

# Limpieza de datos Pt. 1

## Importación y manipulación de datos



Daniel Sánchez Pazmiño

# El proceso Tidyverse

# El proceso Tidyverse: Los primeros pasos en análisis de datos



# El proceso Tidyverse: Los primeros pasos en análisis de datos



- **Import:** Obtener y entender los datos
- **Tidy:** Ordenar los datos de tal manera que sea sencillo transformarlos, sumarizarlo, visualizarlos o realizar un modelo con ellos
- **Transform:** Manipular los datos hasta obtener el input que el análisis o técnica estadística necesita
- **Visualize:** Realizar el análisis exploratorio de datos
- **Model:** Aplicar técnicas estadísticas para el entendimiento del problema o tomar decisiones

# Importando datos

# Importando datos: Importar csv

Desde RStudio (R-base o paquete 'readr'):

Import Dataset > From Text File > Escoger archivo > Abrir > Escribir nombre a la variable > Import

Con comando:

```
read.csv( file, sep = ",", dec = ",", stringsAsFactors= FALSE)
```

Para grandes volúmenes de datos usar paquete 'data.table':

```
fread()
```

Paquete 'vroom': actualmente ya en tidyverse

<https://www.tidyverse.org/blog/2019/05/vroom-1-0-0/>

<https://vroom.r-lib.org/>

# Importando datos: Importar desde excel

Copiando desde un archivo de excel abierto:

```
read.table("clipboard", sep="\t", header=TRUE)
```

Desde RStudio:

Rstudio > Import Dataset > From Excel > Escoger archivo > Abrir > Escribir nombre a la variable > Import

Usando el paquete 'openxlsx':

```
read.xlsx(xlsxFile , sheet , startRow , colNames , skipEmptyRows, rowNames)  
data_tiempo_espera <- read.xlsx(xlsxFile = 'Data/Data_Banco.xlsx')
```

Otros paquetes

```
excel.link, XLConnect, xlsx, readxl, rio
```

# Importando datos: Importar desde SPSS, SAS, Stata, etc

## Desde RStudio:

Rstudio > Import Dataset > From SPSS/SAS/STATA

## Usando el paquete 'foreign':

SAS: `read.xport()`

SPSS: `read.spss()`

Stata: `read.dta()`

Soporta otros formatos

## Usango el paquete 'haven':

SAS: `read_sas()` y `read_xpt()`

SPSS: `read_sav()` y `read_por()`

Stata: `read_dta()`

Se puede usar el paquete 'rio'

## Otros paquetes

# Ejemplo - Data de transacciones bancarias

## Ejemplo: Data de transacciones bancarias

Un banco 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: Data de transacciones bancarias

Revisar archivo de excel: Data\_Banco.xlsx

1. Crear un proyecto en RStudio, con las carpetas Data, Exports, etc
2. Poner en la carpeta 'Data', el excel suministrado

```
# Cargar la librería a utilizar
library(openxlsx)
library(here)

# Leer el archivo de excel y asignarlo al objeto data_banco
data_banco <- read.xlsx(xlsxFile=here("spanish/slides/limpieza_pt1/data/Data_Banco.xlsx"), sheet
data_sucursal <- read.xlsx(xlsxFile =here("spanish/slides/limpieza_pt1/data/Data_Banco.xlsx"), sh
```

# Explorando la data

## ¿Qué tipo de estructura hemos importado?

```
str(data_banco)
```

```
## 'data.frame': 24299 obs. of 7 variables:
## $ Sucursal      : num  62 62 62 62 62 62 62 62 62 ...
## $ Cajero        : num  4820 4820 4820 4820 4820 4820 4820 4820 4820 ...
## $ ID_Transaccion: chr  "2" "2" "2" "2" ...
## $ Transaccion   : chr  "Cobro/Pago (Cta externa)" "Cobro/Pago (Cta externa)" "Cobro/Pago (Cta ex...
## $ Tiempo_Servicio_seg: num  311 156 248 99 123 172 140 247 183 91 ...
## $ Satisfaccion  : chr  "Muy Bueno" "Malo" "Regular" "Regular" ...
## $ Monto         : chr  "2889,3" "1670,69" "3172,49" "1764.92" ...
```

# Entender los datos

**Luego de importar se debe entender los datos**

- ¿Qué representa cada columna?
- ¿Qué tipo de dato debería tener cada columna?
- ¿Qué granularidad o atomicidad tiene la data?
- Si es que se tiene varios conjuntos de datos, ¿Cómo se relacionan los datos?
- A qué periodo de tiempo corresponde la data
- Muchas veces se obtiene la información desde una base de datos y por tanto toca entender la base y el query que genera los datos

```
# Podríamos ver las primeras 5 filas
head(data_sucursal, n = 5)

# Listar los nombres de las columnas
names(data_banco)
names(data_sucursal)
```

```
# Podríamos ver las primeras 5 filas  
head(data_sucursal, n = 5)
```

```
##   ID_Sucursal      Sucursal Nuevo_Sistema  
## 1          62  Riocentro Sur            No  
## 2          85       Centro            Si  
## 3         267    Alborada            Si  
## 4         443  Mall del Sol            Si  
## 5         586     Via Daule            No
```

```
# Listar los nombres de las columnas  
names(data_banco)
```

```
## [1] "Sucursal"           "Cajero"             "ID_Transaccion"      "Transaccion"        "Tiempo_Se  
## [6] "Satisfaccion"        "Monto"
```

```
names(data_sucursal)
```

```
## [1] "ID_Sucursal"    "Sucursal"        "Nuevo_Sistema"
```

# Manipulación de datos: Básico

R tiene sus comandos predeterminados para manipular datos, esto se conoce como 'R Base', sin embargo existen varios paquetes que simplifican esta tarea, en este curso veremos como hacerlo con el paquete `dplyr` (y `magrittr`) que están dentro del conjunto de paquetes llamado `tidyverse`

```
# Cargar la librería
library(tidyverse)
```

# Manipulación de datos: Básico

## Tibbles (un dataframe mejorado):

Tibble es un objeto del paquete dplyr, entre las mejoras que da es que no imprime todo el objeto en pantalla, sino un resumen del mismo. (más información tipeando ?tibble)

```
# Convertir el data_banco a un tibble
data_banco <- tibble:::as_tibble(data_banco)
# Muestra data_banco
head(data_banco, n = 5)
```

```
## # A tibble: 5 × 7
##   Sucursal Cajero ID_Transaccion Transaccion      Tiempo_Servicio_seg Satisfaccion Monto
##     <dbl>   <dbl>      <chr>        <chr>                  <dbl>      <chr>       <chr>
## 1       62     4820  Cobro/Pago (Cta externa) 311  Muy Bueno  2889,3
## 2       62     4820  Cobro/Pago (Cta externa) 156  Malo        1670,69
## 3       62     4820  Cobro/Pago (Cta externa) 248  Regular     3172,49
## 4       62     4820  Cobro/Pago (Cta externa)  99  Regular     1764.92
```

## Operador Pipe: %>%

El operador Pipe `%>%` del paquete `magrittr` permiten que el código sea más legible porque:

- Permite secuencias estructurantes de operaciones de datos de izquierda a derecha (a diferencia de dentro y fuera)
- Evitando llamadas a funciones anidadas
- Minimiza la necesidad de variables locales y definiciones de funciones
- Facilita agregar pasos en cualquier lugar de la programación

# Operador Pipe: %>%

```
library(magrittr)
data_banco %>% names

## [1] "Sucursal"           "Cajero"              "ID_Transaccion"      "Transaccion"         "Tiempo_Se
## [6] "Satisfaccion"       "Monto"

data_banco %>% dim

## [1] 24299      7

data_banco %>% names %>% length

## [1] 7
```

## Operador Pipe: %>%

Con la última línea de código le estamos diciendo a R que:

- tome el objeto `data_banco`
- luego (`%>%`)
- que entregue los nombres de las variables del objeto
- luego (`%>%`)
- que entregue la longitud (`length`) del objeto

# Comparación de sintaxis

Ambas dan el mismo resultado pero una es menos intuitiva que otra, ¿cuál?

```
library(magrittr)
data_banco %>% names %>% length
```

```
## [1] 7
```

```
#sintaxis R base
length(names(data_banco)) # equivalencia del código anterior
```

```
## [1] 7
```

# Los verbos del Tidyverse

# Filtrar filas: filter()

Filtrar las filas según las condiciones dadas en `filter()`

```
# Filtrar las filas correspondientes a la sucursal 62
data_banco %>% filter( Sucursal== 62 ) %>% View
# Filtrar las filas correspondientes a la sucursal 62 y hayan durado más de 120 segundos
data_banco %>% filter( Sucursal== 62 & Tiempo_Servicio_seg > 120 ) %>% View
# Filtrar las filas correspondientes a la sucursal 62, hayan durado
# más de 120 segundos y la evaluación a la satisfacción sea Bueno
data_banco %>% filter( Sucursal== 62 & Tiempo_Servicio_seg > 120 & Satisfaccion== "Muy Bueno" ) %>% View
```

## Filtrar filas y seleccionar

```
# Con el data banco
# Filtrar las filas correspondientes a la sucursal 85
# calcular la correlacion entre Tiempo_Servicio_seg y Monto
data_banco %>% # Operador pipe total
  filter( Sucursal== 85 ) %$% # Operador pipe para seleccion de columnas
  cor(Tiempo_Servicio_seg, as.numeric(Monto))
```

```
## [1] 0.5339392
```

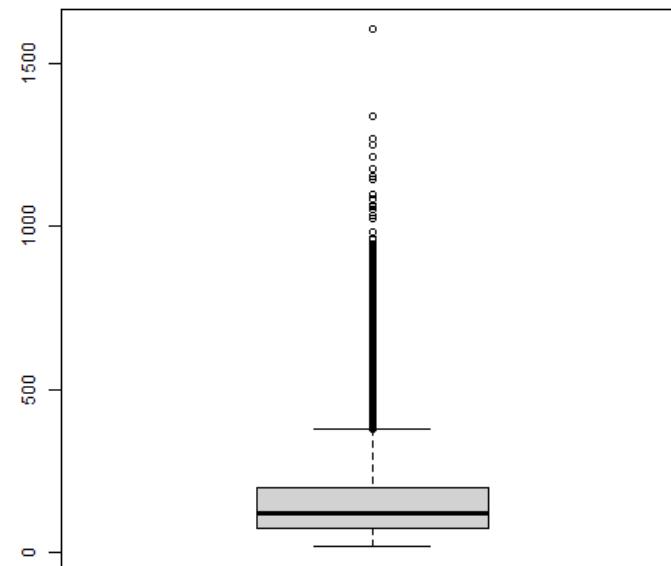
## Seleccionar columnas: select()

Seleccionar las columnas Transaccion, Tiempo\_Servicio\_seg del data frame 'data\_banco' pero usando `%>%`, lo que permite programar como si se escribiese "del data\_banco, selecciona las columnas Transaccion y Tiempo\_Servicio\_seg"

```
# Note que como no se asignó, R evalúa la expresión y presenta el resultado
# data_banco[ , c("Transaccion", "Tiempo_Servicio_seg") ] ## Base de R
data_banco %>% select(Transaccion, Tiempo_Servicio_seg) %>% head

## # A tibble: 6 × 2
##   Transaccion           Tiempo_Servicio_seg
##   <chr>                  <dbl>
## 1 Cobro/Pago (Cta externa)     311
## 2 Cobro/Pago (Cta externa)     156
## 3 Cobro/Pago (Cta externa)     248
## 4 Cobro/Pago (Cta externa)      99
## 5 Cobro/Pago (Cta externa)    123
```

```
data_banco %>%
  select(Tiempo_Servicio_seg) %>%
  boxplot
```



```
# boxplot(data_banco$Tiempo_Servicio_seg) ## Base de R
data_banco %>%
  select(Tiempo_Servicio_seg) %>%
  boxplot
```

## Seleccionar columnas: select()

Seleccionar y ver en el visor de datos de RStudio

```
# Seleccionar y ver en el visor de datos de RStudio
data_banco %>% select(Transaccion, Tiempo_Servicio_seg) %>% View
```

Seleccionar todas las columnas menos Cajero

```
# Seleccionar todas las columnas menos Cajero
data_banco %>% select(-Cajero) %>% View
```

# select() y sus funciones auxiliares

Seleccionar según nombre de la columna/variable

```
# Seleccionar todas las columnas cuyo nombre contenga el texto "Tra"
data_banco %>% select( contains("Tra")) %>% View
# Seleccionar todas las columnas cuyo nombre inicie con "S"
data_banco %>% select( starts_with("S")) %>% View
# Seleccionar todas las columnas cuyo nombre finalice con "on"
data_banco %>% select( ends_with("on")) %>% View
# Seleccionar todas las columnas cuyo nombre contenga una "r" o un "sa"
data_banco %>% select( matches("r?sa")) %>% View
# Más información sobre expresiones regulares usando: ?base:::regex
```

## Ordenar las filas: `arrange()`

Ordenar las filas según lo expresado en `arrange()`

```
# Ordenar por la satisfaccion
data_banco %>% arrange( Satisfaccion ) %>% View
# Ordenar cada Transaccion y dentro de cada transaccion de mayor a menor por tiempo de servicio
data_banco %>% arrange( Transaccion, desc(Tiempo_Servicio_seg) ) %>% View
```

Por defecto, `arrange()` organiza los datos de menor a mayor.

# Crear o modificar columnas/variables: mutate()



Crear una nueva columna con el tiempo en minutos. Nótese que **no se asignó**, el objeto `data_banco` no tiene la columna `Tiempo_Servicio_Min`

```
# Crear una nueva columna con el tiempo en minutos
data_banco %>% mutate(Tiempo_Servicio_Min= Tiempo_Servicio_seg/60) %>% head

## # A tibble: 6 × 8
##   Sucursal Cajero ID_Transaccion Transaccion      Tiempo_Servicio_seg Satisfaccion Monto     Tie
##       <dbl>  <dbl>    <chr>          <chr>                  <dbl>    <chr>        <chr>
## 1       62     4820  2             Cobro/Pago (Cta externa)      311  Muy Bueno  2889,3 
## 2       62     4820  2             Cobro/Pago (Cta externa)      156  Malo      1670,69 
## 3       62     4820  2             Cobro/Pago (Cta externa)      248  Regular   3172,49 
## 4       62     4820  2             Cobro/Pago (Cta externa)      99   Regular   1764.92 
## 5       62     4820  2             Cobro/Pago (Cta externa)     123  Muy Bueno  1835.69 
## 6       62     4820  2             Cobro/Pago (Cta externa)     172  Bueno     2165.42
```

## Crear o modificar columnas/variables: mutate()

```
# Crear una nueva columna en data_banco con el tiempo en minutos
data_banco <- data_banco %>%
  mutate(Tiempo_Servicio_Min= Tiempo_Servicio_seg/60)
# Mostrar primeras 6 filas
head(data_banco)
```

```
## # A tibble: 6 × 8
##   Sucursal Cajero ID_Transaccion Transaccion      Tiempo_Servicio_seg Satisfaccion Monto     Tie
##   <dbl>    <dbl>    <chr>           <dbl> <chr>        <dbl>
## 1       62     4820 2 Cobro/Pago (Cta externa) 311  Muy Bueno  2889,3
## 2       62     4820 2 Cobro/Pago (Cta externa) 156  Malo       1670,69
## 3       62     4820 2 Cobro/Pago (Cta externa) 248  Regular    3172,49
## 4       62     4820 2 Cobro/Pago (Cta externa)  99  Regular    1764,92
## 5       62     4820 2 Cobro/Pago (Cta externa) 123  Muy Bueno  1835,69
## 6       62     4820 2 Cobro/Pago (Cta externa) 172  Bueno      2165,42
```

## Nuevas columnas: `transmute()`

Para conservar solamente las nuevas columnas se usa `transmute()`

```
# Crear una nueva columna con el tiempo en minutos
data_banco %>%
  transmute(Tiempo_Servicio_Min= Tiempo_Servicio_seg/60) %>%
  head
```

```
## # A tibble: 6 × 1
##   Tiempo_Servicio_Min
##       <dbl>
## 1      5.18
## 2      2.6
## 3      4.13
## 4      1.65
## 5      2.05
## 6      2.87
```

## Crear resúmenes: summarise()

`summarise()` permite aplicar funciones a nuestro `data.frame`, en R-base se usa `tapply()`, otra opción es `ddply()` del paquete 'plyr'.

```
# Obtener la media 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),
    CANTIDAD= n()
  )

## # A tibble: 1 × 3
##   MEDIA MEDIA_ACOT CANTIDAD
##     <dbl>      <dbl>     <int>
## 1    156.       142.     24299
```

# Crear resúmenes para datos agrupados

Obtener medidas de tendencia central para el tiempo de servicio para cada Transacción

```
# Obtener medidas de tendencia central para el tiempo de servicio para cada tipo de transaccion
data_banco %>%
  group_by(Transaccion) %>%
  summarise_at( vars(Tiempo_Servicio_seg),
    funs (
      MEDIA= mean(., na.rm=TRUE),
      MEDIA_ACOT= mean(., na.rm = TRUE, trim = 0.05),
      CANTIDAD= n())))
## # A tibble: 3 × 4
##   Transaccion          MEDIA  MEDIA_ACOT  CANTIDAD
##   <chr>              <dbl>     <dbl>     <int>
## 1 Cobrar cheque (Cta del Bco) 186.     175.     5407
## 2 Cobro/Pago (Cta externa) 301.     285.     3005
```