MBA
USP
ESALQ

ENGENHARIA DE DADOS III

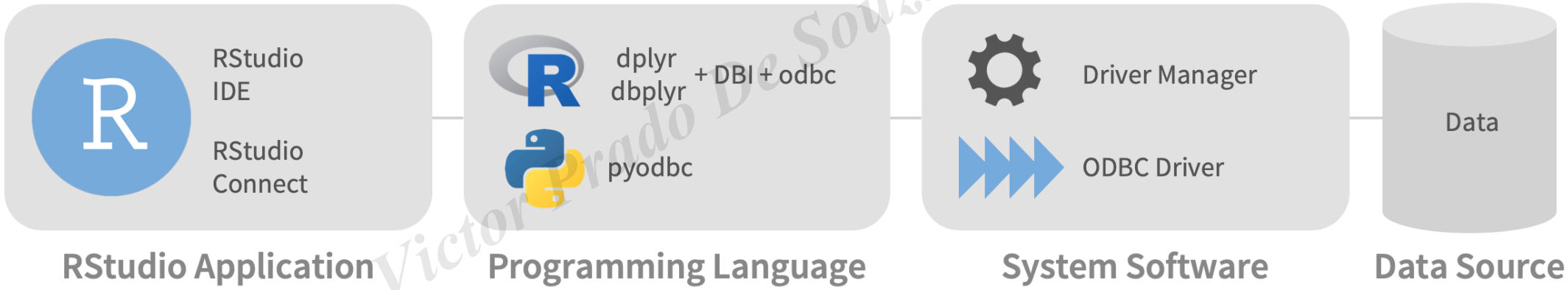
Prof. Dr. Jeronymo Marcondes

*A responsabilidade pela idoneidade, originalidade e licitude dos conteúdos didáticos apresentados é do professor.

ODBC e JDBC

- Foco da aula: ODBC
- Objetivo: utilizar as capacidades do banco de dados para realizar nossas operações.
- Limitações das linguagens ao trabalhar com arquivos texto com os dados necessários.

Exemplo no R Studio



ODBC e JDBC

- Abordagem da aula:

1. Executar as querys e trazer para o R.
2. Utilizar o dplyr para realizar as operações dentro do database.

- Tudo isso depende do entendimento dos pacotes DBI e DPLYR.

Pacote DBI

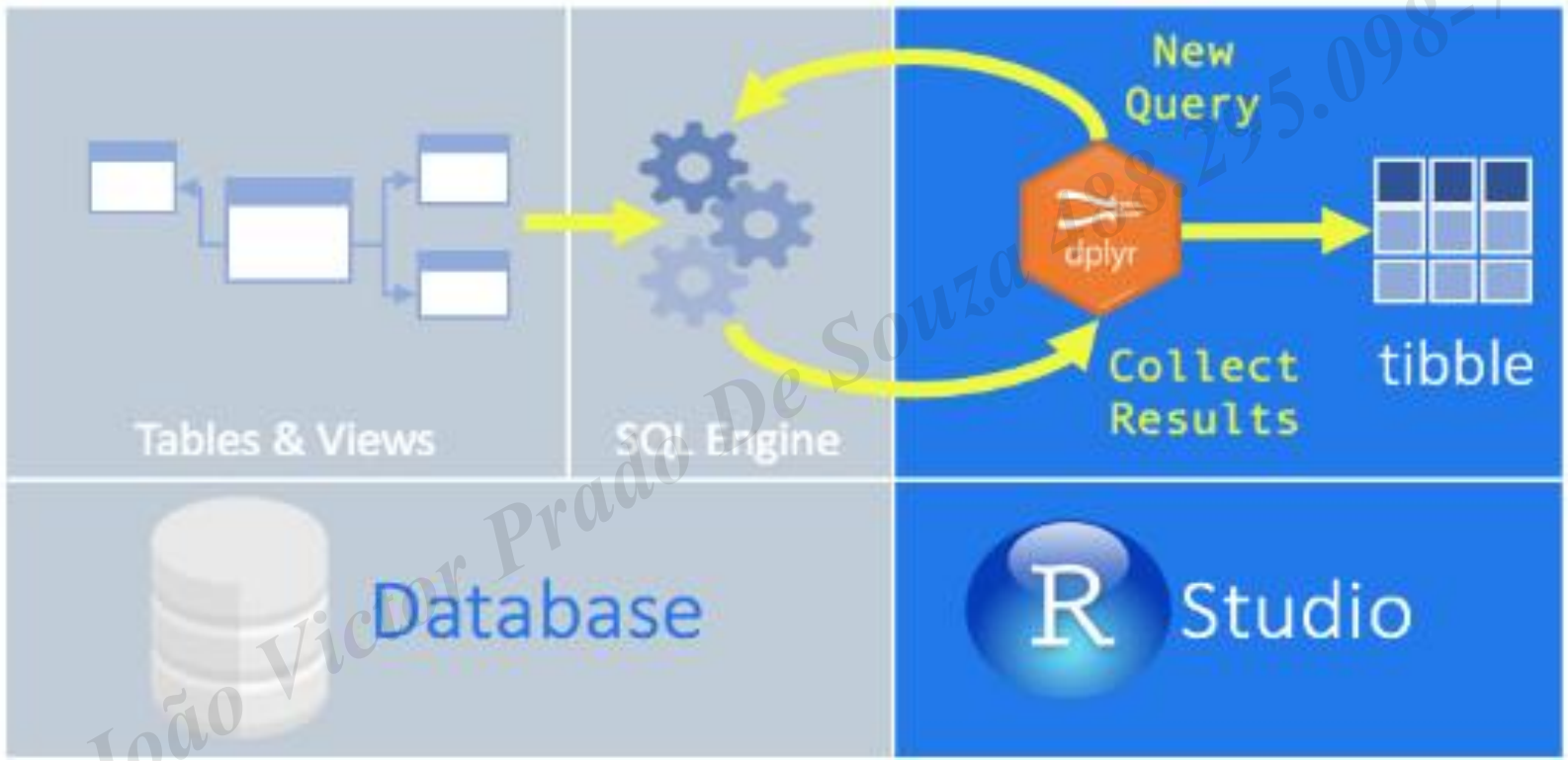
https://db.rstudio.com/r-packages/dbi/

DBI

- Permite a divisão da conexão em um front-end e um back-end
- Executa query no banco
- Permite fazer interação entre outros pacotes e o banco (dplyr)

Bases comerciais e open source

DBI



Fonte: https://db.rstudio.com/getting-started/

DBI - estrutura

Install.packages("DBI")

library("DBI")

```
con <- DBI::dbConnect(odbc::odbc(),
                      Driver   = "[your driver's name]",
                      Server   = "[your server's path]",
                      UID      = rstudioapi::askForPassword("Database user"),
                      PWD      = rstudioapi::askForPassword("Database password"),
                      Port     = 3306)
```

DBI - execução de consultas

```
dbListTables(con)
dbWriteTable(con, "mtcars", mtcars)
dbListTables(con)

dbListFields(con, "mtcars")
dbReadTable(con, "mtcars")
```

DBI - execução de consultas

```
res <- dbSendQuery(con, "SELECT * FROM mtcars WHERE cyl = 4")
dbFetch(res)
dbClearResult(res)
```

DBI - boas práticas

- Fechar conexão após uso
- Atentar para drivers em ambientes corporativos: https://db.rstudio.com/best-practices/drivers/
- Saber o que está ocorrendo quando criamos um dataframe local com consulta.

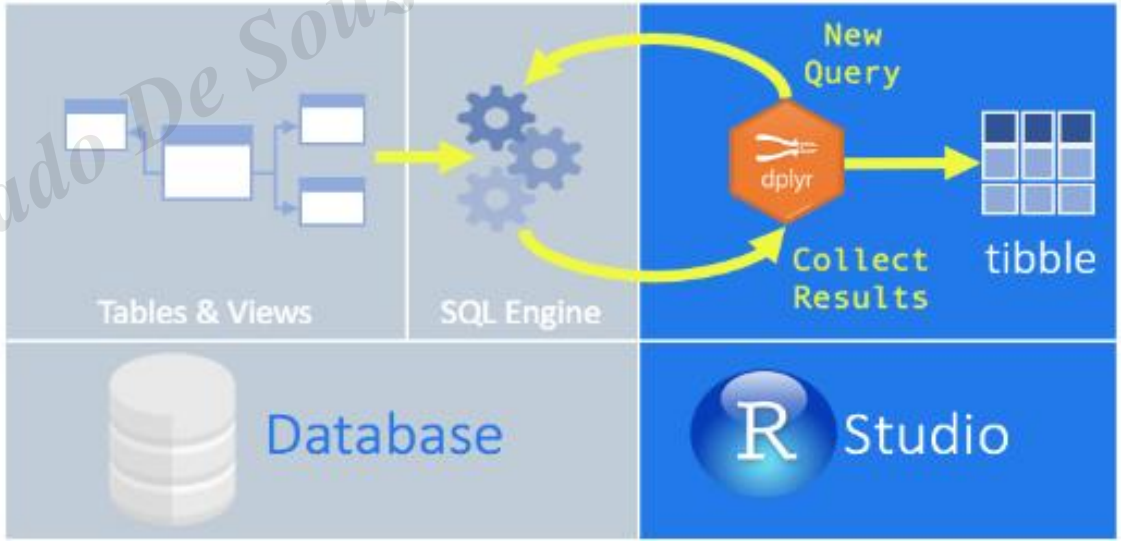
DPLYR

- Principal pacote para Data Wrangling.
- Rápido e de fácil manipulação – resolve mais de 90% dos problemas de estrutura que podem surgir.
- Trabalha com dataframes que seguem padrões de tidydata (tibble).

DPLYR

- Dplyr vai permitir a execução de algumas operações de wrangling diretamente no banco e utilizando o poder computacional do banco sem sobrecarregar o R.

```
install.packages("dplyr")
library("dplyr")
```



TIbble

- Hadley Wickham:

1. Cada variável é uma coluna
2. Cada observação é uma linha
3. Cada unidade observacional é a tabela

DPLYR

- As principais funções do dplyr são:

1. Select()
2. Arrange()
3. Filter()
4. Mutate()
5. Group_by()
6. Summarise()

DPLYR - estrutura básica

- Uso básico do select – seleciona colunas de uma tabela

select(dataframe, campos_separados_por_virgula)

DPLYR - uso do pipe

- Uso do pipe – mais de uma operação do dplyr em uma única tabela
- Ideia de sequencial

dataframe %>% select(campos_separados_por_virgula)

DPLYR - exemplo de select

```
dim(starwars)
#> [1] 87 14
starwars
#> # A tibble: 87 × 14
#>   name       height  mass hair_color skin_color eye_color birth_year sex
#>   <chr>       <int> <dbl> <chr>      <chr>      <chr>           <dbl> <chr>
#> 1 Luke Skyw…    172    77 blond      fair       blue             19   male
#> 2 C-3PO         167    75 NA         gold       yellow          112   none
#> 3 R2-D2          96    32 NA         white, bl… red              33   none
#> 4 Darth Vad…    202   136 none       white      yellow           41.9 male
#> # … with 83 more rows, and 6 more variables: gender <chr>,
#> #   homeworld <chr>, species <chr>, films <list>, vehicles <list>,
#> #   starships <list>
```


https://dplyr.tidyverse.org/articles/dplyr.html

DPLYR - exemplo de select

```
starwars %>% select(hair_color, skin_color, eye_color)
#> # A tibble: 87 × 3
#>   hair_color skin_color  eye_color
#>   <chr>      <chr>       <chr>
#> 1 blond      fair        blue
#> 2 NA         gold        yellow
#> 3 NA         white, blue red
#> 4 none       white       yellow
#> # … with 83 more rows
```

https://dplyr.tidyverse.org/articles/dplyr.html

DPLYR - exemplo de filter

dataframe %>% filter(campos == “determinado valor”)

dataframe %>% filter(campos > “determinado valor”)

dataframe %>% filter(campos < “determinado valor”)

dataframe %>% filter(campos != “determinado valor”)

DPLYR - exemplo de filter

```
filter(starwars, species == "Human")
#> # A tibble: 35 × 14
#>    name      height  mass hair_color skin_color eye_color birth_year sex
#>    <chr>      <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr>
#>  1 Luke Sky…    172    77 blond      fair       blue            19   male
#>  2 Darth Va…    202   136 none       white      yellow          41.9 male
#>  3 Leia Org…    150    49 brown      light      brown           19   fema…
#>  4 Owen Lars    178   120 brown, gr… light      blue            52   male
#>  5 Beru Whi…    165    75 brown      light      blue            47   fema…
#>  6 Biggs Da…    183    84 black      light      brown           24   male
#>  7 Obi-Wan …    182    77 auburn, w… fair       blue-gray       57   male
#>  8 Anakin S…    188    84 blond      fair       blue            41.9 male
#>  9 Wilhuff …    180    NA auburn, g… fair       blue            64   male
#> 10 Han Solo     180    80 brown      fair       brown           29   male
#> # … with 25 more rows, and 6 more variables: gender <chr>,
#> #   homeworld <chr>, species <chr>, films <list>, vehicles <list>,
#> #   starships <list>
```


https://dplyr.tidyverse.org/articles/dplyr.html

DPLYR - exemplo de group_by e summarise

dataframe %>% summarise(nova_variável = função aplicada em variável)

dataframe %>% group_by(variavel) %>% summarise(nova_variável = função aplicada em variável)

DPLYR - exemplo de group_by e summarise

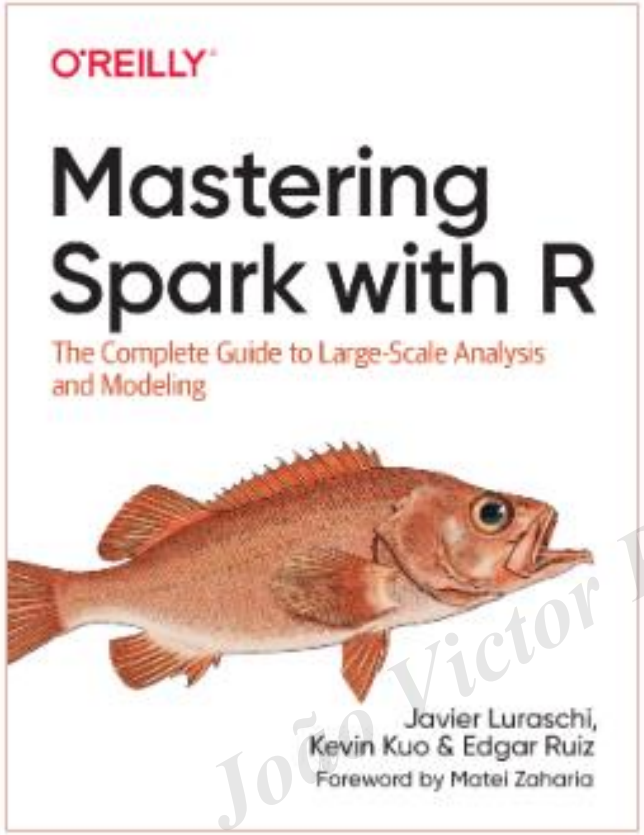
```
starwars %>% summarise(height = mean(height, na.rm = TRUE))
#> # A tibble: 1 × 1
#>   height
#>    <dbl>
#> 1   174.
```

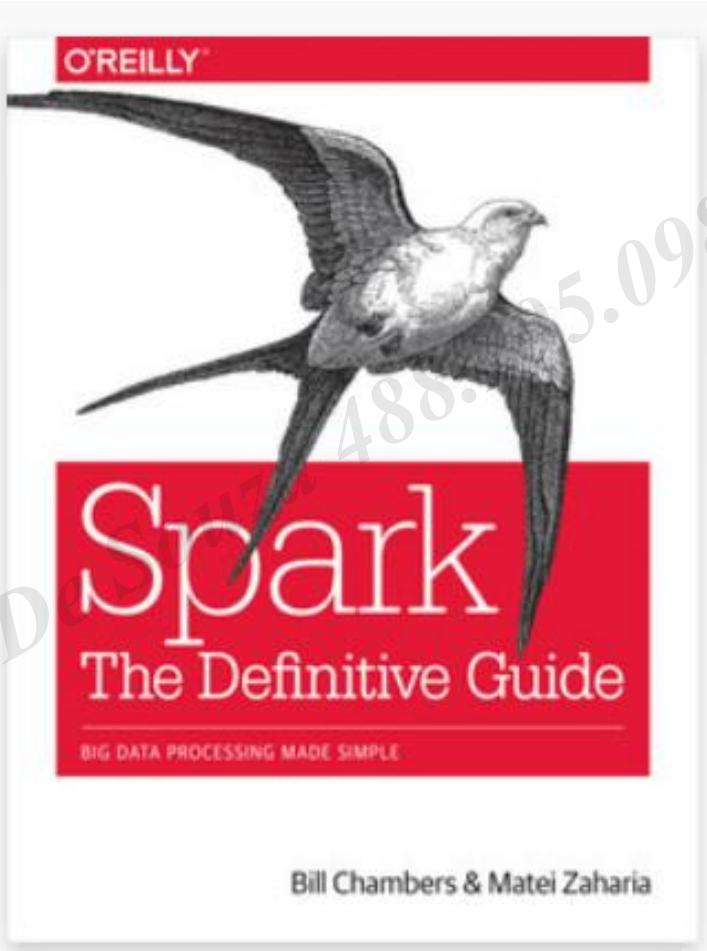
https://dplyr.tidyverse.org/articles/dplyr.html

DPLYR - outros casos

- Amplitude de uso do dplyr
- Foco da aula

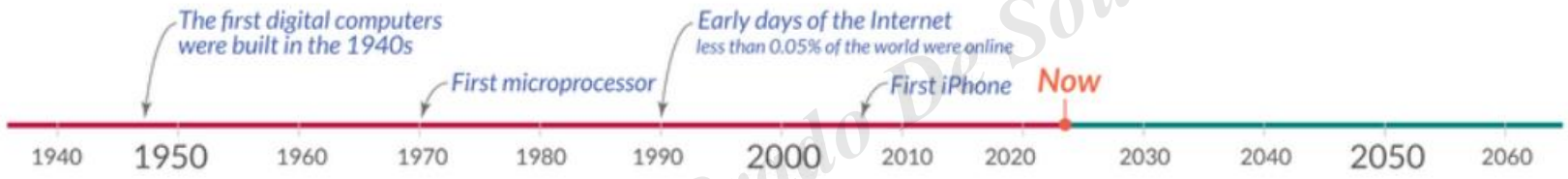
Apache Spark





Big Data

- Evolução constante dos computadores



Fonte: https://ourworldindata.org/brief-history-of-ai

Big Data

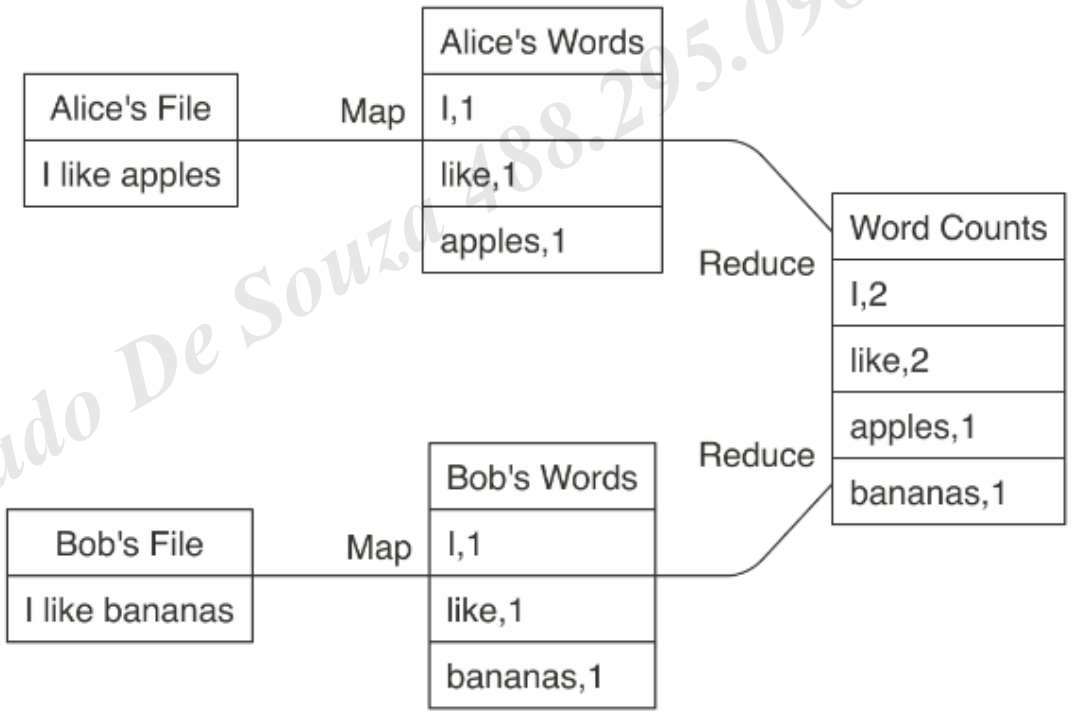
- Computadores cada vez mais potentes = + dados.
- 2005 – velocidade dos dados é grande demais.
- Inicia o processo de computação paralelizada.

Big Data

- Custo de armazenamento também não reduz.
- Só paralelizar CPU não é mais suficiente.
- Entra o Google File System

Hadoop

- *Map Reduce*
- *HDFS*



Apache Spark

- UC Berkeley
- “Spark: Cluster Computing with Working Sets”.
- Vantagens e Desvantagens do Map Reduce.

Apache Spark

- Entra Spark.
- Possibilidade de processamento feito “in memory”.
- Velocidade de execução.

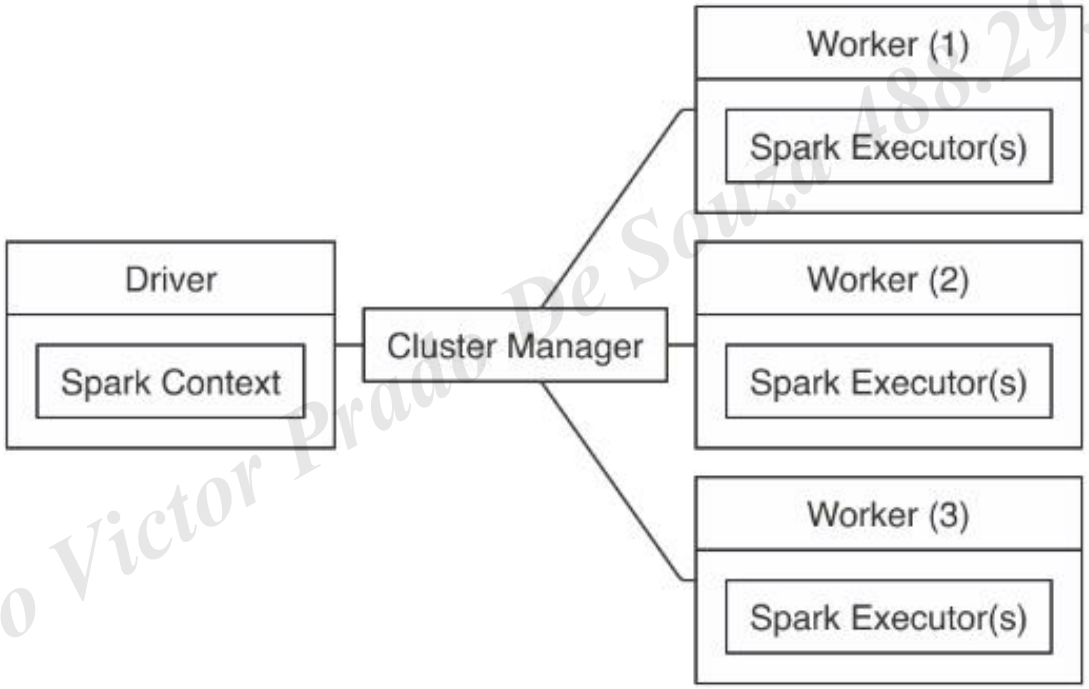
Apache Spark

- É uma *Engine* - ou seja, serve para melhorar os mais diversos processos. Se adapta a diversas linguagens.

- Estrutura:

1. Driver
2. Cluster Manager
3. Executors

Apache Spark



Apache Spark

- Cluster Managers: *Spark Standalone*, *YARN*, *Mesos*, and *Kubernetes*.

- Nosso foco será em execução local.

- Linguagens suportadas: Scala, Python, R, Java, etc.

Apache Spark

- Porta de entrada: Spark Session.
- Não precisamos nos preocupar com isso – interface com o R.
- Conceito do *sparklyr*

Apache Spark com sparklyr

```
install.packages("sparklyr")
```

```
spark_available_versions()
```

```
spark_install(version = "1.6")
```

Apache Spark

- Importante destacar do nosso exercício:

1. Vamos executar localmente.

2. Mais dados = melhor!

Apache Spark

colab

MBA USP ESALQ

Prof. Dr. Jeronymo Marcondes

https://www.linkedin.com/in/jeronymo-marcondes-585a26186