Análisis exploratorio de datos (EDA)

Figure 1: Artwork por @allison_horst

Introducción

El análisis exploratorio de datos (conocido como **EDA**, su sigla en inglés) es un enfoque de análisis de datos para resumir y visualizar las características importantes de un conjunto de datos.

John Tukey, estadístico estadounidense, fue el principal propulsor contribuyendo de manera significativa al desarrollo del análisis exploratorio de datos al publicar, en 1977, su libro que lleva ese nombre donde entre otras cosas introdujo el *gráfico boxplot* (diagrama de caja y bigotes).

En términos simples, antes de avanzar con la etapa analítica y de construir modelos estadísticos, es relevante explorar, conocer y describir las variables de interés en nuestra tabla de datos.

Los principales objetivos perseguidos por EDA son:

- Conocer la estructura de la tabla de datos y sus tipos de variable
- Detectar observaciones incompletas (valores missing)
- Conocer la distribución de las variables de interés a partir de:
  - Resumir datos mediante estadísticos
  - Resumir datos mediante gráficos
- Detectar valores atípicos (outlier)

Aclaración: En este documento mostraremos funciones del lenguaje R que se pueden aplicar en este proceso basadas en la filosofía tidyverse. También aplicaremos otros paquetes diseñados para tareas específicas que le serán de mucha utilidad. Esto no quiere decir que no se pueda hacer la misma exploración con funciones del R `base` pero el ecosistema facilita el entendimiento de lo que estamos haciendo.

Presentaremos estas diferentes funciones de distintos paquetes que pueden servir en cada etapa de un EDA. Los paquetes con los que trabajaremos son:

- tidyverse
- skimr
- dlookr
- janitor

Para instalarlos puede copiar y ejecutar el siguiente código:

```
install.packages(c("tidyverse", "skimr", "janitor", "dlookr"))
```

Nota: Algunos paquetes, entre estos `dlookr`, pueden ocasionar un falso positivo en la detección del antivirus durante el proceso de instalación. Sugerimos que desactive momentáneamente su antivirus para instalarlo sin inconvenientes.

Una vez instalados los podemos activar:

```
library(skimr)
library(janitor)
library(dlookr)
library(tidyverse)
```

Cabe aclarar que no existe un solo camino y/o función del lenguaje para obtener la información requerida y que esta selección de paquetes puede cambiarse y ampliarse según la conveniencia del usuario. Es decir, aquellxs estudiantes que ya utilicen R y estén familiarizados con funciones y/o paquetes que realicen la misma tarea pueden seguir usándolos.

Con el fin de ejemplificar este análisis exploratorio vamos a utilizar un archivo con datos ficticios y variables de distinto tipo.

Conocer la estructura de la tabla de datos y sus tipos de variable

El primer paso en la exploración de un conjunto de datos es conocer su estructura y tamaño.

El tamaño está definido por la cantidad de observaciones (filas) y la cantidad de variables (columnas).

Llamamos estructura a la forma en se organizan sus variables, sus tipos de datos y sus categorías/valores.

```
datos <- read_csv2("datos/datos2.txt")
```

La función `glimpse()` del `tidyverse` le da un vistazo a los datos:

```
glimpse(datos)
```

```
Rows: 74
Columns: 7
$ id      <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,…
$ sexo    <chr> "M", "M", "M", "M", "M", "M", "M", "M", NA, "F", "F", "M", "F"…
$ edad    <dbl> 76, 68, 50, 49, 51, 68, 70, 64, 60, 57, 83, 76, 27, 34, 17, 45…
$ peso    <dbl> 71, 71, 79, 71, 87, 75, 80, 83, 69, 73, 60, 70, 648, 718, 61, …
$ talla   <dbl> 167, 164, 164, 164, 1675, 170, 166, 160, 160, 155, 155, 167, 1…
$ trabaja <lgl> FALSE, FALSE, FALSE, TRUE, TRUE, FALSE, NA, TRUE, TRUE, TRUE, …
$ fecha   <date> 2020-10-20, 2020-10-20, 2020-10-20, 2020-11-05, 2020-11-05, 2…
```

Nos informa que la tabla tiene 74 observaciones, 7 variables con su tipo de dato y los primeros valores de cada una al lado.

Los tipos de datos que nos podemos encontrar son:

- **int** (integer): números enteros
- **dbl** (double): números reales
- **lgl** (logical): valores lógicos
- **chr** (character): caracteres (texto)
- **Date**: fechas
- **fct** (factor): factores
- **dttm** (date-time): fechas y horas

Esta exploración inicial de la estructura generalmente viene acompañada por el "diccionario de datos" asociado a la tabla de datos, ya sea que esta tabla provenga de un proyecto de investigación propio (fuente primaria) o producto de una fuente secundaria.

En algunas situaciones el tipo de dato del dato coincidirá con la clasificación de la variable (por ejemplo, que sea numérica -dbl- para variables cuantitativas continuas) pero en otros casos podemos tener variables codificadas donde el dato es numérico pero representa una categoría de una variable cualitativa (por ejemplo, si a una variable de respuesta Si - No, la codificamos como 1 y 0).

Detectar observaciones incompletas (valores missing)

Sabemos que los valores perdidos o faltantes (conocidos en inglés como missing), que se gestionan en R mediante el valor especial reservado `NA`, constituyen un serio problema en nuestras variables de análisis.

Existen numerosos libros sobre como tratarlos y sobre diversos algoritmos de imputación que no vamos a incluir en este curso.

Sólo vamos a enfocarnos en como podemos utilizar algunas funciones de R para detectarlos, contabilizarlos y en algunas situaciones excluirlos.

Cada vez que ejecutemos un `count()` a una variable nos informará, al final de la tabla de salida, la cantidad de valores `NA`.

```
datos |>
  count(trabaja)
```

```
# A tibble: 3 × 2
  trabaja     n
  <lgl>   <int>
1 FALSE      26
2 TRUE       39
3 NA          9
```

Mucho mejor es la función `find_na()` que proviene del paquete `dlookr`:

```
find_na(datos, rate = T)
```

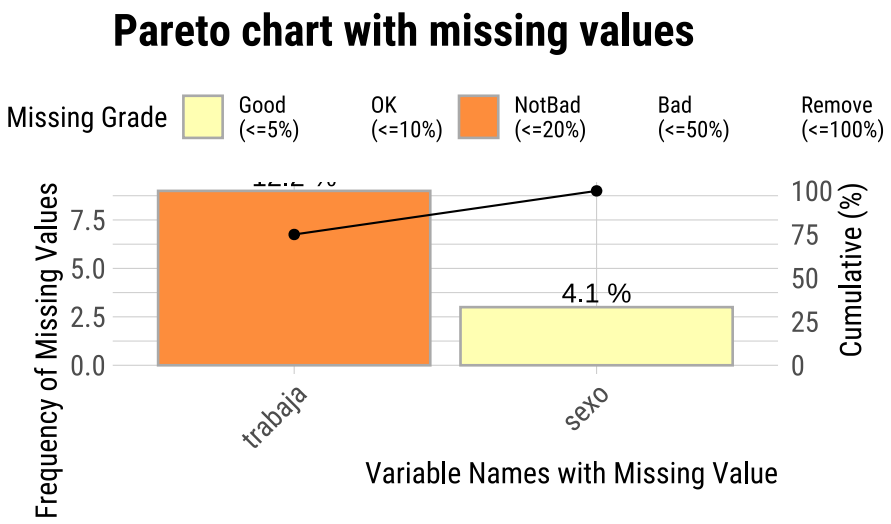
```
     id    sexo    edad    peso   talla trabaja   fecha
  0.000   4.054   0.000   0.000   0.000  12.162   0.000
```

Podemos aplicarla a todo el dataframe y nos dice que porcentaje de valores `NA` hay en cada variable. En este ejemplo la variable sexo tiene alrededor de un 4 % de valores faltantes y trabaja un poco más de 12 %.

Estos porcentajes, siempre y cuando los `NA` sean el resultado de la falta de dato, nos hará decidir si una variable es apropiada para incluirla en un análisis y/o si conviene excluir observaciones con esa situación.

El mismo paquete trae una función gráfica llamada `plot_na_pareto()`:

```
plot_na_pareto(datos, only_na = T)
```



Algo mas completo se logra con `diagnose()` (también de `dlookr`) que diagnostica la calidad de las variables:

```
diagnose(datos)
```

```
# A tibble: 7 × 6
  variables types     missing_count missing_percent unique_count unique_rate
  <chr>     <chr>             <int>           <dbl>        <int>       <dbl>
1 id        numeric               0            0              74      1
2 sexo      character             3            4.05            3      0.0405
3 edad      numeric               0            0              45      0.608
4 peso      numeric               0            0              56      0.757
5 talla     numeric               0            0              38      0.514
6 trabaja   logical               9           12.2             3      0.0405
7 fecha     Date                  0            0              11      0.149
```

Conocer la distribución de las variables de interés

Resumir variables cuantitativas mediante estadísticos

La instalación básica de R tiene incorporadas muchas funciones estadísticas con las cuales calcular medidas resumen de variables cuantitativas que podemos incluir dentro de los resúmenes de tidyverse.

Medidas de tendencia central

Estas medidas son parte de las medidas de posición o localización, pero tiene la intención de resumir la información en torno a un valor central, respecto al cual parece agruparse de un modo más o menos concentrado la distribución de los demás valores.

```
datos |>
  summarise(media = mean(edad), # media
            mediana = median(edad))  # mediana
```

```
# A tibble: 1 × 2
  media mediana
  <dbl>   <dbl>
1  48.1    52.5
```

En cambio, no hay ninguna función base que calcule la moda. Tendremos que escribir una forma de cálculo o bien buscar y activar algún paquete extra de las numerosas librerías que tiene el lenguaje R que la tenga implementada.

Medidas de posición

Las medidas de posición dividen un conjunto de datos en grupos con el mismo número de individuos. Entre los más utilizados tenemos los cuartiles y percentiles.

Por ejemplo, con la función `quantile()`, del paquete `stats`, calculamos los cuartiles Q1 y Q3. Indicamos como argumento los valores 0.25 y 0.75.

```
datos |>
  summarise(cuartil1 = quantile(edad, probs = 0.25),
            cuartil3 = quantile(edad, probs = 0.75))
```

```
# A tibble: 1 × 2
  cuartil1 cuartil3
     <dbl>    <dbl>
1       28       64
```

Si queremos obtener el mínimo y máximo de este conjunto de valores numéricos podemos hacerlo con:

```
datos |>
  summarise(minimo = min(edad),
            maximo = max(edad))
```

```
# A tibble: 1 × 2
  minimo maximo
   <dbl>  <dbl>
1     13     86
```

Medidas de dispersión

Cuando intentamos saber que tan dispersos están los valores o que tan variables son los datos dentro del conjunto de datos, usamos estadísticos de dispersión.

Los clásicos conocidos como la varianza `var()` y el desvío estándar `sd()` se pueden aplicar directamente al `summarise()`:

```
datos |>
  summarise(varianza = var(edad),
            desvio = sd(edad))
```

```
# A tibble: 1 × 2
  varianza desvio
     <dbl>  <dbl>
1     405.   20.1
```

También se puede calcular el rango y el rango intercuartílico.

Para el primero debemos hacer la diferencia entre el máximo y el mínimo, en cambio para el RIC se usa la función `IQR()` que devuelve el rango entre el tercer y el primer cuartil de una distribución:

```
datos |>
  summarise(rango = max(edad)-min(edad),
            ric = IQR(edad))
```

```
# A tibble: 1 × 2
  rango   ric
  <dbl> <dbl>
1    73    36
```

En `dlookr` tenemos la función `describe()` dedicada a describir variables numéricas.

```
describe(datos, -id)
```

```
# A tibble: 3 × 26
  described_variables     n    na  mean    sd se_mean   IQR skewness kurtosis
  <chr>               <int> <int> <dbl> <dbl>   <dbl> <dbl>    <dbl>    <dbl>
1 edad                   74     0  48.1  20.1    2.34  36     -0.211    -1.11
2 peso                   74     0 358.  323.    37.6  626.     0.451    -1.41
3 talla                  74     0 363.  505.    58.7   12.8    2.19      2.90
# ℹ 17 more variables: p00 <dbl>, p01 <dbl>, p05 <dbl>, p10 <dbl>, p20 <dbl>,
#   p25 <dbl>, p30 <dbl>, p40 <dbl>, p50 <dbl>, p60 <dbl>, p70 <dbl>,
#   p75 <dbl>, p80 <dbl>, p90 <dbl>, p95 <dbl>, p99 <dbl>, p100 <dbl>
```

Observemos que se puede aplicar sobre la tabla de datos completa. La propia función seleccionará las variables numéricas, pero con `-id` le estamos indicando que no tome en cuenta la variable id (el identificador es una variable que no tiene sentido analizar estadísticamente).

Los resultados comprenden: observaciones con datos, observaciones con valores `NA`, media, desvío estandar, error de la media, intervalo intercuartílico, medidas de forma como la simetría (*skewness*) y la curtosis (*kurtosis*) y los quintiles (incluye la mediana p50 y los cuartiles p25-p75).

Resumir variables cualitativas mediante estadísticos

Las variables cualitativas o categóricas pueden encontrarse bajo el tipo de dato **character** o **factor**. Ocasionalmente vamos a necesitar que las variables se encuentren bajo este último formato que el lenguaje R reserva para efectuar algunos procedimientos con estas variables.

Frecuencias y tablas de contingencia

Podemos resumir individualmente variables de tipo cualitativo mediante las frecuencias absolutas y relativas de sus categorías. La función `count()` nos muestra el conteo absoluto de las diferentes categorías de la variable:

```
datos |>
  count(sexo)
```

```
# A tibble: 3 × 2
  sexo      n
  <chr> <int>
1 F        27
2 M        44
3 <NA>      3
```

En la salida, además de las categorías explícitas, aparece el valor `NA` con un conteo de 3 observaciones.

Tener en cuenta o no los valores faltantes es una decisión propia del que conduce el análisis y puede cambiar dependiendo de los objetivos buscados.

Una forma de evitarlos con `count()` es utilizar luego la función `drop_na()`:

```
datos |>
  count(sexo) |>
  drop_na()  # saltea los valores NA
```

```
# A tibble: 2 × 2
  sexo      n
  <chr> <int>
1 F        27
2 M        44
```

Obtenemos frecuencias relativas porcentuales así:

```
datos |>
  count(sexo) |>
  drop_na() |>
  mutate(porc = 100*n/sum(n))
```

```
# A tibble: 2 × 3
  sexo      n  porc
  <chr> <int> <dbl>
1 F        27  38.0
2 M        44  62.0
```

Redondeamos el valor del porcentaje con `round()`:

```
datos |>
  count(sexo) |>
  drop_na() |>
  mutate(porc = 100*n/sum(n),
         porc = round(porc, digits = 2))
```

```
# A tibble: 2 × 3
  sexo      n  porc
  <chr> <int> <dbl>
1 F        27  38.0
2 M        44  62.0
```

Una opción más completa es utilizar funciones del paquete `janitor` como `tabyl()`:

```
datos |>
  tabyl(sexo)
```

```
 sexo  n    percent valid_percent
    F 27 0.36486486     0.3802817
    M 44 0.59459459     0.6197183
 <NA>  3 0.04054054            NA
```

Calcula las frecuencias relativas incluyendo y no incluyendo los valores `NA` (porcentaje de valores válidos).

Podemos modificar sus argumentos y asociar otras funciones del paquete mediante tuberías para obtener mejores resultados (es compatible con `tidyverse`).

```
datos |>
  tabyl(sexo, show_na = F) |> # anulamos valores na
  adorn_totals(where = "row") |> # agregamos totales
  adorn_pct_formatting(digits = 2) # porcentaje con dos decimales
```

```
  sexo  n percent
     F 27  38.03%
     M 44  61.97%
 Total 71 100.00%
```

Tablas de contingencia

La forma más adecuada de describir la relación entre dos variables categóricas es a partir de la construcción de una tabla de contingencia. Para ello se introduce en cada fila de la tabla las categorías de una de las variables y las categorías de la otra variable se asocian a cada una de las columnas de la tabla, en cada celda de la tabla aparecerá el número de observaciones correspondientes a la combinación oportuna de ambas variables.

Con la misma función `tabyl()` se puede realizar una tabla de contingencia, incluyendo a la variable *trabaja* (aunque tenga formato lógico puede utilizarse igual si conceptualmente la variable es categórica):

```
datos |>
  tabyl(sexo, trabaja)
```

```
 sexo FALSE TRUE NA_
    F     8   15   4
    M    17   22   5
 <NA>     1    2   0
```

Recordemos que en orden dentro de los paréntesis de la función es igual al de los índices, el primer argumento es la variable que aparecerá en las filas y el segundo la variable de las columnas. Por ese motivo, en la tabla de contingencia absoluta tenemos el *sexo* en las filas y a *trabaja* en las columnas.

Se puede mejorar sin valores `NA` y agregando totales:

```
datos |>
  tabyl(sexo, trabaja, show_na = F) |>
  adorn_totals(where = "row")
```

```
  sexo FALSE TRUE
     F     8   15
```

```
    M    17   22
Total    25   37
```

Calculamos frecuencias relativas porcentuales por columna:

```
datos |>
  tabyl(sexo, trabaja, show_na = F) |>
  adorn_totals(where = "row") |>
  adorn_percentages(denominator = "col") |> #  % por columna
  adorn_pct_formatting(digits = 2) # redondea con 2 decimales
```

```
  sexo   FALSE    TRUE
     F  32.00%  40.54%
     M  68.00%  59.46%
 Total 100.00% 100.00%
```

Calculamos frecuencias relativas porcentuales por fila:

```
datos |>
  tabyl(sexo, trabaja, show_na = F) |>
  adorn_totals(where = "col") |>
  adorn_percentages(denominator = "row") |> #  % por fila
  adorn_pct_formatting(digits = 2) # redondea con 2 decimales
```

```
 sexo  FALSE   TRUE   Total
    F 34.78% 65.22% 100.00%
    M 43.59% 56.41% 100.00%
```

Cambiando el argumento **denominator** de `adorn_percentages()` a *"all"* se calculan relativas al total.

```
datos |>
  tabyl(sexo, trabaja, show_na = F) |>
  adorn_totals(where = "col") |>
  adorn_percentages(denominator = "all") |> #  % por total
  adorn_pct_formatting(digits = 2) # redondea con 2 decimales
```

```
 sexo  FALSE   TRUE  Total
    F 12.90% 24.19% 37.10%
    M 27.42% 35.48% 62.90%
```

Explorar variables mediante gráficos

Uno de los aportes más relevantes de Tukey respecto al análisis es el uso de los gráficos cómo método exploratorio.

Los gráficos básicos más útiles que muestran la distribución univariada de variables que podemos hacer con R son:

- **variables cuantitativas**: histogramas, densidad, *boxplot* y *violinplot*.
- **variables cualitativas**: barras

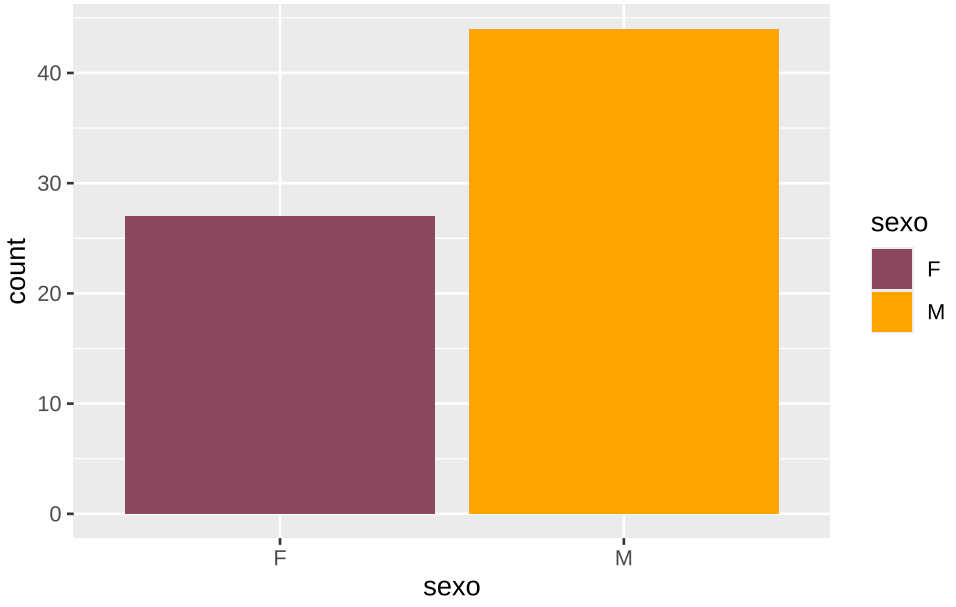
Cuando interviene más de una variable aparecen comúnmente los puntos, las líneas y los gráficos de mosaico.

El lenguaje R soporta una serie de sistemas gráficos asociados a paquetes como `graphics`, `lattice`, `ggplot2`, etc. que sirven de base incluso para otros paquetes con funciones más específicas. Actualmente el estándar gráfico en R es `ggplot2`.

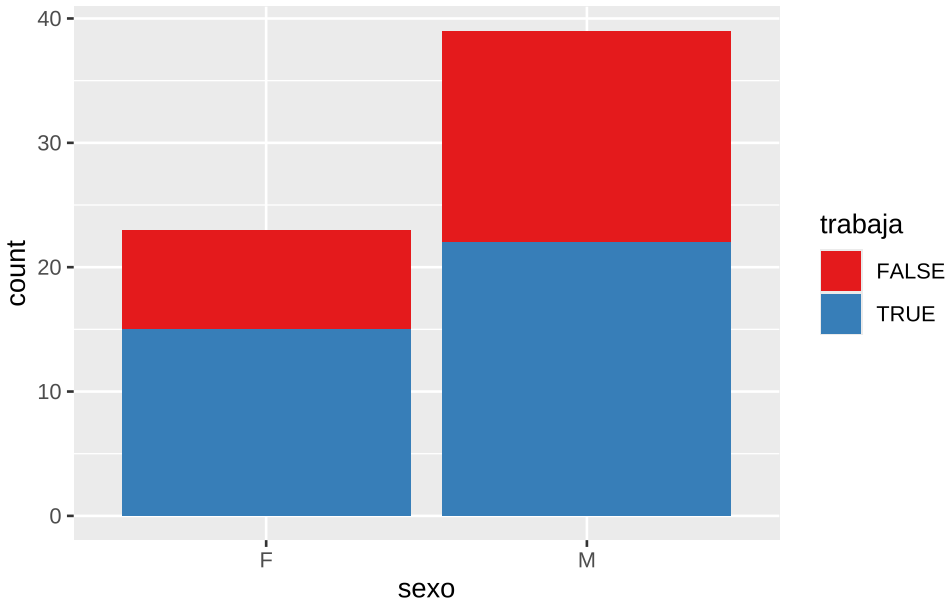
En el documento referido a tidyverse explicamos el funcionamiento de `ggplot2` y sus capas gráficas.

Ahora sólo ejecutaremos los distintos elementos geométricos para representar los diferentes gráficos mencionados.

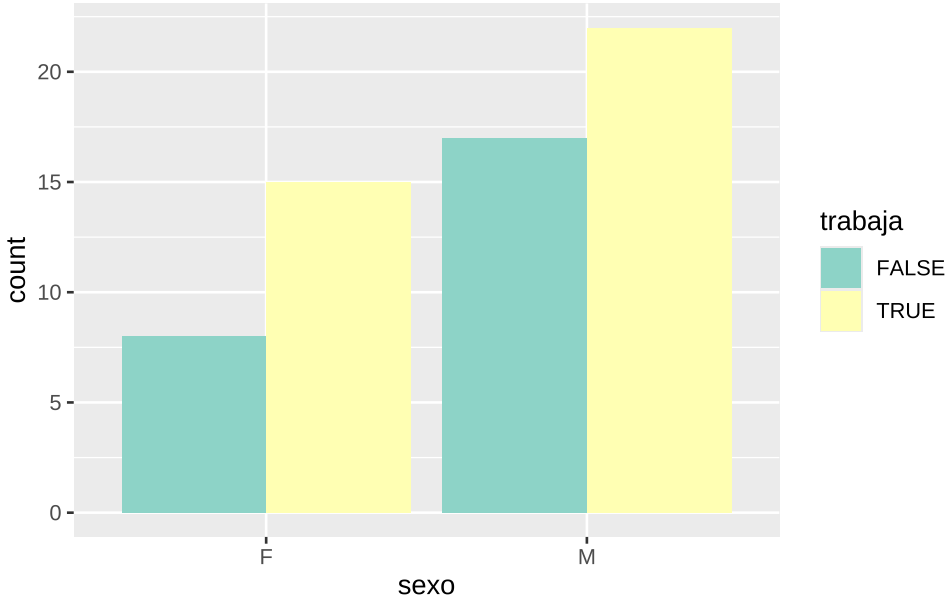
Barras en univariado



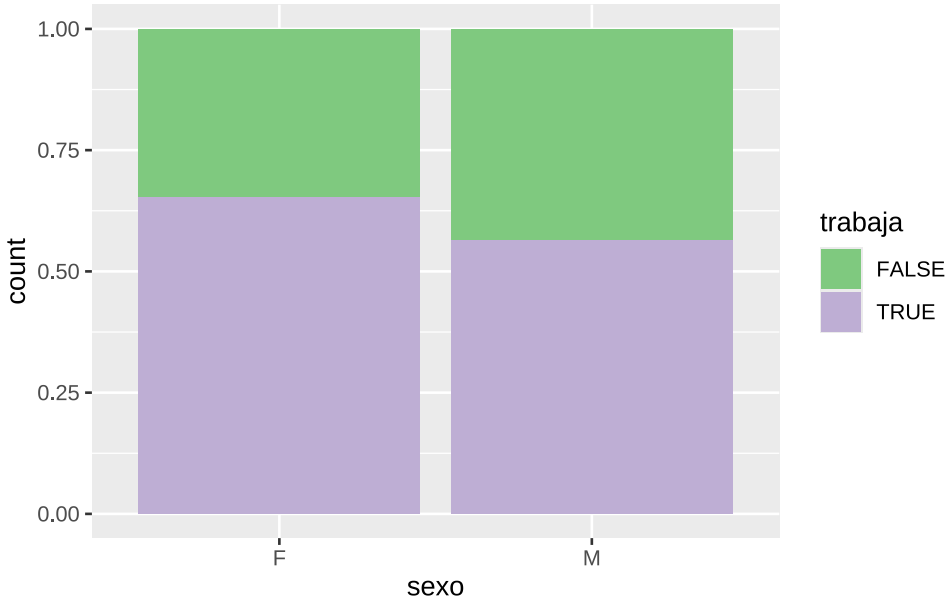
Barras en bivariado - posición stack



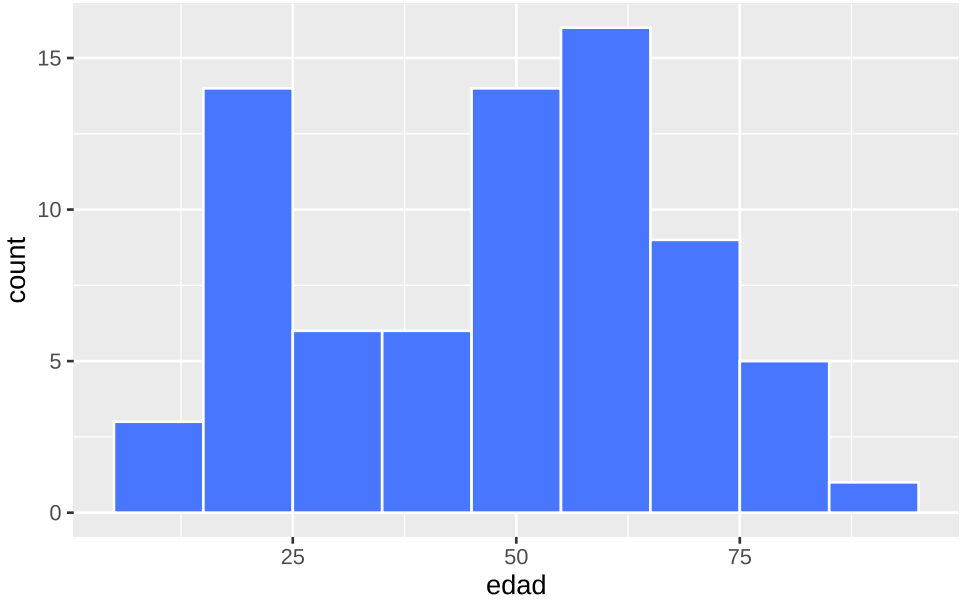
Barras en bivariado - posición dodge



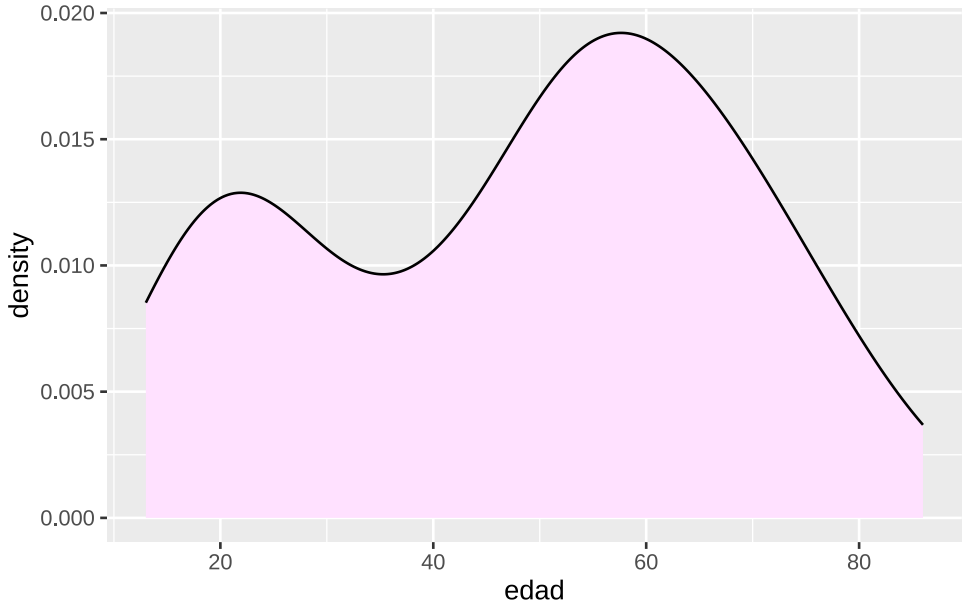
Barras en bivariado - posición fill



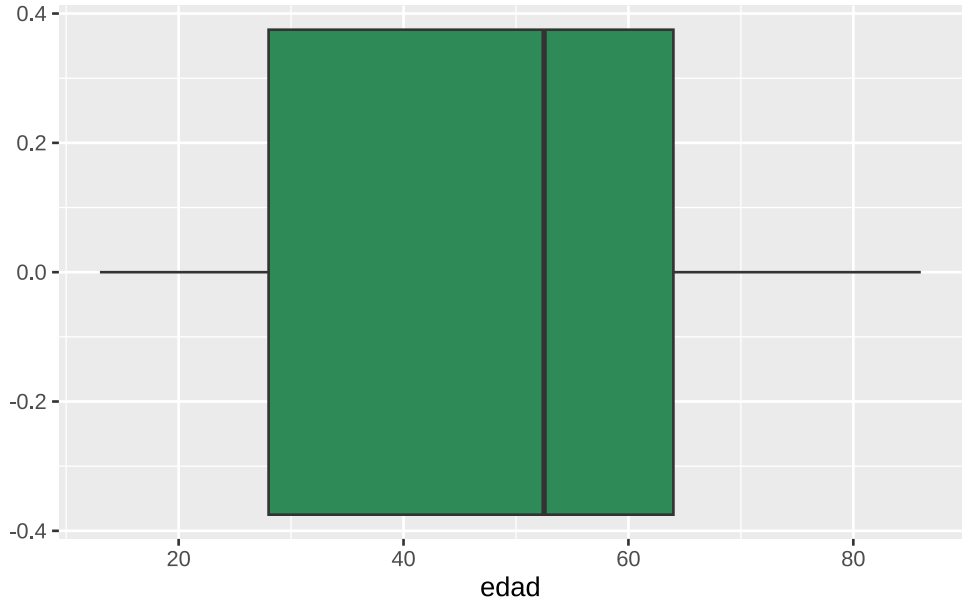
Histograma



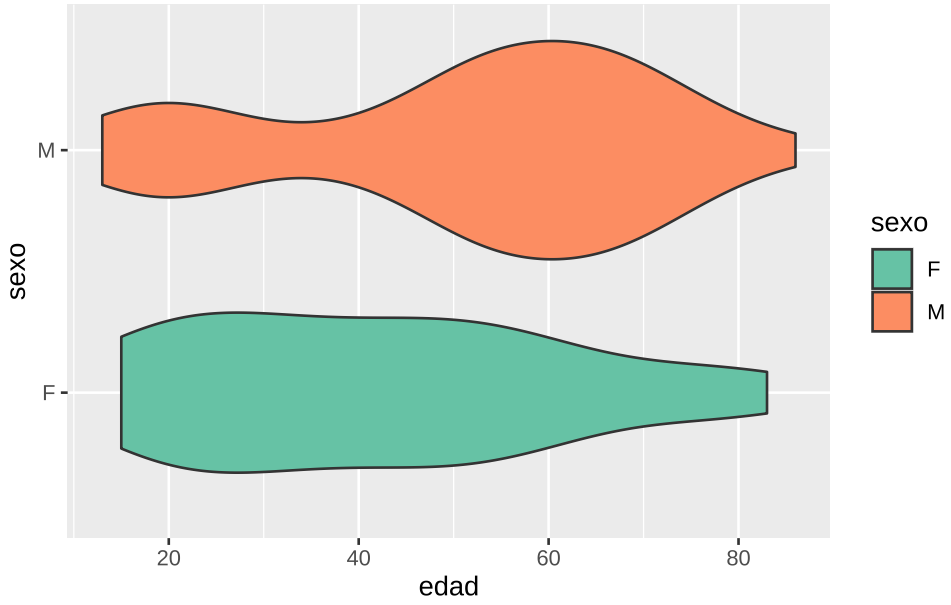
Densidad



Boxplot



Violinplot



Q-Q Plot

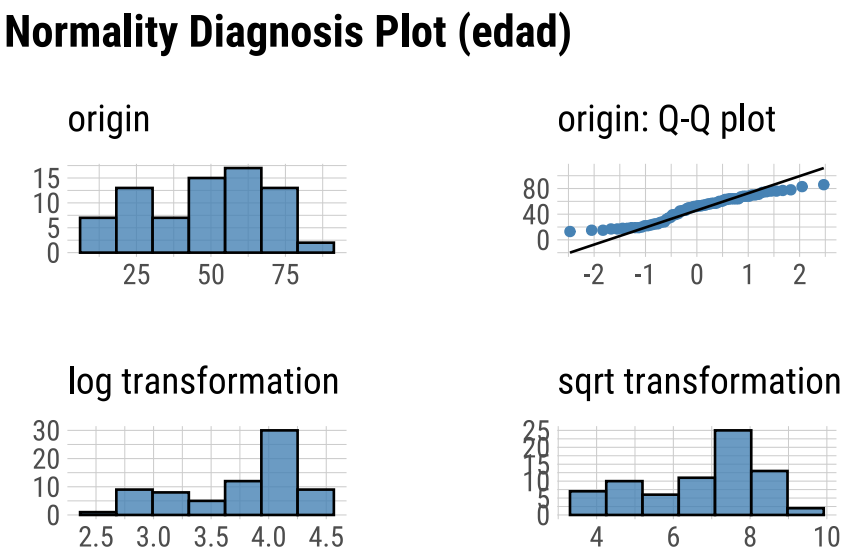
Los Q-Q Plot (Cuantil-Cuantil) son gráficos especiales que permiten observar cuan cerca está la distribución de un conjunto de datos a alguna distribución ideal o comparar la distribución de dos conjuntos de datos.

Suele usarse como método gráfico para analizar "normalidad", es decir cuanto se asemeja la distribución de la variable a la distribución normal ("gaussiana").

La función `plot_normality()` de `dlookr` muestra un diagnóstico gráfico de normalidad de una variable usando histogramas y Q-Q plot. Además muestra otros histogramas con conversiones de datos (logarítmico y raíz cuadrada por defecto, pero también "Box-Cox" y otras)

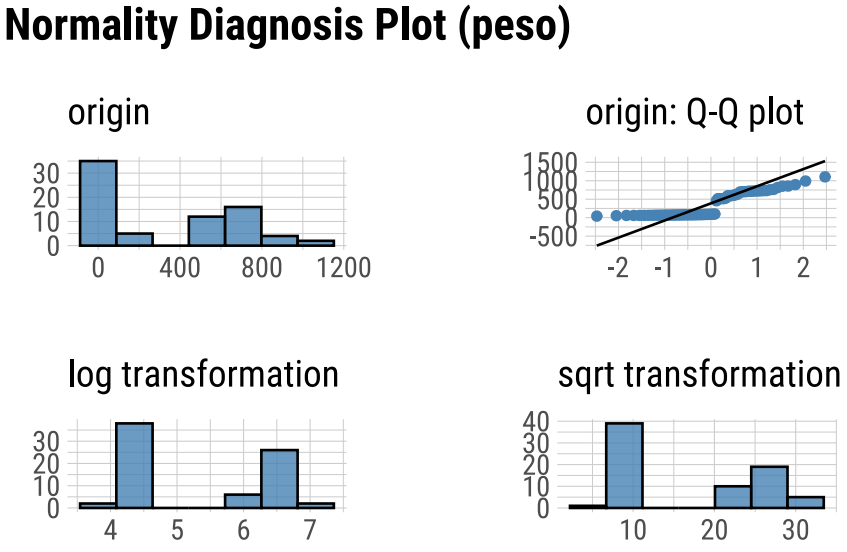
Sobre la variable **edad**:

```
datos |>
  plot_normality(edad)
```



Sobre la variable **peso**:

```
datos |>
  plot_normality(peso)
```



En la comparación podemos decir que los puntos se desarrollan mas cercanos a la línea teórica normal para la variable peso que para la variable edad.

Estos gráficos siempre se analizan acompañados por *test de bondad de ajuste específicos de normalidad* que veremos en la unidad 2.

Detectar valores atípicos (*outlier*)

Un valor atípico (en inglés *outlier*) es una observación que está numéricamente distante del resto de los datos.

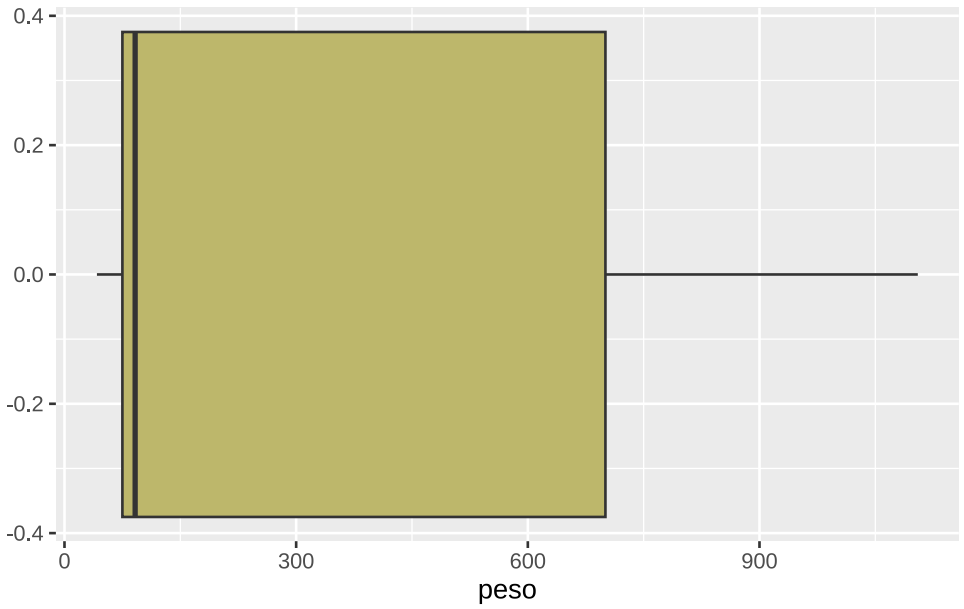
Los *outliers* pueden deberse a:

- Errores de carga o procedimiento (se deben reparar)
- Valores extremos posibles (hay que evaluarlos)
- Acontecimientos extraordinarios o causas desconocidas (suelen eliminarse)

Estos valores desproporcionados pueden conducir a interpretar erróneamente algunos estadísticos como la media, ya que los distorsionan.

Una forma habitual de detección es mediante los gráficos *boxplot*.

Veamos el ejemplo con la variable peso donde existe un valor considerado atípico en el extremo superior de la distribución (punto rojo):



Coincide con el valor máximo de 1105 kgrs.

Dentro de los paquetes incluidos en este documento la función `diagnose_outlier()` de `dlookr` esta pensada para la detección de datos atípicos.

```
diagnose_outlier(datos)
```

```
# A tibble: 4 × 6
  variables outliers_cnt outliers_ratio outliers_mean with_mean without_mean
  <chr>            <int>          <dbl>         <dbl>     <dbl>        <dbl>
1 id                   0            0            NaN      37.5         37.5
2 edad                 0            0            NaN      48.1         48.1
3 peso                 0            0            NaN     358.         358.
4 talla               10           13.5         1630     363.         165.
```

Se puede aplicar a todo el conjunto de datos y nos devuelve una tabla con la cantidad de outliers detectados por variable, la proporción, la media considerando estos valores y la media sin considerarlos.

En función de estos dos estadísticos se puede comparar el efecto de los valores atípicos en la media.

skimr

Finalmente una función interesante por lo sencillo de su uso es `skim()` de `skimr`.

```
skim(datos)
```

Name	datos
Number of rows	74
Number of columns	7
_____	
Column type frequency:	
character	1
Date	1
logical	1
numeric	4
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
sexo	3	0.96	1	1	0	2	0

Variable type: Date

skim_variable	n_missing	complete_rate	min	max	median	n_unique
fecha	0	1	2020-10-20	2020-12-15	2020-11-11	11

Variable type: logical

skim_variable	n_missing	complete_rate	mean	count
trabaja	9	0.88	0.6	TRU: 39, FAL: 26

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
id	0	1	37.50	21.51	1	19.25	37.5	55.75	74	▇▇▇▇▇
edad	0	1	48.07	20.12	13	28.00	52.5	64.00	86	▇▃▇▇▃
peso	0	1	358.05	323.31	42	75.00	91.5	700.50	1105	▇▁▁▂▁
talla	0	1	363.09	504.93	148	161.00	166.0	173.75	1745	▇▁▁▁▁

Muestra en una tabla características y estadísticos descriptivos de las variables del dataframe y se puede combinar dentro de estructuras `tidyverse` con tuberías.

```
datos |>
  drop_na(sexo) |>
  group_by(sexo) |>
  select(where(is.numeric), -id) |>  # solo variables numéricas - id
  skim()
```

Name	select(...)
Number of rows	71
Number of columns	4
_____	
Column type frequency:	
numeric	3
_____	
Group variables	sexo

Variable type: numeric

skim_variable	sexo	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
edad	F	0	1	41.89	19.64	15	26.00	39.0	54.50	83	
edad	M	0	1	51.59	19.56	13	40.75	55.5	64.75	86	
peso	F	0	1	426.78	300.85	42	70.50	516.0	677.00	856	
peso	M	0	1	306.95	331.27	64	78.00	86.5	647.00	1105	
talla	F	0	1	421.74	562.57	148	155.50	161.0	167.50	1625	
talla	M	0	1	340.77	485.79	158	164.00	168.5	177.25	1745	

En este ejemplo, mostramos resultados de variables numéricas menos de id agrupados por sexo (sin considerar valores `NA` en las categorías de sexo)