Manual Práctico de easystats en R

Análisis Estadístico Aplicado a Datos Abiertos del Gobierno del Perú

Encuesta de Intenciones de Siembra 2023–2024
Región Cusco

Material didáctico para usuarios principiantes

December 15, 2025

Contents

1 Introducción

1.1 ¿Qué es R y por qué usarlo en análisis estadístico?

R es un lenguaje de programación y entorno de software libre especializado en análisis estadístico y visualización de datos. Desde su creación en 1993 por Ross Ihaka y Robert Gentleman en la Universidad de Auckland, R se ha convertido en una herramienta fundamental en ciencia de datos, investigación académica, y análisis aplicado.

1.1.1 Ventajas de R para análisis estadístico

R ofrece múltiples ventajas que lo hacen ideal para el análisis de datos:

- **Gratuito y de código abierto**: No requiere licencias costosas y su código es auditable por la comunidad.
- **Amplia comunidad**: Miles de paquetes desarrollados por expertos en diversas áreas.
- **Reproducibilidad**: Los análisis pueden ser compartidos y replicados fácilmente.
- **Flexibilidad**: Desde estadística básica hasta machine learning avanzado.
- **Visualización de alta calidad**: Gráficos publicables para revistas académicas.
- **Integración**: Funciona con bases de datos, APIs, y otros lenguajes de programación.

1.2 ¿Qué es el ecosistema easystats?

El ecosistema `easystats` es una colección integrada de paquetes de R diseñada para facilitar el análisis estadístico. Desarrollado por Dominique Makowski, Daniel Lüdecke, Mattan S. Ben-Shachar y colaboradores, easystats busca hacer la estadística más accesible, intuitiva y reproducible.

A diferencia de enfoques tradicionales que requieren conocer múltiples funciones dispersas en diferentes paquetes, easystats proporciona una interfaz coherente y consistente para todo el flujo de análisis estadístico.

1.3 ¿Para qué sirve easystats dentro de un flujo de análisis de datos?

El ecosistema easystats cubre las cinco etapas fundamentales del análisis estadístico:

1. **Modelamiento**: Ajustar modelos estadísticos con sintaxis estándar de R
2. **Evaluación**: Verificar supuestos y desempeño del modelo
3. **Interpretación**: Extraer y comprender parámetros y efectos
4. **Visualización**: Crear gráficos diagnósticos y de resultados

5. **Reporte**: Generar descripciones textuales automáticas de resultados

Easystats actúa como un "traductor" que convierte objetos complejos de modelos estadísticos en tablas, gráficos y texto comprensibles, facilitando tanto el análisis como la comunicación de resultados.

1.4 Ventajas de easystats frente a enfoques tradicionales

1.4.1 Comparación con métodos tradicionales

En R tradicional, obtener un análisis completo requiere conocer funciones de múltiples paquetes:

Listing 1: Enfoque tradicional (fragmentado)

```r
# Ajustar modelo
modelo <- lm(y ~ x1 + x2, data = datos)

# Ver resultados (base R)
summary(modelo)

# Calcular R^2 ajustado manualmente
# Extraer residuos con resid()
# Calcular IC con confint()
# Hacer diagnósticos con plot()
# Calcular tamaños de efecto manualmente
# Escribir interpretación manualmente
```

Con easystats, el mismo análisis es más intuitivo:

Listing 2: Enfoque easystats (integrado)

```r
library(easystats)

# Ajustar modelo
modelo <- lm(y ~ x1 + x2, data = datos)

# Análisis completo
parameters(modelo)   # Parámetros con IC
performance(modelo)  # Métricas de bondad de ajuste
check_model(modelo)  # Diagnósticos visuales
effectsize(modelo)   # Tamaños de efecto
report(modelo)       # Reporte en lenguaje natural
```

1.4.2 Ventajas clave de easystats

- **Consistencia**: La misma sintaxis funciona para regresión lineal, logística, modelos mixtos, etc.
- **Automatización**: Los reportes se generan automáticamente en formato APA.
- **Visualización integrada**: Gráficos diagnósticos con una sola función.

- **Interpretabilidad**: Resultados en lenguaje claro, no solo números.
- **Mejores prácticas**: Incorpora recomendaciones estadísticas modernas.
- **Interoperabilidad**: Funciona con cientos de tipos de modelos existentes.

2 El ecosistema easystats

El ecosistema easystats está compuesto por varios paquetes especializados que trabajan conjuntamente. Cada paquete tiene un propósito específico pero comparte una filosofía común de diseño.

2.1 Paquetes principales

2.1.1 insight: Acceso unificado a modelos

El paquete `insight` es la base del ecosistema. Proporciona una interfaz consistente para extraer información de más de 200 tipos de modelos estadísticos diferentes.

Funciones clave:

- `get_data()`: Extrae los datos utilizados en el modelo
- `find_variables()`: Identifica variables del modelo
- `get_parameters()`: Obtiene coeficientes del modelo
- `model_info()`: Información sobre el tipo de modelo

Utilidad: Permite que los demás paquetes de easystats funcionen con cualquier tipo de modelo sin necesidad de código específico para cada uno.

2.1.2 parameters: Extracción y descripción de parámetros

El paquete `parameters` extrae, calcula y presenta los parámetros de modelos estadísticos de forma estandarizada.

Funciones clave:

- `model_parameters()`: Tabla completa de parámetros con intervalos de confianza
- `standardize_parameters()`: Coeficientes estandarizados
- `p_value()`: Valores p con diferentes métodos
- `bootstrap_parameters()`: Parámetros mediante bootstrap

Utilidad: Proporciona estimaciones, errores estándar, intervalos de confianza y estadísticos de prueba en un formato consistente y legible.

2.1.3 performance: Evaluación de modelos

El paquete `performance` evalúa la calidad y adecuación de modelos estadísticos.

Funciones clave:

- `model_performance()`: Índices de bondad de ajuste (R^2, AIC, RMSE, etc.)
- `check_model()`: Diagnósticos visuales completos
- `check_normality()`: Prueba de normalidad de residuos
- `check_heteroscedasticity()`: Prueba de homogeneidad de varianzas
- `compare_performance()`: Comparación entre múltiples modelos

Utilidad: Verifica supuestos estadísticos y cuantifica qué tan bien el modelo se ajusta a los datos.

2.1.4 effectsize: Tamaños de efecto

El paquete `effectsize` calcula medidas estandarizadas de magnitud de efectos.

Funciones clave:

- `standardize()`: Estandariza variables o coeficientes
- `cohens_d()`: d de Cohen para diferencias entre grupos
- `eta_squared()`: Eta cuadrado para ANOVA
- `effectsize()`: Detecta automáticamente el tamaño de efecto apropiado
- `interpret_*()`: Interpreta la magnitud (pequeño, mediano, grande)

Utilidad: Los tamaños de efecto complementan los valores p al indicar la magnitud práctica de las diferencias o relaciones, no solo su significancia estadística.

2.1.5 modelbased: Estimaciones basadas en modelos

El paquete `modelbased` genera predicciones y estimaciones marginales a partir de modelos.

Funciones clave:

- `estimate_means()`: Medias estimadas por grupo
- `estimate_contrasts()`: Contrastes entre niveles
- `estimate_slopes()`: Pendientes (efectos simples)
- `estimate_response()`: Predicciones del modelo

Utilidad: Facilita la interpretación de modelos complejos (con interacciones o efectos no lineales) mediante estimaciones concretas en valores específicos de las variables.

2.1.6 see: Visualización

El paquete `see` extiende `ggplot2` para crear visualizaciones especializadas de resultados estadísticos.

Funciones clave:

- `plot()`: Método genérico que crea gráficos apropiados para cada objeto easystats
- Gráficos de diagnóstico automáticos
- Visualizaciones de efectos e interacciones
- Gráficos de comparación de modelos

Utilidad: Produce gráficos de calidad publicable con configuración mínima, manteniendo la flexibilidad de ggplot2.

2.1.7 report: Reportes automáticos

El paquete `report` genera descripciones textuales de análisis estadísticos en formato académico.

Funciones clave:

- `report()`: Genera un reporte completo del análisis
- `report_text()`: Solo el texto narrativo
- `report_table()`: Solo la tabla de resultados
- `report_performance()`: Reporte de bondad de ajuste

Utilidad: Automatiza la redacción de resultados estadísticos siguiendo convenciones académicas (APA), ahorrando tiempo y reduciendo errores de transcripción.

2.2 Flujo de trabajo integrado

El poder de easystats reside en la integración de estos paquetes. El flujo típico es:

1. **Ajustar modelo** con funciones estándar de R (`lm()`, `glm()`, `lmer()`, etc.)
2. **Extraer parámetros** con `parameters()`
3. **Evaluar desempeño** con `performance()` y `check_model()`
4. **Calcular efectos** con `effectsize()` y `modelbased()`
5. **Visualizar** con `plot()` del paquete `see`
6. **Reportar** con `report()`

Este flujo es consistente independientemente del tipo de modelo, lo que facilita el aprendizaje y la aplicación a nuevos problemas.

3 Descripción de los datasets utilizados

3.1 Procedencia y contexto

Los datos utilizados en este manual provienen de la plataforma de Datos Abiertos del Gobierno del Perú, específicamente de la Gerencia Regional de Agricultura del Cusco (GERAGRI).

3.1.1 Datos Abiertos del Gobierno del Perú

El Gobierno del Perú, mediante la Autoridad Nacional de Transparencia y Acceso a la Información Pública, promueve la apertura de datos gubernamentales. Estos datos son:

- De acceso público y gratuito
- Legalmente reutilizables para cualquier propósito
- Actualizados periódicamente
- Disponibles en formatos procesables (CSV, JSON, etc.)

Acceso: https://www.datosabiertos.gob.pe/

3.2 Encuesta de Intenciones de Siembra 2023–2024

3.2.1 Descripción general

La Encuesta de Intenciones de Siembra es un instrumento estadístico aplicado por las Direcciones y Gerencias Regionales de Agricultura para estimar las áreas que los agricultores planean sembrar en la próxima campaña agrícola.

Objetivo del dataset: Proporcionar información oportuna sobre las intenciones de siembra de cultivos por distrito, cultivo y mes de siembra, para:

- Planificación agrícola regional
- Estimación de producción futura
- Política de seguridad alimentaria
- Asignación de recursos (semillas, fertilizantes, créditos)

3.2.2 Características técnicas

Table 1: Características del dataset

Característica	**Descripción**
Institución	GERAGRI Cusco
Período	Campaña agrícola 2023–2024
Ámbito geográfico	Región Cusco (13 provincias)
Formato	CSV
Separador	Punto y coma (;)
Codificación	ISO-8859-1 (Latin-1)
Número de registros	Variable (miles de observaciones)
Fecha de corte	10 de noviembre de 2025

3.3 Estructura del dataset

El dataset contiene información desagregada geográficamente y por cultivo:

3.3.1 Variables geográficas

- `DEPARTAMENTO`: Nombre del departamento (Cusco)
- `PROVINCIA`: Nombre de la provincia (13 provincias del Cusco)
- `COD_UBIGEO`: Código único de ubicación geográfica (INEI)
- `DISTRITO`: Nombre del distrito
- `SECTOR_ESTADISTICO`: Subdivisión del distrito para fines de encuesta

3.3.2 Variables agrícolas

- `CULTIVO_AGRICOLA`: Nombre del cultivo (papa, maíz, quinua, etc.)
- `CAMPANIA_AGRICOLA`: Período agrícola (20232024)
- `SIEMBRAS_[MES]`: Área de siembra en hectáreas por mes
  - AGO, SET, OCT, NOV, DIC (agosto–diciembre 2023)
  - ENE, FEB, MAR, ABR, MAY, JUN, JUL (enero–julio 2024)
- `FECHA_CORTE`: Fecha de corte de la información

3.3.3 Tipos de variables

Table 2: Tipología de variables

Variable	Tipo	Escala
DEPARTAMENTO	Categórica nominal	–
PROVINCIA	Categórica nominal	–
COD_UBIGEO	Identificador	–
DISTRITO	Categórica nominal	–
SECTOR_ESTADISTICO	Categórica nominal	–
CULTIVO_AGRICOLA	Categórica nominal	–
CAMPANIA_AGRICOLA	Categórica ordinal	–
SIEMBRAS_*	Numérica continua	Razón (ha)
FECHA_CORTE	Fecha	–

3.4 Justificación del dataset para análisis estadístico

Este dataset es ideal para aprender análisis estadístico por varias razones:

1. **Realidad aplicada**: Datos reales con implicaciones en política pública y economía agrícola.
2. **Estructura jerárquica**: Los datos están naturalmente anidados (sectores dentro de distritos, distritos dentro de provincias), lo que permite explorar modelos multinivel.
3. **Variables diversas**: Combina variables categóricas (cultivo, ubicación) y continuas (área), permitiendo múltiples tipos de análisis.
4. **Potencial de análisis**:
   - Modelos de regresión (¿qué factores explican el área de siembra?)
   - Comparaciones entre grupos (¿difiere la siembra entre provincias?)
   - Series temporales (¿cuáles son los patrones estacionales?)
   - Modelos espaciales (¿existe autocorrelación geográfica?)
5. **Interpretabilidad**: Los resultados son comprensibles para no expertos (hectáreas de papa, no variables abstractas).
6. **Relevancia local**: Para usuarios peruanos, trabajar con datos nacionales es más motivador y relevante que datasets extranjeros.
7. **Accesibilidad**: Datos abiertos, legales y gratuitos que pueden ser utilizados sin restricciones.

4 Preparación de los datos

La preparación de datos es un paso crucial que puede representar hasta el 80% del tiempo en un proyecto de análisis. En esta sección aprenderemos a cargar, explorar, limpiar y transformar los datos de forma reproducible.

4.1 Configuración inicial

Antes de comenzar, debemos cargar los paquetes necesarios y configurar nuestro entorno de trabajo.

Listing 3: Configuración inicial del entorno

```r
# Instalar easystats (solo la primera vez)
# install.packages("easystats")

# Cargar paquetes
library(easystats)    # Ecosistema completo
library(tidyverse)    # Manipulación de datos
library(here)         # Gestión de rutas de archivos

# Configurar opciones
options(scipen = 999) # Desactivar notación científica
options(digits = 3)   # Limitar decimales en salida

# Verificar versiones
sessionInfo()
```

Nota sobre tidyverse: Aunque easystats puede trabajar con R base, utilizaremos funciones de `tidyverse` (particularmente `dplyr` y `tidyr`) para manipulación de datos, ya que su sintaxis es más intuitiva para principiantes.

4.2 Carga de datos

4.2.1 Lectura desde archivo CSV

El dataset está en formato CSV con separador de punto y coma y codificación ISO-8859-1 (Latin-1), común en datos del gobierno peruano.

Listing 4: Carga del dataset desde archivo local

```r
# Cargar datos con encoding correcto
datos_raw <- read.csv2(
  file = "intencionesSIEMBRA.csv",
  encoding = "Latin1",
  stringsAsFactors = FALSE
)

# Alternativa con readr (tidyverse)
library(readr)
datos_raw <- read_delim(
```

```
  file = "intencionesSIEMBRA.csv",
  delim = ";",
  locale = locale(encoding = "Latin1")
)

# Verificar carga exitosa
dim(datos_raw) # Dimensiones: filas y columnas
```

Parámetros importantes:

- `sep = ";"`: El separador es punto y coma, no coma
- `encoding = "Latin1"`: Preserva caracteres especiales (tildes, eñes)
- `stringsAsFactors = FALSE`: Mantiene texto como texto, no factor

4.3 Exploración inicial

Una vez cargados los datos, debemos explorarlos antes de cualquier análisis.

Listing 5: Exploración inicial del dataset

```
# Ver primeras filas
head(datos_raw, n = 10)

# Estructura del dataset
str(datos_raw)

# Resumen estadístico básico
summary(datos_raw)

# Nombres de columnas
names(datos_raw)

# Número de observaciones y variables
nrow(datos_raw) # Filas (observaciones)
ncol(datos_raw) # Columnas (variables)

# Ver valores únicos de variables categóricas
unique(datos_raw$PROVINCIA)
unique(datos_raw$CULTIVO_AGRICOLA)

# Contar observaciones por categoría
table(datos_raw$PROVINCIA)
table(datos_raw$CULTIVO_AGRICOLA)
```

4.3.1 Diagnóstico de datos con datawizard

El paquete `datawizard` (parte de easystats) ofrece funciones especializadas para exploración:

Listing 6: Diagnóstico avanzado con datawizard

```r
library(datawizard)

# Resumen descriptivo completo
describe_distribution(datos_raw)

# Identificar valores faltantes
data_missing(datos_raw)

# Identificar variables constantes
data_findconstants(datos_raw)

# Ver tipos de datos
data_properties(datos_raw)
```

4.4 Limpieza de datos

4.4.1 Manejo de nombres de columnas

Los nombres de columnas con caracteres especiales pueden causar problemas:

Listing 7: Estandarización de nombres de columnas

```r
# Ver nombres actuales
names(datos_raw)

# Opción 1: Limpiar nombres con janitor
library(janitor)
datos <- clean_names(datos_raw)

# Verificar cambios
names(datos)
# "sector_estadístico" se convierte en "sector_estadistico"

# Opción 2: Renombrar manualmente si es necesario
datos <- datos %>%
  rename(
    departamento = DEPARTAMENTO,
    provincia = PROVINCIA,
    cod_ubigeo = COD_UBIGEO,
    distrito = DISTRITO,
    sector = SECTOR_ESTADISTICO,
    cultivo = CULTIVO_AGRICOLA,
    campania = CAMPANIA_AGRICOLA
  )
```

4.4.2 Conversión de tipos de datos

Listing 8: Conversión de tipos de datos

```r
# Convertir variables categóricas a factor
datos <- datos %>%
  mutate(
    departamento = as.factor(departamento),
    provincia = as.factor(provincia),
    distrito = as.factor(distrito),
    cultivo = as.factor(cultivo),
    sector = as.factor(sector)
  )

# Verificar conversión
str(datos)

# Convertir fecha
datos <- datos %>%
  mutate(
    fecha_corte = as.Date(fecha_corte, format = "%Y%m%d")
  )
```

4.4.3 Manejo de valores faltantes

Listing 9: Identificación y tratamiento de valores faltantes

```r
# Contar valores faltantes por columna
colSums(is.na(datos))

# Visualizar patrón de valores faltantes
library(naniar)
vis_miss(datos)

# Decidir estrategia según el caso:

# 1. Eliminar filas con NA (si son pocas)
datos_completos <- na.omit(datos)

# 2. Reemplazar NA con 0 en columnas de siembra
# (asumiendo que NA = no hay siembra)
meses_siembra <- grep("^siembras_", names(datos), value = TRUE)
datos <- datos %>%
  mutate(across(all_of(meses_siembra), ~replace_na(., 0)))

# 3. Imputación más sofisticada (si es necesario)
# library(mice)
# datos_imputados <- mice(datos, m = 5, method = 'pmm')
```

4.5 Creación de variables derivadas

Crear nuevas variables puede facilitar el análisis y la interpretación.

Listing 10: Creación de variables derivadas

```r
# 1. Total de área de siembra por observación
datos <- datos %>%
  mutate(
    area_total = rowSums(select(., starts_with("siembras_")),
                         na.rm = TRUE)
  )

# 2. Indicador binario: hubo siembra?
datos <- datos %>%
  mutate(
    hubo_siembra = ifelse(area_total > 0, 1, 0),
    hubo_siembra = as.factor(hubo_siembra)
  )

# 3. Categorizar cultivos por tipo
datos <- datos %>%
  mutate(
    tipo_cultivo = case_when(
      grepl("PAPA|OLLUCO|OCA|MASHUA", cultivo) ~ "Tubérculos",
      grepl("MAIZ|TRIGO|CEBADA|QUINUA|AVENA", cultivo) ~ "Cereales",
      grepl("HABA|ARVEJA|TARWI", cultivo) ~ "Leguminosas",
      TRUE ~ "Otros"
    ),
    tipo_cultivo = as.factor(tipo_cultivo)
  )

# 4. Mes con mayor siembra
datos <- datos %>%
  rowwise() %>%
  mutate(
    mes_max_siembra = names(.)[which.max(c_across(starts_with("siembras_"))
        )]
  ) %>%
  ungroup()

# 5. Número de meses con siembra
datos <- datos %>%
  mutate(
    n_meses_siembra = rowSums(select(., starts_with("siembras_")) > 0)
  )
```

4.6 Transformación de datos

Para ciertos análisis, necesitamos transformar la estructura de los datos.

4.6.1 De formato ancho a largo

El formato actual es "ancho" (una columna por mes). Para algunos análisis es útil el formato "largo":

Listing 11: Transformación a formato largo

```
# Convertir de ancho a largo
datos_largo <- datos %>%
  pivot_longer(
    cols = starts_with("siembras_"),
    names_to = "mes",
    values_to = "area_ha",
    names_prefix = "siembras_"
  )

# Ver resultado
head(datos_largo)

# Ahora cada combinación cultivo-distrito-mes es una fila
# Esto facilita análisis por mes o gráficos temporales
```

4.7 Verificación final

Antes de proceder al análisis, verificamos que los datos estén listos:

Listing 12: Verificación final de datos preparados

```
# 1. Dimensiones correctas
dim(datos)

# 2. No hay valores faltantes en variables clave
colSums(is.na(datos))

# 3. Tipos de datos correctos
str(datos)

# 4. Rangos razonables en variables numéricas
summary(datos$area_total)

# 5. Niveles apropiados en factores
levels(datos$cultivo)
levels(datos$provincia)

# 6. Guardar datos limpios para análisis
saveRDS(datos, file = "datos_limpios.rds")
write.csv(datos, file = "datos_limpios.csv", row.names = FALSE)
```

4.8 Resumen de la preparación

El proceso de preparación de datos incluye:

1. **Carga**: Leer el archivo con encoding correcto
2. **Exploración**: Entender estructura y contenido
3. **Limpieza**: Estandarizar nombres, tipos, valores faltantes
4. **Transformación**: Crear variables derivadas útiles
5. **Verificación**: Confirmar que los datos están listos

Con los datos preparados, podemos proceder al análisis estadístico con confianza.

5 Caso 1: Regresión Lineal

La regresión lineal es uno de los métodos estadísticos más utilizados y fundamentales. Permite modelar la relación entre una variable respuesta continua y una o más variables predictoras.

5.1 Objetivo del análisis

Queremos responder: **¿Qué factores explican el área total de siembra de papa en la región Cusco?**

5.2 Pregunta estadística

Específicamente, investigaremos:

- ¿El área de siembra varía significativamente entre provincias?
- ¿Existe relación entre el número de meses de siembra y el área total?
- ¿Ciertas variedades de papa (color vs. nativa) se siembran en mayor extensión?

5.3 Preparación de datos específica

Para este análisis, filtraremos solo las observaciones de papa:

Listing 13: Preparación de datos para regresión lineal

```
# Filtrar solo cultivos de papa
datos_papa <- datos %>%
  filter(grepl("PAPA", cultivo)) %>%
  filter(area_total > 0) %>% # Solo donde hay siembra
  droplevels() # Eliminar niveles no usados de factores

# Crear variable de tipo de papa
datos_papa <- datos_papa %>%
  mutate(
    tipo_papa = case_when(
      grepl("COLOR", cultivo) ~ "Papa Color",
```

```
      grepl("NATIVA", cultivo) ~ "Papa Nativa",
      grepl("BLANCA", cultivo) ~ "Papa Blanca",
      TRUE ~ "Otra Papa"
    ),
    tipo_papa = as.factor(tipo_papa)
  )

# Crear variable log de área (para normalizar distribución)
datos_papa <- datos_papa %>%
  mutate(
    log_area = log(area_total + 1) # +1 para evitar log(0)
  )

# Resumen descriptivo
summary(datos_papa$area_total)
table(datos_papa$provincia)
table(datos_papa$tipo_papa)
```

5.4 Modelo estadístico propuesto

Ajustaremos un modelo de regresión lineal múltiple:

$$\text{log\_area}_i = \beta_0 + \beta_1 \text{provincia}_i + \beta_2 \text{tipo\_papa}_i + \beta_3 \text{n\_meses}_i + \epsilon_i \quad (1)$$

Donde:

- log_area_i: Logaritmo del área total de siembra (variable respuesta)
- provincia_i: Provincia (variable categórica)
- tipo_papa_i: Tipo de papa (variable categórica)
- n_meses_i: Número de meses con siembra (variable continua)
- ϵ_i: Error aleatorio, $\epsilon_i \sim N(0, \sigma^2)$

Nota: Usamos transformación logarítmica porque el área de siembra suele tener distribución asimétrica (muchos valores pequeños, pocos valores grandes).

5.5 Ajuste del modelo en R

Listing 14: Ajuste de regresión lineal múltiple

```
# Ajustar modelo
modelo_lm <- lm(
  log_area ~ provincia + tipo_papa + n_meses_siembra,
  data = datos_papa
)

# Ver resumen básico (base R)
summary(modelo_lm)
```

5.6 Análisis con easystats

5.6.1 Extracción de parámetros

Listing 15: Parámetros del modelo con parameters()

```r
# Tabla de parámetros con easystats
params <- model_parameters(modelo_lm)
print(params)

# Parámetros estandarizados (coeficientes beta)
params_std <- standardize_parameters(modelo_lm)
print(params_std)

# Interpretación: Los coeficientes estandarizados permiten comparar
# la magnitud relativa del efecto de cada predictor
```

Interpretación de coeficientes:

- **Intercepto**: Valor esperado de log_area cuando todas las demás variables están en su categoría de referencia
- **provincia[nivel]**: Diferencia en log_area respecto a la provincia de referencia
- **tipo_papa[nivel]**: Diferencia en log_area respecto al tipo de papa de referencia
- **n_meses_siembra**: Por cada mes adicional de siembra, log_area aumenta en β_3 unidades

5.6.2 Evaluación del desempeño del modelo

Listing 16: Métricas de bondad de ajuste con performance()

```r
# Métricas de desempeño
perf <- model_performance(modelo_lm)
print(perf)

# Métricas importantes:
# - R2: Proporción de varianza explicada (0-1)
# - R2_adjusted: R2 ajustado por número de predictores
# - RMSE: Raíz del error cuadrático medio
# - Sigma: Desviación estándar residual
# - AIC/BIC: Criterios de información (menor es mejor)
```

Interpretación de métricas:

Table 3: Interpretación de métricas de desempeño

Métrica	Rango	Interpretación
R^2	0–1	% de varianza explicada
R^2 ajustado	0–1	R^2 penalizado por predictores
RMSE	$0 - \infty$	Error promedio (mismas unidades que Y)
MAE	$0 - \infty$	Error absoluto promedio
AIC	$-\infty - \infty$	Comparar modelos (menor mejor)
BIC	$-\infty - \infty$	AIC con penalización mayor

5.6.3 Verificación de supuestos

La regresión lineal asume:

1. **Linealidad**: Relación lineal entre X e Y
2. **Normalidad**: Residuos siguen distribución normal
3. **Homoscedasticidad**: Varianza constante de residuos
4. **Independencia**: Observaciones independientes

Listing 17: Diagnóstico visual completo con check_model()

```
# Gráficos diagnósticos automáticos
check_model(modelo_lm)

# Este comando genera múltiples gráficos:
# 1. Posterior Predictive Check: Ajuste general
# 2. Linearity: Linealidad de la relación
# 3. Homogeneity of Variance: Homoscedasticidad
# 4. Influential Observations: Valores influyentes
# 5. Normality of Residuals: Normalidad de residuos
# 6. Multicollinearity: Correlación entre predictores
```

Listing 18: Pruebas específicas de supuestos

```
# Prueba de normalidad de residuos
check_normality(modelo_lm)

# Prueba de homogeneidad de varianza
check_heteroscedasticity(modelo_lm)

# Detección de multicolinealidad
check_collinearity(modelo_lm)

# Detección de valores influyentes
check_outliers(modelo_lm)
```

Interpretación de diagnósticos:

- **Si normalidad se viola**: Considerar transformación de Y o modelo robusto
- **Si homoscedasticidad se viola**: Usar errores estándar robustos
- **Si multicolinealidad es alta (VIF $>$ 5)**: Eliminar variables redundantes
- **Si hay outliers influyentes**: Investigar y potencialmente excluir

5.7 Cálculo de tamaños de efecto

Los valores p indican si un efecto es estadísticamente significativo, pero no su magnitud práctica. Los tamaños de efecto cuantifican la importancia práctica.

Listing 19: Tamaños de efecto con effectsize()

```r
# Tamaño de efecto para el modelo completo
effectsize(modelo_lm)

# f^2 de Cohen para cada predictor
f2 <- cohens_f_squared(modelo_lm)
print(f2)

# Interpretación de f^2:
# Pequeño: 0.02
# Mediano: 0.15
# Grande: 0.35

# Interpretar automáticamente
interpret_f_squared(f2)
```

5.8 Visualización de resultados

5.8.1 Gráfico de coeficientes

Listing 20: Visualización de coeficientes estimados

```r
library(see)

# Gráfico de coeficientes con IC al 95%
plot(params)

# Alternativa: solo coeficientes significativos
plot(params, show_intercept = FALSE)

# Los intervalos que no cruzan cero son estadísticamente
# significativos al nivel alfa = 0.05
```

5.8.2 Gráficos de efectos predichos

Listing 21: Predicciones y efectos marginales

```r
library(modelbased)

# Estimar medias por provincia
pred_provincia <- estimate_means(modelo_lm, at = "provincia")
plot(pred_provincia)

# Estimar medias por tipo de papa
pred_tipo <- estimate_means(modelo_lm, at = "tipo_papa")
plot(pred_tipo)

# Efecto del número de meses (manteniendo otras variables constantes)
pred_meses <- estimate_relation(modelo_lm, at = "n_meses_siembra")
plot(pred_meses)
```

5.9 Reporte automático

Listing 22: Generación de reporte con report()

```r
# Reporte completo
report_modelo <- report(modelo_lm)
print(report_modelo)

# Solo texto narrativo
report_text(modelo_lm)

# Solo tabla de resultados
report_table(modelo_lm)

# Reporte de desempeño
report_performance(modelo_lm)
```

El reporte automático genera texto como:

> *"We fitted a linear model (estimated using OLS) to predict log_area with provincia, tipo_papa and n_meses_siembra (formula: log_area ~ provincia + tipo_papa + n_meses_siembra). The model explains a statistically significant and moderate proportion of variance (R2 = 0.XX, F(XX, XXX) = XX.XX, p ¡ .001, adj. R2 = 0.XX). The model's intercept, corresponding to provincia = [referencia], is at X.XX (95% CI [X.XX, X.XX], t(XXX) = XX.XX, p ¡ .001). Within this model: ..."*

5.10 Interpretación de resultados

Para interpretar correctamente los resultados:

1. **Coeficientes**: Cuantifican el cambio en log_area por unidad de cambio en el predictor. Para interpretar en escala original: area $= e^{\text{log\_area}}$

2. **Significancia estadística**: Un valor p < 0.05 indica que el efecto es improbable bajo la hipótesis nula (no hay efecto).

3. **Intervalos de confianza**: Si el IC al 95% no incluye cero, el efecto es significativo al $\alpha = 0.05$.

4. R^2: Indica qué porcentaje de la variabilidad en área de siembra es explicado por el modelo. Un $R^2 = 0.40$ significa que el modelo explica el 40% de la varianza.

5. **Tamaños de efecto**: Complementan los valores p indicando la magnitud práctica del efecto.

Ejemplo de interpretación integrada:

El modelo de regresión lineal explica el 45% de la varianza en el área de siembra de papa ($R^2 = 0.45$, p ¡ 0.001). La provincia tiene un efecto significativo (F = 12.5, p ¡ 0.001, $f^2 = 0.18$, efecto mediano). Específicamente, la provincia de Canchis siembra en promedio 35% más área que la provincia de referencia (Acomayo), manteniendo constantes el tipo de papa y meses de siembra (b = 0.30, IC 95% [0.15, 0.45], p ¡ 0.001). El número de meses de siembra también predice positivamente el área (b = 0.12 por mes adicional, IC 95% [0.08, 0.16], p ¡ 0.001).

5.11 Comparación de modelos

A menudo queremos comparar modelos con diferente complejidad:

Listing 23: Comparación de modelos anidados

```
# Modelo simple (solo provincia)
modelo_1 <- lm(log_area ~ provincia, data = datos_papa)

# Modelo con tipo de papa
modelo_2 <- lm(log_area ~ provincia + tipo_papa, data = datos_papa)

# Modelo completo (añade meses)
modelo_3 <- lm(log_area ~ provincia + tipo_papa + n_meses_siembra,
               data = datos_papa)

# Comparar desempeño
comparacion <- compare_performance(modelo_1, modelo_2, modelo_3)
print(comparacion)

# Visualizar comparación
plot(comparacion)

# Test de razón de verosimilitud (likelihood ratio test)
test_likelihoodratio(modelo_1, modelo_2, modelo_3)
```

Criterios de selección:

- **AIC/BIC más bajo**: Mejor balance entre ajuste y complejidad
- R^2 **ajustado más alto**: Mayor varianza explicada ajustada por predictores
- **Test de razón de verosimilitud significativo**: El modelo más complejo mejora significativamente el ajuste

5.12 Predicción con el modelo

Una vez ajustado y validado el modelo, podemos hacer predicciones:

Listing 24: Predicciones con el modelo ajustado

```
# Crear datos nuevos para predicción
nuevos_datos <- expand.grid(
  provincia = c("CUSCO", "CANCHIS", "QUISPICANCHI"),
  tipo_papa = c("Papa Color", "Papa Nativa"),
  n_meses_siembra = c(1, 2, 3)
)

# Predicciones puntuales
predicciones <- predict(modelo_lm, newdata = nuevos_datos)

# Predicciones con intervalos de confianza
predicciones_ic <- predict(modelo_lm, newdata = nuevos_datos,
                           interval = "confidence", level = 0.95)

# Predicciones con intervalos de predicción (más amplios)
predicciones_pred <- predict(modelo_lm, newdata = nuevos_datos,
                             interval = "prediction", level = 0.95)

# Combinar con datos originales
resultado <- cbind(nuevos_datos, predicciones_ic)

# Volver a escala original (des-logaritmizar)
resultado$area_pred <- exp(resultado$fit)
resultado$area_lwr <- exp(resultado$lwr)
resultado$area_upr <- exp(resultado$upr)

print(resultado)
```

5.13 Resumen del flujo de regresión lineal

El análisis completo de regresión lineal con easystats sigue estos pasos:

1. Preparar datos (filtrado, transformaciones)
2. Ajustar modelo con `lm()`

3. Extraer parámetros con `parameters()`
4. Evaluar desempeño con `performance()`
5. Verificar supuestos con `check_model()`
6. Calcular tamaños de efecto con `effectsize()`
7. Visualizar con `plot()` y `see`
8. Generar reporte con `report()`
9. Interpretar resultados en contexto
10. Comparar modelos alternativos si es relevante
11. Hacer predicciones si es necesario

Este flujo es sistemático, reproducible y aplicable a cualquier análisis de regresión lineal.

6 Caso 2: Regresión Logística

La regresión logística se utiliza cuando la variable respuesta es binaria (0/1, sí/no, presencia/ausencia). Es fundamental en clasificación y predicción de probabilidades.

6.1 Objetivo del análisis

Queremos responder: **¿Qué factores predicen la presencia o ausencia de siembra de quinua en un sector?**

6.2 Pregunta estadística

Específicamente:

- ¿La probabilidad de sembrar quinua varía entre provincias?
- ¿Existe relación entre la altitud (aproximada por distrito) y la siembra de quinua?
- ¿Podemos predecir qué sectores sembrarán quinua basándonos en características geográficas?

6.3 Preparación de datos específica

Listing 25: Preparación de datos para regresión logística

```
# Crear variable binaria: se sembró quinua?
datos_quinua <- datos %>%
  mutate(
    siembra_quinua = ifelse(cultivo == "QUINUA" & area_total > 0, 1, 0)
```

```r
  ) %>%
  # Agregar por sector (un sector puede tener múltiples cultivos)
  group_by(departamento, provincia, distrito, sector) %>%
  summarise(
    siembra_quinua = max(siembra_quinua), # 1 si hubo quinua
    n_cultivos = n(),                     # Número de cultivos
    area_total_sector = sum(area_total), # Área total del sector
    .groups = "drop"
  ) %>%
  mutate(
    siembra_quinua = as.factor(siembra_quinua)
  )

# Ver distribución de la variable respuesta
table(datos_quinua$siembra_quinua)
prop.table(table(datos_quinua$siembra_quinua))

# Idealmente, la distribución no debería ser extremadamente
# desbalanceada (ej. 99% en una categoría)
```

6.4 Modelo estadístico propuesto

Ajustaremos un modelo de regresión logística:

$$\text{logit}(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 \text{provincia}_i + \beta_2 \text{n\_cultivos}_i + \beta_3 \log(\text{area}_i) \tag{2}$$

Donde:

- p_i: Probabilidad de sembrar quinua en el sector i
- $\text{logit}(p_i)$: Transformación logit (log-odds)
- Variables predictoras: provincia, número de cultivos, log(área)

La relación entre probabilidad y log-odds es:

$$p_i = \frac{e^{\beta_0+\beta_1 x_1+\ldots}}{1+e^{\beta_0+\beta_1 x_1+\ldots}} = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1+\ldots)}} \tag{3}$$

6.5 Ajuste del modelo en R

Listing 26: Ajuste de regresión logística

```r
# Ajustar modelo logístico
modelo_logit <- glm(
  siembra_quinua ~ provincia + n_cultivos + log(area_total_sector + 1),
  data = datos_quinua,
  family = binomial(link = "logit")
```

```
)

# Resumen básico
summary(modelo_logit)
```

Parámetros importantes de `glm()`:

- `family = binomial`: Especifica distribución binomial
- `link = "logit"`: Función de enlace logit (por defecto para binomial)
- Alternativas: `link = "probit"` (distribución normal acumulativa)

6.6 Análisis con easystats

6.6.1 Extracción de parámetros

Listing 27: Parámetros del modelo logístico

```
# Parámetros en escala log-odds
params_logit <- model_parameters(modelo_logit)
print(params_logit)

# Parámetros como Odds Ratios (OR)
params_or <- model_parameters(modelo_logit, exponentiate = TRUE)
print(params_or)
```

Interpretación de Odds Ratios:

- OR = 1: No hay efecto
- OR > 1: Aumenta las odds de Y = 1 (efecto positivo)
- OR < 1: Disminuye las odds de Y = 1 (efecto negativo)
- OR = 2: Duplica las odds
- OR = 0.5: Reduce las odds a la mitad

Ejemplo: Si el OR para n_cultivos es 1.5, significa que por cada cultivo adicional, las odds de sembrar quinua aumentan en 50%.

6.6.2 Métricas de desempeño

Listing 28: Evaluación del modelo logístico

```
# Métricas de bondad de ajuste
perf_logit <- model_performance(modelo_logit)
print(perf_logit)

# Métricas importantes:
# - AIC/BIC: Criterios de información
```

```
# - Tjur's R2: Pseudo-R2 apropiado para logística
# - Log-likelihood: Verosimilitud logarítmica
# - Score_log: Score logarítmico (menor es mejor)
```

Pseudo-R^2 en regresión logística:
A diferencia de la regresión lineal, no existe un R^2 único. Existen múltiples pseudo-R^2:

Table 4: Tipos de Pseudo-R^2 para regresión logística

Tipo	Rango	Interpretación
Tjur's R^2	0–1	Diferencia de prob. entre Y=0 y Y=1
McFadden's R^2	0–1	Basado en log-verosimilitud
Nagelkerke's R^2	0–1	Versión ajustada de Cox-Snell

6.6.3 Matriz de confusión y métricas de clasificación

Listing 29: Evaluación de capacidad predictiva

```
# Obtener probabilidades predichas
datos_quinua$prob_pred <- predict(modelo_logit, type = "response")

# Clasificar con umbral 0.5
datos_quinua$pred_clase <- ifelse(datos_quinua$prob_pred > 0.5, 1, 0)
datos_quinua$pred_clase <- as.factor(datos_quinua$pred_clase)

# Matriz de confusión
library(caret)
confusionMatrix(datos_quinua$pred_clase, datos_quinua$siembra_quinua)

# Alternativamente con performance
performance_accuracy(modelo_logit)

# Curva ROC y AUC
library(pROC)
roc_obj <- roc(datos_quinua$siembra_quinua, datos_quinua$prob_pred)
auc(roc_obj)
plot(roc_obj, main = "Curva ROC - Predicción Siembra Quinua")
```

Métricas de clasificación:

- **Accuracy (Precisión)**: % de predicciones correctas
- **Sensitivity (Sensibilidad/Recall)**: % de 1's correctamente identificados
- **Specificity (Especificidad)**: % de 0's correctamente identificados
- **Precision (Precisión positiva)**: % de predicciones "1" que son correctas
- **AUC (Area Under Curve)**: Capacidad discriminatoria general (0.5–1)

6.6.4 Verificación de supuestos

Listing 30: Diagnósticos del modelo logístico

```
# Diagnóstico visual
check_model(modelo_logit)

# Multicolinealidad
check_collinearity(modelo_logit)

# Valores influyentes
check_outliers(modelo_logit)

# Overdispersion (para modelos binomiales)
check_overdispersion(modelo_logit)
```

Nota: La regresión logística no asume normalidad de residuos (el outcome es binario), pero sí asume:

1. Independencia de observaciones
2. Linealidad entre log-odds y predictores continuos
3. Ausencia de multicolinealidad perfecta

6.7 Interpretación de Odds Ratios

Los Odds Ratios son la medida de efecto natural en regresión logística:

Listing 31: Cálculo e interpretación de Odds Ratios

```
# Extraer Odds Ratios con IC
or_tabla <- model_parameters(modelo_logit, exponentiate = TRUE)
print(or_tabla)

# Visualizar Odds Ratios
plot(or_tabla)

# Interpretar automáticamente
interpret_oddsratio(or_tabla$Coefficient)
```

Guía de interpretación:

Table 5: Interpretación práctica de Odds Ratios

OR	Cambio en odds	Interpretación
0.5	−50%	Reduce odds a la mitad
0.8	−20%	Reducción moderada
1.0	0%	Sin efecto
1.2	+20%	Aumento moderado
2.0	+100%	Duplica las odds
3.0	+200%	Triplica las odds

Ejemplo de interpretación completa:

El modelo logístico muestra que el número de cultivos en un sector predice significativamente la siembra de quinua (OR = 1.35, IC 95% [1.18, 1.54], p ¡ 0.001). Específicamente, por cada cultivo adicional, las odds de sembrar quinua aumentan en 35%, manteniendo constantes la provincia y el área. La provincia de Chumbivilcas tiene 2.5 veces más odds de sembrar quinua que la provincia de referencia (OR = 2.50, IC 95% [1.80, 3.47], p ¡ 0.001).

6.8 Predicción de probabilidades

Listing 32: Estimación de probabilidades con modelbased

```r
# Probabilidad media por provincia
prob_provincia <- estimate_expectation(
  modelo_logit,
  at = "provincia"
)
plot(prob_provincia)

# Probabilidades según número de cultivos
prob_cultivos <- estimate_expectation(
  modelo_logit,
  at = "n_cultivos = c(1, 3, 5, 7, 9)"
)
plot(prob_cultivos)

# Superficie de respuesta (2 predictores)
prob_superficie <- estimate_expectation(
  modelo_logit,
  at = c("n_cultivos", "area_total_sector")
)
plot(prob_superficie)
```

6.9 Efectos marginales

Los efectos marginales expresan el cambio en probabilidad (no en odds):

Listing 33: Cálculo de efectos marginales

```r
library(marginaleffects)

# Efecto marginal promedio (AME)
ame <- marginaleffects(modelo_logit)
summary(ame)

# Efecto de un cambio de 1 unidad en cada variable
# sobre la probabilidad predicha

```

```r
# Efectos marginales en valores específicos
mfx_cultivos <- marginaleffects(
  modelo_logit,
  variables = "n_cultivos",
  newdata = datagrid(n_cultivos = 1:10)
)
plot(mfx_cultivos)
```

6.10 Visualización de resultados

Listing 34: Visualizaciones para regresión logística

```r
# 1. Gráfico de Odds Ratios
plot(model_parameters(modelo_logit, exponentiate = TRUE))

# 2. Probabilidades predichas por grupo
library(ggplot2)
ggplot(prob_provincia, aes(x = provincia, y = Predicted)) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = CI_low, ymax = CI_high), width = 0.2) +
  coord_flip() +
  labs(
    title = "Probabilidad de Siembra de Quinua por Provincia",
    x = "Provincia",
    y = "Probabilidad Predicha"
  ) +
  theme_minimal()

# 3. Curva de probabilidad según predictor continuo
ggplot(prob_cultivos, aes(x = n_cultivos, y = Predicted)) +
  geom_line(size = 1) +
  geom_ribbon(aes(ymin = CI_low, ymax = CI_high), alpha = 0.2) +
  labs(
    title = "Efecto del Número de Cultivos en Prob. de Quinua",
    x = "Número de Cultivos",
    y = "Probabilidad de Siembra de Quinua"
  ) +
  theme_minimal()

# 4. Curva ROC
plot(roc_obj,
     main = "Curva ROC",
     print.auc = TRUE,
     col = "blue",
     lwd = 2)
```

6.11 Reporte automático

Listing 35: Reporte de regresión logística

```r
# Reporte completo
report(modelo_logit)

# Reporte de desempeño
report_performance(modelo_logit)

# Reporte de efectos (Odds Ratios)
report_parameters(modelo_logit)
```

6.12 Comparación de modelos logísticos

Listing 36: Comparación de modelos logísticos anidados

```r
# Modelo 1: Solo provincia
m1 <- glm(siembra_quinua ~ provincia,
          data = datos_quinua, family = binomial)

# Modelo 2: + número de cultivos
m2 <- glm(siembra_quinua ~ provincia + n_cultivos,
          data = datos_quinua, family = binomial)

# Modelo 3: + área
m3 <- glm(siembra_quinua ~ provincia + n_cultivos +
          log(area_total_sector + 1),
          data = datos_quinua, family = binomial)

# Comparar
comp <- compare_performance(m1, m2, m3, rank = TRUE)
print(comp)
plot(comp)

# Test de razón de verosimilitud
test_likelihoodratio(m1, m2, m3)
```

6.13 Validación cruzada

Para evaluar la capacidad predictiva en datos no vistos:

Listing 37: Validación cruzada del modelo logístico

```r
library(caret)

# Configurar validación cruzada de 10 pliegues
control <- trainControl(
  method = "cv",
  number = 10,
  classProbs = TRUE,
  summaryFunction = twoClassSummary
```

```r
)

# Preparar datos (caret requiere factores con nombres)
datos_cv <- datos_quinua %>%
  mutate(
    siembra_quinua = factor(
      siembra_quinua,
      levels = c(0, 1),
      labels = c("No", "Si")
    )
  )

# Entrenar con CV
modelo_cv <- train(
  siembra_quinua ~ provincia + n_cultivos +
    log(area_total_sector + 1),
  data = datos_cv,
  method = "glm",
  family = binomial,
  trControl = control,
  metric = "ROC"
)

# Ver resultados
print(modelo_cv)
modelo_cv$results
```

6.14 Resumen del flujo de regresión logística

El análisis completo de regresión logística incluye:

1. Crear variable binaria (0/1)
2. Ajustar modelo con `glm(family = binomial)`
3. Extraer coeficientes en log-odds y OR con `parameters()`
4. Evaluar con pseudo-R^2 y métricas de clasificación
5. Verificar supuestos con `check_model()`
6. Interpretar Odds Ratios
7. Estimar probabilidades con `modelbased`
8. Visualizar OR, probabilidades y curva ROC
9. Generar reporte con `report()`
10. Validar con datos independientes o CV

7 Caso 3: Comparación de Grupos

La comparación de grupos es fundamental cuando queremos determinar si existen diferencias significativas entre dos o más grupos. Utilizaremos pruebas t y ANOVA, complementadas con tamaños de efecto.

7.1 Objetivo del análisis

Queremos responder: **¿Existen diferencias significativas en el área de siembra entre diferentes provincias y tipos de cultivos?**

7.2 Preguntas estadísticas

Investigaremos:

- ¿Difiere el área promedio de siembra entre dos provincias específicas? (prueba t)
- ¿Existen diferencias en área entre múltiples provincias? (ANOVA)
- ¿El tipo de cultivo (tubérculos vs cereales vs leguminosas) afecta el área sembrada?
- ¿Qué tan grandes son estas diferencias? (tamaños de efecto)

7.3 Preparación de datos

Listing 38: Preparación de datos para comparación de grupos

```
# Datos agregados por sector y cultivo
datos_grupos <- datos %>%
  filter(area_total > 0) %>%
  mutate(
    # Clasificar cultivos por tipo
    tipo_cultivo = case_when(
      grepl("PAPA|OLLUCO|OCA|MASHUA", cultivo) ~ "Tubérculos",
      grepl("MAIZ|TRIGO|CEBADA|QUINUA|AVENA|KIWICHA", cultivo) ~ "Cereales"
          ,
      grepl("HABA|ARVEJA|TARWI|FRIJOL", cultivo) ~ "Leguminosas",
      TRUE ~ "Otros"
    ),
    tipo_cultivo = as.factor(tipo_cultivo),
    # Log-transformación
    log_area = log(area_total + 1)
  )

# Resumen descriptivo por grupo
datos_grupos %>%
  group_by(provincia) %>%
  summarise(
    n = n(),
    media = mean(area_total),
```

```
    sd = sd(area_total),
    mediana = median(area_total)
  )

datos_grupos %>%
  group_by(tipo_cultivo) %>%
  summarise(
    n = n(),
    media = mean(area_total),
    sd = sd(area_total),
    mediana = median(area_total)
  )
```

7.4 Caso 3.1: Prueba t para dos grupos

7.4.1 Prueba t independiente

Comparemos el área de siembra entre dos provincias específicas: Cusco y Canchis.

Listing 39: Prueba t independiente con easystats

```
# Filtrar dos provincias
datos_2prov <- datos_grupos %>%
  filter(provincia %in% c("CUSCO", "CANCHIS"))

# Prueba t tradicional (para referencia)
t.test(area_total ~ provincia, data = datos_2prov, var.equal = TRUE)

# Con easystats: información más completa
# Primero ajustamos un modelo lineal simple (equivalente a t-test)
modelo_t <- lm(log_area ~ provincia, data = datos_2prov)

# Parámetros del modelo
parameters(modelo_t)

# El coeficiente de provincia es la diferencia entre grupos

# Tamaño de efecto: d de Cohen
d_cohen <- cohens_d(log_area ~ provincia, data = datos_2prov)
print(d_cohen)
interpret_cohens_d(d_cohen$Cohens_d)

# Visualización
library(ggplot2)
ggplot(datos_2prov, aes(x = provincia, y = log_area, fill = provincia)) +
  geom_boxplot() +
  geom_jitter(alpha = 0.2, width = 0.2) +
  labs(
    title = "Comparación de Área de Siembra entre Cusco y Canchis",
    x = "Provincia",
```

```
    y = "Log(Área en hectáreas)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")
```

Interpretación de d de Cohen:

Table 6: Interpretación de d de Cohen

Valor —d—	Magnitud
< 0.2	Muy pequeño
$0.2 - 0.5$	Pequeño
$0.5 - 0.8$	Mediano
> 0.8	Grande

7.4.2 Prueba t pareada

Si tuviéramos mediciones repetidas (ej. mismo sector en dos años), usaríamos:

Listing 40: Ejemplo de prueba t pareada

```
# Ejemplo hipotético con datos pareados
# t.test(antes, despues, paired = TRUE)

# Con lmer para diseños más complejos:
# library(lme4)
# modelo_pareado <- lmer(area ~ tiempo + (1|sector), data = datos_long)
```

7.5 Caso 3.2: ANOVA de una vía

Cuando comparamos más de dos grupos, utilizamos ANOVA.

Listing 41: ANOVA de una vía con easystats

```
# Modelo ANOVA: área según tipo de cultivo
modelo_anova <- lm(log_area ~ tipo_cultivo, data = datos_grupos)

# Tabla ANOVA
anova_tabla <- anova(modelo_anova)
print(anova_tabla)

# Con easystats
params_anova <- model_parameters(modelo_anova)
print(params_anova)

# Pruebas post-hoc (comparaciones múltiples)
library(emmeans)
medias_est <- emmeans(modelo_anova, specs = "tipo_cultivo")
contrastes <- contrast(medias_est, method = "pairwise", adjust = "tukey")
print(contrastes)
```

```r

# Con modelbased
medias_tipo <- estimate_means(modelo_anova, at = "tipo_cultivo")
print(medias_tipo)
plot(medias_tipo)

# Contrastes
contrasts <- estimate_contrasts(modelo_anova, contrast = "tipo_cultivo")
print(contrasts)
plot(contrasts)
```

7.5.1 Tamaño de efecto para ANOVA

Listing 42: Eta cuadrado y omega cuadrado

```r
# Eta cuadrado (  )
eta_sq <- eta_squared(modelo_anova)
print(eta_sq)
interpret_eta_squared(eta_sq$Eta2)

# Omega cuadrado (  ) - menos sesgado
omega_sq <- omega_squared(modelo_anova)
print(omega_sq)

# Epsilon cuadrado (  ) - el menos sesgado
epsilon_sq <- epsilon_squared(modelo_anova)
print(epsilon_sq)
```

Interpretación de ²:

Table 7: Interpretación de eta cuadrado

²	**Magnitud**
< 0.01	Muy pequeño
$0.01 - 0.06$	Pequeño
$0.06 - 0.14$	Mediano
> 0.14	Grande

7.6 Caso 3.3: ANOVA de dos vías (factorial)

Cuando tenemos dos factores, podemos evaluar sus efectos principales e interacción.

Listing 43: ANOVA factorial con easystats

```r
# Seleccionar provincias principales
datos_factorial <- datos_grupos %>%
  filter(provincia %in% c("CUSCO", "CANCHIS", "QUISPICANCHI")) %>%
  filter(tipo_cultivo %in% c("Tubérculos", "Cereales", "Leguminosas")) %>%
  droplevels()
```

```r

# Modelo ANOVA 2x3 (provincia x tipo_cultivo)
modelo_2way <- lm(log_area ~ provincia * tipo_cultivo,
                  data = datos_factorial)

# Tabla ANOVA tipo II (para diseños desbalanceados)
library(car)
Anova(modelo_2way, type = "II")

# Parámetros del modelo
parameters(modelo_2way)

# Medias estimadas
medias_2way <- estimate_means(
  modelo_2way,
  at = c("provincia", "tipo_cultivo")
)
plot(medias_2way)

# Efectos simples (efecto de provincia para cada tipo de cultivo)
efectos_simples <- estimate_slopes(
  modelo_2way,
  trend = "provincia",
  at = "tipo_cultivo"
)
print(efectos_simples)
plot(efectos_simples)

# Tamaño de efecto para cada factor
eta_sq_2way <- eta_squared(modelo_2way, partial = TRUE)
print(eta_sq_2way)
```

7.6.1 Interpretación de interacciones

Una interacción significativa indica que el efecto de un factor depende del nivel del otro factor.

Listing 44: Visualización de interacciones

```r
# Gráfico de interacción
library(ggplot2)
ggplot(medias_2way, aes(x = provincia, y = Mean,
                        color = tipo_cultivo, group = tipo_cultivo)) +
  geom_line(size = 1) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = CI_low, ymax = CI_high), width = 0.2) +
  labs(
    title = "Interacción entre Provincia y Tipo de Cultivo",
    subtitle = "Efecto en Log(Área de Siembra)",
    x = "Provincia",
```

```
    y = "Media Estimada (Log-escala)",
    color = "Tipo de Cultivo"
  ) +
  theme_minimal()

# Las líneas paralelas indican ausencia de interacción
# Las líneas que se cruzan indican interacción
```

7.7 Verificación de supuestos de ANOVA

Listing 45: Diagnósticos para ANOVA

```
# Diagnósticos visuales
check_model(modelo_anova)

# Supuestos específicos:

# 1. Normalidad de residuos
check_normality(modelo_anova)
plot(check_normality(modelo_anova))

# 2. Homogeneidad de varianzas (homocedasticidad)
check_heteroscedasticity(modelo_anova)

# Test de Levene para homogeneidad de varianzas
library(car)
leveneTest(log_area ~ tipo_cultivo, data = datos_grupos)

# Si se viola homogeneidad: considerar Welch ANOVA
oneway.test(log_area ~ tipo_cultivo, data = datos_grupos,
            var.equal = FALSE)
```

Si los supuestos se violan:

- **No normalidad**: Transformar datos (log, sqrt) o usar pruebas no paramétricas (Kruskal-Wallis)
- **No homogeneidad**: Usar correcciones (Welch ANOVA) o modelos robustos

7.8 Alternativas no paramétricas

Si los supuestos de ANOVA se violan severamente:

Listing 46: Pruebas no paramétricas

```
# Kruskal-Wallis (alternativa no paramétrica a ANOVA)
kruskal.test(area_total ~ tipo_cultivo, data = datos_grupos)

# Pruebas post-hoc para Kruskal-Wallis
library(FSA)
```

```r
dunnTest(area_total ~ tipo_cultivo, data = datos_grupos,
         method = "bonferroni")

# Mann-Whitney U (alternativa no paramétrica a prueba t)
wilcox.test(area_total ~ provincia, data = datos_2prov)

# Tamaño de efecto para Mann-Whitney
rank_biserial(area_total ~ provincia, data = datos_2prov)
```

7.9 Comparaciones múltiples y corrección por múltiples pruebas

Cuando realizamos múltiples comparaciones, aumenta el riesgo de error tipo I (falsos positivos).

Listing 47: Corrección para comparaciones múltiples

```r
# Comparaciones pareadas con ajuste
library(emmeans)
emm <- emmeans(modelo_anova, specs = "tipo_cultivo")

# Diferentes métodos de ajuste:
# 1. Tukey HSD (para todas las comparaciones pareadas)
pairs(emm, adjust = "tukey")

# 2. Bonferroni (más conservador)
pairs(emm, adjust = "bonferroni")

# 3. Holm (menos conservador que Bonferroni)
pairs(emm, adjust = "holm")

# 4. FDR - False Discovery Rate (control de tasa de descubrimientos falsos)
pairs(emm, adjust = "fdr")

# Con modelbased
contrasts_ajustados <- estimate_contrasts(
  modelo_anova,
  contrast = "tipo_cultivo",
  adjust = "tukey"
)
print(contrasts_ajustados)
```

7.10 Visualizaciones avanzadas

Listing 48: Visualizaciones para comparación de grupos

```r
# 1. Boxplot con significancia
library(ggpubr)
ggboxplot(datos_grupos, x = "tipo_cultivo", y = "log_area",
          fill = "tipo_cultivo", palette = "jco") +
```

```
  stat_compare_means(method = "anova", label.y = max(datos_grupos$log_area)
      + 0.5) +
  stat_compare_means(comparisons = list(
    c("Tubérculos", "Cereales"),
    c("Cereales", "Leguminosas"),
    c("Tubérculos", "Leguminosas")
  ), method = "t.test")

# 2. Violin plot con puntos
ggplot(datos_grupos, aes(x = tipo_cultivo, y = log_area,
                         fill = tipo_cultivo)) +
  geom_violin(alpha = 0.7) +
  geom_boxplot(width = 0.2, fill = "white", outlier.shape = NA) +
  geom_jitter(alpha = 0.1, width = 0.2) +
  labs(
    title = "Distribución de Área de Siembra por Tipo de Cultivo",
    x = "Tipo de Cultivo",
    y = "Log(Área en hectáreas)"
  ) +
  theme_minimal() +
  theme(legend.position = "none")

# 3. Gráfico de medias con IC
plot(estimate_means(modelo_anova, at = "tipo_cultivo"))

# 4. Gráfico de contrastes
plot(estimate_contrasts(modelo_anova, contrast = "tipo_cultivo"))
```

7.11 Reporte automático

Listing 49: Reporte de comparación de grupos

```
# Reporte de ANOVA
report(modelo_anova)

# Reporte de prueba t
report(modelo_t)

# Reporte de tamaños de efecto
report(cohens_d(log_area ~ provincia, data = datos_2prov))
report(eta_squared(modelo_anova))
```

7.12 Ejemplo de interpretación integrada

Se realizó un ANOVA de una vía para comparar el área de siembra entre tipos de cultivo (Tubérculos, Cereales, Leguminosas). El análisis reveló un efecto significativo del tipo de cultivo sobre el área sembrada, F(2, 1247) = 45.3, p ¡ 0.001, ^{2} = 0.068 (efecto mediano).

Las comparaciones post-hoc (Tukey HSD) mostraron que los Tubérculos se siembran en mayor extensión promedio (M = 2.8 ha, SD = 1.5) que los Cereales (M = 2.1 ha, SD = 1.2, p ¡ 0.001, d = 0.51) y las Leguminosas (M = 1.6 ha, SD = 0.9, p ¡ 0.001, d = 0.95). Los Cereales también difieren significativamente de las Leguminosas (p ¡ 0.01, d = 0.43). Los tamaños de efecto indican diferencias de magnitud práctica moderada.

7.13 Resumen del flujo de comparación de grupos

Para comparar grupos con easystats:

1. Explorar datos y verificar distribución de grupos
2. Dos grupos: usar `lm(y ~ grupo)` o `t.test()`
3. Múltiples grupos: usar `lm(y ~ grupo)` y `anova()`
4. Calcular tamaños de efecto: `cohens_d()`, `eta_squared()`
5. Verificar supuestos con `check_model()`
6. Comparaciones post-hoc con `emmeans` o `estimate_contrasts()`
7. Visualizar con boxplots, violin plots, gráficos de medias
8. Reportar con `report()`
9. Interpretar significancia estadística Y práctica (tamaño de efecto)

8 Caso 4: Modelos con Interacciones

Las interacciones ocurren cuando el efecto de una variable sobre la respuesta depende del nivel de otra variable. Son comunes en ciencias agrícolas, donde factores ambientales y de manejo interactúan.

8.1 Objetivo del análisis

Queremos responder: **¿El efecto de la provincia sobre el área de siembra difiere según el tipo de cultivo?**

8.2 Pregunta estadística

Específicamente:

- ¿Existe interacción significativa entre provincia y tipo de cultivo?
- ¿Cómo varía el efecto de la provincia para cada tipo de cultivo?
- ¿Cómo se interpretan y visualizan estas interacciones?

8.3 Modelo estadístico con interacción

$$y_{ij} = \beta_0 + \beta_1 X_i + \beta_2 Z_j + \beta_3 (X_i \times Z_j) + \epsilon_{ij} \tag{4}$$

Donde:

- X_i: Provincia (factor A)
- Z_j: Tipo de cultivo (factor B)
- $X_i \times Z_j$: Término de interacción
- β_3: Coeficiente de interacción

Interpretación del término de interacción:

- Si $\beta_3 = 0$: Los efectos son aditivos (sin interacción)
- Si $\beta_3 \neq 0$: Los efectos no son aditivos (hay interacción)

8.4 Preparación de datos

Listing 50: Preparación de datos para modelo con interacciones

```r
# Seleccionar provincias principales y tipos de cultivo
datos_interaccion <- datos %>%
  filter(
    provincia %in% c("CUSCO", "CANCHIS", "QUISPICANCHI", "URUBAMBA"),
    area_total > 0
  ) %>%
  mutate(
    tipo_cultivo = case_when(
      grepl("PAPA|OLLUCO|OCA", cultivo) ~ "Tubérculos",
      grepl("MAIZ|TRIGO|CEBADA|QUINUA", cultivo) ~ "Cereales",
      grepl("HABA|ARVEJA|TARWI", cultivo) ~ "Leguminosas",
      TRUE ~ "Otros"
    ),
    tipo_cultivo = as.factor(tipo_cultivo),
    log_area = log(area_total + 1)
  ) %>%
  filter(tipo_cultivo != "Otros") %>%
  droplevels()

# Resumen por combinación
datos_interaccion %>%
  group_by(provincia, tipo_cultivo) %>%
  summarise(
    n = n(),
    media = mean(area_total),
    sd = sd(area_total),
    .groups = "drop"
  )
```

8.5 Modelos con y sin interacción

Listing 51: Ajuste de modelos con y sin interacción

```
# Modelo sin interacción (efectos principales)
modelo_aditivo <- lm(
  log_area ~ provincia + tipo_cultivo,
  data = datos_interaccion
)

# Modelo con interacción
modelo_interaccion <- lm(
  log_area ~ provincia * tipo_cultivo,
  data = datos_interaccion
)

# El símbolo * indica: efectos principales + interacción
# Equivalente a: provincia + tipo_cultivo + provincia:tipo_cultivo

# Comparar modelos
compare_performance(modelo_aditivo, modelo_interaccion, rank = TRUE)

# Test de significancia de la interacción
test_likelihoodratio(modelo_aditivo, modelo_interaccion)
anova(modelo_aditivo, modelo_interaccion)
```

8.6 Análisis del modelo con interacción

Listing 52: Parámetros del modelo con interacción

```
# Parámetros del modelo
params_int <- model_parameters(modelo_interaccion)
print(params_int)

# Los coeficientes de interacción representan la diferencia
# en el efecto de provincia para cada nivel de tipo_cultivo
# (respecto a los niveles de referencia)

# Desempeño del modelo
performance(modelo_interaccion)

# Tamaño de efecto
eta_squared(modelo_interaccion, partial = TRUE)
```

8.7 Interpretación de interacciones

8.7.1 Efectos simples (Simple Effects)

Los efectos simples muestran el efecto de un factor en cada nivel del otro factor.

Listing 53: Análisis de efectos simples

```r
library(emmeans)

# Medias estimadas para cada combinación
emm_int <- emmeans(modelo_interaccion,
                   specs = ~ provincia | tipo_cultivo)
print(emm_int)

# Efecto de provincia para cada tipo de cultivo
contrast(emm_int, method = "pairwise", by = "tipo_cultivo")

# Alternativamente, efecto de tipo de cultivo para cada provincia
emm_int2 <- emmeans(modelo_interaccion,
                    specs = ~ tipo_cultivo | provincia)
contrast(emm_int2, method = "pairwise", by = "provincia")

# Con modelbased
efectos_prov <- estimate_slopes(
  modelo_interaccion,
  trend = "provincia",
  at = "tipo_cultivo"
)
print(efectos_prov)
plot(efectos_prov)
```

8.8 Visualización de interacciones

La visualización es crucial para entender interacciones.

Listing 54: Visualización de interacciones

```r
# 1. Gráfico de perfiles (líneas)
medias_int <- estimate_means(
  modelo_interaccion,
  at = c("provincia", "tipo_cultivo")
)

library(ggplot2)
ggplot(medias_int, aes(x = provincia, y = Mean,
                       color = tipo_cultivo, group = tipo_cultivo)) +
  geom_line(size = 1.2) +
  geom_point(size = 3) +
  geom_errorbar(aes(ymin = CI_low, ymax = CI_high),
                width = 0.2, alpha = 0.7) +
  labs(
    title = "Interacción: Provincia  Tipo de Cultivo",
    subtitle = "Efecto sobre Log(Área de Siembra)",
    x = "Provincia",
    y = "Media Estimada (Log-escala)",
```

```
    color = "Tipo de Cultivo"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "bottom"
  )

# Interpretación:
# - Líneas paralelas  No hay interacción
# - Líneas que se cruzan  Interacción presente
# - Mayor diferencia entre líneas  Interacción más fuerte

# 2. Gráfico de barras con facetas
ggplot(medias_int, aes(x = provincia, y = Mean, fill = provincia)) +
  geom_col() +
  geom_errorbar(aes(ymin = CI_low, ymax = CI_high), width = 0.3) +
  facet_wrap(~ tipo_cultivo, ncol = 3) +
  labs(
    title = "Área de Siembra por Provincia y Tipo de Cultivo",
    x = "Provincia",
    y = "Media Estimada (Log-escala)"
  ) +
  theme_minimal() +
  theme(
    axis.text.x = element_text(angle = 45, hjust = 1),
    legend.position = "none"
  )

# 3. Heatmap de interacción
ggplot(medias_int, aes(x = provincia, y = tipo_cultivo, fill = Mean)) +
  geom_tile() +
  geom_text(aes(label = round(Mean, 2)), color = "white", size = 5) +
  scale_fill_viridis_c() +
  labs(
    title = "Heatmap de Interacción",
    x = "Provincia",
    y = "Tipo de Cultivo",
    fill = "Log(Área)"
  ) +
  theme_minimal() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

8.9 Descomposición de la interacción

Listing 55: Análisis detallado de la interacción

```
# Contrastes de interacción
contrast(emm_int, interaction = c("pairwise", "pairwise"))
```

```r

# Con modelbased: contrastes específicos
contrasts_int <- estimate_contrasts(
  modelo_interaccion,
  contrast = c("provincia", "tipo_cultivo")
)
print(contrasts_int)

# Visualizar contrastes
plot(contrasts_int)
```

8.10 Interacciones continuas × categóricas

Cuando una variable es continua y otra categórica:

Listing 56: Interacción continua × categórica

```r
# Modelo: área según número de meses y tipo de cultivo
modelo_cont_cat <- lm(
  log_area ~ n_meses_siembra * tipo_cultivo,
  data = datos_interaccion
)

# Parámetros
parameters(modelo_cont_cat)

# La interacción indica que la pendiente de n_meses difiere
# entre tipos de cultivo

# Visualizar pendientes
pendientes <- estimate_slopes(
  modelo_cont_cat,
  trend = "n_meses_siembra",
  at = "tipo_cultivo"
)
print(pendientes)

# Gráfico de pendientes
ggplot(datos_interaccion,
       aes(x = n_meses_siembra, y = log_area, color = tipo_cultivo)) +
  geom_point(alpha = 0.3) +
  geom_smooth(method = "lm", se = TRUE, size = 1.5) +
  labs(
    title = "Interacción entre Meses de Siembra y Tipo de Cultivo",
    x = "Número de Meses con Siembra",
    y = "Log(Área en hectáreas)",
    color = "Tipo de Cultivo"
  ) +
  theme_minimal()

```

```
# Si las pendientes son similares  poca interacción
# Si las pendientes difieren sustancialmente  interacción fuerte
```

8.11 Interacciones continuas × continuas

Para dos variables continuas:

Listing 57: Interacción entre dos variables continuas

```
# Ejemplo: área según n_meses y log(area_total_sector)
# (requiere datos agregados por sector)

datos_continuas <- datos %>%
  group_by(distrito, sector) %>%
  summarise(
    n_cultivos = n(),
    area_total = sum(area_total),
    n_meses_promedio = mean(n_meses_siembra),
    .groups = "drop"
  ) %>%
  filter(area_total > 0) %>%
  mutate(log_area = log(area_total))

modelo_cont_cont <- lm(
  log_area ~ n_cultivos * n_meses_promedio,
  data = datos_continuas
)

parameters(modelo_cont_cont)

# Superficie de respuesta 3D
library(modelbased)
pred_superficie <- estimate_response(
  modelo_cont_cont,
  data = expand.grid(
    n_cultivos = seq(min(datos_continuas$n_cultivos),
                     max(datos_continuas$n_cultivos),
                     length.out = 30),
    n_meses_promedio = seq(min(datos_continuas$n_meses_promedio),
                           max(datos_continuas$n_meses_promedio),
                           length.out = 30)
  )
)

# Gráfico de contorno
ggplot(pred_superficie,
       aes(x = n_cultivos, y = n_meses_promedio, z = Predicted)) +
  geom_contour_filled(alpha = 0.8) +
  geom_point(data = datos_continuas,
             aes(x = n_cultivos, y = n_meses_promedio),
```

```
            inherit.aes = FALSE, alpha = 0.3) +
  labs(
    title = "Superficie de Respuesta: Interacción Continua",
    x = "Número de Cultivos",
    y = "Número de Meses Promedio",
    fill = "Log(Área)\nPredicha"
  ) +
  theme_minimal()
```

8.12 Cuándo incluir interacciones

Razones teóricas para incluir interacciones:

- Existe evidencia previa o teoría que sugiere interacción
- Los factores están relacionados conceptualmente
- Se busca modelar efectos contextuales o de moderación

Razones empíricas:

- La interacción mejora significativamente el ajuste del modelo
- Los gráficos exploratorios sugieren no-aditividad
- El test de razón de verosimilitud es significativo

Precauciones:

- Las interacciones de orden superior (≥ 3 factores) son difíciles de interpretar
- Aumentan la complejidad del modelo y reducen grados de libertad
- Requieren más datos para estimarse con precisión
- Pueden llevar a sobreajuste si no están justificadas

8.13 Verificación de supuestos

Listing 58: Diagnósticos del modelo con interacción

```
# Diagnósticos visuales completos
check_model(modelo_interaccion)

# Normalidad
check_normality(modelo_interaccion)

# Homoscedasticidad
check_heteroscedasticity(modelo_interaccion)

# Multicolinealidad (importante en modelos con interacciones)
```

```r
check_collinearity(modelo_interaccion)

# Nota: La interacción X*Z está naturalmente correlacionada con X y Z
# VIF altos para términos de interacción son esperables
# Lo importante es centrar las variables antes si VIF > 10
```

8.14 Centrado de variables

Para reducir multicolinealidad en interacciones con variables continuas:

Listing 59: Centrado de variables para reducir multicolinealidad

```r
# Centrar variables continuas (restar la media)
datos_centrados <- datos_continuas %>%
  mutate(
    n_cultivos_c = n_cultivos - mean(n_cultivos),
    n_meses_c = n_meses_promedio - mean(n_meses_promedio)
  )

# Modelo con variables centradas
modelo_centrado <- lm(
  log_area ~ n_cultivos_c * n_meses_c,
  data = datos_centrados
)

# Comparar multicolinealidad
compare_performance(modelo_cont_cont, modelo_centrado)
check_collinearity(modelo_cont_cont)
check_collinearity(modelo_centrado)

# Las estimaciones son idénticas, pero la multicolinealidad se reduce
# La interpretación del intercepto cambia (ahora es el valor en las medias)
```

8.15 Reporte de modelos con interacción

Listing 60: Reporte automático de interacciones

```r
# Reporte completo
report(modelo_interaccion)

# Reporte enfocado en la interacción
report_text(modelo_interaccion)

# Tabla de parámetros
report_table(modelo_interaccion)
```

8.16 Ejemplo de interpretación integrada

Se ajustó un modelo de regresión lineal con interacción para evaluar si el efecto de la provincia sobre el área de siembra varía según el tipo de cultivo. El modelo con interacción ($R^2 = 0.52$) tuvo un ajuste significativamente mejor que el modelo aditivo ($R^2 = 0.48$), $\chi^2(6) = 45.2$, p ¡ 0.001, indicando que la interacción es significativa.

El análisis de efectos simples reveló que el efecto de la provincia difiere sustancialmente entre tipos de cultivo. Para Tubérculos, la provincia de Canchis siembra 45% más área que Cusco (b = 0.37, IC 95% [0.25, 0.49], p ¡ 0.001), mientras que para Cereales esta diferencia es solo del 15% (b = 0.14, IC 95% [0.05, 0.23], p ¡ 0.01). Para Leguminosas, no se observó diferencia significativa entre provincias (b = 0.08, IC 95% [-0.05, 0.21], p = 0.23).

Estos resultados sugieren que las políticas agrícolas deben considerar no solo las diferencias provinciales, sino también cómo estas varían según el tipo de cultivo cultivado.

8.17 Resumen del análisis de interacciones

Pasos para analizar interacciones con easystats:

1. Justificar teórica o empíricamente la inclusión de interacciones
2. Ajustar modelo con término de interacción (`X * Z`)
3. Comparar con modelo aditivo usando `compare_performance()`
4. Extraer parámetros con `parameters()`
5. Analizar efectos simples con `emmeans` o `estimate_slopes()`
6. Visualizar interacciones con gráficos de perfiles o superficies
7. Verificar supuestos (especialmente multicolinealidad)
8. Centrar variables continuas si es necesario
9. Interpretar en contexto del problema
10. Reportar con `report()`

9 Caso 5: Modelos de Efectos Mixtos

Los modelos mixtos (también llamados multinivel o jerárquicos) son apropiados cuando los datos tienen estructura anidada o agrupada, como observaciones dentro de distritos, distritos dentro de provincias, etc.

9.1 Objetivo del análisis

Queremos responder: **¿Cómo modelar el área de siembra considerando la estructura jerárquica geográfica de los datos?**

9.2 ¿Cuándo usar modelos mixtos?

Los modelos mixtos son necesarios cuando:

- **Datos agrupados**: Observaciones anidadas en grupos (ej. sectores dentro de distritos)
- **Medidas repetidas**: Múltiples mediciones del mismo sujeto
- **Estructura jerárquica**: Niveles naturales de organización (estudiantes en escuelas, pacientes en hospitales)
- **Heterogeneidad entre grupos**: Los grupos difieren sistemáticamente
- **Variabilidad intra e intergrupo**: Interesa modelar ambas fuentes de variación

Ventajas sobre regresión estándar:

- Controla por la no independencia de observaciones
- Proporciona estimaciones específicas por grupo (BLUPs)
- Permite modelar variabilidad en interceptos y pendientes
- Más potencia estadística y estimaciones precisas

9.3 Estructura de datos jerárquicos

En nuestro dataset:

- **Nivel 1**: Observaciones individuales (cultivo en sector)
- **Nivel 2**: Sectores estadísticos
- **Nivel 3**: Distritos
- **Nivel 4**: Provincias

Listing 61: Exploración de estructura jerárquica

```
# Contar observaciones por nivel
datos %>%
  summarise(
    n_obs = n(),
    n_sectores = n_distinct(sector),
    n_distritos = n_distinct(distrito),
    n_provincias = n_distinct(provincia)
```

```
  )

# Observaciones por grupo
datos %>%
  group_by(distrito) %>%
  summarise(n = n()) %>%
  summary()

# Idealmente, cada grupo debería tener múltiples observaciones
```

9.4 Modelo estadístico de efectos mixtos

9.4.1 Modelo con intercepto aleatorio

El modelo más simple incluye solo interceptos aleatorios:

$$y_{ij} = (\beta_0 + u_{0j}) + \beta_1 X_{ij} + \epsilon_{ij} \tag{5}$$

Donde:

- y_{ij}: Observación i en grupo j
- β_0: Intercepto fijo (media poblacional)
- u_{0j}: Desviación del grupo j respecto al intercepto ($u_{0j} \sim N(0, \tau^2)$)
- β_1: Pendiente fija (igual para todos los grupos)
- ϵ_{ij}: Residuo ($\epsilon_{ij} \sim N(0, \sigma^2)$)

Interpretación: Cada grupo tiene su propio intercepto, pero la pendiente es común.

9.4.2 Modelo con pendiente aleatoria

Podemos permitir que la pendiente también varíe entre grupos:

$$y_{ij} = (\beta_0 + u_{0j}) + (\beta_1 + u_{1j}) X_{ij} + \epsilon_{ij} \tag{6}$$

Donde u_{1j} es la desviación de la pendiente del grupo j respecto a la pendiente media.

9.5 Ajuste de modelos mixtos en R

Listing 62: Preparación de datos para modelos mixtos

```
# Filtrar y preparar datos
datos_mixtos <- datos %>%
  filter(
    area_total > 0,
    provincia %in% c("CUSCO", "CANCHIS", "QUISPICANCHI", "URUBAMBA")
  ) %>%
  mutate(
```

```
    tipo_cultivo = case_when(
      grepl("PAPA|OLLUCO|OCA", cultivo) ~ "Tubérculos",
      grepl("MAIZ|TRIGO|CEBADA|QUINUA", cultivo) ~ "Cereales",
      grepl("HABA|ARVEJA|TARWI", cultivo) ~ "Leguminosas",
      TRUE ~ "Otros"
    ),
    tipo_cultivo = as.factor(tipo_cultivo),
    log_area = log(area_total + 1)
  ) %>%
  filter(tipo_cultivo != "Otros") %>%
  droplevels()

# Verificar número de observaciones por distrito
datos_mixtos %>%
  group_by(distrito) %>%
  summarise(n = n()) %>%
  arrange(n)

# Filtrar distritos con suficientes observaciones (>= 5)
distritos_suficientes <- datos_mixtos %>%
  group_by(distrito) %>%
  summarise(n = n()) %>%
  filter(n >= 5) %>%
  pull(distrito)

datos_mixtos <- datos_mixtos %>%
  filter(distrito %in% distritos_suficientes) %>%
  droplevels()
```

9.5.1 Modelo nulo (solo intercepto aleatorio)

Listing 63: Modelo nulo con lme4

```
library(lme4)

# Modelo nulo (sin predictores, solo intercepto aleatorio por distrito)
modelo_nulo <- lmer(
  log_area ~ 1 + (1 | distrito),
  data = datos_mixtos,
  REML = TRUE
)

summary(modelo_nulo)

# Con easystats
parameters(modelo_nulo)
performance(modelo_nulo)
```

Cálculo del ICC (Coeficiente de Correlación Intraclase):

Listing 64: Coeficiente de Correlación Intraclase

```
# ICC: proporción de varianza entre grupos
library(performance)
icc_resultado <- icc(modelo_nulo)
print(icc_resultado)

# Interpretación del ICC:
# ICC = 0: No hay agrupación (usar regresión estándar)
# ICC = 0.05-0.10: Agrupación pequeña pero presente
# ICC > 0.10: Agrupación sustancial (modelos mixtos recomendados)
# ICC cercano a 1: Observaciones dentro de grupos son muy similares
```

9.5.2 Modelo con predictores de nivel 1

Listing 65: Modelo con predictores fijos e intercepto aleatorio

```
# Modelo con tipo de cultivo (predictor de nivel 1)
modelo_nivel1 <- lmer(
  log_area ~ tipo_cultivo + (1 | distrito),
  data = datos_mixtos,
  REML = TRUE
)

# Parámetros
params_m1 <- model_parameters(modelo_nivel1)
print(params_m1)

# Desempeño
perf_m1 <- model_performance(modelo_nivel1)
print(perf_m1)

# Comparar con modelo nulo
compare_performance(modelo_nulo, modelo_nivel1)
test_likelihoodratio(modelo_nulo, modelo_nivel1)
```

9.5.3 Modelo con predictores de nivel 2

Listing 66: Modelo con predictor de nivel 2 (provincia)

```
# Provincia es nivel 2 (contiene distritos)
# Primero agregar provincia al nivel de distrito
datos_mixtos <- datos_mixtos %>%
  mutate(provincia = as.factor(provincia))

modelo_nivel2 <- lmer(
  log_area ~ tipo_cultivo + provincia + (1 | distrito),
  data = datos_mixtos,
  REML = TRUE
```

```
)

# Nota: provincia es un predictor de nivel 2, pero distrito está
# anidado en provincia, así que el efecto aleatorio controla
# por variabilidad dentro de provincia

parameters(modelo_nivel2)
performance(modelo_nivel2)
```

9.5.4 Modelo con pendiente aleatoria

Listing 67: Modelo con intercepto y pendiente aleatorios

```
# Permitir que el efecto de tipo_cultivo varíe por distrito
modelo_pendiente <- lmer(
  log_area ~ tipo_cultivo + (1 + tipo_cultivo | distrito),
  data = datos_mixtos,
  REML = TRUE
)

# Este modelo permite que cada distrito tenga:
# - Su propio intercepto
# - Su propia pendiente para tipo_cultivo

summary(modelo_pendiente)
parameters(modelo_pendiente)

# Comparar modelos
compare_performance(modelo_nivel1, modelo_pendiente)

# Nota: Modelos con pendientes aleatorias requieren más datos
# y pueden no converger si hay pocos grupos o datos por grupo
```

9.5.5 Estructura de efectos aleatorios anidados

Listing 68: Efectos aleatorios anidados en múltiples niveles

```
# Efectos aleatorios en distrito Y provincia
modelo_anidado <- lmer(
  log_area ~ tipo_cultivo + n_meses_siembra +
    (1 | provincia/distrito),
  data = datos_mixtos,
  REML = TRUE
)

# La sintaxis (1 | provincia/distrito) es equivalente a:
# (1 | provincia) + (1 | provincia:distrito)
# Es decir, efectos aleatorios en provincia y en distrito dentro de
    provincia
```

```
12
13 summary(modelo_anidado)
14 parameters(modelo_anidado)
15
16 # Extrae componentes de varianza
17 get_variance(modelo_anidado)
```

9.6 Análisis de resultados

9.6.1 Extracción de efectos fijos

Listing 69: Parámetros de efectos fijos

```
1 # Efectos fijos (parámetros poblacionales)
2 efectos_fijos <- model_parameters(
3   modelo_nivel2,
4   effects = "fixed"
5 )
6 print(efectos_fijos)
7
8 # Visualizar efectos fijos
9 plot(efectos_fijos)
```

9.6.2 Extracción de efectos aleatorios

Listing 70: Efectos aleatorios (BLUPs)

```
1 # Efectos aleatorios (desviaciones por grupo)
2 efectos_aleatorios <- ranef(modelo_nivel2)
3 print(efectos_aleatorios)
4
5 # BLUPs: Best Linear Unbiased Predictors
6 # Son las estimaciones específicas de cada grupo
7
8 # Visualizar con lattice
9 library(lattice)
10 dotplot(ranef(modelo_nivel2, condVar = TRUE))
11
12 # Distritos con efectos aleatorios más extremos
13 efectos_df <- as.data.frame(efectos_aleatorios)
14 efectos_df %>% arrange(desc(condval))
15
16 # Estos representan distritos con mayor/menor área
17 # de siembra después de controlar por tipo de cultivo
```

9.6.3 Componentes de varianza

Listing 71: Descomposición de varianza

```r
# Extraer componentes de varianza
var_components <- get_variance(modelo_nivel2)
print(var_components)

# Varianza explicada por efectos fijos
var_fixed <- var_components$var.fixed

# Varianza entre grupos (distrito)
var_random <- var_components$var.random

# Varianza residual (dentro de grupos)
var_residual <- var_components$var.residual

#  R  condicional: varianza explicada por efectos fijos + aleatorios
#  R  marginal: varianza explicada solo por efectos fijos
r2_modelo <- r2(modelo_nivel2)
print(r2_modelo)

#  R  marginal: proporción explicada por predictores fijos
#  R  condicional: proporción explicada por modelo completo (fijos +
    aleatorios)
```

9.7 Evaluación del modelo

Listing 72: Diagnósticos de modelos mixtos

```r
# Diagnósticos visuales
check_model(modelo_nivel2)

# Normalidad de residuos
check_normality(modelo_nivel2)
plot(check_normality(modelo_nivel2))

# Normalidad de efectos aleatorios
qqnorm(ranef(modelo_nivel2)$distrito[,1])
qqline(ranef(modelo_nivel2)$distrito[,1])

# Homoscedasticidad
check_heteroscedasticity(modelo_nivel2)

# Valores influyentes
check_outliers(modelo_nivel2)

# Convergencia (importante en modelos mixtos complejos)
check_convergence(modelo_nivel2)
```

9.8 Predicciones con modelos mixtos

Listing 73: Predicciones de modelos mixtos

```r
# Predicciones de nivel poblacional (usando solo efectos fijos)
pred_poblacion <- predict(modelo_nivel2, re.form = NA)

# Predicciones de nivel de grupo (usando efectos fijos + aleatorios)
pred_grupo <- predict(modelo_nivel2)

# Con modelbased
# Medias estimadas por tipo de cultivo (nivel poblacional)
medias_poblacion <- estimate_means(
  modelo_nivel2,
  at = "tipo_cultivo"
)
print(medias_poblacion)
plot(medias_poblacion)

# Predicciones específicas por distrito
# (usando el efecto aleatorio de cada distrito)
nuevos_datos <- expand.grid(
  tipo_cultivo = levels(datos_mixtos$tipo_cultivo),
  distrito = unique(datos_mixtos$distrito)[1:5], # Primeros 5 distritos
  provincia = unique(datos_mixtos$provincia)[1]
)

pred_distrito <- predict(modelo_nivel2, newdata = nuevos_datos)
```

9.9 Visualización de modelos mixtos

Listing 74: Visualizaciones para modelos mixtos

```r
# 1. Gráfico de efectos aleatorios (caterpillar plot)
library(sjPlot)
plot_model(modelo_nivel2, type = "re")

# 2. Gráfico de efectos fijos
plot_model(modelo_nivel2, type = "est")

# 3. Predicciones por grupo
library(ggeffects)
pred_eff <- ggpredict(modelo_nivel2, terms = c("tipo_cultivo", "distrito [
    sample=10]"))
plot(pred_eff)

# 4. Comparar interceptos entre distritos
ggplot(data.frame(
  distrito = rownames(ranef(modelo_nivel2)$distrito),
```

```
  intercepto = ranef(modelo_nivel2)$distrito[,1]
)) +
  aes(x = reorder(distrito, intercepto), y = intercepto) +
  geom_point(size = 3) +
  geom_hline(yintercept = 0, linetype = "dashed", color = "red") +
  coord_flip() +
  labs(
    title = "Efectos Aleatorios por Distrito",
    subtitle = "Desviaciones del intercepto poblacional",
    x = "Distrito",
    y = "Efecto Aleatorio (Intercepto)"
  ) +
  theme_minimal()

# 5. Gráfico de líneas por grupo (si hay pendiente aleatoria)
# Mostrar regresión específica de cada grupo
if("tipo_cultivo" %in% names(ranef(modelo_pendiente)$distrito)) {
  ggplot(datos_mixtos, aes(x = tipo_cultivo, y = log_area, group = distrito
      )) +
    geom_line(alpha = 0.3) +
    geom_line(stat = "smooth", method = "lm", se = FALSE, aes(group = NULL)
        ,
              size = 2, color = "red") +
    labs(
      title = "Pendientes Específicas por Distrito",
      subtitle = "Línea roja = efecto fijo poblacional",
      x = "Tipo de Cultivo",
      y = "Log(Área)"
    ) +
    theme_minimal()
}
```

9.10 Comparación de modelos mixtos

Listing 75: Selección del mejor modelo mixto

```
# Comparar múltiples modelos
comp_mixtos <- compare_performance(
  modelo_nulo,
  modelo_nivel1,
  modelo_nivel2,
  modelo_pendiente,
  rank = TRUE
)
print(comp_mixtos)
plot(comp_mixtos)

# Criterios de selección:
# - AIC/BIC más bajo
```

```r
# - R condicional más alto
# - Parsimonia (modelo más simple que explica bien)

# Test de razón de verosimilitud (para modelos anidados)
# IMPORTANTE: Usar REML = FALSE para comparar modelos con diferentes efectos fijos
modelo_ml1 <- lmer(log_area ~ tipo_cultivo + (1 | distrito),
                   data = datos_mixtos, REML = FALSE)
modelo_ml2 <- lmer(log_area ~ tipo_cultivo + provincia + (1 | distrito),
                   data = datos_mixtos, REML = FALSE)

test_likelihoodratio(modelo_ml1, modelo_ml2)
```

9.11 Modelos mixtos generalizados (GLMM)

Para respuestas no gaussianas (ej. binarias, conteos):

Listing 76: Modelo logístico mixto

```r
# Ejemplo: Presencia/ausencia de quinua por sector
datos_glmm <- datos %>%
  mutate(
    siembra_quinua = ifelse(cultivo == "QUINUA" & area_total > 0, 1, 0)
  ) %>%
  group_by(distrito, sector) %>%
  summarise(
    siembra_quinua = max(siembra_quinua),
    n_cultivos = n(),
    .groups = "drop"
  ) %>%
  filter(distrito %in% distritos_suficientes)

# Modelo logístico mixto
modelo_logit_mixto <- glmer(
  siembra_quinua ~ n_cultivos + (1 | distrito),
  data = datos_glmm,
  family = binomial(link = "logit")
)

summary(modelo_logit_mixto)
parameters(modelo_logit_mixto, exponentiate = TRUE) # Odds Ratios
performance(modelo_logit_mixto)

# ICC para modelos binomiales
icc(modelo_logit_mixto)
```

9.12 Reporte de modelos mixtos

Listing 77: Reporte automático de modelos mixtos

```r
# Reporte completo
report(modelo_nivel2)

# Reporte de efectos fijos
report_parameters(modelo_nivel2, effects = "fixed")

# Reporte de efectos aleatorios
report_parameters(modelo_nivel2, effects = "random")

# Reporte de desempeño
report_performance(modelo_nivel2)
```

9.13 Ejemplo de interpretación integrada

Se ajustó un modelo lineal mixto para predecir el área de siembra (log-transformada), incluyendo tipo de cultivo y provincia como efectos fijos, y distrito como efecto aleatorio. El modelo mostró un buen ajuste (R^2 condicional = 0.58, R^2 marginal = 0.35), indicando que los efectos fijos explican el 35% de la varianza y el modelo completo (fijos + aleatorios) explica el 58%.

El coeficiente de correlación intraclase fue ICC = 0.28, indicando que el 28% de la varianza en área de siembra se debe a diferencias entre distritos, justificando el uso de modelos mixtos. El tipo de cultivo tuvo un efecto significativo: Los Tubérculos se siembran en mayor área que los Cereales (b = 0.42, SE = 0.08, t = 5.25, p ¡ 0.001) y las Leguminosas (b = 0.68, SE = 0.09, t = 7.56, p ¡ 0.001).

Los efectos aleatorios mostraron heterogeneidad sustancial entre distritos (SD = 0.52), con algunos distritos sembrado consistentemente más área que otros, incluso después de controlar por tipo de cultivo y provincia. Esta variabilidad podría deberse a factores locales no medidos como altitud, acceso a agua, o tradiciones agrícolas específicas.

9.14 Resumen del flujo de modelos mixtos

Pasos para analizar con modelos mixtos:

1. Identificar estructura jerárquica o agrupación en los datos
2. Calcular ICC con modelo nulo para justificar modelo mixto
3. Ajustar modelos con `lmer()` (continuos) o `glmer()` (otros)
4. Especificar efectos aleatorios apropiados: `(1 | grupo)`
5. Extraer efectos fijos y aleatorios con `parameters()`
6. Evaluar con R^2 marginal y condicional

7. Verificar supuestos con `check_model()`
8. Comparar modelos con `compare_performance()`
9. Visualizar efectos con `plot_model()` o `ggeffects`
10. Interpretar en términos de efectos poblacionales (fijos) y variabilidad entre grupos (aleatorios)
11. Reportar con `report()`

10 Visualización y Comunicación de Resultados

La visualización efectiva es fundamental para comunicar resultados estadísticos. El paquete `see` de easystats facilita la creación de gráficos de calidad publicable.

10.1 Filosofía de visualización en easystats

El paquete `see` extiende `ggplot2` con métodos especializados para objetos de easystats:

- **Automatización inteligente**: Detecta el tipo de objeto y crea el gráfico apropiado
- **Estética profesional**: Paletas de colores y temas diseñados para publicación
- **Personalizable**: Todos los gráficos son objetos ggplot2 que pueden modificarse
- **Consistencia**: Misma sintaxis para diferentes tipos de análisis

10.2 Gráficos de parámetros

Listing 78: Visualización de coeficientes de regresión

```
library(see)

# Ajustar modelo
modelo <- lm(log_area ~ provincia + tipo_cultivo + n_meses_siembra,
             data = datos_papa)

# Extraer parámetros
params <- model_parameters(modelo)

# Gráfico básico
plot(params)

# Personalizar
plot(params,
     show_intercept = FALSE, # Omitir intercepto
     sort = TRUE) +         # Ordenar por magnitud
  labs(title = "Coeficientes del Modelo de Regresión",
       subtitle = "Intervalos de confianza al 95%") +
```

```
  theme_modern()

# Parámetros estandarizados
params_std <- standardize_parameters(modelo)
plot(params_std) +
  labs(title = "Coeficientes Estandarizados (Betas)")
```

10.3 Gráficos de diagnóstico

Listing 79: Diagnósticos visuales completos

```
# Diagnósticos automáticos (múltiples paneles)
check_model(modelo)

# Diagnósticos individuales
check_normality(modelo)
plot(check_normality(modelo))

check_heteroscedasticity(modelo)
plot(check_heteroscedasticity(modelo))

check_collinearity(modelo)
plot(check_collinearity(modelo))

check_outliers(modelo)
plot(check_outliers(modelo))

# Personalizar diagnósticos
check_model(modelo,
            check = c("linearity", "homogeneity", "qq", "outliers"),
            colors = c("blue", "red"))
```

10.4 Gráficos de efectos y predicciones

Listing 80: Visualización de efectos marginales

```
library(modelbased)

# Medias estimadas
medias <- estimate_means(modelo, at = "tipo_cultivo")
plot(medias) +
  labs(title = "Medias Estimadas por Tipo de Cultivo",
       y = "Log(Área de Siembra)")

# Efectos según variable continua
efecto_meses <- estimate_relation(modelo, at = "n_meses_siembra")
plot(efecto_meses) +
  labs(title = "Efecto del Número de Meses de Siembra",
```

```
      x = "Meses de Siembra",
      y = "Predicción de Log(Área)")

# Contrastes entre grupos
contrasts <- estimate_contrasts(modelo, contrast = "tipo_cultivo")
plot(contrasts) +
  labs(title = "Contrastes entre Tipos de Cultivo")
```

10.5 Gráficos de comparación de modelos

Listing 81: Comparación visual de modelos

```
# Comparar desempeño
m1 <- lm(log_area ~ provincia, data = datos_papa)
m2 <- lm(log_area ~ provincia + tipo_papa, data = datos_papa)
m3 <- lm(log_area ~ provincia + tipo_papa + n_meses_siembra, data = datos_
    papa)

comp <- compare_performance(m1, m2, m3)
plot(comp) +
  labs(title = "Comparación de Modelos")

# Visualizar múltiples métricas
plot(comp, type = "radar")
```

10.6 Gráficos para modelos logísticos

Listing 82: Visualizaciones específicas para regresión logística

```
# Modelo logístico
modelo_logit <- glm(siembra_quinua ~ provincia + n_cultivos,
                    data = datos_quinua, family = binomial)

# Odds Ratios
or_params <- model_parameters(modelo_logit, exponentiate = TRUE)
plot(or_params) +
  geom_vline(xintercept = 1, linetype = "dashed", color = "red") +
  labs(title = "Odds Ratios",
       subtitle = "OR > 1: aumenta probabilidad | OR < 1: disminuye
           probabilidad")

# Probabilidades predichas
probs <- estimate_expectation(modelo_logit, at = "provincia")
plot(probs) +
  labs(title = "Probabilidad de Siembra de Quinua por Provincia",
       y = "Probabilidad")
```

10.7 Paletas de colores y temas

Listing 83: Personalización de estética

```r
library(see)

# Temas disponibles
# theme_modern() - Limpio y moderno
# theme_lucid() - Alto contraste
# theme_abyss() - Fondo oscuro
# theme_blackboard() - Estilo pizarra

# Paletas de colores
# Paletas discretas
plot(params) +
  scale_color_material_d() + # Material Design
  theme_modern()

# Paletas continuas
plot(params) +
  scale_color_viridis_c() +
  theme_lucid()

# Paletas personalizadas
colores_cusco <- c("#E63946", "#F1FAEE", "#A8DADC", "#457B9D", "#1D3557")
plot(medias) +
  scale_fill_manual(values = colores_cusco) +
  theme_abyss()
```

10.8 Exportación de gráficos

Listing 84: Guardar gráficos para publicación

```r
# Crear gráfico
p <- plot(params, show_intercept = FALSE) +
  labs(title = "Coeficientes del Modelo de Papa",
       subtitle = "Región Cusco, Campaña 2023-2024") +
  theme_modern()

# Guardar en alta resolución
ggsave(
  filename = "grafico_coeficientes.png",
  plot = p,
  width = 10,
  height = 6,
  dpi = 300,
  units = "in"
)

```

```r
# Formato PDF (vectorial, ideal para publicaciones)
ggsave(
  filename = "grafico_coeficientes.pdf",
  plot = p,
  width = 10,
  height = 6
)

# Formato TIFF (requerido por algunas revistas)
ggsave(
  filename = "grafico_coeficientes.tiff",
  plot = p,
  width = 10,
  height = 6,
  dpi = 600,
  compression = "lzw"
)
```

10.9 Gráficos combinados (multipanel)

Listing 85: Combinación de múltiples gráficos

```r
library(patchwork)

# Crear gráficos individuales
g1 <- plot(params, show_intercept = FALSE) +
  labs(title = "A) Coeficientes")

g2 <- plot(check_normality(modelo)) +
  labs(title = "B) Normalidad")

g3 <- plot(medias) +
  labs(title = "C) Medias Estimadas")

g4 <- plot(check_collinearity(modelo)) +
  labs(title = "D) Multicolinealidad")

# Combinar en panel 2x2
(g1 | g2) / (g3 | g4) +
  plot_annotation(
    title = "Análisis Completo del Modelo de Siembra de Papa",
    subtitle = "Región Cusco",
    caption = "Fuente: GERAGRI Cusco, Encuesta de Intenciones de Siembra
        2023-2024"
  )

# Guardar panel
ggsave("panel_analisis.png", width = 14, height = 10, dpi = 300)
```

10.10 Tablas para publicación

Listing 86: Exportación de tablas en formato publicable

```r
# Tabla de parámetros
params_table <- model_parameters(modelo)

# Exportar como dataframe
as.data.frame(params_table)

# Formato APA con kableExtra
library(kableExtra)
kable(params_table,
      format = "latex",
      booktabs = TRUE,
      caption = "Parámetros del modelo de regresión lineal") %>%
  kable_styling(latex_options = c("striped", "hold_position"))

# Formato para Word
library(flextable)
flextable(as.data.frame(params_table)) %>%
  theme_vanilla() %>%
  autofit()

# Exportar a Excel
library(writexl)
write_xlsx(as.data.frame(params_table), "parametros_modelo.xlsx")
```

10.11 Reportes reproducibles con R Markdown

Listing 87: Integración con R Markdown

```r
# En un documento .Rmd:
# ```{r analisis, echo=TRUE, message=FALSE, warning=FALSE}
# library(easystats)
#
# # Cargar datos
# datos <- readRDS("datos_limpios.rds")
#
# # Ajustar modelo
# modelo <- lm(log_area ~ provincia + tipo_papa, data = datos)
#
# # Reportar automáticamente
# report(modelo)
# ```
#
# ```{r diagnosticos, fig.width=12, fig.height=8}
# check_model(modelo)
# ```
```

```
#
# El reporte se genera automáticamente en HTML, PDF o Word
```

11 Buenas Prácticas y Errores Comunes

Esta sección recopila recomendaciones para un análisis estadístico robusto y reproducible, así como errores frecuentes a evitar.

11.1 Buenas prácticas generales

11.1.1 Documentación y reproducibilidad

Listing 88: Estructura de proyecto reproducible

```
# Estructura recomendada de directorios:
# proyecto/
#          datos/
#               raw/            # Datos originales (nunca modificar)
#               processed/      # Datos limpios
#          scripts/
#               01_cargar_datos.R
#               02_limpiar_datos.R
#               03_analisis_exploratorio.R
#               04_modelamiento.R
#          resultados/
#               figuras/
#               tablas/
#          reportes/
#          README.md

# Usar proyectos de RStudio (.Rproj)
# Usar control de versiones (Git)

# Documentar sesión
sessionInfo()
writeLines(capture.output(sessionInfo()), "session_info.txt")

# Usar here() para rutas relativas
library(here)
datos <- read.csv(here("datos", "processed", "datos_limpios.csv"))
```

11.1.2 Comentarios y estilo de código

Listing 89: Código bien documentado

```
# BUENA PRÁCTICA: Comentarios claros y código legible

# Cargar paquetes necesarios
```

```r
library(easystats)
library(tidyverse)

# Cargar y preparar datos de siembra
datos <- read.csv("intencionesSIEMBRA.csv", encoding = "Latin1") %>%
  # Filtrar solo observaciones con siembra
  filter(area_total > 0) %>%
  # Crear variable log-transformada para normalizar
  mutate(log_area = log(area_total + 1)) %>%
  # Convertir variables categóricas a factor
  mutate(across(c(provincia, cultivo), as.factor))

# Ajustar modelo de regresión lineal
# Variable respuesta: log del área
# Predictores: provincia y tipo de cultivo
modelo <- lm(log_area ~ provincia + tipo_cultivo, data = datos)

# MALA PRÁCTICA: Código sin comentarios
d<-read.csv("i.csv",encoding="L1")%>%filter(at>0)%>%mutate(la=log(at+1))%>%
    mutate(across(c(p,c),as.factor))
m<-lm(la~p+tc,data=d)
```

11.1.3 Flujo de análisis sistemático

1. **Planificación**: Definir preguntas de investigación y análisis antes de ver los datos
2. **Exploración**: Entender los datos antes de modelar
3. **Limpieza**: Documentar todas las decisiones de limpieza
4. **Modelamiento**: Ajustar modelo, verificar supuestos, interpretar
5. **Validación**: Verificar robustez con análisis de sensibilidad
6. **Comunicación**: Reportar con claridad, incluir limitaciones

11.2 Errores comunes en análisis estadístico

11.2.1 Error 1: P-hacking y HARKing

P-hacking: Probar múltiples análisis hasta encontrar un valor p ¡ 0.05.

HARKing: Hypothesizing After Results are Known (formular hipótesis después de ver resultados).

Listing 90: Evitar P-hacking

```r
# MALO: Probar muchos modelos hasta encontrar significancia
modelo1 <- lm(y ~ x1, data = datos) # p = 0.08, no significativo
modelo2 <- lm(y ~ x2, data = datos) # p = 0.06, no significativo
modelo3 <- lm(y ~ x3, data = datos) # p = 0.04, significativo!
# ... reportar solo modelo3
```

```
6
7  # BUENO: Pre-registrar análisis, reportar todos los modelos
8  # 1. Definir hipótesis a priori
9  # 2. Especificar análisis planeados
10 # 3. Reportar todos los resultados, no solo los "significativos"
11 # 4. Distinguir análisis confirmatorios de exploratorios
12
13 # Si múltiples modelos son plausibles, reportar todos
14 modelos <- list(
15   "Modelo 1" = lm(y ~ x1, data = datos),
16   "Modelo 2" = lm(y ~ x2, data = datos),
17   "Modelo 3" = lm(y ~ x1 + x2, data = datos)
18 )
19
20 compare_performance(modelos)
```

11.2.2 Error 2: Ignorar supuestos del modelo

Listing 91: Verificar siempre los supuestos

```
1  # MALO: Ajustar modelo y reportar sin verificar supuestos
2  modelo <- lm(area_total ~ provincia, data = datos)
3  summary(modelo) # Asumir que está bien
4
5  # BUENO: Verificar supuestos sistemáticamente
6  modelo <- lm(log_area ~ provincia, data = datos) # Log para normalizar
7
8  # Verificación completa
9  check_model(modelo)
10
11 # Verificaciones específicas
12 check_normality(modelo)    # Residuos normales?
13 check_heteroscedasticity(modelo) # Varianza homogénea?
14 check_collinearity(modelo) # Multicolinealidad?
15 check_outliers(modelo)     # Valores influyentes?
16
17 # Si supuestos se violan, tomar medidas correctivas:
18 # - Transformar variables
19 # - Usar modelos robustos
20 # - Usar métodos no paramétricos
21 # - Incluir más predictores
```

11.2.3 Error 3: Confundir significancia estadística con importancia práctica

Listing 92: Reportar tamaños de efecto

```
1  # MALO: Enfocarse solo en valores p
2  modelo <- lm(log_area ~ provincia, data = datos)
```

```r
summary(modelo) # "p < 0.001, es significativo!"

# BUENO: Reportar significancia estadística Y tamaño de efecto
parameters(modelo)        # Coeficientes e IC
effectsize(modelo)        # Tamaño de efecto
cohens_f_squared(modelo)  #  f  de Cohen

# Interpretar
# p < 0.001: Poco probable que no haya efecto (significancia)
#  f  = 0.02: Pero el efecto es muy pequeño (importancia práctica)

# Reportar: "El efecto es estadísticamente significativo (p < 0.001)
# pero de magnitud práctica pequeña (f = 0.02)."
```

11.2.4 Error 4: Sobreajuste del modelo

Listing 93: Evitar sobreajuste

```r
# MALO: Incluir demasiados predictores
# Regla general: Al menos 10-20 observaciones por predictor
n <- nrow(datos) # 100 observaciones
modelo_sobreajustado <- lm(
  y ~ x1 + x2 + x3 + x4 + x5 + x6 + x7 + x8 + x9 + x10, # 10 predictores
  data = datos
)  # Ratio: 10 obs/predictor (límite)

# BUENO: Usar selección de variables justificada
# 1. Basar en teoría/conocimiento previo
# 2. Usar validación cruzada
# 3. Comparar modelos con criterios de información

# Selección paso a paso (con precaución)
library(MASS)
modelo_completo <- lm(y ~ ., data = datos)
modelo_step <- stepAIC(modelo_completo, direction = "both")

# Validación cruzada
library(caret)
train_control <- trainControl(method = "cv", number = 10)
modelo_cv <- train(y ~ ., data = datos, method = "lm",
                   trControl = train_control)
```

11.2.5 Error 5: Ignorar datos faltantes

Listing 94: Manejo apropiado de valores faltantes

```r
# MALO: Eliminar filas con NA sin pensar
datos_completos <- na.omit(datos) # Sesgo? Pérdida de información?
```

```
3
4  # BUENO: Analizar patrón de valores faltantes
5  library(naniar)
6  vis_miss(datos) # Visualizar NA
7  gg_miss_var(datos) # NA por variable
8
9  # Son MCAR (completamente aleatorios)?
10 # Son MAR (aleatorios condicionalmente)?
11 # Son MNAR (no aleatorios)?
12
13 # Opciones según el caso:
14 # 1. Si pocos NA y MCAR: na.omit() es aceptable
15 # 2. Si muchos NA o MAR: imputación múltiple
16 library(mice)
17 imp <- mice(datos, m = 5, method = "pmm")
18 modelo_imp <- with(imp, lm(y ~ x1 + x2))
19 pool(modelo_imp)
20
21 # 3. Si MNAR: modelos de selección o pattern-mixture
```

11.2.6 Error 6: Múltiples comparaciones sin ajuste

Listing 95: Ajustar por comparaciones múltiples

```
1  # MALO: Hacer muchas comparaciones sin ajuste
2  # Probabilidad de al menos un falso positivo aumenta con cada prueba
3  pvalues <- c()
4  for(i in 1:20) {
5    test <- t.test(grupo1[[i]], grupo2[[i]])
6    pvalues[i] <- test$p.value
7  }
8  sum(pvalues < 0.05) # Probablemente algunos falsos positivos!
9
10 # BUENO: Ajustar valores p
11 library(emmeans)
12 emm <- emmeans(modelo_anova, specs = "grupo")
13
14 # Método conservador
15 pairs(emm, adjust = "bonferroni")
16
17 # Método balanceado
18 pairs(emm, adjust = "holm")
19
20 # Control de False Discovery Rate
21 pairs(emm, adjust = "fdr")
22
23 # Para familias de hipótesis
24 p_adjust(pvalues, method = "BH") # Benjamini-Hochberg
```

11.3 Buenas prácticas específicas con easystats

11.3.1 Usar funciones consistentes

Listing 96: Aprovechar la consistencia de easystats

```
# La misma sintaxis funciona para múltiples tipos de modelos

# Regresión lineal
m1 <- lm(y ~ x, data = datos)

# Regresión logística
m2 <- glm(y ~ x, data = datos, family = binomial)

# Modelo mixto
library(lme4)
m3 <- lmer(y ~ x + (1 | grupo), data = datos)

# Para todos:
parameters(m1)
parameters(m2)
parameters(m3)

performance(m1)
performance(m2)
performance(m3)

check_model(m1)
check_model(m2)
check_model(m3)

report(m1)
report(m2)
report(m3)
```

11.3.2 Combinar paquetes del ecosistema

Listing 97: Flujo integrado completo

```
library(easystats) # Carga todos los paquetes

# 1. Preparar datos (datawizard)
datos <- datos_raw %>%
  standardize() # Estandarizar variables

# 2. Ajustar modelo
modelo <- lm(y ~ x1 + x2, data = datos)

# 3. Parámetros (parameters)
params <- model_parameters(modelo)
```

```
12
13 # 4. Desempeño (performance)
14 perf <- model_performance(modelo)
15
16 # 5. Verificar supuestos (performance)
17 check_model(modelo)
18
19 # 6. Tamaños de efecto (effectsize)
20 efectos <- effectsize(modelo)
21
22 # 7. Estimaciones (modelbased)
23 medias <- estimate_means(modelo, at = "x1")
24
25 # 8. Visualizar (see)
26 plot(params)
27 plot(medias)
28
29 # 9. Reportar (report)
30 report(modelo)
```

11.4 Checklist de análisis

Antes de reportar resultados, verificar:

- ☐ ¿Los datos están limpios y bien preparados?
- ☐ ¿La pregunta de investigación está clara?
- ☐ ¿El modelo es apropiado para el tipo de datos?
- ☐ ¿Los supuestos del modelo se cumplen?
- ☐ ¿Se reportan intervalos de confianza, no solo valores p?
- ☐ ¿Se calculan y reportan tamaños de efecto?
- ☐ ¿Se consideran explicaciones alternativas?
- ☐ ¿El análisis es reproducible?
- ☐ ¿Se documentan limitaciones?
- ☐ ¿Las visualizaciones son claras e informativas?

11.5 Recomendaciones para adaptar a nuevos datasets

11.5.1 Lista de verificación al usar nuevos datos

1. **Entender el contexto**
    - ¿Cuál es la fuente de los datos?

- ¿Cómo se recopilaron?
- ¿Cuál es la unidad de análisis?
- ¿Hay documentación o libro de códigos?

2. **Exploración inicial**
   - Dimensiones (filas, columnas)
   - Tipos de variables
   - Valores faltantes
   - Valores atípicos
   - Distribuciones
3. **Definir preguntas**
   - ¿Qué se quiere predecir o explicar?
   - ¿Qué variables son relevantes?
   - ¿Hay hipótesis a priori?
4. **Seleccionar método apropiado**
   - Tipo de variable respuesta (continua, binaria, conteo, etc.)
   - Estructura de datos (independientes, agrupados, temporales)
   - Supuestos del método
5. **Implementar con easystats**
   - Usar funciones consistentes
   - Verificar supuestos
   - Interpretar en contexto
   - Validar resultados

11.5.2 Template de análisis reutilizable

Listing 98: Script template para nuevos análisis

```
# ============================================
# ANÁLISIS ESTADÍSTICO CON EASYSTATS
# Dataset: [NOMBRE]
# Fecha: [FECHA]
# Analista: [NOMBRE]
# ============================================

# 1. CONFIGURACIÓN ----------------------------
library(easystats)
library(tidyverse)

```

```r
# Configurar opciones
options(scipen = 999, digits = 3)

# 2. CARGAR DATOS ------------------------------
datos_raw <- read.csv("ruta/al/archivo.csv")

# 3. EXPLORACIÓN INICIAL -----------------------
glimpse(datos_raw)
summary(datos_raw)
vis_miss(datos_raw)

# 4. LIMPIEZA Y PREPARACIÓN --------------------
datos <- datos_raw %>%
  # Filtros
  filter(...) %>%
  # Transformaciones
  mutate(...) %>%
  # Conversiones de tipo
  mutate(across(c(...), as.factor))

# 5. ANÁLISIS DESCRIPTIVO ----------------------
datos %>%
  group_by(...) %>%
  summarise(...)

# 6. MODELAMIENTO ------------------------------
modelo <- lm(y ~ x1 + x2, data = datos)

# 7. EVALUACIÓN --------------------------------
# Parámetros
parameters(modelo)

# Desempeño
performance(modelo)

# Supuestos
check_model(modelo)

# Tamaños de efecto
effectsize(modelo)

# 8. VISUALIZACIÓN -----------------------------
plot(parameters(modelo))
plot(estimate_means(modelo, at = "x1"))

# 9. REPORTE -----------------------------------
report(modelo)

# 10. GUARDAR RESULTADOS -----------------------
```

```r
saveRDS(modelo, "resultados/modelo.rds")
ggsave("resultados/figuras/grafico.png", width = 10, height = 6, dpi = 300)

# 11. INFORMACIÓN DE SESIÓN --------------------
sessionInfo()
```

12 Conclusiones

12.1 Resumen del manual

Este manual ha cubierto el uso completo del ecosistema easystats para análisis estadístico aplicado, utilizando datos reales de la Encuesta de Intenciones de Siembra de la Región Cusco.

Conceptos clave aprendidos:

1. **Ecosistema easystats**: Una colección integrada de paquetes que facilita todo el flujo de análisis estadístico, desde el modelamiento hasta el reporte.
2. **Flujo de análisis sistemático**:
   - Preparar datos (limpieza, transformación)
   - Ajustar modelos apropiados
   - Extraer parámetros (`parameters`)
   - Evaluar desempeño (`performance`)
   - Verificar supuestos (`check_model`)
   - Calcular tamaños de efecto (`effectsize`)
   - Visualizar (`see`)
   - Reportar (`report`)
3. **Métodos estadísticos aplicados**:
   - Regresión lineal (variables continuas)
   - Regresión logística (variables binarias)
   - Comparación de grupos (t-tests, ANOVA)
   - Modelos con interacciones
   - Modelos de efectos mixtos (datos jerárquicos)
4. **Buenas prácticas**:
   - Verificar supuestos siempre
   - Reportar tamaños de efecto, no solo p-valores
   - Documentar y hacer análisis reproducibles
   - Visualizar resultados efectivamente
   - Interpretar en contexto del problema

12.2 Ventajas de easystats

Para principiantes:

- Sintaxis consistente y predecible
- Automatización de tareas complejas
- Reportes en lenguaje natural
- Visualizaciones profesionales con mínimo código
- Documentación extensa y ejemplos

Para usuarios avanzados:

- Flexibilidad y personalización
- Integración con otros paquetes de R
- Soporte para cientos de tipos de modelos
- Implementación de métodos estadísticos modernos
- Código abierto y extensible

12.3 Próximos pasos

Para continuar desarrollando habilidades en análisis estadístico con R y easystats:

12.3.1 Práctica

- Aplicar los métodos a sus propios datos
- Explorar otros datasets de Datos Abiertos del Gobierno del Perú
- Replicar análisis de artículos científicos
- Participar en comunidades de R (R-Ladies, useR!, etc.)

12.3.2 Profundización

- Modelos más avanzados (GAM, machine learning, etc.)
- Análisis bayesianos con `bayestestR`
- Series temporales y análisis espacial
- Meta-análisis y síntesis de evidencia

12.4 Recursos adicionales

12.4.1 Documentación oficial de easystats

- Sitio web: https://easystats.github.io/easystats/
- GitHub: https://github.com/easystats
- Viñetas de cada paquete (muy detalladas)
- Blog con tutoriales: https://easystats.github.io/blog/

12.4.2 Libros recomendados

- *R for Data Science* - Hadley Wickham & Garrett Grolemund
- *Statistical Rethinking* - Richard McElreath
- *An Introduction to Statistical Learning* - James et al.
- *Regression and Other Stories* - Gelman, Hill & Vehtari
- *Data Analysis Using Regression and Multilevel/Hierarchical Models* - Gelman & Hill

12.4.3 Cursos en línea

- DataCamp: Cursos de R y estadística
- Coursera: Especialización en Data Science (Johns Hopkins)
- edX: Estadística y R (Harvard)
- RStudio Education: https://education.rstudio.com/

12.4.4 Comunidades y foros

- RStudio Community: https://community.rstudio.com/
- Stack Overflow (etiqueta [r])
- Cross Validated (Stack Exchange para estadística)
- Twitter: #rstats, #easystats
- R-Ladies: https://rladies.org/

12.4.5 Datos abiertos en Perú

- Portal de Datos Abiertos: https://www.datosabiertos.gob.pe/
- INEI (Instituto Nacional de Estadística): https://www.inei.gob.pe/
- MIDAGRI (Ministerio de Desarrollo Agrario): https://www.midagri.gob.pe/
- SENAMHI (Servicio Nacional de Meteorología): https://www.senamhi.gob.pe/

12.5 Palabras finales

El análisis estadístico es una habilidad fundamental en la era de los datos. El ecosistema easystats democratiza el acceso a métodos estadísticos sofisticados, permitiendo que investigadores, estudiantes y profesionales realicen análisis rigurosos y reproducibles sin necesidad de ser expertos en programación o estadística avanzada.

Principios para recordar:

1. **Entender antes de aplicar**: Los métodos estadísticos son herramientas, no recetas. Comprender los supuestos y limitaciones es crucial.
2. **Simplicidad y parsimonia**: Comenzar con modelos simples y añadir complejidad solo cuando esté justificado.
3. **Transparencia y reproducibilidad**: Documentar todo el proceso analítico. La ciencia se basa en la replicabilidad.
4. **Significancia estadística $\neq$ importancia práctica**: Los valores p indican evidencia contra la hipótesis nula, pero los tamaños de efecto indican magnitud práctica.
5. **Comunicación clara**: Los resultados estadísticos deben ser comprensibles para la audiencia objetivo, no solo para estadísticos.
6. **Ética en análisis**: Reportar todos los análisis realizados, no solo los "significativos". Reconocer limitaciones y sesgos potenciales.
7. **Aprendizaje continuo**: La estadística y sus métodos evolucionan constantemente. Mantenerse actualizado es esencial.

Este manual es solo el comienzo de su viaje en el análisis estadístico con R y easystats. La práctica constante, la curiosidad y la disposición para aprender de los errores son las claves para desarrollar experticia. Los datos abiertos del Gobierno del Perú ofrecen innumerables oportunidades para aplicar estos conocimientos a problemas reales que afectan a millones de personas.

¡Buena suerte en sus análisis futuros!

13 Referencias

13.1 Publicaciones de easystats

- Lüdecke, D., Ben-Shachar, M.S., Patil, I., Waggoner, P., & Makowski, D. (2021). *performance: An R Package for Assessment, Comparison and Testing of Statistical Models*. Journal of Open Source Software, 6(60), 3139. https://doi.org/10.21105/joss.03139
- Makowski, D., Ben-Shachar, M.S., & Lüdecke, D. (2019). *bayestestR: Describing Effects and their Uncertainty, Existence and Significance within the Bayesian Framework*. Journal of Open Source Software, 4(40), 1541. https://doi.org/10.21105/joss.01541

- Ben-Shachar, M.S., Lüdecke, D., & Makowski, D. (2020). *effectsize: Estimation of Effect Size Indices and Standardized Parameters*. Journal of Open Source Software, 5(56), 2815. https://doi.org/10.21105/joss.02815
- Lüdecke, D., Waggoner, P., & Makowski, D. (2019). *insight: A Unified Interface to Access Information from Model Objects in R*. Journal of Open Source Software, 4(38), 1412. https://doi.org/10.21105/joss.01412

13.2 Libros y artículos de referencia

- Gelman, A., & Hill, J. (2006). *Data Analysis Using Regression and Multilevel/Hierarchical Models*. Cambridge University Press.
- Field, A., Miles, J., & Field, Z. (2012). *Discovering Statistics Using R*. Sage Publications.
- Wickham, H., & Grolemund, G. (2017). *R for Data Science*. O'Reilly Media. https://r4ds.had.co.nz/
- Cohen, J. (1988). *Statistical Power Analysis for the Behavioral Sciences* (2nd ed.). Lawrence Erlbaum Associates.

13.3 Recursos en línea

- Easystats website: https://easystats.github.io/easystats/
- The R Graph Gallery: https://r-graph-gallery.com/
- STHDA (Statistical tools for high-throughput data analysis): http://www.sthda.com/
- Datos Abiertos del Gobierno del Perú: https://www.datosabiertos.gob.pe/

A Instalación y configuración

A.1 Instalación de R y RStudio

1. Descargar R desde CRAN: https://cran.r-project.org/
2. Descargar RStudio Desktop: https://www.rstudio.com/products/rstudio/download/
3. Instalar ambos programas siguiendo las instrucciones

A.2 Instalación de easystats

Listing 99: Instalación completa del ecosistema

```r
# Instalar desde CRAN
install.packages("easystats")

# Alternativamente, instalar versión de desarrollo desde GitHub
# install.packages("remotes")
# remotes::install_github("easystats/easystats")

# Cargar el ecosistema
library(easystats)

# Verificar paquetes cargados
easystats::easystats_packages()
```

A.3 Paquetes complementarios recomendados

Listing 100: Otros paquetes útiles

```r
# Manipulación de datos
install.packages("tidyverse")

# Visualización avanzada
install.packages("ggpubr")
install.packages("patchwork")

# Modelos mixtos
install.packages("lme4")
install.packages("lmerTest")

# Análisis de valores faltantes
install.packages("naniar")
install.packages("mice")

# Tablas
install.packages("kableExtra")
install.packages("flextable")

# Reportes
install.packages("rmarkdown")
install.packages("bookdown")
```

B Glosario de términos estadísticos

AIC (Akaike Information Criterion) Criterio de información para comparación de modelos. Menor es mejor.

ANOVA (Analysis of Variance) Análisis de varianza, prueba para comparar medias de múltiples grupos.

BIC (Bayesian Information Criterion) Similar a AIC pero con mayor penalización por complejidad.

BLUE (Best Linear Unbiased Estimator) Mejor estimador lineal insesgado.

BLUP (Best Linear Unbiased Predictor) Predictor lineal óptimo en modelos mixtos.

Coeficiente de determinación (R^2) Proporción de varianza explicada por el modelo.

d de Cohen Medida de tamaño de efecto para diferencias entre grupos.

Efecto aleatorio Variación entre grupos que se modela como una distribución.

Efecto fijo Parámetro del modelo que se estima directamente.

Efecto simple Efecto de un factor en un nivel específico de otro factor.

Eta cuadrado (^{2}) Tamaño de efecto para ANOVA, proporción de varianza explicada.

Heteroscedasticidad Varianza no constante de los residuos.

Homoscedasticidad Varianza constante de los residuos (supuesto de regresión).

ICC (Intraclass Correlation) Proporción de varianza entre grupos en modelos mixtos.

Intervalo de confianza Rango de valores plausibles para un parámetro.

Multicolinealidad Correlación alta entre predictores.

Odds Ratio (OR) Razón de momios, medida de asociación en regresión logística.

Pseudo-R^2 Análogos al R^2 para modelos no lineales.

REML (Restricted Maximum Likelihood) Método de estimación para modelos mixtos.

RMSE (Root Mean Square Error) Raíz del error cuadrático medio.

Tamaño de efecto Magnitud estandarizada de un efecto estadístico.

Valor p Probabilidad de observar los datos (o más extremos) si la hipótesis nula es cierta.

VIF (Variance Inflation Factor) Medida de multicolinealidad.

C Atajos de teclado útiles en RStudio

Table 8: Atajos de teclado en RStudio

Atajo	**Acción**
Ctrl + Enter	Ejecutar línea/selección
Ctrl + Shift + M	Insertar pipe (%¿%)
Ctrl + Shift + C	Comentar/descomentar
Alt + -	Insertar operador de asignación (¡-)
Ctrl + Shift + K	Compilar documento R Markdown
Ctrl + Shift + F10	Reiniciar sesión de R
Tab	Autocompletar código
Ctrl + 1	Ir a editor de código
Ctrl + 2	Ir a consola
Ctrl + L	Limpiar consola

— Fin del Manual —