dplyr

Isidio Martins e Tomás Barcellos

27 de janeiro de 2017

Objetivo

- Apresentar pacotes para a exploração de dados. Estes permitirão um 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 que um conjunto de funções
  - O programa evolui por meio dos pacotes
- Flexibilidade para criar soluções
- Comunidade (múltiplas formações acadêmicas)

Linguagem de Programação R

Tudo pode ser reformulado! Observe o código da função que calcula a correlação entre dois vetores, dê o comando:

```
cor
```

- No R temos funções proprias do R-BASE, 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"
```

A tecla Shift

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

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

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), à direita o elemento a ser inserido:

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

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

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

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

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

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

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

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

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

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`.

Escreva 'read' e tecle Shift. Surgirão várias funções para a leitura de dados externos.

Opinião: "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()`.

Caregando tabelas (.csv)

Escreva `read.csv2("")`, clique entre as aspas e tecle Shift.
Surgirão os arquivos disponíveis no seu WD.

```
PAM<-read.csv2("PAM.csv", stringsAsFactors = F,
               encoding = "UTF-8")
#Sempre use stringsAsFactors = F, até o dia em que souber
```

Para uma melhor viasualização da tabela carregada de o comando:
`View(PAM)`

Estrutura da tabela

```
str(PAM) # Nomes das variáreis, classes, amostras. Obs: Asp
```

```
## 'data.frame':    8580 obs. of  14 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
##  $ 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
##  $ cultura                          : chr  "temporária" "temp
##  $ Área.colhida                     : int  1955 17582 NA 133
##  $ Área.destinada.à.colheita        : int  NA NA NA NA NA NA
##  $ Área.plantada                    : int  2234 17582 NA 138
##  $ Quantidade.produzida             : int  27522 13732 NA 157
##  $ Rendimento.médio.da.produção: num  14077 781 NA 1180
```

Caregando tabelas (.xls e .xlsx)

O pacote `readxl` tem sido a melhor solução.

```
PAM2<-readxl::read_excel("PAM.xlsx")
head(PAM2,4) # Apresenta os primeiros valores
```

```
## # A tibble: 4 × 14
##       `` ano_cod ano_desc regiao_cod regiao_desc cultura_
##    <dbl>   <dbl>    <dbl>      <dbl>       <chr>       <c
## 1      1    1990     1990          1       Norte       2
## 2      2    1990     1990          1       Norte       2
## 3      3    1990     1990          1       Norte       2
## 4      4    1990     1990          1       Norte       2
## # ... with 8 more variables: cultura_desc <chr>, cultura
## #   colhida` <chr>, `Área destinada à colheita` <chr>, `
## #   plantada` <chr>, `Quantidade produzida` <chr>, `Rend
## #   produção` <chr>, `Valor da produção` <chr>
```

Pipe Operator '%>%'

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

O pipe inverte a lógica de se primeiro determinar qual a 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. Dentre as contribuições de HW estão os pacotes `dplyr`, `ggplot2` e `tidyr`.

O `dplyr` simplifica as tarefas de transformar um dataframe.

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 do dplyr

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 está sendo guardado 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	7150
2010	Nordeste	Uva	26709
2010	Sudeste	Uva	18674
2010	Sul	Uva	14451

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

Para realizar uma soma esse procedimento parece exagerado. Mas no último exemplo podemos mostrar q ele pode ser bem útil.

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.

Exercício 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	1053206
2011	Sul	Laranja	1257463
2012	Sul	Laranja	1338379
2013	Sul	Laranja	1424666

Exercício 3

A média das produtividades anuais da Manga, por região.

regiao_desc	Média
Centro-Oeste	15549.85
Nordeste	25477.69
Norte	23577.73
Sudeste	18753.04
Sul	19040.85

Exercício 4

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	0.3634159
Nordeste	0.4810500
Norte	0.8558595
Sudeste	0.3265371
Sul	0.4176403

Exercício 5

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	1854310
Uva	2120892

Exercício 6

O valor da produção por região, nos anos de 2010 e 2015

regiao_desc	ano_desc	Valor
Centro-Oeste	2010	30151873
Centro-Oeste	2015	70019558
Nordeste	2010	22390610
Nordeste	2015	34804885
Norte	2010	7075639
Norte	2015	14432943

Exercício 7

Calcule o coeficiente de correlação entre o valor da produção e a área destinada à colheita para as culturas de Café (em grão) Arábica e laranja.

cultura_desc	COR
Café (em grão) Arábica	NA
Laranja	0.2524253

Exercício Final (Very Hard)

Calcule os betas estimados da regressão entre o valor da produção e a área destinada à colheita para as culturas de Café (em grão) Arábica e laranja.

```
BETA<-function(x,y){
  LM<-lm(x~y)
  LM<-LM$coefficients
  LM<-as.numeric(LM[2])
  LM
}
ex.final<-
  PAM %>%
  select(ano_desc,cultura_desc,Valor.da.produção,
         Área.destinada.à.colheita) %>%
  filter(cultura_desc %in% c("Café (em grão) Arábica","Lara
  group_by(cultura_desc) %>%
  summarise(Beta=BETA(Valor.da.produção,Área.destinada.à.co
```

Exercício Final (Very Hard)

cultura_desc	Beta
Café (em grão) Arábica	8.231372
Laranja	578.507335