

Data Analytics com R e Bancos de Dados – SQL

Flávio Brito



DBI (*R* Database Interface)

- ❑ Disponibiliza uma interface comum a maioria dos pacotes do R para bancos de dados. Todas as classes neste pacote são virtuais, sendo assim necessitam ser estendidas por várias implementações utilizando códigos específicos em sub-pacotes
 - ❑ **RMySQL, RPostgreSQL, ROracle, RSQLite, RJDBC**
- ❑ Utiliza funções para abrir e fechar conexões
 - ❑ `dbConnect()`, `dbDisconnect()`,

```
con <- dbConnect(RMySQL::MySQL(),  
  username = "usuario",  
  password = "senha",  
  host = "192.168.100.143",  
  port = 3306,  
  dbname = "mysql")
```



Utilizando o DBI

- ❑ `dbListTables()` – lista as tabelas do banco de dados
- ❑ `dbReadTable()` e `dbWriteTable()` – são funções para executar respectivamente a leitura de tabela e escrita de tabelas. O `dbReadTable()` utiliza o identificador `row_names`

```
df <- dbReadTable(con, "funcionario")
```

- ❑ `dbGetQuery()` – submete o comando SQL e retorna o conjunto de dados após o processamento

```
df_bd <- dbGetQuery(con, "SELECT * FROM user")
```

As funções `dbSendQuery()` e `fetch()` – são utilizados para transmitir grande volume de dados

Outras funções avançadas também estão disponíveis para acesso a definição de schemas, manipulação de transações, e chamada de procedures.



SQL simples

- ❑ Buscar uma coluna sem filtragem e de forma distinta

```
df <-dbGetQuery(con, "SELECT DISTINCT codigo FROM pedido")
```

- ❑ Agregação e ordenação de resultados

```
df <-dbGetQuery(
```

```
  con,
```

```
  "SELECT cargo, avg(salario_liquido) as MEDIA_SAL  
  FROM folha GROUP BY cargo ORDER BY MEDIA_SAL DESC"
```

```
  )
```

Carregando um DF para o BD

```
data("mtcars")
df <- mtcars
dbWriteTable(con,
  "mtcars",
  value = df,
  append = TRUE,
  row.names = FALSE)
dbListTables(con)
df_bd <- dbGetQuery(con, "SELECT * FROM mtcars")
head(df_bd, 4)
```

	mpg	cyl	disp	hp	drat	wt	qsec	vs	am	gear	carb
1	21.0	6	160	110	3.90	2.620	16.46	0	1	4	4
2	21.0	6	160	110	3.90	2.875	17.02	0	1	4	4
3	22.8	4	108	93	3.85	2.320	18.61	1	1	4	1
4	21.4	6	258	110	3.08	3.215	19.44	1	0	3	1

Indo além

- ❑ # carregando os dados

```
tabela_B <- dbGetQuery(con, "SELECT * FROM amostras JOIN  
tabela_A USING (codigo)")
```

- ❑ # O mesmo resultado do join se consegue em R com a função merge()

```
tabela_B <- merge(amostras, tabela_A, by = "codigo")
```

- ❑ # usando a função merge só que para campos com nomes diferentes

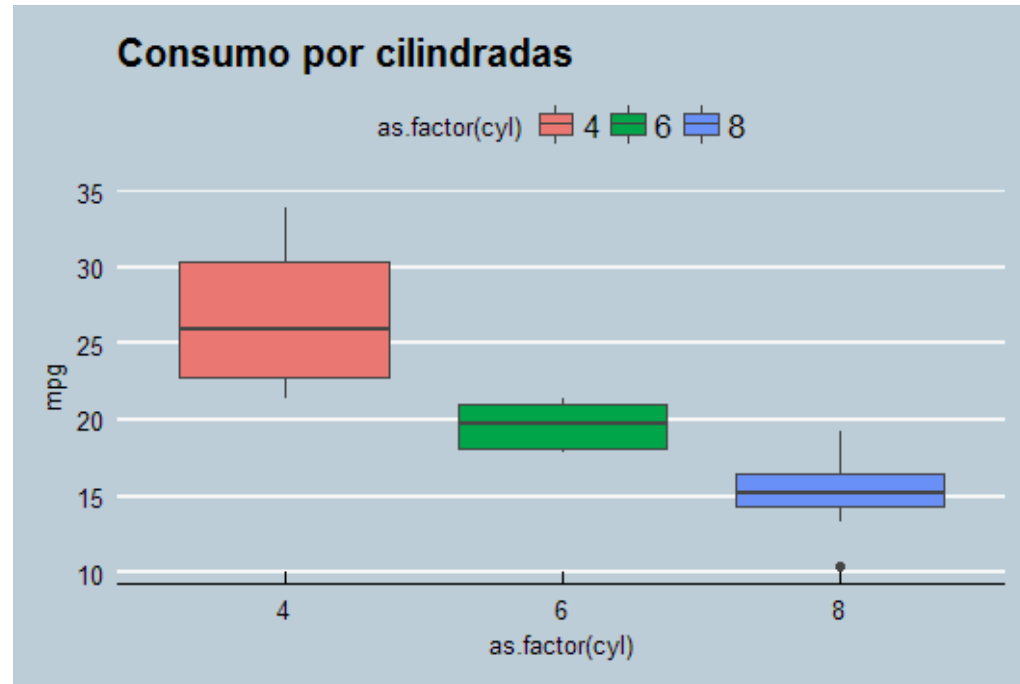
```
tabela_B <-
```

```
  merge(amostras, tabela_A, by.x = "cod_am", by.y =  
  "codigo_a")
```

```
dbDisconnect(con)
```

Visualizando os dados – BD, GGPLOT, GGTHEMES

```
require(ggplot2)
require(ggthemes)
df_bd <- dbGetQuery(con, "SELECT
* FROM mtcars")
g <- ggplot(df_bd, aes(
  x = as.factor(cyl),
  y = mpg,
  fill = as.factor(cyl)
)) +
  geom_boxplot()
g + theme_economist() +
  scale_color_economist() +
  ggtitle("Consumo por
cilindradas")
```



Exemplo de conexão com Bancos



MySQL (RMySQL)

- ❑ RMySQL: Esta versão está em acordo com as definições implementadas no pacote DBI 0.2-2.
- ❑ TSMysql: Ele fornece uma extensão da interface Tsdbi (Time Series Database Interface) para manipulação de séries temporais no MySQL.

```
con <- dbConnect(  
  RMySQL::MySQL(),  
  username = "usuario",  
  password = "senha",  
  host = "192.168.100.143",  
  port = 3306,  
  dbname = "mysql"  
)
```

PostgreSQL (RPostgreSQL)

- ❑ RPostgreSQL: Interface R para o Banco de Dados PostgreSQL
- ❑ TSPostgreSQL: Oferece uma extensão da interface Tdsdbi para PostgreSQL

```
library(RPostgreSQL)
con <-
  dbConnect(
    'PostgreSQL',
    user = 'USUARIO',
    password = 'SENHA', host = '192.168.1.10',
    port = 5432,
    dbname = 'MEU_BANCO')

data(iris)
dbWriteTable(con, 'iris', iris, row.names = FALSE)
output <- dbGetQuery(con, "SELECT * FROM iris")
summary(output)
```

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
Min. :4.300	Min. :2.000	Min. :1.000	Min. :0.100	setosa :50
1st Qu.:5.100	1st Qu.:2.800	1st Qu.:1.600	1st Qu.:0.300	versicolor:50
Median :5.800	Median :3.000	Median :4.350	Median :1.300	virginica :50
Mean :5.843	Mean :3.057	Mean :3.758	Mean :1.199	
3rd Qu.:6.400	3rd Qu.:3.300	3rd Qu.:5.100	3rd Qu.:1.800	
Max. :7.900	Max. :4.400	Max. :6.900	Max. :2.500	

ODBC (RODBC)

```
library(RODBC)
con <-odbcConnect("DSN1",
  uid = "usuario",
  pwd = "senha",
  rows_at_time = 500)
sqlSave(con, test_table, "TEST_TABLE")
sqlQuery(con, "SELECT COUNT(*) FROM TEST_TABLE")
d <-sqlQuery(con, "SELECT * FROM TEST_TABLE")
close(con)
```

❑ Onde:

- ❑ **odbcConnect(*dsn*, uid="", pwd="")** - abre uma conexão
- ❑ **sqlFetch(*channel*, *sqtable*)** – Lê uma tabela para um data frame
- ❑ **sqlQuery(*channel*, *query*)**
- ❑ **sqlSave(*channel*, *mydf*, tablename = *sqtable*, append = **FALSE**)** – grava ou atualiza (append =TRUE) um data frame para a tabela na base do ODBC



Oracle(RORACLE)

```
library(RJDBC)
drv <- JDBC("oracle.jdbc.OracleDriver",
  classPath = "...tklocal/instantclient_11_2/ojdbc5.jar", " ")
con <- dbConnect(drv, " jdbc:oracle:thin:@192.168.1.10:1521:db",
  "usuario", "senha")
dbWriteTable(con, "TEST_TABLE", test_table)
dbGetQuery(con, "SELECT COUNT(*) FROM TEST_TABLE")
d<- dbReadTable(con, "TEST_TABLE")
dbDisconnect(con)
```

```
library(ROracle)
drv<- dbDriver("Oracle")
con<- dbConnect(drv, "usuario", "senha")
dbWriteTable(con, "TEST_TABLE", test_table)
dbGetQuery(con, "SELECT COUNT(*) FROM TEST_TABLE")
d<- dbReadTable(con, "TEST_TABLE")
dbDisconnect(con)
```



MongoDB

- ❑ RMongo: Interface para R com MongoDB. Utiliza Java.
- ❑ rmongodb: Fornece interface ao MongoDB para R. Removido do CRAN.
- ❑ mongodblite – Interface leve para conexão do R com o MongoDB.

```
library(RMongo)
mongo <- mongoDbConnect("SER", "localhost", 27017)
output <-
  dbInsertDocument(
    mongo,
    "evento2",
    '{nome:"Flavio", sobrenome:"Brito",
email:"flaviobrito@live.com"}'
```

MongoDB

```
db.evento.drop()  
db.createCollection('evento');  
show collections  
db.evento.insert({ nome: "Flavio", sobrenome: "Brito",  
email: "flaviobrito@live.com" });  
db.evento.find()  
db.evento.insert([ { nome: "Antonio", sobrenome: "Jose",  
email: "ajose@gmail.com" }, { nome: "Jose", sobrenome:  
"Manoel", email: "jmanoel@gmail.com" } ] );  
db.evento.find()
```



MongoDB

//Atualizando

```
db.evento.update({ nome: "Flavio" }, { $set: {  
palestrante: "Sim" } });
```

```
db.evento.find({ nome: "Flavio" });
```

```
db.evento.find({ $or: [{ nome: "Manoel" }, { palestrante:  
"Sim" }] });
```

```
db.evento.find({ $and: [{ nome: "Manoel" }, { palestrante:  
"Sim" }] });
```

//ordenando

```
db.evento.find().sort({ nome: 1 }); //ascendente
```

```
db.evento.find().sort({ nome: -1 }); //descendente
```

//Contagem de documentos

```
db.evento.find().count();
```

```
db.evento.find({ palestrante: "Sim" }).count();
```



MongoDB

//Remover documentos

```
db.evento.remove({ sobrenome: "Manoel" });
```

//Limitando os registros

```
db.evento.find().limit(1)
```


SQLite

- ❑ RSQLite: Este pacote embarca a engine do banco de dados SQLite e fornece uma interface compatível com o pacote DBI
- ❑ TSSQLite: Fornece uma extensão da interface Tsdbi para o SQLite

```
con <- dbConnect(RSQLite::SQLite(), ":memory:")  
dbWriteTable(con, "iris", iris)  
dbGetQuery(con, "SELECT * FROM iris WHERE [Petal.Width] >  
2.3")  
dbDisconnect(con)
```

Data Analytics com R e Banco de Dados - SQL e NOSQL

Flávio Brito - Fundação CECIERJ

E-mail: flaviobrito@live.com



<https://flaviobrito.github.io/IISERUFF>

