Resumir informacion y y aplicar funciones por Rangos

Curso: Manejo de datos y reportería con R

Néstor Montaño

Sociedad Ecuatoriana de Estadística

Enero-2021

Nota:
Con *Alt* + *F* o *Option* + *F* puede hacer que estas dapositivas ocupen todo el navegador (es decir que se ignore el aspecto de diapositiva que tiene por default la presentación)

R - cargar librerías

```
#### LIBRERIAS ------
library(openxlsx) # para importar desde excel
library(tidyverse) # manipulacion de datos
library(ggplot2) # graficos
library(magrittr) # %>%
library(lubridate) # Manipulacion de fechas
library(stringr) # Manipulacion de texto
```

Ejemplo: Transacciones bancarias

El Banco del Pacífico requiere mejorar los tiempos de atención al cliente en ventanilla, para ello ha recolectado esta información anónimamente para cada cajero y transacción realizada.

Le suministran un excel con dos hojas:

1. Tiene los datos de las transacciones, columnas: Sucursal, Cajero, ID_Transaccion, Transaccion, Tiempo_Servicio_seg, Nivel de satisfacción, Monto de la transaccion.
2. Otra hoja que indica si en la sucursal se ha puesto o no el nuevo sistema.

Ejemplo - Importar

```
# Leer el archivo de excel y asignarlo al objeto data_banco
data_banco <- read.xlsx(xlsxFile = "Data/Data_Banco.xlsx", sheet = "Data")
data_sucursal <- read.xlsx(xlsxFile = "Data/Data_Banco.xlsx", sheet = "Data_Sucursal")
```

Ejemplo - Convertir a tibbles (un dataframe mejorado):

```
# Convertir el data_banco a un tibble
data_banco <- tbl_df( data_banco)
```

```
## Warning: `tbl_df()` is deprecated as of dplyr 1.0.0.
## Please use `tibble::as_tibble()` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```
# Convertir el data_sucursal a un tibble
data_sucursal <- tbl_df(data_sucursal)
```

Ejemplo - Manipulacion de datos

Lo primero que necesitamos es corregir los tipos de datos, nótese que *Monto* tiene una mezcla de "," y ".".

```
data_banco <- data_banco %>%
  mutate( Monto= str_replace(Monto, pattern = ",", replacement = ".") ) %>%
  mutate(Sucursal= as.character(Sucursal),
         Cajero = as.character(Cajero),
         Satisfaccion = parse_factor(Satisfaccion,
                                     levels= c('Muy Malo', 'Malo', 'Regular',
                                               'Bueno', 'Muy Bueno')),
         Monto= parse_number(Monto, locale = locale(decimal_mark = ".")))
```

Explorar los datos

Con las columnas corregidas, podemos empezar a explorar nuestros datos; para ello podemos empezar seleccionando columnas y filtrar filas para dar vistazos a los valores que toma cada variable.

Sin embargo, es imposible poder descubrir las estructuras subyacentes de los datos de esa manera; necesitamos resumir la complejidad de los las cientos, miles o millones de observaciones en unos pocos valores; aquí entran en acción las medidas estadísticas descriptivas.

Estadística descriptiva - Estadisticos|Medidas

Curso: Manejo de datos y reportería con R

Néstor Montaño

Medidas de Tendencia Central

Media.- Promedio de los valores

- Se la puede entender como el punto de equilibrio
- Muy sensible a valores aberrantes
- En R: mean(x, na.rm= TRUE)

Media Acotada.- Promedio de los valores, pero quitando un porcentaje de valores extremos.

- Es menos sensible a valores aberrantes
- Se puede perder información importante
- En R: mean(x, na.rm= TRUE, trim)

Medidas de Tendencia Central

Mediana.- Punto medio de los valores una vez que se han ordenado de menor a mayor o de mayor a menor.

- Valor importante pero poco usado
- No es sensible a valores aberrantes
- En R: median(x, na.rm= TRUE)

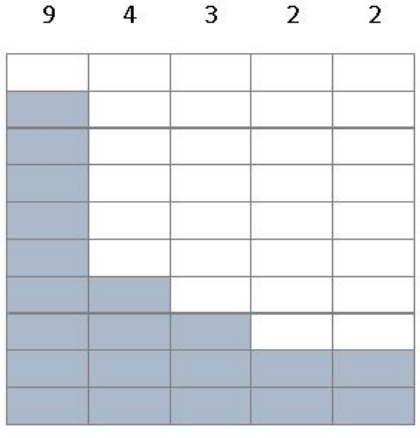
Media Ponderada.- Promedio de los valores, pero asignando un peso diferente a cada valor.

- Normalmente se utiliza cuando se tiene datos agrupados
- Es también sensible a valores aberrantes
- En R: weighted.mean(x, w, ..., na.rm = TRUE)

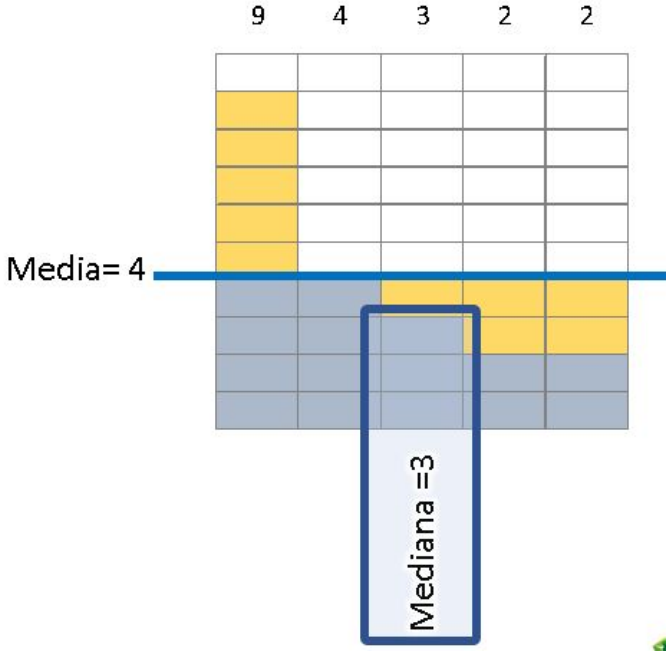
Medidas de Tendencia Central

Entendiendo media vs mediana

- Observaciones



- Media y mediana



Medidas de Tendencia Central

La idea general de los **métodos robustos** es obtener estimadores que no se vean muy afectados por los valores aberrantes.
Media Robusta de Huber- Propuesto por él en los 60s, es un método asigna (por medio de un algoritmo) un peso a cada observación, siendo los outliers castigados con un peso menor.

- Poco usado pero totalmente recomendado.
- Da un valor entre la media y la mediana
- Tiene un algoritmo numérico inmerso, por lo que si cálculo no es inmediato
- En R: huber(y, k = 1.5, tol = 1e-06)

Medidas de Tendencia Central

Moda.- Valor de la observación que aparece con mayor frecuencia, esto funciona sobre todo cuando la variable es discreta o categórica; cuando la variable es contínua la moda es el valor que maximiza la función de densidad (o valor más probable).

- Mejor análisis se obtiene con una tabla de frecuencias
- Pueden existir más de una moda
- Incluso pueden haber modas locales
- En R:
  - library('DescTools')
  - Mode(x)
  - library('modeest')
  - mlv(x, method= "") <- este es más completo pero requiere java

Medidas de Tendencia Central

Calcular las medidas de Tendencia central para la data de Banco

```
# Media
# mean(data_banco$Tiempo_Servicio_seg, na.rm = TRUE) ## Base R
data_banco %$% mean(Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 155.58
```

```
# Media acotada al 10%
data_banco %$% mean(Tiempo_Servicio_seg, trim = 0.05, na.rm = TRUE)
```

```
## [1] 141.9233
```

Medidas de Tendencia Central

Calcular las medidas de Tendencia central para la data de Banco

```
# Mediana
data_banco %$% median(Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 122.4523
```

```
# Media de Huber
# MASS::huber(data_banco$Tiempo_Servicio_seg)$mu ## Base R
data_banco %$% MASS::huber(Tiempo_Servicio_seg) %>% as.data.frame() %>% select(mu)
```

```
##          mu
## 1 138.0164
```

Medidas de Tendencia Central: Moda

Calcular la Moda del Tiempo de servicio en segundo

```
data_banco %$%
  DescTools::Mode(Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 46
## attr(,"freq")
## [1] 60
```

```
data_banco %$%
  modeest::mlv(Tiempo_Servicio_seg, method= "mfv")
```

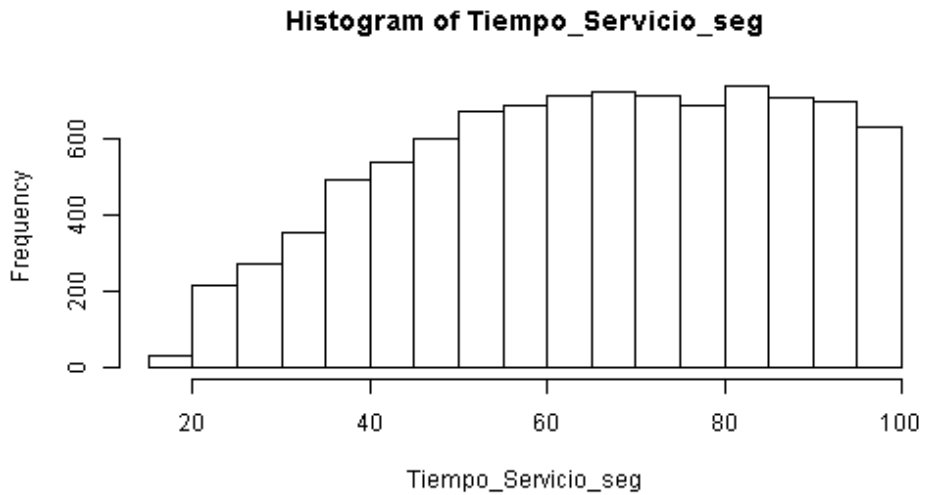
```
## Registered S3 method overwritten by 'rmutil':
##   method         from
##   print.response httr

## [1] 46
```

```
data_banco %$%
  modeest::mlv(Tiempo_Servicio_seg, method= "Venter")
```

```
## [1] 85.4812
```

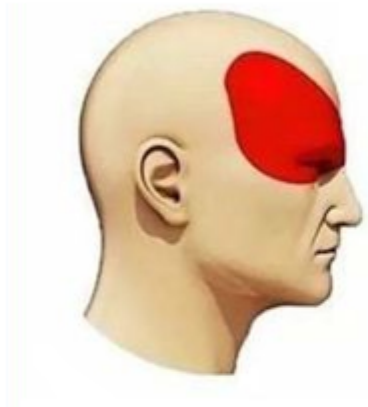
Para entender la diferencia entre valor más repetido o valor que maximiza la densidad.

```
data_banco %>%
  filter(Tiempo_Servicio_seg <100) %$%
  hist(Tiempo_Servicio_seg)
```

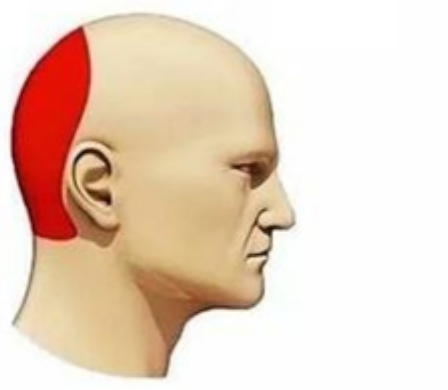


Los dolores de cabeza de un estadístico o científico de datos

Migraña

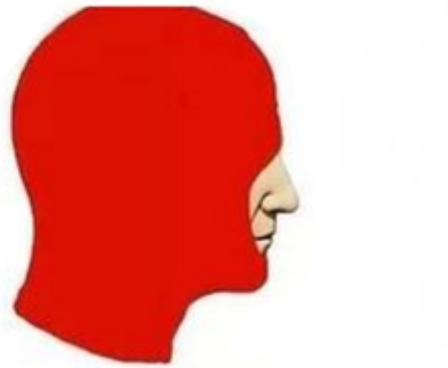


Hipertensión



Estrés

Ver a un "data scientist" obteniendo promedio de una variable categórica



SOCIEDAD ECUATORIANA DE ESTADISTICA

Medidas de Posición

- Min y Max
  - En R, `min(x, na.rm = TRUE)`, `max(x, na.rm = TRUE)`
- Cuartiles.- Dividen al conjunto de observaciones en **4** partes iguales
  - El segundo cuartil es la mediana
  - En R `quantile(x , probs = c(0.25, 0.50, 0.75))`
- Deciles.- Dividen al conjunto de observaciones en **10** partes iguales
  - El quinto decil sería igual a la mediana
  - En R `quantile(x , probs = seq(from = 0.1, to = 1, by = 0.1))`
- Centiles.- Dividen al conjunto de observaciones en **100** partes iguales
  - El quincuagésimo centil es la mediana
  - En R `quantile(x, probs )`

Medidas de Posición

Entendiendo los cuartiles

Posición	1	2	3	4	5	6	7	8	9	10	11
Observación	11	25	38	41	57	62	71	79	84	91	99
			Cuartil 1			Cuartil 2			Cuartil 3		

Medidas de Posición

Calcular las medidas de Posición para el tiempo de servicio data de Banco

```
# Minimo y Máximo
min(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 18.13177
```

```
max(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 1602.698
```

```
# Cuartiles
quantile(data_banco$Tiempo_Servicio_seg , probs = c(0.25, 0.50, 0.75))
```

```
##       25%       50%       75%
##  75.69119 122.45229 197.73046
```

Medidas de Posición

Calcular las medidas de Posición para el tiempo de servicio data de Banco

```
# Deciles
quantile(data_banco$Tiempo_Servicio_seg , probs = seq(from = 0.1, to = 1, by = 0.1))
```

```
##       10%       20%       30%       40%       50%       60%       70%       80%
##   49.6230   67.0000   84.0000  102.0000  122.4523  146.9901  178.8348  220.2469
##       90%      100%
##  298.7826 1602.6983
```

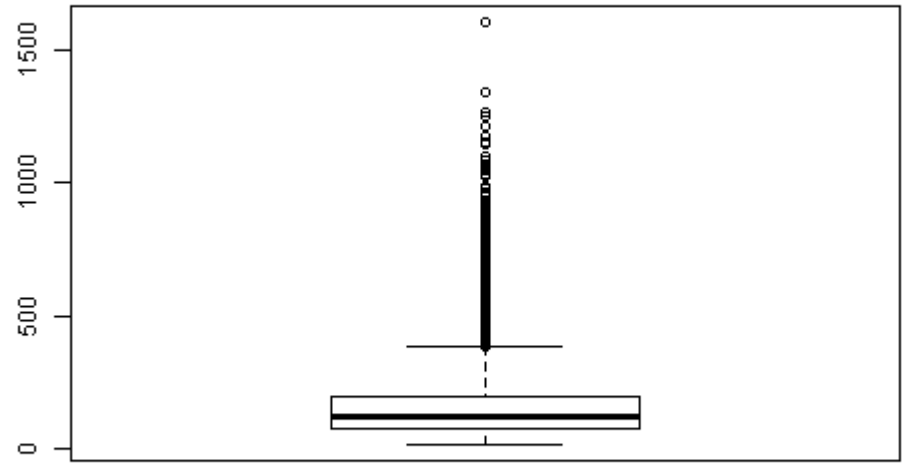
```
# Centil 5% y 95%
quantile(data_banco$Tiempo_Servicio_seg , probs = c(0.05, 0.95))
```

```
##       5%      95%
##  39.0000 382.9779
```

Medidas de Posición - Boxplot

Boxplot.- Muestra gráficamente las medidas de posición

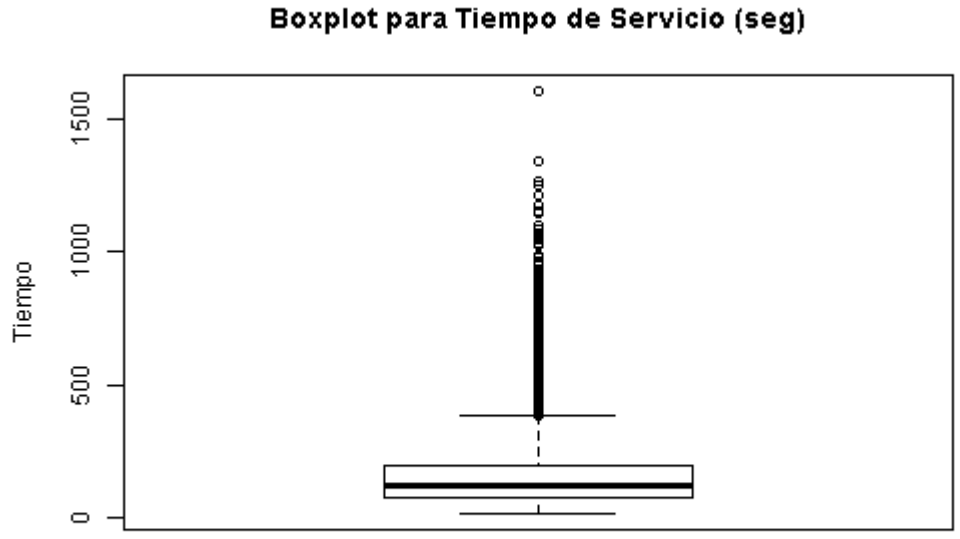
```
# Un primer Boxplot
boxplot(data_banco$Tiempo_Servicio_seg)
```



Medidas de Posición - Boxplot

Boxplot.- Muestra gráficamente las medidas de posición

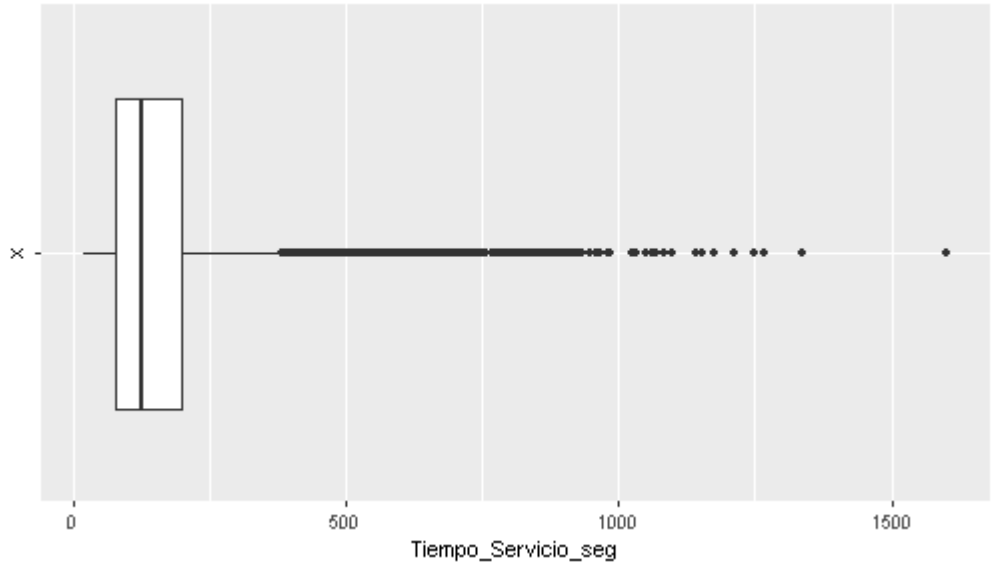
```
boxplot(data_banco$Tiempo_Servicio_seg,
        main= "Boxplot para Tiempo de Servicio (seg)", ylab= "Tiempo")
```



Medidas de Posición - Boxplot

Boxplot.- Muestra gráficamente las medidas de posición
Se puede usar el paquete ggplot2 para gráficos profesionales, esto no se verá en el presente curso, sin mebargo se lo muestra:

```
library("ggplot2")
ggplot(data = data_banco, aes(x = "", y = Tiempo_Servicio_seg)) +
  geom_boxplot() +
   coord_flip()
```



Medidas de Dispersión

- Varianza.- Media aritmética de las desviaciones de la media elevadas al cuadrado
  - $s^2 = \frac{\sum(x-\bar{x})^2}{n-1}$
  - En R: var(x, na.rm = TRUE)
- Desviación Estándar.- Raíz cuadrada de la varianza.
  - Esta medida se utiliza frecuentemente para realizar comparaciones entre dos conjuntos de datos
  - $s = \sqrt{\frac{\sum(x-\bar{x})^2}{n-1}}$
  - En R: sd(x, na.rm = TRUE)
- Mediana de las desviaciones absolutas.- Estimador Robusto
  - $mad = mediana[x - mediana(x)]$

Medidas de Dispersión

- Rango intercuartil.- Distancia entre el cuartil 1 y 3
  - En R IQR(x, na.rm = TRUE)
- Rango.- Diferencia entre maximo y minimo valor
  - En R diff(range(x, na.rm = TRUE))

Medidas de Dispersión

Calcular las medidas de dispersión

```
# Varianza
var(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 14402.27
```

```
# Desviacion
sd(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 120.0095
```

```
# Mediana de las desviaciones absolutas
mad(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 82.35503
```

Medidas de Dispersión

Calcular las medidas de dispersión

```
# Rango intercuartil
IQR(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1] 122.0393
```

```
# Min Max
range(data_banco$Tiempo_Servicio_seg, na.rm = TRUE)
```

```
## [1]   18.13177 1602.69832
```

```
# Rango
diff(range(data_banco$Tiempo_Servicio_seg, na.rm = TRUE))
```

```
## [1] 1584.567
```

Descriptivas - Summary

Obtiene las siguientes estadísticas descriptivas:

- Variables Numéricas: Min, Max, Cuartiles y Media
- Variables carácter: El total de datos
- Variables factor: frecuencias
- summary()

```
summary( data_banco$Tiempo_Servicio_seg)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   18.13   75.69  122.45  155.58  197.73 1602.70
```

Descriptivas - Summary

```
summary( data_banco)
```

```
##    Sucursal            Cajero          ID_Transaccion     Transaccion
##  Length:24299       Length:24299       Length:24299       Length:24299
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  Tiempo_Servicio_seg    Satisfaccion      Monto
##  Min.   :  18.13     Muy Malo :3009   Min.   :  53.82
##  1st Qu.:  75.69     Malo     :4474   1st Qu.:1417.73
##  Median : 122.45     Regular  :4639   Median :2087.43
##  Mean   : 155.58     Bueno    :5915   Mean   :1996.16
##  3rd Qu.: 197.73     Muy Bueno:6262   3rd Qu.:2482.09
##  Max.   :1602.70                      Max.   :6278.02
```

Descriptivas - PrettyR

Otra opción es usar paquetes especiales como `PrettyR` que tiene el comando `describe()` que es más completo que `summary()`, les dejo un ejemplo:

```
library(prettyR)
describe( data_banco,
          num.desc = c("mean", "sd", "median",
                       "min", "max", "valid.n"))
```

```
## Description of data_banco

##
##  Numeric
##                          mean     sd  median   min     max valid.n
## Tiempo_Servicio_seg  155.58 120.01  122.45 18.13 1602.70   24299
## Monto               1996.16 816.15 2087.43 53.82 6278.02   24299
##
##  Factor
##
## Satisfaccion Muy Bueno   Bueno Regular    Malo Muy Malo
##      Count     6262.00 5915.00 4639.00 4474.00  3009.00
##      Percent     25.77   24.34   19.09   18.41    12.38
## Mode Muy Bueno
```

Resumir/Agregar los Datos

Hemos aprendido ciertas generalidades de R, importar datos, entender qué tipo de objetos tenemos dentro de R, manipular nuestro dataframe para seleccionar columnas, filtrar filas, modificar o crear columnas y cómo explorar nuestro dataframe usando estadística descriptiva.

Sin embargo, supongamos que tengo la impresión de que ciertas sucursales demoran más sus transacciones, ¿Cómo podría explorar esa hipótesis?. Es decir, si quisiera explorar la relación entre el tiempo (numérica) versus la sucursal (categórica) ¿Qué debería hacer?.

Pues aquí se empieza a utilizar lo que se conoce como cálculos por grupos de filas.

Resumir/Agregar los Datos

Curso: Manejo de datos y reportería con R

Néstor Montaño

Crear resúmenes: summarise()

`summarise()` permite aplicar funciones a las columnas de nuestro data.frame. En R-base se usaría `tapply()` y otra opcion muy conocida es `ddply()` del paquete plyr.

```
# Obtener la medidas descriptivas del tiempo de servicio
data_banco %>%
  summarise(
    MEDIA= mean(Tiempo_Servicio_seg, na.rm=TRUE),
    MEDIA_ACOT= mean(Tiempo_Servicio_seg, na.rm = TRUE, trim = 0.05),
    DESV= sd(Tiempo_Servicio_seg, na.rm=TRUE),
    RANGO= diff(range(Tiempo_Servicio_seg)),
        CANTIDAD= n() # n() permite contar el número de filas
  )
```

```
## # A tibble: 1 x 5
##   MEDIA MEDIA_ACOT  DESV RANGO CANTIDAD
##   <dbl>      <dbl> <dbl> <dbl>    <int>
## 1  156.       142.  120. 1585.    24299
```

Crear resúmenes: summarise()

`summarise_at()` para escoger la(s) variable(s) a utilizar en los cálculos, la diferencia con lo anterior es que no se repite el nombre de la variable en cada línea.

```
# Obtener la media del tiempo de servicio
data_banco %>%
  summarise_at( vars(Tiempo_Servicio_seg),
                list(
                  MEDIA= ~mean(., na.rm=TRUE),
                  MEDIA_ACOT= ~mean(., na.rm = TRUE, trim = 0.05),
                  DESV= ~sd(., na.rm=TRUE),
                  RANGO= ~diff(range(.)),
                  CANTIDAD= ~n()
                )
  )
```

```
## # A tibble: 1 x 5
##   MEDIA MEDIA_ACOT  DESV RANGO CANTIDAD
##   <dbl>      <dbl> <dbl> <dbl>    <int>
## 1  156.       142.  120. 1585.    24299
```

Crear resúmenes: summarise()

`summarise_at()` escogiendo varias variables, nótese el nombre de las columnas resultantes

```
# Obtener la media del tiempo de servicio y el Monto
data_banco %>%
  summarise_at( vars(Tiempo_Servicio_seg, Monto),
                  list(
                  MEDIA= ~mean(., na.rm=TRUE),
                  DESV= ~sd(., na.rm = TRUE),
                  CANTIDAD= ~n()
                )
  )
```

```
## # A tibble: 1 x 6
##   Tiempo_Servicio~ Monto_MEDIA Tiempo_Servicio~ Monto_DESV Tiempo_Servicio~
##              <dbl>       <dbl>            <dbl>      <dbl>            <int>
## 1             156.       1996.             120.       816.            24299
## # ... with 1 more variable: Monto_CANTIDAD <int>
```

Crear resúmenes: summarise()

`summarise_if()` permite escoger varias variables que cumplan una condición

```
# Obtener la media del tiempo de servicio y el Monto
data_banco %>%
  summarise_if( is.numeric,
                  list(
                  MEDIA= ~mean(., na.rm=TRUE),
                  MEDIA_ACOT= ~mean(., na.rm = TRUE, trim = 0.05)
                )
  )
```

```
## # A tibble: 1 x 4
##   Tiempo_Servicio_seg_ME~ Monto_MEDIA Tiempo_Servicio_seg_MEDI~ Monto_MEDIA_ACOT
##                     <dbl>       <dbl>                     <dbl>             <dbl>
## 1                    156.       1996.                      142.             1983.
```

Crear resúmenes para datos agrupados

`group_by()` permite aplicar funciones a nuestro data.frame separado por una o más variables, por ejemplo para obtener medidas de tendencia central para el tiempo de servicio según cada Transacción se haría:

```
# Obtener medidas de tendencia central para el tiempo de servicio para cada tipo de transaccion
data_banco %>%
  group_by(Transaccion) %>%
  summarise(
    MEDIA= mean(Tiempo_Servicio_seg, na.rm=TRUE),
    MEDIA_ACOT= mean(Tiempo_Servicio_seg, na.rm = TRUE, trim = 0.05),
    DESV= sd(Tiempo_Servicio_seg, na.rm=TRUE),
       CANTIDAD= n()
  )
```

```
## `summarise()` ungrouping output (override with `.groups` argument)

## # A tibble: 3 x 5
##   Transaccion                MEDIA MEDIA_ACOT  DESV CANTIDAD
##   <chr>                      <dbl>      <dbl> <dbl>    <int>
## 1 Cobrar cheque (Cta del Bco)  186.       175. 115.      5407
## 2 Cobro/Pago (Cta externa)     301.       285. 185.      3005
## 3 Deposito                     118.       112.  72.4    15887
```

Crear resúmenes para datos agrupados

Obtener medidas de tendencia central del tiempo de servicio para cada combinación de Transaccion y Nivel de Satisfaccion.

```
# Obtener medidas de tendencia central del
# tiempo de servicio para cada combinación de
# Transaccion y Nivel de Satisfaccion.
data_banco %>%
  group_by(Transaccion, Satisfaccion) %>%
  summarise(
    MEDIA= mean(Tiempo_Servicio_seg, na.rm=TRUE),
    MEDIA_ACOT= mean(Tiempo_Servicio_seg,
                     na.rm = TRUE, trim = 0.05),
    DESV= sd(Tiempo_Servicio_seg, na.rm=TRUE),
       CANTIDAD= n()
  )
```

```
## `summarise()` regrouping output by 'Transaccion' (override wi

## # A tibble: 15 x 6
## # Groups:   Transaccion [3]
##    Transaccion                  Satisfaccion MEDIA MEDIA_ACOT
##    <chr>                        <fct>        <dbl>      <dbl>
##  1 Cobrar cheque (Cta del Bco) Muy Malo      186.       175.
##  2 Cobrar cheque (Cta del Bco) Malo          183.       174.
##  3 Cobrar cheque (Cta del Bco) Regular       188.       177.
##  4 Cobrar cheque (Cta del Bco) Bueno         185.       174.
##  5 Cobrar cheque (Cta del Bco) Muy Bueno     187.       176.
##  6 Cobro/Pago (Cta externa)    Muy Malo      272.       260.
##  7 Cobro/Pago (Cta externa)    Malo          290.       271.
##  8 Cobro/Pago (Cta externa)    Regular       307.       293.
##  9 Cobro/Pago (Cta externa)    Bueno         309.       292.
## 10 Cobro/Pago (Cta externa)    Muy Bueno     304.       287.
## 11 Deposito                    Muy Malo      113.       108.
## 12 Deposito                    Malo          116.       110.
## 13 Deposito                    Regular       120.       113.
## 14 Deposito                    Bueno         118.       112.
## 15 Deposito                    Muy Bueno     119.       114.
```

Crear resúmenes para datos agrupados y filtrados

Para la Sucursal 62, obtener medidas de tendencia central del tiempo de servicio para cada combinación de Transaccion y Nivel de Satisfaccion.

```
# Para la Sucursal 62
# Obtener medidas de tendencia central del
# tiempo de servicio para cada combinación de
# Transaccion y Nivel de Satisfaccion.
data_banco %>%
  filter( Sucursal== 62) %>%
  group_by(Transaccion, Satisfaccion) %>%
  summarise(
    MEDIA= mean(Tiempo_Servicio_seg, na.rm=TRUE),
    MEDIA_ACOT= mean(Tiempo_Servicio_seg,
                     na.rm = TRUE, trim = 0.05),
    DESV= sd(Tiempo_Servicio_seg, na.rm=TRUE),
       CANTIDAD= n()
  )
```

```
## `summarise()` regrouping output by 'Transaccion' (override wi

## # A tibble: 15 x 6
## # Groups:   Transaccion [3]
##    Transaccion                 Satisfaccion MEDIA MEDIA_ACOT
##    <chr>                       <fct>        <dbl>      <dbl>
##  1 Cobrar cheque (Cta del Bco) Muy Malo     102.        99.0
##  2 Cobrar cheque (Cta del Bco) Malo         117.       114.
##  3 Cobrar cheque (Cta del Bco) Regular      104.       101.
##  4 Cobrar cheque (Cta del Bco) Bueno        108.       105.
##  5 Cobrar cheque (Cta del Bco) Muy Bueno    102.        98.4
##  6 Cobro/Pago (Cta externa)    Muy Malo     161.       158.
##  7 Cobro/Pago (Cta externa)    Malo         152.       146.
##  8 Cobro/Pago (Cta externa)    Regular      171.       165.
##  9 Cobro/Pago (Cta externa)    Bueno        165.       159.
## 10 Cobro/Pago (Cta externa)    Muy Bueno    167.       163.
## 11 Deposito                    Muy Malo      67.2       65.4
## 12 Deposito                    Malo          64.1       62.2
## 13 Deposito                    Regular       63.7       62.3
## 14 Deposito                    Bueno         65.8       64.3
## 15 Deposito                    Muy Bueno     67.1       65.4
```

Crear resúmenes con funciones de más de una salida

Suponga que desea obtener medidas de posición para el tiempo de servicio según el Nivel de Satisfaccion, ya vimos que la función quantile genera un vector y no sólo un número, ¿Cómo podríamos usar esta salida dentro de summarise?

```
# Obtener medidas de tendencia central
data_banco %>%
  group_by(Satisfaccion) %>%
  summarise(
    tibble(
      Quartil= c("Min","Q1", "Mediana", "Q3","Max"),
      Valor= quantile(Tiempo_Servicio_seg,
                      c(0, 0.25, 0.5, 0.75, 1))
      )
    )
```

```
## `summarise()` regrouping output by 'Satisfaccion' (override w

## # A tibble: 25 x 3
## # Groups:   Satisfaccion [5]
##    Satisfaccion Quartil  Valor
##    <fct>        <chr>    <dbl>
##  1 Muy Malo     Min       18.1
##  2 Muy Malo     Q1        69.5
##  3 Muy Malo     Mediana  111
##  4 Muy Malo     Q3       176.
##  5 Muy Malo     Max      823.
##  6 Malo         Min       20
##  7 Malo         Q1        71.7
##  8 Malo         Mediana  115.
##  9 Malo         Q3       184.
## 10 Malo         Max     1337.
## # ... with 15 more rows
```

Estadística descriptiva - Explorar la forma de nuestros datos

Curso: Manejo de datos y reportería con R

Néstor Montaño

Tablas de Frecuencia

- Agrupación de datos en clases mutuamente excluyentes, que muestra el número de observaciones que hay en cada clase.
  - Se agrupa en Intervalos si la variable es cuantitativa.
  - Se cuenta cada elemento si la variable es cualitativa.
  - Se lo muestra gráficamente con un histograma o gráfico de barras

Tablas de Frecuencia - V. Numérica

Calcular tabla de frecuencias

- library('fdth')
- fdt(data , breaks="Sturges")
- fdt(data , start, end, h, right = FALSE)

Intervalos	Frecuencia	Frecuencia Relativa	F. Relativa Acumulada
[LI_1 - LS_1)			
[LI_2 - LS_2)	Número de observaciones que caen en cada intervalo	Porcentaje de observaciones que caen en cada intervalo	Acumulado de los Porcentajes
.			
.			
[LI_n - LS_n]			

Tablas de Frecuencia - V. Numérica

```
library('fdth')
```

```
## Warning: package 'fdth' was built under R version 3.6.3
```

```
## tabl_frec <- fdt( data_banco$Tiempo_Servicio_seg ,
##                          breaks="Sturges" ) ## Base R
tabl_frec <- data_banco %$%
  fdt(Tiempo_Servicio_seg, breaks="Sturges" )
tabl_frec
```

```
##          Class limits     f   rf rf(%)    cf  cf(%)
##  [17.95045,117.9989) 11642 0.48 47.91 11642  47.91
##  [117.9989,218.0473)  7697 0.32 31.68 19339  79.59
##  [218.0473,318.0957)  2910 0.12 11.98 22249  91.56
##  [318.0957,418.1442)  1119 0.05  4.61 23368  96.17
##  [418.1442,518.1926)   454 0.02  1.87 23822  98.04
##   [518.1926,618.241)   241 0.01  0.99 24063  99.03
##   [618.241,718.2894)   114 0.00  0.47 24177  99.50
##  [718.2894,818.3379)    63 0.00  0.26 24240  99.76
##  [818.3379,918.3863)    34 0.00  0.14 24274  99.90
##  [918.3863,1018.435)    10 0.00  0.04 24284  99.94
##  [1018.435,1118.483)     7 0.00  0.03 24291  99.97
##  [1118.483,1218.532)     4 0.00  0.02 24295  99.98
##   [1218.532,1318.58)     2 0.00  0.01 24297  99.99
##   [1318.58,1418.628)     1 0.00  0.00 24298 100.00
##  [1418.628,1518.677)     0 0.00  0.00 24298 100.00
```

Tablas de Frecuencia - V. Numérica

tabl_frec es una lista, veamos sus componentes (esto es útil para cuando quieran exportar a excel por ejemplo)

```
# Limites de los intervalos
tabl_frec$breaks
```

```
##      start        end          h      right
##   17.95045 1618.72530  100.04843    0.00000
```

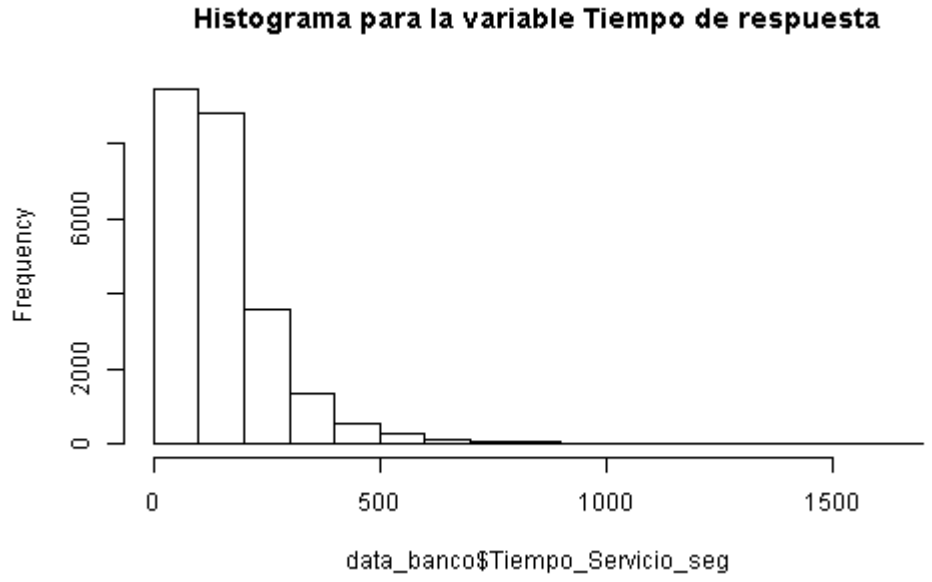
```
# Tabla de frecuencias
tabl_frec$table
```

```
##           Class limits     f           rf        rf(%)    cf     cf(%)
## 1  [17.95045,117.9989) 11642 4.791144e-01 47.911436685 11642  47.91144
## 2  [117.9989,218.0473)  7697 3.167620e-01 31.676200667 19339  79.58764
## 3  [218.0473,318.0957)  2910 1.197580e-01 11.975801473 22249  91.56344
## 4  [318.0957,418.1442)  1119 4.605128e-02  4.605127783 23368  96.16857
## 5  [418.1442,518.1926)   454 1.868390e-02  1.868389646 23822  98.03696
## 6   [518.1926,618.241)   241 9.918104e-03  0.991810363 24063  99.02877
## 7   [618.241,718.2894)   114 4.691551e-03  0.469155109 24177  99.49792
## 8  [718.2894,818.3379)    63 2.592699e-03  0.259269929 24240  99.75719
## 9  [818.3379,918.3863)    34 1.399235e-03  0.139923454 24274  99.89712
## 10 [918.3863,1018.435)    10 4.115396e-04  0.041153957 24284  99.93827
## 11 [1018.435,1118.483)     7 2.880777e-04  0.028807770 24291  99.96708
## 12 [1118.483,1218.532)     4 1.646158e-04  0.016461583 24295  99.98354
```

Tablas de Frecuencia - V. Numérica

Histograma

```
hist( data_banco$Tiempo_Servicio_seg , breaks="Sturges" ,
      main = "Histograma para la variable Tiempo de respuesta")
```



Tablas de Frecuencia - V. Numérica

Histograma

```
# Con ggplot2
ggplot(data = data_banco, aes(x= Tiempo_Servicio_seg)) +
  geom_histogram( aes(y= ..count..)) +
  labs(title= 'Histograma para Tiempo de Servicio (seg)', y= "Cantidad", x= "Tiempo")
```

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

Tablas de Frecuencia - V. Numérica

Calcular tabla de frecuencias

```
# Definiendo nosotros mismos los rangos
## fdt(data_banco$Tiempo_Servicio_seg, start = 0, end = 500, h = 50, right = FALSE) ## BaseR
tabl_frec <- data_banco %$%
  fdt(Tiempo_Servicio_seg, start = 0, end = 500, h = 50, right = FALSE)
tabl_frec
```

```
##  Class limits    f   rf rf(%)    cf cf(%)
##         [0,50) 2464 0.10 10.14  2464 10.14
##        [50,100) 6990 0.29 28.77  9454 38.91
##       [100,150) 5378 0.22 22.13 14832 61.04
##       [150,200) 3507 0.14 14.43 18339 75.47
##       [200,250) 2281 0.09  9.39 20620 84.86
##       [250,300) 1280 0.05  5.27 21900 90.13
##       [300,350)  816 0.03  3.36 22716 93.49
##       [350,400)  515 0.02  2.12 23231 95.60
##       [400,450)  311 0.01  1.28 23542 96.88
##       [450,500)  213 0.01  0.88 23755 97.76
```

Tablas de Frecuencia - V. Cualitativa

- Calcular tabla de frecuencias
  - En R, comando table()
  - Se representa con gráficos de barra
  - barplot(table(data))
  - barplot(table(data), horiz = TRUE)

Tablas de Frecuencia - V. Cualitativa

Frecuencia para la transacción que se está realizando y para nivel de satisfacción

```
# Table
table(data_banco$Transaccion)
```

```
##
## Cobrar cheque (Cta del Bco)    Cobro/Pago (Cta externa)
##                        5407                        3005
##                     Deposito
##                        15887
```

```
table(data_banco$Satisfaccion)
```

```
##
##  Muy Malo      Malo   Regular     Bueno Muy Bueno
##      3009      4474      4639      5915      6262
```

Tablas de Frecuencia - V. Cualitativa

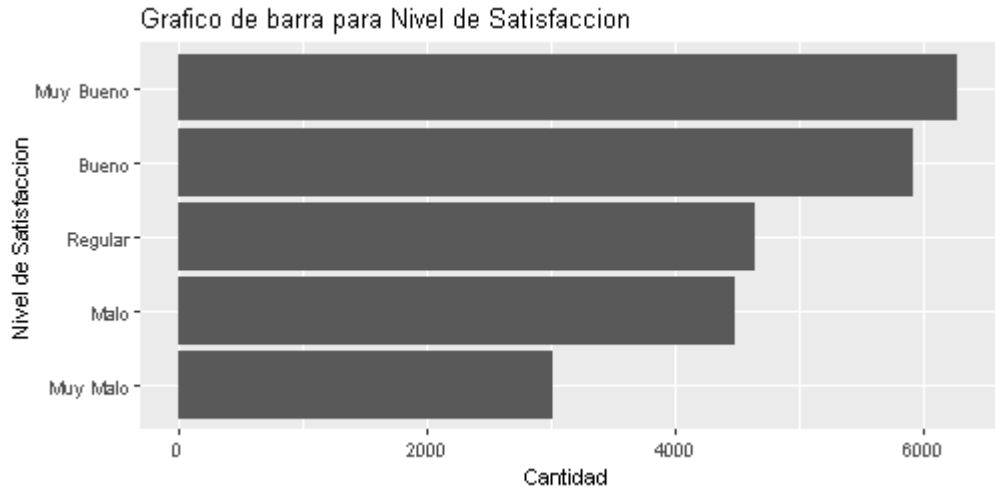
Frecuencia para la transacción que se está realizando y para nivel de satisfacción

```
# Gráfico Vertical
barplot( table( data_banco$Transaccion ))
```

Tablas de Frecuencia - V. Cualitativa

Grafico usando ggplot2 (en otro curso se enseñará todo lo que está detrás de ggplot2)

```
# Con ggplot2
ggplot(data = data_banco, aes(x= Satisfaccion)) +
  geom_bar( ) +
  coord_flip() +
  labs(title= 'Grafico de barra para Nivel de Satisfaccion',
       y= "Cantidad", x= "Nivel de Satisfaccion")
```



Tablas de Frecuencia - V. Cualitativa

También se podría calcular las frecuencias usando `groub_by + summarise + mutate` o con `count + mutate`, con la ventaja de poder agregar más columnas

```
# Frecuencua con Tidyverse
data_banco %>%
  group_by(Transaccion) %>%
  count(name = "Frec") %>%
  ungroup() %>%
  mutate(
    Porc= round(Frec/ sum( Frec ) * 100, 2)
  )
```

```
## # A tibble: 3 x 3
##   Transaccion                  Frec  Porc
##   <chr>                       <int> <dbl>
## 1 Cobrar cheque (Cta del Bco)  5407  22.2
## 2 Cobro/Pago (Cta externa)     3005  12.4
## 3 Deposito                    15887  65.4
```

Tablas de Frecuencia - V. Cualitativa

También se podría calcular las frecuencias usando `groub_by + summarise + mutate` o con `count + mutate`, con la ventaja de poder agregar más columnas.

¿Y si queremos agregar la columna de frecuencia acumulada? Necesitamos saber entonces sobre **las funciones a rangos de filas**

```
# Frecuencua con Tidyverse
data_banco %>%
  group_by(Transaccion) %>%
  count(name = "Frec") %>%
  ungroup() %>%
  mutate(
    Porc= round(Frec/ sum( Frec ) * 100, 2)
  )
```

```
## # A tibble: 3 x 3
##   Transaccion                  Frec  Porc
##   <chr>                       <int> <dbl>
## 1 Cobrar cheque (Cta del Bco)  5407  22.2
## 2 Cobro/Pago (Cta externa)     3005  12.4
## 3 Deposito                    15887  65.4
```

Funciones a rangos de filas

Curso: Manejo de datos y reportería con R

Néstor Montaño

Ejemplo: Venta de empresa retail

Sólo para estos pocos ejemplos vamos a armar una estructura de datos típica de empresas de retail, donde se tiene columnas para el Almacen, Producto, Periodo (Año-Mes) y variables como la venta en dólares, el margen y en este caso el presupuesto.

```
set.seed(123)
df_1 <- data.frame(
  ALMACEN= rep(c("Mall del Sol", "Riocentro"), each= 6),
  PRODUCTO= paste("Prod", LETTERS[1:6], sep="_"),
  PERIODO= rep(seq.Date(from=as.Date("2015-01-01"),
                        to=as.Date("2017-06-01"), by="month"),
               each=12),
  VTA= runif(n = 360, min = 1000, max= 7000),
  MARGEN= rnorm(n = 360, mean = 30, sd= 6)/100,
  PPTO= runif(n = 360, min = 1000, max= 7000),
  stringsAsFactors = FALSE
)
df_1 %<>% mutate(VTA_COSTO=VTA*(1-MARGEN))
df_1 <- as_tibble(df_1)
```

Ejemplo: Venta de empresa retail

Sólo para estos pocos ejemplos vamos a armar una estructura de datos típica de empresas de retail, donde se tiene columnas para el Almacen, Producto, Periodo (Año-Mes) y variables como la venta en dólares, el margen y en este caso el presupuesto.

```
df_1
```

```
## # A tibble: 360 x 7
##    ALMACEN      PRODUCTO PERIODO      VTA MARGEN  PPTO VTA_COSTO
##    <chr>        <chr>    <date>     <dbl>  <dbl> <dbl>     <dbl>
##  1 Mall del Sol Prod_A   2015-01-01 2725.  0.236 3354.     2082.
##  2 Mall del Sol Prod_B   2015-01-01 5730.  0.376 2032.     3577.
##  3 Mall del Sol Prod_C   2015-01-01 3454.  0.279 2918.     2490.
##  4 Mall del Sol Prod_D   2015-01-01 6298.  0.248 3481.     4736.
##  5 Mall del Sol Prod_E   2015-01-01 6643.  0.286 5556.     4744.
##  6 Mall del Sol Prod_F   2015-01-01 1273.  0.288 2631.      906.
##  7 Riocentro    Prod_A   2015-01-01 4169.  0.367 1693.     2640.
##  8 Riocentro    Prod_B   2015-01-01 6355.  0.305 1176.     4416.
##  9 Riocentro    Prod_C   2015-01-01 4309.  0.345 5841.     2821.
## 10 Riocentro    Prod_D   2015-01-01 3740.  0.270 3881.     2730.
## # ... with 350 more rows
```

Funciones para rangos de filas

Obtener el *Ranking mensual de productos más vendidos por almacén*

```
df_1 %>%
  arrange(ALMACEN, PERIODO) %>%
  group_by(ALMACEN, PERIODO) %>%
  mutate(Prod_rank = min_rank(desc(VTA)))
```

```
## # A tibble: 360 x 8
## # Groups:   ALMACEN, PERIODO [60]
##    ALMACEN      PRODUCTO PERIODO      VTA MARGEN  PPTO VTA_COSTO Prod_rank
##    <chr>        <chr>    <date>     <dbl>  <dbl> <dbl>     <dbl>     <int>
##  1 Mall del Sol Prod_A   2015-01-01 2725.  0.236 3354.     2082.         5
##  2 Mall del Sol Prod_B   2015-01-01 5730.  0.376 2032.     3577.         3
##  3 Mall del Sol Prod_C   2015-01-01 3454.  0.279 2918.     2490.         4
##  4 Mall del Sol Prod_D   2015-01-01 6298.  0.248 3481.     4736.         2
##  5 Mall del Sol Prod_E   2015-01-01 6643.  0.286 5556.     4744.         1
##  6 Mall del Sol Prod_F   2015-01-01 1273.  0.288 2631.      906.         6
##  7 Mall del Sol Prod_A   2015-02-01 5065.  0.306 1692.     3517.         2
##  8 Mall del Sol Prod_B   2015-02-01 4436.  0.246 1776.     3343.         3
##  9 Mall del Sol Prod_C   2015-02-01 1618.  0.221 5432.     1260.         5
## 10 Mall del Sol Prod_D   2015-02-01 6399.  0.420 5543.     3712.         1
## # ... with 350 more rows
```

Funciones para rangos de filas

Porcentaje de venta menor a la venta del producto, para cada almacén y mes

```
df_1 %>%
  arrange(ALMACEN, PERIODO) %>%
  group_by(ALMACEN, PERIODO) %>%
  mutate(Porc = percent_rank(VTA))
```

```
## # A tibble: 360 x 8
## # Groups:   ALMACEN, PERIODO [60]
##    ALMACEN      PRODUCTO PERIODO      VTA MARGEN  PPTO VTA_COSTO  Porc
##    <chr>        <chr>    <date>     <dbl>  <dbl> <dbl>     <dbl> <dbl>
##  1 Mall del Sol Prod_A   2015-01-01 2725.  0.236 3354.     2082.   0.2
##  2 Mall del Sol Prod_B   2015-01-01 5730.  0.376 2032.     3577.   0.6
##  3 Mall del Sol Prod_C   2015-01-01 3454.  0.279 2918.     2490.   0.4
##  4 Mall del Sol Prod_D   2015-01-01 6298.  0.248 3481.     4736.   0.8
##  5 Mall del Sol Prod_E   2015-01-01 6643.  0.286 5556.     4744.   1
##  6 Mall del Sol Prod_F   2015-01-01 1273.  0.288 2631.      906.   0
##  7 Mall del Sol Prod_A   2015-02-01 5065.  0.306 1692.     3517.   0.8
##  8 Mall del Sol Prod_B   2015-02-01 4436.  0.246 1776.     3343.   0.6
##  9 Mall del Sol Prod_C   2015-02-01 1618.  0.221 5432.     1260.   0.2
## 10 Mall del Sol Prod_D   2015-02-01 6399.  0.420 5543.     3712.   1
## # ... with 350 more rows
```

Funciones para rangos de filas

Para el almacén y producto, la venta acumulada a cada cierre de mes empezando desde enero

```
df_1 %>%
  arrange(ALMACEN, PRODUCTO, PERIODO) %>%
  group_by(ALMACEN, PRODUCTO, ANIO=lubridate::year(PERIODO)) %>%
  mutate(VTA_ACUM = cumsum(VTA))
```

```
## # A tibble: 360 x 9
## # Groups:   ALMACEN, PRODUCTO, ANIO [36]
##    ALMACEN      PRODUCTO PERIODO      VTA MARGEN  PPTO VTA_COSTO  ANIO VTA_ACUM
##    <chr>        <chr>    <date>     <dbl>  <dbl> <dbl>     <dbl> <dbl>    <dbl>
##  1 Mall del Sol Prod_A   2015-01-01 2725.  0.236 3354.     2082.  2015    2725.
##  2 Mall del Sol Prod_A   2015-02-01 5065.  0.306 1692.     3517.  2015    7791.
##  3 Mall del Sol Prod_A   2015-03-01 4934.  0.275 6200.     3577.  2015   12725.
##  4 Mall del Sol Prod_A   2015-04-01 5551.  0.274 3531.     4032.  2015   18276.
##  5 Mall del Sol Prod_A   2015-05-01 2596.  0.353 1707.     1679.  2015   20872.
##  6 Mall del Sol Prod_A   2015-06-01 4991.  0.253 5991.     3730.  2015   25862.
##  7 Mall del Sol Prod_A   2015-07-01 5261.  0.279 5498.     3791.  2015   31124.
##  8 Mall del Sol Prod_A   2015-08-01 1617.  0.438 1712.      910.  2015   32741.
##  9 Mall del Sol Prod_A   2015-09-01 5694.  0.310 5320.     3926.  2015   38434.
## 10 Mall del Sol Prod_A   2015-10-01 3464.  0.287 5157.     2470.  2015   41899.
## # ... with 350 more rows
```

Funciones para rangos de filas

Para el almacén y producto, el promedio de venta mensual del año a cada cierre de mes empezando desde enero y el promedio de los últimos 3 meses

```
library(RcppRoll)
df_1 %>%
  arrange(ALMACEN, PRODUCTO, PERIODO) %>%
  group_by(ALMACEN, PRODUCTO, ANIO=lubridate::year(PERIODO)) %>%
  mutate(PROM_ESTE_ANIO = cummean(VTA) ) %>%
  group_by(ALMACEN, PRODUCTO) %>%
  mutate(PROM_ULT_TRIM = roll_mean(VTA, n = 3, fill= NA, align = "right" ) ) %>%
  select(ALMACEN, PRODUCTO, VTA, PROM_ULT_TRIM) %>% head(3)
```

```
## # A tibble: 3 x 4
## # Groups:   ALMACEN, PRODUCTO [1]
##   ALMACEN      PRODUCTO   VTA PROM_ULT_TRIM
##   <chr>        <chr>    <dbl>         <dbl>
## 1 Mall del Sol Prod_A   2725.           NA
## 2 Mall del Sol Prod_A   5065.           NA
## 3 Mall del Sol Prod_A   4934.         4242.
```

Funciones para rangos de filas

Ahora lo mismo pero también, para cada producto-almacén, la diferencia de la venta del mes versus el mes anterior

```
df_1 %>%
  arrange(ALMACEN, PRODUCTO, PERIODO) %>%
  group_by(ALMACEN, PRODUCTO, ANIO=lubridate::year(PERIODO)) %>%
  mutate(PROM_ESTE_ANIO = cummean(VTA) ) %>%
  select(ALMACEN, PRODUCTO, VTA) %>%
  group_by(ALMACEN, PRODUCTO) %>%
  mutate(PROM_ULT_TRIM = roll_mean(VTA, n = 3, fill= NA, align = "right" ) ,
         DIFF_VS_MES_ANT= VTA - lag(VTA),
         PORC_DIFF_VS_MES_ANT= (VTA - lag(VTA))/lag(VTA)
         ) %>%
  head(3)
```

```
## Adding missing grouping variables: `ANIO`
```

```
## # A tibble: 3 x 7
## # Groups:   ALMACEN, PRODUCTO [1]
##    ANIO ALMACEN   PRODUCTO   VTA PROM_ULT_TRIM DIFF_VS_MES_ANT PORC_DIFF_VS_MES~
##   <dbl> <chr>     <chr>    <dbl>         <dbl>           <dbl>              <dbl>
## 1  2015 Mall del~ Prod_A   2725.           NA              NA              NA
## 2  2015 Mall del~ Prod_A   5065.           NA            2340.              0.859
## 3  2015 Mall del~ Prod_A   4934.         4242.           -131.             -0.0259
```

FIN

Curso: Manejo de datos y reportería con R

Néstor Montaño