlas. Exploremos qué sucede con el ajuste del modelo al agregar valores alejados en el eje *y*, en el eje *x*, y en ambos ejes (figura 2. 13).

```
max(dat.rend$rendimiento)

## [1] 3650

mean(dat.rend$rendimiento)

## [1] 2431.724

max(dat.rend$N)

## [1] 83.4

mean(dat.rend$N)

## [1] 15.04295

outlier <- data.frame(data.frame(rendimiento=7500, # alejado en eje y
                                 N=15 # ~ media eje x
                                 ))

g.lever <- data.frame(data.frame(rendimiento=2430, # ~ media eje y
```

```
                                 N=150 # alejado en eje x
                                 ))

infl <- data.frame(data.frame(rendimiento=7500, # alejado en eje y
                              N=150 # alejado en eje x
                              ))
# figura 2.13
par(mfcol=c(1,1))
plot(dat.rend$N, dat.rend$rendimiento,
     xlab="N (kg/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(0,155), ylim=c(1000, 7550))
points(outlier$N, outlier$rendimiento,
       pch=16, cex=1.2, col="gray30") # agregamos la obs. atípica
points(g.lever$N, g.lever$rendimiento,
       pch=16, cex=1.2, col="green") # agregamos la obs. de gran leverage
points(infl$N, infl$rendimiento,
       pch=16, cex=1.2, col="red") # agregamos la obs. influyente

dat.outlier <- rbind(dat.rend[, c(2,5)], outlier) # incorporamos la obs. atípica
dat.outlier[106, ] # lo verificamos

##     rendimiento  N
## 106        7500 15

dat.glever <- rbind(dat.rend[, c(2,5)], g.lever) # incorporamos el g.lever
dat.glever[106, ] # lo verificamos

##     rendimiento   N
## 106        2430 150

dat.infl <- rbind(dat.rend[, c(2,5)], infl) # incorporamos la obs. influyente
dat.infl[106, ] # lo verificamos

##     rendimiento   N
## 106        7500 150

m.rls.outlier <- update(m.rls, data=dat.outlier)
m.rls.glever <- update(m.rls, data=dat.glever)
m.rls.infl <- update(m.rls, data=dat.infl)
?update

# agregamos las rectas ajustadas en la figura 2.13
abline(m.rls, lwd=3) # con la base de datos original
abline(m.rls.outlier, lty=5, col="gray30") # incluyendo la obs. atípica
abline(m.rls.glever, lty=5, col="green") # incluyendo la obs. de gran leverage
abline(m.rls.infl, lty=5, col="red") # incluyendo la obs. influyente
```

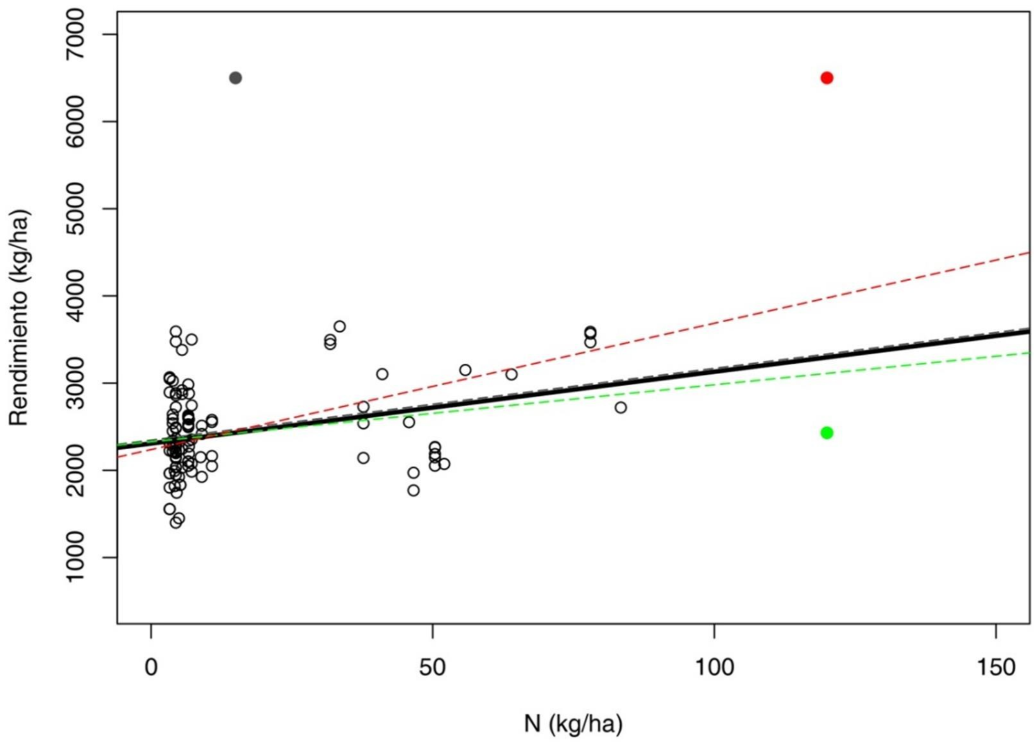

Figura 2. 13. Gráfico de dispersión entre rendimiento y nitrógeno
Nota. El gráfico incluye observaciones atípicas en el eje *y*, (punto gris), el eje *x* (punto verde) y en ambos ejes (punto rojo) y las rectas de regresión que resultan del modelo ajustado con la base de datos original (línea sólida) y de aquellos ajustados incluyendo las observaciones atípicas (en eje *y* la línea punteada gris; en eje *x* la línea punteada verde; y ambos ejes la línea punteada roja).

Con la función `compareCoefs()` del paquete `car` podemos comparar el cambio en los coeficientes estimados en forma directa:

```
compareCoefs(m.rls, m.rls.outlier, m.rls.glever, m.rls.infl)

## Calls:
## 1: lm(formula = rendimiento ~ N, data = dat.rend)
## 2: lm(formula = rendimiento ~ N, data = dat.outlier)
## 3: lm(formula = rendimiento ~ N, data = dat.glever)
## 4: lm(formula = rendimiento ~ N, data = dat.infl)
##
##             Model 1 Model 2 Model 3 Model 4
## (Intercept)  2307.9  2355.8  2337.6  2202.3
## SE             59.8    84.5    58.4    68.9
##
## N              8.23    8.23    5.77   16.99
## SE             2.38    3.38    2.02    2.38
##
```

Y gráficamente (figuras 2. 14, 2. 15 y 2. 16):

```
par(mfcol=c(1,2))
plot(m.rls.outlier, which = c(4,5)) # con la obs. atípica (figura 2.14)
```

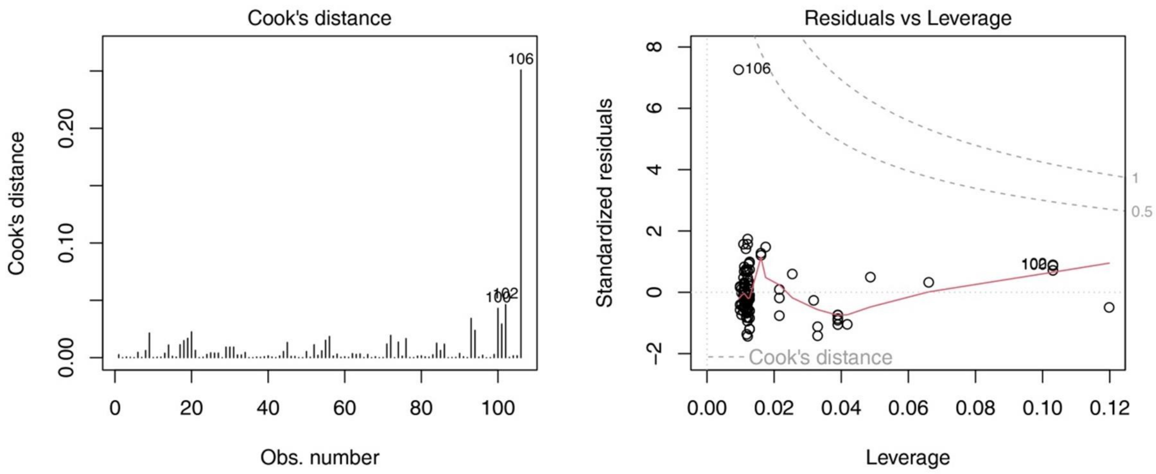

Figura 2. 14. Evaluación de observaciones atípicas y de influencia en base a la Distancia de Cook (izquierda) y el Leverage (derecha) para el conjunto de datos que incluye una observación atípica en el eje *y*

```
plot(m.rls.glever, which = c(4,5)) # con la obs. de gran leverage (figura 2.15)
```

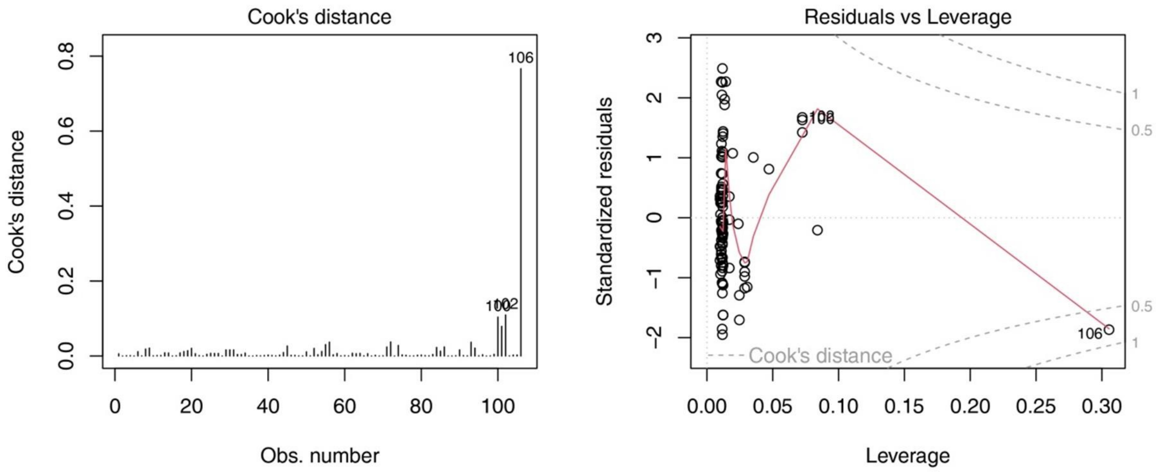

Figura 2. 15. Evaluación de observaciones atípicas y de influencia en base a la Distancia de Cook (izquierda) y el Leverage (derecha) para el conjunto de datos que incluye una observación atípica en el eje *x*

```
plot(m.rls.infl, which = c(4,5)) # con la obs. influyente (figura 2.16)
```

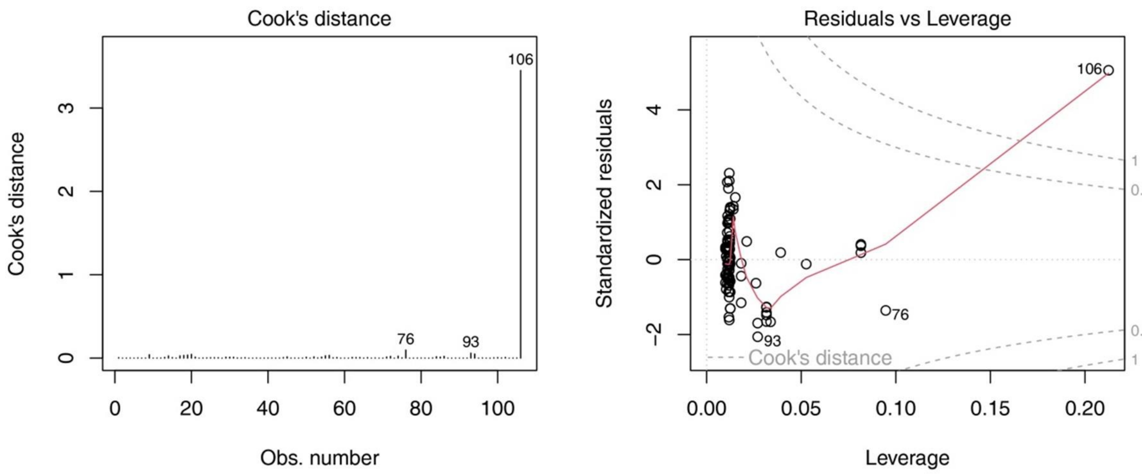

Figura 2. 16. Evaluación de observaciones atípicas y de influencia en base a la Distancia de Cook (izquierda) y el Leverage (derecha) para el conjunto de datos que incluye una observación atípica en ambos ejes

Claramente, la observación alejada en ambos ejes es la que impacta más en el ajuste del modelo. Si bien la observación atípica parece no influir en el ajuste del modelo, aumenta el error estándar residual y de esta forma también el error estándar de la pendiente. Esto resulta en un aumento en el valor *p* de la pendiente, lo cual puede modificar la significancia de la relación.

Capítulo 3. Modelo unifactorial

3. 1. Introducción

En este capítulo abordamos el modelo unifactorial. Al igual que la regresión lineal simple, se trata de un modelo con un único predictor. Sin embargo, en este caso el predictor es categórico y sus niveles determinan la parametrización del modelo. Este modelo nos servirá para introducir el análisis de la varianza o ANOVA (por sus siglas en inglés) y mostrar cómo llevarlo adelante en R. Para esto, el eje del capítulo será la función `anova()`. También, exploramos la distribución F, sumas de cuadrados, grados de libertad y cuadrados medios. Desde el punto de vista del muestreo, el modelo responde a un *diseño completamente aleatorizado* (DCA). Como aspecto importante, discutimos la parametrización de los modelos factoriales y los contrastes *a priori*, además de las comparaciones múltiples *a posteriori*. Asociado a esto, discutimos los problemas de multiplicidad y las diferentes alternativas para controlar el incremento en la tasa de error asociado a las pruebas simultáneas. Tratamos también la importancia del número de repeticiones para detectar los efectos que nos interesan.

3. 2. Problema

El problema consiste en evaluar las diferencias de rendimiento entre regiones. Existen factores bióticos y abióticos que condicionan el crecimiento y desarrollo de los cultivos, y por lo tanto el rendimiento. El territorio se separa espacialmente en regiones de acuerdo a la combinación de tales características. En consecuencia, se busca determinar si existen diferencias entre regiones, además de la magnitud de las mismas, ya que es clave para ajustar las prácticas de manejo.

3. 3. Datos

Cargamos los datos (recordar antes definir el directorio de trabajo).

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
```

En este problema pretendemos determinar si el rendimiento depende de la región, una variable categórica. Cuando el modelo incluye una variable categórica como predictor, a esta la denominamos *factor* y a sus categorías *niveles*. Exploramos la codificación de las variables en la base de datos:

```
str(dat.rend)

## 'data.frame':    105 obs. of  9 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 ...
##  $ borde      : num  30 15.3 34.8 41.2 54.5 ...
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 ...
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 ...
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 ...
##  $ fsiembra   : chr  "16/10/2018" "20/10/2018" "19/10/2018" "26/10/2018" ...
```

```
## $ cprevio    : chr "gram_anual" "gram_anual" "gram_anual" "gram_anual" ...
## $ region     : chr "ra" "ra" "ra" "ra" ...
```

Cuando una variable categórica no está codificada como `factor`, R no reconoce sus niveles:

```
levels(dat.rend$region)

## NULL
```

y se presentan problemas para su análisis. Entre otras cosas, no es posible realizar gráficos como el siguiente:

```
plot(dat.rend$rendimiento ~ dat.rend$region,
     xlab="Región", ylab="Rendimiento (kg/ha)")

##  Error: unexpected symbol in:

"plot(dat.rend$rendimiento ~ dat.rend$region,

     xlab="Región", ylab="Rendimiento"
```

Al realizar la coerción del vector `region` desde `chr` a `factor` podemos obtener la figura 3. 1:

```
dat.rend$region <- as.factor(dat.rend$region) # indicamos que region es un factor
levels(dat.rend$region) # niveles del factor

## [1] "ra" "rb" "rc" "rd"

plot(dat.rend$rendimiento ~ dat.rend$region,
     col=c("red", "green", "darkgrey", "yellow"),
     xlab="Región", ylab="Rendimiento (kg/ha)",
     ylim=c(500,4500)) # figura 3.1
```

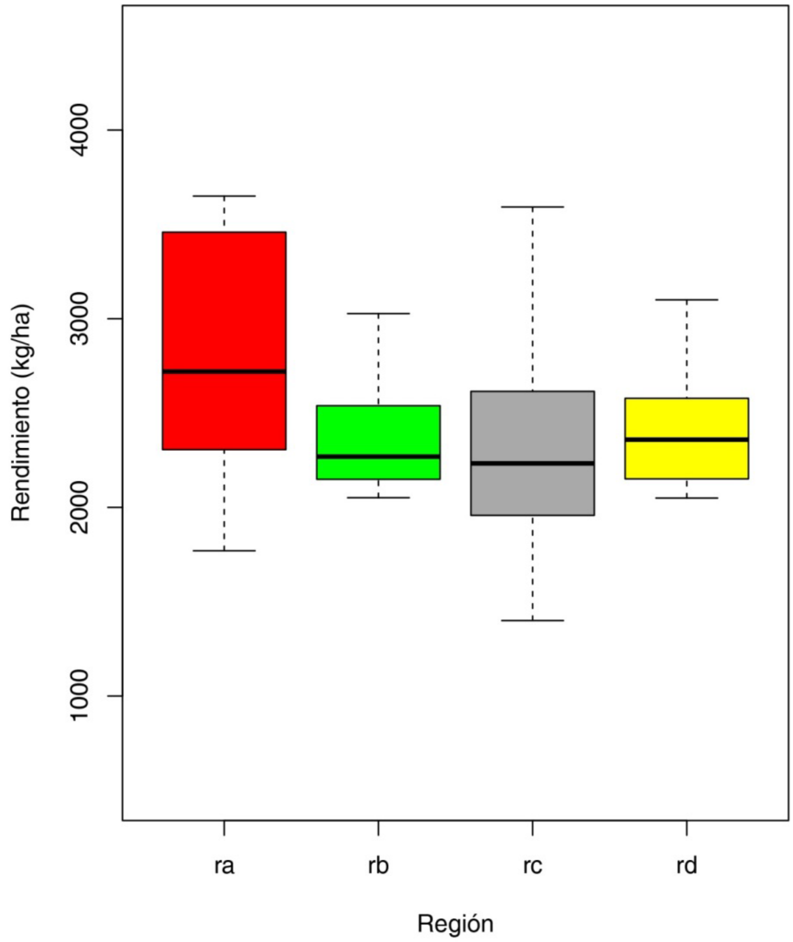

Figura 3. 1. Variabilidad del rendimiento de girasol dentro de las regiones (cada gráfico de caja y bigote) y entre regiones

La función `by()` es útil cuando trabajamos con factores, ya que permite ejecutar una función según sus niveles. En este caso, pedimos el resumen numérico de algunas variables para cada región:

```
by(dat.rend[, c(2:5)] , dat.rend$region, summary) # resumen numérico por región

## dat.rend$region: ra
##   rendimiento       borde          tamanio            N
##  Min.   :1770   Min.   : 6.55   Min.   :15.00   Min.   : 6.71
##  1st Qu.:2306   1st Qu.:22.16   1st Qu.:32.50   1st Qu.: 7.20
##  Median :2720   Median :32.96   Median :48.00   Median : 9.00
##  Mean   :2767   Mean   :35.52   Mean   :45.21   Mean   :30.17
##  3rd Qu.:3458   3rd Qu.:43.52   3rd Qu.:61.00   3rd Qu.:46.60
##  Max.   :3650   Max.   :63.28   Max.   :72.00   Max.   :83.40
## ------------------------------------------------------------
## dat.rend$region: rb
##   rendimiento       borde          tamanio            N
##  Min.   :2051   Min.   :22.03   Min.   : 9.00   Min.   : 3.85
##  1st Qu.:2149   1st Qu.:42.21   1st Qu.:46.00   1st Qu.: 6.60
##  Median :2269   Median :56.53   Median :60.00   Median : 6.60
##  Mean   :2374   Mean   :53.63   Mean   :60.07   Mean   :19.74
##  3rd Qu.:2539   3rd Qu.:60.84   3rd Qu.:73.00   3rd Qu.:37.70
##  Max.   :3027   Max.   :90.21   Max.   :98.00   Max.   :52.05
## ------------------------------------------------------------
```

```
## dat.rend$region: rc
##   rendimiento      borde           tamanio             N
##  Min.   :1400   Min.   : 21.57   Min.   :  6.00   Min.   :3.300
##  1st Qu.:1958   1st Qu.: 38.87   1st Qu.: 26.50   1st Qu.:3.300
##  Median :2233   Median : 54.27   Median : 40.00   Median :4.400
##  Mean   :2303   Mean   : 53.99   Mean   : 41.54   Mean   :4.195
##  3rd Qu.:2615   3rd Qu.: 70.71   3rd Qu.: 56.00   3rd Qu.:4.400
##  Max.   :3592   Max.   :102.24   Max.   :105.00   Max.   :5.500
## ------------------------------------------------------------
## dat.rend$region: rd
##   rendimiento      borde          tamanio            N
##  Min.   :2049   Min.   :11.48   Min.   : 3.50   Min.   : 8.80
##  1st Qu.:2154   1st Qu.:19.83   1st Qu.:16.80   1st Qu.:10.80
##  Median :2359   Median :23.31   Median :39.00   Median :10.80
##  Mean   :2433   Mean   :24.17   Mean   :39.98   Mean   :19.33
##  3rd Qu.:2572   3rd Qu.:30.70   3rd Qu.:57.75   3rd Qu.:10.80
##  Max.   :3100   Max.   :35.11   Max.   :85.00   Max.   :64.00
```

Debemos tener en cuenta que R ordena los niveles alfabéticamente y este orden es tomado por `lm()` que ajusta el modelo utilizando como referencia el primer nivel (ver luego en las secciones «Modelo y ajuste» y «Parametrización y contrastes»). Si por alguna razón quisiéramos cambiar la región de referencia y reordenar los niveles, podemos hacerlo desde el argumento `levels` de la función `factor()`. Por ejemplo, indicamos `"rc"` como la referencia y luego ordenamos como `"rb"`, `"rd"`, `"ra"`:

```
dat.rend$region <- with(dat.rend, factor(region, levels=c("rc", "rb", "rd", "ra")))
levels(dat.rend$region)

## [1] "rc" "rb" "rd" "ra"
```

El argumento `labels` nos permite renombrar los niveles según nuestra conveniencia:

```
dat.rend$region <- with(dat.rend, factor(region, labels=c("Región C",
                                                          "Región B",
                                                          "Región D",
                                                          "Región A")))
levels(dat.rend$region)

## [1] "Región C" "Región B" "Región D" "Región A"
```

Es pertinente notar que para reordenar los niveles desde el argumento `levels` sus nombres se indican tal como están guardados en el objeto, mientras que en `labels` los nombres de los niveles se reemplazan de acuerdo a su orden. Ambos elementos (nombres y orden) pueden extraerse desde la función `levels()`. Es decir, `levels` y, también, `labels` operan como argumentos de `factor()` (y de otras funciones), pero también son funciones en sí mismas. Cargando los datos y codificándolos nuevamente volvemos a la situación inicial:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
dat.rend$region <- as.factor(dat.rend$region)
levels(dat.rend$region)

## [1] "ra" "rb" "rc" "rd"
```

3. 4. Modelo y ajuste

Para estudiar cómo varía el rendimiento, una variable continua, en función de la región, una variable categórica o factor, en un libro de texto tradicional probablemente encontremos que el modelo estadístico se especifica como:

$$r_{ij} = \mu + \alpha_j + \varepsilon_{ij}$$
$$\varepsilon_{ij} \sim \mathcal{N}(0;\ \sigma^2)_{independientes}$$

en donde el rendimiento de girasol del *i*-ésimo lote de la *j*-ésima región (r_{ij}) es obtenido como el rendimiento medio general (μ), más el efecto de la región (α_j), más un término de error (ε_{ij}). Sin embargo, este modelo, a veces denominado *modelo de efectos*, está sobreparametrizado ya que al incluir la media general el efecto de la última región se obtiene por diferencia (en términos técnicos, el modelo no es estimable por MMCO). Una forma de resolver este problema es reemplazar μ por la media de uno de los niveles y expresar el resto de los niveles en términos de diferencias con respecto a esta media de referencia. Para el caso de la región, podremos plantear el modelo como:

$$r_i = \beta_0 + \beta_1 \times rb_i + \beta_2 \times rc_i + \beta_3 \times rd_i + \varepsilon_i$$
$$\varepsilon_i \sim \mathcal{N}(0;\ \sigma^2)_{independientes}$$

Asignando a las variables *rb*, *rc* y *rd* el valor de 1 si el lote *i* corresponde a la región en cuestión y 0 en caso contrario (se las llama variables *dummy* o binarias) de modo que:

r_i = rendimiento de girasol (kg ha^{-1}) del lote *i* (*i* = 1, 2, … N),
rb_i = 1 si el lote se localiza en la región «b» y 0 en caso contrario,
rc_i = 1 si el lote se localiza en la región «c» y 0 en caso contrario,
rd_i = 1 si el lote se localiza en la región «d» y 0 en caso contrario,
ε_i = término de error (kg ha^{-1}).

Al introducir el capítulo indicamos que el modelo responde a un diseño completamente aleatorizado (DCA). Esto quiere decir que no se imponen restricciones a la selección de las unidades experimentales (y a la asignación de tratamientos en el caso de los experimentos manipulativos). Es importante tener claro que estamos modelando el rendimiento en función de un único predictor (la región), pero que dada la naturaleza categórica del mismo recurrimos a un modelo con cinco parámetros: β_0, β_1, β_2, β_3 y σ^2. El intercepto (β_0) expresa el rendimiento promedio de los lotes de la región «a» (μ_{ra}), y el resto de los parámetros del componente lineal (β_1, β_2, β_3) representan diferencias promedio entre los rendimientos de cada una de las regiones y la región de referencia:

$\beta_1 = \mu_{rb} - \mu_{ra}$,
$\beta_2 = \mu_{rc} - \mu_{ra}$,
$\beta_3 = \mu_{rd} - \mu_{ra}$.

Consecuentemente:

$\beta_0 + \beta_1 = \mu_{rb}$ = rendimiento de girasol promedio en la región «b» (kg ha^{-1}),
$\beta_0 + \beta_2 = \mu_{rc}$ = rendimiento de girasol promedio en la región «c» (kg ha^{-1}),
$\beta_0 + \beta_3 = \mu_{rd}$ = rendimiento de girasol promedio en la región «d» (kg ha^{-1}).

La varianza (σ^2) cuantifica la variabilidad de rendimiento entre lotes de una misma región. Lo que acabamos de explicar es el modelo que `lm` estima por defecto al expresar el rendimiento en función de la región:

```
m.unif <- lm(rendimiento ~ region, data=dat.rend) # ajuste del modelo
m.unif # modelo ajustado

##
## Call:
## lm(formula = rendimiento ~ region, data = dat.rend)
##
## Coefficients:
## (Intercept)     regionrb     regionrc     regionrd
##      2766.9       -392.8       -463.8       -334.2

summary(m.unif)

##
## Call:
## lm(formula = rendimiento ~ region, data = dat.rend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -996.91 -310.15  -70.15  255.93 1288.85
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   2766.9      102.1  27.108  < 2e-16 ***
## regionrb      -392.8      136.7  -2.874 0.004939 **
## regionrc      -463.8      124.6  -3.723 0.000324 ***
## regionrd      -334.2      224.4  -1.490 0.139470
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 489.5 on 101 degrees of freedom
## Multiple R-squared:  0.125,  Adjusted R-squared:  0.09901
## F-statistic:  4.81 on 3 and 101 DF,  p-value: 0.003582

confint(m.unif) # IC de las estimaciones

##                 2.5 %    97.5 %
## (Intercept) 2564.4326 2969.3935
## regionrb    -663.9792 -121.7090
## regionrc    -710.8700 -216.6582
## regionrd    -779.3965  110.9037
```

Es oportuno mencionar que hemos estimado cuatro coeficientes para la componente determinística, pero en `lm` solo tuvimos que indicar el factor (`rendimiento ~ region`). Esto se debe a que R reconoce que `region` es un `factor` de cuatro niveles. Con los valores que obtenemos de la función `summary()` aplicada al modelo informamos la ecuación estimada:

$$\hat{r} = 2766{,}9 - 392{,}8 \times rb - 463{,}8 \times rc - 334{,}2 \times rd$$

a partir de la cual podemos predecir de manera puntual el rendimiento de cada región:

```
b0 <- m.unif$coefficients[1] # estimación puntual de β0
b1 <- m.unif$coefficients[2] # estimación puntual de β1
b2 <- m.unif$coefficients[3] # estimación puntual de β2
b3 <- m.unif$coefficients[4] # estimación puntual de β3

b0 + b1*0 + b2*0 + b3*0 # predicción del rendimiento medio en "ra"

## (Intercept)
##    2766.913

b0 + b1*1 + b2*0 + b3*0 # predicción del rendimiento medio en "rb"

## (Intercept)
##    2374.069

b0 + b1*0 + b2*1 + b3*0 # predicción del rendimiento medio en "rc"

## (Intercept)
##    2303.149

b0 + b1*0 + b2*0 + b3*1 # predicción del rendimiento medio en "rd"

## (Intercept)
##    2432.667
```

Como vimos en el capítulo 1, podemos usar directamente la función `predict.lm()` y así obtener los intervalos de confianza y predicción:

```
regiones <- data.frame(region=c("ra", "rb", "rc", "rd"))
predict.lm(m.unif, newdata=regiones, interval="confidence") # IC

##        fit      lwr      upr
## 1 2766.913 2564.433 2969.393
## 2 2374.069 2193.747 2554.391
## 3 2303.149 2161.505 2444.793
## 4 2432.667 2036.232 2829.101

predict.lm(m.unif, newdata=regiones, interval="prediction") # IP

##        fit      lwr      upr
## 1 2766.913 1774.965 3758.861
## 2 2374.069 1386.406 3361.732
## 3 2303.149 1321.811 3284.487
## 4 2432.667 1383.800 3481.534
```

Recordemos que si usamos la función `predict.lm()` omitiendo el argumento `newdata` obtenemos las predicciones para cada uno de los lotes de la muestra:

```
head(predict.lm(m.unif), 10) # predicciones para los lotes de la muestra

##        1        2        3        4        5        6        7        8
## 2766.913 2766.913 2766.913 2766.913 2374.069 2766.913 2766.913 2766.913
```

```
##        9       10 
## 2766.913 2374.069
```

¿Cómo R ha hecho esto? Simplemente creando una matriz con los valores de la codificación *dummy* para cada lote, la cual se denomina *matriz de diseño*, y multiplicándola por el vector de coeficientes estimados:

$$\hat{Y} = X \times \hat{b}$$

Donde

$\hat{Y}$ = vector de rendimientos estimados (de dimensión 105 × 1),
X = matriz de diseño (105 × 4),
$\hat{b}$ = vector de coeficientes estimados (4 × 1),

En el capítulo 7 veremos con mayor detalle este procedimiento. En R podemos acceder a la matriz de diseño mediante la función `model.matrix()`:

```
X <- with(dat.rend, model.matrix(~ region)) # matriz de diseño
head(X, 10)

##    (Intercept) regionrb regionrc regionrd
## 1            1        0        0        0
## 2            1        0        0        0
## 3            1        0        0        0
## 4            1        0        0        0
## 5            1        1        0        0
## 6            1        0        0        0
## 7            1        0        0        0
## 8            1        0        0        0
## 9            1        0        0        0
## 10           1        1        0        0
```

Cada fila de la matriz es un lote y cada columna indica la codificación que se le asignó, incluyendo una columna de unos (1) para el intercepto. Agregamos un vector con el nombre de las regiones y maquillamos la matriz para mostrar la información de manera más clara:

```
Xtun <- data.frame(dat.rend$region, X)
colnames(Xtun) <- c("region", "B0", "rb","rc", "rd")
head(Xtun, 10)

##    region B0 rb rc rd
## 1      ra  1  0  0  0
## 2      ra  1  0  0  0
## 3      ra  1  0  0  0
## 4      ra  1  0  0  0
## 5      rb  1  1  0  0
## 6      ra  1  0  0  0
## 7      ra  1  0  0  0
## 8      ra  1  0  0  0
## 9      ra  1  0  0  0
## 10     rb  1  1  0  0
```

Extraemos el vector de coeficientes estimados:

```
b <- coef(m.unif) # vector de coeficientes estimados
```

y realizamos la operación matricial para predecir el rendimiento de cada lote (para visualizarlo combinamos el resultado con la matriz):

```
Y <- X %*% b # vector de predicciones
head(data.frame(Xtun, "rend"=Y), 10)

##    region B0 rb rc rd     rend
## 1      ra  1  0  0  0 2766.913
## 2      ra  1  0  0  0 2766.913
## 3      ra  1  0  0  0 2766.913
## 4      ra  1  0  0  0 2766.913
## 5      rb  1  1  0  0 2374.069
## 6      ra  1  0  0  0 2766.913
## 7      ra  1  0  0  0 2766.913
## 8      ra  1  0  0  0 2766.913
## 9      ra  1  0  0  0 2766.913
## 10     rb  1  1  0  0 2374.069
```

Como vemos, los rendimientos predichos son los mismos que los obtenidos con `predict.lm`. Veamos la diferencia de esta matriz con la matriz de diseño del modelo de regresión lineal simple del capítulo 1. Al utilizar un único predictor, y este ser cuantitativo, la matriz de diseño resultante tiene dos columnas, una para el intercepto y otra para el nitrógeno:

```
X.rls <- with(dat.rend, model.matrix(~ N)) # matriz de diseño rls (rendimiento ~ N)
head(X.rls, 10)

##    (Intercept)     N
## 1            1  6.71
## 2            1  6.71
## 3            1  6.71
## 4            1  6.71
## 5            1  4.40
## 6            1 41.05
## 7            1  6.71
## 8            1 55.80
## 9            1 46.60
## 10           1  6.60
```

3. 5. ANOVA

La pregunta de nuestro problema implica determinar si existen diferencias de rendimiento entre regiones y para responderla podemos aplicar un ANOVA. Este es un análisis inferencial que se basa en la partición de la varianza, aunque lo que evalúa es la diferencia de medias bajo la hipótesis nula de:

H_0: $\mu_{ra} = \mu_{rb} = \mu_{rc} = \mu_{rd}$

la cual también podemos expresar mediante los parámetros del modelo:

$$H_0: \quad \beta_1 = \beta_2 = \beta_3 = 0$$

Al plantear más de una igualdad en una única hipótesis nula, en el ANOVA las medias de los tratamientos se comparan en forma simultánea. La hipótesis alternativa es que al menos una de las medias difiere (al menos uno de los coeficientes es distinto de cero). Al particionar de la varianza del rendimiento en dos fuentes de variación, la que se debe a la región y la que se debe al error aleatorio, se obtienen dos varianzas y su cociente es usado como un estadístico de prueba para decidir si rechazar o no la hipótesis nula. A estas varianzas se les suele denominar, respectivamente, *cuadrado medio debido a los tratamientos* (CMTr), las regiones en este caso, y *cuadrado medio debido al error* (CME). El CMTr se calcula como:

$$\mathrm{CMTr} = \frac{\sum_{j=1}^{p} n_j (\hat{r}_j - \bar{\bar{r}})^2}{p - 1}$$

donde $\hat{r}_j$ y n_j indican, respectivamente, el rendimiento de girasol promedio y el número de lotes en la j-esima región, $\bar{\bar{r}}$ es el rendimiento de girasol promedio general, y p es el número de regiones. El CME se presentó en el capítulo 1 (sección *1.4.a*) y en el contexto del DCA puede expresarse como:

$$\mathrm{CME} = \frac{\sum_{j=1}^{p} (n_j - 1) s_j^2}{n - p}$$

donde s_j es la varianza del rendimiento de girasol de los lotes de la j-esima región. Al dividir ambos cuadrados medios obtenemos el estadístico F:

$$\mathrm{F} = \frac{\mathrm{CMTr}}{\mathrm{CME}}$$

que representa la variabilidad en el rendimiento de girasol entre regiones relativa a la variabilidad entre lotes de una misma región.

La lógica detrás de este test es que si los lotes muestreados provinieran de una misma población estadística, es decir, si no hubiera diferencia entre regiones, las medias de las regiones serían parecidas y sus diferencias solo serían debidas al proceso de muestreo. Se demuestra que cuano H_0 es verdadera, ambas varianzas (CMTr y CME) son estimadores insesgados de σ^2 (varianza residual del modelo). Entonces, dado que bajo H_0 el valor esperado de CMTr es el mismo que el valor esperado de CME, el valor esperado de F es igual a 1. Dicho de otro modo, de ser cierta H_0, la variabilidad entre regiones resultará similar a la variabilidad entre lotes de una misma región. Ahora bien, en el caso de que H_0 fuera falsa, CMTr sobreestimaría σ^2. En cambio, el CME es un estimador insesgado de σ^2 independientemente de que H_0 sea cierta o no. Esto quiere decir que altos valores de F aportan evidencia de que el rendimiento de girasol varía entre regiones (el CMTr es el numerador de F). Cuando se cumplen los supuestos del modelo, la distribución por muestreo del estadístico F es una distribución F definida por dos parámetros: los grados de libertad del tratamiento (p - 1) y los grados de libertad del error (n - p). Por lo tanto, utilizamos esta distribución para obtener el valor p desde el cual decidimos si el valor de F que observamos en la muestra es lo suficientemente alto como para rechazar H_0.

La función `anova()` ejecutada sobre el modelo ajustado nos devuelve la información de una típica tabla de ANOVA:

```
anova(m.unif) # Tabla de ANOVA

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## region      3  3457476 1152492  4.8096 0.003582 **
## Residuals 101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Los mismos resultados obtenemos usando la función `aov()`:

```
summary(aov(m.unif)) # similar a anova()

##              Df   Sum Sq Mean Sq F value  Pr(>F)
## region        3  3457476 1152492    4.81 0.00358 **
## Residuals   101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La diferencia entre esta función y `anova()` es que `aov()` permite ajustar modelos (para lo cual llama a `lm()`):

```
aov(rendimiento ~ region, data=dat.rend)

## Call:
##    aov(formula = rendimiento ~ region, data = dat.rend)
##
## Terms:
##                   region Residuals
## Sum of Squares   3457476  24201921
## Deg. of Freedom        3       101
##
## Residual standard error: 489.513
## Estimated effects may be unbalanced

aov(rendimiento ~ region, data=dat.rend)$coefficients

## (Intercept)    regionrb    regionrc    regionrd
##   2766.9130   -392.8441   -463.7641   -334.2464
```

y extender el análisis:

```
summary(aov(rendimiento ~ region, data=dat.rend))

##              Df   Sum Sq Mean Sq F value  Pr(>F)
## region        3  3457476 1152492    4.81 0.00358 **
## Residuals   101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

mientras que `anova()` solo analiza el resultado con la información del modelo ya ajustado. Volvemos a la salida de la función `anova()`. En esta se informa que la variable de análisis es el rendimiento (`Response: rendimiento`), cuya suma de cuadrados se divide entre aquella debida a la región (`Sum Sq` de la fila `region`) y la que se debe al error (fila `Residuals`). Estas cantidades divididas por sus respectivos grados de libertad (`Df`) resultan en el CMTr (`Mean Sq` de la fila `region`) y el CME (`Mean Sq` de la fila `Residuals`):

```
(cm1 <- anova(m.unif)[1,2]/anova(m.unif)[1,1]) # SCTr/gl = CMTr

## [1] 1152492

(cm2 <- anova(m.unif)[2,2]/anova(m.unif)[2,1]) # SCE/gl = CME

## [1] 239623
```

Al dividir los cuadrados medios obtenemos el valor de F (`F value`) y con este, el valor *p* (`Pr(>F)`). El valor de F observado es 4,8:

```
cm1/cm2 # estadístico F calculado dividiendo los cuadrados medios

## [1] 4.809606

summary(m.unif)$fstatistic # estadístico F en el summary

##      value      numdf      dendf
##   4.809606   3.000000 101.000000

anova(m.unif)[1,4] # estadístico F en la tabla de ANOVA

## [1] 4.809606
```

y tiene un valor *p* de 0.0035, sugiriendo que el rendimiento de girasol difiere entre regiones:

```
anova(m.unif)[1,c(4,5)] # valor p desde la tabla de ANOVA

##        F value   Pr(>F)
## region  4.8096 0.003582 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Podemos obtener el valor *p* calculando la probabilidad de un F mayor a 4,8 en una distribución F con 4 – 1 = 3 grados de libertad en el numerador (lo indicamos desde el argumento `df1`) y 105 – 4 = 101 grados de libertad en el denominador (`df2`):

```
pf(q=4.809606, df1=3, df2=101, lower.tail=FALSE) # valor p desde la distribución F

## [1] 0.003581885
```

Concluimos entonces que la región explica parte del rendimiento del cultivo.

3. 6. Parametrización y contrastes *a priori*

Al plantear el modelo vimos una forma de predecir el rendimiento en función de la región, un factor de cuatro niveles. Sin embargo, esa no era la única parametrización posible. De modo general, un factor de *p* niveles es introducido a un MLG desagregado sus niveles en *p* - 1 variables cuyos parámetros representan relaciones entre las medias. Siguiendo algunas reglas, podemos parametrizar el modelo de modo de reflejar una hipótesis de trabajo, es decir, ideas *a priori*. Esto se asocia a los denominados *contrastes a priori* o planeados,[1] los cuales permiten comparar relaciones entre medias a partir de *p* - 1 pruebas, planteadas antes de ver los datos. La aplicación de contrastes en los MLG se basa en dos matrices:

- una matriz en la que se hacen explícitos los contrastes (**C**),
- una matriz en la que codifican los niveles del factor (**S**).

Desarrollamos estos conceptos a partir de la codificación *dummy* que R utiliza por defecto y luego los aplicamos a otras codificaciones del paquete base, incluyendo una alternativa definida por el usuario.

3. 6. 1. Parametrización de un modelo factorial

En los experimentos manipulativos es natural que de antemano estemos interesados en realizar comparaciones contra un control. En el caso de un factor con cuatro niveles en el cual uno de ellos es el control (c, t_1, t_2, t_3), los tres (4 - 1) contrastes de interés serían los siguientes:

$$\mu_{t_1} = \mu_c$$
$$\mu_{t_2} = \mu_c$$
$$\mu_{t_3} = \mu_c$$

Si estas igualdades se cumplen,[2] quiere decir que los niveles t_1, t_2 y t_3 no difieren del control. La discrepancia entre cada nivel t y el control puede cuantificarse como una

[1] También se los suele llamar *contrastes preplaneados*.
[2] El contraste de hipótesis desarrollado bajo la inferencia frecuentista asigna la igualdad a la hipótesis nula.

diferencia de medias, es decir, contrastando su media contra el control. Expresamos estas diferencias en términos de parámetros poblacionales:

$$\mu_{t_1} - \mu_c = \beta_1$$
$$\mu_{t_2} - \mu_c = \beta_2$$
$$\mu_{t_3} - \mu_c = \beta_3$$

Bajo la hipótesis nula de que no existen diferencias entre los tratamientos y el control, los parámetros son iguales a cero. Por conveniencia, incluimos un parámetro para el control:

$$\mu_c = \beta_0$$

Técnicamente, los contrastes son sumas ponderadas (combinaciones lineales) de grupos de medias. Si ordenamos los niveles como c, t_1, t_2, t_3, los pesos asignados a las medias de los tres contrastes (más el del control, en la primera fila) son los siguientes:

$$\beta_0 = 1 \times \mu_c + 0 \times \mu_{t_1} + 0 \times \mu_{t2} + 0 \times \mu_{t3}$$
$$\beta_1 = -1 \times \mu_c + 1 \times \mu_{t_1} + 0 \times \mu_{t2} + 0 \times \mu_{t3}$$
$$\beta_2 = -1 \times \mu_c + 0 \times \mu_{t_1} + 1 \times \mu_{t2} + 0 \times \mu_{t3}$$
$$\beta_3 = -1 \times \mu_c + 0 \times \mu_{t_1} + 0 \times \mu_{t2} + 1 \times \mu_{t3}$$

Podemos resumir estos pesos en una matriz C, que llamaremos *matriz de coeficientes de contraste*, en la cual cada fila (i) representa un contraste. Incluso más, podemos expresar el sistema de ecuaciones anterior como una operación matricial:

$$\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} \mu_c \\ \mu_{t_1} \\ \mu_{t_2} \\ \mu_{t_3} \end{bmatrix}$$
$$\mathbf{b} = \mathbf{C} \times \mathbf{Y}$$

donde

b = vector de parámetros,
C = matriz de coeficientes de contraste,
Y = vector de medias.

Los tres contrastes de interés se encuentran en las filas 2, 3 y 4 de C. Como vemos, en cada una de estas tres filas los coeficientes (c_j) suman cero, lo cual es una restricción necesaria en la especificación de un contraste ($\sum c_j = 0$). Es importante que los contrastes sean independientes de modo que no sean redundantes,[3] es decir, que cada uno capture una porción única de la variabilidad explicada por el factor. La independencia entre contrastes se verifica a través de su ortogonalidad. Dos contrastes $c1$ y $c2$ son ortogonales cuando la sumatoria del producto de sus coeficientes es nula ($\sum c1_j \times c2_j = 0$). En otras palabras, cuando las filas de C no están correlacionadas.

Ahora exploramos el vínculo entre C y el modelo parametrizado de acuerdo a diferencias con el control. Para esto, tengamos presente que los modelos lineales también son combinaciones lineales, en este caso de variables independientes: $y = \beta_0 + \sum \beta_j x_j$ (en este contexto, los coeficientes β pueden interpretarse como el peso o ponderación dado a cada variable). Habiendo definido los parámetros de acuerdo a los contrastes, podemos plantear el siguiente modelo:

$$y = \beta_0 + \beta_1 x_1 + \beta_1 x_2 + \beta_3 x_3$$

o, en términos de diferencia de medias:

$$y = \mu_c + (\mu_{t_1} - \mu_c)\, x_1 + (\mu_{t_2} - \mu_c)\, x_2 + (\mu_{t_3} - \mu_c)\, x_3$$

Lo que nos resta es asignar una codificación numérica a las variables (x_1, x_2 y x_3) para que el modelo prediga la media de cada nivel del factor. En este caso, es intuitivo aplicar una codificación *dummy* de modo de mantener ($x = 1$) el término que nos interesa y remover ($x = 0$) el resto:

$$\begin{aligned}
\mu_c &= \beta_0 + \beta_1 \times 0 + \beta_2 \times 0 + \beta_3 \times 0 \\
\mu_{t_1} &= \beta_0 + \beta_1 \times 1 + \beta_2 \times 0 + \beta_3 \times 0 \\
\mu_{t_2} &= \beta_0 + \beta_1 \times 0 + \beta_2 \times 1 + \beta_3 \times 0 \\
\mu_{t_3} &= \beta_0 + \beta_1 \times 0 + \beta_2 \times 0 + \beta_3 \times 1
\end{aligned}$$

De la misma manera que hicimos antes, podemos resumir la codificación de las variables en una matriz S, que denominaremos *matriz de codificación*, en la cual cada fila represente un nivel del factor, y expresar el sistema de ecuaciones como una operación matricial:

$$\underset{\mathbf{Y}}{\begin{bmatrix} \mu_{ra} \\ \mu_{rb} \\ \mu_{rc} \\ \mu_{rd} \end{bmatrix}} = \underset{\mathbf{S}}{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}} \times \underset{\mathbf{b}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \beta_3 \end{bmatrix}}$$

[3] Independientes implica que las comparaciones entre niveles no se repiten entre pruebas. Para ejemplificarlo, si tenemos tres grupos *a*, *b* y *c*, las comparaciones $a \times b$ y $a \times c$ contienen implícitamente $b \times c$. En otras palabras, $b \times c$ no es estadísticamente independiente de las otras dos comparaciones porque está incluido implícitamente en ellas.

donde

Y = vector de medias,
S = matriz de codificación,
b = vector de parámetros.

La columna *i* de una matriz de codificación especifica los valores que asume la variable introducida para cada uno de los *k* niveles del factor (filas), de manera que el resultado de multiplicar una fila *j* por el vector de parámetros es la media del grupo (μ_j). Resumiendo:

$$\mathbf{b} = \mathbf{C} \times \mathbf{Y}$$
$$\mathbf{Y} = \mathbf{S} \times \mathbf{b}$$

De ambas igualdades se deduce que existe una relación directa entre C y S. Concretamente, la inversa de la matriz de contrastes es la matriz de codificación, y la inversa de la matriz de codificación es la matriz de contrastes:

$$\mathbf{C} = \mathbf{S}^{-1}$$
$$\mathbf{S} = \mathbf{C}^{-1}$$

Trabajemos estos conceptos en R. En primer lugar, para pedir la codificación de un factor ejecutamos la función `contrasts()`:

```
contrasts(dat.rend$region)

##    rb rc rd
## ra  0  0  0
## rb  1  0  0
## rc  0  1  0
## rd  0  0  1
```

La salida muestra la codificación de `region`, un factor de cuatro niveles, y responde a lo que en R se conoce como *contrastes por tratamiento*, que no es otra cosa que el uso de un nivel de referencia contra el cual comparar el resto de los niveles. La función `contr.treatment()` crea una matriz con esta codificación. Si en esta indicamos el número de niveles de `region`, obtenemos la misma matriz (en este caso los nombres de los niveles en filas y columnas son reemplazados por números), con lo cual constatamos que R codifica por defecto de acuerdo a este sistema:

```
contr.treatment(n=4)

##   2 3 4
## 1 0 0 0
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1
```

Ambas son matrices de codificación (S) excepto por la columna del intercepto. Para seguir con el desarrollo, incluimos la columna del intercepto y guardamos la matriz en un objeto:[4]

```
S <- cbind(1, contr.treatment(n=4)) # incluimos el intercepto
colnames(S) <- c("B0", "x1", "x2", "x3") # asignamos nombres a las columnas
rownames(S) <- levels(dat.rend$region) # asignamos nombres a las filas
S

##    B0 x1 x2 x3
## ra  1  0  0  0
## rb  1  1  0  0
## rc  1  0  1  0
## rd  1  0  0  1
```

Al multiplicar esta matriz por el vector de coeficientes estimados con `lm()`, que hemos guardado en el objeto `b`, obtenemos el vector de medias o predicciones ($\mathbf{Y} = \mathbf{S} \times \mathbf{b}$):

```
Yd <- S %*% b # rendimiento promedio en cada región (predicción)
colnames(Yd) <- "rend. medio" # asignamos nombre a la columna (es un vector)
round(Yd, 1)

##    rend. medio
## ra      2766.9
## rb      2374.1
## rc      2303.1
## rd      2432.7
```

Como podemos verificar, hemos obtenido las medias de cada región:

```
medias <- tapply(dat.rend$rendimiento, dat.rend$region, mean) # medias por región
round(medias, 1)

##     ra     rb     rc     rd
## 2766.9 2374.1 2303.1 2432.7
```

y los mismos valores que devuelve `predict.lm()`:

```
round(predict.lm(m.unif, regiones), 1) # predicción del modelo para cada región

##      1      2      3      4
## 2766.9 2374.1 2303.1 2432.7
```

La matriz C de coeficientes de contraste se obtiene invirtiendo la matriz S:

```
C <- solve(S) # matriz de coeficientes de contraste (inversa de S)
rownames(C) <- c("B0", "c1", "c2", "c3") # renombramos las filas de la matriz
C
```

[4] Las matrices que devuelven `contrast()` y las funciones asociadas a los contrastes predefinidos, como `contr.treatment()`, no incluyen el intercepto. La matriz que resulta de sumar esta columna (es decir, la que presentamos como S) suele denominarse *matriz aumentada* debido a la operación algebraica.

```
##    ra rb rc rd
## B0  1  0  0  0
## c1 -1  1  0  0
## c2 -1  0  1  0
## c3 -1  0  0  1
```

Ahora verificamos que al multiplicar C por el vector de predicciones (lo guardamos en el objeto `Yd`) obtenemos un vector con los parámetros estimados:

```
vb <- C %*% Yd # vector de coeficientes estimados
colnames(vb) <- "est." # renombramos la columna del vector
rownames(vb) <- c("b0", "b1", "b2", "b3") # nombramos sus elementos (filas)
round(vb, 1)

##      est.
## b0 2766.9
## b1 -392.8
## b2 -463.8
## b3 -334.2

round(b, 1) # comparamos con los coeficientes estimados con lm()

## (Intercept)    regionrb    regionrc    regionrd
##      2766.9      -392.8      -463.8      -334.2
```

A continuación, resumimos lo que hemos realizado en R:

- exploramos la codificación del factor e incluimos la columna del intercepto para generar la matriz S,
- con esta matriz y el vector de coeficientes (usamos los coeficientes del modelo ajustado) obtuvimos las medias estimadas de los niveles del factor ($\mathbf{\widehat{Y}} = \mathbf{S} \times \mathbf{\hat{b}}$),
- invertimos S y obtuvimos la matriz de contrastes ($\mathbf{S}^{-1} = \mathbf{C}$),
- a partir de esta matriz y el vector de medias obtuvimos el vector de coeficientes ($\mathbf{\hat{b}} = \mathbf{C} \times \mathbf{\widehat{Y}}$).

Lo anterior nos ha servido para mostrar la relación entre las matrices de contraste y codificación, entre sí y con los coeficientes y predicciones del modelo. En la práctica, sin embargo, lo que hacemos es ajustar el modelo con la matriz de codificación asociada a los contrastes que deseamos poner a prueba. De manera general, debemos:

- definir los contrastes y su matriz C,
- invertir C para obtener S,
- ajustar el modelo aplicando la codificación de S,
- calcular los valores *p* de los coeficientes estimados ($\mathbf{\hat{b}}$) y concluir sobre los contrastes.

No obstante, R posee una serie de sistemas de codificación predefinidos asociados a contrastes comúnmente usados, lo cual nos ahorra la elaboración de C y S. Entre estos se encuentra el sistema *dummy* de los contrastes por tratamiento que utilizamos para la explicación y con el cual ajustamos (por defecto) el modelo `m.unif`. Cada contraste se pone a prueba a partir del valor *p* del estadístico muestral que los refleja (el coeficiente estimado). Individualmente, estos siguen una distribución *t* con *n* - *k* (105 - 4 = 101 en nuestro modelo) grados de libertad. Por lo tanto, podemos calcular la probabilidad con la función `pt()` y multiplicarla por 2 en el caso de las pruebas a dos colas:

```
n <- length(dat.rend$rendimiento) # nº de lotes (tamaño de la muestra)
n

## [1] 105

p <- 4 # nº de niveles del factor región = número de parámetros (b0, b1, b2, b3)
gl <- n - p # grados de libertad de las distribuciones t usadas en cada contraste
ee.b <- summary(m.unif)$coeff[,2] # errores estándar de los coeficientes
valorp <- pt(q=abs(vb)/ee.b, df=gl, lower.tail=FALSE) * 2 # cálculo de los valores p
colnames(valorp) <- "valor p"
round(valorp, 4)

##    valor p
## b0  0.0000
## b1  0.0049
## b2  0.0003
## b3  0.1395

round(valorp[2:4], 4) # valores p de los tres contrastes de interés

## [1] 0.0049 0.0003 0.1395
```

Es necesario notar que para calcular los valores *t* usamos los errores estándar (`ee.b`) y la información del modelo ajustado (para extraerla aplicamos la función `summary()`). En el capítulo 7 vemos cómo podemos obtener analíticamente estas cantidades. Como podemos verificar, los valores *p* que obtuvimos son los mismos que se informan en el resumen del modelo:

```
round(summary(m.unif)$coef[, 4], 4) # valores p informados en el resumen del ajuste

## (Intercept)    regionrb    regionrc    regionrd
##      0.0000      0.0049      0.0003      0.1395
```

Ahora bien, debemos tener presente que hemos puesto a prueba tres contrastes (o cuatro si incluimos el del intercepto) en simultáneo y que esto incrementa la probabilidad conjunta de cometer un error de tipo I, es decir, de rechazar incorrectamente al menos una de las hipótesis nulas (veremos este concepto con más detalle en la sección «Comparaciones múltiples a posteriori»). A este asunto se lo conoce como *problema de multiplicidad* y existen varios métodos para abordarlo, todos ellos basados en la corrección de los valores *p* de los contrastes individuales de manera de controlar la probabilidad de error de tipo I del estudio. El paquete `multcomp` implementa muchos de los métodos disponibles para el ajuste de los valores *p* en contrastes *a priori* (y en comparaciones múltiples en general). La principal función de este paquete es `glth()` y se la puede ejecutar directamente sobre el modelo ajustado:

```
library(multcomp) # instalar antes el paquete
summary(glht(m.unif))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend)
##
## Linear Hypotheses:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept) == 0   2766.9      102.1  27.108  < 0.001 ***
## regionrb == 0      -392.8      136.7  -2.874  0.01617 *
## regionrc == 0      -463.8      124.6  -3.723  0.00122 **
## regionrd == 0      -334.2      224.4  -1.490  0.35047
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

En esta salida los valores *p* difieren de los que se obtienen desde la función `summary()` ya que ahora han sido ajustados considerando una distribución *t* multivariada. Como hemos ingresado un objeto de clase `lm`, la matriz de contrastes que toma `glth()` es la del modelo ajustado. No obstante, podemos indicarle las hipótesis desde una matriz C (de contrastes), para lo cual necesitamos ajustar un modelo sin intercepto. Esto se logra restando 1 en la fórmula de `lm()`, lo cual resulta en un modelo cuyos coeficientes son las medias de cada nivel (ver luego «Modelo de medias»):

```
m.medias <- lm(rendimiento ~ region - 1, data=dat.rend)
coef(m.medias)
```

```
## regionra regionrb regionrc regionrd
## 2766.913 2374.069 2303.149 2432.667
```

```
medias
```

```
##       ra       rb       rc       rd
## 2766.913 2374.069 2303.149 2432.667
```

Para indicar los contrastes desde una matriz usamos el argumento `linfct`:

```
summary(glht(m.medias, linfct=C))
```

```
##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region - 1, data = dat.rend)
##
## Linear Hypotheses:
##         Estimate Std. Error t value Pr(>|t|)
## B0 == 0   2766.9      102.1  27.108  < 0.001 ***
## c1 == 0   -392.8      136.7  -2.874  0.01596 *
## c2 == 0   -463.8      124.6  -3.723  0.00117 **
## c3 == 0   -334.2      224.4  -1.490  0.35052
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Los valores p son similares a los que obtuvimos pasando directamente el modelo `m.unif`. Dado que la corrección de los valores p es mayor cuando se incrementa el número de contrastes, lo que disminuye la potencia de cada test, deberíamos remover el contraste del intercepto ya que no es de interés. Usar la matriz de contrastes en lugar del modelo ajustado facilita esta operación:

```
summary(glht(m.medias, linfct=C[-1,]))

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region - 1, data = dat.rend)
##
## Linear Hypotheses:
##         Estimate Std. Error t value Pr(>|t|)
## c1 == 0   -392.8      136.7  -2.874   0.0138 *
## c2 == 0   -463.8      124.6  -3.723   <0.001 ***
## c3 == 0   -334.2      224.4  -1.490   0.3221
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Como vemos, con tres contrastes los valores p son más chicos y sugieren que los lotes de las regiones «b» y «c» tienen rendimientos promedios inferiores a los de la región «a» (las diferencias son negativas).

3. 6. 2. Codificación por desviación

La *codificación por desviación* es otro de los sistemas predefinidos en R, y con ella se contrastan p - 1 niveles contra la gran media (μ_μ).[5] Para esto, la gran media constituye el intercepto del modelo ($\beta_0 = \mu_\mu$), a diferencia de la codificación *dummy* en la que el intercepto expresa la media de uno de los niveles ($\beta_0 = \mu_{ra}$ en el caso de `m.unif`). En términos del problema que estamos evaluando, cada una de las medias de las tres primeras regiones se contrastan contra el promedio de las cuatro medias regionales:

$$\beta_1 = \mu_{ra} - \mu_\mu$$
$$\beta_2 = \mu_{rb} - \mu_\mu$$
$$\beta_3 = \mu_{rc} - \mu_\mu$$

Con esta codificación se obtiene la parametrización que más se parece al modelo de efectos ($y = \mu + \alpha_j$). La diferencia con aquel es que no hay un contraste para la media del último nivel (región «d») ya que, como indicamos antes, incluirlo sobreparametrizaría el modelo. Siendo una codificación predefinida, se obtiene con una función específica: `contr.sum()`. Vemos la matriz que genera esta función en un factor de cuatro niveles como `region` y la comparamos contra la de la codificación por defecto:

[5] El promedio de las medias de los niveles del factor. En el caso de `region` la gran media se obtiene como: $\mu_\mu = (\mu_{ra} + \mu_{rb} + \mu_{rc} + \mu_{rd})/4$.

```r
contr.sum(n=4) # codificación por deviación
```

```
##   [,1] [,2] [,3]
## 1    1    0    0
## 2    0    1    0
## 3    0    0    1
## 4   -1   -1   -1
```

```r
contr.treatment(n=4) # codificación dummy
```

```
##   2 3 4
## 1 0 0 0
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1
```

Al incluir el intercepto (y renombrar las columnas según los contrastes y las filas según las regiones) obtenemos la matriz S:

```r
Ss <- cbind(1, contr.sum(4)) # matriz S
colnames(Ss) <- c("B0", "x1", "x2", "x3") # asignamos nombres a las columnas
rownames(Ss) <- levels(dat.rend$region)   # y a las filas
Ss
```

```
##    B0 x1 x2 x3
## ra  1  1  0  0
## rb  1  0  1  0
## rc  1  0  0  1
## rd  1 -1 -1 -1
```

Para ajustar el modelo con la codificación por desviación debemos codificar el factor de acuerdo a este sistema. Existen tres formas de hacerlo. La primera es configurarlo como la opción por defecto para los factores, para lo cual usamos la función `options()`[6]:

```r
# options(contrasts=c("contr.sum","contr.poly"))
```

Utilizando esta alternativa todos los factores serán codificados según el contraste que le pasamos a `options()`. Otra posibilidad es asignar el contraste desde la función en la que está definido, en este caso `contr.sum()`:

```r
contrasts(dat.rend$region) <- "contr.sum"
contrasts(dat.rend$region)
```

```
##    [,1] [,2] [,3]
## ra    1    0    0
## rb    0    1    0
## rc    0    0    1
## rd   -1   -1   -1
```

Ahora `region` incluye el contraste dentro de sus atributos:

[6] Es necesario indicar dos codificaciones, la segunda de ellas para los factores ordenados (para estos, por defecto R utiliza `"contr.poly"`).

```
str(dat.rend$region)

##  Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
##  - attr(*, "contrasts")= chr "contr.sum"
```

con lo cual el factor mantendrá esta codificación en los posteriores análisis de la sesión. Si queremos volver a la codificación *dummy* podemos usar la misma estrategia:

```
contrasts(dat.rend$region) <- "contr.treatment"
str(dat.rend$region)

##  Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
##  - attr(*, "contrasts")= chr "contr.treatment"
```

o cargar nuevamente los datos:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
dat.rend$region <- as.factor(dat.rend$region)
str(dat.rend$region)

##  Factor w/ 4 levels "ra","rb","rc",..: 1 1 1 1 2 1 1 1 1 2 ...
```

Al cargar los datos `region` ya no tiene el atributo del contraste. Esto quiere decir que R por defecto codifica los factores con un sistema *dummy* pero no les asigna un atributo que lo indique explícitamente. La tercera opción es codificar el factor directamente dentro de `lm()`. En este caso, la codificación será utilizada solo en el ajuste del modelo:

```
# modelo ajustado con la codificación por desviación
m.unif.s <- lm(rendimiento ~ region, data=dat.rend,
                             contrasts=list(region="contr.sum"))
```

Extrayendo el vector de coeficientes estimados:

```
bs <- coef(m.unif.s) # coeficientes del modelo ajustado
```

podemos corroborar que sus valores coinciden con lo que representan los parámetros del modelo bajo esta codificación. Para ello utilizaremos el objeto `medias` que generamos anteriormente. Como vemos, el intercepto es igual a la gran media:

```
bs[1] # b0

## (Intercept)
##    2469.199

sum(medias)/4 # gran media

## [1] 2469.199
```

y el resto de los coeficientes son diferencias con respecto a la gran media:

```
bs[2] # b1

##  region1
## 297.7136

medias[1] - sum(medias)/4 # media ra - gran media

##       ra
## 297.7136

bs[3] # b2

##   region2
## -95.13044

medias[2] - sum(medias)/4 # media rb - gran media

##        rb
## -95.13044

bs[4] # b3

##   region3
## -166.0505

medias[3] - sum(medias)/4 # media rc - gran media

##        rc
## -166.0505
```

Los valores *p* de los contrastes individuales son informados en el resumen del modelo:

```
summary(m.unif.s)$coef

##               Estimate Std. Error   t value     Pr(>|t|)
## (Intercept) 2469.19940   63.10546 39.128145 7.726863e-63
## region1      297.71364   95.87228  3.105315 2.467056e-03
## region2      -95.13044   90.07624 -1.056110 2.934375e-01
## region3     -166.05047   80.81757 -2.054633 4.249636e-02
```

(Es necesario advertir que los efectos han sido nombrados de manera diferente a los del ajuste del modelo con la codificación *dummy*). Como antes, para los valores *p* ajustados por multiplicidad podemos usar `glht()` pasándole el modelo ajustado con el contraste, o directamente C (debe recordarse que esta opción trabaja con el modelo de medias). En este caso usaremos la matriz ya que nos permite remover la prueba del intercepto que no es de interés. Entonces, primero obtenemos C invirtiendo S (y le asignamos nombres a las filas para una salida más clara):

```
Cs <- solve(Ss) # matriz de coeficientes de contraste (inversa de S)
rownames(Cs) <- c("B0", "c1", "c2", "c3")
Cs

##       ra    rb    rc    rd
## B0  0.25  0.25  0.25  0.25
## c1  0.75 -0.25 -0.25 -0.25
## c2 -0.25  0.75 -0.25 -0.25
## c3 -0.25 -0.25  0.75 -0.25
```

y luego calculamos los valores *p* ajustados:

```
summary(glht(m.medias, linfct=Cs[-1,])) # ajuste valores p por 3 pruebas simultáneas

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region - 1, data = dat.rend)
##
## Linear Hypotheses:
##         Estimate Std. Error t value Pr(>|t|)
## c1 == 0   297.71      95.87   3.105  0.00734 **
## c2 == 0   -95.13      90.08  -1.056  0.64365
## c3 == 0  -166.05      80.82  -2.055  0.12120
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Si el ajuste por multiplicidad se realiza considerando la prueba del intercepto (cuatro pruebas en lugar de tres), no necesitamos definir las matrices S y C. En este caso solo debemos ajustar el modelo indicando la codificación predefinida (lo que hemos hecho al crear el objeto `m.unif.s`) y pasarlo dentro de la función `glht()`:[7]

```
summary(glht(m.unif.s)) # ajuste por 4 pruebas simultáneas

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend, contrasts = list(region = "c
ontr.sum"))
##
## Linear Hypotheses:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept) == 0  2469.20      63.11  39.128  < 0.001 ***
## region1 == 0       297.71      95.87   3.105  0.00944 **
## region2 == 0       -95.13      90.08  -1.056  0.71769
## region3 == 0      -166.05      80.82  -2.055  0.14919
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

El modelo ajustado con la codificación por desviación nos permite concluir que los lotes de la región «a» tienen un rendimiento promedio superior a la media general.

3. 6. 3. Codificación Helmert

Esta es la codificación que resulta de los contrastes Helmert, en los cuales cada nivel del factor se contrasta con la media de los niveles subsecuentes. Codificar `region` de esta manera implica comparar: (β_1) la media de «rb» con la media de «ra», (β_2) la media de

[7] En el caso de que los contrastes no se encuentren dentro de las opciones de R, el ajuste del modelo requiere haber generado las matrices **C** y **S** (ver la subsección *2.5.*y «Contraste definido por el usuario»).

«rc» contra el promedio de las medias de «ra» y «rb», y (β_3) la media de «rd» contra el promedio entre las medias del resto de las regiones (tabla 3. 1). El intercepto del modelo es la gran media ($\beta_0 = \mu_\mu$). La codificación Helmert se obtiene con la función `contr.helmert()` que le pasamos `lm()` para ajustar el modelo con este sistema:

```
m.unif.h <- lm(rendimiento ~ region, data=dat.rend,
                             contrasts=list(region="contr.helmert")) # ajuste modelo
summary(glht(m.unif.h)) # estimaciones y valores p corregidos por 4 pruebas simult.

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend, contrasts = list(region = "c
ontr.helmert"))
##
## Linear Hypotheses:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept) == 0  2469.20      63.11  39.128   <0.001 ***
## region1 == 0      -196.42      68.34  -2.874   0.0189 *
## region2 == 0       -89.11      32.95  -2.705   0.0302 *
## region3 == 0       -12.18      51.59  -0.236   0.9984
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Con la codificación Helmert concluimos que la región «b» tiene un rendimiento promedio inferior al de la región «a» (fila `region1 == 0`) y que el de la región «c» es inferior al rendimiento medio de las regiones «a» y «b» (fila `region2 == 0`).

3. 6. 4. Modelo de medias

En este modelo cada parámetro es la media de uno de los niveles. No tiene asociado ningún contraste especial y se lo ajusta restando uno a la derecha de la fórmula de `lm()`. Es el modelo que hemos guardado en el objeto `m.medias`:

```
m.medias$call # modelo

## lm(formula = rendimiento ~ region - 1, data = dat.rend)
```

Si bien presenta la ventaja de la comunicación de los resultados ya que cada parámetro es la media de un nivel, en general, el contraste asociado no es de interés (comúnmente la hipótesis de que la media es diferente de cero es trivial):

```
summary(m.medias)$coef

##          Estimate Std. Error  t value     Pr(>|t|)
## regionra 2766.913  102.07052 27.10786 3.777537e-48
## regionrb 2374.069   90.90028 26.11729 1.018985e-46
## regionrc 2303.149   71.40281 32.25572 5.437865e-55
## regionrd 2432.667  199.84285 12.17290 1.596244e-21
```

3.6.5. Contraste definido por el usuario

Muchas veces nuestro problema no se ajusta a ninguno de los contrastes predefinidos. En este caso, podemos establecerlos nosotros. Imaginemos que «ra» es la región núcleo en la zona de estudio, y que «rb», «rc» y «rd» son clasificadas como regiones que pueden presentar algunas limitaciones productivas. Supongamos además que «rd» es una región que impulsa el manejo agroecológico. Con base en ello, nos interesa parametrizar el modelo de acuerdo a:

- el rendimiento promedio en la zona de estudio: $(\mu_{ra} + \mu_{rb} + \mu_{rc} + \mu_{rd})/4$ (intercepto),
- la diferencia entre la región núcleo y el resto de la zona: μ_{ra} - $(\mu_{rb} + \mu_{rc} + \mu_{rd})/3$,
- la diferencia con y sin manejo agroecológico en regiones con limitaciones: μ_{rd} - $(\mu_{rb} + \mu_{rc})/2$,
- la diferencia entre las regiones con limitaciones sin manejo agroecológico: μ_{rb} - μ_{rc}.

A partir de esto, definimos los coeficientes de contraste que se incluyen en la matriz C (recordemos que en la fila de cada contraste los coeficientes deben sumar cero):

- primera fila: $1/4 \times \mu_{ra} + 1/4 \times \mu_{rb} + 1/4 \times \mu_{rc} + 1/4 \times \mu_{rd}$,
- segunda fila: $1 \times \mu_{ra} - 1/3 \times \mu_{rb} - 1/3 \times \mu_{rc} - 1/3 \times \mu_{rd}$,
- tercera fila: $0 \times \mu_{ra} - 1/2 \times \mu_{rb} - 1/2 \times \mu_{rc} + 1 \times \mu_{rd}$,
- cuarta fila: $0 \times \mu_{ra} + 1 \times \mu_{rb} - 1 \times \mu_{rc} + 0 \times \mu_{rd}$.

La creamos en R:

```
B0 <- c(1/4, 1/4, 1/4, 1/4) # intercepto (gran media)

c1 <- c(1, -1/3, -1/3, -1/3) # primer contraste
c2 <- c(0, -1/2, -1/2, 1)    # segundo contraste
c3 <- c(0, 1, -1, 0)         # tercer contraste

Cu <- rbind(B0, c1, c2, c3) # matriz C
colnames(Cu) <- levels(dat.rend$region)
MASS::fractions(Cu)

##      ra   rb   rc   rd
## B0  1/4  1/4  1/4  1/4
## c1    1 -1/3 -1/3 -1/3
## c2    0 -1/2 -1/2    1
## c3    0    1   -1    0
```

Al invertir C obtenemos la matriz de S con la codificación que necesitamos para nuestros contrastes:

```
Su <- solve(Cu) # matriz S (inversa de C)
colnames(Su) <- c("B0", "x1", "x2", "x3")
```

que es la que debemos pasarle a `lm()` removiendo la columna del intercepto:

```
m.unif.u <- lm(rendimiento ~ region, data=dat.rend,
                             contrasts=list(region=Su[,-1]))
```

Luego, calculamos los valores *p* ajustados. Dado que hemos definido las matrices, podemos descartar la prueba del intercepto y ajustar por tres contrastes:

```
summary(glht(m.medias, linfct=Cu[-1,])) # valores p ajustados a 3 contrastes simult.

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Fit: lm(formula = rendimiento ~ region - 1, data = dat.rend)
##
## Linear Hypotheses:
##          Estimate Std. Error t value Pr(>|t|)
## c1 == 0   396.95     127.83   3.105  0.00729 **
## c2 == 0    94.06     208.03   0.452  0.95342
## c3 == 0    70.92     115.59   0.614  0.89423
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

Con esta parametrización encontramos que los lotes de la zona núcleo («ra») tienen un rinde promedio significativamente más alto que el resto de los lotes (fila `c1 == 0`).

3. 6. 6. Comparación de contrastes

Es importante tener en claro que al aplicar diferentes contrastes no hemos cambiado el modelo, sino que lo reparametrizamos para poner a prueba distintas hipótesis (en la práctica, utilizamos una única parametrización). En otras palabras, los parámetros del modelo (β_0, β_1, β_2, β_3) representan cosas diferentes (tabla 3. 1).

Tabla 3. 1. Interpretación de los parámetros de un modelo que predice el rendimiento de un cultivo en función de la región, un factor de cuatro niveles («ra», «rb», «rc», «rd») según diferentes esquemas de codificación.

Codificación	**Parámetro**			
	β_0	β_1	β_2	β_3
dummy (defecto)	μ_{ra}	μ_{rb} - μ_{ra}	μ_{rc} - μ_{ra}	μ_{rd} - μ_{ra}
desviación	μ_{μ}	μ_{ra} - μ_{μ}	μ_{rb} - μ_{μ}	μ_{rc} - μ_{μ}
Helmert	μ_{μ}	μ_{rb} - μ_{ra}	μ_{rc} - $(\mu_{ra} + \mu_{ra})/2$	μ_{rd} - $(\mu_{ra} + \mu_{rb} + \mu_{rc})/3$
medias	μ_{ra}	μ_{rb}	μ_{rc}	μ_{rd}
usuario	μ_{μ}	μ_{ra} - $(\mu_{rb} + \mu_{rc} + \mu_{rd})/3$	μ_{rd} - $(\mu_{rb} + \mu_{rc})/2$	μ_{rb} - μ_{rc}

Consecuentemente, los coeficientes estimados difieren:

```r
bh <- m.unif.h$coeff # coeficientes estimados codificación Helmert
bm <- m.medias$coeff # coeficientes estimados modelo de medias
bu <- m.unif.u$coeff # coeficientes estimados codificación definida por el usuario

betas <- rbind(b, bs, bh, bm, bu)
rownames(betas) <- c("dummy", "desv.", "Helmert", "medias", "usuario")
colnames(betas) <- c("b0", "b1", "b2", "b3")
round(betas, 1)

##             b0     b1     b2     b3
## dummy   2766.9 -392.8 -463.8 -334.2
## desv.   2469.2  297.7  -95.1 -166.1
## Helmert 2469.2 -196.4  -89.1  -12.2
## medias  2766.9 2374.1 2303.1 2432.7
## usuario 2469.2  397.0   94.1   70.9
```

Por ello, también deben modificarse los valores que asumen las variables (x_1, x_2, x_3) en cada nivel del factor (tabla 3. 2), pero todas las parametrizaciones predicen el mismo rendimiento:

```r
pr <- predict.lm(m.unif, regiones)     # dummy
pr.s <- predict.lm(m.unif.s, regiones) # desviación
pr.h <- predict.lm(m.unif.h, regiones) # Helmert
pr.m <- predict.lm(m.unif.m, regiones) # modelo de medias
pr.u <- predict.lm(m.unif.u, regiones) # usuario

pred <- rbind(pr, pr.s, pr.h, pr.m, pr.u)
rownames(pred) <- c("dummy", "desv.", "Helmert", "medias", "usuario")
colnames(pred) <- c("ra", "rb", "rc", "rd")
round(pred, 1)

##             ra     rb     rc     rd
## dummy   2766.9 2374.1 2303.1 2432.7
## desv.   2766.9 2374.1 2303.1 2432.7
## Helmert 2766.9 2374.1 2303.1 2432.7
## medias  2766.9 2374.1 2303.1 2432.7
## usuario 2766.9 2374.1 2303.1 2432.7
```

Tabla 3. 2. Valores que en los diferentes esquemas de codificación asumen las variables introducidas al modelo lineal general según la región (nivel del factor) en el que se encuentre el lote (unidad experimental)

Nivel del factor	Codificación	Variable en el modelo		
		x_1	x_2	x_3
región «a»	*dummy* (defecto)	0	0	0
	desviación	1	0	0
	Helmert	-1	-1	-1
	uruario	3/4	0	0
región «b»	*dummy* (defecto)	1	0	1
	desviación	0	1	0
	Helmert	1	-1	-1
	usuario	-1/4	1/3	1/2
región «c»	*dummy* (defecto)	0	1	0
	desviación	0	0	1
	Helmert	0	2	-1
	usuario	-1/4	1/3	-1/2
región «d»	*dummy* (defecto)	0	0	1
	desviación	-1	-1	-1
	Helmert	0	0	3
	usuario	-1/4	-2/3	0

Finalmente, observamos que la partición de la varianza del rendimiento ha sido idéntica en los diferentes contrastes:

```
anova(m.unif)   # anova codificación dummy

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## region      3  3457476 1152492  4.8096 0.003582 **
## Residuals 101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.unif.s) # anova codificación por desviación

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## region      3  3457476 1152492  4.8096 0.003582 **
## Residuals 101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(m.unif.h) # anova codificación Helmert

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## region      3  3457476 1152492  4.8096 0.003582 **
## Residuals 101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.unif.u) # anova codificación usuario

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## region      3  3457476 1152492  4.8096 0.003582 **
## Residuals 101 24201921  239623
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Esto quiere decir que los contrastes han separado de manera diferente la misma porción de la variabilidad del rendimiento, esto es, la porción explicada por la región.

3. 7. Comparaciones *a posteriori* y pruebas múltiples

Hemos visto que el ANOVA nos permite evaluar si la región afecta al rendimiento, pero no informa acerca de cuál región difiere de cual. Por otro lado, los contrastes *a priori* son una forma de detectar qué medias difieren desde los parámetros del MLG. Con un factor de p niveles podemos incluir en el modelo p - 1 contrastes. Por ejemplo, al codificar el factor `region` de acuerdo al sistema *dummy* comparamos tres pares de medias (μ_{rb} *vs.* μ_{ra}; μ_{rc} *vs.* μ_{ra}; μ_{rd} *vs.* μ_{ra}). Sin embargo, el número posible de comparaciones de a pares es $p(p - 1)/2$, es decir, $4 \times 3/2 = 6$ pares de regiones, y de existir diferencias entre muchas de ellas (por ejemplo entre μ_{rb} *vs.* μ_{rc}), estas no serán detectadas.[8] Para eso necesitamos de las comparaciones múltiples. El propósito de estas es ayudar al analista a tomar decisiones respecto a los tratamientos que se comparan. No obstante la gran utilidad de las comparaciones múltiples, deben ser aplicadas con las consideraciones del caso. Veamos algunos aspectos que debemos tener presentes al aplicarlas (basado en Garibaldi y otros, 2023):

a) se conocen como pruebas *a posteriori* porque no se planean de antemano y se realizan solamente luego de encontrar efectos significativos en el ANOVA,

b) cuando se realizan múltiples comparaciones sobre los mismos datos (se prueban más de una hipótesis a la vez), la tasa de error de tipo 1 del experimento (α_e), es decir, la

[8] El ANOVA podría sugerir que las medias son diferentes, pero ninguno de los contrastes puede ser significativo de modo que no es posible identificar cuál media difiere de cual. Más aún, aunque exista un efecto del factor podría ser que las diferencias no fueran detectadas incluso si se comparasen todos los pares de medias. Esto sucede porque el ANOVA pone a prueba simultáneamente todas las combinaciones lineales posibles, incluidas las comparaciones de a pares. Por lo tanto, la significancia detectada podría deberse a una combinación de tratamientos (por ejemplo, el rendimiento promedio de «ra» y «rb» respecto al rendimiento promedio de «rc» y «rd»).

probabilidad de rechazar en forma equivocada la igualdad de medias en al menos una de las comparaciones, se incrementa con el número de contrastes (m) de acuerdo a:

$$\alpha_e = 1 - (1 - \alpha_c)^m$$

donde α_c es la probabilidad de error de tipo 1 de cada comparación. Por ejemplo, si α_c = 0,05 entonces para m = 1, 2, …, 10:

m	1	2	3	4	5	6	7	8	9	10
α_e	0,05	0,10	0,14	0,19	0,23	0,26	0,30	0,34	0,37	0,40

c) corregir α_c para mantener α_e en niveles aceptables, aumenta el error de tipo 2 de cada comparación (disminuye la capacidad de detectar la diferencia de medias cuando existen), lo cual es especialmente problemático en estudios con muchos tratamientos y pocas repeticiones,

d) existen distintos métodos para controlar las tasas de error, aunque no hay uno único que sea el mejor para todas las situaciones (tabla 3. 3). Aquí vemos cómo aplicar tres de ellos usando dos paquetes de R.

3. 7. 1. Test de Fisher

Se basa en el método de la *diferencia mínima significativa* (LSD = *Least Significant Difference*) propuesto por Fisher. El paquete `agricolae` incluye la función `LSD.test()` que permite aplicar el test de Fisher y otros métodos de corrección. Si bien en este test no se corrige el α_c, al aplicarse cuando el ANOVA ha indicado diferencias entre tratamientos se considera que $\alpha_e < 1 - (1 - \alpha_c)^m$. Se concluye que las medias poblacionales de dos tratamientos a y b son estadísticamente diferentes ($\mu_a \neq \mu_b$) cuando:

$$|\bar{y}_a - \bar{y}_b| > \text{LSD}$$

LSD surge de transformar el valor crítico de una distribución t con n - m grados de libertad.[9] En la función `LSD.test()`, el nivel de significancia para obtener el t crítico se define en el argumento `alpha`. Al indicar `p.adj="none"` realizamos comparaciones de t pareadas sin corrección del α_c, implementando el método propuesto originalmente por Fisher:

```
library(agricolae) # instalar antes el paquete
LSD.test(m.unif, "region", p.adj="none", alpha = 0.05, console=TRUE)

##
## Study: m.unif ~ "region"
##
```

[9] La diferencia con la comparación de p grupos de a pares aplicando el clásico test de Student es que en este se usa una distribución t con r – 2 grados de libertad (siendo r la suma de réplicas los dos grupos comparados) y se estima una varianza para cada una de las m pruebas. En cambio, el método de la LSD considera los m grados de libertad requeridos por el total de comparaciones y la estimación de la (única) varianza incluye toda la información disponible.

```
## LSD t Test for rendimiento
##
## Mean Square Error:  239623
##
## region,  means and individual ( 95 %) CI
##
##    rendimiento      std  r      LCL      UCL  Min  Max
## ra    2766.913 621.2537 23 2564.433 2969.393 1770 3650
## rb    2374.069 269.9960 29 2193.747 2554.391 2051 3027
## rc    2303.149 529.3526 47 2161.505 2444.793 1400 3592
## rd    2432.667 394.9419  6 2036.232 2829.101 2049 3100
##
## Alpha: 0.05 ; DF Error: 101
## Critical Value of t: 1.983731
##
## Groups according to probability of means differences and alpha level( 0.05 )
##
## Treatments with the same letter are not significantly different.
##
##    rendimiento groups
## ra    2766.913      a
## rd    2432.667     ab
## rb    2374.069      b
## rc    2303.149      b
```

Alternativamente, podemos usar el paquete `multcomp` que aplicamos a los contrastes *a priori*. Primero debemos indicarle a la función `glht` que queremos una matriz de contrastes que incluya todas las comparaciones pareadas y una forma de hacerlo es pasar la función `mcp()` al argumento `linfct`. Las comparaciones pareadas se indican desde esta función con la opción `"Tukey"`, en este caso aplicada a `region`, el factor cuyos niveles deseamos comparar:

```
cp <- glht(m.unif, linfct=mcp(region="Tukey"))
```

Luego, desde el argumento `test` de las funciones `summary()` y `cld()` indicamos el método de corrección que pretendemos:

```
summary(cp, test=adjusted(type="none"))

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend)
##
## Linear Hypotheses:
##              Estimate Std. Error t value Pr(>|t|)
## rb - ra == 0  -392.84     136.68  -2.874 0.004939 **
## rc - ra == 0  -463.76     124.57  -3.723 0.000324 ***
## rd - ra == 0  -334.25     224.40  -1.490 0.139470
## rc - rb == 0   -70.92     115.59  -0.614 0.540896
## rd - rb == 0    58.60     219.55   0.267 0.790086
## rd - rc == 0   129.52     212.22   0.610 0.543026
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- none method)
```

```
cld(cp, test=adjusted(type="none"))

##   ra   rb   rc   rd
##  "b"  "a"  "a" "ab"
```

Al seleccionar `"none"` hemos indicado que los valores p de las comparaciones no sean ajustados, es decir, aplicamos el test de Fisher. Cabe señalar que la salida anterior informa la diferencia de medias mientras que `LSD.test()` reporta la estimación de cada media.

3. 7. 2. Corrección de Bonferroni

En `LSD.test()` tenemos varias alternativas de corrección al α_c:

```
?LSD.test()
```

Entre estas, la corrección de Bonferroni (`p.adj="bonferroni"`):

$$\alpha_e = \alpha_c/m$$

Al dividir el α_e por el número de contrastes (m), mantenemos la tasa de error de tipo 1 del experimento en el nivel deseado. Sin embargo, aumentamos la probabilidad de error de tipo 2 de cada comparación. Cuando la cantidad de tratamientos es elevada, la capacidad de detectar diferencias se reduce considerablemente, es decir, resignamos potencia estadística.

```
LSD.test(m.unif, trt="region", p.adj="bonferroni", alpha = 0.05, console=TRUE)

##
## Study: m.unif ~ "region"
##
## LSD t Test for rendimiento
## P value adjustment method: bonferroni
##
## Mean Square Error:  239623
##
## region,  means and individual ( 95 %) CI
##
##    rendimiento      std  r      LCL      UCL  Min  Max
## ra    2766.913 621.2537 23 2564.433 2969.393 1770 3650
## rb    2374.069 269.9960 29 2193.747 2554.391 2051 3027
## rc    2303.149 529.3526 47 2161.505 2444.793 1400 3592
## rd    2432.667 394.9419  6 2036.232 2829.101 2049 3100
##
## Alpha: 0.05 ; DF Error: 101
## Critical Value of t: 2.691216
##
## Groups according to probability of means differences and alpha level( 0.05 )
##
## Treatments with the same letter are not significantly different.
##
##    rendimiento groups
## ra    2766.913      a
## rd    2432.667     ab
```

```
## rb    2374.069      b
## rc    2303.149      b
```

Para aplicar la corrección de Bonferroni con el paquete `multcomp` reemplazamos `"none"` por `"bonferroni"` (en el objeto `cp` ya hemos indicado que queremos contrastar todas las medias de a pares):

```
summary(cp, test=adjusted(type="bonferroni"))

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend)
##
## Linear Hypotheses:
##              Estimate Std. Error t value Pr(>|t|)
## rb - ra == 0  -392.84     136.68  -2.874  0.02963 *
## rc - ra == 0  -463.76     124.57  -3.723  0.00194 **
## rd - ra == 0  -334.25     224.40  -1.490  0.83682
## rc - rb == 0   -70.92     115.59  -0.614  1.00000
## rd - rb == 0    58.60     219.55   0.267  1.00000
## rd - rc == 0   129.52     212.22   0.610  1.00000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- bonferroni method)

cld(cp, test=adjusted(type="bonferroni"))

##   ra   rb   rc   rd
##  "b"  "a"  "a" "ab"
```

3. 7. 3. Test de Tukey

Este test es uno de los más utilizados en la práctica. En el paquete `agricolae` podemos obtenerlo mediante la función `HSD.test()`:

```
tukey <- HSD.test(m.unif, "region", console=TRUE)

##
## Study: m.unif ~ "region"
##
## HSD Test for rendimiento
##
## Mean Square Error:  239623
##
## region,  means
##
##    rendimiento      std  r  Min  Max
## ra    2766.913 621.2537 23 1770 3650
## rb    2374.069 269.9960 29 2051 3027
## rc    2303.149 529.3526 47 1400 3592
## rd    2432.667 394.9419  6 2049 3100
##
## Alpha: 0.05 ; DF Error: 101
## Critical Value of Studentized Range: 3.694382
##
```

```
## Groups according to probability of means differences and alpha level( 0.05 )
##
## Treatments with the same letter are not significantly different.
##
##    rendimiento groups
## ra    2766.913      a
## rd    2432.667     ab
## rb    2374.069      b
## rc    2303.149      b
```

Alternativamente, podríamos usar la función `TukeyHSD()` que viene incluida en el paquete base, aunque para que funcione el modelo debería haberse ajustado con la función `aov()`:

```
TukeyHSD(aov(rendimiento ~ region, data=dat.rend))

##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = rendimiento ~ region, data = dat.rend)
##
## $region
##              diff       lwr        upr     p adj
## rb-ra -392.84408 -749.8947  -35.79343 0.0250677
## rc-ra -463.76411 -789.1714 -138.35684 0.0018165
## rd-ra -334.24638 -920.4527  251.95999 0.4475899
## rc-rb  -70.92003 -372.8804  231.04034 0.9275554
## rd-rb   58.59770 -514.9248  632.12022 0.9933062
## rd-rc  129.51773 -424.8583  683.89378 0.9285891
```

En el caso del paquete `multcomp`, simplemente no usamos el argumento `test` ya que Tukey es el método de corrección que hemos asignamos desde `mcp()`:

```
summary(cp) # mulcomp

##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = rendimiento ~ region, data = dat.rend)
##
## Linear Hypotheses:
##               Estimate Std. Error t value Pr(>|t|)
## rb - ra == 0  -392.84     136.68  -2.874  0.02310 *
## rc - ra == 0  -463.76     124.57  -3.723  0.00171 **
## rd - ra == 0  -334.25     224.40  -1.490  0.43354
## rc - rb == 0   -70.92     115.59  -0.614  0.92390
## rd - rb == 0    58.60     219.55   0.267  0.99293
## rd - rc == 0   129.52     212.22   0.610  0.92499
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)

cld(cp)

##   ra   rb   rc   rd
##  "b"  "a"  "a" "ab"
```

Graficamos los resultados que obtuvimos e incluimos los rendimientos observados (figura. 3. 2):

```
# figura 3.2
plot(tukey, variation="SE",
     main="", ylim=c(1000, 4500),
     ylab="Rendimiento (kg/ha)", xlab="Región")
points(dat.rend$rendimiento ~ dat.rend$region,
       col="grey", cex=0.75)
```

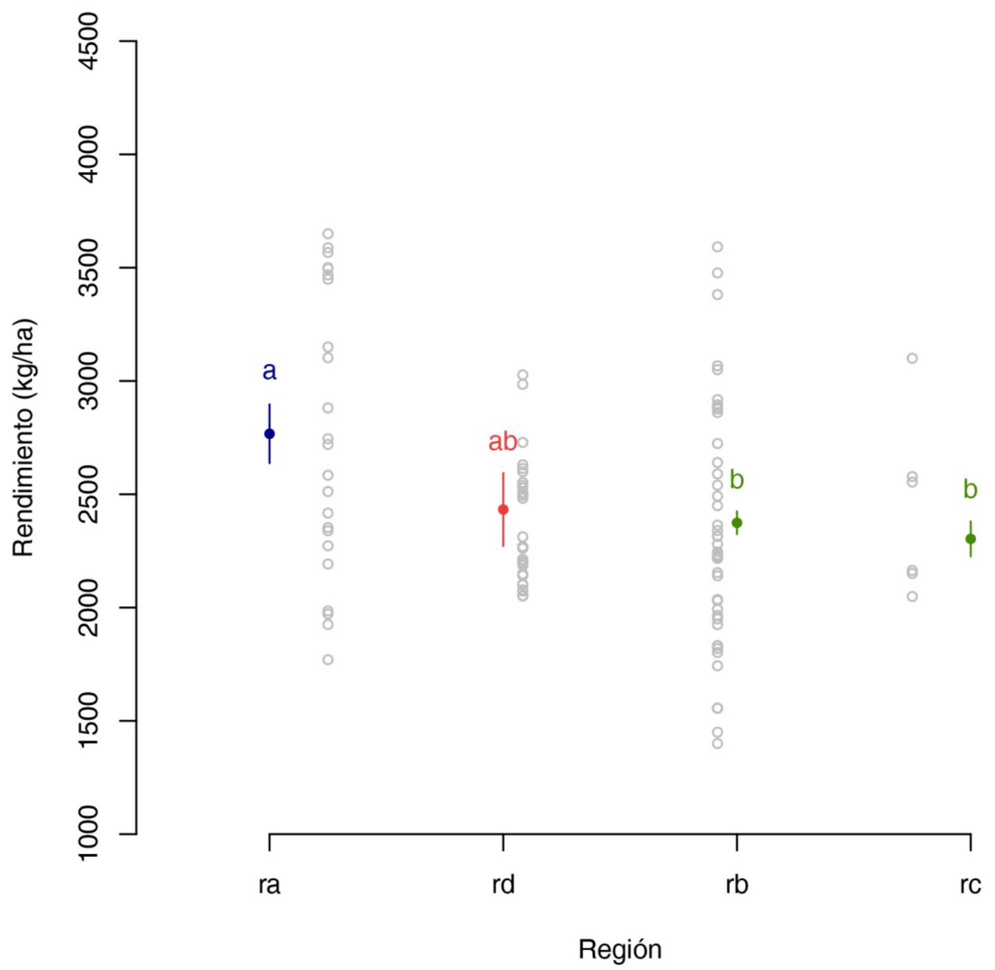

Figura 3. 2. Rendimiento de girasol en cada región
Nota. Las letras informan las medias que difieren de acuerdo a un test de Tukey. Los puntos sólidos indican las medias de cada región y las barras verticales su error estándar. El gráfico incluye los rendimientos observados en cada región (puntos sin relleno).

Tabla 3. 3. Resumen de los métodos de corrección de la probabilidad de error de tipo I en las comparaciones múltiples

	Método		
	Fisher	*Bonferroni*	*Tukey*
α_e	$1 - (1 - \alpha_c)^m$	α	$\leq \alpha$
α_c	α	α/m	depende de k y r
estadístico de prueba	t	t	q
consideraciones prácticas	- control del error de tipo 1 de las comparaciones - incrementa la probabilidad de error de tipo 1 del experimento	- control del error de tipo 1 del experimento - aumenta el error de tipo 2 de las comparaciones	- control parcial del error de tipo 1 del experimento y comparaciones

Nota. α: nivel de significancia definido antes del experimento; α_e: probabilidad de cometer un error de tipo I en al menos una comparación; α_c: probabilidad de cometer un error de tipo I en cada comparación; m: cantidad de comparaciones; k: cantidad de tratamientos; r: número de repeticiones.

El test de Tukey presenta un compromiso entre mantener α_c (Fisher) y mantener α_e (Bonferroni). En particular, cuando los tratamientos tienen la misma cantidad de repeticiones (diseño balanceado) α_e es igual al nivel de significancia deseado. Para los contrastes se utiliza el estadístico q en lugar de t. El estadístico q es la diferencia estandarizada entre las medias muestrales más extremas (la más grande menos la más chica). Bajo la hipótesis nula, q sigue una distribución de rango «estudentizado» y su valor crítico (q_{cr}) se utiliza para rechazar la igualdad de las medias de dos tratamientos. En la salida, q_{cr} es informado como `Critical Value of Studentized Range`:

```
tukey$parameters$StudentizedRange # q crítico

## [1] 3.694382
```

Pero también podemos obtenerlo desde una distribución q con 4 medias y 101 grados de libertad:

```
qtukey(0.95, nmeans=4, df=101) # q crítico obtenido desde la distribución

## [1] 3.694382
```

El valor crítico expresado en la escala de la media se denomina *diferencia honestamente significativa* (HSD = *Honestly Significant Difference*) e indica la capacidad de detectar diferencias del test (toda diferencia de medias menor a HSD será no significativa) y puede interpretarse como un indicador de su potencia estadística:

$$\text{HSD} = q_{cr} \times \sqrt{\text{CME}/r}$$

donde *r* es el número de repeticiones por tratamiento.

```
q.crt <- qtukey(0.95, nmeans=4, df=101) # q
cme <- tukey$statistics$MSerror # CME
rep <- tukey$parameters$ntr/sum(1/tukey$means$r) # repeticiones (media armónica)

hsd = q.crt * sqrt((cme/rep))
hsd

## [1] 466.2714
```

Cuando el diseño es desbalanceado, como en nuestro caso, el número de repeticiones por tratamiento se obtiene como su media armónica:

```
rep

## [1] 15.04301

4/(1/23+1/29+1/47+1/6) # para ver más claro la media armónica del nº de rep.

## [1] 15.04301

summary(dat.rend$region)

## ra rb rc rd
## 23 29 47  6
```

Los tres test sugieren que los lotes de la región «a» rinden significativamente más que los lotes de las regiones «b» y «c».

3. 7. 4. Número de repeticiones necesarias

Antes indicamos que HSD es la capacidad del test de Tukey de detectar diferencias. Su formulación indica que depende de la variabilidad entre unidades experimentales (CME), de la confianza (implícita en el valor crítico) y del tamaño muestral (número de repeticiones), de manera similar a lo que ocurre con la potencia. Si antes de realizar el experimento queremos que nuestro test sea capaz de detectar cierta diferencia de rendimiento entre regiones, en caso de que la hubiera, de la ecuación de HSD podemos despejar el número de repeticiones necesarias:

$$r = \text{CME} \times (q/\text{HSD})^2$$

Veamos las repeticiones que necesitamos para detectar 250 kg ha^{-1}:

```
hds.obj <- 250 # diferencia que se desea detectar
rep.obj <- cme * (q.crt/hds.obj)^2 # repeticiones necesarias
rep.obj

## [1] 52.32776
```

3. 7. 5. Sobre los contrastes (*a priori*) y las comparaciones múltiples (*a posteriori*)

De modo general, tanto las pruebas *a priori* y como las *a posteriori* son comparaciones múltiples que se establecen como contrastes. Sin embargo, ambos abordajes presentan diferencias conceptuales y técnicas, en particular dentro del contexto de los MLG. Conceptualmente, los contrastes *a priori* reflejan hipótesis de trabajo planteadas antes de tomar los datos mientras que las comparaciones *a posteriori* han sido diseñadas para detectar patrones luego de observarlos. En este sentido, a los contrastes *a priori* se los suele vincular con los experimentos manipulativos, cuyo diseño responde a hipótesis de trabajo específicas. A las comparaciones *a posteriori* se las suele asociar con los estudios observacionales de tipo exploratorio. Sin embargo, nada restringe el uso de los primeros en los estudios observacionales y viceversa. En cuanto a lo técnico, los contrastes *a priori* son incluidos como parámetros en los MLG y permiten comparar entre grupos de medias. Las comparaciones *a posteriori*, en cambio, se realizan a partir de las predicciones del modelo (las medias de los niveles) y evalúan diferencias de a pares. Los usuarios de R están familiarizados con la codificación *dummy* y es común utilizarla combinada con las comparaciones a *posteriori*. De igual manera, se puede aplicar la codificación que se considere más conveniente para comunicar el modelo y aplicar las comparaciones múltiples una vez que el ANOVA indica que hay diferencia de medias. Lo que debe evitarse es guiar el análisis (por ejemplo, seleccionar un contraste en particular) luego de ver los datos. A modo de ejemplo, si se contrastan sistemáticamente los grupos cuyas medias están más alejadas (esto se conoce luego de observar los datos), se infla la probabilidad de error de tipo 1.

3. 8. Bondad de ajuste

La evaluamos de manera similar que en el caso de la regresión lineal simple. La varianza residual o CME que podemos obtener mediante varias funciones:

```
summary(m.unif)$sigma^2 # CME en el summary

## [1] 239623

anova(m.unif)[2,3] # CME en la tabla de ANOVA

## [1] 239623

tukey$statistics$MSerror # CME en test Tukey

## [1] 239623
```

El error estándar residual:

```
summary(m.unif)$sigma # error estándar residual

## [1] 489.513
```

El coeficiente de determinación:

```
summary(m.unif)$r.squared # coeficiente de determinación R²

## [1] 0.1250019
```

Como vemos, la región y la fertilización nitrogenada proveen modelos de rendimiento con similar bondad de ajuste.

3. 9. Validación de supuestos

Debemos validar los mismos supuestos que en el modelo de regresión lineal simple, de modo que procedemos en forma similar:

```
residuos <- resid(m.unif) # vector de residuos
predichos <- fitted(m.unif) # vector de predichos

# figura 3.3
par(mfrow=c(2,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend$region,
     xlab="Región", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

hist(residuos, main="", xlab="Residuos (kg/ha)", ylab="Frecuencia")
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")
```

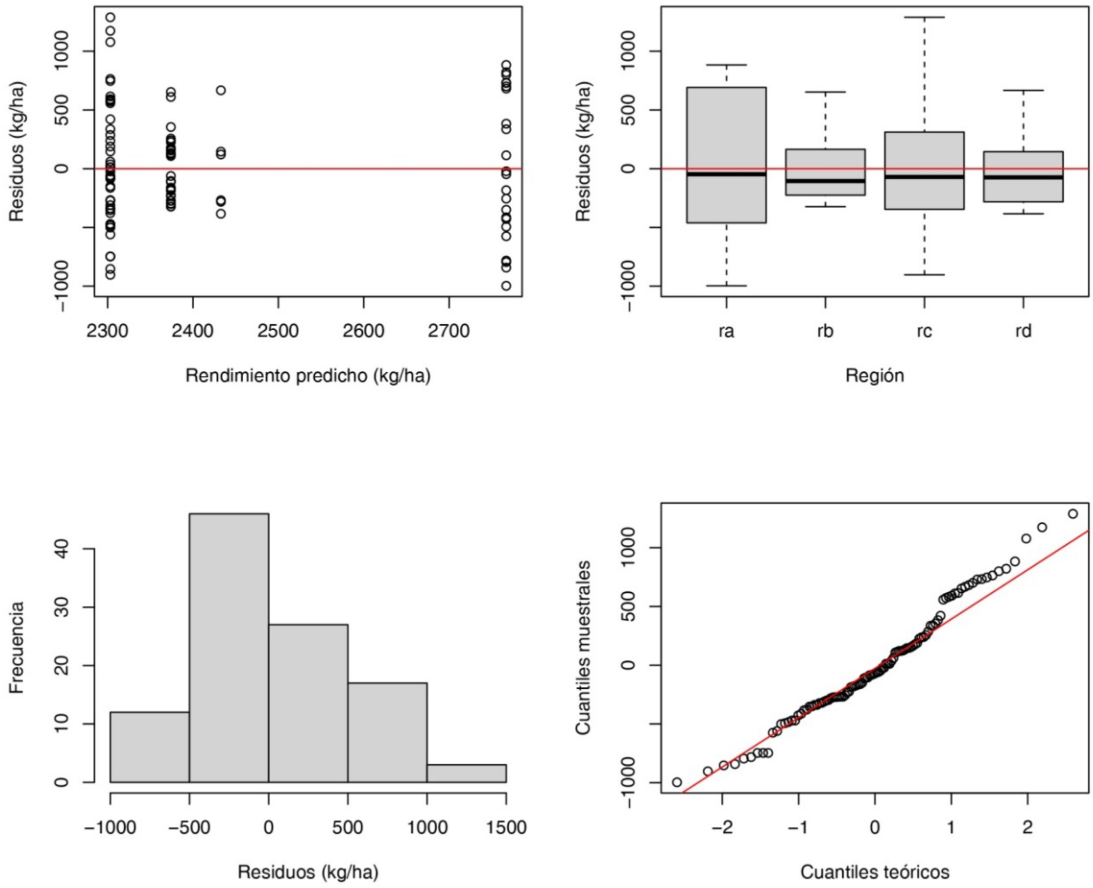

Figura 3. 3. Evaluación gráfica de los supuestos del modelo

En el capítulo 1 complementamos el análisis gráfico de la normalidad con el test de Kolmogorov-Smirnov, señalando que también existen otras pruebas. En este caso, aplicamos la modificación de «Lilliefors» al test usado antes (esta modificación considera que los parámetros son estimados):

```
library(nortest) # instalar antes el paquete
lillie.test(residuos) # modificación de "Lilliefors"

##
##  Lilliefors (Kolmogorov-Smirnov) normality test
##
## data:  residuos
## D = 0.072069, p-value = 0.1984
```

El análisis gráfico de los residuos (figura 3. 3) muestra que podría haber un problema con el supuesto de homogeneidad de varianzas. Dado que la región es el único predictor, este problema ya podía detectarse en el gráfico exploratorio del comienzo del capítulo (figura 3. 1).

Capítulo 4. Modelos multifactoriales

4. 1. Introducción

El capítulo expande los conceptos del capítulo anterior y es la puerta de entrada a los modelos con más de un predictor. Dado que seguimos abordando predictores que son factores a los modelos que presentamos se les suele llamar *multifactoriales*. Con base en estos modelos, discutimos aspectos del muestreo y del diseño de experimentos. En particular, abordamos los *diseños en bloques completos aleatorizados* (DBCA) y los *diseños factoriales*, distinguiéndolos del DCA del capítulo anterior. Asociado a estos diseños, introducimos el concepto de *interacción* como componente clave para el modelado estadístico.

4. 2. Problema

Se busca evaluar el efecto que tiene el cultivo previo en el rendimiento del girasol. El desarrollo de un cultivo y su cosecha impactan en las condiciones edáficas del lote. La modificación del ambiente edáfico puede afectar el rendimiento del cultivo posterior. La rotación del lote, en efecto, es la aplicación de cierta secuencia de cultivos ordenada en el tiempo para preservar la salud del suelo y sostener la producción agrícola. Dado que las características ambientales de una región impactan en el rendimiento de los cultivos, sería interesante que nuestro modelo estadístico (que es la traducción del diseño del experimento), incorpore ambos efectos (el del cultivo previo y el de la región).

4. 3. Datos

Para abordar el problema, agrupamos los cultivos previos en cuatro categorías: pastura, cereal, soja de primera y soja de segunda. Esto puede realizarse de múltiples formas en R, y lo que hacemos nosotros es recodificar la variable:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
dat.rend$region <- as.factor(dat.rend$region) # indicamos que region es un factor
dat.rend$cprevio <- as.factor(dat.rend$cprevio) # indicamos que cprev es un factor
levels(dat.rend$cprevio)

## [1] "gram_anual" "maiz"       "pastura"    "soja_1ra"   "soja_2da"
## [6] "trigo"      "verdeo"

# recodificamos los niveles de cprev
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="gram_anual"] <- "pastura"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="verdeo"] <- "pastura"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="trigo"] <- "cereal"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="maiz"] <- "cereal"

levels(dat.rend$cprevio)

## [1] "pastura"  "cereal"   "soja_1ra" "soja_2da"
```

Además, dado que existen pocos lotes en la región «d», removamos esta categoría del análisis:

```
dat.rend2 <- dat.rend[!dat.rend$region=="rd", ] # removemos los lotes de la región d
length(dat.rend2$region)

## [1] 99

levels(dat.rend2$region)

## [1] "ra" "rb" "rc" "rd"
```

Como vemos, aunque hemos removido los lotes de la región «d», el factor `region` sigue manteniendo la estructura de niveles original (se dice que la hereda). Entonces, le indicamos a R que el factor tiene tres niveles, y además cuales son:

```
dat.rend2$region <- with(dat.rend2, factor(region, levels=c("ra", "rb", "rc")))
levels(dat.rend2$region)

## [1] "ra" "rb" "rc"
```

Ahora sí. Trabajamos con dos factores, el cultivo previo que tiene cuatro niveles, y la región con tres niveles. En los modelos multifactoriales, los tratamientos son la combinación de los niveles de los factores. Por lo tanto, la combinación de niveles de ambos factores implica 12 tratamientos:

```
length(levels(dat.rend2$region)) # nº niveles region

## [1] 3

length(levels(dat.rend2$cprevio)) # nº niveles cprev

## [1] 4

length(levels(dat.rend2$region)) * length(levels(dat.rend2$cprevio)) # nº de tratam.

## [1] 12
```

4. 4. Diseño en bloques completos aleatorizados (DBCA)

El DBCA es un diseño multifactorial en el cual el segundo factor no es de interés, pero se lo incluye para controlar/separar la variabilidad no atribuida al factor que se quiere evaluar. Los bloques se establecen en áreas o períodos de tiempo relativamente homogéneos.[1] Las unidades experimentales se seleccionan aleatoriamente dentro de cada bloque (aleatorización restringida). Si el experimento es manipulativo, la asignación del tratamiento a cada unidad experimental se realiza aleatoriamente. Dentro del bloque no hay réplicas (cada bloque tiene una unidad experimental por nivel del factor de interés), lo cual, si bien es una manera *a priori* eficiente de reducir el error residual, no permite estimar la

[1] El bloque también puede ser un individuo u objeto.

interacción entre los efectos de los factores. Al no poder estimar estos efectos, la *ausencia interacción* es un supuesto extra de los modelos estadísticos de un DBCA.

En nuestro caso, queremos evaluar si el rendimiento del lote es afectado por el cultivo previo. Supongamos que aún no realizamos el muestreo y bloqueamos por región para separar la variabilidad debida al ambiente. Supongamos además que en cada región seleccionamos aleatoriamente un lote por cada tipo de cultivo previo.

4. 4. 1. Muestreo

Para el muestreo simulamos una selección aleatoria restringida de lotes:

```
dat.rend2 <- dat.rend2[order(dat.rend2$region, dat.rend2$cprevio), ]
View(dat.rend2)

set.seed(111) # para reproducir los resultados

# bloque 1
b11 <- dat.rend2[sample(which(dat.rend2$region=="ra" & dat.rend2$cprevio=="pastura"), 1)
, c(1,2,8,9)]
b12 <- dat.rend2[sample(which(dat.rend2$region=="ra" & dat.rend2$cprevio=="cereal"), 1),
c(1,2,8,9)]
b13 <- dat.rend2[sample(which(dat.rend2$region=="ra" & dat.rend2$cprevio=="soja_1ra"), 1
), c(1,2,8,9)]
b14 <- dat.rend2[sample(which(dat.rend2$region=="ra" & dat.rend2$cprevio=="soja_2da"), 1
), c(1,2,8,9)]

# bloque 2
b21 <- dat.rend2[sample(which(dat.rend2$region=="rb" & dat.rend2$cprevio=="pastura"), 1)
, c(1,2,8,9)]
b22 <- dat.rend2[sample(which(dat.rend2$region=="rb" & dat.rend2$cprevio=="cereal"), 1),
c(1,2,8,9)]
b23 <- dat.rend2[sample(which(dat.rend2$region=="rb" & dat.rend2$cprevio=="soja_1ra"), 1
), c(1,2,8,9)]
b24 <- dat.rend2[sample(which(dat.rend2$region=="rb" & dat.rend2$cprevio=="soja_2da"), 1
), c(1,2,8,9)]

# bloque 3
b31 <- dat.rend2[sample(which(dat.rend2$region=="rc" & dat.rend2$cprevio=="pastura"), 1)
, c(1,2,8,9)]
b32 <- dat.rend2[sample(which(dat.rend2$region=="rc" & dat.rend2$cprevio=="cereal"), 1),
c(1,2,8,9)]
b33 <- dat.rend2[sample(which(dat.rend2$region=="rc" & dat.rend2$cprevio=="soja_1ra"), 1
), c(1,2,8,9)]
b34 <- dat.rend2[sample(which(dat.rend2$region=="rc" & dat.rend2$cprevio=="soja_2da"), 1
), c(1,2,8,9)]

dat.rend.dbca <- rbind(b11, b12, b13, b14,
                       b21, b22, b23, b24,
                       b31, b32, b33, b34)
dat.rend.dbca

##     lote rendimiento  cprevio region
## 102  102        3588  pastura     ra
## 9      9        1972   cereal     ra
## 72    72        3650 soja_1ra     ra
## 76    76        2720 soja_2da     ra
## 5      5        2312  pastura     rb
## 20    20        2053   cereal     rb
## 50    50        3027 soja_1ra     rb
## 79    79        2078 soja_2da     rb
## 105  105        2031  pastura     rc
```

```
## 31     31         1556    cereal      rc
## 68     68         2640  soja_1ra      rc
## 89     89         2540  soja_2da      rc
```

4. 4. 2. Exploración de datos

A continuación, exploramos gráficamente los datos (es necesario notar que en el primer `plot()` aplicamos una paleta de verdes con la codificación hexadecimal del color):

```
#figura 4.1
par(mfrow=c(1,2))
plot(dat.rend.dbca$rendimiento ~ dat.rend.dbca$cprevio,
     ylab="Rendimiento (ka/ha)", xlab="Cultivo previo",
     names=c("pastura", "cereal", "soja 1ra", "soja 2da"),
     col=c("#bbf65e", "#9fe115", "#81a366", "#4d7902"),
     ylim=c(500,4500))
plot(dat.rend.dbca$rendimiento ~ dat.rend.dbca$region,
     ylab="Rendimiento (ka/ha)", xlab="Región",
     col=c("red", "green", "darkgrey"),
     ylim=c(500,4500))
```

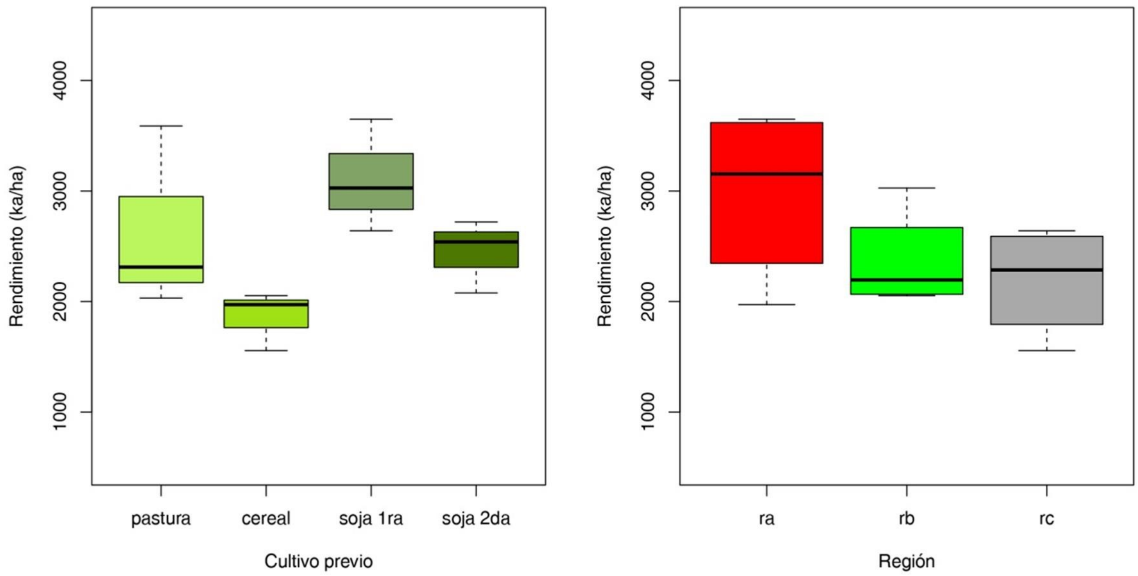

Figura 4. 1. Variabilidad del rendimiento de lotes con diferente cultivo previo (izquierda) en diferentes regiones (derecha)

Si queremos el rendimiento medio de cada nivel de cada factor:

```
tapply(dat.rend.dbca$rendimiento, dat.rend.dbca$cprevio, mean) # rend. medio x region

##  pastura   cereal soja_1ra soja_2da
## 2643.667 1860.333 3105.667 2446.000

tapply(dat.rend.dbca$rendimiento, dat.rend.dbca$region, mean) # rend. medio x cprev

##       ra      rb      rc
## 2982.50 2367.50 2191.75
```

En los DBCA es esencial explorar gráficamente si se cumple el supuesto de ausencia de interacción. Cuando no existe interacción, el efecto de un factor es independiente de los niveles del otro. Por ejemplo, si la soja de segunda perjudica el rendimiento del cultivo posterior, en ausencia de interacción el efecto (reducción del rendimiento) sería idéntico en cada una de las regiones. Gráficamente, esto implica líneas paralelas. Un gráfico para esta evaluación lo provee la función `interaction.plot()`:

```
# figura 4.2
par(mfrow=c(1,1))
with(dat.rend.dbca, interaction.plot(cprevio, region, rendimiento,
                                     xlab="Cultivo previo",
                                     ylab="Rendimiento (kg/ha)",
                                     col=c("red", "green", "gray50"),
                                     lwd=2, lty=1))
```

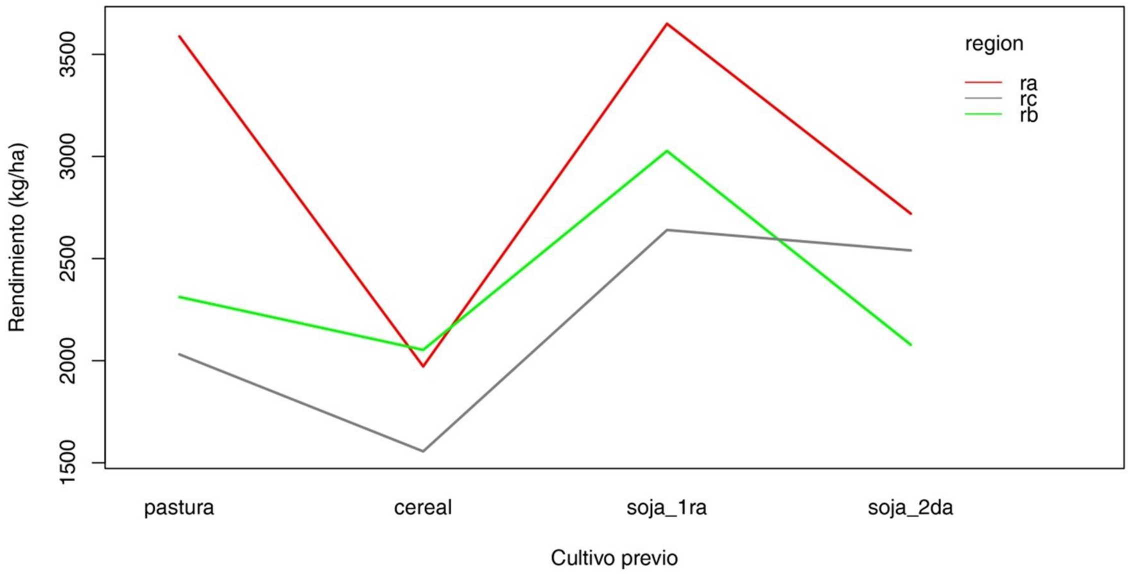

Figura 4. 2. Evaluación gráfica del supuesto de no interacción entre el cultivo previo (factor de interés) y la región (bloque)

El gráfico muestra rectas que siguen una misma tendencia, lo que implica que es razonable suponer que no existe interacción entre región y cultivo previo.

4. 4. 3. Modelo y ajuste

Planteamos un modelo multifactorial sin interacción (asumimos que el parámetro asociado es igual a cero):

$$r_i = \beta_0 + \beta_1 \times c_i + \beta_2 \times s_i + \beta_3 \times ss_i + \beta_4 \times rb_i + \beta_5 \times rc_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0;\ \sigma^2)_{independientes}$$

donde

r_i = rendimiento de girasol (kg ha^{-1}) del lote i (i = 1, 2, … N),
c_i = 1 si el lote tuvo cereal como cultivo previo y 0 en caso contrario,
s_i = 1 si el lote tuvo soja de primera como cultivo previo y 0 en caso contrario,
ss_i = 1 si el lote tuvo soja de segunda como cultivo previo y 0 en caso contrario,
rb_i = 1 si el lote se localiza en la región «b» y 0 en caso contrario,
rc_i = 1 si el lote se localiza en la región «c» y 0 en caso contrario,
ε_i = término de error (kg ha^{-1}).

En los modelos multifactoriales con una codificación *dummy*, la ordenada al origen es el promedio de la variable dependiente en uno de los tratamientos (combinación de niveles de los factores). En este caso, es el rendimiento medio de los lotes de la región «a» que previamente habían estado ocupados por una pastura:

$$r = \beta_0 + \beta_1 \times 0 + \beta_2 \times 0 + \beta_3 \times 0 + \beta_4 \times 0 + \beta_5 \times 0 = \beta_0$$

El resto de los parámetros de la componente lineal representa el efecto del cambio de nivel en el factor del que forman parte. Si queremos predecir el rendimiento medio de lotes de la región «b» con cereales como cultivo previo:

$$r = \beta_0 + \beta_1 \times 1 + \beta_2 \times 0 + \beta_3 \times 0 + \beta_4 \times 1 + \beta_5 \times 0 = \beta_0 + \beta_1 + \beta_4$$

El rendimiento medio de los 12 tratamientos se obtiene como:

β_0 = lotes de la región «a» con pastura como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_1$ = lotes de la región «a» con cereal como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_2$ = lotes de la región «a» con soja de primera como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_3$ = lotes de la región «a» con soja de segunda como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_4$ = lotes de la región «b» con pastura como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_1 + \beta_4$ = lotes de la región «b» con cereal como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_2 + \beta_4$ = lotes de la región «b» con soja de primera como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_3 + \beta_4$ = lotes de la región «b» con soja de segunda como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_5$ = lotes de la región «c» con pastura como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_1 + \beta_5$ = lotes de la región «c» con cereal como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_2 + \beta_5$ = lotes de la región «c» con soja de primera como cultivo previo (kg ha^{-1}),
$\beta_0 + \beta_3 + \beta_5$ = lotes de la región «c» con soja de segunda como cultivo previo (kg ha^{-1}).

En ausencia de interacción los efectos de los factores se consideran aditivos. Es así que para ajustar un modelo sin interacción en R usamos el operador `+`:

```
m.dbca <- lm(rendimiento ~ cprevio + region,
             data=dat.rend.dbca) # ajuste del modelo
m.dbca # modelo ajustado

##
## Call:
```

```
## lm(formula = rendimiento ~ cprevio + region, data = dat.rend.dbca)
##
## Coefficients:
##     (Intercept)    cpreviocereal  cpreviosoja_1ra  cpreviosoja_2da
##          3112.3           -783.3            462.0           -197.7
##        regionrb         regionrc
##          -615.0           -790.8
```

La salida nos permite informar la ecuación estimada para predecir el rendimiento medio:

$$\hat{r} \;=\; 1454{,}3 \;+\; 194{,}3 \times c \;+\; 698 \times s \;+\; 235{,}7 \times ss \;+\; 791{,}7 \times rb \;+\; 1520{,}5 \times rc$$

Informamos los intervalos de confianza de los coeficientes:

```
confint(m.dbca) # IC de las estimaciones

##                      2.5 %     97.5 %
## (Intercept)      2450.0721 3774.42787
## cpreviocereal   -1547.9505  -18.71619
## cpreviosoja_1ra  -302.6171 1226.61714
## cpreviosoja_2da  -962.2838  566.95047
## regionrb        -1277.1779   47.17787
## regionrc        -1452.9279 -128.57213
```

De la misma manera que en el caso del modelo unifactorial, el modelo puede ser parametrizado con diferentes codificaciones. Lógicamente, la interpretación de los parámetros y su valor estimado cambia. Por ejemplo, aplicando una codificación por desviación a ambos factores:

```
m.dbca.s <- lm(rendimiento ~ cprevio + region,
                             contrasts=list(cprevio="contr.sum", region="contr.sum"),
                             data=dat.rend.dbca) # ajuste del modelo

m.dbca.s # modelo ajustado con codificación por desviación

##
## Call:
## lm(formula = rendimiento ~ cprevio + region, data = dat.rend.dbca,
##     contrasts = list(cprevio = "contr.sum", region = "contr.sum"))
##
## Coefficients:
## (Intercept)     cprevio1     cprevio2     cprevio3      region1      region2
##      2513.9        129.8       -653.6        591.7        468.6       -146.4

confint(m.dbca.s) # IC de las estimaciones

##                   2.5 %    97.5 %
## (Intercept)  2243.58368 2784.2496
## cprevio1     -338.48046  597.9805
## cprevio2    -1121.81379 -185.3529
## cprevio3      123.51954 1059.9805
## region1        86.27476  850.8919
## region2      -528.72524  235.8919
```

encontramos que el intercepto es la gran media:

```
medias <- tapply(dat.rend.dbca$rendimiento,
                 list(dat.rend.dbca$cprevio, dat.rend.dbca$region),
                 mean) # medias por cultivo previo y región
medias

##            ra   rb   rc
## pastura  3588 2312 2031
## cereal   1972 2053 1556
## soja_1ra 3650 3027 2640
## soja_2da 2720 2078 2540

coef(m.dbca.s) # coeficientes estimados

## (Intercept)    cprevio1    cprevio2    cprevio3     region1     region2
##   2513.9167    129.7500   -653.5833    591.7500    468.5833   -146.4167

coef(m.dbca.s)[1] # b0

## (Intercept)
##    2513.917

mean(medias) # gran media

## [1] 2513.917
```

Los efectos del cultivo previo representan las diferencias de los tres primeros cultivos con la gran media. Aquí R respeta el orden alfabético de los datos originales por lo que conviene revisar cómo se ordena `cprevio` (recordemos que lo recategorizamos):

```
coef(m.dbca.s)[2] # b1

## cprevio1
##   129.75

levels(dat.rend.dbca$cprevio)

## [1] "pastura"  "cereal"   "soja_1ra" "soja_2da"

mean(medias[1, ]) - mean(medias) # media pastura - gran media

## [1] 129.75

coef(m.dbca.s)[3] # b2

##  cprevio2
## -653.5833

mean(medias[2, ]) - mean(medias) # media cereal - gran media

## [1] -653.5833

coef(m.dbca.s)[4] # b3

## cprevio3
##   591.75

mean(medias[3, ]) - mean(medias) # media soja 1ra - gran media

## [1] 591.75
```

De la misma manera, los efectos del bloque (`region`) son diferencias con la gran media:

```
coef(m.dbca.s)[5] # b4

##  region1
## 468.5833

mean(medias[ ,1]) - mean(medias) # media ra - gran media

## [1] 468.5833

coef(m.dbca.s)[6] # b5

##   region2
## -146.4167

mean(medias[ ,2]) - mean(medias) # media rb - gran media

## [1] -146.4167
```

4. 4. 4. ANOVA y comparaciones múltiples

Nuestro modelo nos permite separar la variabilidad del rendimiento entre lotes en tres fuentes de variación, que en la jerga estadística les llamamos *tratamiento* (cultivo previo), *bloque* (región) y *residual* (el resto de las causas que afectan al rendimiento):

```
anova(m.dbca) # ANOVA

## Analysis of Variance Table
##
## Response: rendimiento
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## cprevio    3 2396361  798787  5.4537 0.03775 *
## region     2 1379198  689599  4.7082 0.05895 .
## Residuals  6  878808  146468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Obtenemos el mismo ANOVA con una codificación por desviación:

```
anova(m.dbca.s) # ANOVA modelo con codificación por desviación

## Analysis of Variance Table
##
## Response: rendimiento
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## cprevio    3 2396361  798787  5.4537 0.03775 *
## region     2 1379198  689599  4.7082 0.05895 .
## Residuals  6  878808  146468
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El valor p del estadístico F en el cultivo previo, el factor de interés, es significativo considerando un α de 0,05. Esto quiere decir que tenemos evidencia de que el cultivo previo afecta el rendimiento. Por lo tanto, procedemos a realizar las comparaciones múltiples entre los cuatro tipos de cultivos, en este caso mediante un test de Tukey:

```
library(agricolae)
test.cp <- HSD.test(m.dbca, "cprevio", console=TRUE)
```

```
##
## Study: m.dbca ~ "cprevio"
##
## HSD Test for rendimiento
##
## Mean Square Error:  146468
##
## cprevio,  means
##
##           rendimiento      std r  Min  Max
## cereal       1860.333 266.6540 3 1556 2053
## pastura      2643.667 829.7978 3 2031 3588
## soja_1ra     3105.667 509.5747 3 2640 3650
## soja_2da     2446.000 331.1616 3 2078 2720
##
## Alpha: 0.05 ; DF Error: 6
## Critical Value of Studentized Range: 4.895599
##
## Minimun Significant Difference: 1081.724
##
## Treatments with the same letter are not significantly different.
##
##           rendimiento groups
## soja_1ra     3105.667      a
## pastura      2643.667     ab
## soja_2da     2446.000     ab
## cereal       1860.333      b
```

Lo que nos indica este análisis es que los lotes en los que el cultivo fue precedido por soja de primera en promedio rindieron más que aquellos en los que el cultivo previo era un cereal. Gráficamente:

```
# figura 4.3
plot(test.cp, variation="SE",
     main="", ylim=c(1000, 4500),
     ylab="Rendimiento (kg/ha)", xlab="Cultivo previo")
```

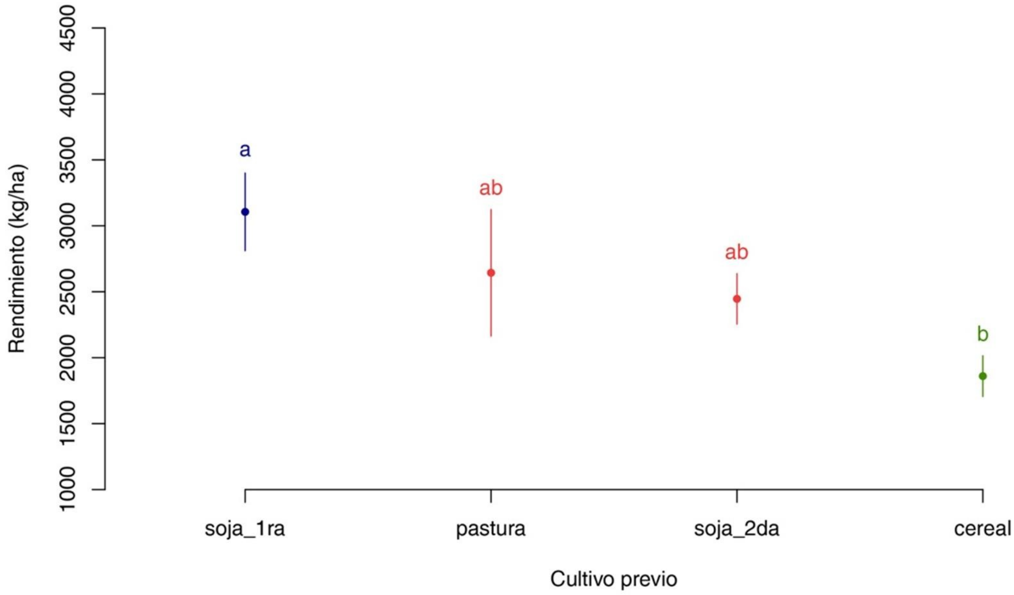

Figura 4. 3. Rendimiento medio (predicción puntual y error estándar) de girasol con diferentes cultivos previos
Nota. Las letras informan qué medias difieren de acuerdo a un test de Tukey.

4. 4. 5. Bondad de ajuste y eficiencia del bloqueo

Una de las grandes ventajas del DBCA es que permite sustraer parte de la variabilidad residual de modo de incrementar la capacidad de detectar el efecto del factor de interés sin aumentar el tamaño de la muestra. Ajustamos un modelo sin considerar el efecto de la región (bloque) y comparamos las sumas de cuadrados residuales:

```
m.cprev <- lm(rendimiento ~ cprevio, data=dat.rend.dbca) # modelo sin el bloque
anova(m.cprev)

## Analysis of Variance Table
##
## Response: rendimiento
##           Df  Sum Sq Mean Sq F value Pr(>F)
## cprevio    3 2396361  798787  2.8301 0.1064
## Residuals  8 2258006  282251
```

La suma de cuadrados (SC) correspondiente al cultivo previo es igual en ambos modelos, y la SC total, lógicamente, también. Por lo tanto, la SC residual de `m.cprev` es igual a la SC residual más la SC de la región en `m.dbca`:

```
# Sumas de cuadrados
anova(m.dbca)[2,2] + anova(m.dbca)[3,2] # SCB + SCR

## [1] 2258006

anova(m.cprev)[2,2] # SCR

## [1] 2258006
```

Esto quiere decir que la región absorbe parte de la variabilidad que no es explicada por el cultivo previo. En efecto, al bloquear mejoramos notablemente la bondad de ajuste:

```
# Error estándar residual
summary(m.dbca)$sigma # con el bloque

## [1] 382.7113

summary(m.cprev)$sigma # sin el bloque

## [1] 531.2728

# Coeficiente de determinación (r2)
summary(m.dbca)$r.squared # con el bloque

## [1] 0.8111864

summary(m.cprev)$r.squared # sin el bloque

## [1] 0.5148629
```

Ahora bien, los bloques consumen grados de libertad:

```
# Grados de libertad (gl)
anova(m.dbca)[2,1] + anova(m.dbca)[3,1] # gl bloque + gl error

## [1] 8

anova(m.cprev)[2,1] # gl error

## [1] 8
```

Vemos que si no hubiéramos efectuado un diseño en bloques, los grados de libertad para dividir sobre la SCE hubieran sido ocho (CM = SC/gl). Al bloquear por región, un factor de tres niveles, consumimos dos grados de libertad. Por lo tanto, si la reducción de la SCE que obtenemos por bloquear no fuera sustancial, estaríamos reduciendo la capacidad de detectar diferencias, es decir, perderíamos potencia. En este caso, podríamos pensar que evitamos la pérdida de potencia removiendo el factor de bloqueo (como hicimos con `m.cprev`). La respuesta es que no. Es muy importante tener en cuenta que el modelo estadístico debe reflejar el proceso que genera los datos, en este caso la aleatorización restringida de las unidades experimentales que hemos aplicado con el diseño en bloques. Si el factor de bloqueo no fuera parte del modelo estaríamos violando el supuesto de independencia.

4. 4. 6. Validación de supuestos

El supuesto de no interacción fue evaluado antes gráficamente (figura 4. 2). Evaluemos el resto de los supuestos (figura 4. 4):

```
residuos <- resid(m.dbca)
predichos <- fitted(m.dbca)

# figura 3.4
par(mfrow=c(2,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend.dbca$region,
     xlab="Región", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend.dbca$cprevio,
     xlab="Cultivo previo", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

# Normalidad
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")
```

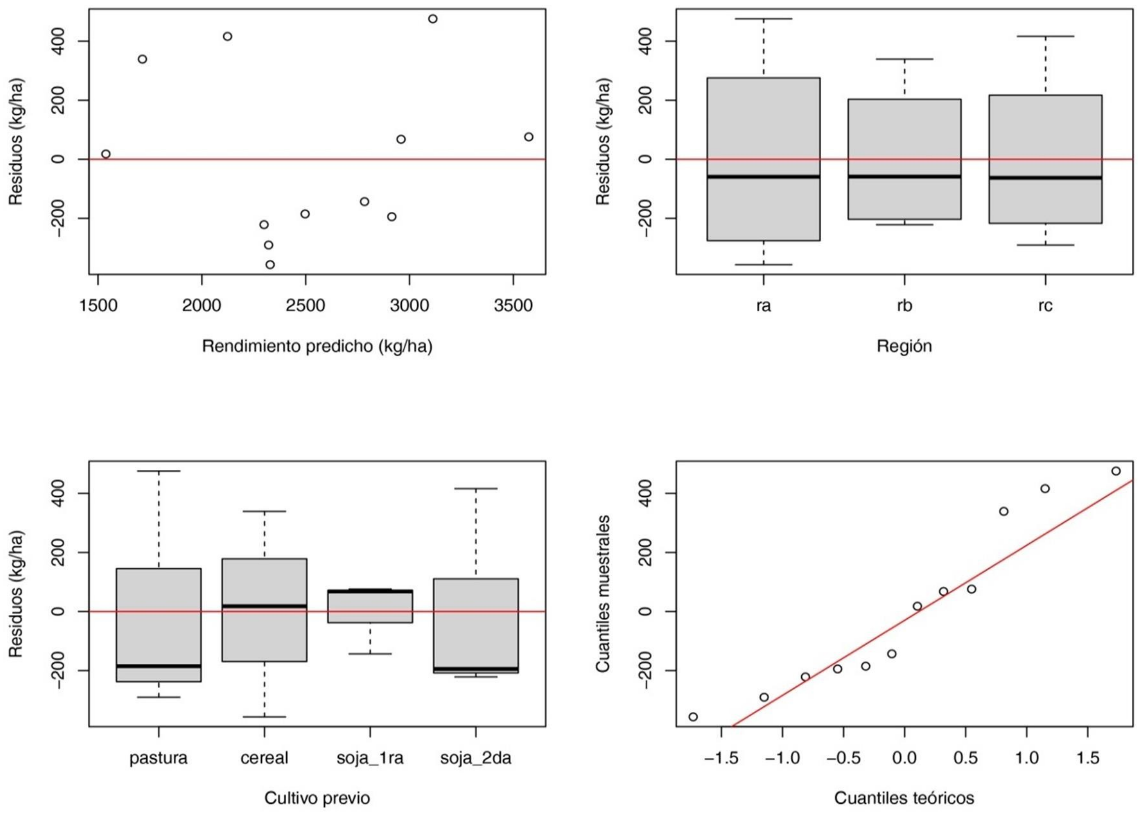

Figura 4. 4. Evaluación gráfica de los supuestos (linealidad, independencia, homocedasticidad y normalidad) del modelo de dos factores (región y cultivo previo) sin interacción

```
# - Test Shapiro-Wilk (normalidad)
shapiro.test(residuos)

##
##  Shapiro-Wilk normality test
##
## data:  residuos
## W = 0.90859, p-value = 0.2047
```

4. 5. Diseño factorial

En los diseños factoriales, en general todos los factores son de interés. A diferencia del DBCA, en este diseño contamos con replicas dentro de cada tratamiento. Por lo tanto, es posible estimar los efectos de interacción entre los niveles de los factores. Incluir la interacción flexibiliza los patrones que pueden modelarse, además de que su evaluación a menudo es conceptualmente relevante.

Una pregunta que podemos responder con este diseño es si existe efecto del cultivo previo y, en caso de que así sea, si este depende de (varía con) la región. Esto es, en efecto, el concepto de interacción. Para trabajar con el diseño factorial usamos el conjunto de datos previo a la simulación del muestreo en bloques que guardamos en el objeto `dat.rend2`:

```
length(dat.rend2$region)

## [1] 99

levels(dat.rend2$region)

## [1] "ra" "rb" "rc"

levels(dat.rend2$cprevio)

## [1] "pastura"  "cereal"   "soja_1ra" "soja_2da"
```

4. 5. 1. Exploración de datos

Una información de relevancia es el número de repeticiones por tratamiento. Veamos:

```
xtabs(~cprevio + region, data=dat.rend2)

##           region
## cprevio    ra rb rc
##   pastura  11  4  4
##   cereal    5 16 15
##   soja_1ra  4  6 18
##   soja_2da  3  3 10
```

El paquete `FSA` contiene una función para realizar un resumen descriptivo de los datos en diseños factoriales:

```
library(FSA) # recordar previamente instalar el paquete

## ## FSA v0.9.3. See citation('FSA') if used in publication.
## ## Run fishR() for related website and fishR('IFAR') for related book.

res <- Summarize(rendimiento ~ region + cprevio, data=dat.rend2)
res # resumen descriptivo por tratamiento

##    region  cprevio  n     mean       sd  min      Q1 median     Q3  max
## 1      ra  pastura 11 2678.000 615.3550 1925 2265.50 2512.0 3174.5 3588
## 2      rb  pastura  4 2375.250 252.7573 2051 2246.75 2410.0 2538.5 2630
## 3      rc  pastura  4 2010.500 201.7284 1743 1959.00 2033.0 2084.5 2233
```

```
## 4      ra   cereal  5 2453.600 640.0307 1770 1972.00 2273.0 3103.0 3150
## 5      rb   cereal 16 2380.062 271.7191 2053 2177.50 2265.0 2564.0 2985
## 6      rc   cereal 15 1974.133 321.3038 1556 1810.50 1925.0 2267.5 2590
## 7      ra soja_1ra  4 3237.000 595.5530 2353 3175.00 3472.5 3534.5 3650
## 8      rb soja_1ra  6 2406.167 353.5265 2100 2130.50 2354.0 2498.0 3027
## 9      rc soja_1ra 18 2617.778 526.3013 1450 2263.25 2682.0 2892.5 3592
## 10     ra soja_2da  3 2988.333 443.2926 2720 2732.50 2745.0 3122.5 3500
## 11     rb soja_2da  3 2276.333 202.0998 2078 2173.50 2269.0 2375.5 2482
## 12     rc soja_2da 10 2347.400 562.3179 1400 2064.75 2248.5 2517.5 3381
```

Aquí, la cantidad de repeticiones por tratamiento es informada en la columna `n`. A continuación, exploramos gráficamente la variabilidad del rendimiento según el cultivo previo en las diferentes regiones (figura 4. 5) y complementamos esta información con un gráfico de interacción (figura 4. 6):

```
# figura 4.5
par(mfrow=c(1,1))
boxplot(rendimiento ~ cprevio + region, data = dat.rend2,
        xlab="Cultivo previo", ylab="Rendimiento (Kg/ha)",
        names=rep(c("p", "c", "s", "ss"), 3),
        at = c(1:4, 7:10, 13:16),
        col=rep(c("#bbf65e", "#9fe115", "#81a366", "#4d7902"),3),
        ylim=c(500,4500))
abline(v=5.5)
abline(v=11.5)
text("Región A", x=0.85, y=4400)
text("Región B", x=6.45, y=4400)
text("Región C", x=12.45, y=4400)
```

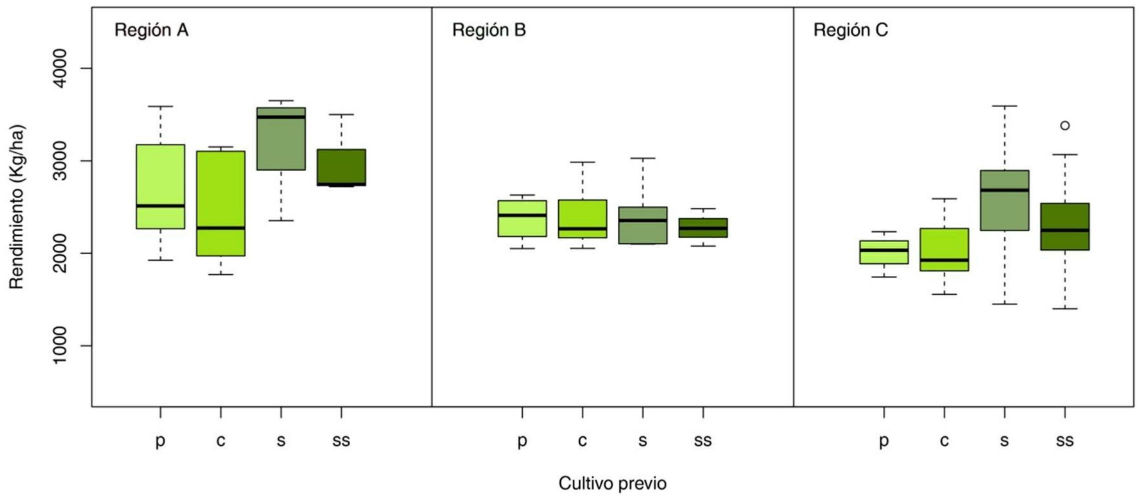

Figura 4. 5. Variabilidad del rendimiento de lotes con diferente cultivo previo (izquierda) dentro de cada región

```
# medias muestrales
with(dat.rend2, interaction.plot(cprevio, region, rendimiento,
                                 xlab="Cultivo previo",
                                 ylab="Rendimiento (kg/ha)",
                                 ylim=range(rendimiento),
                                 type="b", pch=c(16,16,16),
                                 col=c("red", "green", "gray50"),
                                 cex=2)) # figura 4.6
```

```
# agregamos los rendimientos de cada lote (figura 4.6)
with(dat.rend2, points(jitter(as.numeric(cprevio), factor=0.5),
                       rendimiento, col=ifelse(region == "ra", "red",
                                        ifelse(region == "rb", "green",
                                               "gray50")), cex=0.8))
```

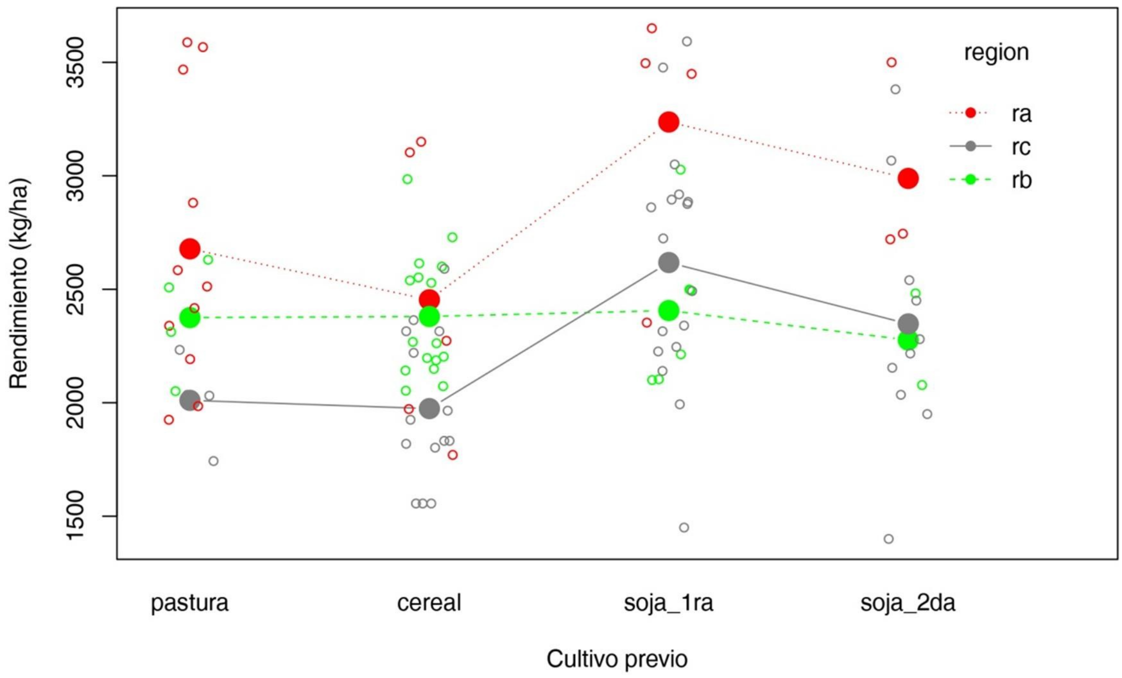

Figura 4. 6. Evaluación gráfica de la interacción entre el efecto del cultivo previo y la región **Nota.** Los puntos sin relleno indican el rendimiento de un lote y los sólidos la media de cada tratamiento.

Alternativamente, podemos usar la información obtenida con `Summarize()` que guardamos en el objeto `res` y generar un gráfico de medias con errores estándar (figura 4. 7) desde la función `qplot()` de `ggplot2`.

```
# figura 4.7
library(ggplot2)

res$ee <- res$sd/sqrt(res$n) # error estándar del tratamiento
qplot(x = cprevio, y = mean,
      color = region,
      data =  res) +
  geom_errorbar(aes(ymin = mean - ee,
                    ymax = mean + ee,
                    width = 0.05), size=0.6) +
  xlab("Cultivo previo") + ylab("Rendimiento (kg/ha)")
```

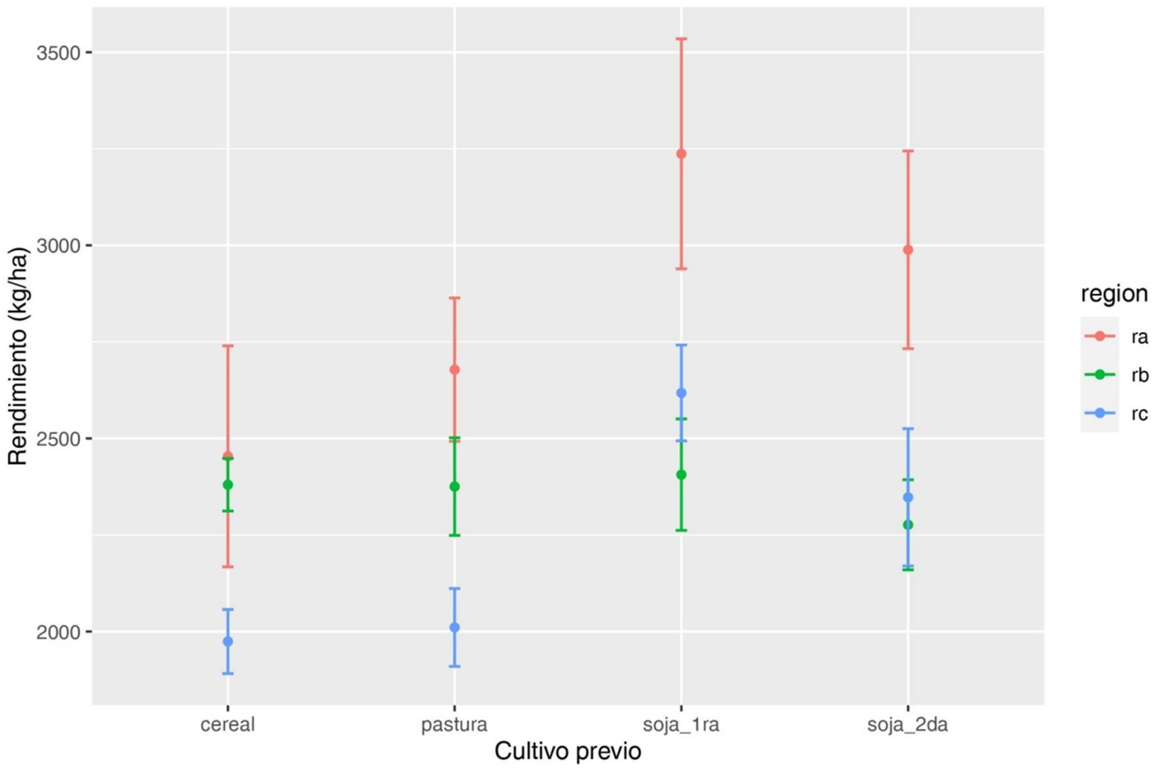

Figura 4. 7. Gráfica de rendimientos medios por tratamiento incluyendo barras de error estándar

Los gráficos muestran que el efecto del cultivo previo parece ser similar en las tres regiones evaluadas.

4. 5. 2. Modelo y ajuste

El modelo estadístico ahora incorpora los efectos de interacción entre los niveles del cultivo previo y la región (modelo multifactorial con interacción):

$$r_i = \beta_0 + \beta_1 \times c_i + \beta_2 \times s_i + \beta_3 \times ss_i + \beta_4 \times rb_i + \beta_5 \times rc_i + \beta_6 \times c_i \times rb_i + \beta_7 \times s_i \times rb_i + \beta_8 \times ss_i \times rb_i + \beta_9 \times c_i \times rc_i + \beta_{10} \times s_i \times rc_i + \beta_{11} \times ss_i \times rc_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0;\, \sigma^2)_{independientes}$$

Donde

r_i = rendimiento de girasol (kg ha^{-1}) del lote *i* (*i* = 1, 2, … N),
c_i = 1 si el lote tuvo cereal como cultivo previo y 0 en caso contrario,
s_i = 1 si el lote tuvo soja de primera como cultivo previo y 0 en caso contrario,
ss_i = 1 si el lote tuvo soja de segunda como cultivo previo y 0 en caso contrario,
rb_i = 1 si el lote se localiza en la región b y 0 en caso contrario,
rc_i = 1 si el lote se localiza en la región c y 0 en caso contrario,
ε_i = término de error (kg ha^{-1}).

En este modelo debemos estimar 12 parámetros, 11 de la componente determinística más la varianza (se debe notar que un factor de cuatro niveles interactuando con uno de tres niveles agrega seis parámetros respecto al modelo sin interacción). Para indicar que queremos un modelo con interacción usamos el operador `*`:

```
m.fact <- lm(rendimiento ~ cprevio * region,
             data=dat.rend2) # ajuste del modelo
```

Pedimos a R los coeficientes estimados y sus intervalos de confianza:

```
coef(m.fact) # estimación puntual

##              (Intercept)             cpreviocereal           cpreviosoja_1ra
##               2678.00000                -224.40000                 559.00000
##          cpreviosoja_2da                  regionrb                  regionrc
##                310.33333                -302.75000                -667.50000
##   cpreviocereal:regionrb cpreviosoja_1ra:regionrb cpreviosoja_2da:regionrb
##                229.21250                -528.08333                -409.25000
##   cpreviocereal:regionrc cpreviosoja_1ra:regionrc cpreviosoja_2da:regionrc
##                188.03333                  48.27778                  26.56667

confint(m.fact) # IC de las estimaciones

##                                 2.5 %    97.5 %
## (Intercept)               2405.33322 2950.6668
## cpreviocereal             -712.16117  263.3612
## cpreviosoja_1ra             30.98305 1087.0170
## cpreviosoja_2da           -278.69424  899.3609
## regionrb                  -830.76695  225.2670
## regionrc                 -1195.51695 -139.4830
## cpreviocereal:regionrb    -473.26836  931.6934
## cpreviosoja_1ra:regionrb -1315.20453  259.0379
## cpreviosoja_2da:regionrb -1317.00245  498.5024
## cpreviocereal:regionrc    -516.86875  892.9354
## cpreviosoja_1ra:regionrc  -678.83364  775.3892
## cpreviosoja_2da:regionrc  -769.16576  822.2991
```

Podemos usar el siguiente código para clarificar los resultados anteriores e informar la ecuación de predicción estimada:

```
round(cbind("β est."=coef(m.fact), confint(m.fact)), 1) # más claro

##                          β est.   2.5 % 97.5 %
## (Intercept)              2678.0  2405.3 2950.7
## cpreviocereal            -224.4  -712.2  263.4
## cpreviosoja_1ra           559.0    31.0 1087.0
## cpreviosoja_2da           310.3  -278.7  899.4
## regionrb                 -302.7  -830.8  225.3
## regionrc                 -667.5 -1195.5 -139.5
## cpreviocereal:regionrb    229.2  -473.3  931.7
## cpreviosoja_1ra:regionrb -528.1 -1315.2  259.0
## cpreviosoja_2da:regionrb -409.3 -1317.0  498.5
## cpreviocereal:regionrc    188.0  -516.9  892.9
## cpreviosoja_1ra:regionrc   48.3  -678.8  775.4
## cpreviosoja_2da:regionrc   26.6  -769.2  822.3
```

$$\hat{r} = 2010{,}5 - 36{,}4 \times c + 607{,}3 \times s + 336{,}9 \times ss + 364{,}7 \times rb + 667{,}5 \times rc + 41{,}2 \times c \times rb - 576{,}4 \times s \times rb - 435{,}8 \times s \times rb - 188{,}0 \times c \times rc + 210{,}7 \times s \times rc + 54{,}6 \times ss \times rc$$

Exploramos las primeras filas de la matriz del modelo:

```
head(model.matrix(~cprevio * region, data=dat.rend2)) # matriz modelo (1ras filas)

##   (Intercept) cpreviocereal cpreviosoja_1ra cpreviosoja_2da regionrb regionrc
## 1           1             0               0               0        0        0
## 2           1             0               0               0        0        0
## 3           1             0               0               0        0        0
## 4           1             0               0               0        0        0
## 5           1             0               0               0        1        0
## 6           1             1               0               0        0        0
##   cpreviocereal:regionrb cpreviosoja_1ra:regionrb cpreviosoja_2da:regionrb
## 1                      0                        0                        0
## 2                      0                        0                        0
## 3                      0                        0                        0
## 4                      0                        0                        0
## 5                      0                        0                        0
## 6                      0                        0                        0
##   cpreviocereal:regionrc cpreviosoja_1ra:regionrc cpreviosoja_2da:regionrc
## 1                      0                        0                        0
## 2                      0                        0                        0
## 3                      0                        0                        0
## 4                      0                        0                        0
## 5                      0                        0                        0
## 6                      0                        0                        0
```

Si observamos el primer lote de la muestra, encontramos que se localiza en la región «a» y que previo a la siembra estaba ocupado por una pastura:

```
dat.rend2[1, c(8:9)] # primer lote de la muestra

##   cprevio region
## 1 pastura     ra
```

Entonces, para predecir el rendimiento medio de un lote con estas características calculamos:

$$\hat{r} = 2010{,}5 - 36{,}4 \times 0 + 607{,}3 \times 0 + 336{,}9 \times 0 + 364{,}7 \times 0 + 667{,}5 \times 1 + 41{,}2 \times (0 \times 0) - 576{,}4 \times (0 \times 0) - 435{,}8 \times (0 \times 0) - 188{,}0 \times (0 \times 1) + 210{,}7 \times (0 \times 1) + 54{,}6 \times (0 \times 1)$$

Así lo indica la primera fila de la matriz:

```
model.matrix(~cprevio * region, data=dat.rend2)[1, ] # 1ra fila matriz del modelo
```

```
##              (Intercept)            cpreviocereal          cpreviosoja_1ra
##                        1                        0                        0
##          cpreviosoja_2da                 regionrb                 regionrc
##                        0                        0                        0
##   cpreviocereal:regionrb cpreviosoja_1ra:regionrb cpreviosoja_2da:regionrb
##                        0                        0                        0
##   cpreviocereal:regionrc cpreviosoja_1ra:regionrc cpreviosoja_2da:regionrc
##                        0                        0                        0
```

Para lotes con estas características el modelo predice un rendimiento medio de 2010,5 kg ha^{-1} + 667,5 kg ha^{-1} = 2678 kg ha^{-1}:

```
predict.lm(m.fact)[1]
```

```
##    1
## 2678
```

El paquete `effects` permite mostrar esta información gráficamente (figura 4. 8) como alternativa a las figuras 4. 6 y 4. 7:

```
library(effects)
```

```
## Loading required package: carData
```

```
## lattice theme set by effectsTheme()
## See ?effectsTheme for details.
```

```
plot(effect("cprevio:region", m.fact)) # figura 4.8
```

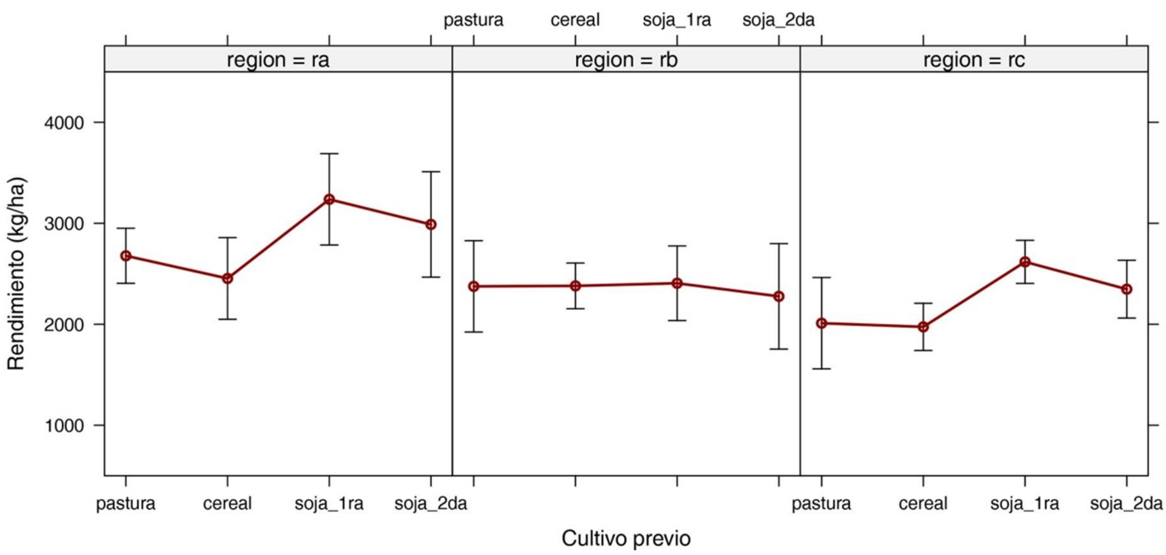

Figura 4. 8. Estimación puntual (puntos) e intervalos de confianza (barras) del rendimiento medio en cada cultivo previo y región

El rendimiento de girasol estimado por el modelo para los lotes de la región «a» con pastura como cultivo previo se encuentra en el punto más a la izquierda del primer gráfico de la figura 4. 8. Como se observa en las figuras 4. 6, 4. 7 y 4. 8, este modelo realiza 12 predicciones, las cuales constituyen los rendimientos promedio predichos para los 12 tratamientos (4 cultivos previos por 3 regiones). Veamos una forma de obtener sus intervalos de confianza:

```
regiones <- rep(c("ra", "rb", "rc"), each=4)
cp <- rep(c("pastura", "cereal", "soja_1ra", "soja_2da"), times=3)
predict.lm(m.fact, data.frame(region=regiones, cprevio=cp), interval="confidence")

##         fit      lwr      upr
## 1  2678.000 2405.333 2950.667
## 2  2453.600 2049.170 2858.030
## 3  3237.000 2784.833 3689.167
## 4  2988.333 2466.216 3510.450
## 5  2375.250 1923.083 2827.417
## 6  2380.063 2153.979 2606.146
## 7  2406.167 2036.974 2775.359
## 8  2276.333 1754.216 2798.450
## 9  2010.500 1558.333 2462.667
## 10 1974.133 1740.635 2207.631
## 11 2617.778 2404.624 2830.931
## 12 2347.400 2061.425 2633.375
```

Para mostrarlo de manera más clara:

```
pred.ic <- predict.lm(m.fact, data.frame(region=regiones, cprevio=cp),
                      interval="confidence")
ic <- data.frame(cp, regiones, round(pred.ic, 1))
ic

##          cp regiones    fit    lwr    upr
## 1   pastura       ra 2678.0 2405.3 2950.7
## 2    cereal       ra 2453.6 2049.2 2858.0
## 3  soja_1ra       ra 3237.0 2784.8 3689.2
## 4  soja_2da       ra 2988.3 2466.2 3510.5
## 5   pastura       rb 2375.3 1923.1 2827.4
## 6    cereal       rb 2380.1 2154.0 2606.1
## 7  soja_1ra       rb 2406.2 2037.0 2775.4
## 8  soja_2da       rb 2276.3 1754.2 2798.5
## 9   pastura       rc 2010.5 1558.3 2462.7
## 10   cereal       rc 1974.1 1740.6 2207.6
## 11 soja_1ra       rc 2617.8 2404.6 2830.9
## 12 soja_2da       rc 2347.4 2061.4 2633.4
```

Y ahora los intervalos de predicción:

```
pred.ip <- predict.lm(m.fact, data.frame(region=regiones, cprevio=cp),
                      interval="prediction")
ip <- data.frame(cp, regiones, round(pred.ip, 1))
ip
```

```
##          cp regiones    fit    lwr    upr
## 1   pastura       ra 2678.0 1733.5 3622.5
## 2    cereal       ra 2453.6 1463.0 3444.2
## 3  soja_1ra       ra 3237.0 2225.9 4248.1
## 4  soja_2da       ra 2988.3 1944.1 4032.6
## 5   pastura       rb 2375.3 1364.2 3386.3
## 6    cereal       rb 2380.1 1447.9 3312.2
## 7  soja_1ra       rb 2406.2 1429.4 3383.0
## 8  soja_2da       rb 2276.3 1232.1 3320.6
## 9   pastura       rc 2010.5  999.4 3021.6
## 10   cereal       rc 1974.1 1040.1 2908.1
## 11 soja_1ra       rc 2617.8 1688.7 3546.9
## 12 soja_2da       rc 2347.4 1398.9 3295.9
```

4. 5. 3. ANOVA y comparaciones múltiples

El diseño factorial incluye la interacción como fuente de variabilidad. En términos del problema, separamos la variabilidad en el rendimiento en cuatro fuentes: el efecto principal del cultivo previo, el efecto principal de la región, la interacción entre cultivo previo y región, y el resto de las causas no consideradas (el error):

```
anova(m.fact) # ANOVA del modelo con interacción entre cprev y region
```

```
## Analysis of Variance Table
##
## Response: rendimiento
##                Df   Sum Sq Mean Sq F value   Pr(>F)
## cprevio         3  3108628 1036209  5.0056 0.003001 **
## region          2  4162407 2081203 10.0535 0.000118 ***
## cprevio:region  6  1598416  266403  1.2869 0.271709
## Residuals      87 18010045  207012
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El ANOVA nos indica que el efecto de interacción no es significativo, en concordancia con lo que se observa en los gráficos exploratorios. Solo en caso de descartar la interacción es que se evalúan los efectos principales. Así, vemos que el rendimiento es afectado por el cultivo previo y por la región de manera aditiva (ambos efectos son significativos). Esto quiere decir que existe un efecto del cultivo previo y este no depende de la región en la que se encuentre el lote. Al no haber detectado interacción, realizamos comparaciones múltiples para cada efecto principal por separado:

```
library(agricolae)
tukey.reg <- HSD.test(m.fact, "region", console=TRUE) # entre regiones

##
## Study: m.fact ~ "region"
##
## HSD Test for rendimiento
##
## Mean Square Error:  207012
##
## region,  means
##
##    rendimiento      std  r  Min  Max
## ra     2766.913 621.2537 23 1770 3650
## rb     2374.069 269.9960 29 2051 3027
## rc     2303.149 529.3526 47 1400 3592
##
## Alpha: 0.05 ; DF Error: 87
## Critical Value of Studentized Range: 3.372163
##
## Groups according to probability of means differences and alpha level( 0.05 )
##
## Treatments with the same letter are not significantly different.
##
##    rendimiento groups
## ra     2766.913      a
## rb     2374.069      b
## rc     2303.149      b

tukey.cp <- HSD.test(m.fact, "cprevio", console=TRUE) # entre cultivos previos

##
## Study: m.fact ~ "cprevio"
##
## HSD Test for rendimiento
##
## Mean Square Error:  207012
##
## cprevio,  means
##
##          rendimiento      std  r  Min  Max
## cereal      2221.139 406.4061 36 1556 3150
## pastura     2473.737 550.5790 19 1743 3588
## soja_1ra    2660.893 549.3500 28 1450 3650
## soja_2da    2454.250 540.7023 16 1400 3500
##
## Alpha: 0.05 ; DF Error: 87
## Critical Value of Studentized Range: 3.704375
##
## Groups according to probability of means differences and alpha level( 0.05 )
##
## Treatments with the same letter are not significantly different.
##
##          rendimiento groups
## soja_1ra    2660.893      a
## pastura     2473.737     ab
## soja_2da    2454.250     ab
## cereal      2221.139      b
```

Las comparaciones múltiples nos indican que, en promedio, los lotes precedidos por soja rindieron más que aquellos en los que el cultivo previo fue un cereal. Además, los lotes de la región «a» alcanzaron mayores rindes que las regiones «b» y «c». Gráficamente (figura 4. 9):

```r
# figura 4.9
par(mfrow=c(1,2))
plot(tukey.reg, variation="SE",
     main="", ylim=c(1000, 4500),
     ylab="Rendimiento (kg/ha)", xlab="Región")
points(dat.rend$rendimiento ~ dat.rend$region,
       col="grey", cex=0.75)

plot(tukey.cp, variation="SE",
     main="", ylim=c(1000, 4500),
     ylab="Rendimiento (kg/ha)", xlab="Cultivo previo")
points(dat.rend$rendimiento ~ dat.rend$cprevio,
       col="grey", cex=0.75)
```

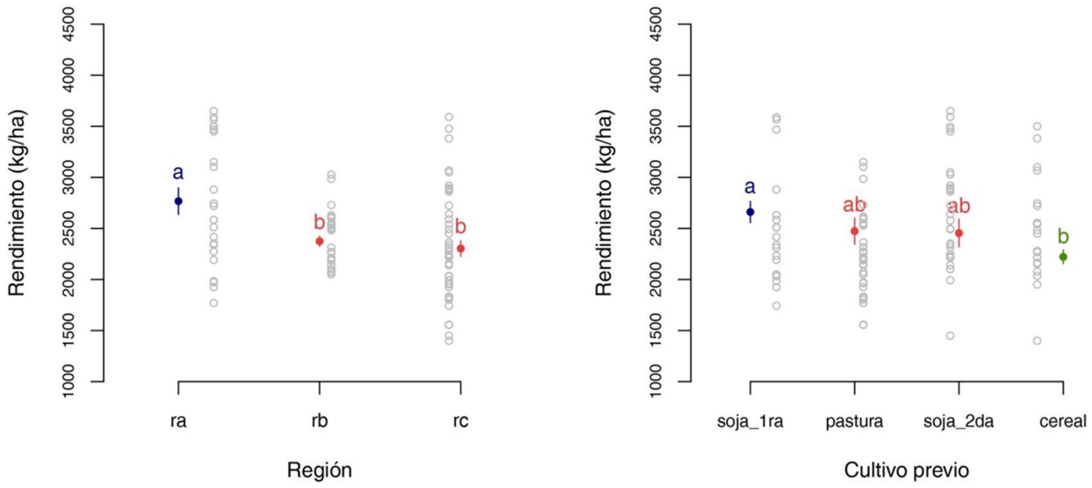

Figura 4. 9. Rendimiento de girasol por región y cultivo previo
Nota. Las letras informan qué medias difieren de acuerdo a un test de Tukey aplicado a cada factor por separado. Los puntos sólidos indican los rendimientos medios, con su error estándar expresado como barras verticales, y los puntos grises sin relleno muestran los rendimientos observados en cada lote.

4. 5. 4. Bondad de ajuste

Veamos algunas medidas de bondad de ajuste:

```r
summary(m.fact)$sigma # error estándar residual

## [1] 454.9857

summary(m.fact)$r.squared # coeficiente de determinación (R²)

## [1] 0.3299709
```

Encontramos que el cultivo previo y la región explican algo más del 30 % de la variabilidad del rendimiento de los lotes.

4. 5. 5. Validación de supuestos

Como en todos los modelos estadísticos, las conclusiones a las que arribamos serán válidas en la medida que el modelo que las sustenta cumpla con sus supuestos:

```
residuos <- resid(m.fact) # residuos del modelo
predichos <- fitted(m.fact) # predichos
# figura 4.10
par(mfrow=c(2,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend2$region,
     xlab="Región", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend2$cprevio,
     xlab="Cultivo previo", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

# Normalidad
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")
```

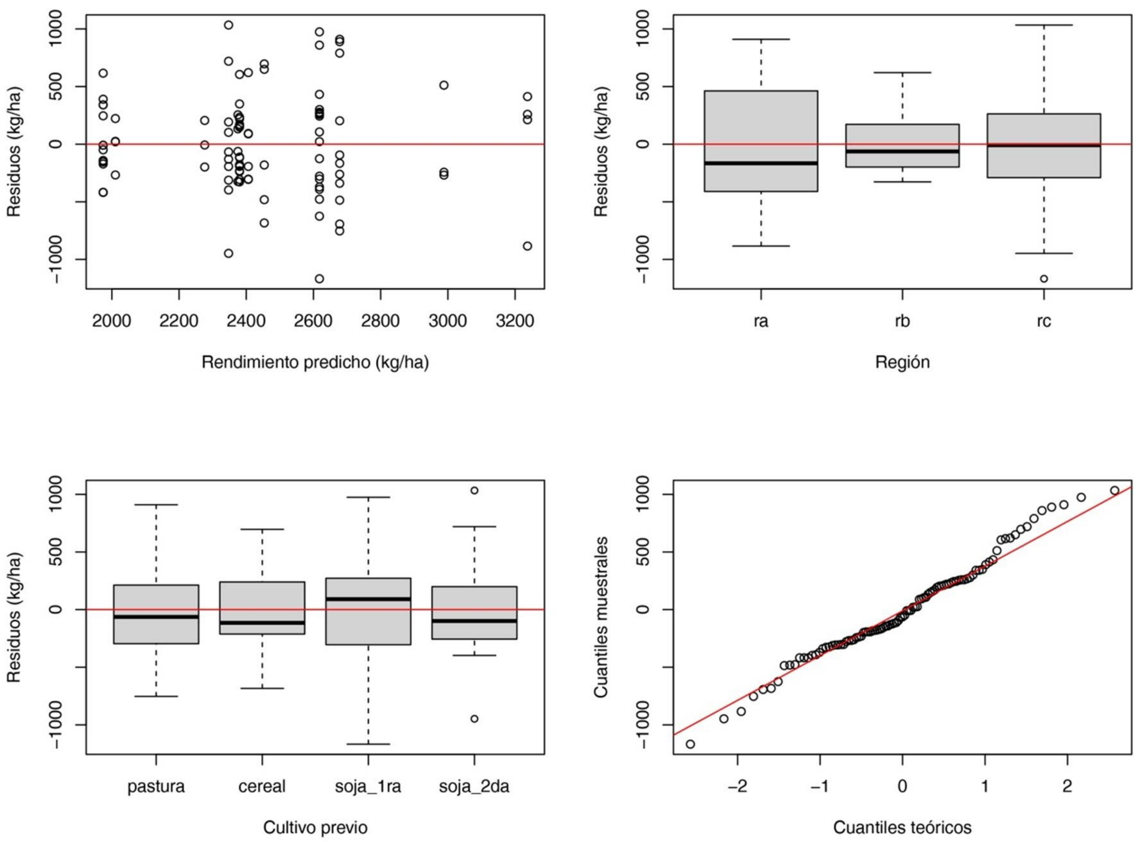

Figura 4. 10. Evaluación gráfica de los supuestos (linealidad, independencia, homocedasticidad y normalidad) del modelo de dos factores (región y cultivo previo) con interacción

```
# - Test Kolmogorov-Smirnov
ks.test(residuos,"pnorm", mean(residuos), sd(residuos))

## Warning in ks.test.default(residuos, "pnorm", mean(residuos), sd(residuos)):
## ties should not be present for the Kolmogorov-Smirnov test

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuos
## D = 0.077858, p-value = 0.5858
## alternative hypothesis: two-sided
```

Capítulo 5. Regresión lineal múltiple

5. 1. Introducción

En este capítulo abordamos los modelos de regresión lineal múltiple. Estos modelos pueden verse como la expansión natural de los modelos de regresión lineal simple. Al igual que los modelos multifactoriales, la regresión lineal múltiple alude a más de un predictor, pero en este caso continuos. La literatura denomina *coeficientes de regresión parcial* a los efectos de estos predictores, y en este capítulo discutimos su significado. Mediante estos modelos presentamos las sumas de cuadrados Tipo I y Tipo III asociados a los análisis de la varianza parciales y marginales, respectivamente. Introducimos el concepto de *multicolinealidad* y su diagnóstico, aspecto central para la correcta aplicación de los modelos lineales. Discutimos el coeficiente de determinación ajustado como un indicador que permite comparar la bondad de ajuste de modelos con diferente cantidad de parámetros. También abordamos los modelos polinómicos como una alternativa para modelar relaciones no lineales mediante regresiones lineales múltiples. Estos modelos nos sirven para indagar sobre un aspecto sumamente relevante: el equilibrio entre el ajuste a los datos (de la muestra) y la capacidad predictiva (fuera de ella).

5. 2. Problema

Nos interesa conocer si la configuración del paisaje afecta el rendimiento del cultivo. Se propone la hipótesis de que los bordes de cultivo son refugios de biodiversidad que funcionan como hábitat de organismos que proveen múltiples servicios ambientales, entre ellos la polinización de cultivos. En este sentido, el tamaño del lote afecta la relación perímetro-área de parches (menor tamaño de lote, mayor peso de borde). De ser correcta la hipótesis de trabajo, se esperan mayores rendimientos en lotes más chicos y con mayor densidad de borde. Dado que el manejo afecta el rendimiento del lote, para aumentar la potencia estadística y la capacidad predictiva del modelo incluimos como covariable la cantidad de fertilizante nitrogenado.

5. 3. Datos

Cargamos los datos:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
```

Exploramos la relación entre el rendimiento con la densidad de borde y el tamaño del lote (figura 5. 1):

```
# figura 5.1
par(mfrow=c(1,2))
# Rendimiento vs. densidad de borde
plot(dat.rend$borde, dat.rend$rendimiento, ylim=c(500,4500),
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)")
# Rendimiento vs. tamaño del lote
plot(dat.rend$tamanio, dat.rend$rendimiento, ylim=c(500,4500),
     xlab="Tamaño del lote (ha)", ylab="Rendimiento (kg/ha)")
```

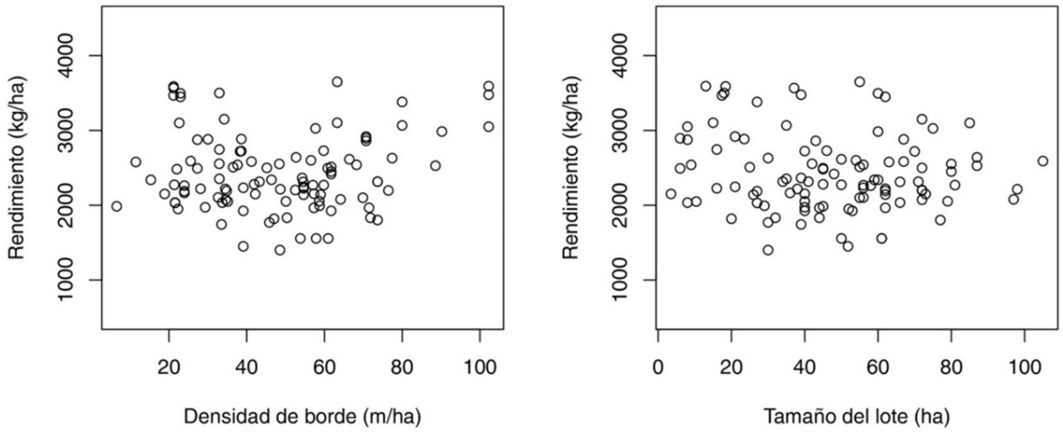

Figura 5. 1. Gráficos de dispersión entre rendimiento y densidad de borde (izquierda) y entre rendimiento tamaño del lote (derecha)

Podemos combinar ambos gráficos de dispersión en uno de tres dimensiones mediante la función `scatterplot3d()` incluida en el paquete con el mismo nombre (figura 5. 2):

```
# figura 5.2
library(scatterplot3d) # install.packages ("scatterplot3d")
par(mfrow=c(1,1))
s3d=scatterplot3d(x=dat.rend$borde, xlab="Densidad de borde (m/ha)",
                  y=dat.rend$tamanio, ylab="Tamaño del lote (ha)",
                  z=dat.rend$rendimiento, zlab="Rendimiento (kg/ha)",
                  box=FALSE, pch=16, highlight.3d=TRUE, angle=50, type="h")
```

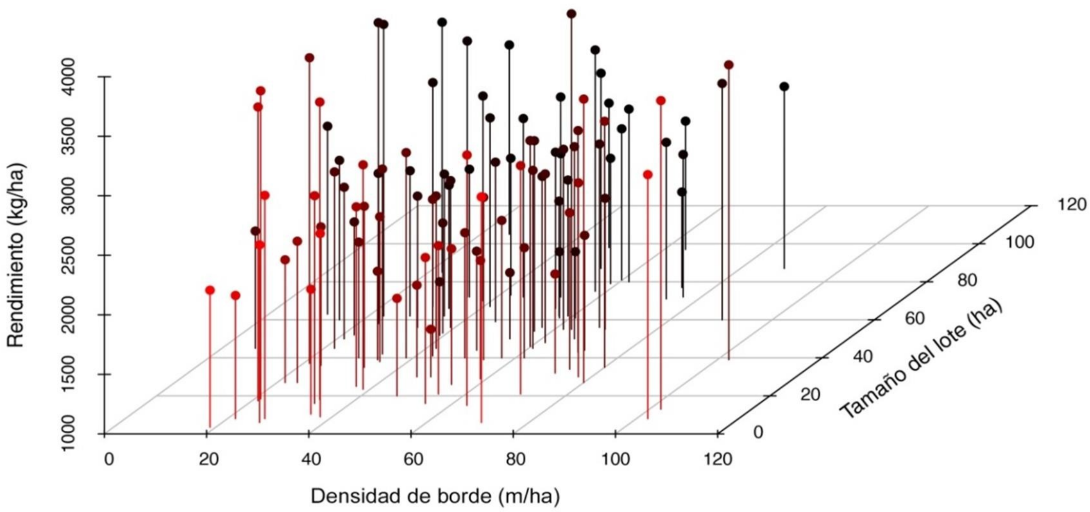

Figura 5. 2. Variabilidad del rendimiento de girasol en función de la densidad de borde y el tamaño del lote

La función `plot()` ejecutada sobre un objeto de clase `data.frame` nos permite explorar de manera sencilla las relaciones entre las variables de interés (figura 5. 3):

```
# figura 5.3
plot(dat.rend[, c(2:5)])
```

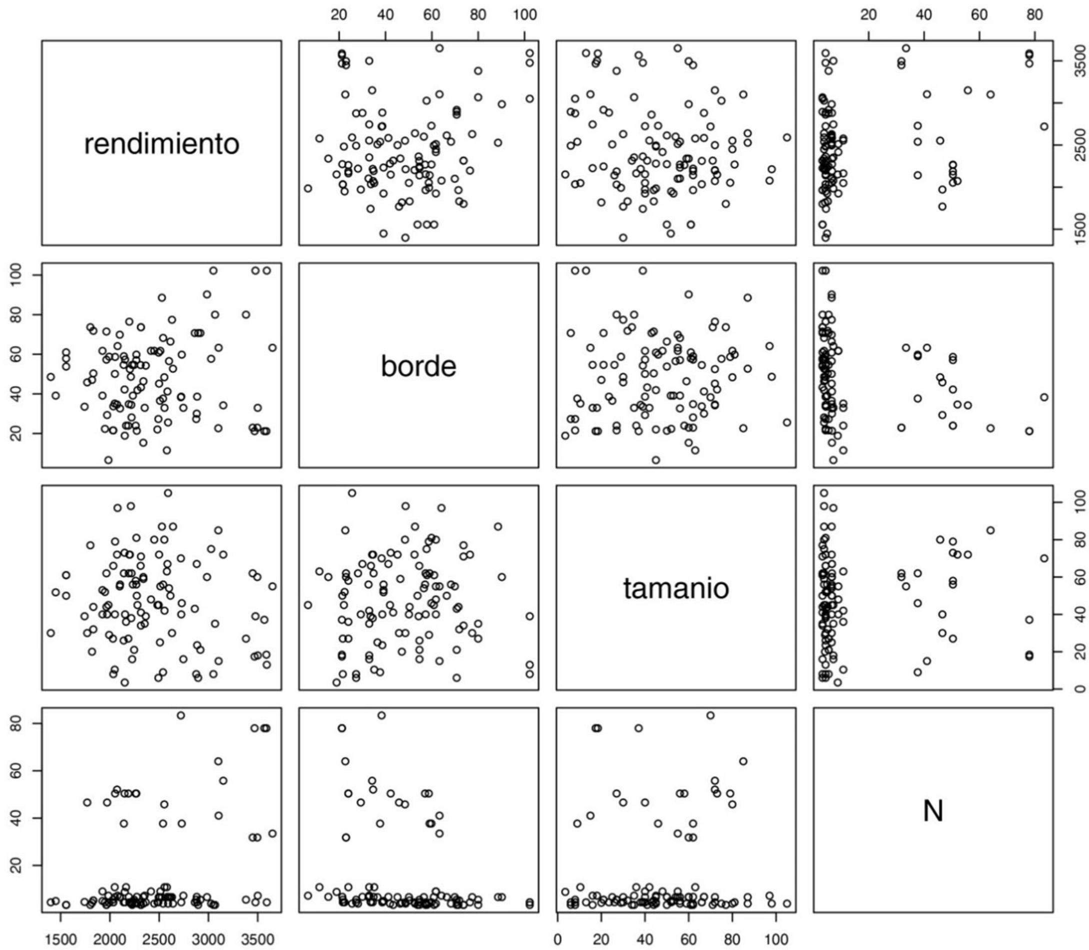

Figura 5. 3. Gráficos de dispersión entre rendimiento, densidad de borde, tamaño del lote y fertilización nitrogenada

5. 4. Modelo y ajuste

Planteamos un modelo de regresión lineal múltiple para comprender cómo varía el rendimiento promedio en función de la cantidad de fertilizante nitrogenado aplicado, la densidad de borde y el tamaño del lote, tres predictores continuos:

$$r_i = \beta_0 + \beta_1 \times n_i + \beta_2 \times b_i + \beta_3 \times t_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0; \sigma^2)_{independientes}$$

donde

r_i = rendimiento de girasol (kg ha^{-1}) del lote i (i = 1, 2, … N)
n_i = cantidad de fertilizante nitrogenado aplicado en el lote i (kg ha^{-1})
b_i = densidad de borde en el contexto (radio de 1,5 km) del lote i (m ha^{-1})
t_i = tamaño del lote i (ha)
ε_i = término de error (kg ha^{-1})

El modelo tiene cinco parámetros a estimar: β_0, β_1, β_2, β_3 y σ^2. En los modelos de regresión múltiple, a los parámetros asociados a los predictores (β_1, β_2 y β_3, en este caso) se les llama *coeficientes de regresión parcial*. Estos expresan el cambio promedio en la variable dependiente por cada unidad que aumenta el predictor asociado manteniendo constantes el resto de los predictores. Por ejemplo, el parámetro β_1 representa la variación promedio en el rendimiento por cada metro que aumenta la densidad de borde a niveles constantes de fertilizante y para un mismo tamaño de lote. Las unidades en las que está expresado cada parámetro, en este caso kg ha^{-1} m^{-1}, son importantes tanto para mantener la coherencia de los cálculos (el modelo predice rendimiento en kg ha^{-1}) como para su interpretación. La ordenada al origen (β_0) expresa el rendimiento medio (su unidad es kg ha^{-1}) de los lotes en que no aplican fertilizante y no tienen bordes en el paisaje, y en los que su tamaño es cero. Esta última característica lo convierte en un parámetro no interpretable en términos del problema. Ello no quiere decir que no haya que estimarlo; geométricamente, el intercepto proporciona la altura del hiperplano de predicción (del plano en el caso de una regresión lineal de dos variables predictoras, o de la recta en una regresión lineal simple).

5. 4. 1. Ajuste del modelo

Como en el resto de los modelos, estimamos los parámetros con `lm()` indicando los predictores a evaluar y luego le pedimos a R los resultados:

```
m.rlm <- lm(rendimiento ~ N + borde + tamanio,
                    data=dat.rend) # ajuste del modelo
coef(m.rlm) # modelo ajustado

## (Intercept)           N       borde     tamanio
## 2190.213844   10.130810    5.432658   -3.639833

round(cbind("β est."=coef(m.rlm), confint(m.rlm)), 2) # IC 95%

##              β est.   2.5 %  97.5 %
## (Intercept) 2190.21 1867.90 2512.53
## N             10.13    5.31   14.96
## borde          5.43    0.76   10.10
## tamanio       -3.64   -7.67    0.39
```

Y con estos informamos la ecuación de regresión estimada:

$$\hat{r} = 2190{,}2 + 10{,}1 \times n + 5{,}4 \times b - 3{,}6 \times t$$

5. 5. ANOVA secuencial y marginal

Veamos la salida de la función `anova()` aplicada al modelo de regresión lineal múltiple que ajustamos:

```
anova(m.rlm)

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716 12.6219 0.0005812 ***
## borde       1  1113467 1113467  4.9008 0.0290938 *
## tamanio     1   730822  730822  3.2166 0.0758868 .
## Residuals 101 22947392  227202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

En el capítulo 2 indicamos que el ANOVA es una partición de la varianza de la variable que se modela. Ahora profundizamos sobre este aspecto. Cuando el modelo tiene más de un predictor, las sumas de cuadrados (SC) y los cuadrados medios resultantes pueden calcularse de diversas formas. Esto resulta en diferentes tipos de ANOVAs (la salida anterior muestra uno de ellos). Las pruebas de hipótesis detrás de estos difieren y, consecuentemente, también lo hacen las preguntas que nos permiten responder.

5. 5. 1. ANOVA secuencial o Tipo I

El ANOVA Tipo I cuantifica la reducción secuencial de la suma de cuadrados del error (SCE) por el ingreso uno a uno de los predictores del modelo. El ingreso de los predictores se da en el orden en el que los escribimos al ajustar el modelo en `lm()`. En nuestro caso: 1) fertilización nitrogenada, 2) densidad de borde, 3) tamaño del lote. Para comprender el abordaje secuencial, primero ajustamos un modelo solo con nitrógeno como predictor:

```
m1p <- lm(rendimiento ~ N, data=dat.rend) # r ~ N
```

y luego exploramos si los residuos del modelo, es decir, la parte de la variabilidad que no explica la fertilización, pueden ser explicados por la densidad de bordes (figura 5. 4):

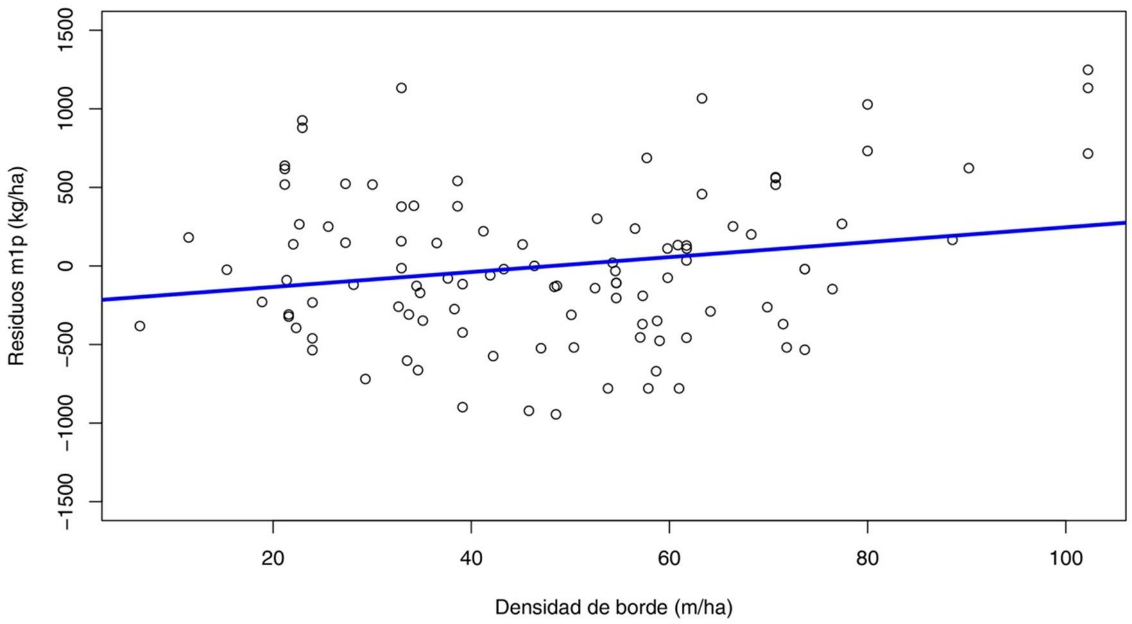

Figura 5. 4. Regresión lineal entre la densidad de borde y los residuos del modelo `mp1` (rendimiento en función de la fertilización nitrogenada)

```
# figura 5.4
par(mfrow=c(1,1))
plot(dat.rend$borde, resid(m1p),
     xlab="Densidad de borde (m/ha)",
     ylab="Residuos m1p (kg/ha)",
     ylim=c(-1500, 1500)) # Residuos m1p vs. borde
abline(lm(resid(m1p) ~ dat.rend$borde), col="blue", lwd=3)
```

En la figura 5. 4 observamos que la densidad de borde parece relacionarse con (explicar parte de) los residuos del modelo que incluye un solo predictor (es decir, con la parte de la variabilidad del rendimiento que no explica la adición de fertilizante). Seguimos con la secuencia y ajustamos un modelo que, además del nitrógeno, incluya la densidad de borde y exploramos si sus residuos pueden ser explicados por el tamaño del lote (figura 5. 5):

```
m2p <- lm(rendimiento ~ N + borde, data=dat.rend) # r ~ N + borde
# figura 5.5
plot(dat.rend$tamanio, resid(m2p),
     xlab="Tamaño de lote (ha)",
     ylab="Residuos m2p (kg/ha)",
     ylim=c(-1500, 1500))
abline(lm(resid(m2p) ~ dat.rend$tamanio), col="blue", lwd=3)
```

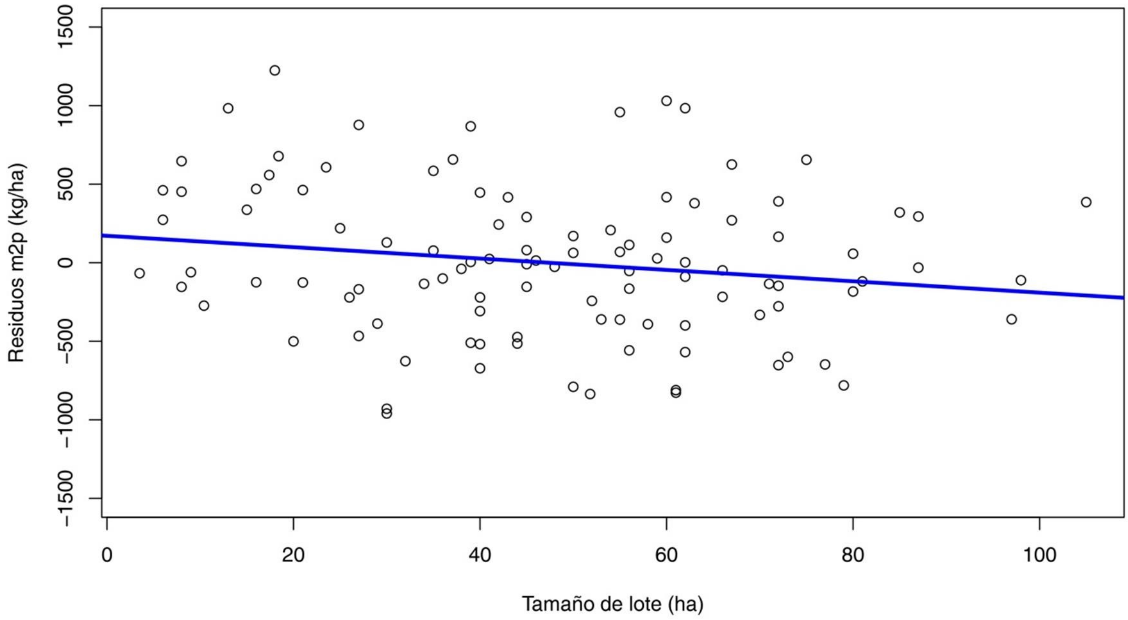

Figura 5. 5. Regresión lineal entre el tamaño del lote y los residuos del modelo `mp2` (rendimiento en función de la fertilización nitrogenada y la densidad de borde)

Restaría estimar un modelo con los tres predictores, pero esto lo hicimos precisamente al ajustar el modelo de regresión múltiple original (el modelo completo en la jerga del modelado estadístico). Comparamos las SC de los tres modelos construidos secuencialmente:

```
anova(m1p) # r ~ N

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716  11.914 0.0008092 ***
## Residuals 103 24791681  240696
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m2p) # r ~ N + borde

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## N           1  2867716 2867716 12.3534 0.000658 ***
## borde       1  1113467 1113467  4.7965 0.030795 *
## Residuals 102 23678214  232139
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.rlm) # r ~ N + borde + tamaño

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716 12.6219 0.0005812 ***
```

```
## borde       1  1113467 1113467  4.9008 0.0290938 *
## tamanio     1   730822  730822  3.2166 0.0758868 .
## Residuals 101 22947392  227202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Encontramos que la SC del nitrógeno (el predictor que ingresa primero al modelo completo) es la misma en las tres tablas de ANOVA:

```
anova(m1p)[1,2] # SC N en m1p

## [1] 2867716

anova(m2p)[1,2] # SC N en m2p

## [1] 2867716

anova(m.rlm)[1,2] # SC N en m.rlm

## [1] 2867716
```

También observamos que la SCE del primer modelo (`m1p`) es igual a la SC de la densidad de borde más la SCE del error del segundo modelo (`m2p`):

```
anova(m1p)[2,2] # SCE en m1p

## [1] 24791681

anova(m2p)[2,2] + anova(m2p)[3,2] # SC borde + SCE en m2p

## [1] 24791681
```

La misma igualdad se observa entre la SCE del segundo modelo (`m2p`) y SC del tamaño de lote más la SCE del error del modelo completo (`m.rlm`):

```
anova(m2p)[3,2] # SCE de m2p

## [1] 23678214

anova(m.rlm)[3,2] + anova(m.rlm)[4,2]  # SC tamaño + SCE en m.rlm

## [1] 23678214
```

Con este ejemplo hemos visto cómo el ANOVA Tipo I realiza una partición secuencial de la variabilidad del rendimiento entre lotes. Además, muestra que la función `anova()` ejecutada sobre un objeto `lm` aplica este análisis. Las hipótesis que ponemos a prueba al aplicar el ANOVA secuencial a partir de nuestro modelo de regresión múltiple son las siguientes:

$$H_0: \quad \beta_1 = 0$$
$$H_0: \quad \beta_2|_{\beta_1} = 0$$
$$H_0: \quad \beta_3|_{\beta_1 \; y \; \beta_2} = 0$$

La primera de las hipótesis pone a prueba si la aplicación de fertilizante es importante para explicar parte de la variabilidad del rendimiento entre lotes. La segunda, examina si la densidad de borde explica la variabilidad del rendimiento que no fue explicada por la fertilización ($\beta_2|_{\beta_1}$ se lee β_2 estando β_1 en el modelo) y la restante, si el tamaño del lote explica la variabilidad del rendimiento que no fue explicada ni por la fertilización ni por la densidad de bordes.

5. 5. 2. ANOVA marginal o Tipo III

Con este análisis evaluamos el efecto de cada predictor sobre la SCE marginal del modelo que lo excluye, es decir, sobre la variabilidad no explicada por el resto de los predictores. Esto implica que, independientemente del orden en el que escribimos el modelo, cada predictor es evaluado como si ingresara último. Para explicar los resultados de este ANOVA cambiamos el predictor que ingresa en último lugar (en el modelo `m.rlm` el predictor que ingresa último es `tamanio`):

```
m.rlm.b <- lm(rendimiento ~ N + tamanio + borde,
               data=dat.rend) # borde ingresa último

m.rlm.c <- lm(rendimiento ~ tamanio + borde + N,
                 data=dat.rend) # N ingresa último
```

para aplicar un ANOVA secuencial sobre cada modelo:

```
anova(m.rlm) # ANOVA secuencial con tamaño ingresando último

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716 12.6219 0.0005812 ***
## borde       1  1113467 1113467  4.9008 0.0290938 *
## tamanio     1   730822  730822  3.2166 0.0758868 .
## Residuals 101 22947392  227202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.rlm.b) # ANOVA secuencial con borde ingresando último

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716 12.6219 0.0005812 ***
## tamanio     1   634557  634557  2.7929 0.0977772 .
## borde       1  1209731 1209731  5.3245 0.0230682 *
## Residuals 101 22947392  227202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.rlm.c) # ANOVA secuencial con N ingresando último

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## tamanio     1   488399  488399  2.1496    0.1457
```

```
## borde       1   280937  280937  1.2365    0.2688
## N           1  3942669 3942669 17.3531 6.553e-05 ***
## Residuals 101 22947392  227202
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

La función `Anova()` del paquete `car` permite realizar un ANOVA tipo III, para lo cual debemos indicarlo en el argumento `type`. Vemos qué sucede cuando aplicamos un análisis marginal sobre el modelo original:

```
library(car)

## Loading required package: carData

Anova(m.rlm, type="III")

## Anova Table (Type III tests)
##
## Response: rendimiento
##               Sum Sq  Df  F value    Pr(>F)
## (Intercept) 41283776   1 181.7052 < 2.2e-16 ***
## N            3942669   1  17.3531 6.553e-05 ***
## borde        1209731   1   5.3245   0.02307 *
## tamanio       730822   1   3.2166   0.07589 .
## Residuals   22947392 101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Vemos que las SC y los valores *p* asociados a los predictores en el ANOVA marginal son iguales a aquellas del ANOVA secuencial cuando los predictores entran últimos. Como puede deducirse, cambiar el orden de los predictores no altera los resultados de este ANOVA:

```
Anova(m.rlm.b, type="III")

## Anova Table (Type III tests)
##
## Response: rendimiento
##               Sum Sq  Df  F value    Pr(>F)
## (Intercept) 41283776   1 181.7052 < 2.2e-16 ***
## N            3942669   1  17.3531 6.553e-05 ***
## tamanio       730822   1   3.2166   0.07589 .
## borde        1209731   1   5.3245   0.02307 *
## Residuals   22947392 101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova(m.rlm.c, type="III")

## Anova Table (Type III tests)
##
## Response: rendimiento
##               Sum Sq  Df  F value    Pr(>F)
## (Intercept) 41283776   1 181.7052 < 2.2e-16 ***
## tamanio       730822   1   3.2166   0.07589 .
## borde        1209731   1   5.3245   0.02307 *
## N            3942669   1  17.3531 6.553e-05 ***
## Residuals   22947392 101
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Las pruebas de hipótesis reportadas en el `summary()` también son marginales, pero en este caso se realizan mediante el estadístico *t*. Comparamos sus valores *p* con los de los ANOVAs marginales:

```
summary(m.rlm)

##
## Call:
## lm(formula = rendimiento ~ N + borde + tamanio, data = dat.rend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1032.04  -332.61   -13.11   309.05  1123.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2190.214    162.481  13.480  < 2e-16 ***
## N             10.131      2.432   4.166 6.55e-05 ***
## borde          5.433      2.354   2.307   0.0231 *
## tamanio       -3.640      2.029  -1.793   0.0759 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 476.7 on 101 degrees of freedom
## Multiple R-squared:  0.1704, Adjusted R-squared:  0.1457
## F-statistic: 6.913 on 3 and 101 DF,  p-value: 0.0002794
```

Las hipótesis del ANOVA marginal aplicado a nuestro modelo son las siguientes:

$$H_0: \quad \beta_{1|_{\beta_2\, y\, \beta_3}} = 0$$
$$H_0: \quad \beta_2|_{|_{\beta_1\, y\, \beta_3}} = 0$$
$$H_0: \quad \beta_3|_{\beta_1\, y\, \beta_2} = 0$$

En todas ellas ponemos a prueba si el ingreso del predictor evaluado reduce la variabilidad del rendimiento no explicada por el resto de los predictores.

Los ANOVAs del modelo nos sugieren que la densidad de borde afecta el rendimiento y encontramos cierta evidencia sobre el efecto del tamaño del lote, aunque esta no es concluyente. Ambos efectos cobran importancia una vez removida la variabilidad explicada por la fertilización nitrogenada. Estos resultados en combinación con los signos de los efectos, positivo para la densidad de borde y negativo para el tamaño del lote, muestran evidencia a favor de la hipótesis de trabajo.

5. 6. Multicolinealidad

Los predictores que son redundantes (no confundir con no importantes) no deben estar en el modelo. Si esto ocurre, genera problemas en el ajuste del modelo y en las conclusiones que se derivan. Por ejemplo, no sería razonable ajustar un modelo con el tamaño de lote medido en hectáreas y, además, con el tamaño de lote medido en m^2. En términos numéricos los valores son diferentes, pero es en esencia la misma variable. Entonces, al ajustar el modelo, ¿a cuál de ellas se le asigna el efecto del tamaño? Este es un caso extremo (correlación perfecta), pero sirve para ejemplificar el hecho de que no debemos incluir variables altamente correlacionadas ya que explican la misma porción de la variabilidad. La multicolinealidad no es una característica binaria (la hay o no la hay) sino que representa un problema de grado, y existen diferentes formas de evaluarla. A modo de ejemplo, generamos una nueva variable, el total de nitrógeno aplicado al lote, que obtenemos multiplicando la cantidad de fertilizante por hectárea por el área del lote:

```
dat.rend$Nt <- dat.rend$N * dat.rend$tamanio # creamos la variable Nt
names(dat.rend)

##  [1] "lote"        "rendimiento" "borde"       "tamanio"     "N"
##  [6] "dsiembra"    "fsiembra"    "cprevio"     "region"      "Nt"
```

Al ajustar un modelo con `Nt` en función de `N` y `tamanio`:

```
summary(lm(Nt ~ N + tamanio, data=dat.rend)) # Nt ~ N + tamanio

##
## Call:
## lm(formula = Nt ~ N + tamanio, data = dat.rend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1963.06  -186.36    13.41   193.09  1743.11
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -728.669    114.972  -6.338 6.41e-09 ***
## N             48.429      2.414  20.063  < 2e-16 ***
## tamanio       15.601      2.108   7.401 3.95e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 495.8 on 102 degrees of freedom
## Multiple R-squared:  0.8235, Adjusted R-squared:   0.82
## F-statistic: 237.9 on 2 and 102 DF,  p-value: < 2.2e-16
```

Vemos que el 82 % de la variabilidad en el nitrógeno total está explicado por la fertilización nitrogenada por hectárea y el tamaño del lote (si multiplicamos los efectos explicaremos el 100 %). Veamos qué sucede al incluir esta variable para explicar el rendimiento.

5. 6. 1. Cambio en el orden de predictores

Incluimos también la densidad de borde y ajustamos dos modelos para predecir el rendimiento. En un modelo el nitrógeno total ingresa primero y en el otro modelo ingresa al final:

```
m.rlm.plus <- lm(rendimiento ~ N + tamanio + borde + Nt,
                   data=dat.rend) # rend ~ N + tamanio + borde + Nt

m.rlm.plus2 <- lm(rendimiento ~ Nt + N + tamanio + borde,
                    data=dat.rend) # rend ~ Nt + N + tamanio + borde
```

Comparamos sus ANOVAs secuenciales:

```
anova(m.rlm.plus)

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value    Pr(>F)
## N           1  2867716 2867716 12.5735 0.0005971 ***
## tamanio     1   634557  634557  2.7822 0.0984452 .
## borde       1  1209731 1209731  5.3041 0.0233469 *
## Nt          1   139711  139711  0.6126 0.4356742
## Residuals 100 22807681  228077
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

anova(m.rlm.plus2)

## Analysis of Variance Table
##
## Response: rendimiento
##            Df   Sum Sq Mean Sq F value   Pr(>F)
## Nt          1  1142258 1142258  5.0082 0.027445 *
## N           1  2248610 2248610  9.8590 0.002222 **
## tamanio     1   209348  209348  0.9179 0.340342
## borde       1  1251500 1251500  5.4872 0.021138 *
## Residuals 100 22807681  228077
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Y también los ANOVAs marginales:

```
Anova(m.rlm.plus, type="III")

## Anova Table (Type III tests)
##
## Response: rendimiento
##               Sum Sq  Df  F value  Pr(>F)
## (Intercept) 32054490   1 140.5425 < 2e-16 ***
## N            1563730   1   6.8562 0.01021 *
## tamanio       221045   1   0.9692 0.32726
## borde        1251500   1   5.4872 0.02114 *
## Nt            139711   1   0.6126 0.43567
## Residuals   22807681 100
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Anova(m.rlm.plus2, type="III")

## Anova Table (Type III tests)
##
## Response: rendimiento
##               Sum Sq  Df  F value  Pr(>F)
## (Intercept) 32054490   1 140.5425 < 2e-16 ***
## Nt            139711   1   0.6126 0.43567
## N            1563730   1   6.8562 0.01021 *
## tamanio       221045   1   0.9692 0.32726
## borde        1251500   1   5.4872 0.02114 *
## Residuals   22807681 100
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Y finalmente:

```
summary(m.rlm.plus)

##
## Call:
## lm(formula = rendimiento ~ N + tamanio + borde + Nt, data = dat.rend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1077.59  -328.91   -11.13   329.91  1142.05
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2130.61685  179.72210  11.855   <2e-16 ***
## N             13.78250    5.26366   2.618   0.0102 *
## tamanio       -2.47827    2.51738  -0.984   0.3273
## borde          5.53396    2.36244   2.342   0.0211 *
## Nt            -0.07476    0.09552  -0.783   0.4357
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 477.6 on 100 degrees of freedom
## Multiple R-squared:  0.1754, Adjusted R-squared:  0.1424
## F-statistic: 5.318 on 4 and 100 DF,  p-value: 0.0006336

summary(m.rlm.plus2)

##
## Call:
## lm(formula = rendimiento ~ Nt + N + tamanio + borde, data = dat.rend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1077.59  -328.91   -11.13   329.91  1142.05
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2130.61685  179.72210  11.855   <2e-16 ***
## Nt            -0.07476    0.09552  -0.783   0.4357
## N             13.78250    5.26366   2.618   0.0102 *
## tamanio       -2.47827    2.51738  -0.984   0.3273
## borde          5.53396    2.36244   2.342   0.0211 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 477.6 on 100 degrees of freedom
## Multiple R-squared:  0.1754, Adjusted R-squared:  0.1424
## F-statistic: 5.318 on 4 and 100 DF,  p-value: 0.0006336
```

Se observa que `Nt` no es significativo cuando ingresa último en el ANOVA secuencial, pero sí lo es cuando ingresa primero. Además, cuando ingresa primero modifica sustancialmente los valores *p* de `N` y `tamanio` en el ANOVA secuencial. Por el contrario, independientemente del orden en el que ingrese, `Nt` nunca aparece como significativo en el ANOVA marginal. Esto quiere decir que si bien `Nt` explica parte de la variabilidad del rendimiento entre lotes, se trata de la misma porción explicada por `N` y `tamanio`, lo que resulta un claro ejemplo de multicolinealidad.

5. 6. 2. Correlación entre predictores

La detección de problemas de multicolinealidad cambiando el orden de predictores es la más robusta para el análisis, pero se vuelve inviable cuando el modelo cuenta con muchos predictores. Una rápida detección de posibles problemas puede obtenerse explorando gráfica (figura 5. 6) y numéricamente la correlación lineal entre predictores; aquellos con alta correlación presentarán mayores problemas:

```
pairs(dat.rend[ ,c(3:5,10)], panel=panel.smooth) # figura 5.6
```

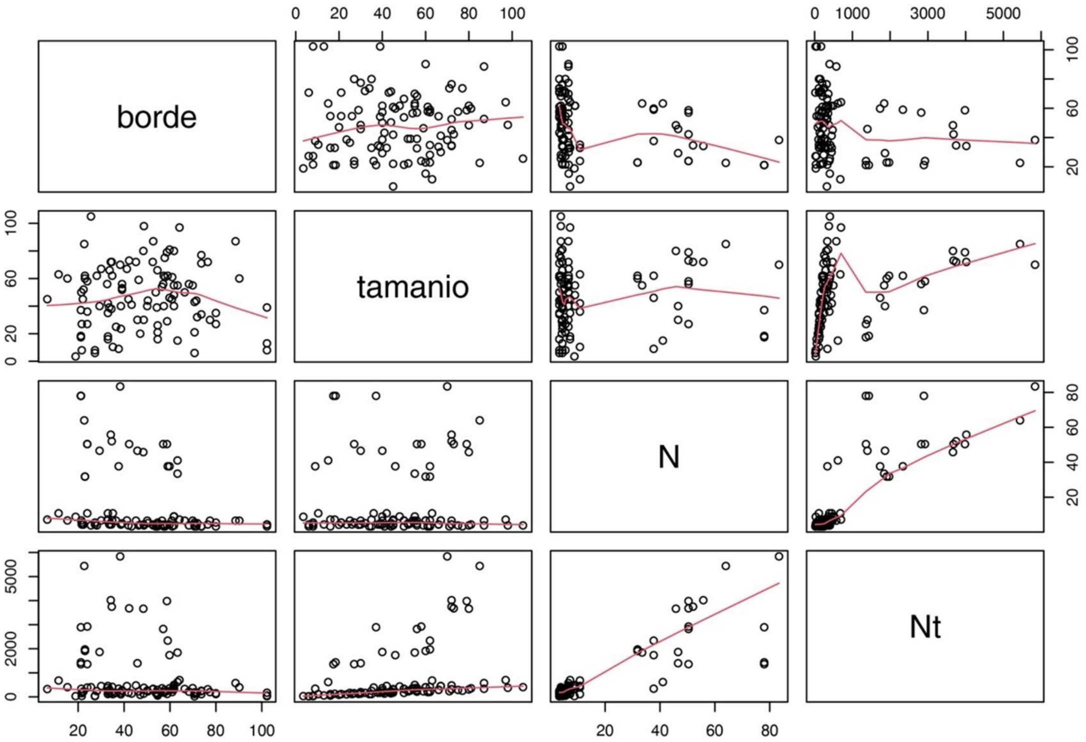

Figura 5. 6. Exploración gráfica de la relación de a pares entre los predictores del modelo

Las líneas rojas (que pedimos desde el argumento `panel`) son curvas suavizadas y ayudan a reconocer las tendencias de las relaciones. Para cuantificar tales relaciones podemos generar una matriz de correlación:

```
round(cor(dat.rend[ ,c(3:5,10)]), 2)

##         borde tamanio     N    Nt
## borde    1.00    0.03 -0.30 -0.22
## tamanio  0.03    1.00  0.06  0.36
## N       -0.30    0.06  1.00  0.85
## Nt      -0.22    0.36  0.85  1.00
```

La función `corrplot()` del paquete del mismo nombre ofrece una serie de opciones útiles para evaluar matrices de correlación de manera gráfica y numérica. Vemos tres alternativas (figura 5. 7):

```
library(corrplot) # recordar instalar el paquete antes
?corrplot()

# figura 5.7
par(mfrow=c(1,3))
corrplot(cor(dat.rend[ ,c(3:5, 10)]))
corrplot(cor(dat.rend[ ,c(3:5,10)]), method="number")
corrplot(cor(dat.rend[ ,c(3:5,10)]), addCoef.col = TRUE)
```

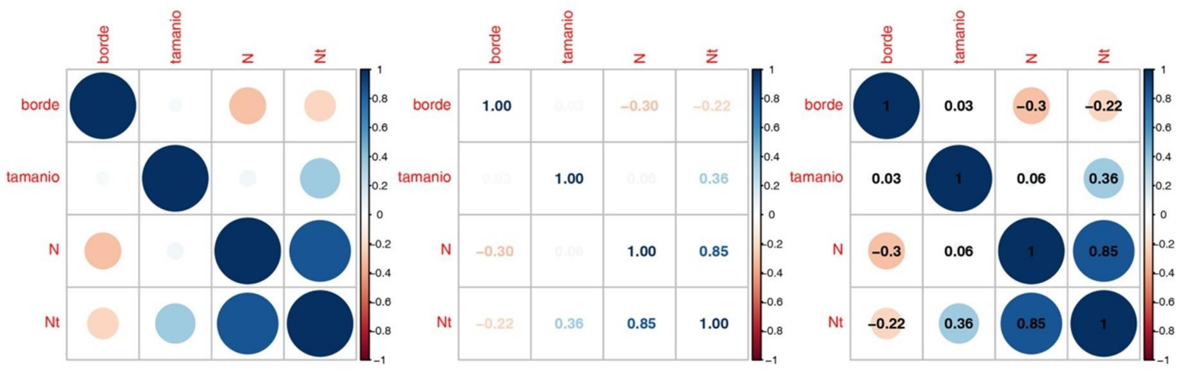

Figura 5. 7. Tres alternativas que brinda la función `corrplot()` para evaluar múltiples correlaciones lineales

La alternativa de `ggplot2` a `pairs()` es la función `ggpairs()`, contenida en el paquete `GGally`:

```
library(GGally) # instalar antes el paquete
ggpairs(dat.rend[,c(3:5,10)],
        diag = list(continuous = "barDiag")) # figura 5.8
```

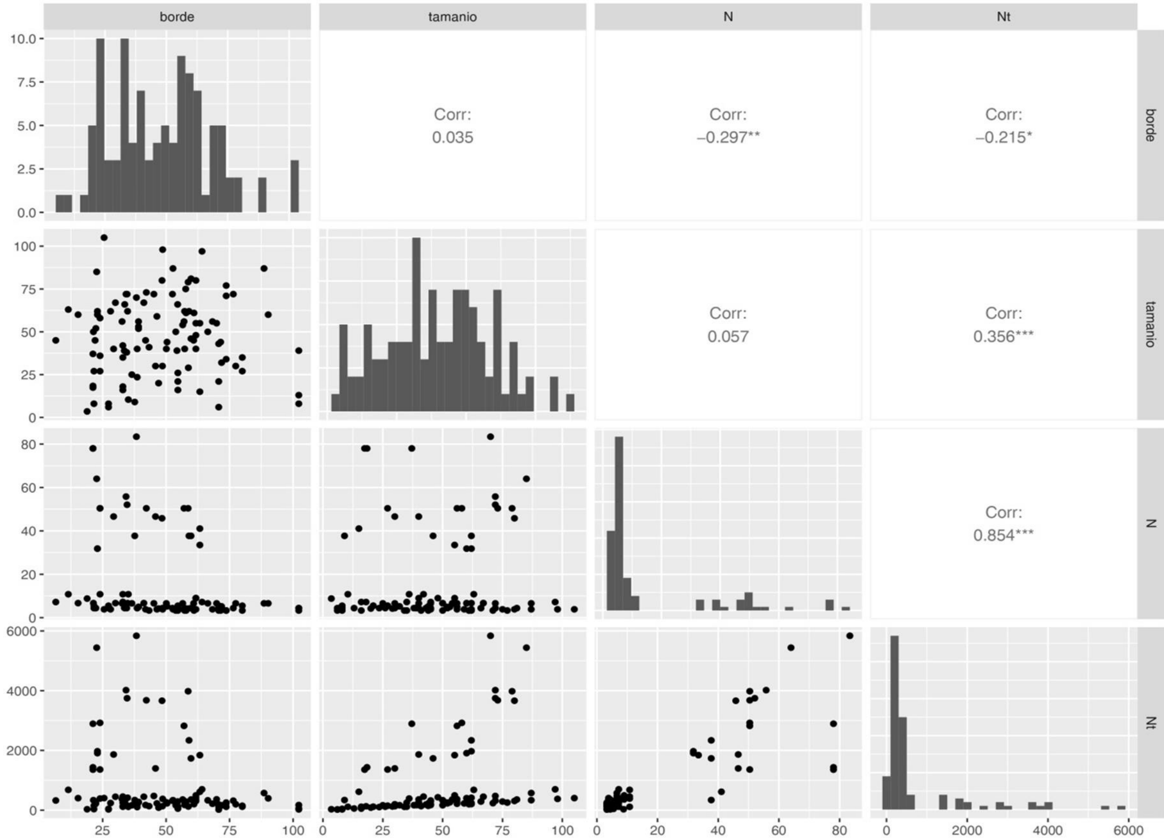

Figura 5. 8. Matriz de correlación y gráficos de dispersión obtenidos mediante la función `ggpairs()` (en la diagonal se muestran los histogramas de las variables)

5. 6. 3. Factor de inflación de varianza

El coeficiente de correlación lineal evalúa predictores de a pares (técnicamente, estamos evaluando colinealidad). Lo más común, sin embargo, es que las relaciones ocurran entre más de dos y un predictor pueda ser explicado por varios otros. Para esto usamos el factor de inflación de varianzas o VIF, por sus siglas en inglés. La formulación del VIF se basa en el r^2 de una regresión lineal entre el predictor evaluado y el resto de los predictores del modelo:

$$\text{VIF} = \frac{1}{1 - r^2}$$

Antes vimos que el 82 % del nitrógeno total era explicado por el nitrógeno por hectárea y el tamaño del lote (r^2 = 0,82). Por lo tanto, en este caso su VIF será de 1/(1 – 0,82) ≈ 5,5. Calculamos el VIF del nitrógeno total cuando también está presente la densidad de borde (`m.rlm.plus` y `m.rlm.plus2`):

```
r2.nt <- summary(lm(Nt ~ N + tamanio + borde, data=dat.rend))$r.squared
r2.nt

## [1] 0.8239894

VIF.nt <- 1/(1 - r2.nt)
VIF.nt

## [1] 5.681476
```

Y para el resto de los predictores:

```
r2.n <- summary(lm(N ~ Nt + tamanio + borde, data=dat.rend))$r.squared
r2.n

## [1] 0.8055273

VIF.n <- 1/(1 - r2.n)
VIF.n

## [1] 5.142109

r2.tam <- summary(lm(tamanio ~ borde + N + Nt, data=dat.rend))$r.squared
r2.tam

## [1] 0.3515844

VIF.tam <- 1/(1 - r2.tam)
VIF.tam

## [1] 1.542221

r2.borde <- summary(lm(borde ~ tamanio + N + Nt, data=dat.rend))$r.squared
r2.borde

## [1] 0.09335261

VIF.borde <- 1/(1 - r2.borde)
VIF.borde

## [1] 1.102965

r2 <- c(r2.n, r2.tam, r2.borde, r2.nt)
VIF <- c(VIF.n, VIF.tam, VIF.borde, VIF.nt)

tabla.mc <- data.frame(r2, VIF)
rownames(tabla.mc) <- c("N", "tamanio", "borde", "Nt")
round(tabla.mc, 3)

##            r2   VIF
## N       0.806 5.142
## tamanio 0.352 1.542
## borde   0.093 1.103
## Nt      0.824 5.681

vif(m.rlm.plus) # más fácil

##        N  tamanio    borde       Nt
## 5.142109 1.542221 1.102965 5.681476
```

De la fórmula del VIF se deduce que este aumenta a mayor r^2. Por lo tanto, VIF más altos implican mayor multicolinealidad. La bibliografía indica que los VIF mayores a 10 representan problemas de multicolinealidad aunque algunos autores sugieren que son los VIF superiores a 3, que en efecto es el caso de nuestro ejemplo. Cuando removemos `Nt` encontramos que los VIFs se vuelven cercanos a 1, lo que sugiere que un modelo con `N`, `borde` y `tamanio` no presenta problemas de multicolinealidad:

```
vif(m.rlm) # VIF modelo original

##        N    borde  tamanio
## 1.101907 1.099654 1.006194
```

5. 7. Intervalos de confianza y de predicción

Utilizamos nuestro modelo para predecir el rendimiento de lotes y conjuntos de lotes no observados. Por ejemplo, podemos predecir el rendimiento medio de lotes grandes ubicados en un paisaje con baja densidad de borde y contrastarlo con el que se obtendría en lotes chicos con alta densidad de bordes. A fin de caracterizarlos, pedimos un resumen de los predictores:

```
summary(dat.rend[, c(3:5)])

##      borde           tamanio             N
##  Min.   :  6.55   Min.   :  3.50   Min.   : 3.30
##  1st Qu.: 32.96   1st Qu.: 30.00   1st Qu.: 4.40
##  Median : 48.39   Median : 46.00   Median : 5.50
##  Mean   : 48.14   Mean   : 47.37   Mean   :15.04
##  3rd Qu.: 61.72   3rd Qu.: 62.00   3rd Qu.: 9.00
##  Max.   :102.24   Max.   :105.00   Max.   :83.40
```

Consideramos los valores máximos y mínimos de `borde` y `tamanio` como indicadores de lote grande/chico y alta/baja densidad de borde, y realizamos las predicciones para lotes que aplican nitrógeno (`N`) en cantidades promedio (15 kg ha^{-1}). De acuerdo a esto, la predicción e intervalo de confianza para el rendimiento medio de lotes grandes con baja densidad de borde es la siguiente:

```
predict(m.rlm,
        data.frame(N=15, borde=7, tamanio=105),
        interval="confidence") # IC lote grande/baja densidad de borde

##        fit      lwr      upr
## 1 1998.022 1675.356 2320.688
```

y para lotes chicos con alta densidad de borde:

```
predict(m.rlm,
        data.frame(N=15, borde=102, tamanio=4),
        interval="confidence") # IC lote chico/alta densidad de borde

##        fit      lwr      upr
## 1 2881.748 2554.656 3208.839
```

El intervalo de predicción para estos lotes:

```
predict(m.rlm,
        data.frame(N=15, borde=7, tamanio=105),
        interval="prediction") # IP lote grande/baja densidad de borde

##        fit      lwr     upr
## 1 1998.022 998.9247 2997.12

predict(m.rlm,
        data.frame(N=15, borde=102, tamanio=4),
        interval="prediction") # IP lote chico/alta densidad de borde
```

```
##        fit      lwr      upr
## 1 2881.748 1881.212 3882.283
```

Nuestro modelo predice mayores rendimientos para lotes chicos con alta densidad de borde, de acuerdo con nuestra hipótesis de trabajo. Combinamos las capacidades de `ggplot` con la función `ggpredict()` del paquete `ggeffects` para mostrar los intervalos de confianza del rendimiento variando la densidad de borde, el tamaño del lote y la cantidad de fertilizante aplicado (figura 5. 9, usamos la función `grid.arrange()` del paquete `gridExtra` para mostrar los tres gráficos en una misma salida). Para cada predictor, los intervalos de confianza se calculan fijando el resto de los predictores en su valor promedio:

```
library(ggplot2)
library(ggeffects) # instalar antes el paquete

# figura 5.9
g1 <- ggplot(ggpredict(m.rlm, c("borde")), aes(x = x, y = predicted)) +
       geom_line(colour="blue", size=1) +
       geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.095) +
       geom_point(data=dat.rend, aes(borde, rendimiento)) +
       labs(x = "Densidad de borde (m/ha)", y="Rendimiento (kg/ha)") +
       ylim(500, 4500)

g2 <- ggplot(ggpredict(m.rlm, c("tamanio")), aes(x = x, y = predicted)) +
       geom_line(colour="blue", size=1) +
       geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.095) +
       geom_point(data=dat.rend, aes(tamanio, rendimiento)) +
       labs(x = "Tamaño de lote (ha)", y="Rendimiento (kg/ha)") +
       ylim(500, 4500)

g3 <- ggplot(ggpredict(m.rlm, c("N")), aes(x = x, y = predicted)) +
       geom_line(colour="blue", size=1) +
       geom_ribbon(aes(ymin = conf.low, ymax = conf.high), alpha = 0.095) +
       geom_point(data=dat.rend, aes(N, rendimiento)) +
       labs(x = "N (kg/ha)", y="Rendimiento (kg/ha)") +
       ylim(500, 4500)

library(gridExtra) # instalar antes el paquete
grid.arrange(g1, g2, g3, ncol=3) # función del paquete gridExtra
```

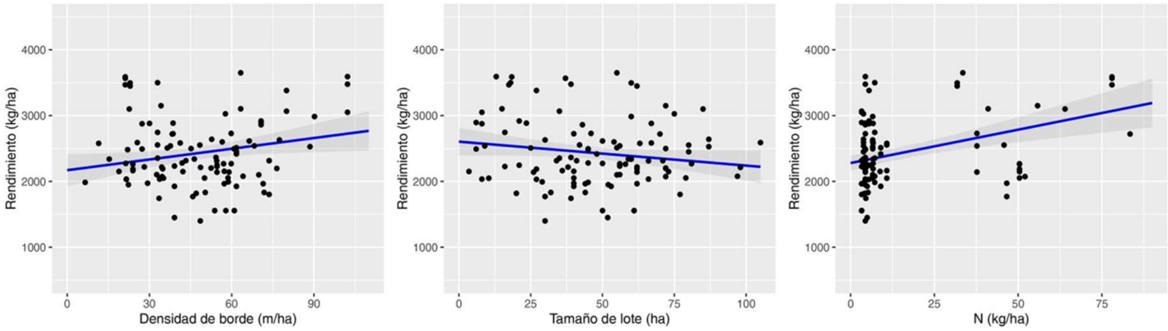

Figura 5. 9. Rendimiento promedio (líneas azules) e intervalos de confianza obtenidos, en cada caso, variando el predictor de interés y manteniendo a los otros dos en su valor promedio

5. 8. Bondad de ajuste

Evaluamos los indicadores de bondad de ajuste de nuestro modelo. Comenzamos por el error estándar residual, recordando que este es la estimación de uno de los parámetros del modelo (σ):

```
summary(m.rlm)$sigma # error estándar residual

## [1] 476.657
```

Al pedir el `summary` del modelo, podemos notar que se informan dos coeficientes de determinación:

```
summary(m.rlm)

##
## Call:
## lm(formula = rendimiento ~ N + borde + tamanio, data = dat.rend)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1032.04  -332.61   -13.11   309.05  1123.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2190.214    162.481  13.480  < 2e-16 ***
## N             10.131      2.432   4.166 6.55e-05 ***
## borde          5.433      2.354   2.307   0.0231 *
## tamanio       -3.640      2.029  -1.793   0.0759 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 476.7 on 101 degrees of freedom
## Multiple R-squared:  0.1704, Adjusted R-squared:  0.1457
## F-statistic: 6.913 on 3 and 101 DF,  p-value: 0.0002794
```

El R^2 aumenta al incluir más predictores al modelo, independientemente de que estos sean importantes para explicar la variable respuesta. Esto distorsiona la comparación del poder explicativo de modelos con diferente número de predictores. Una forma de corregirlo es aplicando una penalización por el número de predictores, de tal forma que no convenga incluir aquellos que no reducen considerablemente la variabilidad no explicada. Esta es la lógica del coeficiente de determinación ajustado, el cual se informa en el `summary` del modelo junto con el R^2:

```
summary(m.rlm)$r.squared # R^2

## [1] 0.1703582

summary(m.rlm)$adj.r.squared # R^2 ajustado

## [1] 0.1457154
```

La fórmula del R^2 ajustado es la siguiente:

$$R_{aj}^2 = 1 - \frac{SCE/(n-p-1)}{SCT/(n-1)}$$

siendo SCE la suma de cuadrados del error, SCT la suma de cuadrados total, n el tamaño muestral y p el número de predictores. Si hacemos esta cuenta obtenemos el mismo valor que el informado en el `summary`:

```
sce = sum(resid(m.rlm)^2) # SC Error
sct = with(dat.rend, sum((rendimiento-(mean(rendimiento)))^2)) # SC Total
n <- length(dat.rend$lote) # tamaño de la muestra
p <- 3 # número de predictores

r2.ajustado <- 1 - (sce/(n-p-1))/(sct/(n-1))
r2.ajustado # R^2 ajustado paso a paso

## [1] 0.1457154
```

Recordemos que el gráfico de observados *vs* predichos es un complemento para evaluar la bondad de ajuste (figura 5. 10):

```
predichos <- fitted(m.rlm) # predichos del modelo de rlm

par(mfrow=c(1,1))
with(dat.rend, plot(predichos, rendimiento,
                    xlab="Rendiiento predicho (kg/ha)",
                    ylab="Rendimiento observado (kg/ha)",
                    ylim=c(500,4500)))
abline(a=0,b=1,lwd=3) # figura 5.10
```

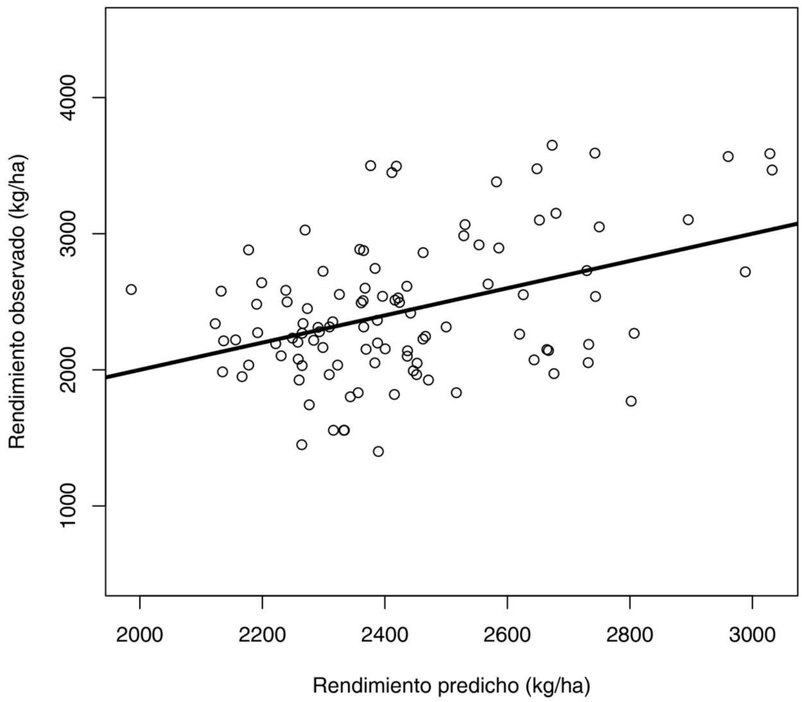

Figura 5. 10. Gráfico de observados *vs* predichos del modelo de regresión lineal múltiple (la línea representa la recta uno a uno)

5. 9. Validación de supuestos

Como siempre, debemos evaluar los supuestos del modelo (linealidad, independencia, homocedasticidad y normalidad):

```
residuos <- resid(m.rlm) # residuos del modelo de rlm

# figura 5.11
par(mfrow=c(3,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend$borde, residuos,
     xlab="Densidad de borde (m/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend$tamanio, residuos,
     xlab="Tamaño de lote (ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend$N, residuos,
     xlab="N (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
hist(residuos, main="", xlab="Residuos (kg/ha)", ylab="Frecuencia")
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")
```

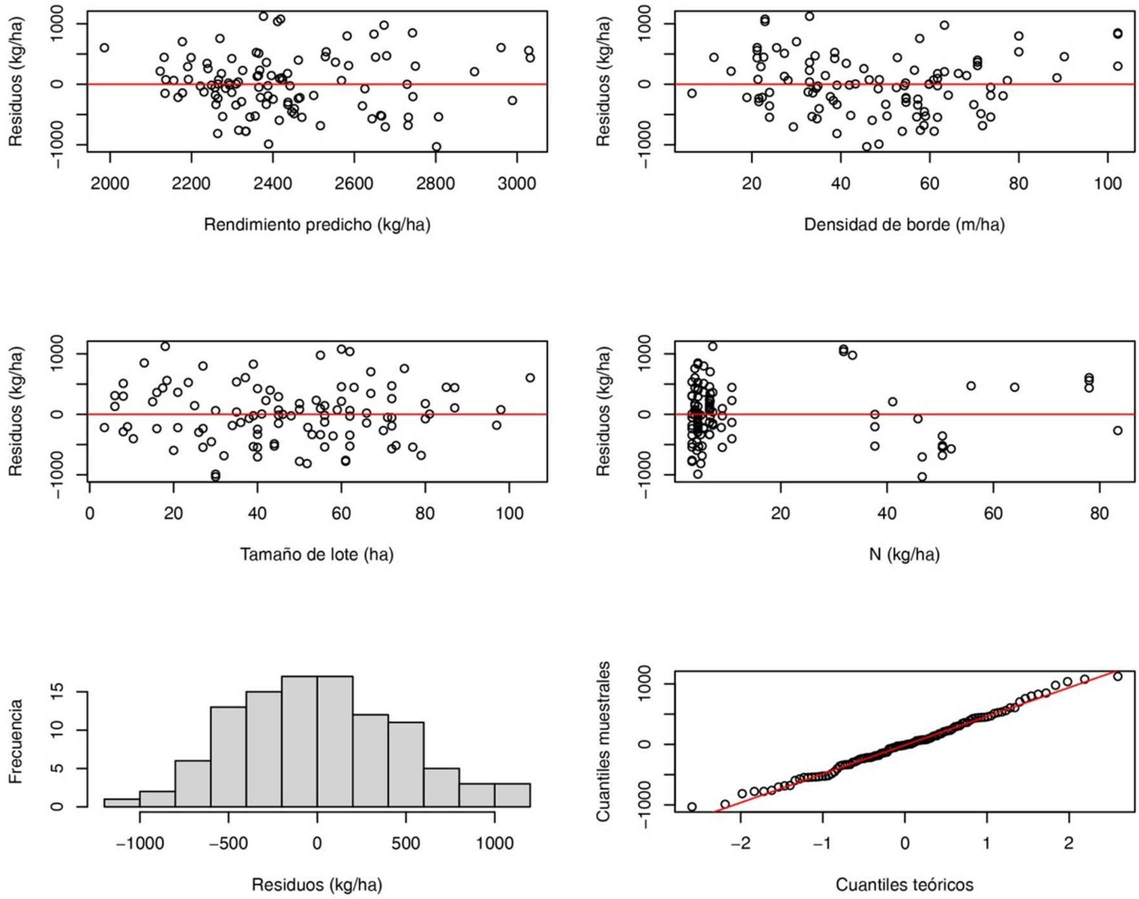

Figura 5. 11. Evaluación gráfica de los supuestos (linealidad, independencia, homocedasticidad y normalidad) del modelo de regresión lineal múltiple

```
# - Test Kolmogorov-Smirnov (normalidad)
ks.test(residuos,"pnorm", mean(residuos), sd(residuos))

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuos
## D = 0.044685, p-value = 0.9848
## alternative hypothesis: two-sided
```

5.10. Modelos polinómicos

Los modelos polinómicos son modelos lineales (las predicciones se obtienen mediante una suma de parámetros) que permiten modelar relaciones no lineales. En efecto, constituyen regresiones lineales múltiples. Lo ejemplificamos con la relación entre rendimiento y densidad de borde.

El patrón observado en la figura 5. 1 (gráfico izquierdo) muestra cierta tendencia a una relación no lineal. Sea por cuestiones teóricas (ecológicas, biológicas, socioeconómicas, etc.,), o simplemente fenomenológicas o predictivas, podemos plantear un modelo polinómico de segundo grado y poner a prueba la relación curvilínea entre la densidad de borde y el rendimiento:

$$r_i \;=\; \beta_0 \;+\; \beta_1 \times b_i \;+\; \beta_2 \times b_i^2 \;+\; \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0;\, \sigma^2)_{independientes}$$

donde

r_i = rendimiento (kg ha^{-1}) del lote *i* (*i* = 1, 2, … N)
b_i = densidad de borde en el contexto (radio de 1,5 km) del lote *i* (m ha^{-1})
ε_i = término de error (kg ha^{-1})

Una forma de hacer esto en R es creando un vector de los valores de borde elevados al cuadrado:

```
dat.rend$borde2 <- dat.rend$borde^2 # densidad de borde al cuadrado
names(dat.rend)

##  [1] "lote"        "rendimiento" "borde"       "tamanio"     "N"
##  [6] "dsiembra"    "fsiembra"    "cprevio"     "region"      "Nt"
## [11] "borde2"
```

y luego ajustar el modelo incluyendo la variable borde y el vector de valores cuadráticos:

```
m.pol <- lm(rendimiento ~ borde + borde2, data=dat.rend) # polinomio de 2do grado
```

Alternativamente, podemos usar la función `I()` dentro de la fórmula del modelo para indicar que `^` sea usado como operador aritmético:

```
m.pol.i <- lm(rendimiento ~ borde + I(borde^2), data=dat.rend) # alternativamente

coef(m.pol)

##  (Intercept)        borde       borde2
## 3177.4356489  -36.8572067    0.3744802

coef(m.pol.i)

##  (Intercept)        borde   I(borde^2)
## 3177.4356489  -36.8572067    0.3744802
```

Al pedir el resumen del modelo encontramos que ambos coeficientes son significativos, lo que sugiere que la relación es no lineal:

```
summary(m.pol)

##
## Call:
## lm(formula = rendimiento ~ borde + borde2, data = dat.rend)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -967.09 -322.36  -55.54  281.20 1305.34
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3177.43565  230.12465  13.807  < 2e-16 ***
## borde        -36.85721    9.29795  -3.964 0.000137 ***
## borde2         0.37448    0.08612   4.348 3.25e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 476.1 on 102 degrees of freedom
## Multiple R-squared:  0.1642, Adjusted R-squared:  0.1478
## F-statistic: 10.02 on 2 and 102 DF,  p-value: 0.0001066
```

Podemos ajustar el modelo sin el término cuadrático, es decir, una regresión lineal simple entre rendimiento y densidad de borde:

```
m.borde <- lm(rendimiento ~ borde, data=dat.rend) # rls
```

y luego usar la función `anova()` para comparar ambos modelos (notemos que ahora estamos ejecutando la función sobre dos modelos):

```
anova(m.borde, m.pol) # comparación de modelos (m.borde está anidado en m.pol)

## Analysis of Variance Table
##
## Model 1: rendimiento ~ borde
## Model 2: rendimiento ~ borde + borde2
##   Res.Df      RSS Df Sum of Sq      F    Pr(>F)
## 1    103 27403865
## 2    102 23118162  1   4285703 18.909 3.255e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El valor *p* indica que el modelo más complejo, es decir, el que incluye el término cuadrático, reduce significativamente la variabilidad no explicada. La función `curve()` es útil para ver el ajuste del modelo polinómico y compararlo con el de la regresión lineal simple (figura 5. 12):

```
# figura 5.12
par(mfrow=c(3,2))
plot(dat.rend$borde, dat.rend$rendimiento, main="RLS",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     ylim=c(500,4500))
abline(m.borde, col="red", lwd=2)

plot(dat.rend$borde, dat.rend$rendimiento, main="Polinomio",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
```

```
     ylim=c(500,4500))
curve(m.pol$coefficients[1] + m.pol$coefficients[2]*x + m.pol$coefficients[3]*x^2,
      add=TRUE, col="red", lwd=2)
```

Complementamos la comparación con otros gráficos de interés:

```
res.borde <- resid(m.borde) # residuos rls
pred.borde <- fitted(m.borde) # predichos rls

res.pol <- resid(m.pol) # residuos polinomio
pred.pol <- fitted(m.pol) # predichos polinomio

plot(pred.borde, res.borde,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

plot(pred.pol, res.pol,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")

plot(pred.borde, dat.rend$rendimiento,
     xlab="Rendimiento predicho (kg/ha)",
     ylab="Rendimiento observado (kg/ha)",
     ylim=c(500,4500))
abline(a=0, b=1, col="red")

plot(pred.pol, dat.rend$rendimiento,
     xlab="Rendimiento predicho (kg/ha)",
     ylab="Rendimiento observado (kg/ha)",
     ylim=c(500,4500))
abline(a=0, b=1, col="red")
```

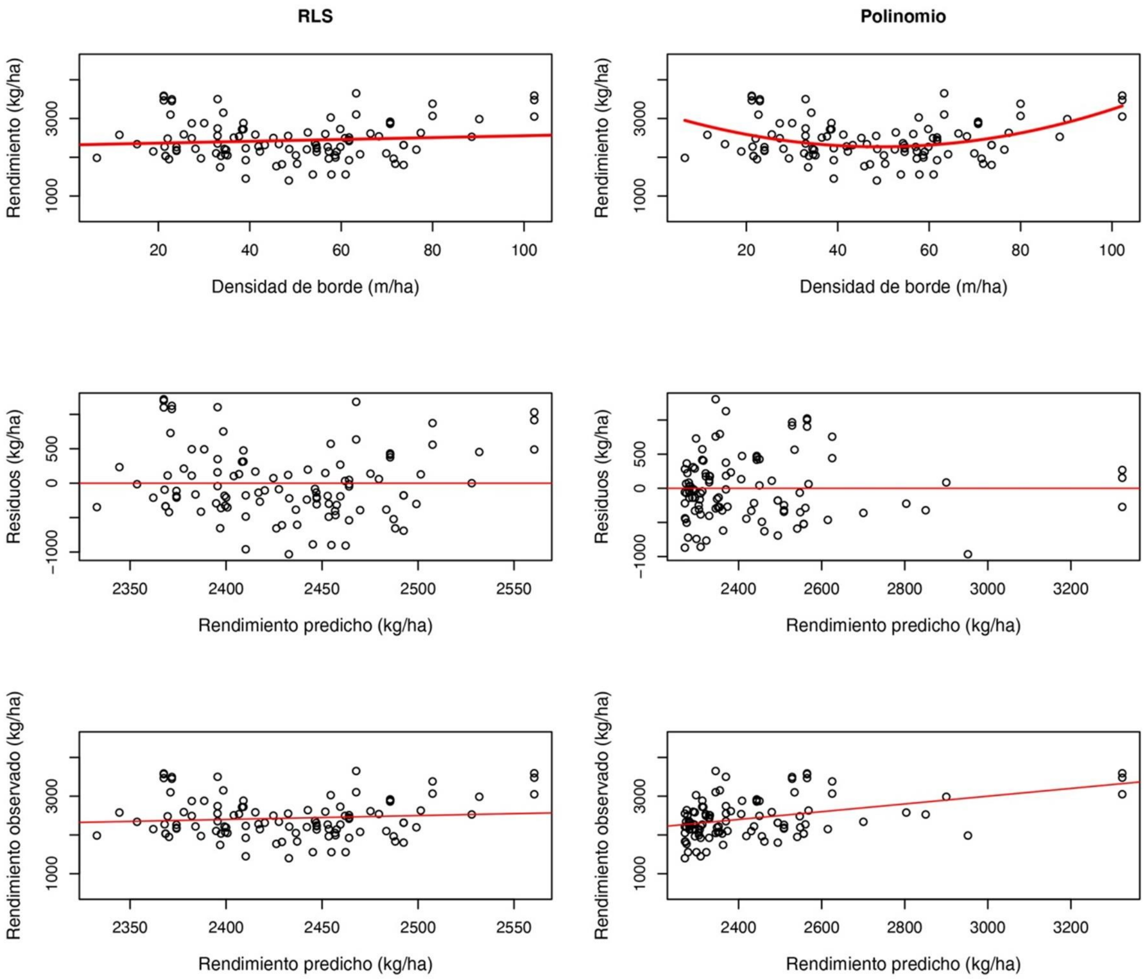

Figura 5. 12. Comparación gráfica del ajuste de una regresión lineal simple (RLS, panel izquierdo) frente al de un polinomio de segundo grado (panel derecho) para modelar el rendimiento en función de la densidad de borde

5. 11. Bondad de ajuste y capacidad predictiva

En este apartado discutiremos algunos conceptos que han cobrado reciente interés debido al auge del aprendizaje automatizado o *machine learning* (el tema se encuentra en cualquier libro de texto de esta rama del análisis de datos). No obstante, estos conceptos son intrínsecos a la estadística debido a que aluden a un aspecto central de esta disciplina: la inferencia.

5. 11. 1. Sobreajuste y capacidad predictiva

Los modelos estadísticos tienen dos grandes utilidades, por un lado, describir la variable respuesta y así facilitar la comprensión del problema bajo estudio y, por otro lado, predecir la variable respuesta de modo de prever cómo se comportaría el sistema ante situaciones no evaluadas, condiciones futuras, etcétera. En el primer caso, nos referimos a la capacidad de explicación del modelo para los datos observados y, en el segundo, a la capacidad predictiva sobre datos no observados.

Sea con fines explicativos o predictivos, ajustamos los modelos a los datos muestrales con el fin de realizar inferencias sobre una población que en general no conocemos en su totalidad. Debemos tener en cuenta que podemos explicar un porcentaje muy alto de variabilidad observada incluyendo más y más parámetros, llegando al 100 % de explicación cuando el número de parámetros de la componente lineal iguala el tamaño muestral. En este caso, el modelo coincidirá con lo observado en la muestra (la describe de manera perfecta) pero es tan complejo que solo nos servirá para este conjunto de datos en particular. Si el modelo enfatiza los detalles (el ruido aleatorio) de manera que no describe el patrón general de los datos (las tendencias promedio), se incurre en lo que se denomina *sobreajuste* y el resultado es una baja capacidad predictiva. En otras palabras, el modelo no nos permite ir más allá de la muestra lo cual invalida las inferencias poblacionales. Lo que debemos lograr es una descripción parsimoniosa de los datos, es decir, un compromiso entre el ajuste y la complejidad del modelo. El concepto de *parsimonia* es de suma relevancia y se expone en el capítulo 7.

Ahora bien, ¿cómo podemos evaluar el comportamiento del modelo fuera de la muestra? En definitiva, esta es nuestra única información sobre la población que queremos estudiar pero que no podemos conocer en su totalidad. Lo que necesitamos son datos independientes, es decir, que no formen parte del ajuste (que el modelo no haya visto, como se dice en la jerga). A estos se les llama datos de *validación* y pueden obtenerse, o bien recolectando nuevas observaciones, o bien usando una parte de la muestra que no participe del ajuste. En general se aplica la segunda estrategia, y la misma permite diferentes formas de *validación cruzada* (*cross-validation* en inglés). Por ejemplo, el método de *validación simple* consiste en dividir la muestra de forma aleatoria en dos subconjuntos. Con uno de ellos ajustamos el modelo (en el ámbito del *machine learning* a esto se lo conoce como *entrenamiento del modelo*) y con el otro lo validamos. Cualquiera sea el método, las métricas de validación se basan en las diferencias entre las predicciones (P) del modelo ajustado y los valores observados (O) en los datos independientes. Lógicamente, la capacidad predictiva del modelo es mayor cuanto menor resulten estas diferencias. Una de las métricas más utilizadas es el *error cuadrático medio* o RSME:

$$\mathrm{RSME} = \sqrt{\frac{\sum_{i=1}^{n}(P_i - O_i)^2}{n}}$$

donde *n* es la cantidad de datos usados para validar el modelo. Veamos cómo aplicar el método de validación simple a la relación entre rendimiento y densidad de borde y discutamos los resultados en el contexto de sobreajuste y capacidad predictiva.

5. 11. 2. Preparación de datos

Para usar el método de validación simple primero debemos separar la muestra (105 lotes) en dos subconjuntos (figura 5. 13). Usaremos 80 % de lotes para el ajuste (84 lotes) y 20 % para la validación (21 lotes):

```
set.seed(111) # para reproducir los resultados

filas.aj <- sort(sample(nrow(dat.rend), nrow(dat.rend)*0.8)) # 84 filas (105 x 0,8)

ajuste <- dat.rend[filas.aj, ] # extracción de las filas en la base de datos
validacion <- dat.rend[-filas.aj, ] # resto de las filas de la base de datos

nrow(ajuste) # número de los lotes para el ajuste

## [1] 84

nrow(validacion) # número de los lotes para la validación

## [1] 21

# figura 5.13
par(mfrow=c(1,2))
plot(ajuste$borde, ajuste$rendimiento,
     cex.lab=0.95, cex.axis=0.85, main="Lotes de ajuste (80%)",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     ylim=c(500,4500)) # lotes de ajuste

plot(validacion$borde, validacion$rendimiento,
     cex.lab=0.95, cex.axis=0.85, main="Lotes de validación (20%)",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     ylim=c(500,4500), pch=16) # lotes de validación
```

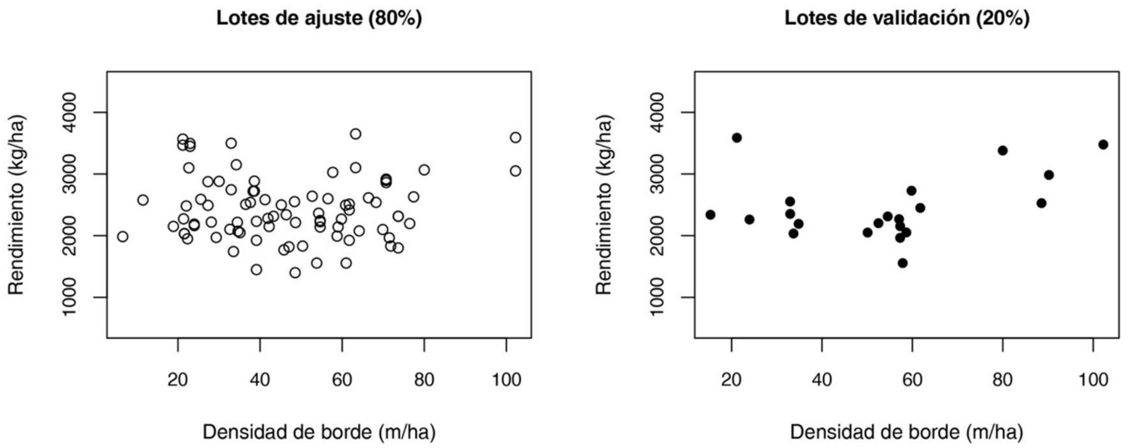

Figura 5. 13. Gráficos de dispersión entre rendimiento y densidad de borde en el subconjunto de datos (80 % de los lotes) usados para el ajuste del modelo (izquierda) y en el resto de los datos usados para la validación del modelo (derecha)

5. 11. 3. Ajuste

Usando el subconjunto que contiene el 80 % de lotes ajustamos modelos de complejidad creciente: desde una regresión lineal simple hasta un polinomio de grado 10 y, luego, para cada uno extraemos medidas de bondad de ajuste (R^2, R^2 ajustado y error estándar residual). Antes de ajustar los modelos debemos crear las variables que son potencia de la densidad de borde e insertarlas dentro de los datos de ajuste (recordemos que podríamos ajustar los modelos usando la función `I()` y evitar la creación de vectores adicionales):

```
ajuste$borde3 <- ajuste$borde^3  # densidad de borde al cubo
ajuste$borde4 <- ajuste$borde^4  # densidad de borde a la cuarta
ajuste$borde5 <- ajuste$borde^5  # densidad de borde a la quinta
ajuste$borde6 <- ajuste$borde^6  # densidad de borde a la sexta
ajuste$borde7 <- ajuste$borde^7  # densidad de borde a la séptima
ajuste$borde8 <- ajuste$borde^8  # densidad de borde a la octava
ajuste$borde9 <- ajuste$borde^9  # densidad de borde a la novena
ajuste$borde10 <-ajuste$borde^10 # densidad de borde a la décima
```

Ahora ajustamos los modelos al 80 % de los lotes y extraemos medidas de bondad de ajuste:

```
# RLS
m.rls.ajuste <- lm(rendimiento ~ borde, data=ajuste)

# polinomio grado 2
m.pol2.ajuste <- lm(rendimiento ~ borde + borde2,
                                  data=ajuste)
# polinomio grado 3
m.pol3.ajuste <- lm(rendimiento ~ borde + borde2 + borde3,
                                  data=ajuste)
# polinomio grado 4 al 10
m.pol4.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4,
                                  data=ajuste)
m.pol5.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5,
                                  data=ajuste)

m.pol6.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5 +
                                  borde6,
                                  data=ajuste)

m.pol7.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5 +
                                  borde6 + borde7,
                                  data=ajuste)

m.pol8.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5 +
                                  borde6 + borde7 + borde8,
                                  data=ajuste)

m.pol9.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5 +
                                  borde6 + borde7 + borde8 + borde9,
                                  data=ajuste)

m.pol10.ajuste <- lm(rendimiento ~ borde + borde2 + borde3 + borde4 + borde5 +
                                   borde6 + borde7 + borde8 + borde9 + borde10,
                                   data=ajuste)

# R²
r2.rls <- summary(m.rls.ajuste)$r.squared      # m.rls
r2.pol2 <- summary(m.pol2.ajuste)$r.squared    # m.pol2
r2.pol3 <- summary(m.pol3.ajuste)$r.squared    # m.pol3
r2.pol4 <- summary(m.pol4.ajuste)$r.squared    # m.pol4
r2.pol5 <- summary(m.pol5.ajuste)$r.squared    # m.pol5
r2.pol6 <- summary(m.pol6.ajuste)$r.squared    # m.pol6
r2.pol7 <- summary(m.pol7.ajuste)$r.squared    # m.pol7
r2.pol8 <- summary(m.pol8.ajuste)$r.squared    # m.pol8
r2.pol9 <- summary(m.pol9.ajuste)$r.squared    # m.pol9
r2.pol10 <- summary(m.pol10.ajuste)$r.squared  # m.pol10

# R² ajustado
r2aj.rls <- summary(m.rls.ajuste)$adj.r.squared     # m.rls
r2aj.pol2 <- summary(m.pol2.ajuste)$adj.r.squared   # m.pol2
r2aj.pol3 <- summary(m.pol3.ajuste)$adj.r.squared   # m.pol3
```

```r
r2aj.pol4 <- summary(m.pol4.ajuste)$adj.r.squared   # m.pol4
r2aj.pol5 <- summary(m.pol5.ajuste)$adj.r.squared   # m.pol5
r2aj.pol6 <- summary(m.pol6.ajuste)$adj.r.squared   # m.pol6
r2aj.pol7 <- summary(m.pol7.ajuste)$adj.r.squared   # m.pol7
r2aj.pol8 <- summary(m.pol8.ajuste)$adj.r.squared   # m.pol8
r2aj.pol9 <- summary(m.pol9.ajuste)$adj.r.squared   # m.pol9
r2aj.pol10 <- summary(m.pol10.ajuste)$adj.r.squared # m.pol10

# error estándar residual
s.rls <- summary(m.rls.ajuste)$sigma     # m.rls
s.pol2 <- summary(m.pol2.ajuste)$sigma   # m.pol2
s.pol3 <- summary(m.pol3.ajuste)$sigma   # m.pol3
s.pol4 <- summary(m.pol4.ajuste)$sigma   # m.pol4
s.pol5 <- summary(m.pol5.ajuste)$sigma   # m.pol5
s.pol6 <- summary(m.pol6.ajuste)$sigma   # m.pol6
s.pol7 <- summary(m.pol7.ajuste)$sigma   # m.pol7
s.pol8 <- summary(m.pol8.ajuste)$sigma   # m.pol8
s.pol9 <- summary(m.pol9.ajuste)$sigma   # m.pol9
s.pol10 <- summary(m.pol10.ajuste)$sigma # m.pol10
```

5. 11. 4. Validación y capacidad predictiva

El próximo paso es validar los modelos sobre el 20 % restante de lotes usando el RSME como medida de capacidad predictiva. Para esto necesitamos las predicciones de los modelos en estos lotes, lo cual requiere que generemos otra vez las variables que son potencia de la densidad de borde en los datos de validación:

```r
names(validacion)

##  [1] "lote"        "rendimiento" "borde"       "tamanio"     "N"
##  [6] "dsiembra"    "fsiembra"    "cprevio"     "region"      "Nt"
## [11] "borde2"

validacion$borde3 <- validacion$borde^3
validacion$borde4 <- validacion$borde^4
validacion$borde5 <- validacion$borde^5
validacion$borde6 <- validacion$borde^6
validacion$borde7 <- validacion$borde^7
validacion$borde8 <- validacion$borde^8
validacion$borde9 <- validacion$borde^9
validacion$borde10 <-validacion$borde^10

n.val <- length(validacion$rendimiento) # nº de lotes de validación

## [1] 21
```

Luego extraemos las predicciones de cada modelo:

```r
m.rls.pred <- predict(m.rls.ajuste, validacion)     # m.rls
m.pol2.pred <- predict(m.pol2.ajuste, validacion)   # m.pol2
m.pol3.pred <- predict(m.pol3.ajuste, validacion)   # m.pol3
m.pol4.pred <- predict(m.pol4.ajuste, validacion)   # m.pol4
m.pol5.pred <- predict(m.pol5.ajuste, validacion)   # m.pol5
m.pol6.pred <- predict(m.pol6.ajuste, validacion)   # m.pol6
m.pol7.pred <- predict(m.pol7.ajuste, validacion)   # m.pol7
m.pol8.pred <- predict(m.pol8.ajuste, validacion)   # m.pol8
m.pol9.pred <- predict(m.pol9.ajuste, validacion)   # m.pol9
m.pol10.pred <- predict(m.pol10.ajuste, validacion) # m.pol10
```

y a estas les restamos los valores observados para obtener el vector de errores sobre el cual calculamos el RSME:

```
errores.m.rls <- m.rls.pred - validacion$rendimiento     # errores m.rls
rsme.m.rls <- sqrt(sum((errores.m.rls)^2)/n.val)         # RSME m.rls

errores.m.pol2 <- m.pol2.pred - validacion$rendimiento   # errores m.pol2
rsme.m.pol2 <- sqrt(sum((errores.m.pol2)^2)/n.val)       # RSME m.pol2

errores.m.pol3 <- m.pol3.pred - validacion$rendimiento   # errores m.pol3
rsme.m.pol3 <- sqrt(sum((errores.m.pol3)^2)/n.val)       # RSME m.pol3

errores.m.pol4 <- m.pol4.pred - validacion$rendimiento   # errores m.pol4
rsme.m.pol4 <- sqrt(sum((errores.m.pol4)^2)/n.val)       # RSME m.pol4

errores.m.pol5 <- m.pol5.pred - validacion$rendimiento   # errores m.pol5
rsme.m.pol5 <- sqrt(sum((errores.m.pol5)^2)/n.val)       # RSME m.pol5

errores.m.pol6 <- m.pol6.pred - validacion$rendimiento   # errores m.pol6
rsme.m.pol6 <- sqrt(sum((errores.m.pol6)^2)/n.val)       # RSME m.pol6

errores.m.pol7 <- m.pol7.pred - validacion$rendimiento   # errores m.pol7
rsme.m.pol7 <- sqrt(sum((errores.m.pol7)^2)/n.val)       # RSME m.pol7

errores.m.pol8 <- m.pol8.pred - validacion$rendimiento   # errores m.pol8
rsme.m.pol8 <- sqrt(sum((errores.m.pol8)^2)/n.val)       # RSME m.pol8

errores.m.pol9 <- m.pol9.pred - validacion$rendimiento   # errores m.pol9
rsme.m.pol9 <- sqrt(sum((errores.m.pol9)^2)/n.val)       # RSME m.pol9

errores.m.pol10 <- m.pol10.pred - validacion$rendimiento # errores m.pol10
rsme.m.pol10 <- sqrt(sum((errores.m.pol10)^2)/n.val)     # RSME m.pol10
```

5. 11. 5. Parsimonia entre ajuste y complejidad

Antes mencionamos que debemos lograr una descripción parsimoniosa de los datos. El Criterio de información de Akaike (AIC, por sus siglas en inglés) es un índice que considera el compromiso entre ajuste y complejidad del modelo. Como vemos con más detalle en el capítulo 7, los modelos con menor AIC se consideran más parsimoniosos. Con la función `AIC()` podemos extraer los AIC de los modelos ajustados:

```
aic.rls <- AIC(m.rls.ajuste)      # m.rls
aic.pol2 <- AIC(m.pol2.ajuste)    # m.pol2
aic.pol3 <- AIC(m.pol3.ajuste)    # m.pol3
aic.pol4 <- AIC(m.pol4.ajuste)    # m.pol4
aic.pol5 <- AIC(m.pol5.ajuste)    # m.pol5
aic.pol6 <- AIC(m.pol6.ajuste)    # m.pol6
aic.pol7 <- AIC(m.pol7.ajuste)    # m.pol7
aic.pol8 <- AIC(m.pol8.ajuste)    # m.pol8
aic.pol9 <- AIC(m.pol9.ajuste)    # m.pol3
aic.pol10 <- AIC(m.pol10.ajuste) # m.pol10
```

Resumimos la información sobre bondad de ajuste, capacidad predictiva y parsimonia de los modelos ajustados:

```
r2 <- round(c(r2.rls, r2.pol2, r2.pol3, r2.pol4, r2.pol5,
              r2.pol6, r2.pol7, r2.pol8, r2.pol9, r2.pol10), 3)

r2aj <- round(c(r2aj.rls, r2aj.pol2, r2aj.pol3, r2aj.pol4, r2aj.pol5,
                r2aj.pol6, r2aj.pol7, r2aj.pol8, r2aj.pol9, r2aj.pol10), 3)

s <- round(c(s.rls, s.pol2, s.pol3, s.pol4, s.pol5,
             s.pol6, s.pol7, s.pol8, s.pol9, s.pol10), 1)

rsme <- round(c(rsme.m.rls, rsme.m.pol2, rsme.m.pol3, rsme.m.pol4, rsme.m.pol5,
                rsme.m.pol6, rsme.m.pol7, rsme.m.pol8, rsme.m.pol9, rsme.m.pol10), 1)

aic <- round(c(aic.rls, aic.pol2, aic.pol3, aic.pol4, aic.pol5,
               aic.pol6, aic.pol7, aic.pol8, aic.pol9, aic.pol10), 1)

tabla.avp <- data.frame(r2, r2aj, s, rsme, aic)
rownames(tabla.avp) <- c("RLS", "Grado 2", "Grado 3", "Grado 4", "Grado 5",
                         "Grado 6", "Grado 7", "Grado 8", "Grado 9","Grado 10")
round(tabla.avp, 3)

##              r2   r2aj     s   rsme    aic
## RLS       0.001 -0.011 520.1  503.8 1293.0
## Grado 2   0.101  0.079 496.5  392.8 1286.2
## Grado 3   0.117  0.084 495.1  395.8 1286.7
## Grado 4   0.145  0.102 490.2  375.8 1285.9
## Grado 5   0.160  0.106 488.9  381.3 1286.5
## Grado 6   0.162  0.096 491.7  405.2 1288.3
## Grado 7   0.166  0.089 493.7  397.2 1289.9
## Grado 8   0.197  0.112 487.5 1418.2 1288.7
## Grado 9   0.214  0.118 485.7 3391.4 1288.9
## Grado 10 0.217  0.110 488.0 5305.3 1290.6
```

Vemos gráficamente cómo cambian estas medidas en función de la complejidad del modelo, es decir, del grado del polinomio (figura 5. 14):

```
# figura 5.14
grado <- seq(1:10)
par(mfrow=c(3,2))
plot(grado, tabla.avp$r2,
     xlab="Grado del polinomio", ylab="R2",
     type="b", col="blue", pch=16)

plot(grado, tabla.avp$r2aj,
     xlab="Grado del polinomio", ylab="R2 ajustado",
     type="b", col="blue", pch=16)

plot(grado, tabla.avp$s,
     xlab="Grado del polinomio", ylab="Error estándar residual",
     type="b", col="blue", pch=16)

plot(grado, tabla.avp$rsme,
     xlab="Grado del polinomio", ylab="RSME",
     type="b", col="blue", pch=16)

plot(grado, tabla.avp$aic,
     xlab="Grado del polinomio", ylab="AIC",
     type="b", col="blue", pch=16)
```

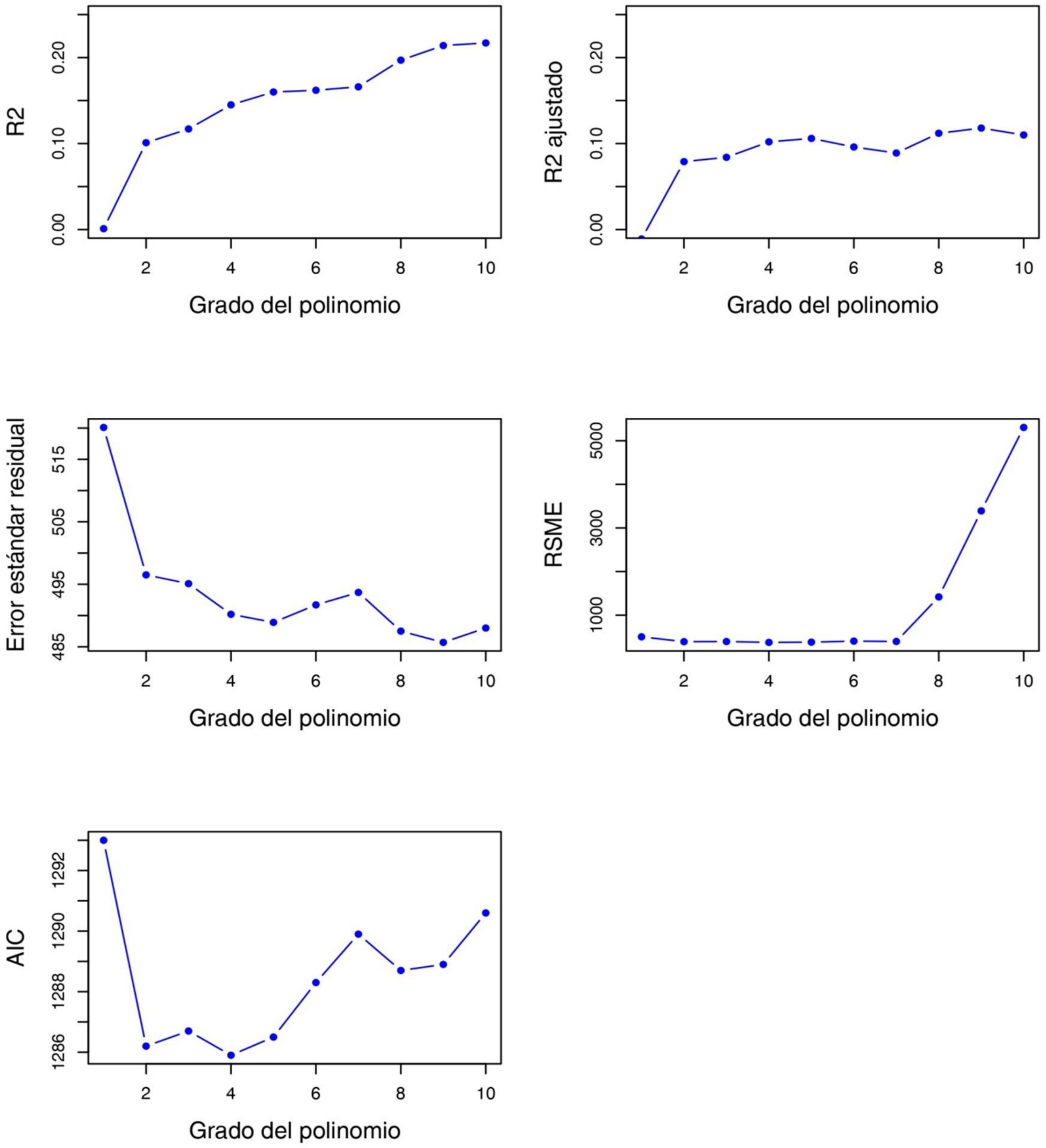

Figura 5. 14. Medidas de bondad de ajuste, capacidad predictiva y parsimonia en función del grado del polinomio

Encontramos que la bondad de ajuste aumenta con el grado del polinomio. Esto es, a mayor complejidad, mayor R^2 y menor error estándar residual. Sin embargo, cuando el modelo se vuelve demasiado complejo (grado 7), el error de predicción (RSME) comienza a aumentar de manera exponencial. Esto indica que estamos sobreajustando el modelo. De acuerdo al AIC, la descripción más parsimoniosa se obtiene entre el grado 2 y el 4. Notemos que el R^2 ajustado, el cual penaliza por el número de parámetros, indica que la bondad de ajuste no mejora sustancialmente luego del grado 2. Para verlo concretamente, comparamos gráficamente el ajuste y la capacidad predictiva de algunos de estos modelos (figura 5. 15):

```r
# figura 5.15
# gráficos ajuste
par(mfrow=c(2,4))
plot(ajuste$borde, ajuste$rendimiento, main="RLS",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500))
abline(m.rls.ajuste, col="red", lwd=2)

plot(ajuste$borde, ajuste$rendimiento, main="Polinomio de grado 2",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500))
curve(m.pol2.ajuste$coefficients[1] + m.pol2.ajuste$coefficients[2]*x + m.pol2.ajuste$co
efficients[3]*x^2,
      add=TRUE, col="red", lwd=2)

plot(ajuste$borde, ajuste$rendimiento, main="Polinomio de grado 3",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500))
curve(m.pol3.ajuste$coefficients[1] + m.pol3.ajuste$coefficients[2]*x +
        m.pol3.ajuste$coefficients[3]*x^2 + m.pol3.ajuste$coefficients[4]*x^3,
      add=TRUE, col="red", lwd=2)

plot(ajuste$borde, ajuste$rendimiento, main="Polinomio de grado 10",
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500))
curve(m.pol10.ajuste$coefficients[1] +
        m.pol10.ajuste$coefficients[2]*x + m.pol10.ajuste$coefficients[3]*x^2 +
        m.pol10.ajuste$coefficients[4]*x^3 + m.pol10.ajuste$coefficients[5]*x^4 +
        m.pol10.ajuste$coefficients[6]*x^5 + m.pol10.ajuste$coefficients[7]*x^6 +
        m.pol10.ajuste$coefficients[8]*x^7 + m.pol10.ajuste$coefficients[9]*x^8 +
        m.pol10.ajuste$coefficients[10]*x^9 + m.pol10.ajuste$coefficients[11]*x^10,
      add=TRUE, col="red", lwd=2)

# gráficos validación
plot(validacion$borde, validacion$rendimiento,
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500), pch=16)
abline(m.rls.ajuste, col="red", lwd=2)

plot(validacion$borde, validacion$rendimiento,
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500), pch=16)
curve(m.pol2.ajuste$coefficients[1] + m.pol2.ajuste$coefficients[2]*x + m.pol2.ajuste$co
efficients[3]*x^2,
      add=TRUE, col="red", lwd=2)

plot(validacion$borde, validacion$rendimiento,
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105),ylim=c(500,4500), pch=16)
curve(m.pol3.ajuste$coefficients[1] + m.pol3.ajuste$coefficients[2]*x +
        m.pol3.ajuste$coefficients[3]*x^2 + m.pol3.ajuste$coefficients[4]*x^3,
      add=TRUE, col="red", lwd=2)

plot(validacion$borde, validacion$rendimiento,
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     xlim=c(5,105), ylim=c(500,4500), pch=16)
curve(m.pol10.ajuste$coefficients[1] +
        m.pol10.ajuste$coefficients[2]*x + m.pol10.ajuste$coefficients[3]*x^2 +
        m.pol10.ajuste$coefficients[4]*x^3 + m.pol10.ajuste$coefficients[5]*x^4 +
        m.pol10.ajuste$coefficients[6]*x^5 + m.pol10.ajuste$coefficients[7]*x^6 +
        m.pol10.ajuste$coefficients[8]*x^7 + m.pol10.ajuste$coefficients[9]*x^8 +
        m.pol10.ajuste$coefficients[10]*x^9 + m.pol10.ajuste$coefficients[11]*x^10,
      add=TRUE, col="red", lwd=2)
```

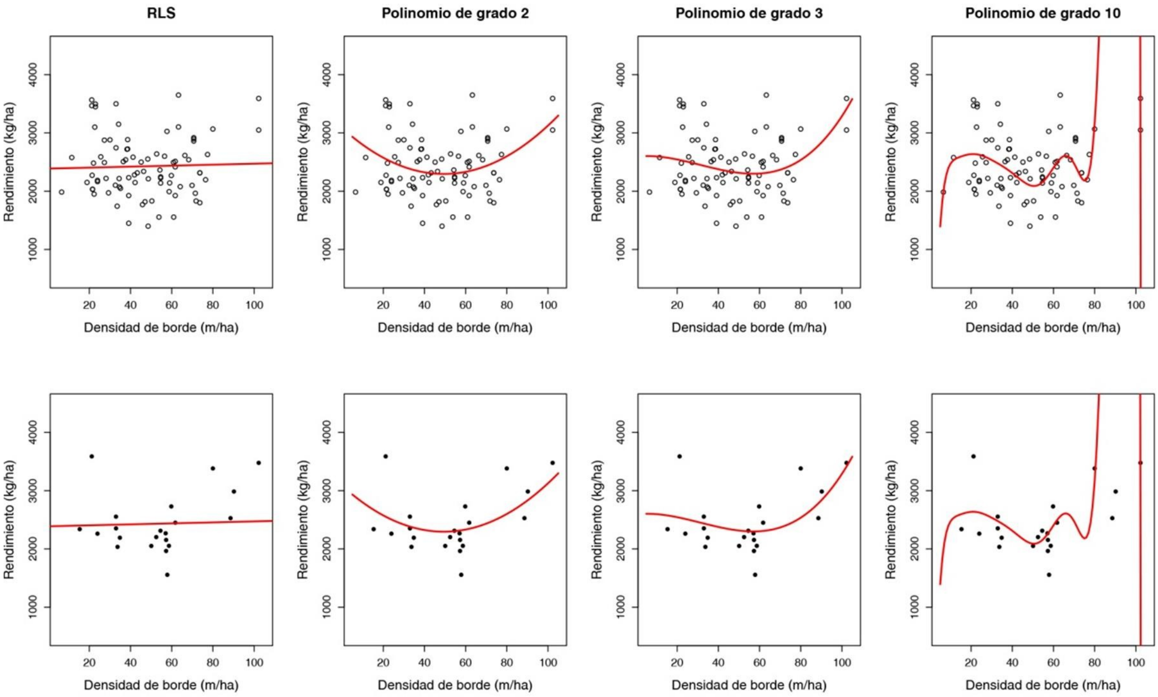

Figura 5. 15. Ajuste (panel superior) y capacidad predictiva (panel inferior) de cuatro modelos de complejidad creciente (de izquierda a derecha: regresión lineal simple, polinomio de grado 2, 3 y 10)

Detengámonos en lo que ocurre en la figura 5. 15 con el polinomio de grado 10 (gráficos del extremo derecho). Este tiende a copiar el rendimiento de los lotes de ajuste, y en consecuencia alcanza el menor error estándar residual. Para lograrlo, el modelo predice que el rendimiento cae abruptamente a densidades de borde algo menores a 10 m/ha (se vuelve negativo a ≈3 m/ha), y entre los 80 y 100 m/ha de densidad de borde indica que el rendimiento aumenta hasta superar la escala gráfica del eje *y* (la curva tiene un máximo de ≈ 42 500 kg/ha a una densidad de borde de 97 m/ha) para luego descender prácticamente en forma vertical (se vuelve negativo cuando se superan los 102 m/ha de densidad de borde). Esta curva contradice el patrón que muestran los datos, además de que un comportamiento de este tipo es ilógico y no podría explicarse conceptualmente. Claramente, el polinomio de grado 10 no ofrece una descripción parsimoniosa y, de acuerdo al AIC, este modelo no debiera elegirse por sobre un modelo cuadrático o cúbico a pesar de tener un R^2 del doble de alto (figura 5. 14).

La validación simple presenta el problema de que la capacidad predictiva varía considerablemente con las unidades (lotes en este caso) que resultan seleccionadas en cada subconjunto. Además, el hecho de no utilizar toda la información en el ajuste lo vuelve un método demandante en cantidad de datos. Como alternativa, se puede utilizar la validación cruzada de k iteraciones (*k-fold*). En esta, lo que hacemos es: 1) dividir la muestra en *k* subconjuntos seleccionados aleatoriamente, 2) ajustar el modelo con los datos de *k* - 1 subconjuntos, 3) validar el modelo con los datos del subconjunto restante, 4) repetir el proceso hasta validar en los *k* subconjuntos. Podemos extender este método realizando múltiples selecciones aleatorias de *k* subconjuntos y en cada una repetir los pasos antes descriptos. A esta variación se la conoce como *k iteraciones repetidas* (*repeated k-fold*).

Otra opción, sumamente útil cuando el tamaño muestral (n) es chico, es ajustar el modelo con n - 1 datos y validarlo con el restante, iterando hasta realizar la validación en los n datos de la muestra. Dado que en cada iteración dejamos de lado una única observación, a esta alternativa se la conoce como LOOVC (*leave one out cross-validation*). Los métodos de validación cruzada se basan en un muestreo sin reemplazo de los datos de la muestra, y se clasifican como métodos de remuestreo. También es posible remuestrear con reemplazo (*bootstrapping*), pero esta alternativa en general se utiliza para derivar la función de distribución (empírica) de la variable de interés cuando no es posible asumir una distribución teórica. Existen varios paquetes de R que realizan estos procedimientos automáticamente, entre ellos los paquetes `caret` y `cvms`.

Capítulo 6. Regresión con variables categóricas

6. 1. Introducción

En este capítulo avanzamos hacia modelos que combinan los predictores continuos de las regresiones con los categóricos de los modelos factoriales. La literatura tradicional expresa que estos modelos están vinculados al análisis de la covarianza (ANCOVA). No presentamos nuevos conceptos estadísticos; nos centramos en la interpretación de los parámetros del modelo, en particular, en los correspondientes a la interacción entre el predictor continuo y el factor.

6. 2. Problema

En el capítulo anterior buscamos determinar el efecto de la configuración del paisaje sobre el rendimiento. Partíamos de la hipótesis de que los bordes de cultivo benefician el rendimiento. Ahora, además, nos interesa evaluar si el efecto de la configuración del paisaje, en particular la densidad de borde, difiere entre regiones.

6. 3. Datos

Cargamos los datos:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",")
dat.rend$region <- as.factor(dat.rend$region) # indicamos que region es un factor
dat.rend3 <- dat.rend[!dat.rend$region=="rd", ] # removemos los lotes de la región d
dat.rend3$region <- with(dat.rend3, factor(region, levels=c("ra", "rb", "rc")))
levels(dat.rend3$region)

## [1] "ra" "rb" "rc"
```

Y, de acuerdo al problema, los exploramos numéricamente:

```
by(dat.rend[, c(2,3)] , dat.rend$region, summary) # resumen numérico por región

## dat.rend$region: ra
##   rendimiento       borde
##  Min.   :1770   Min.   : 6.55
##  1st Qu.:2306   1st Qu.:22.16
##  Median :2720   Median :32.96
##  Mean   :2767   Mean   :35.52
##  3rd Qu.:3458   3rd Qu.:43.52
##  Max.   :3650   Max.   :63.28
## ------------------------------------------------------------
## dat.rend$region: rb
##   rendimiento       borde
##  Min.   :2051   Min.   :22.03
##  1st Qu.:2149   1st Qu.:42.21
##  Median :2269   Median :56.53
##  Mean   :2374   Mean   :53.63
```

```
##   3rd Qu.:2539   3rd Qu.:60.84
##   Max.   :3027   Max.   :90.21
## ------------------------------------------------------------
## dat.rend$region: rc
##    rendimiento        borde
##   Min.   :1400   Min.   : 21.57
##   1st Qu.:1958   1st Qu.: 38.87
##   Median :2233   Median : 54.27
##   Mean   :2303   Mean   : 53.99
##   3rd Qu.:2615   3rd Qu.: 70.71
##   Max.   :3592   Max.   :102.24
## ------------------------------------------------------------
## dat.rend$region: rd
##    rendimiento        borde
##   Min.   :2049   Min.   :11.48
##   1st Qu.:2154   1st Qu.:19.83
##   Median :2359   Median :23.31
##   Mean   :2433   Mean   :24.17
##   3rd Qu.:2572   3rd Qu.:30.70
##   Max.   :3100   Max.   :35.11
```

y gráficamente (es importante notar que creamos objetos para guardar cada región por separado):

```
# figura 6.1
par(mfrow=c(1,3))

# Rendimiento vs. densidad de borde
plot(dat.rend3$borde, dat.rend3$rendimiento, ylim=c(500,4500),
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)")

# Rendimiento vs. tamaño del lote
plot(dat.rend3$rendimiento ~ dat.rend3$region, ylim=c(500,4500),
     xlab="Región", ylab="Rendimiento (kg/ha)",
     col=c("red", "green", "darkgrey"))

# Rendimiento vs. densidad de borde por región
ra <- subset(dat.rend3, region=="ra")   # región a
rb <- subset(dat.rend3, region=="rb") # región b
rc <- subset(dat.rend3, region=="rc") # región c

plot(ra$borde, ra$rendimiento, ylim=c(500,4500),
     xlab="Densidad de borde (m/ha)", ylab="Rendimiento (kg/ha)",
     col="red", pch=16) # región a
abline(lm(ra$rendimiento ~ ra$borde), col="red")

points(rb$borde, rb$rendimiento, col="green", pch=15) # región b
abline(lm(rb$rendimiento ~ rb$borde), col="green")

points(rc$borde, rc$rendimiento, col="darkgrey", pch=17) # región c
abline(lm(rc$rendimiento ~ rc$borde), col="darkgrey")

legend("bottomright", legend=c("Región a","Región b","Región c"),
       col=c("red","green","darkgrey"),
       bty="n", pch=c(15,16,17), lwd=c(2,2,2))
```

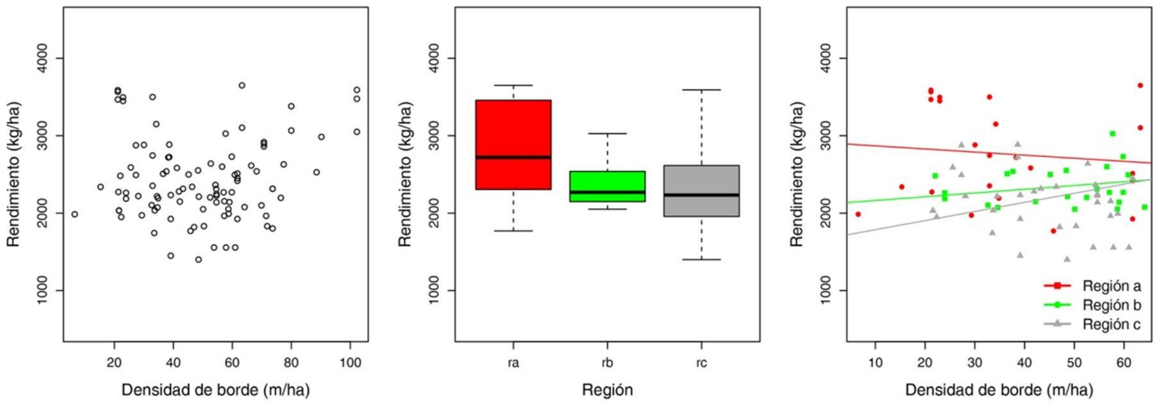

Figura 6. 1. Variabilidad del rendimiento en función de la densidad de borde y la región de manera individual (izquierda y medio, respectivamente) y conjuntamente (derecha)
Nota. En este caso mediante un gráfico de dispersión que incluye rectas de regresión ajustadas a para cada región por separado.

El primer gráfico de la figura 6. 1 muestra la relación entre densidad de borde y rendimiento, y el segundo la variabilidad del rendimiento entre regiones. Dado que estamos particularmente interesados en evaluar si el efecto de la densidad de borde sobre el rendimiento varía entre regiones, en el tercer gráfico distinguimos los lotes de cada región con diferentes colores. En este incluimos la recta que resulta de ajustar tres regresiones entre rendimiento y densidad de borde, una por cada nivel del factor región. Esto nos permite tener una idea de la interacción entre densidad de borde y región; rectas no paralelas implican que el efecto del borde no es el mismo en cada región. En este caso, el efecto positivo de la densidad de borde parece ser más marcado en los lotes de la región «c».

6. 4. Modelo y ajuste

Planteamos un modelo que considera los efectos de la región, la densidad de borde, y la interacción entre ambos predictores:

$$r_i = \beta_0 + \beta_1 \times rb_i + \beta_2 \times rc_i + \beta_3 \times b_i + \beta_4 \times rb_i \times b_i + \beta_5 \times rc_i \times b_i + \varepsilon_i$$
$$\varepsilon_i \sim \mathcal{N}(0;\, \sigma^2)_{independientes}$$

donde

r_i = rendimiento de girasol (kg ha^{-1}) del lote *i* (*i* = 1, 2, … N),
rb_i = 1 si el lote se localiza en la región b y 0 en caso contrario,
rc_i = 1 si el lote se localiza en la región c y 0 en caso contrario,
b_i = densidad de borde en el contexto (radio de 1,5 km) del lote *i* (m ha^{-1}),
ε_i = término de error (kg ha^{-1})

Al incluir predictores cuantitativos y categóricos, se combina la parametrización de los modelos de regresión con aquella de los modelos factoriales. Bajo una codificación *dummy* del factor, la ordenada al origen β_0 indica el rendimiento promedio de los lotes de la región «a» (región de referencia) rodeados por un paisaje sin bordes ($b = 0$), y β_3 expresa la variación promedio en el rendimiento de estos lotes ante aumentos unitarios en la densidad de borde. Geométricamente, β_1 y β_2 constituyen cambios en ordenada al origen. En este problema se expresan diferencias de rendimiento promedio entre las regiones «b» (β_1) y «c» (β_2) con respecto a la región «a» para lotes cuya densidad de borde es nula. Los parámetros de interacción entre región y densidad de borde, β_4 y β_5, constituyen cambios de pendiente. En nuestro caso indican cuánto varía el efecto de la densidad de bordes en las regiones «b» (β_4) y «c» (β_5) con respecto al efecto de este predictor en la región «a» (β_3). De este modelo se derivan tres rectas con las cuales se predice el rendimiento promedio de cada región. En la región «a»:

$$r_{ra} = \beta_0 + \beta_1 \times 0 + \beta_2 \times 0 + \beta_3 \times b + \beta_4 \times 0 \times b + \beta_5 \times 0 \times b$$
$$\Rightarrow$$
$$r_{ra} = \beta_0 + \beta_3 \times b$$

En la región «b»:

$$r_{rb} = \beta_0 + \beta_1 \times 1 + \beta_2 \times 0 + \beta_3 \times b + \beta_4 \times 1 \times b + \beta_5 \times 0 \times b$$
$$\Rightarrow$$
$$r_{rb} = (\beta_0 + \beta_1) + (\beta_3 + \beta_4) \times b$$

En la región «c»:

$$r_{rc} = \beta_0 + \beta_1 \times 0 + \beta_2 \times 1 + \beta_3 \times b + \beta_4 \times 0 \times b + \beta_5 \times 1 \times b$$
$$\Rightarrow$$
$$r_{rc} = (\beta_0 + \beta_2) + (\beta_3 + \beta_5) \times b$$

6. 4. 1. Ajuste del modelo

Dado que hemos planteado un modelo con interacción, para su ajuste utilizamos el operador `*`:

```
m.rcc <- lm(rendimiento ~ region * borde, data=dat.rend3) # ajuste del modelo
```

Al ser un estudio observacional, podría darse el caso que la densidad de borde dependa de (varíe con) la región, lo cual podría generar problemas en el análisis. La función `vif()` nos indica que no parece haber mayores problemas:

```
library(car)

## Loading required package: carData

vif(m.rcc) # no parece haber problemas de multicolinealidad

## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif

##                   GVIF Df GVIF^(1/(2*Df))
## region        60.308090  2        2.786724
## borde          6.599219  1        2.568894
## region:borde 148.714431  2        3.492113
```

Pedimos a R los coeficientes estimados junto con sus intervalos de confianza:

```
round(cbind("β est."=coef(m.rcc), confint(m.rcc)), 1) # IC estimaciones

##                  β est.   2.5 % 97.5 %
## (Intercept)      2908.7  2447.7 3369.8
## regionrb         -793.6 -1518.1  -69.1
## regionrc        -1240.0 -1836.7 -643.3
## borde              -4.0   -15.8    7.8
## regionrb:borde      8.8    -6.6   24.2
## regionrc:borde     15.7     2.3   29.2
```

La salida de R nos informa que el modelo para predecir el rendimiento promedio es el siguiente:

$$\hat{r} = 2908{,}7 - 793{,}6 \times rb - 1240{,}0 \times rc - 4{,}0 \times b - 8{,}8 \times rb \times b - 15{,}7 \times rc \times b$$

Graficar las rectas de a una a la vez nos puede ayudar a comprender mejor el modelo. Primero guardamos los coeficientes estimados:

```
b0 <- coef(m.rcc)[1]
b1 <- coef(m.rcc)[2]
b2 <- coef(m.rcc)[3]
b3 <- coef(m.rcc)[4]
b4 <- coef(m.rcc)[5]
b5 <- coef(m.rcc)[6]
```

Ahora agregamos los puntos por región para luego insertar cada recta desde la función `curve()`:

```
# figura 6.2
par(mfrow=c(1,1))
plot(ra$borde, ra$rendimiento, ylim=c(500,4500),
     xlab="Densidad de borde (m/ha)",
     ylab="Rendimiento (kg/ha)",
     col="red", pch=16) # lotes región a
curve(expr=b0 + b3*x, col="red", lwd=2, add=TRUE) # recta región a

points(rb$borde, rb$rendimiento, col="green", pch=15) # lotes región b
curve(expr=(b0+b1) + (b3+b4)*x, col="green", lwd=2, add=TRUE) # recta región b
```

```
points(rc$borde, rc$rendimiento, col="darkgrey", pch=17) # lotes región c
curve(expr=(b0+b2) + (b3+b5)*x, col="darkgrey", lwd=2, add=TRUE) # recta región c

legend("bottomleft", legend=c("Región a","Región b","Región c"),
       col=c("red","green","darkgrey"),
       bty="n", pch=c(15,16,17), lwd=c(2,2,2))
```

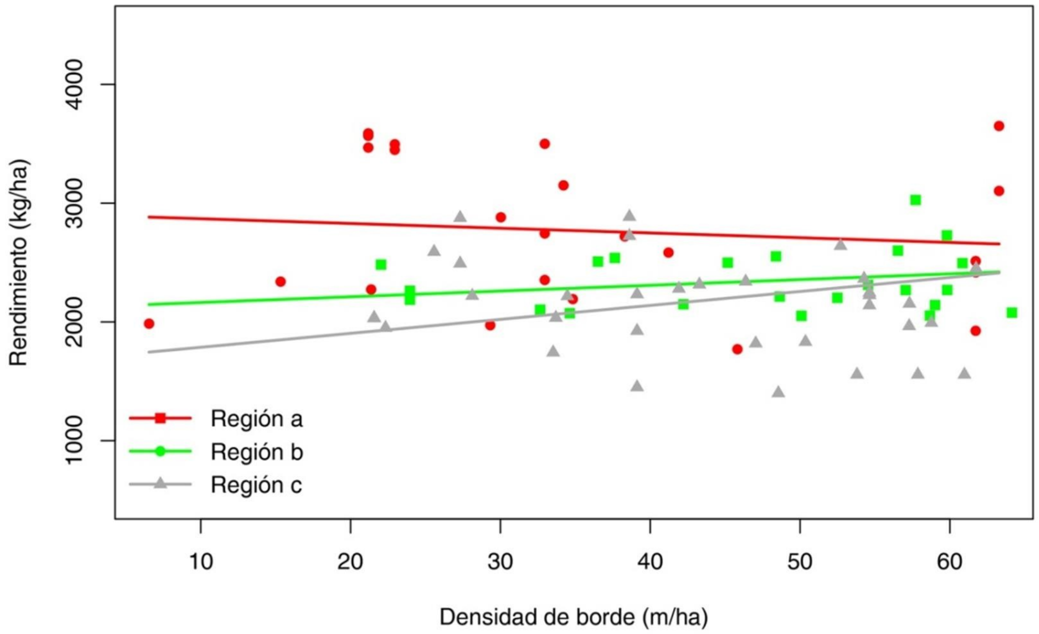

Figura 6. 2. Ajuste del modelo lineal que incluye como predictores la densidad de borde, una variable cuantitativa, y la región, un factor de tres niveles

Si parametrizamos el modelo con una codificación por desviación para la región, la recta de referencia (β_0 y β_3) indica el efecto promedio (figura 6. 3) y el resto de los parámetros representan diferencias en el intercepto (β_1 y β_2) y la pendiente (β_4 y β_5) de las regiones «a» y «b» con respecto a la recta promedio:

```
m.rcc.s <- lm(rendimiento ~ region * borde,
                            contrasts=list(region="contr.sum"),
                            data=dat.rend3) # codificación por desviación

round(cbind("β est."=coef(m.rcc.s), confint(m.rcc.s)), 1) # IC estimaciones por desv.

##                β est.  2.5 % 97.5 %
## (Intercept)   2230.9 1958.4 2503.4
## region1        677.9  296.9 1058.8
## region2       -115.7 -538.1  306.6
## borde            4.2   -1.4    9.8
## region1:borde   -8.2  -17.0    0.6
## region2:borde    0.6   -7.4    8.6

# figura 6.3 (recta promedio)
par(mfrow=c(1,1))
plot(dat.rend3$borde, dat.rend3$rendimiento, ylim=c(500,4500),
     xlab="Densidad de borde (m/ha)",
     ylab="Rendimiento (kg/ha)",
     pch=16)
curve(coef(m.rcc.s)[1] + coef(m.rcc.s)[4]*x, col="blue", lwd=3, add=TRUE) # recta
```

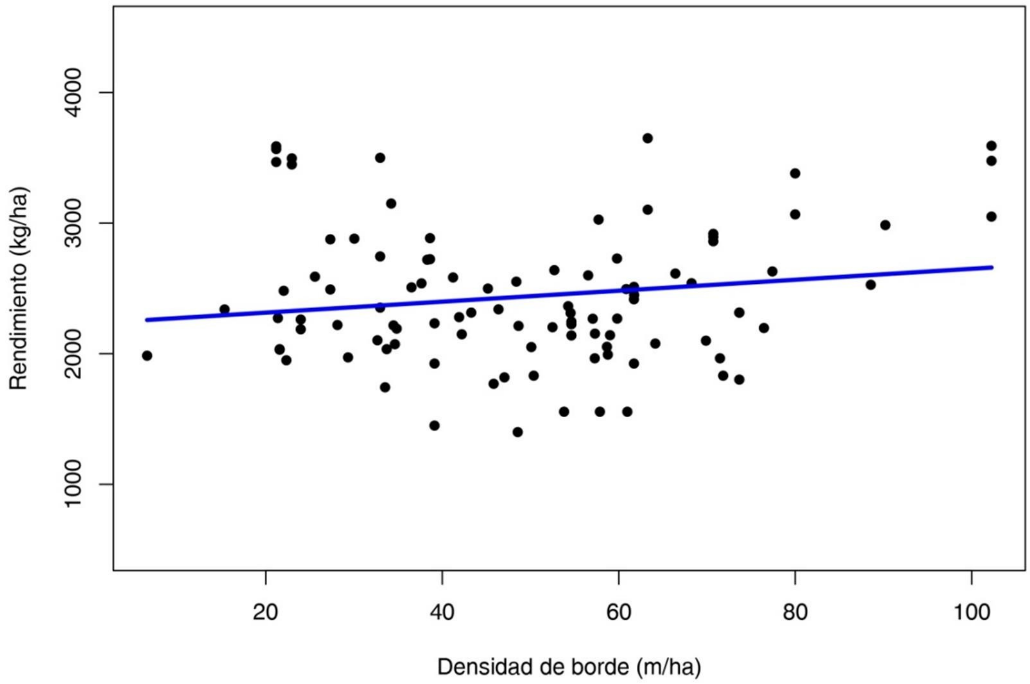

Figura 6. 3. Ajuste del modelo utilizando una codificación desviación para la región
Nota. Bajo esta parametrización, la recta de referencia representa el efecto promedio de la densidad de borde (línea azul).

6. 5. ANOVA

Al igual que con los modelos multifactoriales, la primera hipótesis a contrastar es la de la interacción. En nuestro modelo la ausencia de interacción implica que en la población de lotes las tres rectas son paralelas ($\beta_4 = \beta_5 = 0$). Evaluamos los efectos principales solo en el caso de no rechazar la hipótesis nula, es decir, la ausencia de interacción entre la densidad de borde y la región:

```
anova(m.rcc)

## Analysis of Variance Table
##
## Response: rendimiento
##               Df   Sum Sq Mean Sq F value    Pr(>F)
## region         2  3457470 1728735  7.9029 0.0006763 ***
## borde          1  1834194 1834194  8.3850 0.0047151 **
## region:borde   2  1244457  622229  2.8445 0.0632339 .
## Residuals     93 20343375  218746
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

El resultado del ANOVA nos indica que la interacción entre borde y región presenta una significancia marginal. En otras palabras, tenemos cierto indicio de que el efecto del borde cambia con la región (gráficamente se observa que la pendiente tiende a cambiar entre regiones) aunque la evidencia no es concluyente (valor p = 0,06). Si consideramos que la evidencia no es suficiente para afirmar que existe efecto de interacción, continuamos con la evaluación de los efectos principales y encontramos que el rendimiento varía entre regiones (valor p = 0,0006) y es afectado por la densidad de borde (valor p = 0,0047). En el caso de la región, debemos continuar con las comparaciones múltiples *a posteriori*.

6. 6. Intervalos de confianza y de predicción

Obtenemos los intervalos de confianza y de predicción del rendimiento promedio de lotes con densidades de borde contrastante (30 m ha^{-1} *versus* 90 m ha^{-1}) en cada una de las regiones. Para esto, primero generamos una tabla caracterizando a estos lotes:

```
nd <- data.frame(region=rep(c("ra", "rb", "rc"), times=2),
                 borde=rep(c(30,90), each=3))
nd # características de los lotes en los que se predice el rendimiento

##   region borde
## 1     ra    30
## 2     rb    30
## 3     rc    30
## 4     ra    90
## 5     rb    90
## 6     rc    90
```

Luego le pedimos a R los intervalos de confianza y de predicción para estos lotes:

```
nd$region <- as.factor(nd$region) # región debe estar como factor
round(predict(m.rcc, newdata=nd, interval="confidence"), 1) # IC

##      fit    lwr    upr
## 1 2789.0 2584.7 2993.3
## 2 2260.0 1969.1 2550.9
## 3 2021.3 1813.8 2228.8
## 4 2549.4 1879.1 3219.7
## 5 2549.6 2150.0 2949.2
## 6 2726.3 2454.2 2998.4

round(predict(m.rcc, newdata=nd, interval="prediction"), 1) # IP

##      fit    lwr    upr
## 1 2789.0 1838.0 3739.9
## 2 2260.0 1286.7 3233.2
## 3 2021.3 1069.6 2972.9
## 4 2549.4 1404.0 3694.8
## 5 2549.6 1538.5 3560.7
## 6 2726.3 1758.5 3694.1
```

La función `ggpredict()` del paquete `ggeffects` nos permite obtener los intervalos de confianza y de predicción para cada recta:

```
library(ggeffects)
ic.m.rcc <- ggpredict(m.rcc, c("borde", "region"), interval="confidence") # IC
ip.m.rcc <- ggpredict(m.rcc, c("borde", "region"), interval="prediction") # IP

str(ic.m.rcc[ , c(1:6)]) # información del objeto guardado

## Classes 'ggeffects' and 'data.frame':    36 obs. of  6 variables:
##  $ x        : num  0 0 0 10 10 10 20 20 20 30 ...
##  $ predicted: num  2909 2115 1669 2869 2163 ...
##  $ std.error: num  232 281 191 180 234 ...
##  $ conf.low : num  2454 1564 1295 2516 1704 ...
##  $ conf.high: num  3364 2667 2043 3222 2623 ...
##  $ group    : Factor w/ 3 levels "ra","rb","rc": 1 2 3 1 2 3 1 2 3 1 ...
```

Que podemos combinar con `ggplot()` para graficarlos:

```
library(ggplot2)

# figura 6.4
ggplot(ic.m.rcc, aes(x=x, y=predicted), color=group) +
  geom_line(aes(x=x, y=predicted, color=group), size=1.25, show.legend=FALSE) +
  geom_ribbon(aes(ymin=conf.low, ymax=conf.high, fill=group, color=NULL),
              alpha=0.125, show.legend=FALSE) +
  theme_bw() +
  geom_point(dat.rend3, mapping=aes(x=borde, y=rendimiento, color=region),
             size=1.5, show.legend = FALSE) +
  facet_wrap(~group) +
  geom_vline(xintercept = c(30,90), lty=2) +
  ylim(500, 4500) + xlab("Densidad de borde (m/ha)") + ylab("Rendimiento (kg/ha)") +
  scale_color_manual(values=c("red","green","darkgrey")) +
  scale_fill_manual(values=c("red","green","darkgrey"))
```

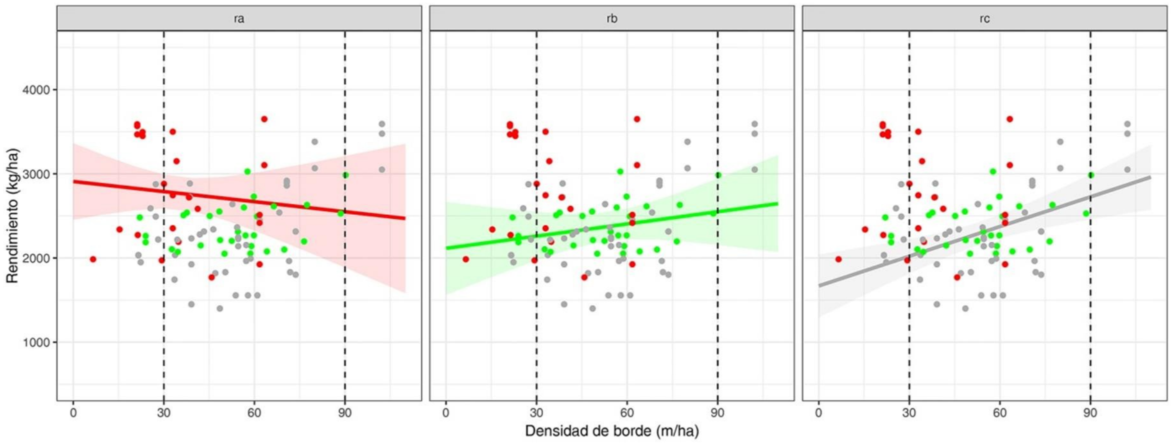

Figura 6. 4. Intervalos de confianza del rendimiento promedio predicho por el modelo en cada región

En los gráficos hemos agregado las líneas verticales a densidades de borde de 30 m ha^{-1} y 90 m ha^{-1} (figura 6.4). La comprensión del código de `ggplot()` queda como tarea para el lector.

6. 7. Bondad de ajuste

Evaluamos los indicadores de bondad de ajuste que venimos utilizando en el resto de los modelos:

```
summary(m.rcc)

##
## Call:
## lm(formula = rendimiento ~ region * borde, data = dat.rend3)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -955.80 -296.59  -14.67  292.83  993.92
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       2908.749    232.179  12.528  < 2e-16 ***
## regionrb          -793.598    364.856  -2.175   0.0322 *
## regionrc         -1240.003    300.479  -4.127 8.01e-05 ***
## borde               -3.993      5.932  -0.673   0.5025
## regionrb:borde       8.820      7.752   1.138   0.2581
## regionrc:borde      15.743      6.787   2.320   0.0226 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 467.7 on 93 degrees of freedom
## Multiple R-squared:  0.2432, Adjusted R-squared:  0.2025
## F-statistic: 5.976 on 5 and 93 DF,  p-value: 7.748e-05

summary(m.rcc)$sigma # error estándar residual

## [1] 467.7029

summary(m.rcc)$r.squared # R^2

## [1] 0.2431638

summary(m.rcc)$adj.r.squared # R^2 ajustado

## [1] 0.2024737
```

El coeficiente de determinación nos indica que la componente determinística del modelo ha capturado aproximadamente el 25 % de la variabilidad del rendimiento de los lotes. Además, el modelo tiene un error estándar residual de 468 kg ha^{-1}. Gráficamente:

```
predichos <- fitted(m.rcc) # predicciones del modelo para los lotes muestreados

# figura 6.5
par(mfrow=c(1,1))
with(dat.rend3, plot(predichos, rendimiento,
                     xlab="Rendimiento predicho (kg/ha)",
                     ylab="Rendimiento observado (kg/ha)",
                     ylim=c(500,4500))) # gráfico de observados vs. predichos
abline(a=0, b=1, lwd=3, col="brown") # línea 1 a 1
```

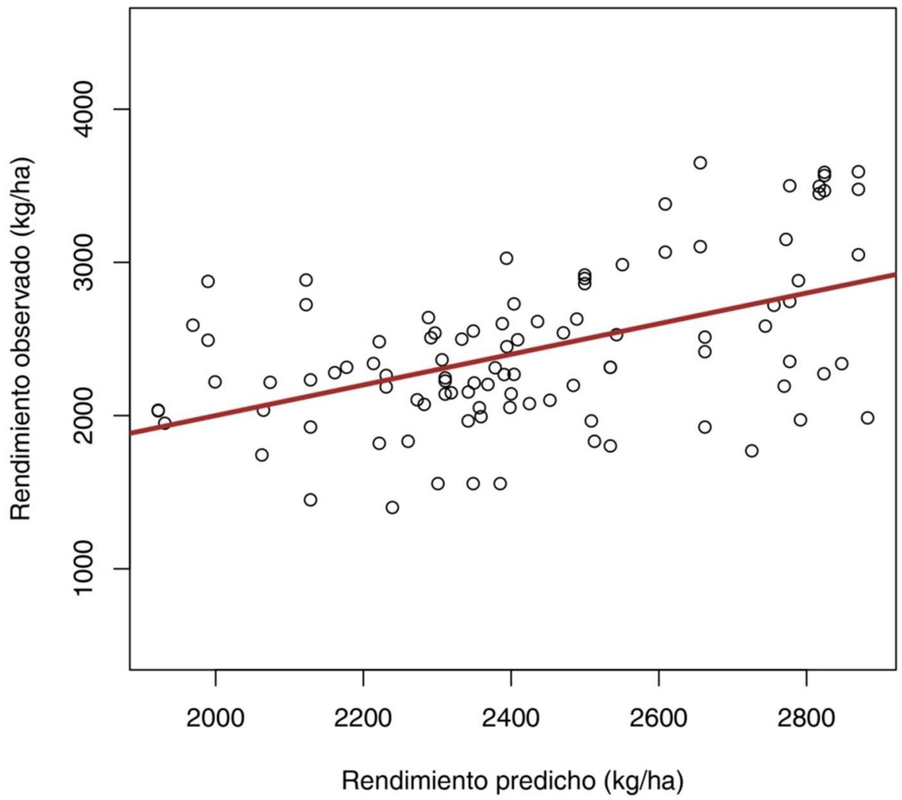

Figura 6. 5. Gráfico de observados *vs* predichos del modelo lineal que incluye la densidad de borde y la región como predictores (la línea representa la recta uno a uno)

6. 8. Validación de supuestos

Evaluamos los supuestos del modelo:

```
residuos <- resid(m.rcc) # residuos del modelo

# figura 6.6
par(mfrow=c(2,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend3$borde, residuos,
     xlab="Densidad de borde (m/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(residuos ~ dat.rend3$region,
     xlab="Región", ylab="Residuos (kg/ha)")
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")
```

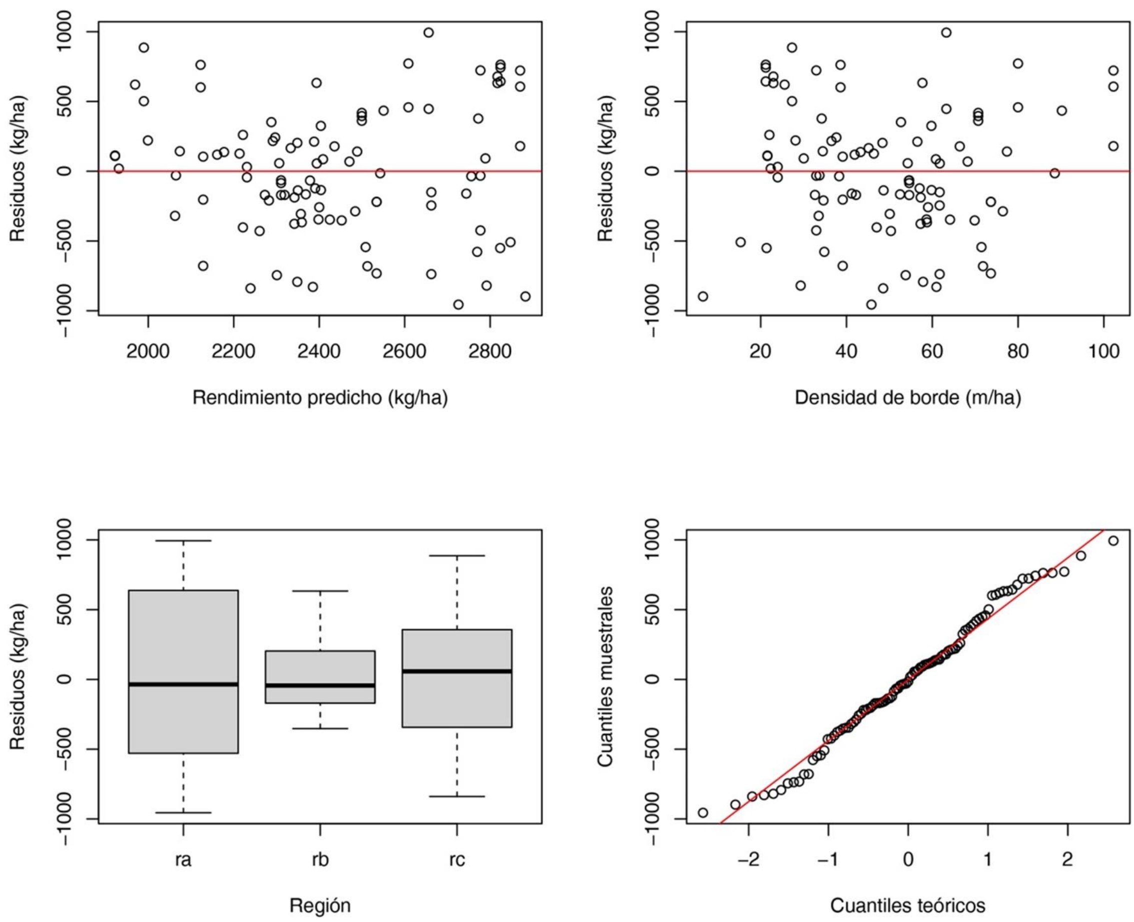

Figura 6. 6. Evaluación gráfica de los supuestos (linealidad, independencia, homocedasticidad y normalidad) del modelo lineal que incluye la densidad de borde y la región como predictores

```
# - Test Kolmogorov-Smirnov
ks.test(residuos,"pnorm", mean(residuos), sd(residuos))
## Warning in ks.test.default(residuos, "pnorm", mean(residuos), sd(residuos)):
## ties should not be present for the Kolmogorov-Smirnov test

##
##  Asymptotic one-sample Kolmogorov-Smirnov test
##
## data:  residuos
## D = 0.058153, p-value = 0.8913
## alternative hypothesis: two-sided
```

Al analizar los residuos (figura 6. 6) observamos que podría haber un problema de linealidad con la densidad de borde y de heterocedasticidad entre regiones. Esto puede deberse a la distribución del rendimiento *per se*, o a la falta de predictores importantes (ver capítulo 7). Entre las posibles soluciones en el contexto de los modelos lineales tenemos la transformación de variables (por ejemplo, aplicar logaritmos), la inclusión de polinomios (por ejemplo, densidad de borde al cuadrado), la inclusión de otros predictores (esto requiere que hayan sido medidos) e interacciones, e incluso utilizar modelos más complejos que incorporen funciones de varianzas (estos modelos están por fuera del alcance de este libro).

Capítulo 7. Modelo lineal general

7. 1. Introducción

En los capítulos anteriores, los modelos fueron introducidos etiquetándolos de acuerdo a como se los suele encontrar en los textos de estadística más tradicionales. Este capítulo expone que estos modelos no son más que componentes de un mismo marco conceptual. Para formalizar la construcción del modelo lineal general, utilizamos la notación matricial y exploramos su vínculo con el método de mínimos cuadrados ordinarios y la función `lm()`. Introducimos los criterios de información y los fundamentos de su uso para la selección de variables. Como aspecto complementario, la función `emmeans()` nos permite flexibilizar las comparaciones múltiples de modo de aplicarlas en el contexto de los modelos lineales generales.

7. 2. Problema

Al introducir los datos y la problemática a analizar establecimos que se pretendía explorar la relación entre producción y configuración del paisaje. Basados en un marco agroecológico y de intensificación ecológica, suponemos que los bordes de cultivo benefician el rendimiento y que lotes más grandes lo perjudican. En el capítulo 4 pusimos a prueba estas hipótesis conceptuales a partir de un modelo de regresión lineal múltiple. En este capítulo integrador hacemos lo mismo, pero modelamos el rendimiento del cultivo aplicando todas las herramientas que hemos visto a lo largo del libro. La densidad de borde y el tamaño del lote son las variables de interés para la investigación, y son analizadas junto a un conjunto de predictores ambientales y de manejo que *a priori* afectan el rendimiento del lote: región, fertilización nitrogenada, cultivo previo, densidad de siembra y fecha de siembra.

7. 3. Datos

Cargamos los datos:

```
dat.rend <- read.table("lotes.txt", header=TRUE, dec=",",
                       stringsAsFactors=TRUE)

# recodificamos los niveles del cultivo previo
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="gram_anual"] <- "pastura"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="verdeo"] <- "pastura"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="trigo"] <- "cereal"
levels(dat.rend$cprevio)[levels(dat.rend$cprevio)=="maiz"] <- "cereal"

# removemos los lotes de la región del sudoeste
dat.rend2 <- dat.rend[!dat.rend$region=="rd", ]
dat.rend2$region <- with(dat.rend2, factor(region, levels=c("ra", "rb", "rc")))

dat.rend2$fsiembra <- as.Date(dat.rend2$fsiembra, format="%d/%m/%Y")
```

Y los exploramos gráficamente:

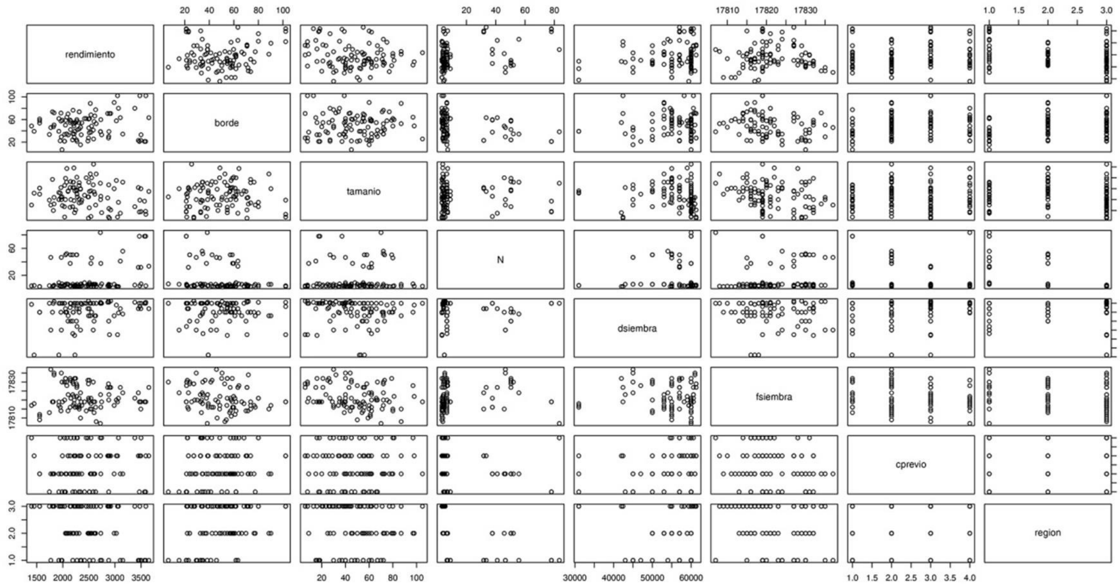

Figura 7. 1. Gráficos de dispersión (y de agrupamiento en el caso de categorías) entre los potenciales predictores del modelo, además del rendimiento, la variable respuesta

```
# figura 7.1
plot(dat.rend2[,c(2:9)])
```

Entre los gráficos de la figura 7. 1 podemos observar que el rendimiento tiende a aumentar con la densidad de borde, la densidad de siembra y la aplicación de fertilizante, y además parece ser máximo en fechas de siembra intermedias. Como se indicó en el capítulo anterior, es conceptualmente interesante evaluar si el efecto del borde varía entre regiones (figura. 7. 2). De la misma forma, podría esperarse que de existir una fecha de siembra óptima difiera entre regiones (figura 7. 3). La función `coplot()` es útil para explorar interacciones entre predictores cuantitativos y factores:

```
# interacciones
coplot(rendimiento ~ borde|region, data=dat.rend2,
       rows=1, panel=panel.smooth,
       ylab="Rendimiento (kg/ha)",
       xlab="Densidad de borde (m/ha)",
       lwd=3, pch=16) # borde:región (figura 7.2)
```

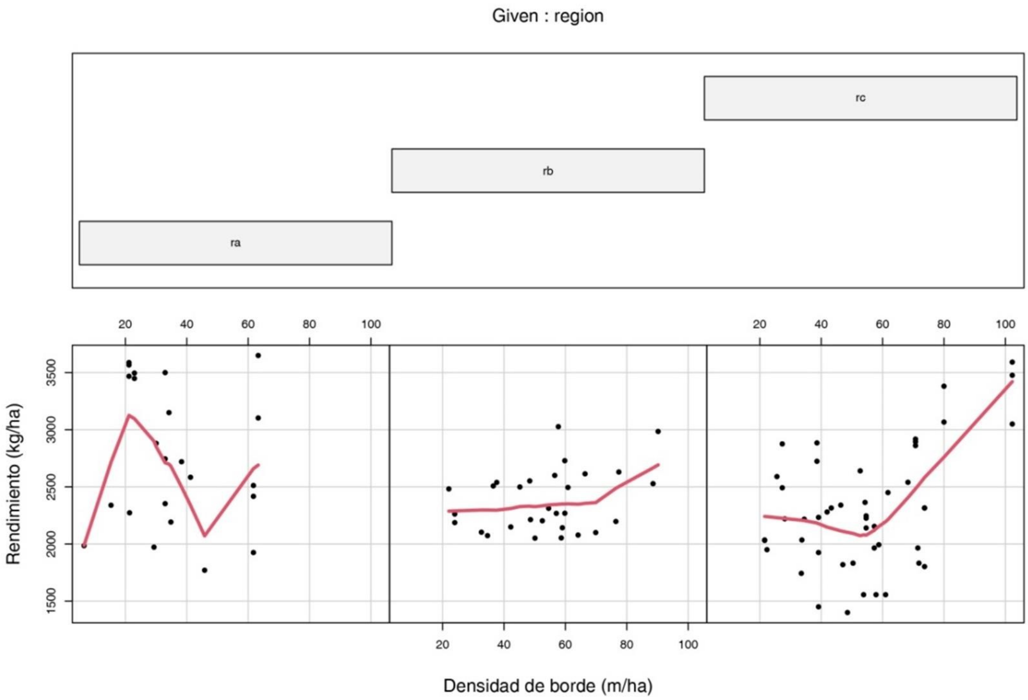

Figura 7. 2. Gráficos de dispersión entre rendimiento y densidad de borde en cada región, incluyendo la tendencia de la relación (líneas rojas) para explorar una potencial interacción

```
coplot(rendimiento ~ fsiembra|region, data=dat.rend2,
       rows=1, panel=panel.smooth,
       ylab="Rendimiento (kg/ha)",
       xlab="Fecha de siembra",
       lwd=3, pch=16) # fs:región (figura 7.3)
```

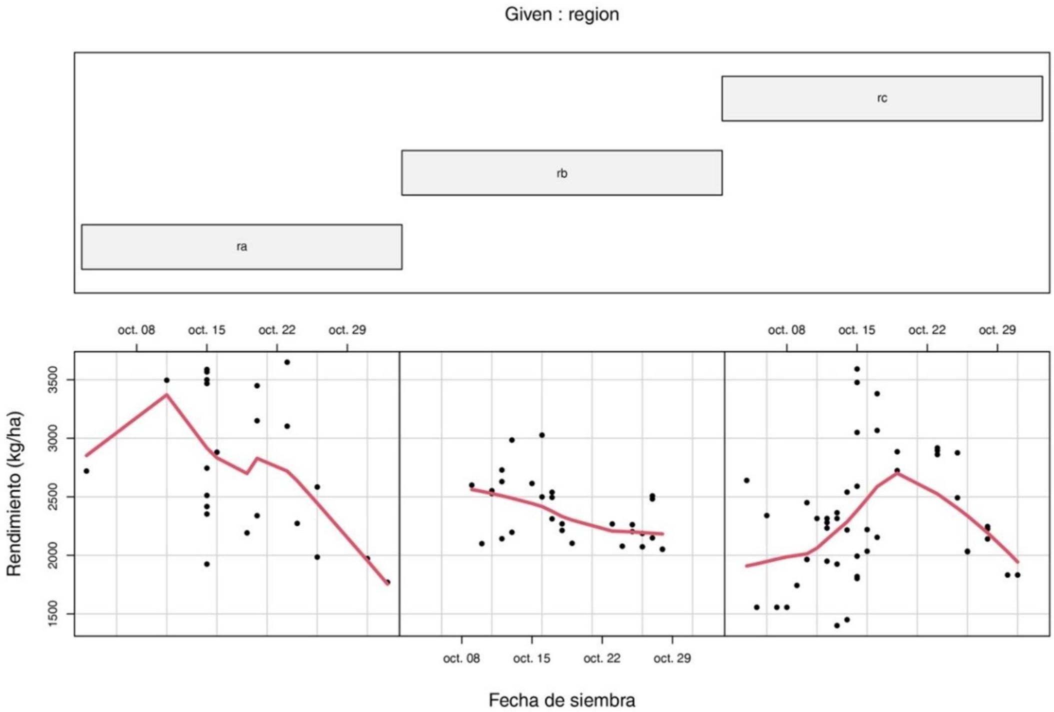

Figura 7. 3. Gráficos de dispersión entre rendimiento y fecha de siembra en cada región, incluyendo la tendencia de la relación (líneas rojas) para explorar una potencial interacción

Como hemos visto en los capítulos anteriores, no parece haber interacción entre región y cultivo previo (figura 7. 4):

```
# figura 7.4
with(dat.rend2, interaction.plot(cprevio, region, rendimiento,
                                 xlab="Cultivo previo",
                                 ylab="Rendimiento (kg/ha)",
                                 col=c("red", "green", "gray50"),
                                 lwd=2, lty=1), ylim=c(500,4500))) # región:cultivo
```

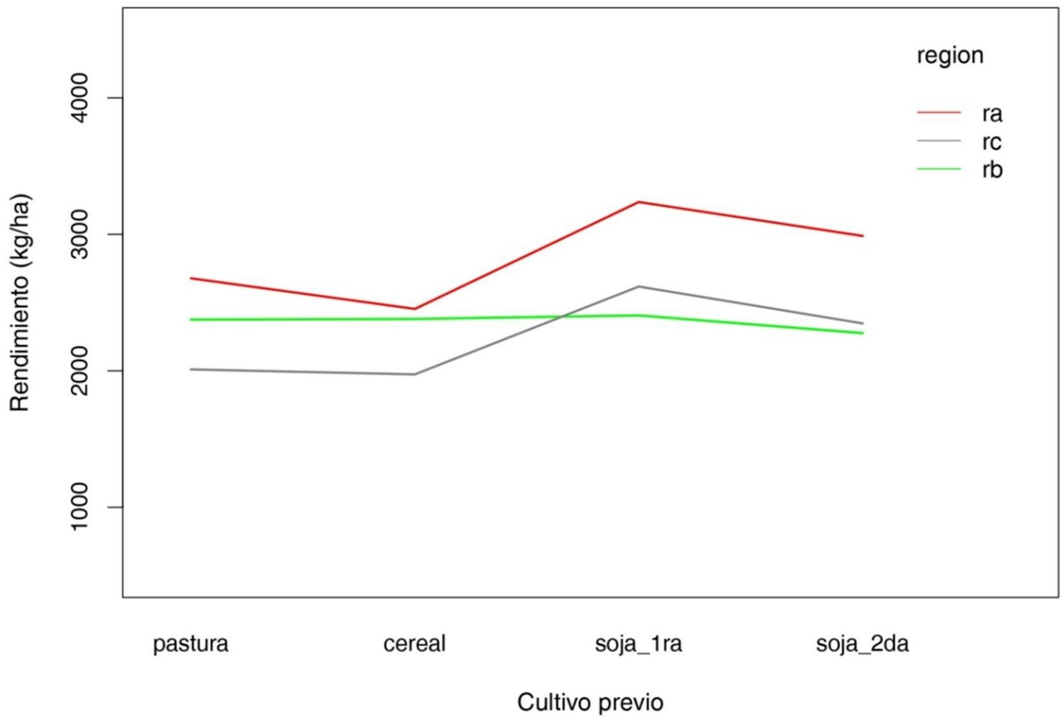

Figura 7. 4. Evaluación gráfica de la interacción entre el cultivo previo y la región

Es importante notar que `coplot()` permite explorar interacciones entre una variable cuantitativa y un factor, mientras que `interaction.plot()` es útil para interacciones entre factores.

Recordemos que los predictores del modelo no deben estar correlacionados. La figura 7. 1 nos permite visualizar las relaciones entre predictores. Otra alternativa más específica para este fin la aporta la función `ggpairs()`:

```
# figura 7.5
library(GGally)

## Loading required package: ggplot2

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2

ggpairs(dat.rend2[,3:7],
        diag = list(continuous="barDiag"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

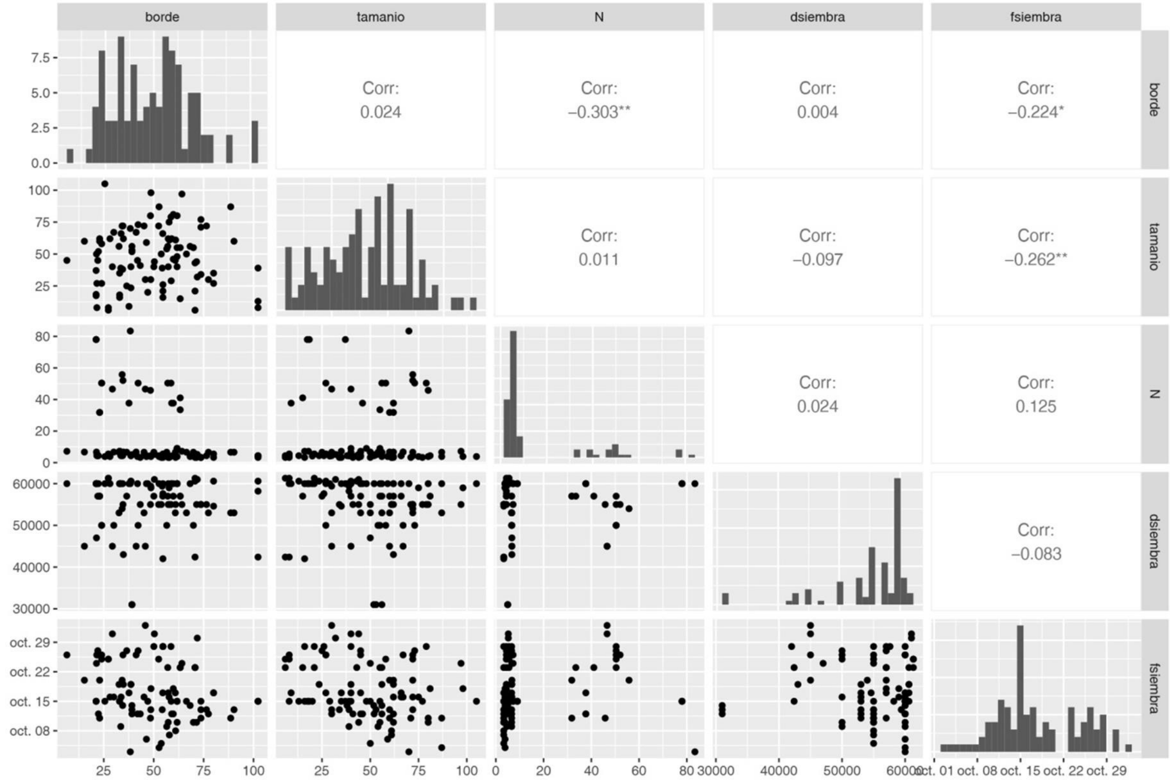

Figura 7. 5. Matriz de correlación y gráficos de dispersión de predictores cuantitativos (en la diagonal se muestran los histogramas de cada variable)

La figura 7. 5 nos indica que los predictores cuantitativos presentan una baja correlación de a pares.

7. 4. Modelo y ajuste

Planteamos un modelo para predecir el rendimiento en función de la densidad de borde y el tamaño del lote, incluyendo además la región, la fertilización nitrogenada, el cultivo previo, la densidad de siembra y la fecha de siembra. Incluimos un término cuadrático para la posible relación no lineal entre densidad de borde y rendimiento. Para esto, generamos una nueva columna con la densidad de borde al cuadrado:

```
dat.rend2$borde2 <- dat.rend2$borde^2 # término cuadrático para la densidad de borde
```

De igual manera procedemos en el caso de la fecha de siembra, ya que si existe un momento óptimo para sembrar y se encuentra dentro del período abarcado por los datos de la muestra, la relación con el rendimiento resultaría ser no lineal. Por defecto, las fechas coercidas a tipo `Date` son almacenadas internamente como el número de días transcurridos desde el 01/01/1970 (a las fechas anteriores a este día se les asigna un valor negativo). Veamos los valores para los primeros cinco lotes:

```
as.numeric(dat.rend2$fsiembra)[0:5] # mostramos la fecha como “días desde” (numérico)

## [1] 17820 17824 17823 17830 17821
```

Estos valores dificultan la interpretación de la variable (tiempo). Lo que hacemos es cambiar el origen e indicar, arbitrariamente, que las fechas se guarden como días transcurridos desde la primavera del año 2018. Para esto, primero definimos este día como el origen, que se guarda como el número de días transcurridos desde el 01/01/1970:

```
desde <- as.Date("2018-09-21") # definimos el origen (primavera 2018)
as.numeric(desde) # días transcurridos desde 1/1/1970 hasta el 21/09/2018

## [1] 17795
```

Y luego restamos esta cantidad a los valores originales. La diferencia obtenida corresponde a los días transcurridos entre el 21/09/2018 y la fecha de siembra:

```
dat.rend2$fs <- as.numeric(dat.rend2$fsiembra - desde)
dat.rend2$fs[0:5] # días desde la primavera de 2018 hasta la fecha de siembra

## [1] 25 29 28 35 26
```

Ahora sí podemos generar una columna con la fecha de siembra elevada al cuadrado:

```
dat.rend2$fs2 <- dat.rend2$fs^2 # cuadrado de fecha de siembra
str(dat.rend2)

## ‘data.frame’:    99 obs. Of  12 variables:
##  $ lote       : int  1 2 3 4 5 6 7 8 9 10 …
##  $ rendimiento: int  2881 2339 2192 2584 2312 3103 2273 3150 1972 2528 …
##  $ borde      : num  30 15.3 34.8 41.2 54.5 …
##  $ tamanio    : num  67 60 62 67 66 15 50 72 40 87 …
##  $ N          : num  6.71 6.71 6.71 6.71 4.4 …
##  $ dsiembra   : int  50000 45000 43000 45000 57000 57000 47000 54000 45000 53000 …
##  $ fsiembra   : Date, format: “2018-10-16” “2018-10-20” …
##  $ cprevio    : Factor w/ 4 levels “pastura”,”cereal”,..: 1 1 1 1 1 2 2 2 2 2 …
##  $ 6egión     : Factor w/ 3 levels “ra”,”rb”,”rc”: 1 1 1 1 2 1 1 1 1 2 …
##  $ borde2     : num  902 235 1213 1699 2975 …
##  $ fs         : num  25 29 28 35 26 32 33 29 40 20 …
##  $ fs2        : num  625 841 784 1225 676 …
```

Finalmente, incluimos las interacciones entre la región con la densidad de borde y con el tamaño de lote. El modelo entonces es el siguiente:

$$r_i = \beta_0 + \beta_1 \times n_i + \beta_2 \times c_i + \beta_3 \times s_i + \beta_4 \times ss_i + \beta_5 \times rb_i + \beta_6 \times rc_i + \beta_7 \times d_i + \beta_8 \times fs_i + \beta_9 \times fs_i^2 + \beta_{10} \times b_i + \beta_{11} \times b_i^2 + \beta_{12} \times t_i \; \beta_{13} \times b_i \times rb_i + \beta_{14} \times b_i \times rc_i + \beta_{15} \times t_i \times rb_i + \beta_{16} \times t_i \times rc_i + \varepsilon_i$$

$$\varepsilon_i \sim \mathcal{N}(0; \sigma^2)_{independientes}$$

donde

r_i = rendimiento de girasol (kg ha^{-1}) del lote i (i = 1, 2, … N),
n_i = cantidad de fertilizante nitrogenado aplicado en el lote i (kg ha^{-1}),
c_i = 1 si el lote tuvo cereal como cultivo previo y 0 en caso contrario,
s_i = 1 si el lote tuvo soja de primera como cultivo previo y 0 en caso contrario,
ss_i = 1 si el lote tuvo soja de segunda como cultivo previo y 0 en caso contrario,
rb_i = 1 si el lote se localiza en la región b y 0 en caso contrario,
rc_i = 1 si el lote se localiza en la región c y 0 en caso contrario,
d_i = densidad de siembra del lote i (n° semillas ha^{-1}),
fs_i = fecha de siembra del lote i (días),
b_i = densidad de borde en el contexto (radio de 1,5 km) del lote i (m ha^{-1}),
t_i = tamaño del lote i (ha),
ε_i = término de error (kg ha^{-1}).

7. 4. 1. Otra forma de escribir el modelo

Nuestro modelo tiene 17 parámetros que predicen el rendimiento promedio (β), más la varianza, es decir, 18 parámetros a estimar. Existen múltiples formas de escribir el mismo modelo, una de ellas es la siguiente:

$$r_i \sim \mathcal{N}(\mu_i; \sigma^2)_{independientes}$$

$$\mu_i = \beta_0 + \beta_1 \times n_i + \beta_2 \times c_i + \beta_3 \times s_i + \beta_4 \times ss_i + \beta_5 \times rb_i + \beta_6 \times rc_i + \beta_7 \times d_i + \beta_8 \times fs_i + \beta_9 \times fs_i^2 + \beta_{10} \times b_i + \beta_{11} \times b_i^2 + \beta_{12} \times t_i \; \beta_{13} \times b_i \times rb_i + \beta_{14} \times b_i \times rc_i + \beta_{15} \times t_i \times rb_i + \beta_{16} \times t_i \times rc_i$$

Mediante esta forma se enfatiza el proceso de generación de datos que se asume desde el modelo, en este caso que el rendimiento de girasol de cada lote (r_i) proviene de una distribución normal cuya media (μ_i) es determinada por una función lineal. En otras palabras, estamos modelando uno de los dos parámetros de la distribución normal mediante un conjunto de (p) parámetros asociados a (p - 1) variables independientes. A diferencia de la media, la varianza no posee subíndice. Esto significa que no es modelada, es decir, que no varía en función de otros predictores, y que la varianza es común a todas las distribuciones normales.

7. 4. 2. Notación matricial

La escritura de los MLG puede generalizarse (modelo lineal general) utilizando notación matricial:

$$\mathbf{Y} = \mathbf{X} \times \mathbf{b} + \mathbf{e}$$

$$\mathbf{e} \sim \mathcal{N}(0;\ \mathbf{I} \times \sigma^2)_{independientes}$$

donde

Y = vector de rendimientos (de dimensión N × 1),
X = matriz de diseño (N × p),
b = vector de parámetros (1 × p),
e = vector de errores (N × 1),
I = matriz identidad, de dimensión N × N.

$$\underset{\mathbf{Y}}{\begin{bmatrix} r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \\ \vdots \\ r_N \end{bmatrix}} = \underset{\mathbf{X}}{\begin{bmatrix} 1 & n_1 & c_1 & s_1 & ss_1 & \cdots & t_1 \times rc_1 \\ 1 & n_2 & c_1 & s_2 & ss_2 & \cdots & t_2 \times rc_2 \\ 1 & n_3 & c_1 & s_3 & ss_3 & \cdots & t_3 \times rc_3 \\ 1 & n_4 & c_1 & s_4 & ss_4 & \cdots & t_4 \times rc_4 \\ 1 & n_5 & c_1 & s_5 & ss_5 & \cdots & t_5 \times rc_5 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & n_N & c_N & s_N & ss_N & \cdots & t_N \times rc_N \end{bmatrix}} \times \underset{\mathbf{b}}{\begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_{16} \end{bmatrix}} + \underset{\mathbf{e}}{\begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \vdots \\ \varepsilon_N \end{bmatrix}}$$

Las matrices constituyen el esqueleto de los MLG. En particular, conocer la estructura de la matriz de diseño es de gran importancia para la comprensión integral del modelo y de su ajuste. Como vimos en el capítulo 2, cada columna de esta matriz se asocia a uno de los predictores del modelo, incluyendo los valores que resultan de la codificación de los factores (si los hubiera) y un vector con valores de 1 para el intercepto. Además, cada fila representa una unidad experimental. Por ejemplo, en la regresión lineal simple del capítulo 1, **X** tiene solo dos columnas. La primera corresponde al intercepto y sus valores son 1, y la segunda al nitrógeno por lo que cada valor de esta columna indica la cantidad de fertilizante aplicado en el lote. En cambio, en el caso del modelo unifactorial con la región como predictor categórico de cuatro niveles, la matriz de diseño presenta cuatro columnas. En la codificación por defecto, la primera de las columnas se vincula a la región base, y las otras tres expresan la información correspondiente al resto de las regiones. Repasando lo visto, se trata de una matriz de ceros y unos en la que cada fila tiene alguna de estas cuatro combinaciones: 1 0 0 0 (para un lote ubicado en la región «a»), 1 1 0 0 (lotes localizados en la región «b»), 1 0 1 0 (región «c») o 1 0 0 1 (región «d»).

7. 4. 3. El método de MCO y lm()

Para el ajuste del modelo, el vector **Y** y la matriz **X** son dados por los datos de la muestra mientras que los vectores **b** y **e** representan las incógnitas. Recordemos que `lm` utiliza mínimos cuadrados ordinarios (MCO), método con el cual se obtienen los valores de los coeficientes (los 17 elementos de **b** en este caso) que hacen que la sumatoria de los residuos cuadráticos (SCE) sea mínima. Dado este criterio de ajuste, expresamos **e** como una función de **b**:

$$\mathbf{e} = \mathbf{Y} - \mathbf{X} \times \mathbf{b}$$

Operando matricialmente, la SCE se obtiene como el producto entre el vector **e** y su transpuesta ($\mathbf{e}^t\mathbf{e}$):

$$\mathrm{SCE} = \mathbf{e}^{\mathrm{t}}\mathbf{e} = [\,\mathrm{e}_1 \quad \mathrm{e}_2 \quad \mathrm{e}_3 \quad \mathrm{e}_4 \quad \mathrm{e}_5 \quad \cdots \quad \mathrm{e}_n] \times \begin{bmatrix} \mathrm{e}_1 \\ \mathrm{e}_2 \\ \mathrm{e}_3 \\ \mathrm{e}_4 \\ \mathrm{e}_5 \\ \vdots \\ \mathrm{e}_n \end{bmatrix}$$

La solución que minimiza $\mathbf{e}^t\mathbf{e}$ se obtiene analíticamente, aplicando derivadas parciales sobre la SCE respecto a cada elemento del vector **b** e igualando a cero:

$$\frac{\partial \mathbf{e}^{\mathrm{t}}\mathbf{e}}{\partial \mathbf{b}} = 0$$

La ecuación anterior expresa un sistema de ecuaciones (17 en este caso) conocido como *ecuaciones normales*. Al resolver el sistema:

$$\hat{\mathbf{b}} = (\mathbf{X}^{\mathrm{t}}\mathbf{X})^{-1} \times \mathbf{X}^{\mathrm{t}}\mathbf{Y}$$

Una vez minimizada la SCE, la estimación de la varianza residual (s^2) se obtiene como:

$$s^2 = \frac{\mathrm{SCE}}{n-p}$$

Y finalmente, los errores estándar de los elementos de $\hat{\mathbf{b}}$:

$$\mathbf{s}^2_{\hat{\mathbf{b}}} = s^2 \times (\mathbf{X}^t\mathbf{X})^{-1}$$

El resultado de la operación del lado derecho de la igualdad es la matriz de varianzas y covarianzas de los coeficientes estimados cuya dimensión es $p \times p$ (17 × 17, en nuestro modelo). Las raíces cuadradas de los valores de la diagonal de esta matriz constituyen los errores estándar de los coeficientes estimados.

Ajustamos el modelo de acuerdo al proceso que hemos descrito, y luego comparamos con el ajuste que obtenemos ejecutando `lm()`. Para realizar el ajuste por mínimos cuadrados ordinarios paso a paso, primero definimos el vector de rendimientos ($\mathbf{Y}$) y la matriz de diseño ($\mathbf{X}$) del modelo:

```
Y  <- dat.rend2$rendimiento # vector de rendimientos

X <- with(dat.rend2,
          model.matrix(~ N + cprevio + region + dsiembra + fs + fs2 +
                         tamanio + borde + borde2 +
                         fs:region + borde:region)) # matriz de diseño

Y[0:5] # primeros cinco valores del vector

## [1] 2881 2339 2192 2584 2312

X[c(0:5), ] # primeras cinco filas de matriz

##   (Intercept)    N cpreviocereal cpreviosoja_1ra cpreviosoja_2da regionrb
## 1           1 6.71             0               0               0        0
## 2           1 6.71             0               0               0        0
## 3           1 6.71             0               0               0        0
## 4           1 6.71             0               0               0        0
## 5           1 4.40             0               0               0        1
##   regionrc dsiembra fs  fs2 tamanio borde    borde2 regionrb:fs regionrc:fs
## 1        0    50000 25  625      67 30.03  901.8009           0           0
## 2        0    45000 29  841      60 15.33  235.0089           0           0
## 3        0    43000 28  784      62 34.83 1213.1289           0           0
## 4        0    45000 35 1225      67 41.22 1699.0884           0           0
## 5        0    57000 26  676      66 54.54 2974.6116          26           0
##   regionrb:borde regionrc:borde
## 1           0.00              0
## 2           0.00              0
## 3           0.00              0
## 4           0.00              0
## 5          54.54              0
```

Lo que hacemos ahora es simplemente ejecutar la solución a las ecuaciones normales ($[\mathbf{X}^t\mathbf{X}]^{-1} \times \mathbf{X}^t\mathbf{Y}$) y obtenemos los coeficientes estimados. Para esto, extraemos la transpuesta de x y realizamos la inversión usando la función `t()` y `solve()`, respectivamente, y multiplicamos:

```
b <- solve(t(X)%*%X)  %*% t(X)%*%Y # estimación puntual de los coef. (elementos de b)
b
```

```
##                          [,1]
## (Intercept)      614.07146627
## N                  9.27515530
## cpreviocereal    -49.33980218
## cpreviosoja_1ra  340.88440775
## cpreviosoja_2da  143.71287777
## regionrb        -408.83654777
## regionrc        -858.57073694
## dsiembra           0.01130215
## fs               138.45883786
## fs2               -2.77287151
## tamanio           -1.00600155
## borde            -20.11680316
## borde2             0.23148516
## regionrb:fs        1.06560851
## regionrc:fs       14.02595087
## regionrb:borde     1.53573677
## regionrc:borde     2.43028403
```

Para estimar la varianza residual necesitamos la SCE, y para esta última requerimos el vector de residuos. Este vector podemos obtenerlo ejecutando la diferencia entre los rendimientos observados (**Y**) y los predichos ($\mathbf{X} \times \hat{\mathbf{b}}$):

```
e <- Y - X%*%b # residuos
sce <- sum(e^2) # suma de cuadrados del error
```

Al dividir la SCE por los grados de libertad, es decir, por la diferencia entre el tamaño muestral (n) y el número de elementos de **b** (p), obtenemos la varianza estimada:

```
n <- length(Y)
p <- 17
s2 <- sce/(n-p) # varianza residual
```

Con la varianza estimada calculamos los errores estándar de los coeficientes estimados. Para eso primero resolvemos la matriz de varianzas y covarianzas de $\hat{\mathbf{b}}$, luego extraemos su diagonal mediante la función `diag()`, y finalmente calculamos la raíz cuadrada de cada elemento de la diagonal:

```
vcov.b <- s2*solve(t(X)%*%X) # matriz de varianzas y covarianzas de los estimadores
ee.b <- sqrt(diag(vcov.b)) # √ elementos de la diagonal (errores est. de los coef.)
```

Dado que tenemos los errores estándar, podemos expresar los coeficientes estimados en la escala t y calcular valores p para una hipótesis nula de $\mathbf{b} = 0$:

```
tobs <- b/ee.b # t observados
vp <- pt(q=abs(tobs), df=n-p, lower.tail=FALSE)*2 # valores p (prueba a dos colas)
```

En cuanto a la componente aleatoria del modelo, la matriz **I** es una matriz identidad (de $n \times n$ en la muestra) por lo que los elementos de su diagonal tienen un valor de 1 y por fuera de ella los elementos valen 0. Multiplicando I por la varianza (un escalar) se obtiene la matriz de varianzas y covarianzas de los residuos (lógicamente, también de dimensión $n \times n$), cuyos valores son σ^2 en la diagonal y 0 por fuera de ella. Si bien esta matriz no forma parte del ajuste por MCO (a diferencia, por ejemplo, del *Método de Máxima Verosimilitud*), es aquí donde se formaliza el supuesto de independencia del modelo; la correlación entre dos errores cualesquiera (ε_i, ε_j) se asume nula desde los ceros por fuera de la diagonal de **I**, con lo cual la covarianza de los errores también es cero. Veamos cómo podemos generar **I** y obtener la matriz de varianzas y covarianzas que asumimos para los residuos:

```
I <- diag(n) # matriz identidad (n x n)
mvc <- I*s2 # matriz de varianzas y covarianzas (n x n)
```

Como podemos ver para los cinco primeros lotes, asumimos que los residuos tienen una covarianza/correlación de cero y provienen de una distribución normal multivariada cuya varianza estimada es de 130883,8 kg^2 ha^{-2}:

```
I[c(1:5), c(1:5)]

##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    0    0    0    0
## [2,]    0    1    0    0    0
## [3,]    0    0    1    0    0
## [4,]    0    0    0    1    0
## [5,]    0    0    0    0    1

mvc[c(1:5), c(1:5)]

##          [,1]     [,2]     [,3]     [,4]     [,5]
## [1,] 130883.8      0.0      0.0      0.0      0.0
## [2,]      0.0 130883.8      0.0      0.0      0.0
## [3,]      0.0      0.0 130883.8      0.0      0.0
## [4,]      0.0      0.0      0.0 130883.8      0.0
## [5,]      0.0      0.0      0.0      0.0 130883.8
```

Ahora ajustamos el modelo usando `lm()`:

```
mlg <- lm(rendimiento ~ N + cprevio + region + dsiembra + fs + fs2 +
                        tamanio + borde + borde2 +
                        fs:region + borde:region, data=dat.rend2) # ajuste del modelo
```

Para comparar ambos ajustes, generaremos una tabla de coeficientes similar a la que devuelve el `summary` de un objeto `lm`:

```
resumen <- data.frame(b, ee.b, tobs, vp)
names(resumen) <- c("β est.", "error est.", "t", "valor-p")
round(resumen, 3) # MCO paso a paso

##                   β est. error est.     t valor-p
## (Intercept)      614.071    896.086 0.685   0.495
```

```
## N                  9.275      2.545  3.645    0.000
## cpreviocereal    -49.340    126.645 -0.390    0.698
## cpreviosoja_1ra  340.884    123.899  2.751    0.007
## cpreviosoja_2da  143.713    135.929  1.057    0.293
## regionrb        -408.837    756.743 -0.540    0.590
## regionrc        -858.571    535.203 -1.604    0.113
## dsiembra           0.011      0.006  1.872    0.065
## fs               138.459     46.943  2.949    0.004
## fs2               -2.773      0.797 -3.479    0.001
## tamanio           -1.006      2.055 -0.489    0.626
## borde            -20.117      8.358 -2.407    0.018
## borde2             0.231      0.087  2.666    0.009
## regionrb:fs        1.066     19.690  0.054    0.957
## regionrc:fs       14.026     15.925  0.881    0.381
## regionrb:borde     1.536      7.418  0.207    0.837
## regionrc:borde     2.430      6.305  0.385    0.701
```

Vemos que los resultados son idénticos a los obtenidos con `lm()`:

```
round(summary(mlg)$coef, 3) # lm()
```

```
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)       614.071    896.086   0.685    0.495
## N                   9.275      2.545   3.645    0.000
## cpreviocereal     -49.340    126.645  -0.390    0.698
## cpreviosoja_1ra   340.884    123.899   2.751    0.007
## cpreviosoja_2da   143.713    135.929   1.057    0.293
## regionrb         -408.837    756.743  -0.540    0.590
## regionrc         -858.571    535.203  -1.604    0.113
## dsiembra            0.011      0.006   1.872    0.065
## fs                138.459     46.943   2.949    0.004
## fs2                -2.773      0.797  -3.479    0.001
## tamanio            -1.006      2.055  -0.489    0.626
## borde             -20.117      8.358  -2.407    0.018
## borde2              0.231      0.087   2.666    0.009
## regionrb:fs         1.066     19.690   0.054    0.957
## regionrc:fs        14.026     15.925   0.881    0.381
## regionrb:borde      1.536      7.418   0.207    0.837
## regionrc:borde      2.430      6.305   0.385    0.701
```

También podemos comparar los intervalos de confianza al 95 % de los estimadores obtenidos paso a paso con aquellos que obtenemos desde el modelo ajustado con la función `lm()`. Para esto necesitamos calcular los intervalos manualmente usando los errores estándar de **b** y el cuantil que acumula el 97,5 % en una distribución t con $n - p$ grados de libertad:

```
t <- qt(p=0.975, df=n-p) # cuantil t que acumula 97,5%
ic.sup <- b + t * ee.b # límite superior IC 95%
ic.inf <- b - t * ee.b # límite inferior IC 95%
```

Al igual que en la comparación anterior, generamos una tabla, en este caso incluyendo el límite inferior y el límite superior de los intervalos de confianza:

```r
int.conf.b <- data.frame(ic.inf, ic.sup)
names(int.conf.b) <- c("2,5 %", "97,5 %")
round(int.conf.b, 3) # IC MCO paso a paso
```

```
##                      2,5 %   97,5 %
## (Intercept)      -1168.530 2396.673
## N                    4.213   14.337
## cpreviocereal     -301.278  202.598
## cpreviosoja_1ra     94.410  587.358
## cpreviosoja_2da   -126.693  414.118
## regionrb         -1914.239 1096.566
## regionrc         -1923.259  206.118
## dsiembra            -0.001    0.023
## fs                  45.074  231.844
## fs2                 -4.359   -1.187
## tamanio             -5.095    3.083
## borde              -36.743   -3.491
## borde2               0.059    0.404
## regionrb:fs        -38.104   40.235
## regionrc:fs        -17.655   45.706
## regionrb:borde     -13.222   16.293
## regionrc:borde     -10.112   14.973
```

Los valores resultan idénticos a los informados al ejecutar la función `confint()`:

```r
round(confint(mlg), 3) # IC lm()
```

```
##                      2.5 %   97.5 %
## (Intercept)      -1168.530 2396.673
## N                    4.213   14.337
## cpreviocereal     -301.278  202.598
## cpreviosoja_1ra     94.410  587.358
## cpreviosoja_2da   -126.693  414.118
## regionrb         -1914.239 1096.566
## regionrc         -1923.259  206.118
## dsiembra            -0.001    0.023
## fs                  45.074  231.844
## fs2                 -4.359   -1.187
## tamanio             -5.095    3.083
## borde              -36.743   -3.491
## borde2               0.059    0.404
## regionrb:fs        -38.104   40.235
## regionrc:fs        -17.655   45.706
## regionrb:borde     -13.222   16.293
## regionrc:borde     -10.112   14.973
```

Las varianzas residuales también coinciden:

```r
s2 # varianza residual MCO paso a paso
```

```
## [1] 130883.8
```

```r
summary(mlg)$sigma^2 # varianza residual por lm()
```

```
## [1] 130883.8
```

Por último, encontramos que el MCO paso a paso predice los mismos rendimientos que la función `predict()`. Comparemos en los primeros cinco lotes:

```
rend.pred <- X%*%b
as.vector(rend.pred[0:5]) # MCO paso a paso

## [1] 2507.086 2553.876 2382.998 2130.893 2252.214

predict(mlg)[0:5] # lm()

##        1        2        3        4        5
## 2507.086 2553.876 2382.998 2130.893 2252.214
```

La secuencia de códigos que hemos mostrado para ajustar el modelo por MCO son, en efecto, similares a aquellos que estamos implementando cada vez que ejecutamos la función `lm()`.

7. 5. ANOVA

Realizamos un ANOVA secuencial para evaluar la significancia de los predictores. Antes de esto, revisamos si existen problemas de multicolinealidad que pudieran invalidar las conclusiones:

```
round(cor(dat.rend2[ ,c(3:6,11)]), 2) # correlaciones de a pares

##          borde tamanio     N dsiembra    fs
## borde     1.00    0.02 -0.30     0.00 -0.22
## tamanio   0.02    1.00  0.01    -0.10 -0.26
## N        -0.30    0.01  1.00     0.02  0.12
## dsiembra  0.00   -0.10  0.02     1.00 -0.08
## fs       -0.22   -0.26  0.12    -0.08  1.00

library(car)

## Loading required package: carData

round(vif(mlg), 1) # VIF

## there are higher-order terms (interactions) in this model
## consider setting type = 'predictor'; see ?vif

##                GVIF Df GVIF^(1/(2*Df))
## N               2.0  1             1.4
## cprevio         2.3  3             1.1
## region       2359.0  2             7.0
## dsiembra        1.2  1             1.1
## fs             74.8  1             8.7
## fs2            66.4  1             8.1
## tamanio         1.6  1             1.3
## borde          21.9  1             4.7
## borde2         28.6  1             5.3
## region:fs     844.8  2             5.4
## region:borde  356.1  2             4.3
```

Observamos una baja correlación de a pares entre los predictores cuantitativos, lo cual combinado con valores de VIF por debajo de 10 indican que no habría graves problemas de multicolinealidad. Veamos lo que muestran el ANOVA secuencial y el marginal:

```r
anova(mlg) # ANOVA secuencial
```

```
## Analysis of Variance Table
##
## Response: rendimiento
##               Df   Sum Sq Mean Sq F value    Pr(>F)    
## N              1  2449894 2449894 18.7181 4.244e-05 ***
## cprevio        3  5124215 1708072 13.0503 4.764e-07 ***
## region         2  1211994  605997  4.6300 0.0124410 *  
## dsiembra       1   738400  738400  5.6416 0.0198726 *  
## fs             1   322341  322341  2.4628 0.1204219    
## fs2            1  2796220 2796220 21.3641 1.397e-05 ***
## tamanio        1   317829  317829  2.4283 0.1230111    
## borde          1  1583209 1583209 12.0963 0.0008112 ***
## borde2         1  1432678 1432678 10.9462 0.0013946 ** 
## region:fs      2   150501   75251  0.5749 0.5649897    
## region:borde   2    19742    9871  0.0754 0.9274187    
## Residuals     82 10732472  130884                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
Anova(mlg, type="III") # ANOVA marginal (paquete car)
```

```
## Anova Table (Type III tests)
##
## Response: rendimiento
##                 Sum Sq Df F value    Pr(>F)    
## (Intercept)      61464  1  0.4696 0.4950990    
## N              1738777  1 13.2849 0.0004681 ***
## cprevio        2064002  3  5.2566 0.0022874 ** 
## region          344641  2  1.3166 0.2736535    
## dsiembra        458493  1  3.5031 0.0648216 .  
## fs             1138620  1  8.6995 0.0041467 ** 
## fs2            1583878  1 12.1014 0.0008093 ***
## tamanio          31354  1  0.2396 0.6258363    
## borde           758294  1  5.7936 0.0183305 *  
## borde2          930211  1  7.1072 0.0092458 ** 
## region:fs       135499  2  0.5176 0.5978643    
## region:borde     19742  2  0.0754 0.9274187    
## Residuals     10732472 82                      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Los resultados sugieren que la densidad de borde afecta al rendimiento en forma no lineal (el término cuadrático es significativo) y que el efecto es independiente de la región (la interacción no es significativa). Por otro lado, no encontramos evidencia de un efecto del tamaño de lote. Sin embargo, el hecho de que su valor p en el ANOVA secuencial sea relativamente bajo (p = 0,12) y que este aumente en gran medida cuando el análisis es de tipo marginal (p = 0,62) sugiere profundizar el análisis de multicolinealidad (si bien su VIF es cercano a 1, recordemos que la multicolinealidad se trata de un problema de grado y que en el capítulo 4 encontramos una significancia marginal para el efecto del tamaño del lote). Por ejemplo, ajustar el modelo con los predictores estandarizados, es decir, restándoles su promedio y dividiéndolos por su desvío estándar, es una forma de reducir potenciales problemas de multicolinealidad. Como esperábamos, el resto de los predictores tienen un efecto en el rendimiento, que es no lineal en el caso de la fecha de siembra (geométricamente, el término lineal positivo y el cuadrático negativo resultan en una parábola con un máximo indicando la existencia de un momento óptimo para la siembra).

7. 6. Comparaciones múltiples *a posteriori*

Dado que los efectos principales de la región y del cultivo previo son importantes, nos podemos preguntar cuáles son sus diferencias. Como vimos en el capítulo 3, esto requiere realizar pruebas *a posteriori*. En este caso, como además de factores hemos incluido predictores cuantitativos e interacciones entre ambos (`fs:region` y `borde:region`), nos conviene utilizar el paquete `emmeans`, mucho más flexible que lo que ofrece `agricolae`. Primero veamos cómo aplicarlo para evaluar diferencias de medias entre regiones y entre cultivos previos:

```
library(emmeans) # recordar instalar el paquete

# entre regiones
emm.reg <- emmeans(mlg, specs = pairwise ~ region)

## NOTE: Results may be misleading due to involvement in interactions

emm.reg

## $emmeans
##  region emmean    SE df lower.CL upper.CL
##  ra       2703 111.9 82     2480     2926
##  rb       2399  83.8 82     2232     2565
##  rc       2337  69.1 82     2200     2474
##
## Results are averaged over the levels of: cprevio
## Confidence level used: 0.95
##
## $contrasts
##  contrast estimate  SE df t.ratio p.value
##  ra - rb     304.4 137 82   2.223  0.0732
##  ra - rc     365.9 141 82   2.591  0.0302
##  rb - rc      61.5 111 82   0.553  0.8450
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Lo que hemos hecho es usar la función `emmeans()` para realizar comparaciones de a pares entre las tres regiones. Como indica la salida, se utilizó la corrección de Tukey para ajustar el valor *p*, la cual es la opción por defecto. El resultado se guarda como una lista de dos componentes. Uno de ellos contiene las medias estimadas para cada región:

```
emm.reg$emmeans # medias

##  region emmean    SE df lower.CL upper.CL
##  ra       2703 111.9 82     2480     2926
##  rb       2399  83.8 82     2232     2565
##  rc       2337  69.1 82     2200     2474
##
## Results are averaged over the levels of: cprevio
## Confidence level used: 0.95
```

Al ser una lista, también podemos acceder usando la doble indexación:

```r
emm.reg[[1]]
```

```
##  region emmean    SE df lower.CL upper.CL
##  ra       2703 111.9 82     2480     2926
##  rb       2399  83.8 82     2232     2565
##  rc       2337  69.1 82     2200     2474
##
## Results are averaged over the levels of: cprevio
## Confidence level used: 0.95
```

El segundo componente contiene la información con los contrastes:

```r
emm.reg$contrasts # diferencia de medias
```

```
##  contrast estimate  SE df t.ratio p.value
##  ra - rb     304.4 137 82   2.223  0.0732
##  ra - rc     365.9 141 82   2.591  0.0302
##  rb - rc      61.5 111 82   0.553  0.8450
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Utilizando la doble indexación:

```r
emm.reg[[2]]
```

```
##  contrast estimate  SE df t.ratio p.value
##  ra - rb     304.4 137 82   2.223  0.0732
##  ra - rc     365.9 141 82   2.591  0.0302
##  rb - rc      61.5 111 82   0.553  0.8450
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

También podemos extraer los contrastes desde las funciones `contrasts()` o `pairs()` aplicadas sobre el objeto `emm.reg` en el que guardamos los resultados de `emmeans()`:

```r
contrast(emm.reg, method = "pairwise")
```

```
## I bet you wanted to call this with just object[[1]] - use '[[]]' or which' if I'm wro
ng.
## See '? emm_list' for more information

##  contrast estimate  SE df t.ratio p.value
##  ra - rb     304.4 137 82   2.223  0.0732
##  ra - rc     365.9 141 82   2.591  0.0302
##  rb - rc      61.5 111 82   0.553  0.8450
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

```r
pairs(emm.reg)
```

```
## I bet you wanted to call this with just object[[1]] - use '[[]]' or which' if I'm wro
ng.
## See '? emm_list' for more information
```

```
##  contrast estimate  SE df t.ratio p.value
##  ra - rb     304.4 137 82   2.223  0.0732
##  ra - rc     365.9 141 82   2.591  0.0302
##  rb - rc      61.5 111 82   0.553  0.8450
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Los resultados indican que los lotes de la región «a» tienen rendimientos promedios más altos que los de las regiones «b» y «c». Ahora comparamos entre cultivos previos:

```
# entre cultivos previos
emm.cp <- emmeans(mlg, specs = pairwise ~ cprevio)
emm.cp

## $emmeans
##  cprevio  emmean    SE df lower.CL upper.CL
##  pastura    2371  95.2 82     2181     2560
##  cereal     2321  79.1 82     2164     2479
##  soja_1ra   2712  80.1 82     2552     2871
##  soja_2da   2514 103.9 82     2308     2721
##
## Results are averaged over the levels of: region
## Confidence level used: 0.95
##
## $contrasts
##  contrast            estimate  SE df t.ratio p.value
##  pastura - cereal        49.3 127 82   0.390  0.9798
##  pastura - soja_1ra    -340.9 124 82  -2.751  0.0360
##  pastura - soja_2da    -143.7 136 82  -1.057  0.7163
##  cereal - soja_1ra     -390.2 105 82  -3.712  0.0021
##  cereal - soja_2da     -193.1 124 82  -1.554  0.4103
##  soja_1ra - soja_2da    197.2 121 82   1.626  0.3701
##
## Results are averaged over the levels of: region
## P value adjustment: tukey method for comparing a family of 4 estimates
```

Estos resultados indican que los lotes que estaban previamente sembrados con soja de primera en promedio rindieron más que aquellos en los que fueron precedidos por un cereal o una pastura. En el caso de que hubiéramos detectado interacción entre la región (un factor) y la densidad de borde o la fecha de siembra (predictores cuantitativos) nos hubiera interesado conocer qué pendientes son las que difieren. Para comparar pendientes podemos usar la función `emtrends()`:

```
# entre pendientes (borde x region)
emm.rxb <- emtrends(mlg, pairwise ~ region, var="borde")
emm.rxb

## $emtrends
##  region borde.trend    SE df lower.CL upper.CL
##  ra           -20.1  8.36 82    -36.7    -3.49
##  rb           -18.6 10.99 82    -40.4     3.27
##  rc           -17.7 10.59 82    -38.7     3.37
##
## Results are averaged over the levels of: cprevio
## Confidence level used: 0.95
##
## $contrasts
```

```
##  contrast estimate   SE df t.ratio p.value
##  ra - rb    -1.536 7.42 82  -0.207  0.9767
##  ra - rc    -2.430 6.30 82  -0.385  0.9214
##  rb - rc    -0.895 5.97 82  -0.150  0.9877
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

Al igual que con la diferencia de medias, la salida informa las pendientes para cada región y las diferencias entre pendientes. Como el efecto de la densidad de borde no depende de la región (en el ANOVA no detectamos interacción), encontramos que no hay diferencias entre pendientes. Finalmente, también podremos preguntarnos por diferencias condicionales a ciertos valores del predictor cuantitativo. Por ejemplo, vemos cómo evaluar diferencias de rendimiento entre regiones para una densidad de borde de 50 m/ha:

```
contrast(emmeans(mlg, ~ region*borde,
                 at=list(borde = 50)),
         "pairwise") # entre regiones condicional a la densidad de borde

##  contrast                estimate  SE df t.ratio p.value
##  ra borde50 - rb borde50    303.8 137 82   2.209  0.0756
##  ra borde50 - rc borde50    364.9 142 82   2.566  0.0321
##  rb borde50 - rc borde50     61.1 111 82   0.553  0.8453
##
## Results are averaged over the levels of: cprevio
## P value adjustment: tukey method for comparing a family of 3 estimates
```

7. 7. Selección de variables

Habiendo planteado un modelo que predice el rendimiento medio a partir de una función lineal de 17 parámetros, podemos preguntarnos si vale la pena mantener todos sus términos. Recordemos que los modelos estadísticos tienen dos grandes utilidades: explicar (comprender el sistema bajo estudio) y predecir (aplicar el modelo bajo situaciones no evaluadas, condiciones futuras, etcétera) la variable respuesta en la población. Por un lado, en el capítulo 4 hemos visto que si incluimos más parámetros obtenemos mayor (o al menos igual) bondad de ajuste. Por otro lado, también vimos que una cantidad excesiva de parámetros resulta en un modelo que se ajusta muy bien a los datos de la muestra, pero presenta un pobre desempeño en la población. En este sentido, el *principio de parsimonia* indica que los modelos no deben ser innecesariamente complejos y su adopción implica asumir un compromiso entre el número de parámetros y el ajuste a los datos. Lo que precisamente buscamos con la selección de variables es optimizar este compromiso.

Ahora bien, ¿cómo seleccionar la mejor combinación de predictores? Para ello podemos elegir entre diversas estrategias, todas ellas englobadas en lo que se conoce como el proceso de selección de variables (a menudo se utiliza el término selección de modelos, aunque éste involucra un concepto más amplio). A continuación, abordamos la selección por criterios de información asociada a la inferencia multimodelo, y discutimos sus diferencias respecto a la clásica selección por contraste de hipótesis del enfoque frecuentista. Para esto, primero necesitamos conocer de qué se tratan los criterios de información.

7. 7. 1. Criterios de información

Los criterios de información (CI) son métricas de ajuste de modelos que derivan de la teoría de la información. Bajo este marco teórico, la divergencia entre la realidad y un modelo que la describe se cuantifica en términos de pérdida de información. Si bien no es posible conocer la realidad en su totalidad (es por eso que tomamos una muestra), y por ende tampoco la divergencia, los CI proveen una medida de distancia relativa para inferir cuál de los modelos se le aproxima más (pierde menos información). De esta manera, los modelos pueden ser ordenados de acuerdo a esta medida de distancia relativa y en función de ella seleccionar el o los mejores: cuanto menor es el CI, más cerca está el modelo de la realidad. Los CI incluyen un componente de bondad de ajuste y otro que penaliza la complejidad del modelo, de manera que se ajustan al principio de parsimonia.

Uno de los CI más utilizados es el AIC (*Akaike Information Criterion*). Este se define como:

$$\text{AIC} = -2\ln(v) + 2p$$

donde

v = verosimilitud del modelo,
p = número de parámetros del modelo.

La verosimilitud es la probabilidad (conjunta) de observar los datos bajo el modelo ajustado (en los modelos lineales generales, bajo una distribución normal multivariada con medias = $\hat{\mu}_i$ y varianza = s^2). Si comparamos dos modelos, la verosimilitud será mayor en aquel que mejor describe los datos. La cantidad -2 ln(v) es el componente de bondad de ajuste. Es una medida de variabilidad no explicada basada en probabilidades que se denomina *devianza* (a diferencia de la varianza, que se basa en desvíos). El término 2p expresa la penalización por la complejidad del modelo. Que el peso de la bondad de ajuste iguale al de la penalidad por la complejidad (2 para ambos términos) no es arbitrario, sino que deriva de cómo se cuantifica la divergencia entre modelo y realidad. Otros criterios de información (AICc, QAIC, BIC, DIC, etcétera) ponderan de manera diferente la bondad de ajuste y complejidad, pero todos se basan en el compromiso entre la devianza y el número de parámetros.

7. 7. 2. Selección de variables por criterios de información

La selección de variables por AIC (u otro CI) requiere ajustar el modelo completo y todos los modelos que se anidan dentro de este, incluso el modelo nulo (aquel que incluye solo el intercepto). Ordenando los modelos por AIC podemos determinar cuál tiene el valor más bajo y seleccionarlo como el mejor en términos de parsimonia. Para esto, en R podemos usar la función `dredge()` del paquete `MuMIn`. Dado que la comparación por AIC requiere que los modelos sean ajustados con el mismo número de observaciones, para utilizar esta función los modelos no deben estimarse si hay valores ausentes. Esto lo aseguramos volviendo a ajustar el modelo completo e indicándolo desde el argumento `na.action=na.fail`:

```
m.global <- update(mlg, na.action=na.fail) # valores ausentes
```

Una vez realizado este paso, ejecutamos la función `dredge()` sobre el modelo completo. En este paso, podemos pedir que se calculen medidas de bondad de ajuste extras, tal como el coeficiente de determinación:

```
library(MuMIn) # instalar antes el pquete
modelos <- dredge(m.global, extra=c("R^2")) # ajuste de todos los modelos posibles

## Fixed term is "(Intercept)"

nrow(modelos) # 832 modelos ajustados

## [1] 832
```

Como vemos, hemos ajustado 832 modelos; el modelo completo y 831 modelos anidados dentro de este. La función `model.sel()` permite ordenar los modelos según su AIC y provee una salida amena para evaluar los modelos (la exploraremos luego).

```
rank.aic <- model.sel(modelos, rank="AIC") # modelos ordenados según AIC

## New rank 'AIC' applied to logLik objects
```

Podemos extraer el modelo de menor AIC ejecutando `get.models()` sobre el objeto que contiene los modelos ordenados. Para esto, indicamos que queremos aquel modelo cuya diferencia en AIC es igual a 0 (este concepto lo desarrollamos a continuación):

```
mejor.aic <- get.models(rank.aic, subset = delta <= 0) # modelo de menor AIC
mejor.aic[1] # mejor modelo (según AIC)

## $`256`
##
## Call:
## lm(formula = rendimiento ~ borde + borde2 + cprevio + dsiembra +
##     fs + fs2 + N + region + 1, data = dat.rend2, na.action = na.fail)
##
## Coefficients:
##     (Intercept)            borde           borde2    cpreviocereal
##       111.91232        -19.94675          0.24983        -76.01581
## cpreviosoja_1ra  cpreviosoja_2da         dsiembra               fs
##       353.93753        128.03415          0.01275        154.32962
##             fs2                N         regionrb         regionrc
##        -2.90624          9.83777       -331.59737       -369.46775
```

La salida nos muestra que el mejor modelo según el AIC contiene la densidad de borde (incluido el término cuadrático), el cultivo previo, la densidad de siembra, la fecha de siembra (incluido el término cuadrático), la fertilización nitrogenada y la región. Ahora bien, no debemos perder de vista que estamos trabajando con una muestra y que los CI, en este caso el AIC, cuantifican el grado de sustento estadístico de un grupo de modelos en base a su ajuste a los datos observados. Por lo tanto, el menor AIC de un determinado modelo puede ser producto del azar. Es así que para concluir que un modelo m_1 es superior a otro m_2 deberíamos observar una diferencia sustancial en el grado de apoyo que los datos le otorgan a cada uno de los modelos. La diferencia de AIC entre modelos (ΔAIC):

$$\Delta \text{AIC} = m_2 - m_1$$

expresa cuánto empeora m_2 la descripción de los datos respecto de la que se obtiene con m_1. En términos generales, si la diferencia entre m_2 y m_1 es positiva, m_2 empeora el ajuste y si es negativa lo mejora. Los expertos aún debaten el valor de ΔAIC a partir del cual dos modelos pueden considerarse diferentes, habiendo consenso en que con diferencias menores a 2 el sustento es similar y que un valor mayor a 10 no deja dudas. En la práctica, los modelos se comparan contra el modelo de menor AIC, el cual tiene un ΔAIC = 0 con sí mismo. Veamos los mejores cinco modelos:

```
rank.aic[c(1:5), ] # mejores cinco modelos (según AIC)

## Global model call: lm(formula = rendimiento ~ N + cprevio + region + dsiembra +
##     fs + fs2 + tamanio + borde + borde2 + fs:region + borde:region,
##     data = dat.rend2, na.action = na.fail)
## ---
## Model selection table
##       (Int)    brd    br2 cpr     dsm    fs    fs2      N rgn    tmn brd:rgn
## 256  111.90 -19.95 0.2498   + 0.01275 154.3 -2.906  9.838   +
## 512  271.40 -19.39 0.2431   + 0.01235 149.9 -2.854  9.530   + -1.429
## 1280 521.30 -20.66 0.2537   + 0.01125 139.8 -2.785  9.598   +
## 248  738.70 -19.40 0.2451   +         157.8 -2.981 10.510   +
## 768   97.28 -20.12 0.2342   + 0.01290 156.3 -2.930  9.925   +              +
##      fs:rgn    R^2 df   logLik    AIC delta weight
## 256         0.5918 13 -715.457 1456.9  0.00  0.466
## 512         0.5944 14 -715.140 1458.3  1.37  0.235
## 1280      + 0.5988 15 -714.602 1459.2  2.29  0.148
## 248         0.5683 12 -718.225 1460.5  3.54  0.079
## 768         0.5928 15 -715.338 1460.7  3.76  0.071
## Models ranked by AIC(x)
```

En la tabla cada fila representa un modelo cuyo identificador se informa en la primera columna. Las siguientes columnas se asocian a cada uno de los predictores del modelo completo. Si el predictor está incluido en el modelo que corresponde a la fila, se lo indica con su coeficiente estimado en caso de ser continuo o simplemente con un `+` en caso de ser un factor. Si el predictor no es parte del modelo, la celda queda vacía. La siguiente columna informa el coeficiente de determinación del modelo, dado que lo pedimos al ejecutar función `dredge()`. En la columna `logLik` se informa el logaritmo de la verosimilitud del modelo, de manera que si multiplicamos esta cantidad por -2 obtenemos su devianza, y en `df` se indica el número de parámetros. Podemos calcular el AIC en base a ambas cantidades:

```
aic <- -2*rank.aic$logLik[1:5] + 2*rank.aic$df[1:5]
round(aic, 1)

## [1] 1456.9 1458.3 1459.2 1460.5 1460.7
```

Los valores son los mismos que los informados en la columna `AIC` de la tabla de salida. Si a cada AIC le restamos el AIC del mejor modelo, obtenemos el ΔAIC que es informado como `delta` en la tabla de salida:

```
aic1 <- rank.aic$AIC[1] # AIC del mejor modelo
delta <- aic - aic1 # ΔAIC de los 5 mejores modelos
round(delta, 2)

## [1] 0.00 1.37 2.29 3.54 3.76
```

Observamos que los primeros cinco modelos se encuentran separados por ≈3,8 unidades de AIC y que el modelo 512, el de segundo menor AIC, tiene un ΔAIC < 2 (1,37).

La función `get.models()` nos permite seleccionar los modelos de acuerdo al ΔAIC que definamos como umbral, los cuales serían los modelos de confianza para realizar inferencias. En efecto, en este enfoque la comparación simultánea de un conjunto de modelos permite realizar inferencias basadas en todos ellos (inferencia multimodelo). A cada modelo se le asigna un peso w de evidencia relativa al resto de los modelos, el cual se conoce como *peso de Akaike* (*Akaike weight*). Al estar en relación al resto de los modelos, la suma del peso de todos los modelos resulta igual a 1 ($\sum w = 1$). En base a ello, el w de un modelo se interpreta como la probabilidad de que el modelo provea la mejor aproximación a la realidad (el que pierde menos información) de entre todos los modelos que se están comparando. Los w son informados en la columna `weight` de la tabla de salida. El cociente entre los pesos de dos modelos m_1 y m_2 expresa la tasa de evidencia que uno tiene respecto al otro:

$$\text{tasa de evidencia} = \frac{w_{m_1}}{w_{m_2}}$$

De esta forma, el modelo 256 con el mínimo AIC ($w_{256} = 0{,}466$) tiene dos veces más evidencia que el modelo 512 ($w_{512} = 0{,}235$). Sumando los w de los modelos que incluyen a un predictor j, obtenemos la importancia relativa de ese predictor (ir_j). Para esto podemos usar la función `sw()`:

```
ir <- sw(modelos) # importancia relativa de los predictores
ir

##                       N    cprevio fs2  fs   borde2 region borde dsiembra
## Sum of weights:       1.00 1.00    0.99 0.99 0.99   0.97   0.88  0.77
## N containing models:  416  416     416  512  416    576    512   416
##                       tamanio fs:region borde:region
## Sum of weights:       0.27    0.14      0.06
## N containing models:  416     192       192
```

Ahora bien, el modelo de mejor ranking es resultado de observar datos muestrales (llamemos a este modelo $mm_{muestra}$) y puede, o no, ser el que más se aproxima a la realidad de entre los modelos comparados (denominemos a este modelo mm_{real}). En el marco de la inferencia multimodelo, la importancia relativa de una variable expresa la probabilidad de que esta forme parte de mm_{real}. Entonces, ir_j provee una medida de evidencia respecto a la inclusión del predictor (un alto valor de ir_j indica que es altamente probable que j esté contenido en mm_{real} y, por lo tanto, que sea un predictor importante) que incorpora la incertidumbre respecto al mejor modelo (es decir, sobre si $mm_{muestra} = mm_{real}$). En otras palabras, ir_j considera todos los predictores y todos los modelos puestos a prueba, lo cual es considerada una alternativa superadora respecto a las clásicas medidas de importancia que solo tienen en cuenta los predictores incluidos en $mm_{muestra}$ (como por ejemplo el coeficiente de determinación parcial o los coeficientes de regresión estandarizados).

Aplicando inferencia multimodelo encontramos que la densidad de borde es un predictor de importancia para explicar el rendimiento, y que tiene un efecto no lineal. El tamaño de lote es incluido en un modelo que presenta un ajuste similar a aquel de menor AIC, pero presenta una baja importancia relativa. De hecho, solo se lo incluye en uno de los mejores modelos, el cual presenta la mitad de evidencia relativa que el mejor modelo. Además, encontramos que la fertilización nitrogenada, la región, el cultivo previo, y la densidad y la fecha de siembra explican el rendimiento, y que las interacciones evaluadas no son importantes. Bajo este análisis podemos proponer el siguiente modelo:

```
mfinal <- mejor.aic$`256` # modelo final seleccionando por AIC

mfinal$call # (b + b² + cprevio  + dsiembra + N + fs + fs² + region)

## lm(formula = rendimiento ~ borde + borde2 + cprevio + dsiembra +
##     fs + fs2 + N + region + 1, data = dat.rend2, na.action = na.fail)

mfinal$call # (N + cprevio + region + dsiembra + fs + fs² + b + b²)

## lm(formula = rendimiento ~ N + cprevio + region + dsiembra +
##     fs + fs2 + borde + borde2, data = dat.rend2)
```

7. 7. 3. Alternativa frecuentista

Podemos preguntarnos por qué no seleccionar directamente el modelo a partir de los términos significativos en el ANOVA secuencial del modelo completo. En el contexto del contraste de hipótesis no sería correcto ya que estaríamos rechazando una hipótesis nula (la de que los coeficientes no significativos son todos a la vez iguales a 0) que no estamos poniendo a prueba. En efecto, la significancia de un término puede cambiar cuando se remueve alguno/s de los otros términos debido a que aumenta el número de grados de libertad. La inferencia frecuentista ofrece los contrastes entre pares de modelos como alternativa para la selección de variables. Bajo este enfoque, se van removiendo (o agregando) términos secuencialmente y en cada paso se contrastan los modelos con (m_+) y sin (m) el predictor asociado. La hipótesis nula de cada contraste afirma que m_+ no mejora la bondad de ajuste de m, es decir, que el predictor no afecta a la variable dependiente. Si el valor p del contraste es menor al nivel de significancia, el predictor es eliminado del modelo. El proceso finaliza cuando ya no hay términos por remover o agregar y todos los que se retienen en el modelo son significativos. Con esta estrategia obtenemos un único modelo final desde el cual realizamos las inferencias.

La selección por contrastes presenta algunas críticas. En primer lugar, comenzar por un modelo que incluye solo el intercepto para ir introduciendo predictores de a uno (esta alternativa se conoce como *selección hacia adelante*) es tan válido como comenzar desde el modelo con todos los predictores y remover términos (*selección hacia atrás*). Si el método fuera robusto se debería llegar al mismo modelo independientemente de la dirección de la selección, lo cual no siempre es el caso.

En segundo lugar, el contraste es siempre entre pares de modelos y no todos los modelos se contrastan entre sí (incluso algunos de los modelos podrían no participar de ningún contraste), con lo cual es posible que el modelo final no presente la mejor combinación de predictores. En tercer lugar, al realizar múltiples contrastes, la probabilidad de error de tipo 1 de cada contraste no se mantiene en el nivel de significancia (similar a las pruebas *a posteriori*). No obstante, ha sido el método dominante en las ciencias ambientales hasta hace algunos años y aún sigue siendo utilizado.

7. 7. 4. Sobre el marco inferencial

La inferencia estadística provee las bases teóricas para tomar decisiones (inferir) sobre el todo (población) a partir de la información parcial (muestra). Existen varios marcos inferenciales con diferencias en su filosofía subyacente. A lo largo del libro hemos trabajado con herramientas de inferencia frecuentista y al abordar la selección de variables introdujimos la inferencia multimodelo. Ambos marcos poseen sólidas bases teóricas y deberían aplicarse de acuerdo a la utilidad que brindan en el contexto del problema.

El marco frecuentista basa sus inferencias estableciendo probabilidades de equivocación en las afirmaciones. Estas probabilidades surgen de determinar cuán frecuente (de aquí lo de *frecuentista*) sería el resultado observado si el experimento se repitiera infinitas veces. En particular, bajo este enfoque se obtiene la probabilidad de los datos (resumidos por un estadístico) dada la hipótesis nula.[1] En el contexto de los MLG, la hipótesis de trabajo generalmente es la influencia del predictor (la densidad de borde, por ejemplo) sobre la variable dependiente (rendimiento de girasol), aunque la hipótesis estadística a evaluar (hipótesis nula) expresa la ausencia de efecto. Esto implica que muchas veces se evalúen hipótesis triviales y de poco interés, lo cual es una crítica que se le hace al marco frecuentista. De todas formas, se pueden plantear y evaluar otras hipótesis nulas. Por ejemplo, supongamos que para cubrir los costos de la fertilización nitrogenada se necesita un incremento del rendimiento de 20 kg ha^{-1} por cada kg ha^{-1} de fertilizante que se aplica. En este caso, podemos plantear una hipótesis nula que exprese que el coeficiente de regresión asociado al nitrógeno es (menor) igual a 20 de manera que su rechazo nos permita afirmar que la práctica es rentable. Otra de las críticas al contraste de hipótesis es que no informa la magnitud de la influencia del predictor y solo permite concluir sobre la existencia de efecto. En este sentido, hay cada vez más consenso de que el énfasis debe estar puesto en la estimación y la evidencia, y no en la dicotomía del rechazo/no rechazo de las hipótesis estadísticas. Para suplir esta deficiencia, la inferencia frecuentista provee los intervalos de confianza que informan el tamaño del efecto y la incertidumbre asociada (se demuestra fácilmente que la probabilidad de error de tipo 1 es igual a la probabilidad de que el intervalo de confianza no contenga al parámetro). Además, como aspecto interesante, bajo este marco se concluye controlando el nivel de incertidumbre.

[1] Refiere a una probabilidad condicional: P(datos|H_0). Estrictamente, es la probabilidad de obtener el valor del estadístico, o uno más extremo dado, H_0.

Los CI pueden usarse en lugar del valor *p* en el método de selección hacia atrás o hacia adelante. Esto es similar a la alternativa frecuentista y simplemente estaríamos sustituyendo la regla de decisión. Es dentro del marco de la inferencia multimodelo que se trabaja en simultáneo con muchos modelos de modo de seleccionar aquél o aquellos que tengan más evidencia en su favor. Se incorpora el concepto de *fuerza de la evidencia*, cuya cuantificación se basa en la probabilidad de las hipótesis (modelos) dado los datos,[2] lo cual difiere radicalmente de la visión frecuentista. La fuerza de la evidencia es de carácter continuo, a diferencia de la dicotomía del contraste de hipótesis. Como vimos antes, los modelos son calificados desde el más al menos apropiado y la fuerza de la evidencia de cada uno (w_j, ir_j y otros indicadores) considera al resto de los modelos evaluados. De esta manera, la evidencia en favor de una hipótesis de trabajo se mide en relación al resto de las hipótesis evaluadas. Por todo ello, se considera que el enfoque multimodelo es superador respecto al contraste de hipótesis frecuentista. No obstante sus ventajas, no está exento de riesgos y abusos. En este sentido, es importante enfatizar que para hacer un buen uso de estas herramientas se requiere conocer sus fundamentos, potencialidades y limitaciones, entendiendo que no reemplazan el esfuerzo intelectual de formación y articulación de ideas, sino que lo complementan.

7. 7. 5. Algunas consideraciones más

Exponemos algunas consideraciones que son independientes del marco inferencial utilizado. En relación al esfuerzo de articulación de ideas, un aspecto al cual debemos prestar especial atención es la especificación del modelo de partida que debería incluir solo predictores *a priori* relevantes o de interés conceptual. Esto implica evitar la inclusión de variables solo por el hecho de contar con ellas (es decir, sin una hipótesis previa), así como las interacciones que no tengan sentido ambiental, biológico, socioecológico o agronómico. De esta forma se reduce el riesgo de encontrar patrones o relaciones espurios, es decir, predictores que expliquen la variable respuesta como resultado del azar.

Hemos visto que la selección de variables nos ayuda a determinar la mejor combinación de predictores. En principio, para ello necesitamos evaluar todos los modelos que anida el modelo de partida. Sin embargo, cuando el modelo incluye interacciones, existen restricciones, llamadas de marginalidad, que deberían respetarse. Estas implican que una interacción necesariamente debe ir acompañada de los términos que la conforman. Por ejemplo, el modelo `rendimiento ~ borde + borde:region` no debe evaluarse ya que omite el efecto principal de la región. Lo mismo aplica a las interacciones de mayor orden y a los modelos con polinomios.

No debemos perder de vista que encontrar el mejor modelo no implica necesariamente que este sea un buen modelo. Con nuestro análisis hemos determinado que sus aptitudes son superiores a las del resto de los modelos comparados, pero no cuál es su bondad de

---

[2] A esta probabilidad se la denomina *verosimilitud* y se la expresa como $\mathcal{L}(\theta|\text{datos})$. La diferencia entre verosimilitud y probabilidad consiste en que en la primera varía el valor de los parámetros (θ) bajo datos que se asumen fijos, mientras que en la segunda el parámetro (dado por H_0) es fijo y varían los datos (el estadístico que los resume). En otras palabras, la verosimilitud es función de los parámetros y la probabilidad de los datos. En el método de máxima verosimilitud, el valor de θ que maximiza $\mathcal{L}$ es la estimación de θ.

ajuste o capacidad predictiva. De la misma forma, la validez de los supuestos no está garantizada y estos deben evaluarse siempre.

Finalmente, es importante separar el marco de inferencia del método de estimación. Los contrastes de hipótesis de la inferencia frecuentista, y de igual manera los CI asociados a la inferencia multimodelo, permiten seleccionar variables tanto en modelos ajustados por mínimos cuadrados como en aquellos ajustados por máxima verosimilitud. Al incluir la devianza como componente de bondad de ajuste, los CI son intrínsecamente interesantes en modelos ajustados por máxima verosimilitud. El método de máxima verosimilitud se aplica a una gran variedad de modelos estadísticos además de los modelos lineales generales, por lo que provee mayor generalidad que el método de mínimos cuadrados ordinarios. De la misma forma, los CI permiten comparar modelos tanto anidados como no anidados, aspecto que los convierte en una herramienta más general que los contrastes de hipótesis. Como vimos, otra ventaja de los CI es la posibilidad de comparar más de dos modelos a la vez. A favor de los contrastes de hipótesis, permiten trabajar con tasas de error controlado y proveen una medida de potencia estadística. Indistintamente del método elegido para seleccionar variables, no es correcto mezclar marcos inferenciales en el análisis.[3]

7. 8. Bondad de ajuste

Evaluemos la bondad de ajuste del modelo seleccionado:

```
summary(mfinal)$sigma # error estándar residual

## [1] 355.138

summary(mfinal)$r.squared # R2

## [1] 0.5917818

summary(mfinal)$adj.r.squared # R2 ajustado

## [1] 0.540168
```

Hemos logrado explicar más del 50 % de la variabilidad del rendimiento de girasol. Esto es muy superior al 10 % del modelo de regresión simple con el que hemos comenzado a transitar este libro sobre modelos lineales generales. Veamos el gráfico de observados *vs* predichos:

```
predichos <- fitted(mfinal) # predicciones del modelo seleccionado

# figura 7.6
par(mfrow=c(1,1))
with(dat.rend2, plot(predichos, rendimiento,
                     xlab="Rendimiento predicho (kg/ha)",
                     ylab="Rendimiento observado (kg/ha)",
                     ylim=c(500,4500))) # gráfico de observados vs. predichos
abline(a=0, b=1, lwd=3, col="brown")
```

[3] Algunos autores hasta indican que bajo el marco de inferencia multimodelo debe evitarse el uso del término *significativo*.

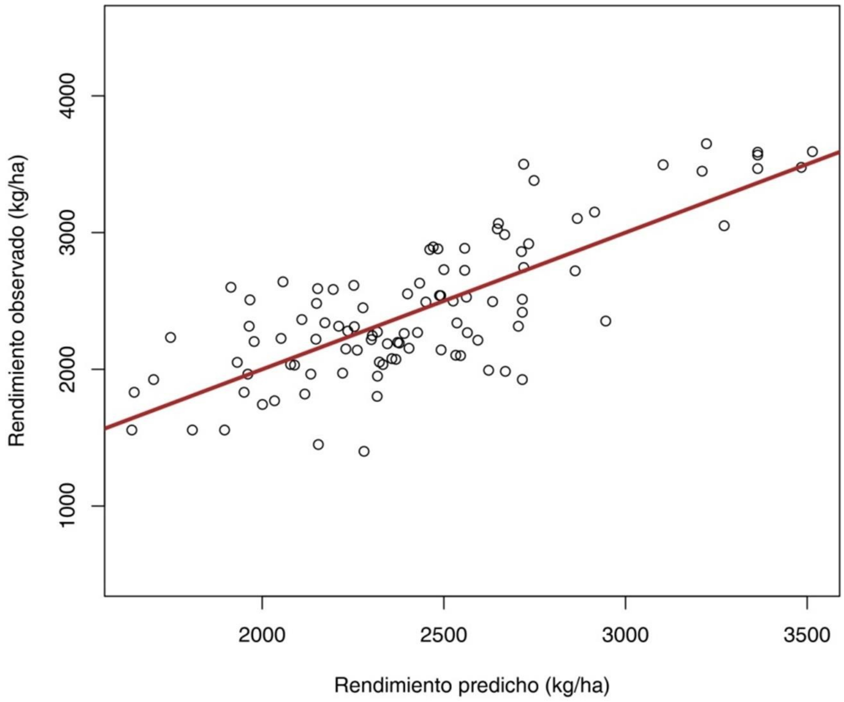

Figura 7. 6. Gráfico de observados *vs* predichos del modelo lineal general (la línea representa la recta uno a uno)

7. 9. Validación de supuestos

Finalmente, evaluamos el cumplimiento de los supuestos:

```
residuos <- resid(mfinal)

# figura 7.7
par(mfrow=c(3,2))
plot(predichos, residuos,
     xlab="Rendimiento predicho (kg/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend2$borde, residuos,
     xlab="Densidad de borde (m/ha)", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend2$fsiembra, residuos,
     xlab="Fecha de siembra", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend2$region, residuos,
     xlab="Región", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
plot(dat.rend2$cprevio, residuos,
     xlab="Cultivo previo", ylab="Residuos (kg/ha)")
abline(a=0, b=0, col="red")
qqnorm(residuos, main="",
       ylab="Cuantiles muestrales", xlab="Cuantiles teóricos")
qqline(residuos, col="red")

# - Test Kolmogorov-Smirnov
ks.test(residuos,"pnorm", mean(residuos), sd(residuos))

##
##  Exact one-sample Kolmogorov-Smirnov test
##
```

```
## data:  residuos
## D = 0.054359, p-value = 0.9162
## alternative hypothesis: two-sided
```

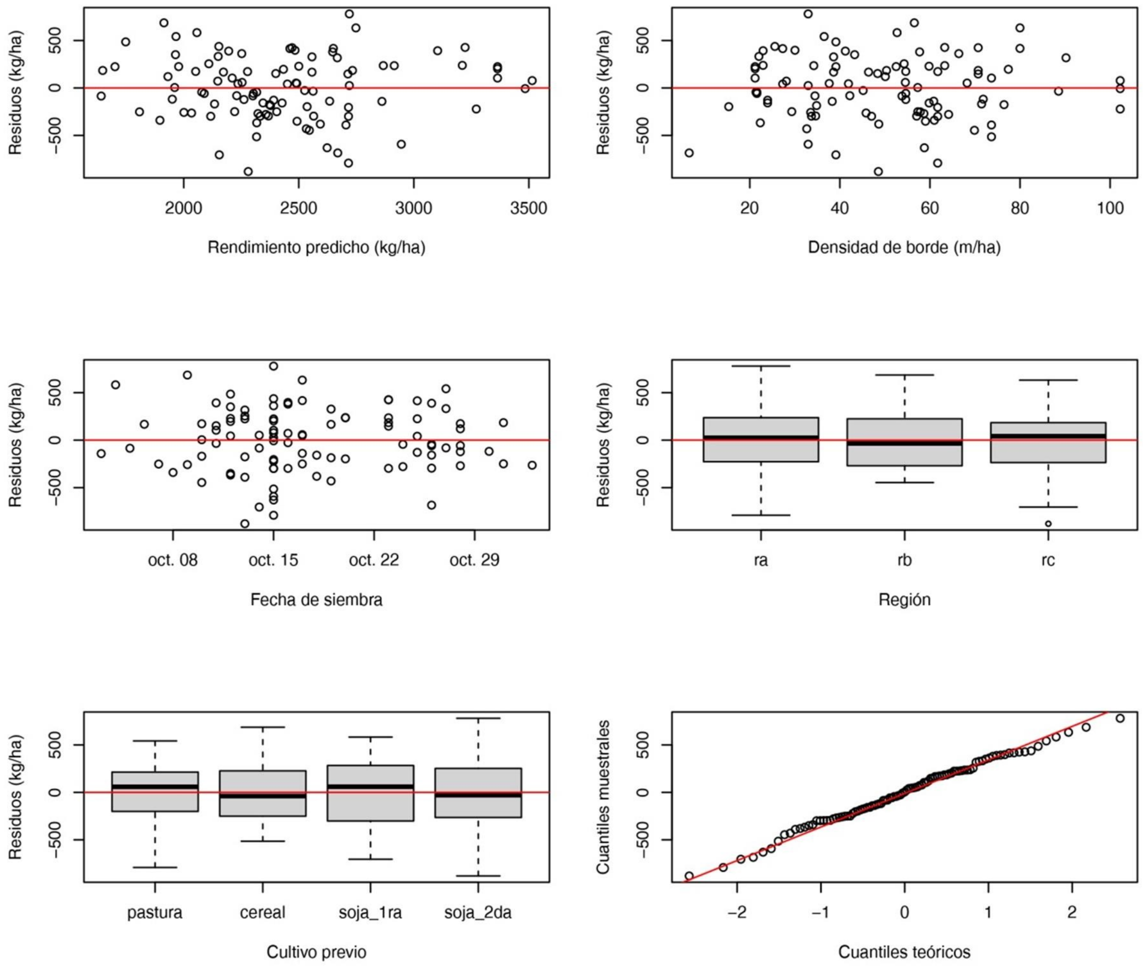

Figura 7. 7. Evaluación gráfica de los supuestos (linealidad, independencia, homocedasticidad y normalidad) del modelo lineal general

La figura 7. 7 muestra que hemos logrado remover todo tipo de patrón en los residuos y, en este sentido, nuestro modelo es robusto al cumplimiento de sus supuestos.

Autorías y colaboraciones

Facundo José Oddi

Universidad de Río Negro, Instituto de Investigaciones en Recursos Naturales, Agroecología y Desarrollo Rural (IRNAD). Río Negro, Argentina.
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET). Argentina.

Es ingeniero forestal por la Universidad Nacional de La Plata (UNLP) y doctor en Biología por la Universidad Nacional del Comahue (UNCO). Es investigador del Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET) y desarrolla sus actividades académicas en el Instituto de Investigaciones en Recursos Naturales, Agroecología y Desarrollo Rural (IRNAD), Universidad Nacional de Río Negro (UNRN). Estudia problemáticas asociadas al uso del suelo y el manejo sustentable de los sistemas forestales y otros paisajes productivos. Sus trabajos de investigación presentan un énfasis espacial, y sus abordajes metodológicos incluyen el uso de las tecnologías de información geográfica y diversas técnicas de modelado. Es director de la carrera de Ingeniería Ambiental de la UNRN, donde además dicta cursos de grado y de posgrado en estadística aplicada.

Lucas Alejandro Garibaldi

Universidad de Río Negro, Instituto de Investigaciones en Recursos Naturales, Agroecología y Desarrollo Rural (IRNAD). Río Negro, Argentina.
Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET). Argentina.

Es ingeniero agrónomo y doctor en Ciencias Agropecuarias, ambos por la Facultad de Agronomía de la Universidad de Buenos Aires (FAUBA). Es investigador del conicet y profesor de la Universidad Nacional de Río Negro. Desarrolla la investigación para fomentar la sostenibilidad ecológica, social y económica de los sistemas agropecuarios y forestales. Sus estudios tratan sobre agroecología, apicultura, biodiversidad, interacciones entre plantas e insectos (herbivoría, polinización, plagas), servicios ambientales y su contribución al bienestar humano. Sus trabajos han tenido un fuerte énfasis cuantitativo, desarrollando y aplicando nuevos modelos estadísticos. Ha recibido numerosos premios y distinciones por su labor científica. Es director del IRNAD, en donde aborda temáticas relevantes y novedosas a nivel local y global, todo ello acoplado con su actividad docente y la vinculación con distintos sectores de la sociedad.

Modelos lineales generales en R. Aplicaciones en ciencias agronómicas y ambientales.
Lucas Alejandro Garibaldi; Facundo José Oddi; Primera edición - Viedma : Universidad Nacional de Río Negro, 2024.
Libro digital, PDF - (Lecturas de Cátedra)
Archivo digital: descarga y online

ISBN 978-987-8258-76-8

1. Estadísticas. 2. Análisis de Datos. 3. Econometría. I. Oddi, Facundo José II. Título
CDD 310

editorial.unrn.edu.ar
Belgrano 526, Viedma, Río Negro, Argentina.

Dirección editorial: Ignacio Artola
Coordinación editorial: Diego Martín Salinas
Edición y corrección de textos: Verónica García Bianchi
Diagramación y diseño: Sergio Campozano
Imagen de tapa: Editorial UNRN, 2024.

Esta obra tuvo el apoyo de la Secretaría de Investigación, Creación Artística, Desarrollo y Transferencia de Tecnología y la Secretaría de Docencia y Vida Estudiantil de la Universidad Nacional de Río Negro.

Modelos lineales generales en R
Aplicaciones en ciencias agronómicas y ambientales.

fue compuesto con la familia tipográfica Liberation
en sus diferentes variables.
Se editó en dicembre de 2024
en la Dirección de Publicaciones-Editorial de la UNRN.

Modelos lineales generales en R

Los profesionales de las ciencias agronómicas y ambientales enfrentan problemas en contextos de incertidumbre, lo que demanda la toma de decisiones basadas en datos. La estadística provee el marco conceptual y metodológico para este propósito, siendo los modelos lineales generales (MLG) una de sus herramientas más útiles.

Este libro, de enfoque aplicado, se centra en los MLG utilizando el programa estadístico R como herramienta. Los contenidos se desarrollan de manera secuencial y están estructurados en torno a una problemática agroecológica recurrente: hábitat natural y rendimiento de cultivos. Cada capítulo comienza con una pregunta profesional específica y guía al lector a través de las etapas esenciales: exploración de datos, formulación del modelo, análisis, toma de decisiones y predicciones. Además, se incluye un capítulo introductorio sobre el manejo básico de R.

Esta obra brinda la capacidad de aplicar con independencia los MLG con la posibilidad de expandir tal capacidad a otras herramientas estadísticas mediante el manejo de R.

LUA
EDITORIAL UNRN
Universidad Nacional de Río Negro
ISBN 978-987-8258-76-8
9 789878 258768