

R y Spark para la Ciencia de Datos

Edgar Ruiz

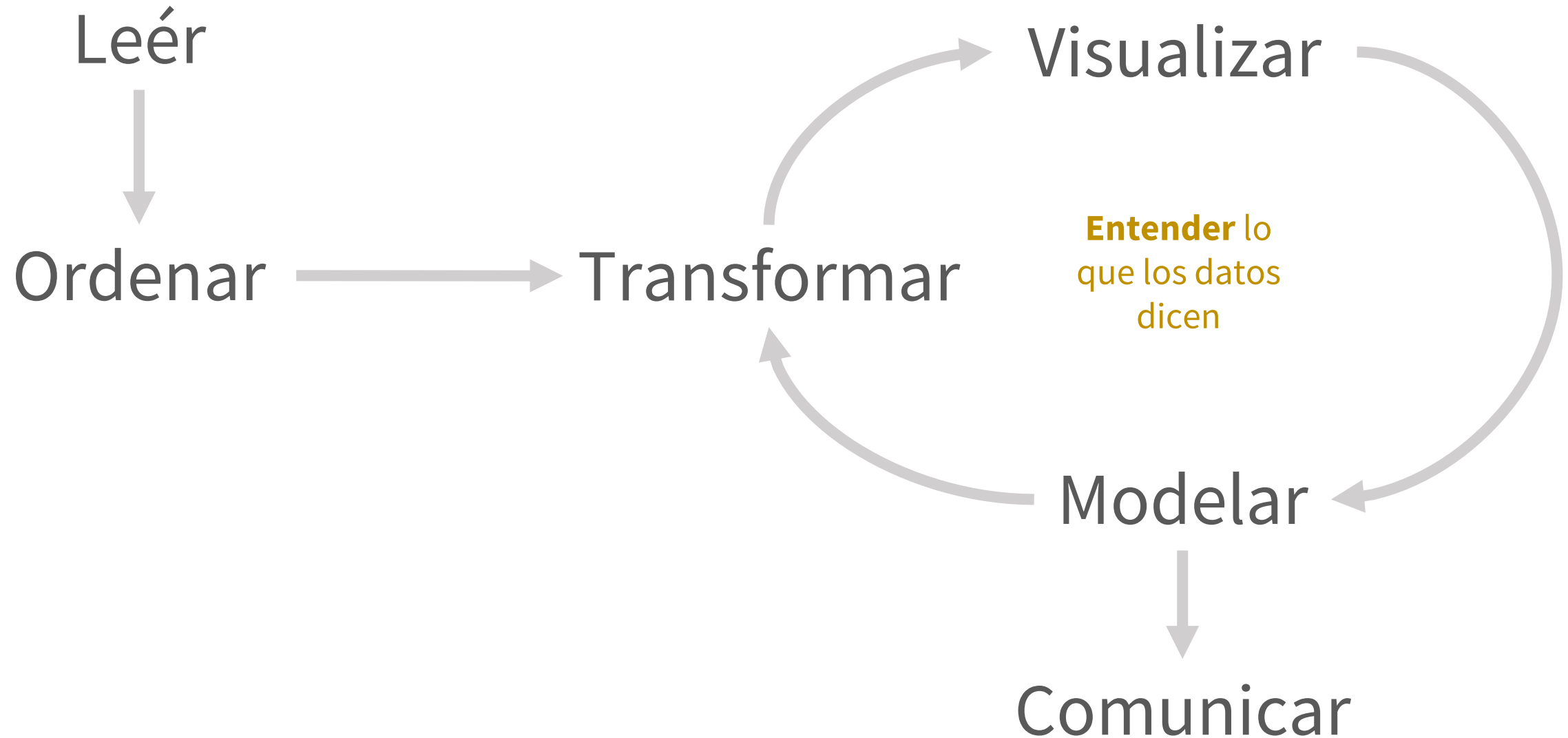
 @theotheredgar

 [linkedin.com/in/edgararuiz](https://www.linkedin.com/in/edgararuiz)

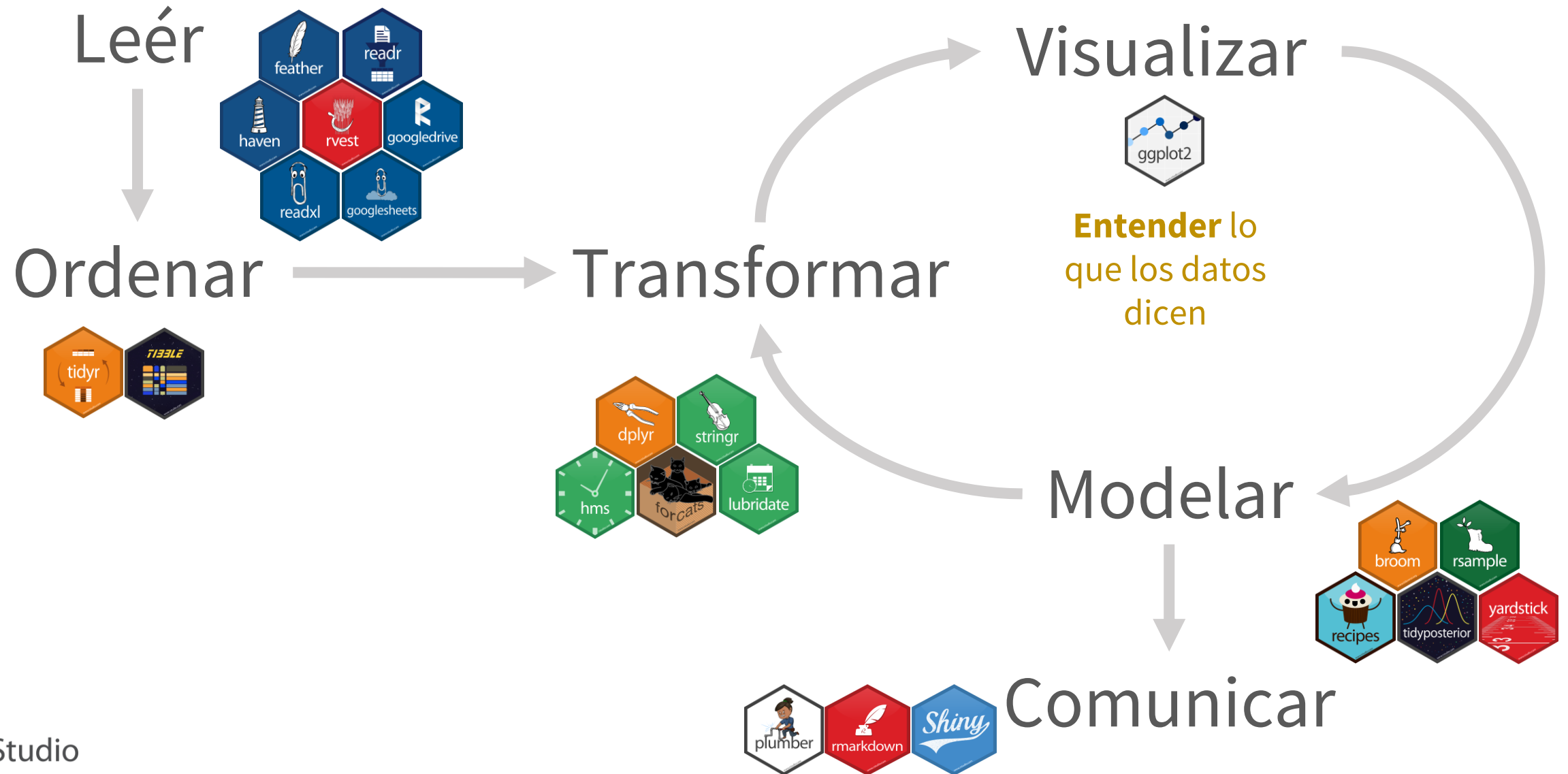
 github.com/edgararuiz

24 de marzo del 2019

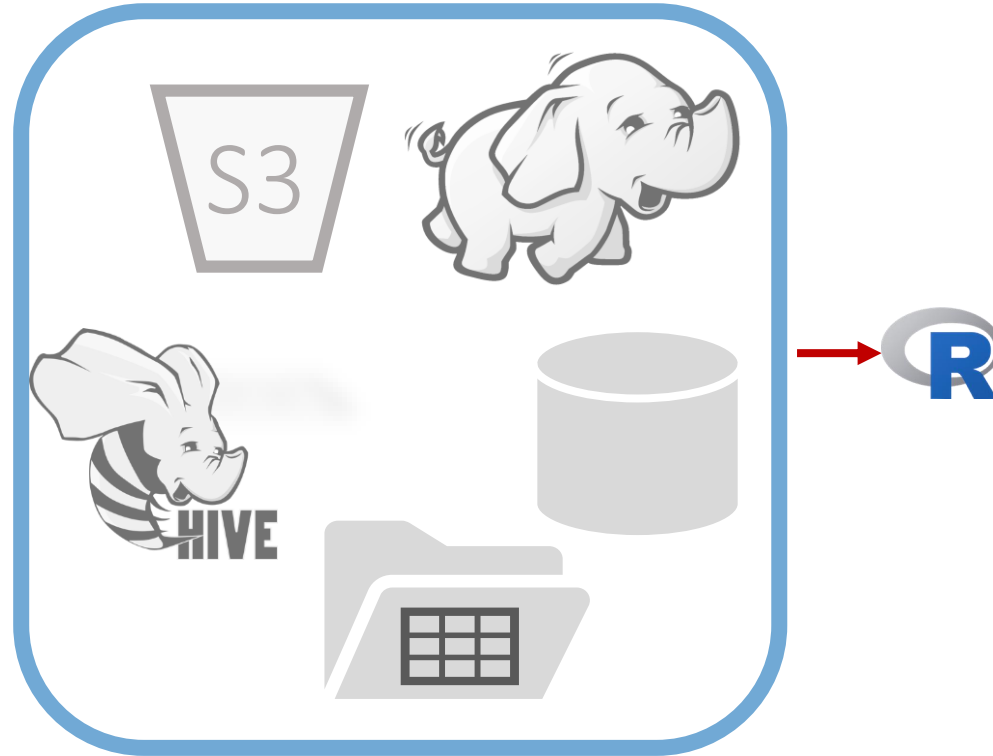
Ciencia de Datos



Todo se prepara dentro de



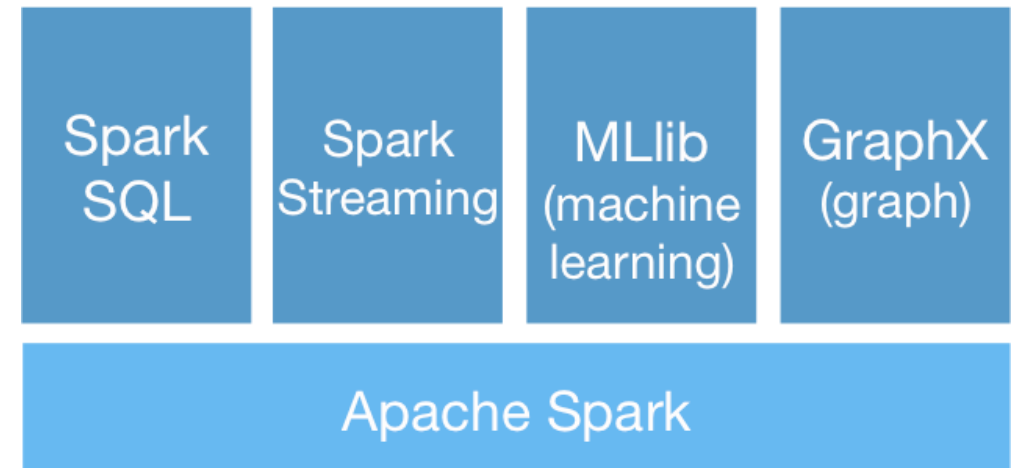
Datos mas grandes que el RAM



Que es ?

Motor analítico para procesar datos a gran escala

- Informática en clúster
- Aprendizaje automático
- Comunicación usando SQL
- API extensible



Tipos de análisis disponibles en Spark

- Modelos de regresión
- Modelos de clasificación
- Modelos de agrupación (clustering)
- Modelos de gráficas (GraphX)
- Análisis sobre datos stream (constante flujo)
- Análisis de texto, incluyendo modelos

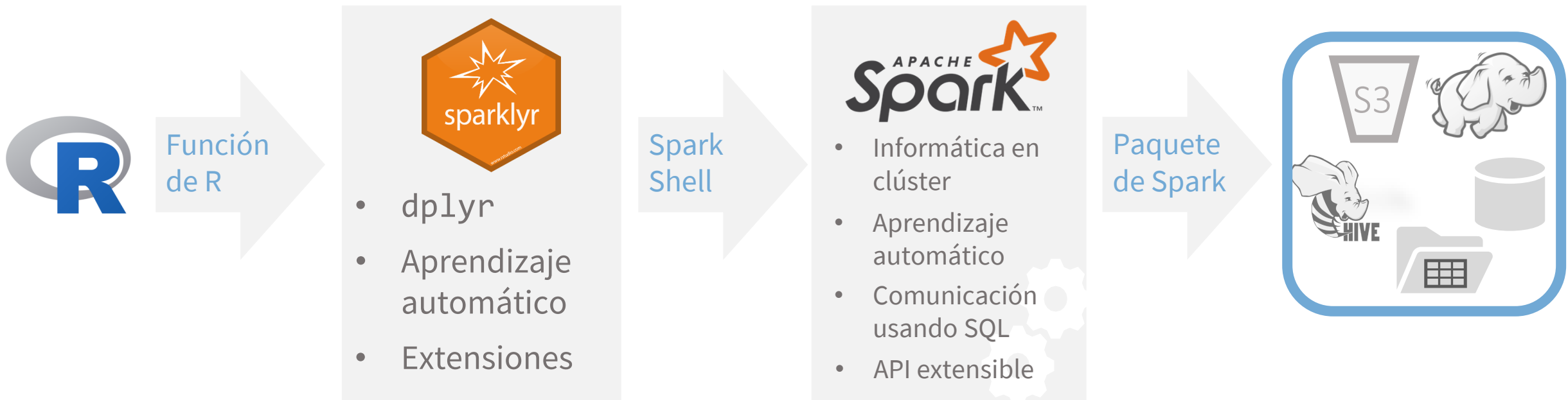
Que es ?

Es una interfaz para R y Spark

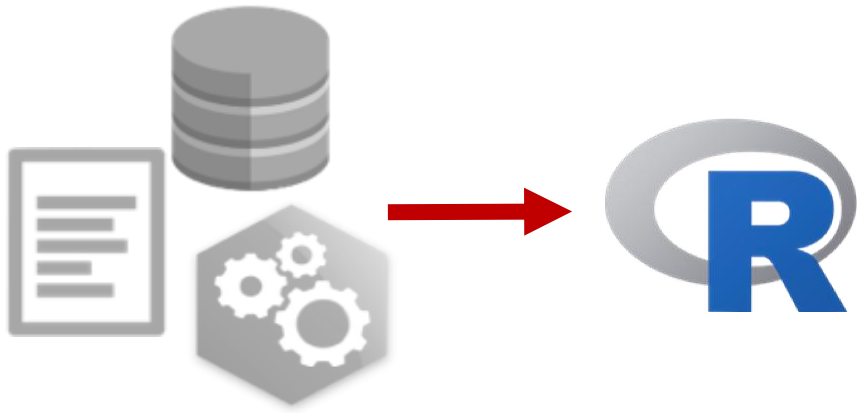
- Provee soporte a dplyr dentro de Spark
- Acceso a todo el API de Spark
- ...incluyendo Pipelines



Como funciona sparklyr



La idea principal

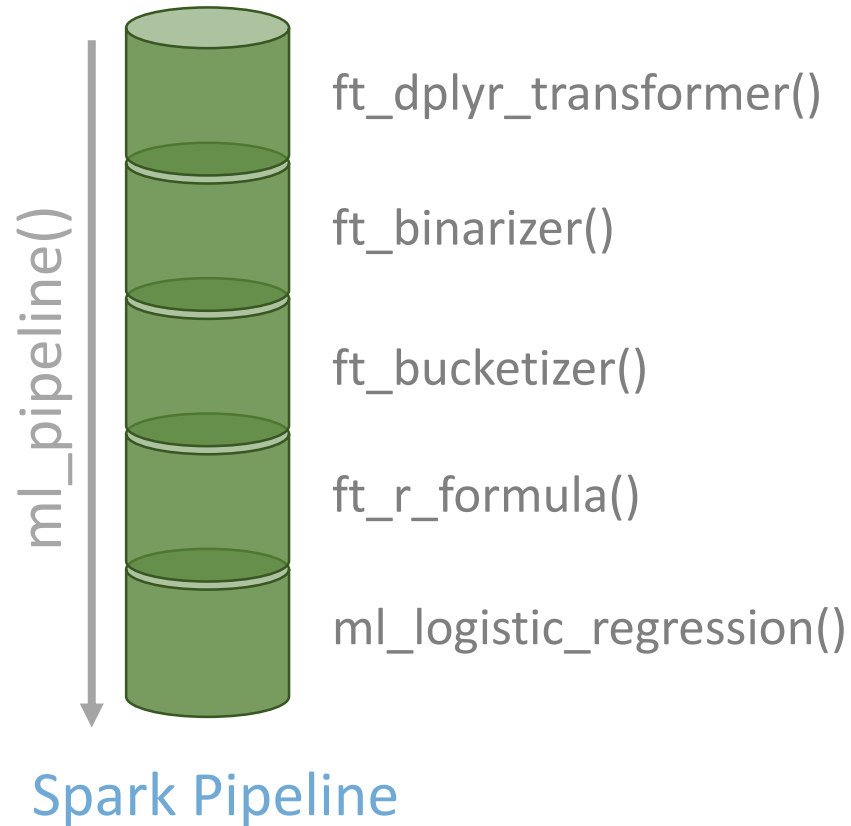


Extraer
datos

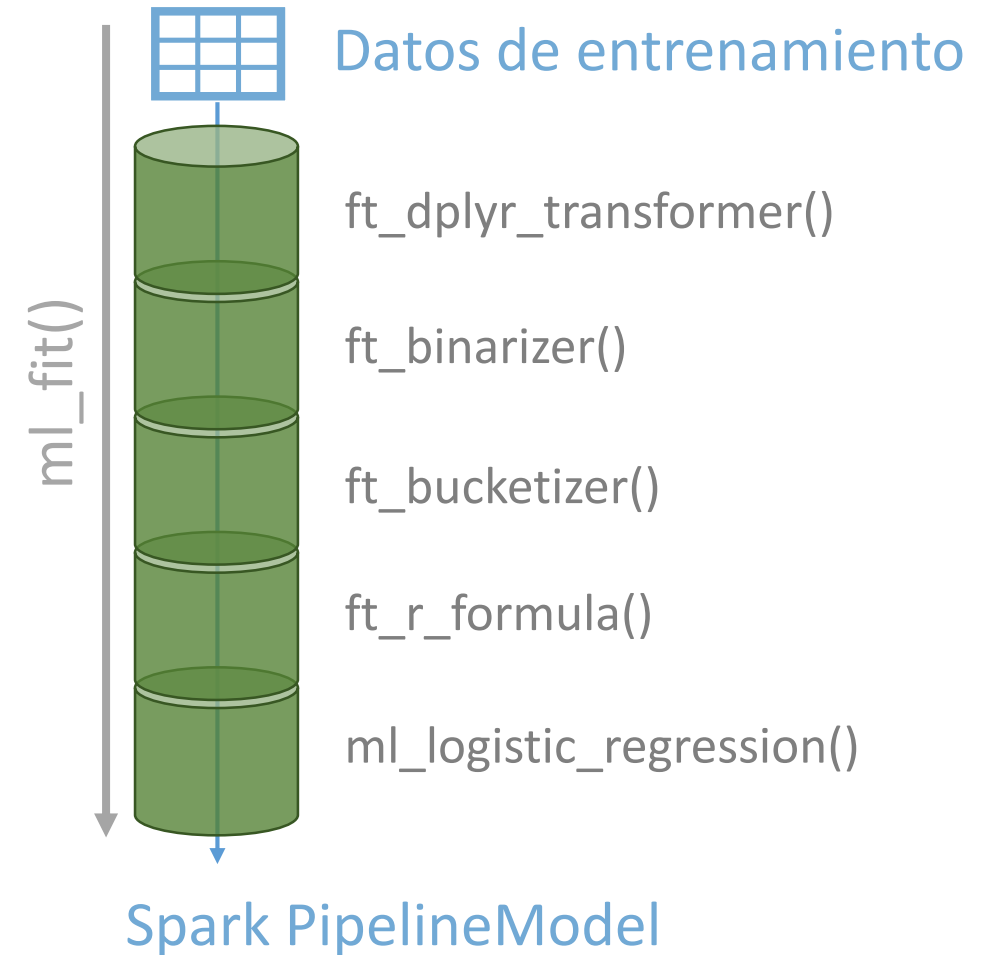


Spark Pipelines (Tubería)

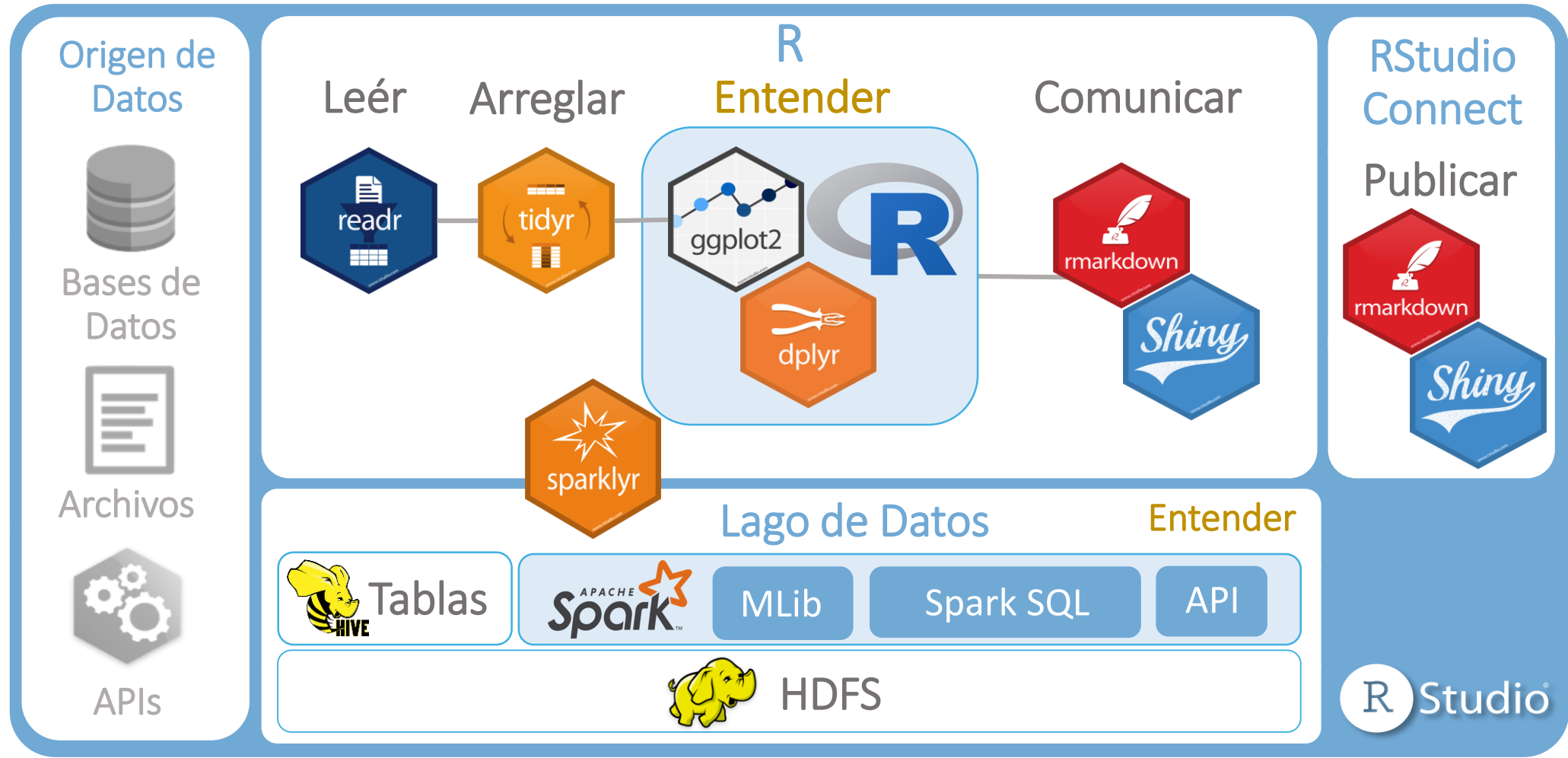
Estimator



Transformer



Ciencia de Datos con R y Spark



Recursos

Sitio oficial de sparklyr:

spark.rstudio.com

Sitio oficial de Spark:

spark.apache.org



Recursos en español

Para aprender como usar los paquetes en práctica, las Hojas de Referencia, o *Cheatsheets*, son los mejores recursos, no importa el idioma

Cortesía de Frans van Dunné, Carlos Ortega y Santiago Mota aquí:

rstudio.com/resources/cheatsheets

Importar Datos

with readr, tibble, and tidy

Guía Rápida

El tidyverse de Restá construido alrededor de los datos ordenados almacenados en tibbles.

Funciones Lectura

Lectura de datos tabulares a tibbles

Estas funciones comparten los siguientes argumentos comunes:

```
read_* (file, col_names = TRUE, col_types = NULL, locale = default_locale(), na = c("NA", "quoted_na" = TRUE, comment = "#"), trim_ws = TRUE, skip = 0, n_max = Inf, guess_max = min(1000, n_max), progress = interactive())
```

Lee ficheros delimitados por comas.

```
read_csv("file.csv")
```

Filtrando tipos de datos

Las funciones de readr interpretan los tipos de cada columna y convierten los tipos de forma apropiada (pero NO convertirán cadenas a factores automáticamente).

Un mensaje muestra el tipo de columna en el resultado.

```
## Parsed with column specification:
##   col1
##   age = col_integer(), age es un entero
##   sex = col_character(), sex es un carácter
##   earn = col_double(), earn es un doble (numérica)
```

1. Usa `problems()` para diagnosticar problemas
2. Usa `col_*` function para guiar el filtrado

```
x <- read_csv("file.csv"); problems(x)
```

```
col_guess()
col_character()
col_double()
col_euro_double()
col_datetime(format = "%Y-%m-%d") También
col_date(format = "%Y-%m-%d") and col_time(format = "%H:%M:%S")
col_factor(levels, ordered = FALSE)
col_integer()
col_logical()
col_number()
col_numeric()
col_skeptic()
x <- read_csv("file.csv", col_types = colf(
  A = col_double(),
  B = col_logical(),
  C = col_factor()
))
```

Visualización de Datos usando ggplot2

Guía Rápida

Studio

Conceptos Básicos

ggplot2 se basa en la idea que cualquier gráfica se puede construir usando estos tres componentes: **datos**, **coordenadas** y **objetos geométricos (geoms)**. Este concepto se llama: **gramática de las gráficas**.

Para visualizar resultados, asigne variables a las propiedades visuales, o **estéticas**, como **tamaño**, **color**, y **posición x o y**.

Para construir una gráfica complete este patrón

```
ggplot(data = <DATOS>) +
  <FUNCIÓN GEOM>()
  <FUNCIÓN ESTÉTICA>()
  <FUNCIÓN COORDENADA>()
  <FUNCIÓN TEMA>()
```

Requerido, se proveen valores iniciales

ggplot(data = mpg, aes(x = cty, y = hwy))

Este comando comienza una gráfica, completa mediante agregando capas, un **geom** por capa.

ggplot(x = cty, y = hwy, data = mpg, geom = "point")

Este comando es una gráfica completa, tiene datos, las estéticas están asignadas y por lo menos un geom.

last_plot()

Devuelve la última gráfica

ggsave("plot.png", width = 5, height = 5)

La última gráfica es grabada como una imagen de 5 por 5 pulgs., usa el mismo tipo de archivo que la extensión

Geoms

Funciones geom se utilizan para visualizar resultados. Asigne variables a las propiedades estéticas del geom. Cada geom forma una capa.

Geométricas Elementales

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

- `geom_blank()` (Bueno para expandir límites)
- `geom_curve()` `aes(yend = lat + 1, vend = long + 1, curvature = 1)`
- `geom_path()` `lineend = "butt", linejoin = "round", linetype = 1`
- `geom_polygon()` `aes(group = group)`
- `geom_rect()` `aes(xmin = long, ymin = lat, xmax = long + 1, ymax = lat + 1)`
- `geom_ribbon()` `aes(ymin = unemploy - 500, ymax = unemploy + 500)`

Segmentos Lineales

```
b <- geom_abline(aes(intercept = 0, slope = 1))
b <- geom_hline(aes(yintercept = lat))
b <- geom_vline(aes(xintercept = long))
b <- geom_segment(aes(yend = lat + 1, vend = long + 1))
b <- geom_spoke(aes(angle = 1:155, radius = 1))
```

Una Variable

Continua

```
c <- ggplot(mpg, aes(hwy))
d <- ggplot(mpg)
```

- `geom_area(stat = "bin")`
- `geom_density(kernel = "gaussian")`
- `geom_dotplot()`
- `geom_freqpoly()`
- `geom_histogram(binwidth = 5)`
- `geom_qq(aes(sample = hwy))`
- `geom_bar()`

Discreta

```
d <- ggplot(mpg, aes(hwy))
d <- geom_bar()
```

Dos Variables

Distribución Bivariada Continua

```
h <- geom_bin2d(binwidth = c(0.25, 500))
h <- geom_density2d()
h <- geom_hex()
```

Función Continua

```
i <- ggplot(economics, aes(date, unemploy))
i <- geom_area()
i <- geom_line()
i <- geom_step(direction = "hv")
```

X Continua, Y Continua

```
j <- geom_label(aes(label = cty, nudges_x = 1, nudges_y = 1, check_overlap = TRUE))
j <- geom_jitter(height = 2, width = 2)
j <- geom_point()
j <- geom_quantile()
j <- geom_rug(sides = "bl")
j <- geom_smooth(method = lm)
j <- geom_text()
j <- geom_xdensity()
j <- geom_ydensity()
```

X Discreta, Y Continua

```
f <- ggplot(mpg)
f <- geom_col()
f <- geom_boxplot()
f <- geom_dotplot(stackin = "true")
f <- geom_violin()
f <- geom_violinwidth()
```

X Discreta, Y Discreta

```
g <- ggplot(diamonds)
g <- geom_count()
g <- geom_tile()
g <- geom_contour()
g <- geom_contour_filled()
```

Domar Datos con dplyr and tidy

Hoja de Referencia

Studio

Sintaxis

Convenciones útiles para la doma

`dplyr::tbl_df(iris)`

Convierte datos a una `tbl`. Objetos `tbl` son más fáciles de inspeccionar que data frames. Ro solo muestra los datos que caben en la pantalla:

```
Source: local data frame [150 x 5]
  Sepal.Length Sepal.Width Petal.Length
1 5.1 3.5 1.4
2 4.9 3.0 1.4
3 4.7 3.2 1.3
4 4.6 3.1 1.3
5 5.0 3.6 1.4
Variables not shown: Petal.Width (dbl), Species (fctr)
```

`dplyr::glimpse(iris)`

Resumen con mucha información sobre los datos `tbl`.

`utils::View(iris)`

Observa el conjunto de datos en lo que parece una hoja de cálculo (nota la V mayúscula).

`dplyr::%>%`

Pasa el objeto a la izquierda al primer argumento (o argumento...) de la función a la derecha creando un tubo.

```
x %>% f(y) es lo mismo que f(x, y)
y %>% f(x, z) es lo mismo que f(x, y, z)
```

El uso de `%>%` y tuberías (en Inglés: "Piping") con `%>%` resulta en código más legible. Por ejemplo:

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Length)) %>%
  arrange(desc(avg))
```

Datos Ordenados

- La base para domar datos en R

En un conjunto de datos ordenado:

Cada **variable** tiene su propia **columna**

Cada **observación** tiene su propia **fila**

$M \times A$

Datos ordenados complementan las operaciones vectorizadas de R. Automáticamente R preserva observaciones mientras manipulas las variables. No hay otro formato que funcione tan intuitivamente en R.

Remodelar Datos

Cambia el esquema de los datos

`tidyr::gather(cases, "year", "n", 2:4)`

Reune columnas en filas.

`tidyr::spread(pollution, size, amount)`

Extiende filas en columnas.

`tidyr::separate(storms, date, c("a", "m", "d"))`

Separa una columna en varias.

`tidyr::unite(data, col, ..., sep)`

Une varias columnas en una.

Subconjuntos de Observaciones

`dplyr::filter(iris, Sepal.Length > 7)`

Extrae filas que cumplen criterios lógicos.

`dplyr::distinct(iris)`

Remueve filas duplicadas.

`dplyr::sample_frac(iris, 0.5, replace = TRUE)`

Selecciona una fracción de filas al azar.

`dplyr::sample_n(iris, 10, replace = TRUE)`

Selecciona n filas al azar.

`dplyr::slice(iris, 10:15)`

Selecciona filas por posición.

`dplyr::top_n(storms, 2, date)`

Selecciona y ordena las n entradas mas altas (por grupo si los datos estan agrupados).

Subconjuntos de Variables

`dplyr::select(iris, Sepal.Length, Petal.Length, Species)`

Selecciona columnas por nombre o funciones de ayuda.

Funciones de ayuda para for select - ?select

`select(iris, contains("n"))`

Selecciona columnas cuyos nombres contienen una cadena de caracteres.

`select(iris, ends_with("Length"))`

Selecciona columnas cuyos nombres terminan con una cadena de caracteres.

`select(iris, everything())`

Selecciona todas las columnas.

`select(iris, matches("a"))`

Selecciona columnas cuyo nombre cumple con una expresión regular.

`select(iris, num_range("1:3"))`

Selecciona columnas con nombres x1, x2, x3, etc.

`select(iris, one_of("Species", "Genus"))`

Selecciona columnas cuyos nombres estan en un grupo de nombres.

`select(iris, starts_with("Sepal"))`

Selecciona columnas cuyos nombres comienzan con una cadena de caracteres.

`select(iris, Sepal.Length[Sepal.Width])`

Selecciona todas las columnas entre Sepal.Length y Petal.Width (incluyendo).

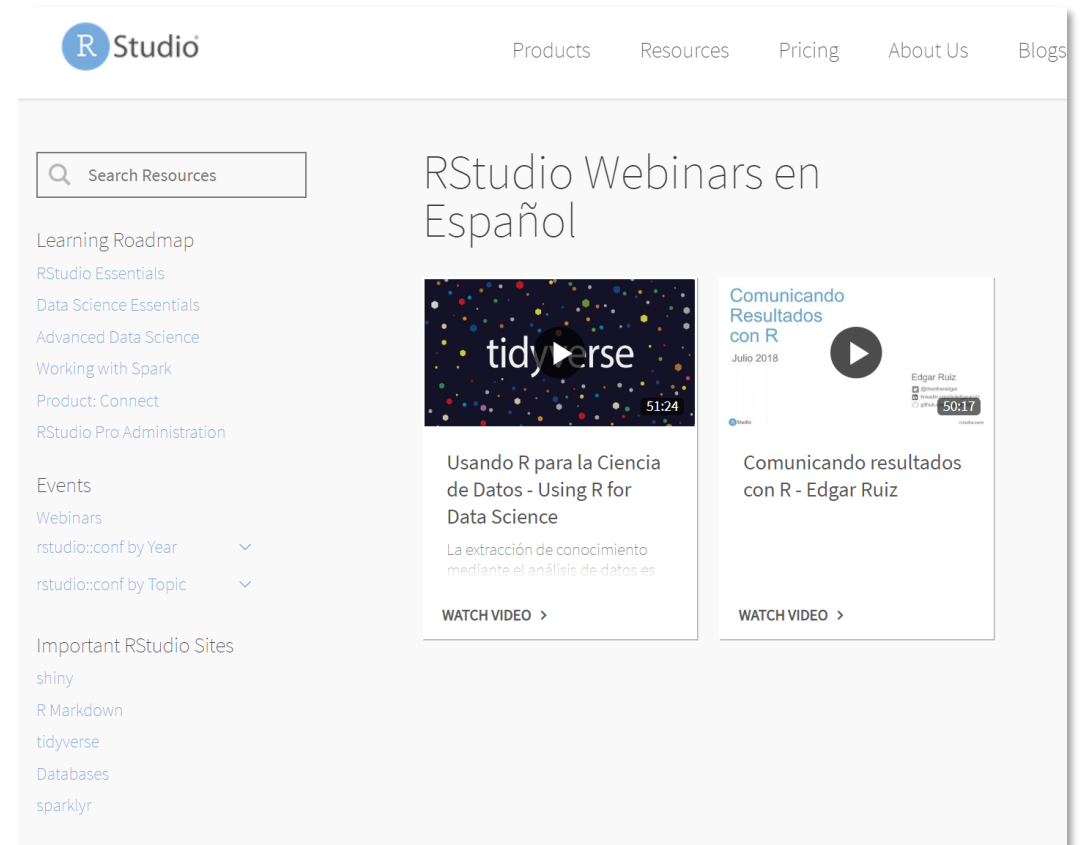
`select(iris, -Species)`

Selecciona todas las columnas excepto Species.

Recursos en español - Webinars

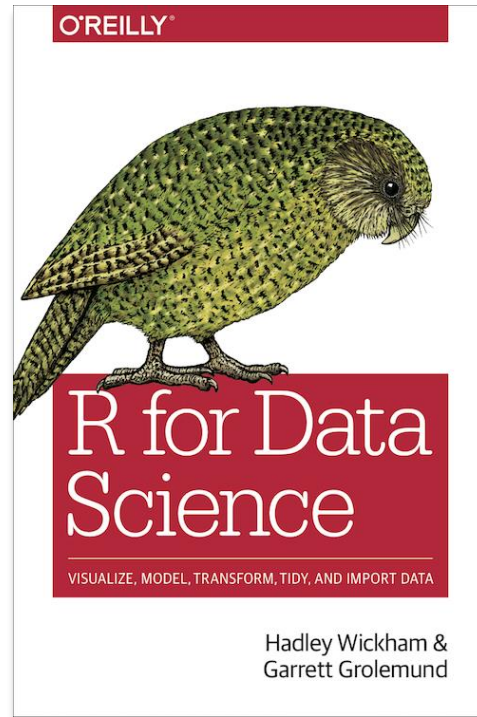
Una base para aprender a utilizar R de manera efectiva en nuestros análisis. Estos webinars proveen tres cosas:

1. Ejemplos de código
2. Presentaciones
3. Video de la sesión



resources.rstudio.com/espanol

Muy pronto!



github.com/cienciadedatos

Materiales

rstudio.io/conectar