**R**

**Contenido**

[Básicos 2](#_Toc183187932)

[Abrir base de datos 2](#_Toc183187933)

[Unir bases de datos 2](#_Toc183187934)

[Guardar una base de datos 3](#_Toc183187935)

[Limpieza de base de datos y variables 4](#_Toc183187936)

[Edición de la base de datos 4](#_Toc183187937)

[Ver las variables 4](#_Toc183187938)

[Eliminar variables 4](#_Toc183187939)

[Filtrar bases de datos 4](#_Toc183187940)

[Añadir nuevas variables 5](#_Toc183187941)

[Recodificar variables 5](#_Toc183187942)

[Estadísticos descriptivos e inferenciales 7](#_Toc183187943)

[Estadísticos descriptivos 7](#_Toc183187944)

[Tablas de frecuencias 7](#_Toc183187945)

[Covarianza y correlación 8](#_Toc183187946)

[Regresiones 8](#_Toc183187947)

[Regresión logística 9](#_Toc183187948)

[Modelos GAMs 10](#_Toc183187949)

[Gráficos 11](#_Toc183187950)

[Paquete ggplot2 11](#_Toc183187951)

[Gráficos animados 13](#_Toc183187952)

[Tipos de gráficos 13](#_Toc183187953)

[Ejemplos de Gráficos 15](#_Toc183187954)

[Gráfico de líneas (evolución temporal variables) 15](#_Toc183187955)

[Gráfico de cajas 16](#_Toc183187956)

[Gráfico de barras para tres variables 17](#_Toc183187957)

[Gráfico de líneas relleno 18](#_Toc183187958)

[Gráfico de barras con error sobre la media 19](#_Toc183187959)

[Diagrama de violines (parecido al de cajas) 20](#_Toc183187960)

[Otras cosas 21](#_Toc183187961)

[Transformar tabla de transferencia de votos para grafico de transferencia 21](#_Toc183187962)

[Añadir nuevas fuentes de texto 21](#_Toc183187963)

[Usar datos CIS si da problemas 21](#_Toc183187964)

[Crear una función propia 21](#_Toc183187965)

[Crear y editar tablas 22](#_Toc183187966)

[Imputar datos perdidos a una base de datos 22](#_Toc183187967)

[Árboles de decisión 22](#_Toc183187968)

Básicos

Escribir en la parte de los comandos (como la // en stata): se usan las **#**

Si se ponen 4# delante y detrás de una palabra/frase, r establece que es un nuevo apartado dentro de los comandos.

Texto

Descripción generada automáticamente

Pulsar ctrl intro sirve para ejecutar los comandos

**Ayuda comandos**

Dos opciones:

* ?? antes del comando
* help(comando/librería)

**Buscar la dirección de un archivo en el ordenador**

file.choose()

Abrir base de datos

Lo más sencillo es abrirlas dándole a la opción de Import Dataset. Antes de abrirlas, R muestra el comando que se usaría para abrir esa misma base.

Ese comando se copia cuando se abre por primera vez y así no tienes que andar buscando la base de datos siempre, ejecutas y ya.

Eliminar una base de datos

rm(nombre)

Unir bases de datos

1. Forma 1

dplyr::bind\_rows(x,y) 🡪 junta todos los datos de las dos

dplyr::union(x,y) 🡪 las junta ignorando los datos que estén en las dos (aparecen solo una vez, no dos repetidos como en el otro comando)

1. Forma 2

Para bases de datos que tienen el mismo número de casos, y al menos una de las variables iguales.

combined\_data <- full\_join(parot\_combined, paroh\_combined, by = "TIME") %>%

full\_join(parom\_combined, by = "TIME")

Guardar una base de datos

Como Excel

install.packages("writexl")

library(writexl)

write\_xlsx(base\_datos, "nombre\_base\_datos.xlsx")

Como .dta (Stata)

library (heaven)

write\_dta(base\_datos, "nombre\_base\_datos.dta")

Limpieza de base de datos y variables

Edición de la base de datos

Ver las variables

* names(base\_datos) #Saca una lista con todas las variables de la base
* list(datos$variable) # Para conocer las etiquetas de cada valor de la variable
* print(datos$variable) # Es lo mismo que list
* table(datos$variable) # Para conocer la distribución de cada valor de la variable (similar al tab en stata pero con cifras absolutas)

Eliminar variables

1. datos <- subset(datos, select = -variable)
2. library(dplyr)

datos <- datos %>% select(-variable)

Mantener solo unas variables en la base (keep if en stata)

1. Crear una lista con el conjunto de todas las variables que se quieran mantener.
2. base\_ datos <- base\_datos[, conjunto\_variables, drop=FALSE]

variables\_a\_mantener <- c("educacion", "edad", "ideología", "renta", "año", "pais")

datos <- datos[, variables\_a\_mantener, drop = FALSE]

Filtrar bases de datos

Diagrama

Descripción generada automáticamentefilter(basedatos, condiciones)

Se usan operadores lógicos (==,>, >=, <, <=, !=)

Si se ponen varias condiciones seguidas por comas, R asume que se tienen que dar todas a la vez. Para precisar más se usan: & (*and*), | (*or*), ! (*not* or *negation*), and xor() (*exactly or*).

Ej.: filter(flights, month == 11 | month == 12)

Para filtrar varias veces una misma variable y no tener que repetirla todo el rato como en el ejemplo se usa **%in% c(**1, 2, …). Ej.: filter(flights, month %in% c(11, 12))

El comando filter ignora los valores perdidos, por lo que si se quieren incluir hay que ponerlo específicamente.

El comando se puede usar para que en limpiar una base de datos dejando solo los valores de una variable. Por ejemplo, para que solo se muestren los datos de un determinado conjunto de países, y no de todos los que aporta la base original:

datos <- datos %>%

filter(pais %in% c(620, 724, 208, 752, 528, 276, 40, 348, 203, 100, 380))

Eliminar un valor de una variable

Para filtrar una base de datos quitando un valor de una variable (p.ej. quitar todos los casos donde edad=0) hay dos opciones:

1. datos <- subset(datos, edad != 0)
2. datos <- datos %>%

filter(edad != 0)

Si se quieren filtrar varios valores de una misma variable:

datos <- datos %>%

filter(!(genero %in% c("Mujer", "Otro")))

También se pueden filtrar varias variables a la vez:

datos <- datos %>%

filter(edad != 0, genero!=1)

Valores perdidos

Se representan con **NA.** na.rm = TRUE 🡪 elimina los valores perdidos del resultado que se quiera obtener (con cualquier comando).

Añadir nuevas variables

1. mutate

mutate() #Se pueden realizar diferentes operaciones dentro de mutate (+, -, >, != …)

mutate(año=2002+(essround-1)\*2)

1. <-

datos$nuevavar <- valor\_nueva\_var

Cambiar el nombre de las variables

1. Rename

datos <- datos %>%

rename(nombre\_nuevo = nombre\_viejo)

1. Mutate

Se usa también el comando mutate.

datos <- datos %>%

mutate(pais = country, democracia = v234))

(En realidad se están creando nuevas variables, no cambiándoles el nombre. Las viejas deberán eliminarse luego)

Recodificar variables

Es lo mismo escribir datos <- datos %>% que datos$var <- comando(datos$var)

comandos

1. ifelse

datos$variablecambio <- ifelse(condiciones, nuevo\_valor, datos$variablecambio)

datos$incumbent1 <- ifelse(datos$voto == 1 & datos$año == 2015, 4, datos$incumbent1)

1. recode

datos$variablecambio <- recode(datos$variablecambio, “*cambios\_en\_valores*”)

1. mutate (de la librería tidyverse)

Combinado con el ifelse es lo que mejor permite recodificar varios valores de una variable, además de añadirle etiquetas a estos.

datos <- datos %>%

mutate(tecno = ifelse(tecno %in% c(1, 2), 1, ifelse(tecno %in% c(3, 4), 0, NA))) %>%

mutate(tecno = factor(tecno, levels = c(0, 1), labels = c("No tecnocracia", "Apoya tecnocracia")))

**ifelse(...)**:permite realizar una operación condicional. Tiene tres argumentos: la condición, el valor a asignar si la condición es verdadera y el valor a asignar si la condición es falsa.

**- tecno %in% c(1, 2)**: Esta es la condición que se evalúa. Significa que estamos verificando si los valores en la variable "tecno" están dentro del conjunto {1, 2}.

Si la condición es verdadera (es decir, si "tecno" está en {1, 2}), se asigna el valor 1.

Si la condición es falsa (es decir, si "tecno" no está en {1, 2}), se evalúa otro **ifelse()**.

**- tecno %in% c(3, 4)**: Esta es la nueva condición que se verifica si la primera condición es falsa. Significa que estamos verificando si los valores en "tecno" están dentro del conjunto {3, 4}.

Si esta segunda condición es verdadera (es decir, si "tecno" está en {3, 4}), se asigna el valor 0.

Si la segunda condición también es falsa (es decir, si "tecno" no está en {3, 4}), se asigna NA (valor perdido).

En resumen, este comando recodifica la variable "tecno" de la siguiente manera: si "tecno" es 1 o 2, se asigna 1; si "tecno" es 3 o 4, se asigna 0; y si "tecno" no está en ninguno de estos conjuntos, se asigna NA (valor perdido).

Para enviar **valores a perdidos**, el procedimiento es el mismo, pero poniendo **NA** en lugar del valor numérico en la condición.

datos <- datos %>%

mutate(incumbent1 = ifelse((voto1 == 1 & año == 2015)|(voto1 == 1 & año == 2016)|(voto1 == 2 & año == 20191)|(voto1 == 2 & año == 20192), 1, 0)) %>%

mutate(incumbent1 = factor(incumbent1, levels = c(0,1), labels = c("No votó al titular", "Sí votó al titular")))

Cambiar el nombre de una variable

Para cambiar el nombre de una variable, aparte de crear una nueva variable con el nuevo nombre (mutate(variable\_nuevo\_nombre=Variable\_original)) también se puede usar el comando rename(variable\_nuevo\_nombre=Variable\_original)

Añadir etiquetas a los valores de las variables

factor(variable, levels = c(“a”, “b”, “c”…), labels = c("AA", "BB", "CC"…))

datos <- datos %>%

mutate(tecno = factor(tecno, levels = c(0, 1), labels = c("No tecnocracia", "Apoya tecnocracia")))

Transformar una variable categórica en una nominal

Para ver cómo esta codificada la variable, se usa el comando class(variable)

base\_datos$variable <- as.numeric(base\_datos$variable)

Se pueden transformar varias variables a la vez del siguiente modo:

datos<-datos

mutate\_at(vars(año,ideol,estudios,ingresos,salud,felicidad,edad,interes),as.numeric)

En las funciones del tidyverse, si se quieren realizar varias funciones a la vez, no se pone + al final de cada comando, sino %>%

datos <- datos %>%

mutate(tecno = ifelse(tecno %in% c(1, 2), 1, ifelse(tecno %in% c(3, 4), 0, NA))) **%>%**

mutate(tecno = factor(tecno, levels = c(0, 1), labels = c("No tecnocracia", "Apoya tecnocracia")))

Estadísticos descriptivos e inferenciales

Estadísticos descriptivos

* Media: mean(var)
* Mediana: median(var)
* Cuartiles: quantile(var)
* Percentiles: se establece a mano en el comando anterior. Ej.: quantile(var, probs=0,55).
* Varianza: var(variable)
* Desviación standard: sd(variable)
* **Conjunto de estadísticos descriptivos**: summary(variable) #Media, mediana, cuartiles, mínimo y máximo.

Tabla-resumen con los descriptivos de todas las variables de un dataframe

library(stargazer)

stargazer(datos, digits=2, type="text") #Para ver la tabla en la consola

stargazer(datos, digits=2, type="html",out="descriptivos.html") #Para descargar la tabla

#Para que el comando funcione todas las variables de la base de datos deben transformarse en numéricas: datos<- as.data.frame(sapply(datos, as.numeric))

Tablas de frecuencias

Para una variable

* table(variable) #Muestra los valores absolutos de cada categoría de la variable
* table(var)/sum(table(var)) # Para las frecuencias relativas
* Para variables discretas, se establecen los límites de los rangos que se quieren mostrar en la tabla previamente:

1. cut(variable,breaks=c(0,3,5,10,20))

table(cut(variable,breaks=c(0,3,5,10,20)))

1. transform(table(cut(variable,breaks=c(0,3,5,10,20))))

Para dos variables

* table(var1,var2) #Para los valores absolutos de los cruces
* round(prop.table(table(mani,genero),2),4)\*100
* El primer número (table(var),2)) es para especificar si se quiere el porcentaje de filas o de columnas (1 para el de filas y 2 para el de columnas).
* El segundo número sirve para definir el número de decimales de la tabla.

Contraste de hipótesis

Estadístico de chi cuadrado: chisq.test(table(var1,var2))

ANOVA

fm=aov(lm(variable1~variable2))

summary(fm)

**Supuestos del modelo:**

**1**-Distribución normal

#Ejemplos con gráficos

boxplot(fm$residuals, horizontal=TRUE)

hist(fm$residuals)

qqnorm(fm$residuals)

qqline(fm$residuals)

#Comprobaición con el test de shafiro

shapiro.test(residuals(fm))

**2**-contraste de homoscedasticidad

bartlett.test(SALARY~MGT)

**3**-independencia de los residuos

plot(fm$residuals) #la distribución de los residuos no es aleatoria

Covarianza y correlación

* Covarianza: con(variable1,variable2)
* Correlación: cor(variable1,variable2)

Regresiones

nombre\_reg <- lm(dependiente~independiente1+independiente2, data=base\_datos)

#Para establecer la categoría de referencia en el caso de las variables categóricas se usa el siguiente comando ANTES de ejecutar la regresión:

datos$var<-relevel(datos$var,ref="Categoría1")

Sumario del modelo (los resultados de la regresión)

summary(nombe\_reg)

Interacciones

Se añade un asterisco\* entre las dos variables para las que se quiere la interacción

glm(mani~edad\*genero+año,family="binomial",data=datos)

Tabla de resultados

library(stargazer)

stargazer(reg1, reg2, reg3, type="text", title=”Título tabla”, align=TRUE, no.space=TRUE)

* type=”formato” #El formato de salida de la tabla. Usamos text para salida en consola, html para páginas web y latex.
* out=”nombre\_tabla.formato” #Para descargar la tabla
* aling=T #Para centrar los números de la tabla
* no.space=T #Para reducir el espacio entre cada cifra
* keep.stat="n", #Restringe la cantidad de estadísticos del modelo colocados en la parte inferior de la tabla.
* single.row=FALSE #Coloca el error estándar de la regresión debajo o al lado del estadístico estimado del modelo.
* digits=número (establece el número de decimales que aparecen en la tabla)
* covariate.labels=c("Etiqueta var1"," Etiqueta var2"," Etiqueta var3"…), cambia el nombre de las variables que aparecen en la tabla
* omit.stat=c("LL","ser","f"), omite solo algunos de los estadísticos de la parte interior(lo contrario de keep.stat, que estableces los que sí quieres)
* style=’estilo’, cambia el estilo de la tabla
* p= , para cambiar los p-valores preestablecidos

stargazer(r1,r2,r3,r4,r5, type="html",out="regresiones1.html",

align=TRUE, no.space=TRUE,

column.labels = c("Modelo 1","Modelo 2","Modelo 3","Modelo 4","Modelo 5"))

Gráfico de coeficientes

library(coefplot)

coefplot(reg1) +

theme\_minimal() +

labs(title="Estimación de coeficientes con error estandar",

x="Estimación",

y="Variable",

caption="Elaboración propia con datos del WVS")

Gráfico de puntos con línea de la regresión

1. Hacer el modelo de la regresión.
2. Guardar los componentes de la recta:

B0 <-coef(modelo\_regresión)[1] # Guarda en el nuevo objeto el componente 1 que se muestra en el coeficiente, que se corresponde con B0

B1<-coef(modelo\_regresión)[2]

1. Hacer el gráfico:

ggplot(datos, aes(x,y))+

geom\_point()+

geom\_abline(intercept=B0, slope=B1)

Regresión logística

glm(dependiente~independiente1+independiente2, family=binomial(), data=base\_datos)

Para saber en la variable dependiente cual es el 0 y cual el 1 se usa el comando: levels(base\_datos$variable)

El segundo argumento de la función, family=binomial(), especifica la función de probabilidad que utilizaremos. Para modelos logit es una función binomial. Dentro de los paréntesis se puede especificar la función de enlace. Para la familia de distribuciones binomial glm() por defecto usa una función logit. Si nos interesa una función probit deberíamos especificar link=probit.

En R las variables categóricas que utilizamos como dependientes en un modelo logit corresponden al tipo de datos factor.

***Testear el modelo y uso para hacer predicciones: en lo del curso***

Modelos GAMs

**Modelo:**

library(mgcv)

prueba<-gam(mani~s(nacimiento, by=genero)+edad+ideol+interes+estudios+hijos+thogar+social+salud+felicidad,data=datos,family=binomial)

**Gráfico:**

plot(prueba, pages = 1, trans = plogis,

shift = coef(prueba)[1],

seWithMean = TRUE,

xlab = "Edad", ylab = "Probabilidad de manifestación",

main = "Relación entre Edad y Probabilidad de Manifestación",

rug=TRUE, #Para añadir todas las líneas de años en el eje x

shade=TRUE)

abline(v = c(1918,1947,1961,1975,1983,1988,2000), col = "red", lty = 2)

Gráficos

Edición de graficas

*(Para los gráficos básicos de R, no ggplot)*

Al comando normal se pueden añadir las siguientes opciones:

* main = 'Nombre Título Gráfica' --> añade un título a la grafica
* xlab = '' --> cambia el nombre que aparece en los ejes. Si no se escribe nada entre las comillas, aparecerá en blanco
* ylab = '' --> los mismo para y
* col = 'nombredelcolor' --> cambia el color de la grafica
* ...

Unir gráficos

1. library(gridExtra)
2. nombre\_gráfico1 <- comandos gráfico1
3. nombre\_gráfico2 <- comandos gráfico2
4. nombre\_gráfico… <- comandos gráfico…
5. grid.arrange(nombre\_gráfico1, nombre\_gráfico2, nombre\_gráfico…, opciones)

Opciones:

* nrow= número 🡪 indica el número de filas (todos los gráficos en 1 fila, en 2…)
* ncol= número 🡪 indica el número de columnas
* widths = c(número tamaño1, número tamaño2…) 🡪 cambia el tamaño de cada uno de los gráficos

grid.arrange(g1, g2, nrow=1, widths = c(1.3, 2)) 🡪 dos gráficos en una sola fila, el primero más pequeño que el segundo

grid.arrange(España, Italia, Holanda, Austria, ncol = 2) 🡪 cuatro gráficos en dos columnas

Paquete ggplot2

El funcionamiento básico es la superposición de distintas capas que formaran el gráfico final, y que pueden incluir varios gráficos a la vez, junto a otros ajustos. Cuanto más arriba en el comando se escriba una función más abajo estará en el gráfico (lo que se añade después en el comando se añade encima en el gráfico).

ggplot(data = nombredatos)

gráfico1(mapping = aes (x = valorenx, y = valoreny)) +

gráfico2(mapping = aes (x = valorenx, y = valoreny)) +

ajustes\_gráfico +

…

Temas de los gráficos

Algunos de los temas disponibles son:

* **theme\_gray()**: El tema predeterminado de ggplot2 con un fondo gris y líneas de cuadrícula blancas.
* **theme\_bw()**: Un tema clásico con fondo blanco y líneas negras.
* **theme\_linedraw()**: Un tema con solo líneas negras de varios anchos sobre fondos blancos, similar a un dibujo a línea.
* **theme\_light()**: Un tema similar a theme\_linedraw() pero con líneas y ejes grises claros para dirigir más atención hacia los datos.
* **theme\_dark()**: El primo oscuro de theme\_light(), con tamaños de línea similares pero un fondo oscuro. Útil para hacer que las líneas delgadas y coloreadas resalten.
* **theme\_minimal()**: Un tema minimalista sin anotaciones de fondo.
* **theme\_classic()**: Un tema clásico con líneas de ejes x e y y sin líneas de cuadrícula.
* **theme\_void()**: Un tema completamente vacío.

(ejemplos de cada uno aquí: [Complete themes — ggtheme • ggplot2 (tidyverse.org)](https://ggplot2.tidyverse.org/reference/ggtheme.html#ref-examples))

Texto en los gráficos

+ theme(plot.title = element\_text(hjust = 0.5)) 🡪 ajusta el título del gráfico al centro

("\*\*Design:\*\* Gilbert Fontana<br>","\*\*Data:\*\* OECD, 2022") 🡪 los dos asteriscos sirven para ponerlo en negrita, <br> indica que el siguiente texto va en la siguiente línea.

Ajustar la base de datos para el gráfico

En algunas ocasiones, el formato de los datos puede no ser el adecuado requerido por el comando del gráfico, por lo que estos se deben transformar.

datos <- datos %>%

pivot\_longer(cols = -TIME, names\_to = "Country", values\_to = "Value")

Por ejemplo, si quieres hacer una gráfica con la evolución temporal de una variable en varios países, necesitas una variable llamada países con los distintos casos, otra con los años correspondientes a cada país, y otra con la variable de estudio, y una cifra correspondiente a cada país en cada caso.

Pero si en la base de datos lo que tienes es una variable para cada país donde aparecen sus datos, debes transformarla para que tenga el formato anterior.

Se debe pasar del primer formato al segundo:

Tabla

Descripción generada automáticamente Imagen que contiene Tabla

Descripción generada automáticamente

Gráficos animados

Paquete gganimate. Hay que instalar también el paquete gifski

Se añade al comando de ggplot como si fuese una grafica más que se suma:

ggplotdata = nombredatos,

mapping = aes (x = valorenx, y = valoreny)) +

tipográfico() +

transition\_tipodefunción()

Tipos de gráficos

Gráfico de puntos

plot(x = varx,

y = vary)

Histogramas

1. R basico: más sencillo. Mejor para solo observar datos

hist(dataframe$columna)

Al comando básico se puede añadir:

hist(dataframe$columna,

breaks = número, --> el número indica cada cuanto se establece la separación por números en el eje x (p.ej. si se pone breaks = 50 en el eje x aparecerán indicaciones en 50, 100, 150...)

breaks = seq(número1, número2, númerocambio), --> en este caso el comando es más preciso. El priemer número que se escribe indica donde empieza el histograma, el segundo dónde termina y el tercero cada cuantos números se establece un nuevo indicador)

1. ggplot2. Usarlo si el gráfico se va a presentar ya que quedan más bonitos:

ggplot(data = nombrebasedatos,

mapping = aes(x = variabledelgráfico)) +

geom\_histogram(bins = número) --> indica el número de columnas que aparecen en el histograma

Este comando da muchas opciones extra, como por ejemplo superponer dos histogramas y que cada uno sea de un color para distinguirlo mejor (mostrar un histograma de un valor en función de una variable dicotómica para ver las diferencias).

Gráfico de barras

1. R básico:

barplot(table(basedatos$variable),

opciones)

1. ggplot2:

ggplot(data = basedatos,

mapping = aes(x = factor(variable))) +

geom\_bar()

Gráfica puntos regresión

Orange %>%

ggplot(aes(x = age,

y = circumference)) +

geom\_point() +

geom\_abline(intercept = 17.3997,

slope = 0.1068,

col = 'blue')

lm(circumference ~ age, data = Orange)

#Esta fórmula saca la pendiente (slope) y el intercept de la regresión --> para ponerlo en el gráfico de antes

Para hacer un gráfico de barras en función de una segunda o tercera variable (p.ej. edad en función del género) hay dos opciones, dejando las barras apiladas o una al lado de la otra. Para que sean barras separadas, en el comando se debe añadir la opción position=”dodge”.

ggplot(d\_brecha\_gen) +

geom\_bar(aes(x = gen, y = porcentaje, fill = genero), stat = "identity", position = "dodge")

Ejemplos de Gráficos

Gráfico de líneas (evolución temporal variables)

ggplot(data = gasto\_desempleo, aes(x = TIME, y = Value, color = Country, group = Country)) +

geom\_line() +

geom\_point() +

labs(

x = "Año",

y = "% PIB",

color = "Países",

caption = "Fuente: elaboración propia a partir de datos de Eurostat") +

scale\_color\_manual(values = c(Spain = "blue", Italy = "red", Netherlands = "green", Austria = "purple")) +

theme\_minimal() +

ggtitle("Gasto gubernamental en prestaciones por desempleo")+

scale\_x\_continuous(

limits = c(2000, 2019),

breaks = c(2000, 2004, 2008, 2012, 2016, 2019),

labels = scales::number\_format(accuracy = 1)) +

geom\_vline(xintercept = 2008, linetype = "dashed", color = "black") +

theme(plot.title = element\_text(hjust = 0.5))

Gráfico, Gráfico de líneas

Descripción generada automáticamente

geom\_line 🡪 para hacer la gráfica de líneas

geom\_point 🡪 para añadir los puntos a cada caso

labs 🡪 añadir etiquetas (x para la del eje x (Año), y para la del y (%PIB), color para el índice, caption para el texto inferior (Fuente))

scale\_color\_manual 🡪 para cambiar el color de la gráfica asignado a cada país

theme\_minimal 🡪 establecer el formato del gráfico (estético)

ggtitle 🡪 añade el título del gráfico

scale\_x\_continuous 🡪 cambia las cifras que se muestran en el eje x

limits = c(2000, 2019) 🡪 solo se muestra el rango que va del 2000 al 2019

breaks = c(2000, 2004, 2008, 2012, 2016, 2019) 🡪 solo aparecen estos números

labels = scales::number\_format(accuracy = 1))

geom\_vline(xintercept = 2008, linetype = "dashed", color = "black") 🡪 añade la línea de puntos en el año 2008

theme(plot.title = element\_text(hjust = 0.5)) 🡪 ajusta el título del gráfico al centro

Gráfico de cajas

ggplot(data = desempleo\_juvenil, aes(x = Country, y = Value, fill=Country)) +

geom\_boxplot(color="black",

alpha=0.3) +

ggtitle("Diagrama de cajas y evolución temporal del paro juvenil")+

theme\_classic()+

geom\_jitter(color="black", size=0.4, alpha=0.3)+

labs(

x = "País",

y = "Desempleo juvenil (%)") +

stat\_boxplot(geom = "errorbar",

width = 0.25) +

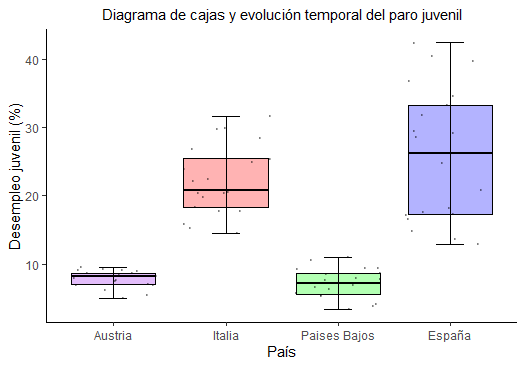
theme(legend.position = "none",

plot.title = element\_text(hjust = 0.5, size = 11))+

scale\_x\_discrete(labels = c(Spain = "España", Italy = "Italia", Netherlands = "Paises Bajos")) +

scale\_fill\_manual(

values = c(Spain = "blue", Italy = "red", Austria = "purple", Netherlands = "green"))



geom\_boxplot 🡪 comando gráfico cajas

color="black", 🡪 para remarcar de negro el contorno de las cajas

alpha=0.3🡪 para cambiar la transparencia del relleno de las cajas (cuanto más bajo el número, más

transparente el color)

ggtitle 🡪 añade el título del gráfico

theme\_classic 🡪 establecer el formato del gráfico (estético)

geom\_jitter(color="black", size=0.4, alpha=0.3) 🡪 añade un punto para cada dato

labs 🡪 añadir etiquetas de los ejes

stat\_boxplot(geom = "errorbar", width = 0.25) 🡪 añade las líneas horizontales que indican los datos extremos (si no lo pones solo aparecen las verticales)

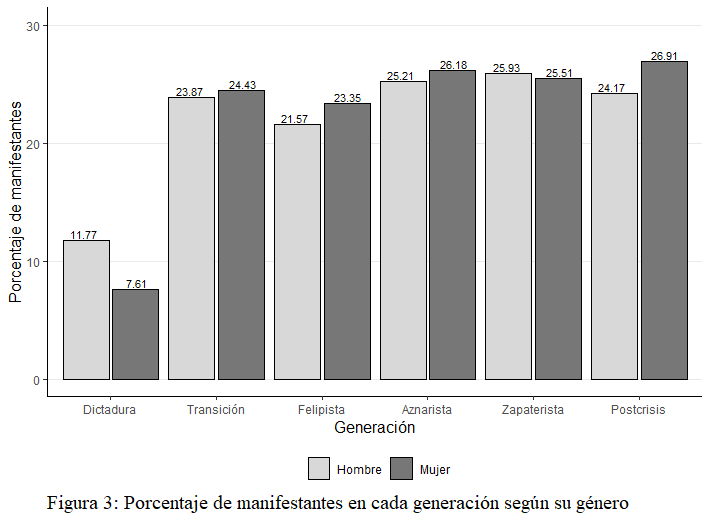
theme(legend position = “none”, 🡪 elimina la leyenda

plot.title = element\_text(hjust = 0.5)) 🡪 ajusta el título del gráfico al centro

scale\_x\_discrete 🡪 cambia el nombre de las etiquetas que se muestran en el eje x

scale\_fill\_manual 🡪 cambia el color de relleno de las cajas

Gráfico de barras para tres variables



ggplot(brecha\_gen, aes(x = gen, y = porcentaje, fill = genero)) +

scale\_y\_continuous(limits=c(0,30))+

geom\_hline(yintercept = c(0, 10, 20,30), color = "lightgrey", alpha = 0.5) +

geom\_bar(stat = "identity", position = position\_dodge(width = 0.94), width = 0.87, colour="black") +

labs(x = "Generación", y = "Porcentaje de manifestantes", fill = "Género", caption = "Figura 3: Porcentaje de manifestantes en cada generación según su género") +

theme\_classic() +

geom\_text(aes(label = porcentaje, y = porcentaje + 1), #Añade el porcentaje sobre las barras

position = position\_dodge(width = 1), vjust = 1.1, size = 3) + #Ajustes del tamaño del porcentaje

theme(plot.caption = element\_text(hjust=0,family="Times New Roman",size=15), #Ajustes del caption

text = element\_text(size = 11), # Tamaño del texto del gráfico

legend.position = "bottom", # Posición de la leyenda

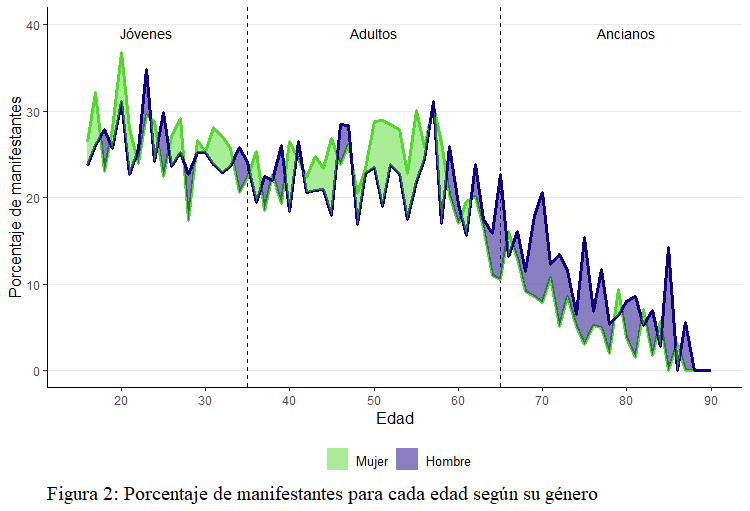
legend.title = element\_blank(), # Elimina el título de la leyenda

legend.text = element\_text(size = 9), # Tamaño del texto de la leyenda

axis.title = element\_text(size = 12)) + # Tamaño del texto de los títulos de los ejes

scale\_fill\_manual(values=c("Hombre"="#d8d8d8", "Mujer"="#777777"))

Gráfico de líneas relleno



library(ggh4x) #Para el gráfico de líneas con relleno

g<-ggplot(data = brecha\_edad, aes(x=edad)) +

scale\_x\_continuous(limits=c(15,90),breaks=c(10,20,30,40,50,60,70,80,90))+

scale\_y\_continuous(limits=c(0,40),breaks=c(0,10,20,30,40))+

geom\_hline(yintercept = c(0, 10, 20,30,40), color = "lightgrey", alpha = 0.5)+

geom\_vline(xintercept=c(35,65), color="black", linetype="dashed")+

theme\_classic()+

labs(x="Edad",y="Porcentaje de manifestantes",caption = "Figura 2: Porcentaje de manifestantes para cada edad según su género" )+

theme(plot.caption = element\_text(hjust=0,family="Times New Roman",size=15), #Ajustes del caption

text = element\_text(size = 11),

legend.position = "bottom",

legend.title = element\_blank(),

legend.text = element\_text(size = 10),

axis.title = element\_text(size = 12)) +

geom\_line(aes(y=Mujer),color="#51DA2C",size=1.15)+

geom\_line(aes(y=Hombre),color="#130081",size=1.15)+

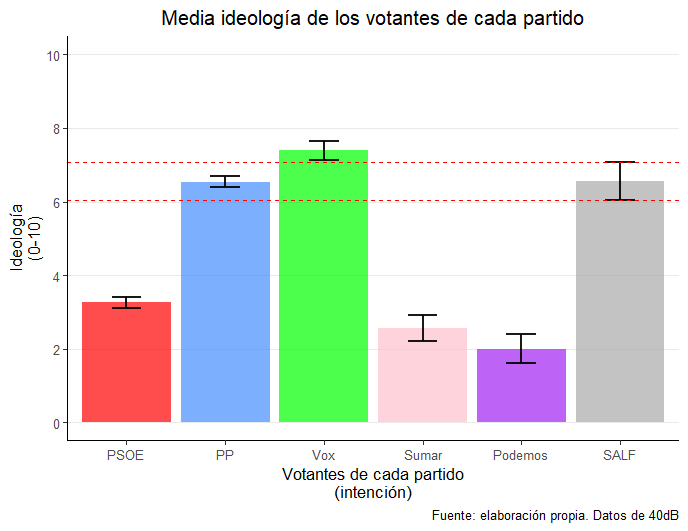
stat\_difference(aes(ymin = Hombre, ymax = Mujer), alpha = 0.5,) + #Este comando y el siguiente son los que se

usan para el relleno

scale\_fill\_manual(values = c("#51DA2C", "#130081"), labels=c("Mujer","Hombre"))

g+annotate("text",x=c(23,50,80), y=c(39,39,39),label=c("Jóvenes", "Adultos", "Ancianos"),size=3.6) #Para añadir el texto sobre el gráfico

Gráfico de barras con error sobre la media



Comandos para calcular el error en la media:

barras<-datos %>%

group\_by(intención) %>%

summarise(

n=n(),

mean=mean(ideol),

sd=sd(ideol)) %>%

mutate(se=sd/sqrt(n)) %>%

mutate(ic=se\*qt((1-0.05)/2 + .5, n-1))

Comandos del gráfico

ggplot(barras)+

theme\_classic()+

scale\_y\_continuous(limits = c(0, 10), breaks = seq(0, 10, by = 2))+

geom\_hline(yintercept = c(0,2,4,6,8,10), color = "lightgrey", alpha = 0.5)+

geom\_bar(aes(x=intención, y=mean, fill=intención), stat="identity",alpha=0.7)+

geom\_errorbar(aes(x=intención, ymin=mean-ic, ymax=mean+ic), width=0.3, colour="black", alpha=0.9, linewidth=0.9)+

labs(title="Media ideología de los votantes de cada partido",x="Votantes de cada partido\n(intención)", y="Ideología\n(0-10)", caption="Fuente: elaboración propia. Datos de 40dB")+

theme(plot.title = element\_text(hjust = 0.5),

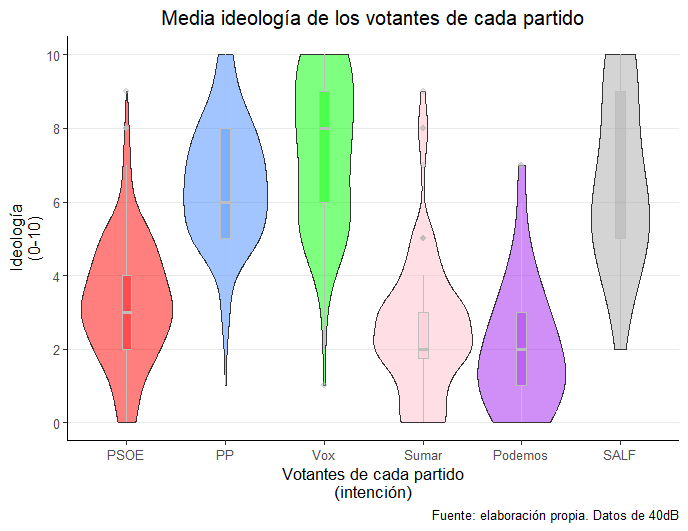
legend.position="none")+

geom\_hline(yintercept = SALF\_min\_ic, color = "red", linetype = "dashed", linewidth = 0.3)+

geom\_hline(yintercept = SALF\_max\_ic, color = "red", linetype = "dashed", linewidth = 0.3)+

scale\_fill\_manual(values = c(PP = "#458CFF", PSOE = "red", Podemos = "purple", Vox = "green", Sumar="pink",SALF="darkgrey"))

Diagrama de violines (parecido al de cajas)



ggplot(datos,aes(x=intención, y=ideol, fill=intención))+

theme\_classic()+

scale\_y\_continuous(limits = c(0, 10), breaks = seq(0, 10, by = 2))+

geom\_hline(yintercept = c(0,2,4,6,8,10), color = "lightgrey", alpha = 0.5)+

geom\_violin(width=1, alpha=0.5, adjust = 1.6)+ #width es para hacer más ancho cada violin, y adjust para lo suaves que son las curvas

geom\_boxplot(width=0.1, color="grey", alpha=0.4) +

scale\_fill\_manual(values = c(PP = "#458CFF", PSOE = "red", Podemos = "purple", Vox = "green", Sumar="pink",SALF="darkgrey"))+

theme(plot.title = element\_text(hjust = 0.5),legend.position="none")+

labs(title="Media ideología de los votantes de cada partido",x="Votantes de cada partido\n(intención)", y="Ideología\n(0-10)", caption="Fuente: elaboración propia. Datos de 40dB")

Otras cosas

Transformar tabla de transferencia de votos para grafico de transferencia

library(tidyverse)

datos <- datos %>%

pivot\_longer(cols = -...1, names\_to = "Voto Segundo Año", values\_to = "Número votantes transferidos")

library(writexl)

write\_xlsx(datos, "datos.xlsx")

Añadir nuevas fuentes de texto

library(extrafont)

loadfonts(device = "win", quiet = TRUE)ç

Usar datos CIS si da problemas

1. Abrir los datos
2. Añadir etiquetas (opcional):

nombrenuevodataset <- as\_factor(dataset)

1. Si no reconoce el nombre de las variables:

library(dplyr)

nuevonombredatos <- as\_tibble(dataset)

Crear una función propia

function(x,y){*parámetros de la función*}

Ejemplos:

analisis1<-function(x,y){

# En esta función vamos a realizar un pequeño análisis estadístico de dos variables

# Vamos a calcular la media, mediana y DT

# También haremos un histograma y un diagrama de caja para cada variable

x.media<-mean(x)

x.mediana<-median(x)

x.dt<-sd(x)

y.media<-mean(y)

y.mediana<-median(y)

y.dt<-sd(y)

# Colocamos estos resultados en una tabla

valores<-c(x.media,x.mediana,x.dt,y.media,y.mediana,y.dt)

etiquetas<-c("x.media","x.mediana","x.dt","y.media","y.mediana","y.dt")

tabla.resumen<-data.frame(medida,resumen)

# Realizamos los gráficos

par(mfrow=c(2,2))

hist(x,main="Histograma de la variable X",col="blue",ylab="frecuencia",xlab="",col.lab="brown",col.axis = "maroon4")

hist(y,main="Histograma de la variable Y",col="cyan",ylab="frecuencia",xlab="",col.lab="brown",col.axis = "maroon4")

boxplot(x,main="Diagrama de caja de X",col="blue",ylab="frecuencia",col.lab="brown",col.axis = "maroon4")

boxplot(y,main="Diagrama de caja de Y",col="cyan",ylab="frecuencia",col.lab="brown",col.axis = "maroon4")

return(tabla=tabla.resumen)

}

Crear y editar tablas

Se usa la librería kableExtra

Imputar datos perdidos a una base de datos

*(está en la presentación del curso)*

Árboles de decisión

https://youtu.be/t0lmtrm\_aa0?si=mqUk2asdYxtc34Fh