



Tecnológico
de Monterrey

03. Ciencia de datos

Ciencia de datos para la toma de decisiones II

Jorge
Juvenal
Campos Ferreira

 juvenal.campos@tec.mx

Programa de la sesión de hoy

- ¿Qué es **Ciencia de datos**?
- **Tareas** que se hacen con datos
- **Estructura** de un código
- **Librerías** en R
- **Cargar archivos** de datos
- **Procesamiento** de datos con tidyverse

¿Qué es un dato?

Desde la ciencia de datos

Son: “*una colección de hechos, cifras, palabras, observaciones u otra información útil*” que deben ser procesados para transformarse en insights valiosos que mejoran la toma de decisiones.

—IBM

Son una “*representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa*”.

—Wikipedia

Desde las ciencias sociales

Un dato es una **representación de la realidad social obtenida a través de un proceso de observación o medición**. No es solo una cifra aislada, sino un registro que adquiere sentido dentro de un marco conceptual

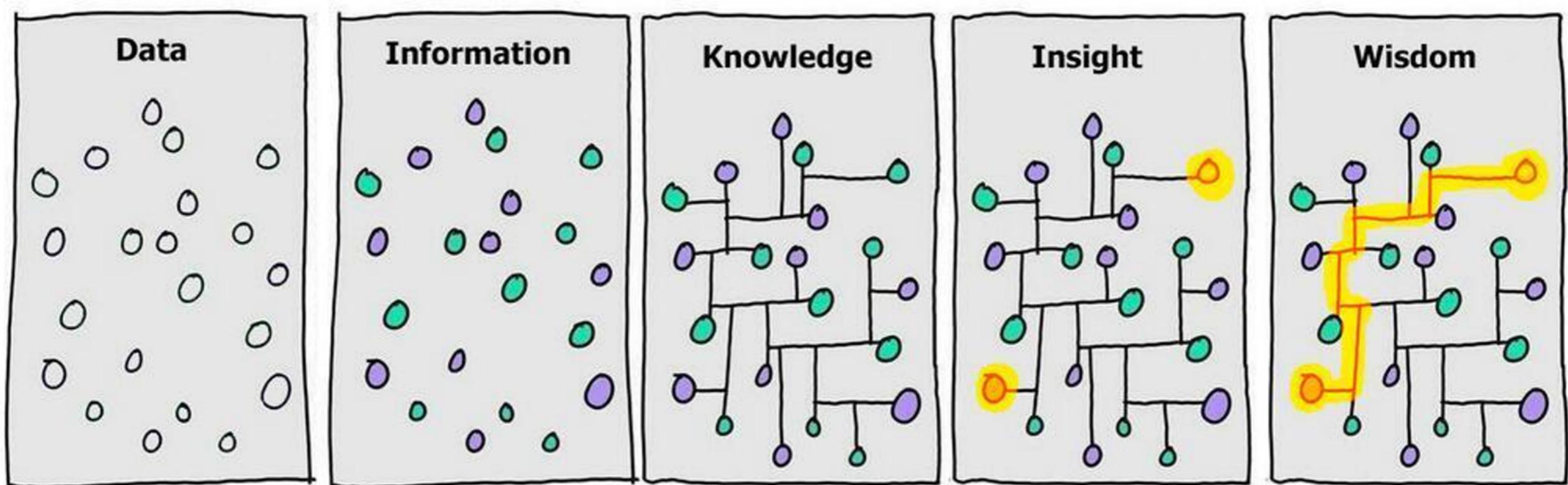
¿Qué es la ciencia de datos?

La **ciencia de datos** es un campo interdisciplinario que combina **estadística, matemáticas, programación, inteligencia artificial** y **conocimiento del dominio** para extraer información útil y generar conocimiento a partir de datos.

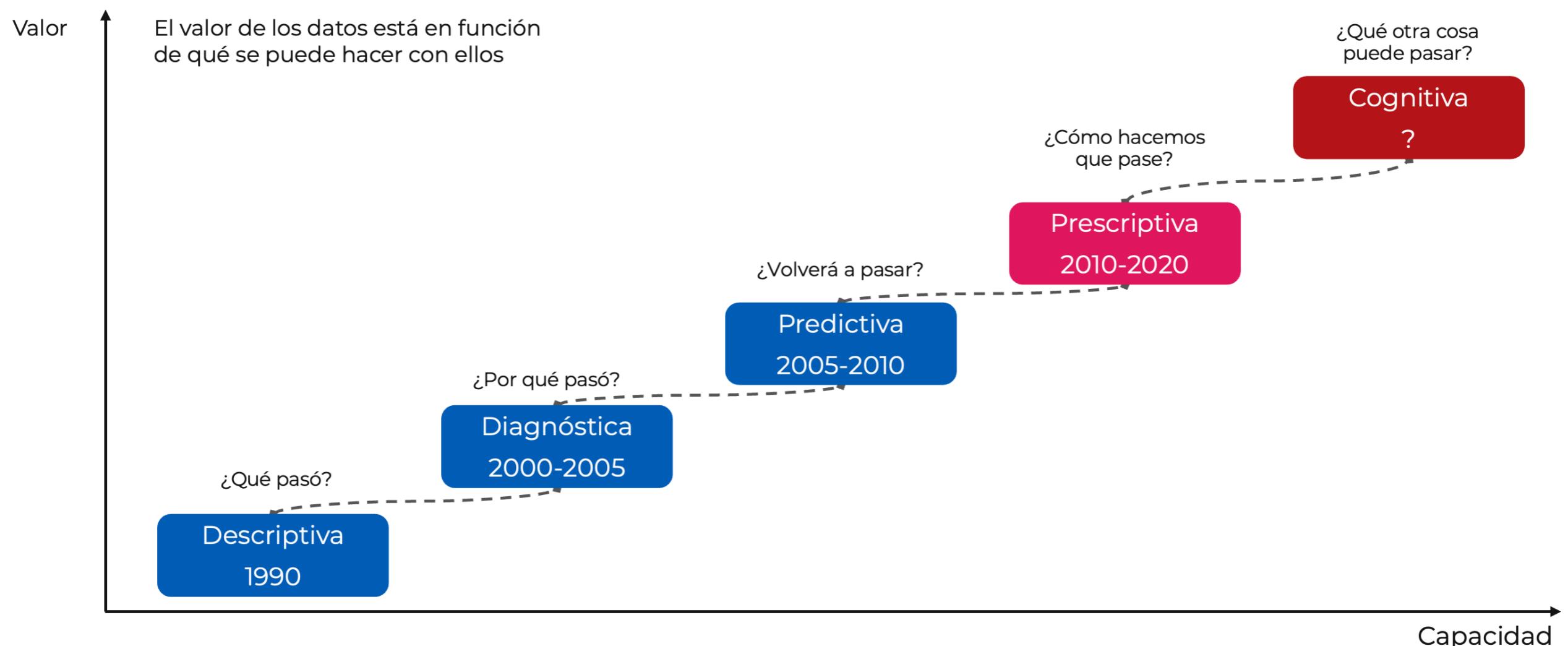


¿Cuál es el objetivo de la Ciencia de datos?

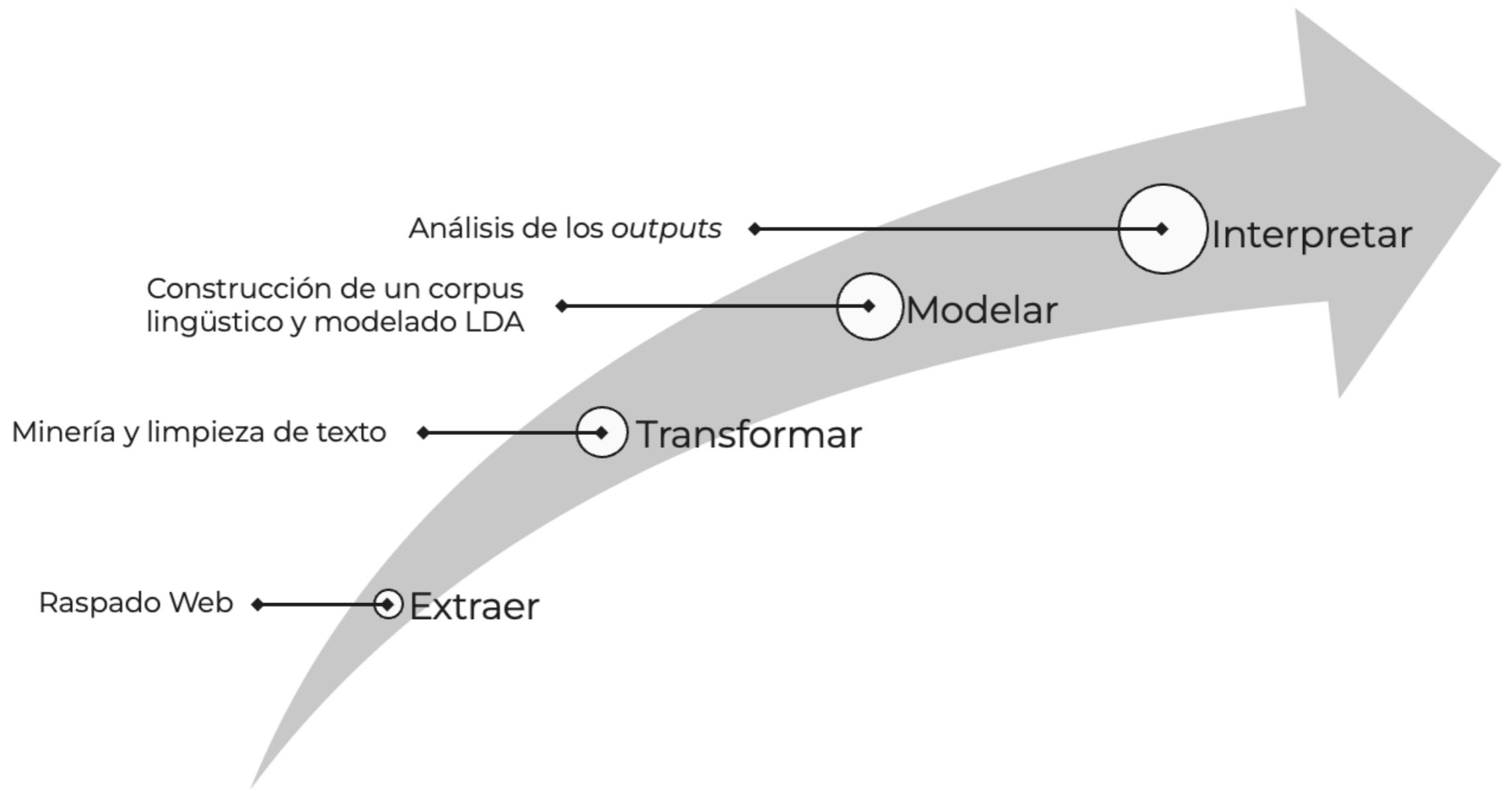
El objetivo principal de la **Ciencia de datos** es transformar grandes volúmenes de datos, muchas veces desordenados o no estructurados, en **hallazgos prácticos** que sirvan para la toma de decisiones.



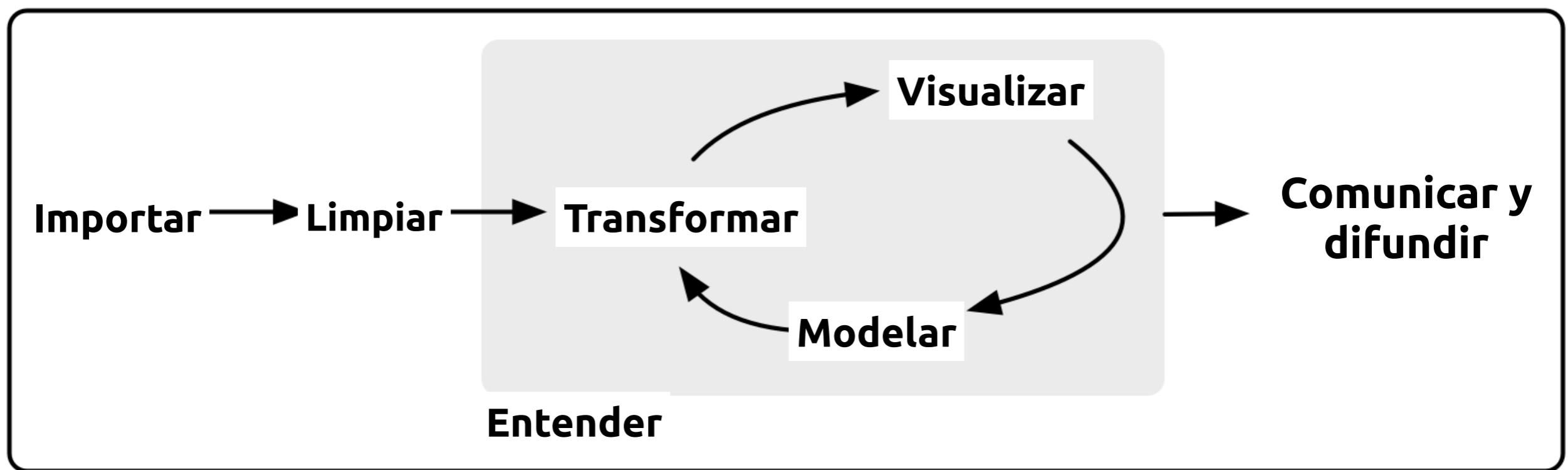
El valor de los datos en la política pública



El proceso del trabajo con datos



El proceso del trabajo con datos



Programar

¿Por qué aprender a programar?

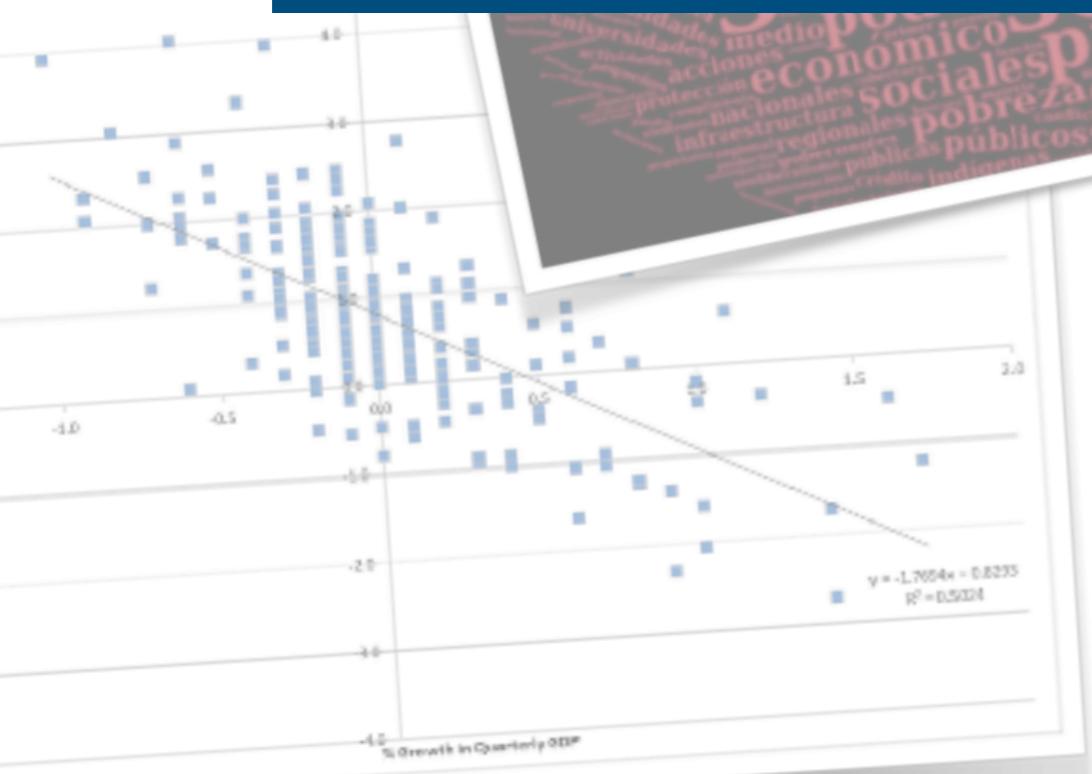
- 1. Actualmente, los datos son enormes**, por lo que se requieren herramientas especializadas para manejarlos
- 2. Automatización = más tiempo para pensar.** Programar te permite automatizar tareas para dedicarte a otras actividades
- 3. El mercado laboral demanda habilidades de programación.** R, Python y SQL entre lo más demandado
- 4. Permite hacer análisis, experimentar y simular escenarios**
- 5. Te da independencia y poder** para realizar tus propios cálculos

La explosión de los datos

Datos generados anualmente en el mundo



Tareas que se hacen con datos

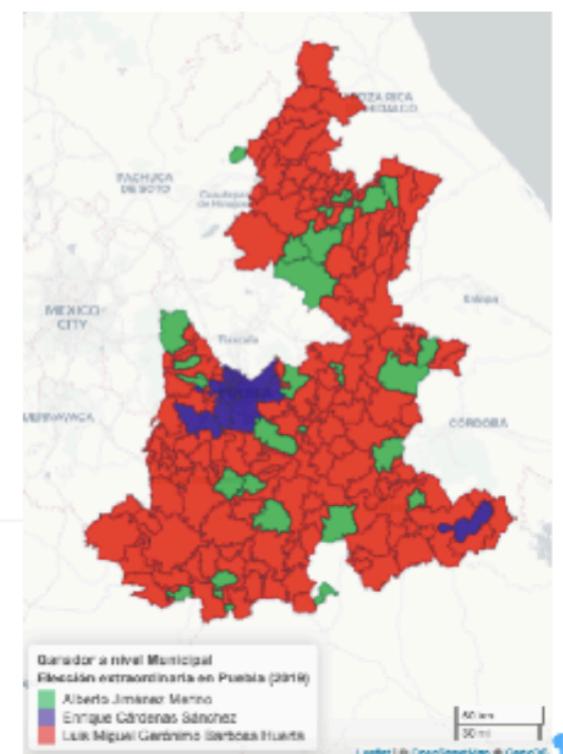


Tareas que se hacen con datos

1. Manejo y manipulación de datos en R.

```
library(tidyverse)
```

```
72   datos <- prep %>%
73     select(ECS, AJM, LMGBH, TOTAL_VOTOS, LISTA_NOMINAL, MUNICIPIO, DISTRITO) %>%
74     filter(!is.na(MUNICIPIO)) %>%
75     group_by(MUNICIPIO) %>%
76     summarise(ECS = sum(ECS, na.rm = TRUE),
77               AJM = sum(AJM, na.rm = TRUE),
78               LMGBH = sum(LMGBH, na.rm = TRUE),
79               Total_Votos = sum(TOTAL_VOTOS, na.rm = TRUE),
80               ListaNominal = sum(LISTA_NOMINAL, na.rm = TRUE)
81 )
```

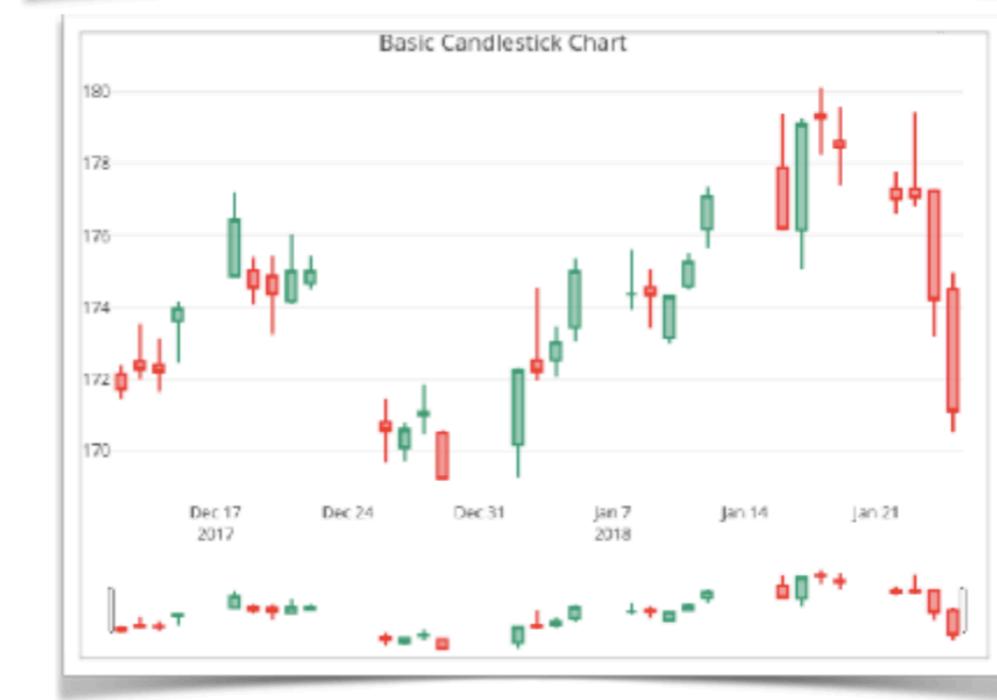
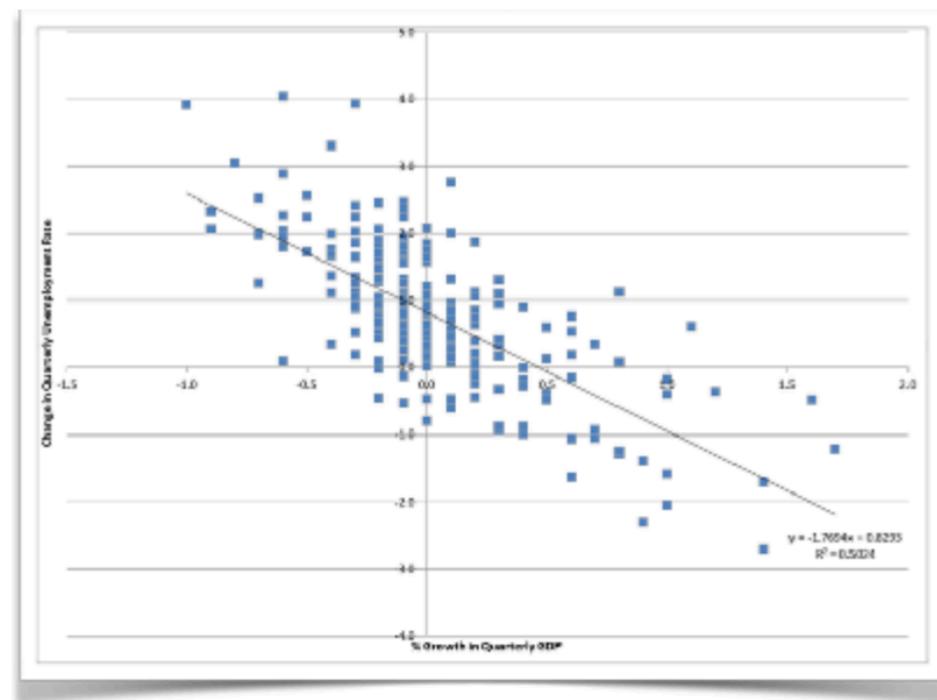
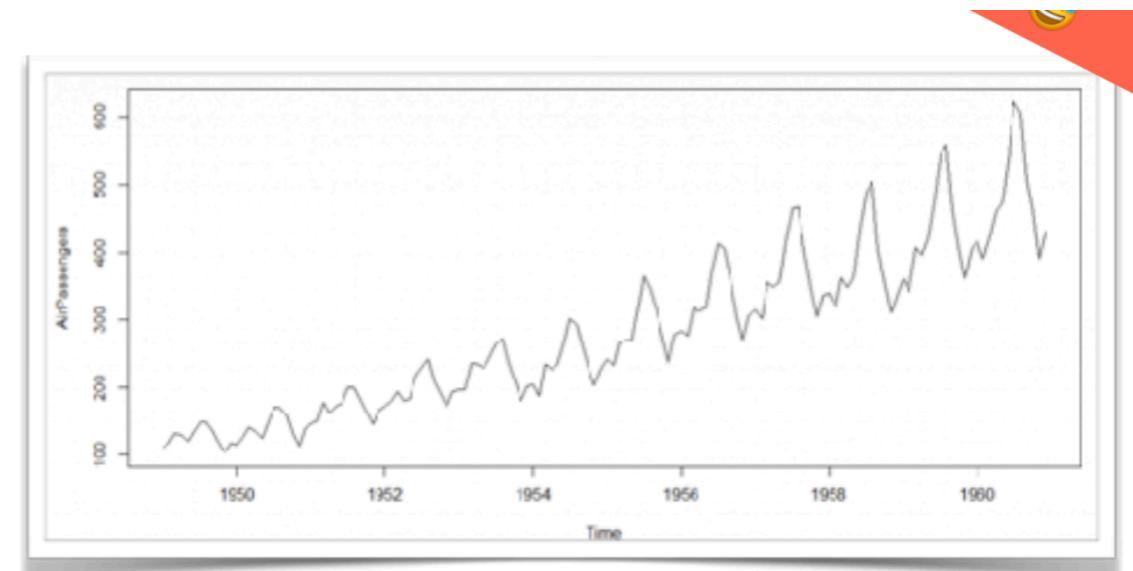


Tareas que se hacen con datos

2. Análisis estadístico y econometría.

library(base)

library(MASS)



Tareas que se hacen con datos

3. Machine Learning y Deep Learning.

`library(e1071)`

`library(tensorflow)`

`library(caret)`

`library(rpart)`

etc.

Classification



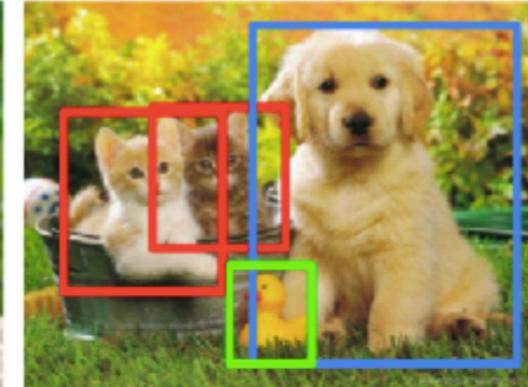
CAT

Classification + Localization



CAT

Object Detection



CAT, DOG, DUCK

Instance Segmentation



CAT, DOG, D

Single object

Multiple objects



Tareas que se hacen con datos

4. Análisis de texto.

library(tm)

```
library(stringr)
```



Nube de palabras.

Solicitudes de Acceso a información realizadas en el Estado de Morelos.



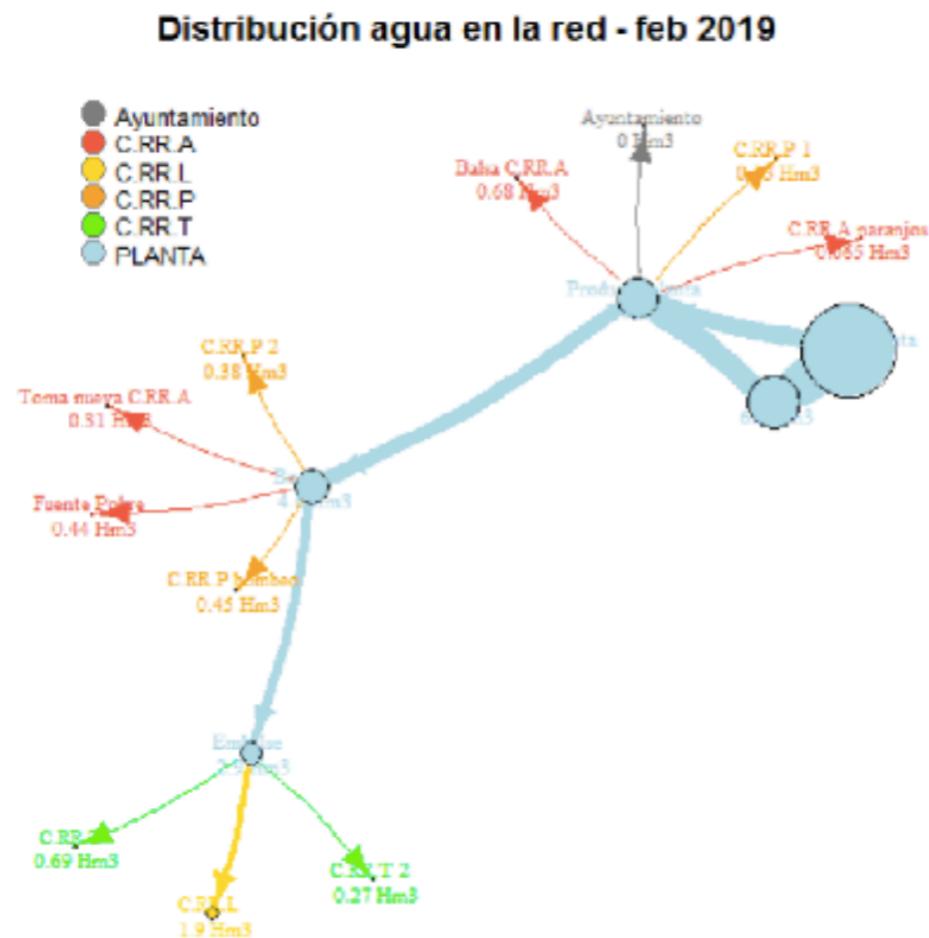
Nube de palabras.

Plan Nacional de Desarrollo. 2019.

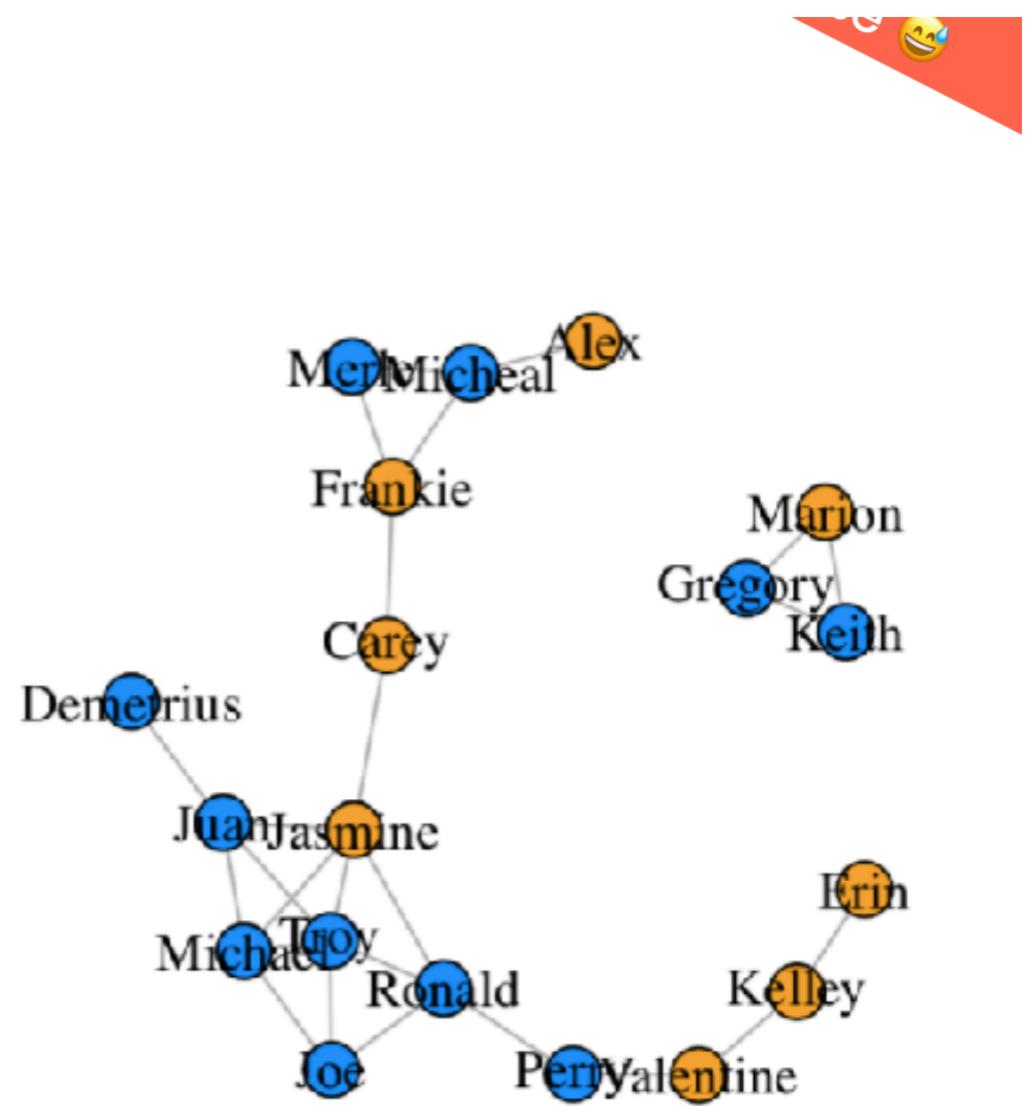
Tareas que se hacen con datos

5. Análisis de redes.

`library(igraph)`



Red de distribución de Agua



Red social de amigos en una prepa

Tareas que se hacen con datos



6. Visualización de datos.

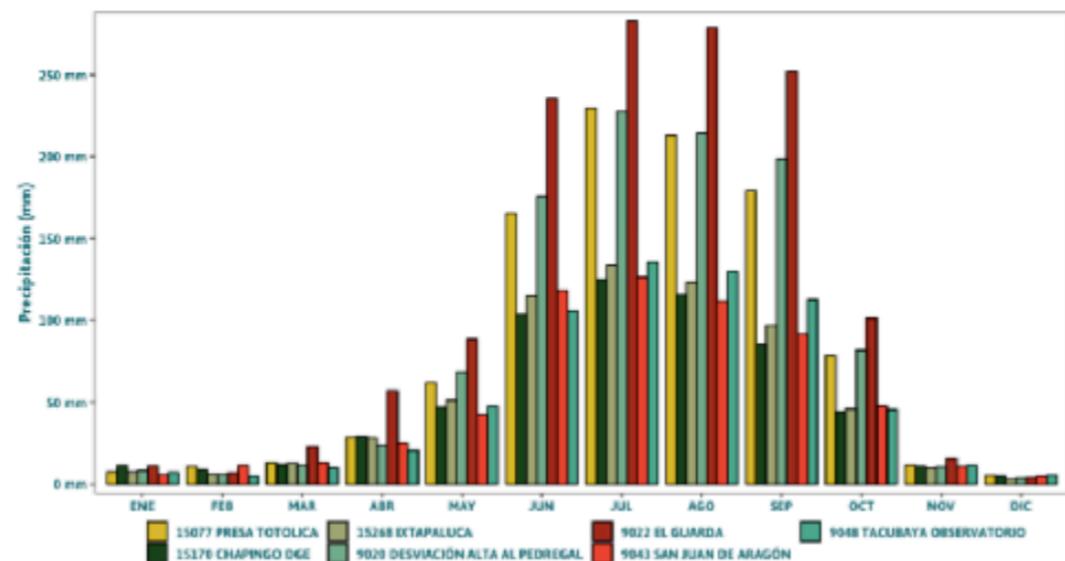
library(ggplot2)

library(plotly)

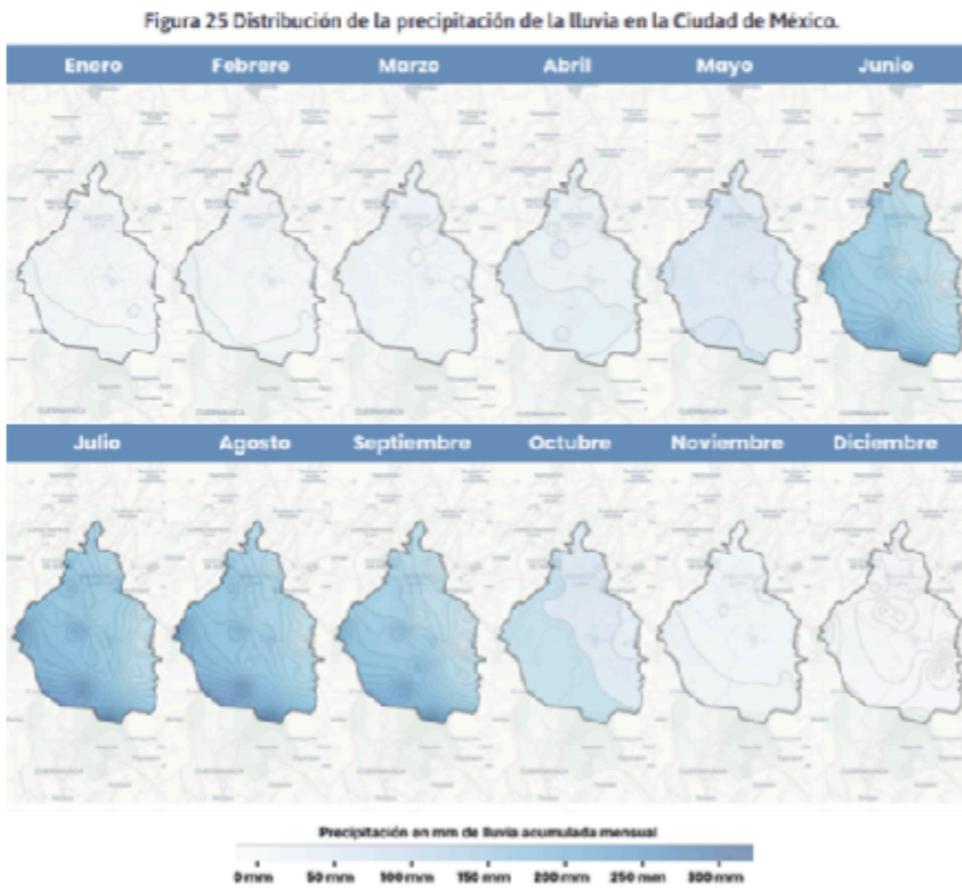
library(leaflet)

library(htmlwidgets)

Figura 24 Precipitación mensual promedio en 7 estaciones seleccionadas de la ZMVM.



Gráfica de barras de la distribución de la lluvia en la CDMX, en el tiempo.



Fuente: Elaboración Propia con datos del Atlas de Riesgo de la Ciudad de México, SGIRPC, 2019.

Mapas de la distribución de la lluvia en la CDMX, en el espacio y tiempo.

Tareas que se hacen con datos

7. Recolección automática de información (Web Scrapping, Data Crawling).

library(rvest)

library(xml)

Ejemplo: Extraer precios e información de vuelos desde Google Flights

The screenshot shows the Google Flights interface. On the left, there's a sidebar with icons for Trips, Explore, and Hotels. The main area displays "Best departing flights" with four results:

Airline	Flight Details	Duration	Stops	Price
United - ANA	06:05 - 14:35*	18 h 30 m	1 stop 2 h 35 m SFO	MX\$20,089 round trip
United, ANA	07:30 - 15:20*	17 h 50 m	1 stop 1 h 35 m IAH	MX\$20,089 round trip
ANA	02:20 - 06:45*	14 h 25 m	Non-stop MEX-NRT	MX\$21,889 round trip
Aeromexico - UA	01:30 - 06:20*	14 h 50 m	Non-stop MEX-NRT	MX\$31,471 round trip

Below this, a note says "Prices are currently typical for your trip." and there's a "Details" button. Further down, under "Other departing flights", there are two additional entries:

Airline	Flight Details	Duration	Stops	Price
United, ANA	17:30 - 14:20 ?	30 h 50 m	2 stops ▲ IAU, ORB	MX\$20,089 round trip
United, ANA	17:30 - 14:20 ?	30 h 50 m	2 stops ▲ IAU, ORB	MX\$20,089 round trip

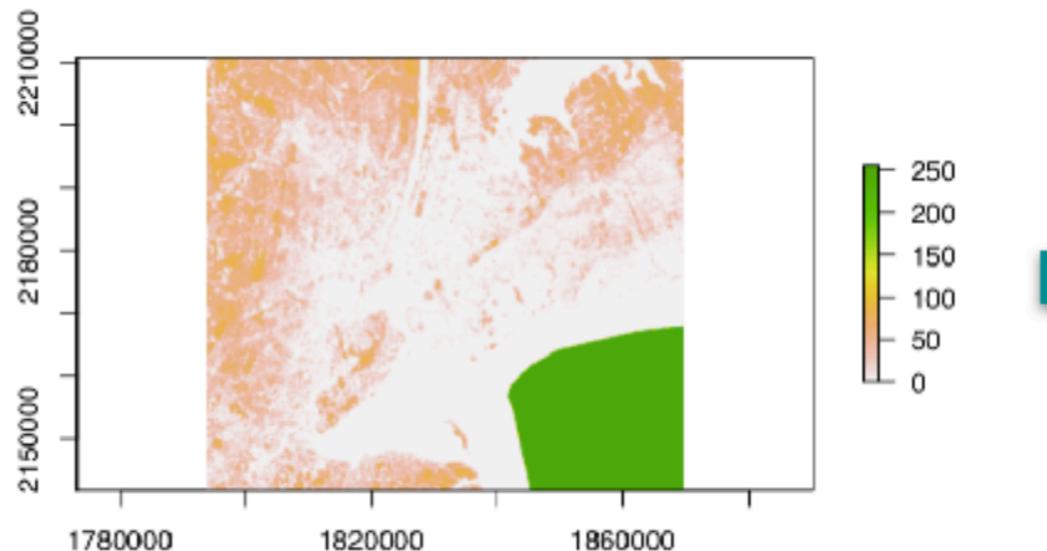
https://www.google.com/flights?lite=0#flt=MEX.NRT.2019-10-21*NRT.MEX.2019-11-05;c:MXN;e:1;sd:1;t:f

Tareas que se hacen con datos

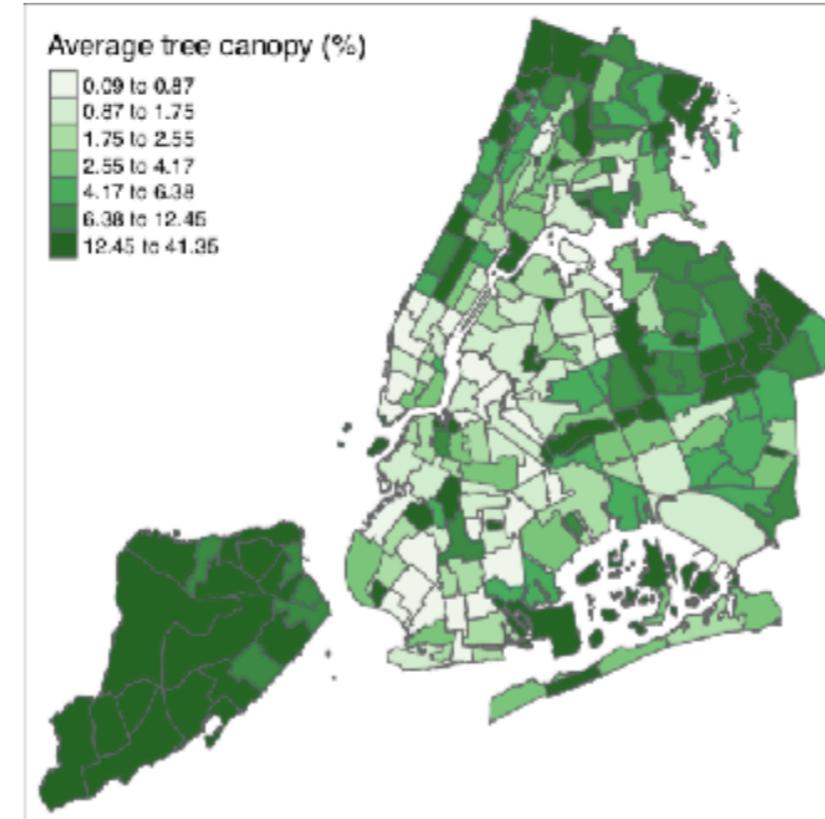
8. Análisis Geoespacial.

`library(sf)`

Abrir información geográfica, modificarla y visualizarla, así como realizar análisis a partir de esta.



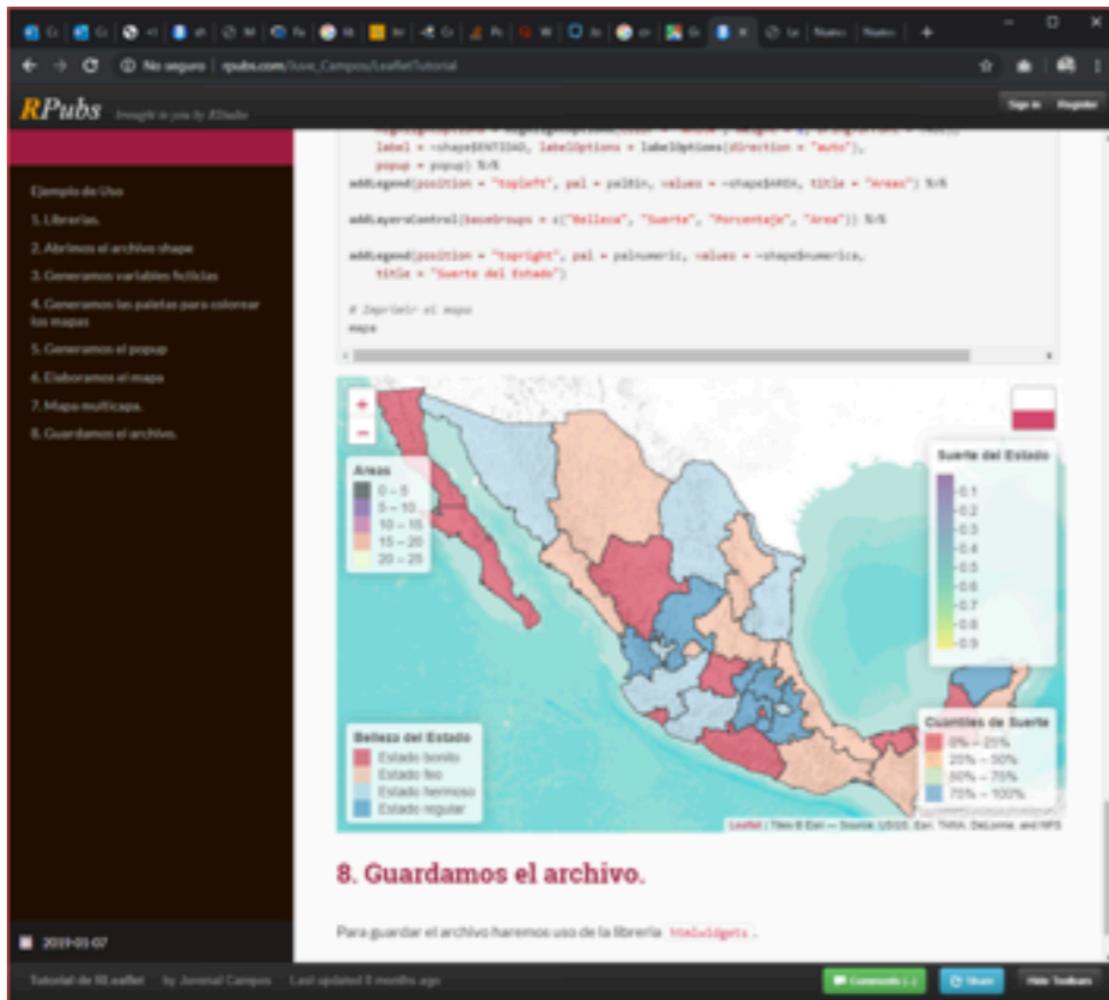
**Datos Crudos
(Raw Data)**



Datos Procesados que permiten llegar a conclusiones

Tareas que se hacen con datos

11. Páginas web y reportes (*.pdf, *.doc, diapositivas, tableros estáticos). library(markdown)



8. Guardamos el archivo.

Para guardar el archivo haremos uso de la librería [readrigner](#).

Unit 2 of the course
Juventud Campos - Based on DevelopML Classification Course
17/1/2018

Vector and raster coordinate systems

In order to perform any kind of analysis with more than one layer, you have to establish the same coordinate reference system (CRS), and the first step is determining what coordinate reference system your data has. To do this you can make use of the `sf` function `st_crs()` and the `raster` function `crs()`.

When the geodata has been read in with `sf` or `raster` has a CRS defined both sf and raster will recognize and retain it. When the CRS is not defined you will need to define it yourself using either the EPSG number or the projection.

```
#Read data
# Load the packages
library(sf)
# Linking to GDAL 3.6.1, Gdal 2.1.3, proj.4 4.9.3
library(raster)

# Loading required package: sp
root <- "/Users/juvicam/Desktop/CienciaCorta/Geovisualization/lat_35_raster_sf"
library(raster)
# Read in the tree shapefile
trees <- st_read(paste0(root, "/trees/tree.shp"))

# Reading layer 'trees' from data source '/Users/juvicam/Desktop/CienciaCorta/Geovisualization/lat_35_raster_sf'
# Simple feature collection with 65227 features and 7 fields
# geometry type: POLYGON
# dimension: XY
# bbox: -16.25000 ymin: 40.40934 ymax: -13.70000 year: 60.01000
# epsg (SRID): 4326
# projection: +proj=longlat +ellps=WGS84 +no_defs
# Read in the neighborhood shapefile
neighborhoods <- st_read(paste0(root, "/neighborhoods/neighborhoods.shp"))

# Reading layer 'neighborhoods' from data source '/Users/juvicam/Desktop/CienciaCorta/Geovisualization/lat_35_raster_sf'
# Simple feature collection with 128 features and 8 fields
# geometry type: MULTIPOLYGON
# dimension: XY
# bbox: -16.25000 ymin: 40.40912 ymax: -13.70000 year: 60.01000
# epsg (SRID): 4326
# projection: +proj=longlat +ellps=WGS84 +no_defs
# Read in the tree canopy raster
canopy <- raster(paste0(root, "/canopy.tif"))
```

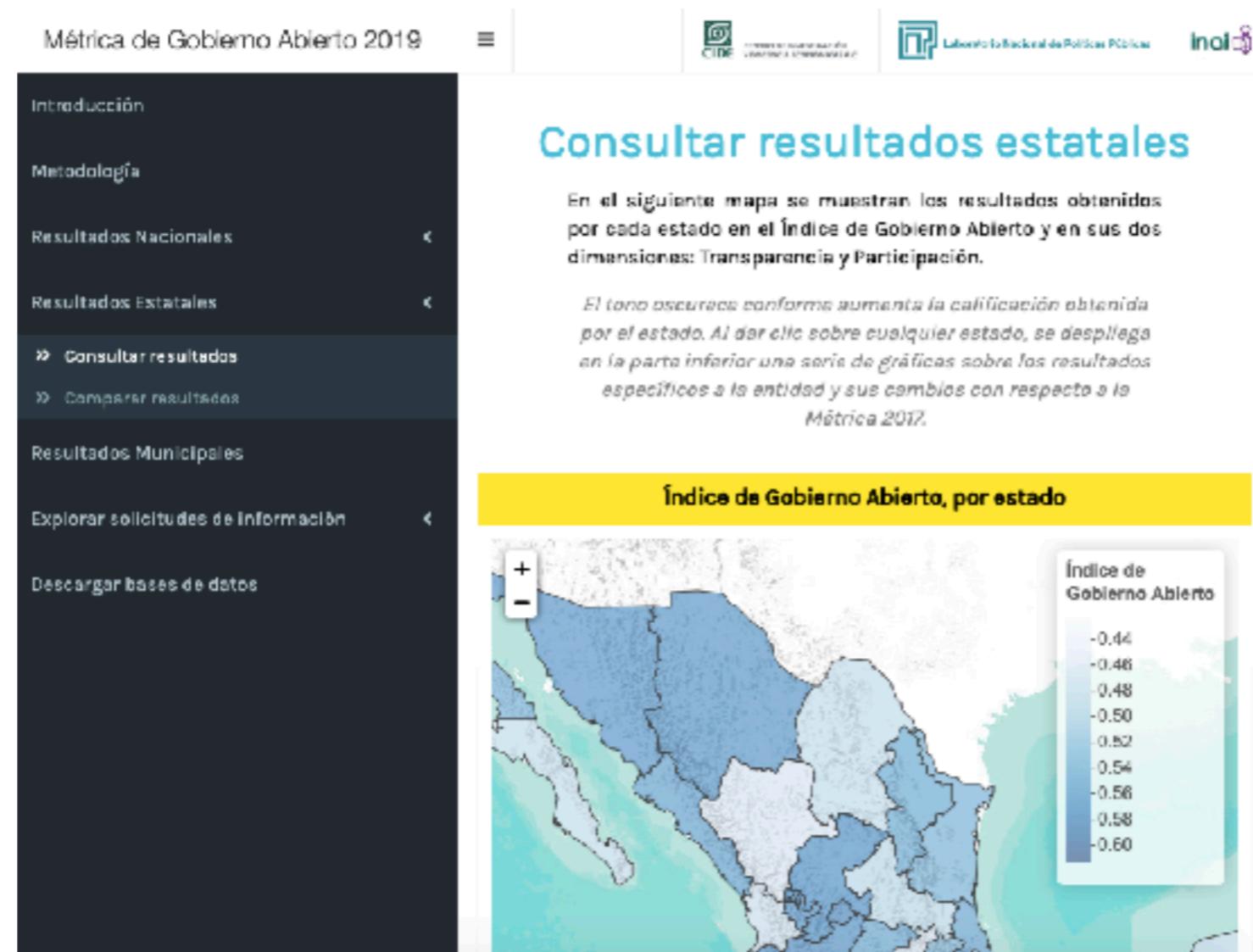
http://rpubs.com/Juve_Campos/LeafletTutorial

IATEX pdf

Tareas que se hacen con datos

12. Web Apps.

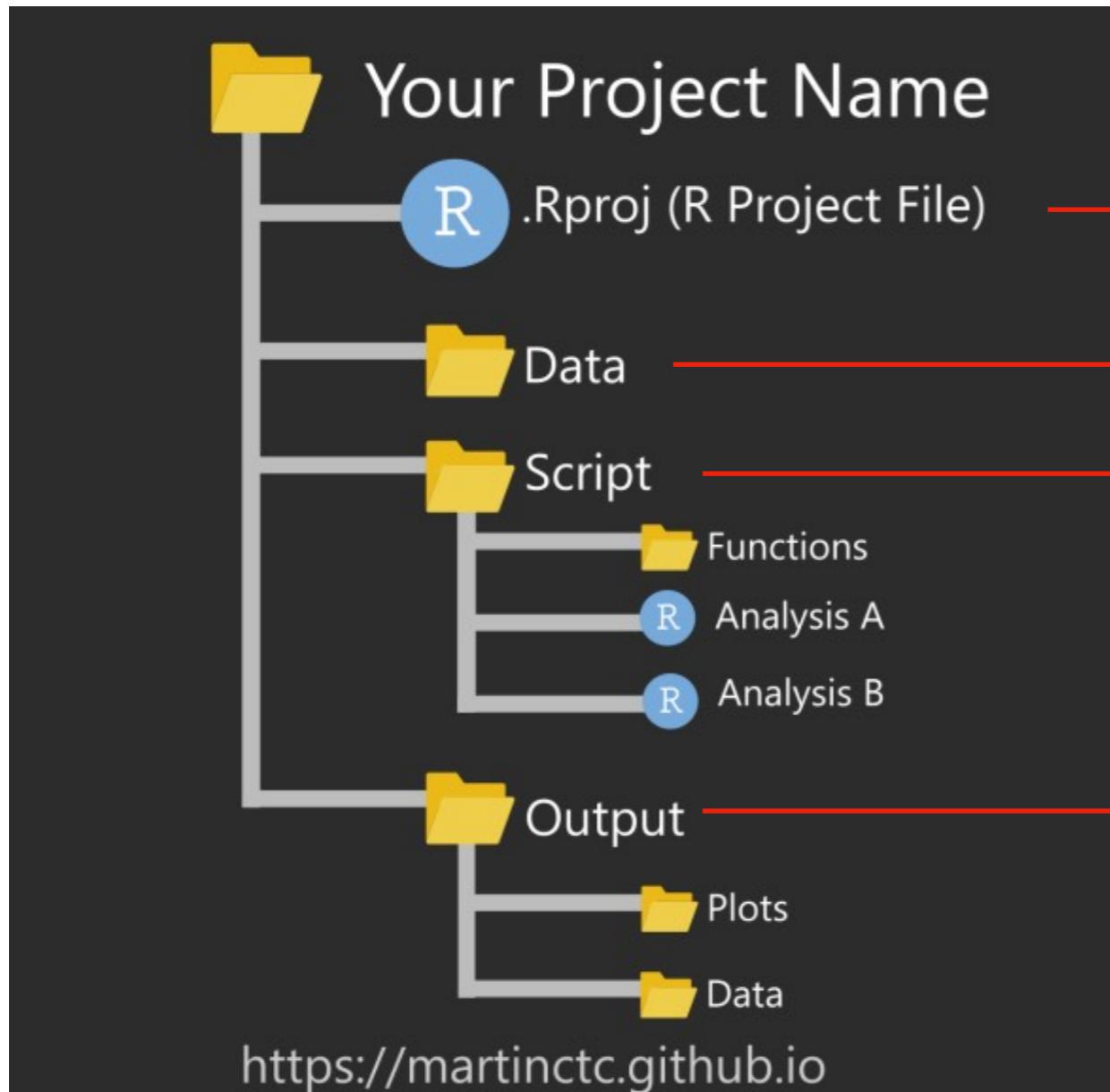
library(shiny)



Página web de resultados de la métrica de gobierno abierto, 2019.

Estructura de un código de R

Configuración de un proyecto básico de R



Archivo de proyecto (.Rproj)

Donde se guardan los archivos de datos

Donde se guardan los scripts

Donde se guardan los resultados del
proceso de trabajo (datos,
visualizaciones, reportes, modelos).

Estructura de un proyecto

- 1. Configuración**
- 2. Carga de librerías**
- 3. Carga de parámetros y funciones propias**
- 4. Carga de datos**
- 5. Limpieza de datos**
- 6. Análisis exploratorio de datos**
- 7. Generación de funciones**
- 8. Generación y pruebas de modelos**
- 9. Generación de gráficas**
- 10. Guardado de archivos**

```
# -----
# Autor: Juvenal Campos
# Fecha: 2025-08-18
# Proyecto: Ejemplo para el Tec de Monterrey
# Descripción: Script dummy que ilustra la estructura básica de un flujo en R.
# -----  
  
# 1) Configuración -----
options(digits = 3)
set.seed(42)  
  
# 2) Carga de librerías -----
library(dplyr)
library(readr)
library(ggplot2)  
  
# 3) Parámetros y funciones propias -----
params <- list(  
  input = "data/input.csv",      # <- reemplaza si tienes archivo  
  output_dir = "out",  
  target = "mpg"  
)  
paleta_colores <- c("#6950d8", "#3CEAFA", "#00b783",  
                   "#ff6260", "#ffaf84", "#ffbd41")  
if (!dir.exists(params$output_dir)) dir.create(params$output_dir, recursive = TRUE)  
  
clean_names <- function(x){  
  names(x) <- tolower(gsub("[^a-z0-9_]+", "_", names(x)))  
  x  
}  
rmse <- function(y, yhat) sqrt(mean((y - yhat)^2))  
  
# 4) Carga de datos -----
# df <- read_csv(params$input)          # <- real
# Dummy para ilustrar:
df <- clean_names(mtcars)
```

Estructura de un proyecto

1. Configuración
2. Carga de librerías
3. Carga de parámetros y funciones propias
4. Carga de datos
- 5. Limpieza de datos**
- 6. Análisis exploratorio de datos**
- 7. Generación de funciones**
- 8. Generación y pruebas de modelos**
- 9. Generación de gráficas**
- 10. Guardado de archivos**

```
# 5) Limpieza de datos -----
df <- df %>%
  mutate(across(everything(), ~v ~ v)) %>%
  filter(!is.na(.data[[params$target]]))

# 6) Análisis exploratorio (EDA) -----
print(summary(df))
print(df %>% summarise(n = n(), p = ncol(.)))

# 7) Generación de funciones -----
# (ya definimos `rmse()` arriba como ejemplo)

# 8) Modelado (dummy) -----
mod <- lm(mpg ~ wt + hp, data = df)
pred <- predict(mod, df)
cat("RMSE:", rmse(df$mpg, pred), "\n")

# 9) Gráficas -----
p <- ggplot(df, aes(wt, mpg)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE)

ggsave(file.path(params$output_dir, "scatter.png"), p,
       width = 5, height = 4, dpi = 150)

# 10) Guardado de archivos -----
write_csv(df, file.path(params$output_dir, "data_clean.csv"))
saveRDS(mod, file.path(params$output_dir, "model.rds"))
cat("Archivos guardados en:", normalizePath(params$output_dir), "\n")
```

Cargar archivos de datos

Archivo de proyecto (.Rproj)

Cargar archivos de datos

Introducción al tidyverse