dplyr

Isidio Martins e Tomás Barcellos

27 janeiro 2017

Objetivo

- Apresentar pacotes para a exploração dos dados que permitam o uso mais intuitivo do R
- O método é dinâmico: O participante deve replicar toda a programação apresentada

Principais aspectos do R

- Múltiplas ferramentas de análise no mesmo ambiente com a mesma linguagem (ex: mapas e regressões)
- Controle de alterações (ex: exclusões e imputações) sem alterar os dados originais
- Reprodutibilidade
- Escala (looping)
- Software Livre e Aberto com baixo uso de RAM
- Pacotes: Nada mais são que um conjunto de funções
  - Estes são o meio pelo qual o programa evolui e se adapta a necessidade dos usuários.
- Flexibilidade
- Comunidade (múltiplas formações acadêmicas)

Linguagem de Programação R

Não se tratam de funções inalteraveis, tudo pode ser reformulado. Observe o código da função que calcula o Índice de Hirschman-Herfindahl de concentração de mercado disponível no pacote antitrust, dê o comando (sem parenteses):

```
antitrust::HHI
```

- No R temos funções proprias do R-BASE, próprias do programador e de outros na forma de Pacotes
- Encontrando soluções
  - Help(),'?' e '??'
  - A comunidade do R é excepcional

Exemplo de função

A sintaxe de uma função consiste no nome da função seguindo por parenteses, dentro os argumentos (inputs), por exemplo:

```
toupper(x = "mapa") #Apresenta os caracteres em caixa alta
```

```
## [1] "MAPA"
```

Tab

Decorar o nome de cada função ou seus argumentos seria insano. A tecla Tab ajuda a elencar o que pode ser feito.

- Encontrando funções:
  - Tecle 'me' e depois Tab
- Funções dentro de pacotes específicos:
  - Tecle 'dplyr::' e depois Tab
- Argumentos dentro de funções:
  - Tecle 'dplyr::full_join()' e dentro dos parênteses tecle Tab

Antes do dplyr: O *Assignment Operator* (Operador de atribuição: '<-')

O R opera com objetos. "Objeto é aonde você guarda o que quer". A sintaxe é: à esquerda do operador ("<-") dá-se nome ao objeto (um texto qualquer), à direita o elemento a ser inserido nele. Exemplos:

```
qualquercoisa<-c(1,2,3)
qualquercoisa
```

```
## [1] 1 2 3
```

```
qualquercoisa<-"mapa"
qualquercoisa
```

```
## [1] "mapa"
```

Antes do dplyr: O *Assignment Operator* (Operador de atribuição: '<-')

Mais exemplos...

```
qualquercoisa<-toupper(x = "mapa")
qualquercoisa
```

```
## [1] "MAPA"
```

```
qualquercoisa<-"mapa" #Podemos guardar em um objeto...
qualquercoisa<-toupper(x = qualquercoisa) # ....e depois r
qualquercoisa
```

```
## [1] "MAPA"
```

Antes do `dplyr`: Caregando tabelas no R

Localize seu diretório de trabalho com o comando `getwd()` e lá coloque os arquivos disponíveis em `https://github.com/tomasbarcellos/palestra-MAPA`.

Se o interesse é o de ler dados externos estão começe escrevendo 'read' e teclando Tab. Surgirão várias funções para a leitura.

Opinião do autor: "Sempre que possível, use o".csv". Apesar de mais pesado, é mais seguro para guardar informação e é lido por praticamente qualquer programa. Usemos o `read.csv2()`.

Antes do dplyr: Caregando tabelas

Escreva `read.csv2("")` e clique entre as aspas e tecle Tab.
Surgirá uma lista de arquivos disponíveis no seu WD.

```
PAM<-read.csv2("PAM.csv", stringsAsFactors = F)
#Sempre ponha stringsAsFactors = F, até o dia em que soube
str(PAM)
```

```
## 'data.frame':    42900 obs. of  17 variables:
##  $ X                                : int  1 2 3 4 5 6 7 8 9
##  $ ano_cod                          : int  1990 1990 1990 199
##  $ ano_desc                         : int  1990 1990 1990 199
##  $ var_cod                          : int  109 109 109 109 10
##  $ regiao_cod                       : int  1 1 1 1 1 1 1 1 1
##  $ regiao_desc                      : chr  "Norte" "Norte" "N
##  $ cultura_cod                      : int  2688 2689 2690 269
##  $ cultura_desc                     : chr  "Abacaxi" "Algodão
##  $ unidade_cod                      : int  1006 1006 1006 100
```

Antes do dplyr: Caregando tabelas

Quando os dados estiverem em '.xls' ou '.xlsx', basta usar o pacote `readxl`.

```
library(readxl)
PAM2<-readxl::read_excel("PAM.xlsx")
str(PAM2)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    42900 obs.
##  $ ano_cod                      : num  1990 1990 1990 199
##  $ ano_desc                     : num  1990 1990 1990 199
##  $ var_cod                      : num  109 109 109 109 10
##  $ regiao_cod                   : num  1 1 1 1 1 1 1 1 1
##  $ regiao_desc                  : chr  "Norte" "Norte" "N
##  $ cultura_cod                  : num  2688 2689 2690 269
##  $ cultura_desc                 : chr  "Abacaxi" "Algodão
##  $ unidade_cod                  : num  1006 1006 1006 100
##  $ unidade_desc                 : chr  "Hectares" "Hectar
```

Antes do dplyr: Pipe Operator '%>%'

O pipe operator esta disponível no pacote magrittr. Outros pacotes tem o magrittr como seu dependente. Ou seja, carregam-no para poder operar. O pacote dplyr é um deles.

O pipe inverte a lógica de se primeiro determinar qual o função a ser usada para depois escolher o objeto a ser transformado. Ou seja, tal como a realidade, parte-se do objeto a ser analisado para a exploração (cálculos e transformações).

```
library(magrittr)
"mapa" %>% toupper()
```

```
## [1] "MAPA"
```

O dplyr

Trata-se de um pacote disponibilizado por Hadley Wickham:

Instalando o dplyr

Os pacotes não vem previamente instalados. Para instalar qualquer pacote deve-se usar o comando: `install.packages()`. Portanto, para instalar o dplyr, execute: `install.packages("dplyr")`

É possível também instalar o pacote por meio de um arquivo .zip disponivel em `https://cran.r-project.org/`.

Dê o comando `library(dplyr)` para carregar o pacote.

Operadores básicos

Comando	Ação
select(col.1, col.2, .)	Selecione das variáveis existentes (colunas)
filter(condição.1, condição.2, . . .)	Filtra a tabela por condições
arrange(col.1, col.2, .)	Classifica a tabela por variáveis ou outros comandos lógicos
mutate(qualquer_nova_coluna = .)	Crie novas variáveis
group_by() + summarize()	Resuma os dados por grupo

Exercício 1

Produtividade Média da Uva por Região a partir de 2010

Selecione as variáveis relevantes:

```
p1 <- PAM %>%
  select(ano_desc, regiao_desc, cultura_desc,
         Rendimento.médio.da.produção)
```

O resultado esta sendo guerdado no objeto `p1` (passo 1)

Passo 1

Para uma melhor vizualização dê o comando: `View(p1)`. Notem os valores `NA`.

ano_desc	regiao_desc	cultura_desc	Rendimento.n
1990	Norte	Abacaxi	
1990	Norte	Algodão herbáceo (em caroço)	
1990	Norte	Alho	
1990	Norte	Amendoim (em casca)	

Passo 2

Filtre por ano e cultura:

```
p2 <- PAM %>%
  select(ano_desc, regiao_desc, cultura_desc,
         Rendimento.médio.da.produção) %>%
  filter(ano_desc >= 2010, cultura_desc=="Uva")
```

Visualize com o comando: `View(p2)`

Passo 2

ano_desc	regiao_desc	cultura_desc	Rendimento.médio.da.produção
2010	Norte	Uva	NA
2010	Nordeste	Uva	NA
2010	Sudeste	Uva	NA
2010	Sul	Uva	NA

Passo 3

Agrupe (desagrupe) a sua base por Região: Mas primeiro...

Eis o que está sendo omitido no processo:

ID	Sexo	Idade
01	F	35
02	M	20
03	M	40
04	F	18
05	F	45

Passo 3

A tabela é desgregada em duas..

ID	Sexo	Idade
01	F	35
04	F	18
05	F	45

Passo 3

E os cálculos são realizados para cada tabela. Por exemplo, a idade média dos Sexos.

ID	Sexo	Idade
02	M	20
03	M	40

Passo 3

Enfim,

```
p3 <- PAM %>%
  select(ano_desc, regiao_desc, cultura_desc,
         Rendimento.médio.da.produção) %>%
  filter(ano_desc >= 2010, cultura_desc=="Uva") %>%
  group_by(regiao_desc)
```

Passo 4

Calcule o rendimento médio por agrupamento.

```
p4 <- PAM %>%
  select(ano_desc, regiao_desc, cultura_desc,
         Rendimento.médio.da.produção) %>%
  filter(ano_desc >= 2010, cultura_desc=="Uva") %>%
  group_by(regiao_desc) %>%
  summarize(Rendimento_Médio_Regional_Uva =
              mean(Rendimento.médio.da.produção, na.rm = TR
```

Passo 4

regiao_desc	Rendimento_Médio_Regional_Uva
Centro-Oeste	22945.500
Nordeste	30197.500
Norte	7283.333
Sudeste	18882.167

Passo 5

Ordene os dados do rendimento médio de forma decrescente:

```
p5 <- PAM %>%
  select(ano_desc, regiao_desc, cultura_desc,
         Rendimento.médio.da.produção) %>%
  filter(ano_desc >= 2010, cultura_desc=="Uva") %>%
  group_by(regiao_desc) %>%
  summarize(Rendimento_Médio_Regional_Uva =
              mean(Rendimento.médio.da.produção, na.rm = TR
  arrange(desc(Rendimento_Médio_Regional_Uva))
```

Passo 5

regiao_desc	Rendimento_Médio_Regional_Uva
Nordeste	30197.50
Centro-Oeste	22945.50
Sudeste	18882.17
Sul	16072.17

Passo 5: Ordenando o objeto 4

Cada novo comando dado após o pipe trabalha em cima de um novo dataframe: com menos colunas, menos linhas, agrupado por região com uma nova variavél chamada "Rendimento_Médio_Regional_Uva".

Essa variável sequer existia na tabela inicial! De forma equivalente, o passo 5 pode se realizar sob a tabela encontrada no passo 4 (p4).

```
p5 <- p4 %>%
  arrange(desc(Rendimento_Médio_Regional_Uva))
```

Passo 5: Ordenando o objeto 4

regiao_desc	Rendimento_Médio_Regional_Uva
Nordeste	30197.50
Centro-Oeste	22945.50
Sudeste	18882.17
Sul	16072.17

“Dividir, Aplicar e Combinar”

O padrão de análise que realizamos anteriormente é recorrente. Mais conhecido como “Split, Apply and Combine” (SAC). No passo 3 dividimos o dataframe pela variável região, aplicamos uma função sobre uma das variaveis em cada parte sepradamente combinando os resultados na forma de uma nova variavel.

Exercicio 2

Agora é com vocês, cheguem no resultado abaixo. Se não conhecem alguma função, usem os meios de busca elencados bem como o google com a tag [R].

A quantidade produzida anual de Laranja desde 2010 na região sul:

ano_desc	regiao_desc	cultura_desc	Quantidade.produzida
2010	Sul	Laranja	NA
2010	Sul	Laranja	NA
2010	Sul	Laranja	1053206
2010	Sul	Laranja	NA
2010	Sul	Laranja	NA
2011	Sul	Laranja	NA
2011	Sul	Laranja	NA
2011	Sul	Laranja	1257463
2011	Sul	Laranja	NA
2011	Sul	Laranja	NA

Exercicio 3

O coeficiente de variação (usa o google!! você terá de criar a função) das produtividades anuais da Manga, por região.

regiao_desc	CV
Centro-Oeste	NaN
Nordeste	NaN
Norte	NaN
Sudeste	NaN
Sul	NaN

Exercicio 4

O valor da produção no ano de 2013 para as culturas de Uva e Abacaxi:

Dica: Para filtrar por vários valores siga o padrão: 'cultura_desc %in% c("Uva","Abacaxi")'

cultura_desc	Valor
Abacaxi	NA
Uva	NA