Clase 3: Análisis exploratorio

Erick Cuevas Fernández

Analisis exploratorio de datos

Analizar datos nos permite tomar mejores decisiones para indagar en eventos de causalidad entre las variables, asi como poder ver la correlación que pudiese existir. Este proceso consiste en la identificación de valores faltantes, imputación, limpieza, y formato.

Missin Values

Son representados como **NA** (*Not Available*) en R. Es necesario identificarlos ya que interfieren con operaciones exclusivas para cada clase de objeto, ejemplo, ecuaciones o funciones númericas; si hacemos una operación con un **NA** presente, por default el resultado será **NA**.

Para evitar los **NAs**, se pueden usar las siguientes estrategias:

Nota: para estos ejemplos solo tomaremos 10 columnas del dataset que cargamos

```
starwars <- dplyr::starwars
starwars$films <- NULL
starwars$vehicles <- NULL
starwars$starships <- NULL

is.na(head(starwars))
```

```
##       name height  mass hair_color skin_color eye_color birth_year gender
## [1,] FALSE  FALSE FALSE      FALSE      FALSE     FALSE      FALSE  FALSE
## [2,] FALSE  FALSE FALSE       TRUE      FALSE     FALSE      FALSE   TRUE
## [3,] FALSE  FALSE FALSE       TRUE      FALSE     FALSE      FALSE   TRUE
## [4,] FALSE  FALSE FALSE      FALSE      FALSE     FALSE      FALSE  FALSE
## [5,] FALSE  FALSE FALSE      FALSE      FALSE     FALSE      FALSE  FALSE
## [6,] FALSE  FALSE FALSE      FALSE      FALSE     FALSE      FALSE  FALSE
##      homeworld species
## [1,]     FALSE   FALSE
## [2,]     FALSE   FALSE
## [3,]     FALSE   FALSE
## [4,]     FALSE   FALSE
## [5,]     FALSE   FALSE
## [6,]     FALSE   FALSE
```

Con esta función obtenemos un vector de **logico** que podemos usar para seleccionar aquellos valores que sean **NA**.

Tambien podemos detectar los **NAs** mediante:

```
summary(starwars)
```

```
##      name              height           mass          hair_color
##  Length:87          Min.   : 66.0   Min.   :  15.00   Length:87
##  Class :character   1st Qu.:167.0   1st Qu.:  55.60   Class :character
##  Mode  :character   Median :180.0   Median :  79.00   Mode  :character
##                     Mean   :174.4   Mean   :  97.31
##                     3rd Qu.:191.0   3rd Qu.:  84.50
```

```
##                    Max.   :264.0   Max.   :1358.00
##                    NA's   :6       NA's   :28
##   skin_color         eye_color           birth_year        gender
##  Length:87          Length:87          Min.   :  8.00   Length:87
##  Class :character   Class :character   1st Qu.: 35.00   Class :character
##  Mode  :character   Mode  :character   Median : 52.00   Mode  :character
##                                        Mean   : 87.57
##                                        3rd Qu.: 72.00
##                                        Max.   :896.00
##                                        NA's   :44
##   homeworld           species
##  Length:87          Length:87
##  Class :character   Class :character
##  Mode  :character   Mode  :character
##
##
##
##
```

```
str(starwars)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    87 obs. of  10 variables:
##  $ name      : chr  "Luke Skywalker" "C-3PO" "R2-D2" "Darth Vader" ...
##  $ height    : int  172 167 96 202 150 178 165 97 183 182 ...
##  $ mass      : num  77 75 32 136 49 120 75 32 84 77 ...
##  $ hair_color: chr  "blond" NA NA "none" ...
##  $ skin_color: chr  "fair" "gold" "white, blue" "white" ...
##  $ eye_color : chr  "blue" "yellow" "red" "yellow" ...
##  $ birth_year: num  19 112 33 41.9 19 52 47 NA 24 57 ...
##  $ gender    : chr  "male" NA NA "male" ...
##  $ homeworld : chr  "Tatooine" "Tatooine" "Naboo" "Tatooine" ...
##  $ species   : chr  "Human" "Droid" "Droid" "Human" ...
```

Una vez que los detectamos, podemos limpiarlos selectivamente.

```
#Limpieza selectiva de los datos sin valor
#Limpiar NA de solamente la variable gender
starwars.cleaned<-starwars[!is.na(starwars$gender),]
#Filas completas para un data frame
complete.cases(starwars) #Checa en que altura de fila hay un NA
```

```
##  [1]  TRUE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
## [12] FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE FALSE
## [23]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE FALSE FALSE
## [34]  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
## [45] FALSE FALSE FALSE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE  TRUE
## [56] FALSE FALSE FALSE FALSE FALSE  TRUE  TRUE FALSE  TRUE FALSE  TRUE
## [67] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [78] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE  TRUE
```

```
starwars.cleaned.2<- starwars[complete.cases(starwars),]

#Convertir algun valor en NA
starwars$skin_color[starwars$skin_color=="gold"]<-NA
```

Una vez que tenemos solo valores disponibles, es decir, ya no tener **NA**, entonces podemos hacer operaciones y obtener medidas de centralidad, como el siguiente ejemplo:

```r
#Medidas de centralización y dispersión
mean(starwars$mass) #Devuelve NA porque aun tiene NA
```

```
## [1] NA
```

```r
mean(starwars$mass, na.rm=TRUE) #Quita los NA y nos arroja un resultado
```

```
## [1] 97.31186
```

```r
sd(starwars$mass)
```

```
## [1] NA
```

```r
sd(starwars$mass,na.rm=TRUE)
```

```
## [1] 169.4572
```

Si no queremos perder información podriamos sustituir los **NA** por valores promedio. *Ojo aquí pueden existir multiples estrategias de imputacion de valores faltantes, eso no se tomara en cuenta en este capitulo*

```r
#Genera una nueva columna, reemplazando NA con el promedio de las observaciones
starwars$birth_year.mean <- ifelse(
                  is.na(starwars$birth_year), #Si esto es verdad
                  mean(starwars$birth_year, na.rm=TRUE), #se ejecuta esto
                      starwars$birth_year #sino pongo el valor original
                           )

starwars$birth_year.mean
```

```
##  [1]  19.00000 112.00000  33.00000  41.90000  19.00000  52.00000  47.00000
##  [8]  87.56512  24.00000  57.00000  41.90000  64.00000 200.00000  29.00000
## [15]  44.00000 600.00000  21.00000  87.56512 896.00000  82.00000  31.50000
## [22]  15.00000  53.00000  31.00000  37.00000  41.00000  48.00000  87.56512
## [29]   8.00000  87.56512  92.00000  87.56512  91.00000  52.00000  87.56512
## [36]  87.56512  87.56512  87.56512  87.56512  62.00000  72.00000  54.00000
## [43]  87.56512  48.00000  87.56512  87.56512  87.56512  72.00000  92.00000
## [50]  87.56512  87.56512  87.56512  87.56512  87.56512  22.00000  87.56512
## [57]  87.56512  87.56512  82.00000  87.56512  58.00000  40.00000  87.56512
## [64] 102.00000  67.00000  66.00000  87.56512  87.56512  87.56512  87.56512
## [71]  87.56512  87.56512  87.56512  87.56512  87.56512  87.56512  87.56512
## [78]  87.56512  87.56512  87.56512  87.56512  87.56512  87.56512  87.56512
## [85]  87.56512  87.56512  46.00000
```

Para evitar datos duplicados, se puede hacer lo siguiente:

```r
family.salary=c(40000,60000,50000,80000,60000,70000,60000)
family.size=c(4,3,2,2,3,4,3)
family.car=c("Lujo","Compacto","Utilitario","Lujo","Compacto","Compacto","Compacto")

family<-data.frame(family.salary,family.size,family.car)
family.unique <- unique(family)

duplicated(family)
```

```
## [1] FALSE FALSE FALSE FALSE  TRUE FALSE  TRUE
```

```r
family[duplicated(family),]
```

```
##   family.salary family.size family.car
```

```
## 5         60000           3   Compacto
## 7         60000           3   Compacto
```

Para ver el comportamiento de los datos, podemos ver su distribución y la correlación entre las variables, para ello en este ejemplo vamos a cargar datos de desordenes de sueño en población mexicana del año 2016, esta información esta disponible en **ensanut**.

```
sleep_disorder_mx <- read.csv("/Users/erickCuevas/Documents/GitProyects/Curso-R-2019/Seccion_Nutricion/

selected <- data.frame(sleep_disorder_mx$cintura, sleep_disorder_mx$cintura2, sleep_disorder_mx$rcintura

selected <- na.omit(selected)

colnames(selected) <- c("cintura","cintura2", "rcintura", "imc", "horasueno", "calidad",
                        "volronquido", "frecron", "hta")

selected <- unique(selected)

selected.cor<-cor(selected, method = "pearson")
round(selected.cor, digits = 2)
```

```
##             cintura cintura2 rcintura   imc horasueno calidad volronquido
## cintura        1.00     1.00    -0.05  0.43     -0.03   -0.08       -0.03
## cintura2       1.00     1.00    -0.02  0.43     -0.03   -0.09       -0.03
## rcintura      -0.05    -0.02     1.00 -0.10      0.43    0.04        0.05
## imc            0.43     0.43    -0.10  1.00     -0.03    0.16        0.17
## horasueno     -0.03    -0.03     0.43 -0.03      1.00    0.14        0.09
## calidad       -0.08    -0.09     0.04  0.16      0.14    1.00        0.69
## volronquido   -0.03    -0.03     0.05  0.17      0.09    0.69        1.00
## frecron        0.07     0.07     0.02  0.09      0.04    0.10        0.22
## hta            0.15     0.15     0.07  0.13      0.09    0.16        0.17
##             frecron  hta
## cintura        0.07 0.15
## cintura2       0.07 0.15
## rcintura       0.02 0.07
## imc            0.09 0.13
## horasueno      0.04 0.09
## calidad        0.10 0.16
## volronquido    0.22 0.17
## frecron        1.00 0.07
## hta            0.07 1.00
```

```
col<-colorRampPalette(c("#BB4444","#EE9988", "#FFFFFF",
                        "#77AADD", "#4477AA"))
corrplot::corrplot(selected.cor,
                   method = "shade",
         shade.col = NA, tl.col = "black",
         tl.srt = 45, col=col(200),
         addCoef.col = "black", addcolorlabel="no",
         order="AOE")
```

```
## Warning in text.default(pos.xlabel[, 1], pos.xlabel[, 2], newcolnames, srt
## = tl.srt, : "addcolorlabel" is not a graphical parameter
```

```
## Warning in text.default(pos.ylabel[, 1], pos.ylabel[, 2], newrownames, col
## = tl.col, : "addcolorlabel" is not a graphical parameter
```

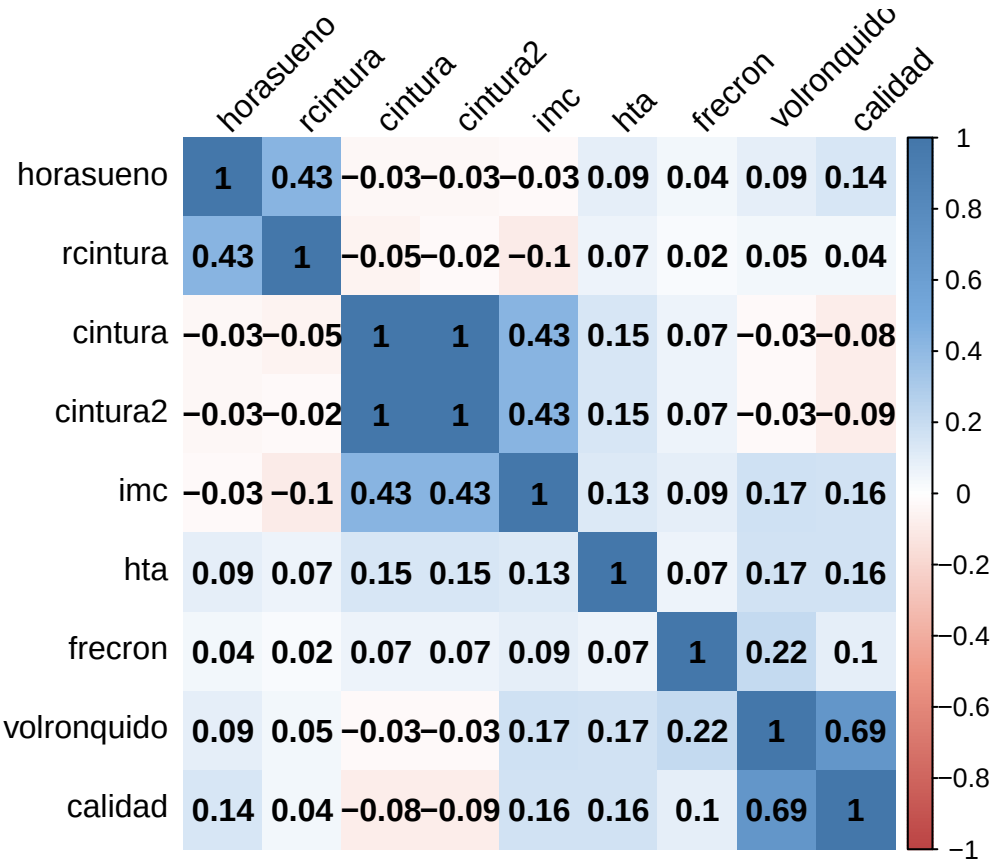
```
## Warning in title(title, ...): "addcolorlabel" is not a graphical parameter
```

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------------------------------------------- tidy
```

```
## v ggplot2 3.2.1     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.3     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
```

```
## -- Conflicts ------------------------------------------------------------------------------- tidyverse_
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

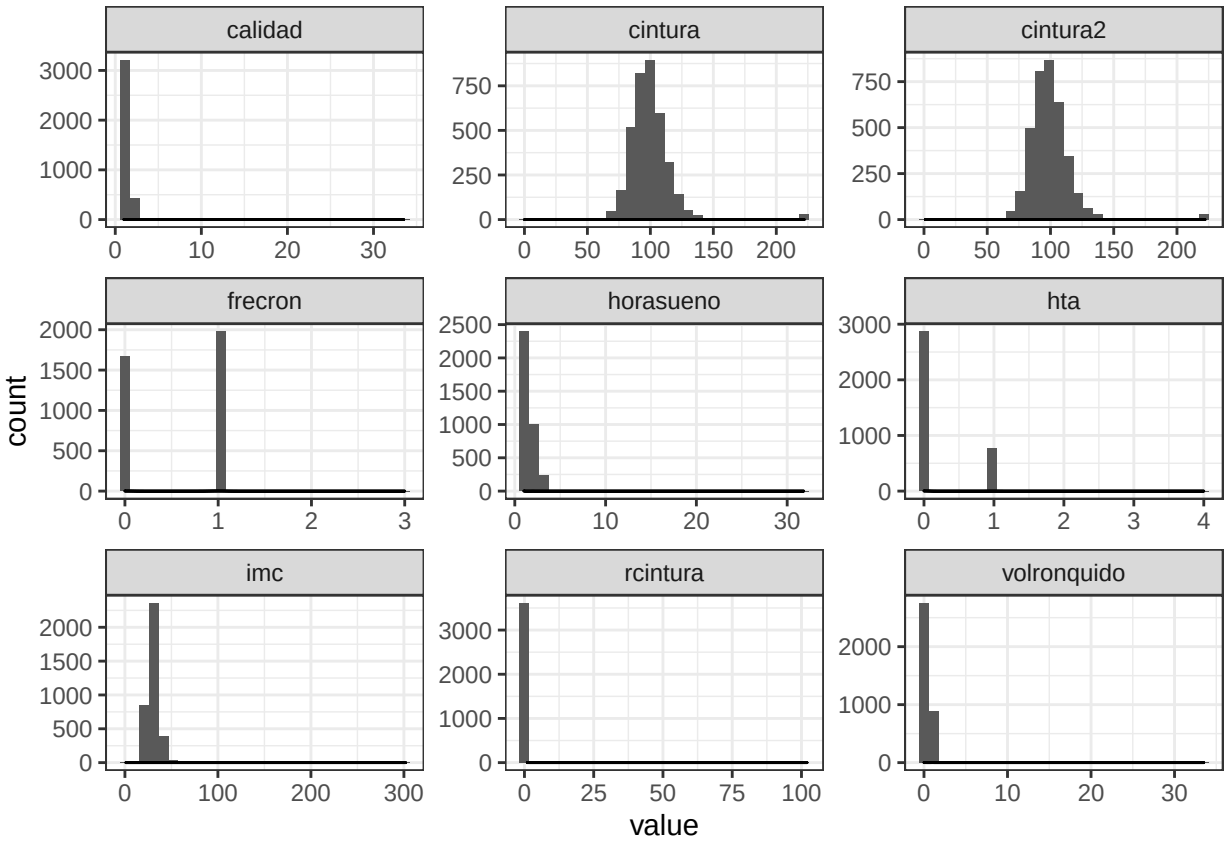



```
select.gathered <- selected %>%
  as_data_frame() %>%
  select_if(is.numeric) %>%
  gather(key = "variable", value = "value")
```

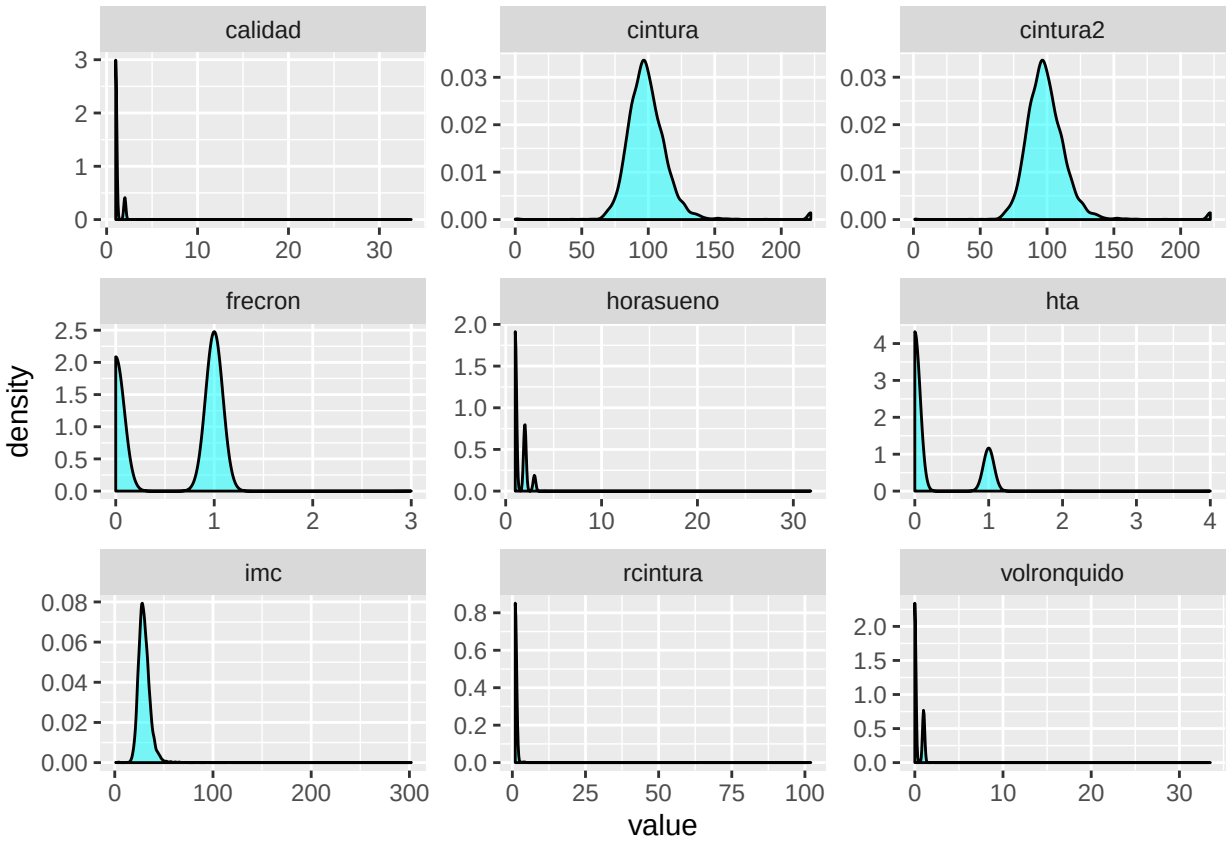
```
## Warning: `as_data_frame()` is deprecated, use `as_tibble()` (but mind the new semantics).
## This warning is displayed once per session.
```

```
ggplot(select.gathered, aes(value)) + geom_histogram() +
  facet_wrap(~variable, scales= "free") +
  geom_density() + theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(select.gathered, aes(value)) +  geom_density(fill= "cyan", color= "black", alpha=0.5) +
  facet_wrap(~variable, scales= "free")
```



Estas son unas funciones básicas para hacer un análisis exploratorio de nuestros datos, y a partir de este poder tomar decision en estadistica.