BDs Relacionais & Programação Estatística

Disciplina MAC5861 - IME/USP

Caio Lente
Fernanda Fortti

13 de novembro de 2019

Motivação

O Problema

- Bancos de Dados Relacionais (BDRs) são limitados no que diz respeito à programação estatística
    - Moda, percentil, modelagem e cálculos estatísticos mais sofisicados são tarefas difíceis para Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDRs)

> *While basic aggregation operations (SUM, AVG) are part of SQL, there is no support for other commonly used operations like variance and covariance. Such computations, as well as more advanced ones like regression and principal component analysis, are usually performed using statistical packages and libraries.*
>
> (Srivastava & Ngo, 2014)

O Problema

- Ferramentas estatísticas nem sempre são viáveis com muitos dados
  - O que cabe em disco não necessariamente cabe na memória
  - Se os dados já estão em um BD, por que trazê-los para memória?

> *If your data fits in memory there is no advantage to putting it in a database: it will only be slower and more frustrating.*
>
> (Wickham, 2019)

Soluções

Solução 1

Implementação da ferramenta estatística dentro dos SGBDRs

- Limitações
  - Nem todo SGBDR apresenta essa funcionalidade (Postgres, p. ex.)
  - Vendedores dos SGBDRs precisam implementar a funcionalidade (o que pode custar caro e/ou demorar para ser feito)
  - Dificilmente a implementação será *open source*

```
-- calculate simple quantiles
EXEC sp_execute_external_script
  @language = N'R'
  ,@script = N'
     res <- quantile(InputDataSet$Ages);
    print(res);'
  ,@input_data_1 = N'
    SELECT Ages = DATEDIFF(YEAR,[BirthDate],GETDATE())
    FROM [AdventureWorksDW2014].[dbo].[DimCustomer];';
```

Solução 2

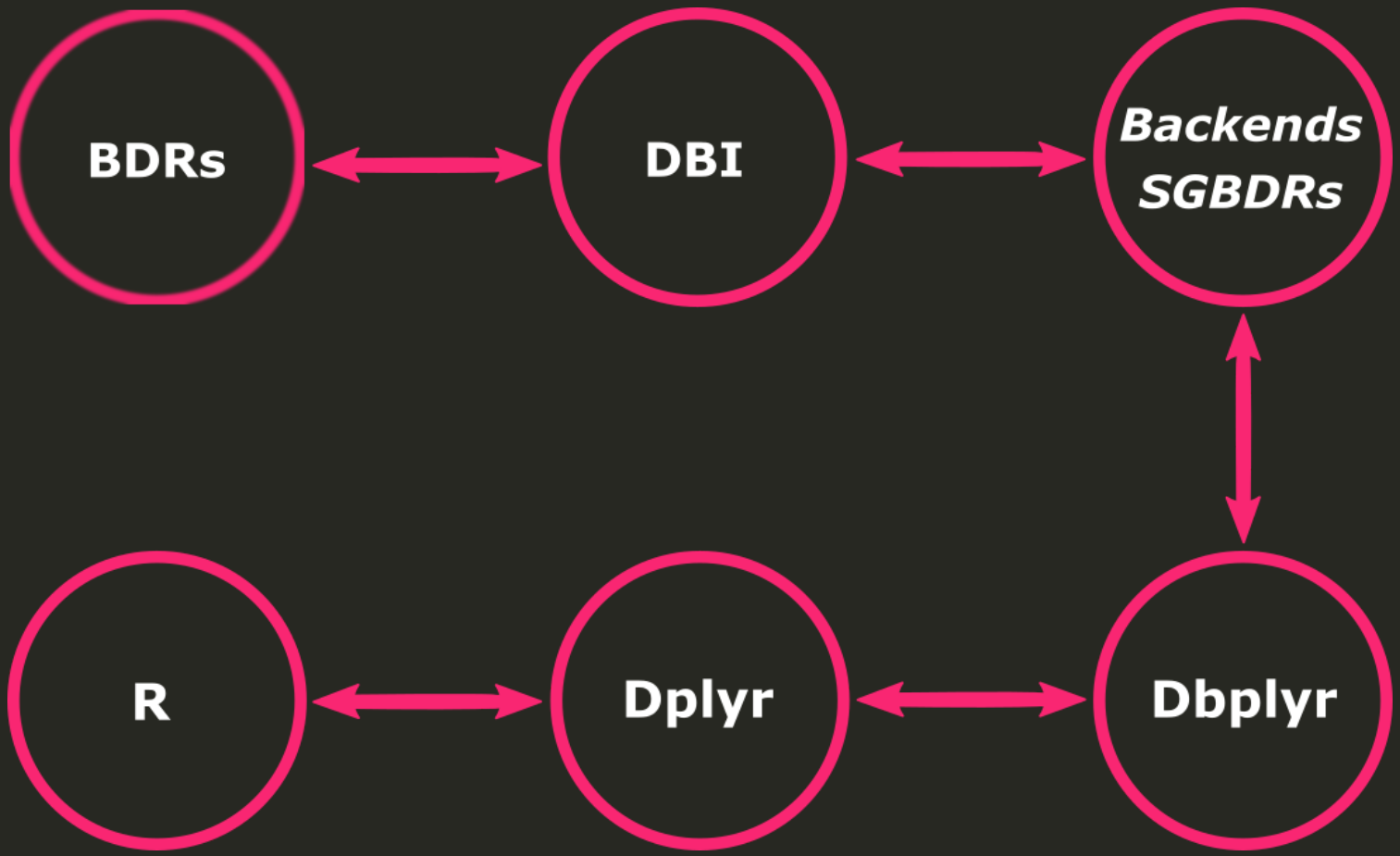
Conexão do software estatístico aos SGBDRs

- Por que usar o **R** como software e principal linguagem estatística?
  - Linguagem estatística mais amplamente adotada
  - *Open source*
  - Comunidade ativa

```
con_postgres <- RPostgres::dbConnect(
  RPostgres::Postgres(),
  dbname = "my_database",
  host = "localhost", port = 54320,
  user = "postgres", pwd
)
```

Dbplyr

Diagrama



BDRs

Qualquer SGBDR pode ser suportado desde que um *backend* seja criado

- Aproveita-se do fato de que todo SGBDR tem suporte para conexões externas

DBI

É uma biblioteca que fornece uma interface consistente para comunicação entre R e SGDBRs

- Permite que o R envie comandos para os bancos de dados, independentemente do SGBDR utilizado
- Padronização das funções possibilita a construção de pacotes genéricos para lidar com os dados de uma conexão, facilitando:
    - A criação e implementação dos pacotes (necessidade de menos decisões arbitrárias)
    - A manipulação dos usuários (padrão familiar)
- Fornece uma especificação de interface que permite qualquer um criar novos *backends* para um BDR

Backends dos SGBDRs

Bibliotecas específicas que permitem a conexão entre o R e SGBDRs

- Permite o envio de *queries* e recebimento de dados do BDR
- Permite ao `dplyr` trabalhar com múltiplas fontes de dados usando o mesmo código
- Funções de leitura (`dbReadTable`) e escrita (`dbWriteTable`) de tabelas através de uma conexão
- Exemplos:
  - `RPostgres`
  - `RSQLite`
  - `RMariaDB`
  - `bigrquery`

BDRs, DBI, *Backends* dos SGBDRs

Os argumentos para `DBI::dbConnect()` variam de BDR para BDR, mas o primeiro argumento é sempre o *backend* do BDR:

- `RSQLite::SQLite()` para `RSQLite`
- `RPostgreSQL::PostgreSQL()` para `RPostgreSQL`
- `RMySQL::MySQL()` para `RMySQL` (abandonado)
- `RMariaDB::MariaDB()` para `RMariaDB`*
- `bigrquery::bigquery()` para `BigQuery`

***Funciona de backend para MariaDB e MySQL; substituiu o pacote `RMySQL`**

`SQLite` precisa de outro argumento: o caminho para o BDR. Porém, a maioria dos BDRs não está em um arquivo, mas sim em outro servidor. Isso faz com que, na prática, o código seja semelhante a:

```
con <- RPostgres::dbConnect(
  RPostgres::Postgres(), dbname = "alunos",
  host = "database.ime.com", port = 54320,
  user = "professorMAC5861"
  password = rstudioapi::askForPassword("database_password")
)
```

Dbplyr

Pacote do R para tradução de código `dplyr` em código SQL e tradução da resposta do BDR em um *R data frame*

- De forma automática, consegue gerar um código em SQL a partir de um código em R
- Capaz de transformar verbos do pacote do R `dplyr` em *queries* SQL completas (atua, portanto, como *backend* do `dplyr`)
- Permite receber resultados parciais dos SGBDRs e traduzí-los em uma resposta interpretável pelo R
- Não consegue traduzir qualquer código, mas ajuda bastante

```
dbplyr::translate_sql(x ^ 2L)
#> <SQL> POWER(`x`, 2)

dbplyr::translate_sql(log(x, 10))
#> <SQL> LOG(10.0, `x`)
```

Dplyr

Pacote do R com gramática para manipulação de dados

- Fornece um conjunto consistente de verbos que ajudam a resolver os desafios mais comuns de manipulação de dados
- Transformação sintática sobre tabelas
- *Pipelines*, o que permite que cada linha delimite uma ação, como se fosse uma receita de bolo
- Operadores infixos, o que permite que se saiba exatamente a ordem das operações
- Metaprogramação, o que garante um código mais limpo
- *Quasi-quotations*, o que garante a escrita de funções confiáveis que reduzem a redundância no código
- Uma das bibliotecas mais utilizadas, graças às vantagens mencionadas

Exemplos

Dplyr

```
cnes %>%
  select(def_atividad, def_turno_at, starts_with("mun_")) %>%
  filter(def_atividad == "Unidade SEM atividade de Ensino") %>%
  mutate(
    so_um_turno = ifelse(grepl("SOMENTE", def_turno_at), 1, 0),
    localizacao = ifelse(mun_CAPITAL == "S", "CAPITAL", "INTERIOR")
  ) %>%
  group_by(localizacao) %>%
  summarise(f_so_um_turno = sum(so_um_turno)*100/n())

#> # A tibble: 2 x 2
#>   localizacao f_so_um_turno
#>   <chr>               <dbl>
#> 1 CAPITAL              1.10
#> 2 INTERIOR             1.80
```

Dplyr + Dbplyr

O processo é *lazy*, nunca puxa dados para o R, a menos que você os solicite explicitamente com `dplyr::collect()`.

```
tbl(con, "cnes") %>%
  select(def_atividad, def_turno_at, starts_with("mun_")) %>%
  filter(def_atividad == "Unidade SEM atividade de Ensino") %>%
  mutate(
    so_um_turno = ifelse(grepl("SOMENTE", def_turno_at), 1, 0),
    localizacao = ifelse(mun_CAPITAL == "S", "CAPITAL", "INTERIOR")
  ) %>%
  group_by(localizacao) %>%
  summarise(f_so_um_turno = sum(so_um_turno)*100/n()) %>%
  collect()
```

```
#> # A tibble: 2 x 2
#>   localizacao f_so_um_turno
#>   <chr>               <dbl>
#> 1 CAPITAL              1.10
#> 2 INTERIOR             1.80
```

Tradução

```
dplyr::show_query()
```

```
dplyr::explain()
```

```
SELECT "localizacao", SUM("so_um_turno") * 100.0 / COUNT(*) AS "f_so_
FROM (SELECT "def_atividad", "def_turno_at", "mun_MUNNOME", "mun_MUNN
FROM (SELECT *
FROM (SELECT "def_atividad", "def_turno_at", "mun_MUNNOME", "mun_MUNN
FROM "cnes") "dbplyr_1033"
WHERE ("def_atividad" = 'Unidade SEM atividade de Ensino')) "dbplyr_1
GROUP BY "localizacao"
```

```
HashAggregate  (cost=4091.28..4091.33 rows=2 width=64)
  Group Key: CASE WHEN (cnes."mun_CAPITAL" = 'S'::text) THEN 'CAPITAL
  ->  Seq Scan on cnes  (cost=0.00..3810.18 rows=22488 width=77)
        Filter: (def_atividad = 'Unidade SEM atividade de Ensino'::te
```

Desempenho

```
microbenchmark::microbenchmark()
```

```
# Unit: milliseconds
#                  expr        min     median        max
#               turno()   25.87095   25.92416   126.6529
#  turno(con_postgres)  151.11062  153.43344  178.16157
#    turno(con_sqlite)   75.66695   76.58116   83.03558
#     turno(con_mysql)  188.36329  191.83665  200.08846
#     turno(con_maria)  172.39715  181.23544  188.76618
```

```
#        turno(con_bq) 4231.28270 4247.64543 4975.46140
```

Referências

Srivastava & Ngo, 2014

Wickham, 2019

Databases

Executing R code in SQL Server

R, Databases & SQL

SQL databases and R

Write advanced R functions with SQL Server Machine Learning Services

R and Data – When Should we Use Relational Databases?

Database Queries With R

nodbi: the NoSQL Database Connector

Advanced R

Referências

Implementing a new backend: DBI

DBI

Database basics - dplyr and DBI

Introduction to dbplyr

Function translation

Verb translation

Databases using dplyr

Introduction to dplyr

Programming with dplyr

PostgreSQL 12.0 Documentation

RMariaDB

Obrigada(o)!