Introdução ao R

Willber Nascimento

2022-06-23

Aula 3 - Introdução a manipulação de dados com R

Importação de dados no R

O R possui suporte nativo ou por meio de pacotes a muitos formatos de arquivos de dados. Vamos mostrar como importar os mais comuns.

CSV

Um arquivo .csv é um arquivo de texto que pode ser lido por qualquer editor de texto. Normalmente ele é padronizado de modo a usar algum caractere específico (tal como `,`, `;`, `tab` e outros) para separar as colunas (Variaáveis) do banco de dados. Para carregar esse tipo de arquivo, você pode usar a função `read.csv` do R.

Observe o padrão para ler um arquivo do tipo '.csv'.

```
df <- read.csv('endereco do arquivo.csv',
                   header = TRUE,
                   sep = 'separador',
                   dec = 'decimal',
                   fileEncoding = 'encoding')
```

Transformar estrutura

Os dados podem ser estruturados na forma longa ou forma larga. Na forma longa, cada caso é uma nova observação e cada coluna uma unica característica - nenhuma outra coluna apresenta a mesma caracteristica (seja por ano, sexo, região, etc). No formato largo, isso é possível. Isto é, há várias colunas para a mesma característica sendo uma variação o tempo ou outra categoria. Veja um exemplo https://www.statology.org/long-vs-wide-data/.

Vamos importar o banco de dados `homicidios_homens_mulheres.csv`.

```
homicidios <- read.csv('../dados/homicidios_homens_mulheres.csv',
                       header = T,
                       sep = ';'
                       )
head(homicidios)
```

```
##   cod nome periodo homens mulheres
## 1  12   AC    2010    146       19
## 2  27   AL    2010   1950      137
## 3  13   AM    2010   1017       65
## 4  16   AP    2010    244       16
## 5  29   BA    2010   5398      438
## 6  23   CE    2010   2515      173
```

Note que a coluna homens e a coluna mulheres trata-se da mesma informação (Homicídios) em duas colunas que indicam o sexo. O ideal, para muitas análises, é que houvesse uma coluna `sexo` e uma coluna `homicidios` separadamente [1]. Logo, precisamos transformar essa estrutura (transpor). Para isso vamos usar um pacote chamado `reshape2`](https://www.rdocumentation.org/packages/reshape2/versions/1.4.4). Para isso, instale e o carregue na sessão.

```
install.packages('reshape2')
library(reshape2)
```

Agora, podemos usar a função `melt` para transformar dados de um formato largo (wide) para um formato longo. E podemos usar a função `dcast` para fazer o contrário, transformar de um formato longo para um largo.

```
melt(homicidios, id.vars = c('cod', 'nome', 'periodo'))
```

```
##    cod nome periodo variable value
## 1   12   AC    2010   homens   146
## 2   27   AL    2010   homens  1950
## 3   13   AM    2010   homens  1017
## 4   16   AP    2010   homens   244
## 5   29   BA    2010   homens  5398
## 6   23   CE    2010   homens  2515
## 7   53   DF    2010   homens   720
## 8   32   ES    2010   homens  1618
## 9   52   GO    2010   homens  1795
## 10  21   MA    2010   homens  1402
## 11  31   MG    2010   homens  3236
## 12  50   MS    2010   homens   579
## 13  51   MT    2010   homens   890
## 14  15   PA    2010   homens  3286
## 15  25   PB    2010   homens  1336
## 16  26   PE    2010   homens  3226
## 17  22   PI    2010   homens   371
## 18  41   PR    2010   homens  3248
## 19  33   RJ    2010   homens  5310
## 20  24   RN    2010   homens   739
## 21  11   RO    2010   homens   508
## 22  14   RR    2010   homens   110
## 23  43   RS    2010   homens  1856
## 24  42   SC    2010   homens   712
## 25  28   SE    2010   homens   633
## 26  35   SP    2010   homens  5355
## 27  17   TO    2010   homens   293
```

[1] Você pode ter uma boa explicação no capítulo 12 do R For Data Science

```
## 28  12   AC    2010 mulheres    19
## 29  27   AL    2010 mulheres   137
## 30  13   AM    2010 mulheres    65
## 31  16   AP    2010 mulheres    16
## 32  29   BA    2010 mulheres   438
## 33  23   CE    2010 mulheres   173
## 34  53   DF    2010 mulheres    66
## 35  32   ES    2010 mulheres   174
## 36  52   GO    2010 mulheres   182
## 37  21   MA    2010 mulheres   117
## 38  31   MG    2010 mulheres   409
## 39  50   MS    2010 mulheres    76
## 40  51   MT    2010 mulheres    80
## 41  15   PA    2010 mulheres   231
## 42  25   PB    2010 mulheres   119
## 43  26   PE    2010 mulheres   247
## 44  22   PI    2010 mulheres    40
## 45  41   PR    2010 mulheres   338
## 46  33   RJ    2010 mulheres   339
## 47  24   RN    2010 mulheres    71
## 48  11   RO    2010 mulheres    37
## 49  14   RR    2010 mulheres    11
## 50  43   RS    2010 mulheres   227
## 51  42   SC    2010 mulheres   110
## 52  28   SE    2010 mulheres    43
## 53  35   SP    2010 mulheres   678
## 54  17   TO    2010 mulheres    34
```

Agora temos um banco de dados onde cada observação representa uma UF e um Sexo. Veja que passamos um conjunto de variáveis como variáveis de ID (`id.vars`). Por padrão fazemos isso para garantir que essas variáveis não sejam transformadas também, já que elas são as unidades de análise. O padrão da função melt é usar as variáveis que sobraram e são caracteres como IDs, mas nem sempre isso será o desejado, por isso opte por expressar esses parâmetros.

Note que por padrão a função nomeia a variável categórica (de grupo) de `variable` e o resultado numérico de `value`, mas podemos alterar esse comportamento passando outros parâmetros.

```
df_longo <-
melt(homicidios, id.vars = c('cod', 'nome', 'periodo'),
                 variable.name = 'sexo', value.name = 'homicidios')

head(df_longo)
```

```
##   cod nome periodo   sexo homicidios
## 1  12   AC    2010 homens        146
## 2  27   AL    2010 homens       1950
## 3  13   AM    2010 homens       1017
## 4  16   AP    2010 homens        244
## 5  29   BA    2010 homens       5398
## 6  23   CE    2010 homens       2515
```

Agora que aprendemos a transpor o para formato longo, podemos fazer a operação inversa e transformar para o formato largo. Para isso usaremos a função `dcast`. Diferente de melt deveremos usar uma notação de fórmula

```
df_largo <- dcast(df_longo, formula = cod + nome + periodo ~ sexo, value.var = 'homicidios')
head(df_largo)
```

```
##   cod nome periodo homens mulheres
## 1  11   RO    2010    508       37
## 2  12   AC    2010    146       19
## 3  13   AM    2010   1017       65
## 4  14   RR    2010    110       11
## 5  15   PA    2010   3286      231
## 6  16   AP    2010    244       16
```

Todas variáveis de ID precisam estar à direta do acento til (~) e aquelas que irão se transformar em colunas ficaram após o til (~). Precisamos indicar no parâmetro `value.var` a variável que será distribuida nas novas colunas, isto é, a variável que detem os valores. No nosso exemplo, as novas colunas serão o `sexo` e os valores será `homicidios`.

Merge de Banco de dados

Frequentemente precisamos criar um banco de dados juntando informações que estão em outros bancos de dados. Você já aprendeu a usar `rbind` e `cbind`para adicionar novas linhas e colunas, mas elas levam em conta a ordem das informações e o tamanho do banco de dados para fazer a operação corretamente. O R conta com a função `merge` que é muito útil para fazer esse tipo de manipulação juntando dados baseados em variáveis de ID - isto é, não é necessário ordernar os dados em uma ordem específica, por exemplo.

Vamos ver na prática. Nossa missão é adicionar a população de cada um dos estados no nosso banco de dados. Carregue o banco de dados `populacao.xlsx`. Primeiro você vai precisar carregar o pacote `readxl` na sua sessão.

```
install.packages('readxl')
library(readxl)
```

Agora podemos importar o banco de dados.

```
populacao <- read_excel('../dados/populacao.xlsx')
head(populacao)
```

```
## # A tibble: 6 x 2
##   SiglaUF `População total 2010`
##   <chr>                    <dbl>
## 1 AC                      733559
## 2 AL                     3120494
## 3 AP                      669526
## 4 AM                     3483985
## 5 BA                    14016906
## 6 CE                     8452381
```

Para fazer um *merge* precisamos que os dois bancos de dados possuam uma mesma chave, ou ID. Isto é, é necessário que há uma variável de identificação comum aos bancos. No nosso caso, essa chave é a UF. Contudo, note que elas possuem nomes diferentes. No banco `df_largo` ela é `nome`, enquanto no banco `populacao` ela é `SiglaUF`. Vamos alterar o nome em um deles para torná-los iguais.

```
names(df_largo)[2] <- 'SiglaUF'

df <- merge(df_largo, populacao, by='SiglaUF')
head(df)
```

```
##   SiglaUF cod periodo homens mulheres População total 2010
## 1      AC  12    2010    146       19               733559
## 2      AL  27    2010   1950      137              3120494
## 3      AM  13    2010   1017       65              3483985
## 4      AP  16    2010    244       16               669526
## 5      BA  29    2010   5398      438             14016906
## 6      CE  23    2010   2515      173              8452381
```

O que o *merge* faz é linkar os dados baseados na variável de ID assinalada. O primeiro passo é passar os dois bancos de dados que serão juntados. Note que a ordem pode fazer diferença, então saiba que o primeiro será `x` e o segundo `y`. Use o argumento `by` para indicar a chave de identificação. Tome cuidado pois o R diferencia maiúsculas de minúsculas, portanto, "PE" é diferente de "Pe". Isso isso ocorrer, o R irá perder essa informação.

Você não precisaria alterar o nome das variáveis no banco de dados, a função merge possui alternativas para isso usando `by.x` e `by.y`.

```
df <- merge(df_largo, populacao,
            by.x='nome',
            by.y='SiglaUF')
head(df)
```

Neste caso, atente-se para a posição do banco de dados. Nesse exemplo, `df_largo` está na posição do arumnento `x`. Logo, `by.x` buscará a variável indicada para esse banco. É comum trocarmos essas posições quando estamos aprendendo, então fique atento.

Vamos aproveitar e remover os espaços do nome da variável `População total 2010` para facilitar sua utilização. Para isso, podemos usar uma função de substituição de strings, `gsub`. Embora não pareça necessário aqui, quando usarmos bancos reais haveram muito mais variáveis o que aumenta o trabalho de renomeá-las. Com o `gsub` podemos remover os espaços (ou qualquer outro caractere) de todas ao mesmo tempo.

```
names(df) <- gsub(' ', '', names(df))
names(df)
```

```
## [1] "SiglaUF"            "cod"                "periodo"
## [4] "homens"             "mulheres"           "PopulaçãototalNA2010"
```

Exportando os dados

Finalmente, se tudo está pronto podemos exportar os dados para nosso computador. Existem múltiplas opções. Por agora, basta fazer o simples.

```
write.csv(df, 'meudf.csv')
```

Agregação de dados

É muito comum criarmos agregações dos nossos dados, sendo recorrente a soma, média, desvio padrão e outros. No R base podemos fazer isso usando a função `aggregate`, mas existem centenas de outras disponíveis em outros pacotes [2]

Vamos trabalhar um pouco. (1) Calcule a média de homicídios para homens e para mulheres e (2) calcule o total de homicidios no Brasil.

Esse tipo de tarefa fica muito mais fácil quando o banco está no formato longo. Imagine se você tivesse ao invés do sexo, os anos desde 1990, ou mesmo a escolaridade, ou blocos econômicos? O numéro de categorias seria mais elevado.

Para essa atividade vamos usar o banco df, mas no formato longo. Primeiro, irei alterar o nome da variável do contingente populacional para facilitar a manipulação e em seguida usar o `melt` para transpor esses dados.

```
names(df)[6] <- 'populacao'
df <-
melt(df, id.vars = c('cod', 'SiglaUF', 'periodo', 'populacao'),
                 variable.name = 'sexo', value.name = 'homicidios')

head(df)
```

```
##   cod SiglaUF periodo populacao   sexo homicidios
## 1  12      AC    2010    733559 homens        146
## 2  27      AL    2010   3120494 homens       1950
## 3  13      AM    2010   3483985 homens       1017
## 4  16      AP    2010    669526 homens        244
## 5  29      BA    2010  14016906 homens       5398
## 6  23      CE    2010   8452381 homens       2515
```

VOcê já viu isso antes, o diferente é que indicamos população também como um ID. Se você prestar bem atenção, vai notar que o `melt` vai repetir o mesmo valor da população para um estado espefíco para cada um dos grupos da variável sexo.

Vamos calcular a média para homens e mulheres no Brasil.

```
aggregate(df$homicidios,by = list(df$sexo), FUN = 'mean')
```

```
##    Group.1         x
## 1   homens 1796.0370
## 2 mulheres  165.8148
```

A soma

```
aggregate(df$homicidios,by = list(df$sexo), FUN = 'sum')
```

```
##    Group.1     x
## 1   homens 48493
## 2 mulheres  4477
```

O desvio padrão

[2] Aqui destacam-se as funções do pacote `data.table` e `dplyr` que ainda veremos mais adiante.

```
aggregate(df$homicidios,by = list(df$sexo), FUN = 'sd')
```

```
##    Group.1         x
## 1   homens 1629.2114
## 2 mulheres  158.8514
```

É possível agregar múltiplas colunas numéricas. Uma forma comum de fazer isso é selecionando as colunas usando a notação de colchetes.

```
aggregate(df[, c('homicidios','populacao')],by = list(df$sexo), FUN = 'mean')
```

```
##    Group.1 homicidios populacao
## 1   homens  1796.0370   7065030
## 2 mulheres   165.8148   7065030
```

Finalmente, é possível usar a notação de fórmulas com o `aggregate`.

```
aggregate(homicidios ~ sexo, data =df, mean)
```

```
##       sexo homicidios
## 1   homens  1796.0370
## 2 mulheres   165.8148
```

Ao usar fórmulas é necessário passa-la como primeiro argumento e indicar o banco de dados. Como uma fórmula você pode adicionar mais variáveis de grupo usando o sinal de adição (`+`) do lado direito da fórmula . Para as variáveis numéricas é necessário usar o `cbind` para combiná-las.

```
aggregate(cbind(homicidios, populacao) ~ sexo, data =df, mean)
```

```
##       sexo homicidios populacao
## 1   homens  1796.0370   7065030
## 2 mulheres   165.8148   7065030
```

Exercícios

1. Importe o banco de dados `regiao.csv` e faça o merge dele com o banco de dados `df` que acabamos de criar.

   1.1. (desafio) Formate os códigos da variável `Regiao` para que cada um represente uma sigla. A ordem vai de 1 a 5, sendo as regiões N, NE, CO, SE e S respectivamente.

   1.2. Calcule a média de homicídios por sexo e região. Onde morrem mais mulheres em média?

   1.3. Calcule a soma de homicídios por sexo e região. Onde o absoluto de mulheres é maior?

2. Crie um bando de dados de homicídios e analfabetismo por estados do Brasil. Para criar o banco de dados `dados_homicidios`, você deve usar dois bancos de dados disponíveis, `analfabetos_15_anosOuMais_cadunico.csv` e `taxa_homicidios.csv`. O resultado final (`dados_homicidios`) deveria ser banco de dados semelhante ao apresentado abaixo.

```
head(dados_homicidios)
```

```
##   sigla_uf  ano      uf taxa_homicidio analfabetos15ouMais Regiao
## 1       AC 2014    Acre          29.62               16.14      N
## 2       AC 2015    Acre          27.50               15.84      N
## 3       AC 2016    Acre          44.94               15.37      N
## 4       AC 2017    Acre          62.68               15.62      N
## 5       AL 2014 Alagoas          62.86               18.59     NE
## 6       AL 2015 Alagoas          52.29               18.33     NE
```

```
tail(dados_homicidios)
```

```
##     sigla_uf  ano           uf taxa_homicidio analfabetos15ouMais Regiao
## 103       SP 2016 S\xc6o Paulo          10.81                6.20     SE
## 104       SP 2017 S\xc6o Paulo          10.34                6.96     SE
## 105       TO 2014    Tocantins          24.99               10.48      N
## 106       TO 2015    Tocantins          32.41               10.15      N
## 107       TO 2016    Tocantins          37.25                9.91      N
## 108       TO 2017    Tocantins          35.80               10.71      N
```

3. Use o banco de dados `dados_homicidios` e calcule (1) a média e (2) o desvio padrão da taxa de homícidios por ano e região.

   3.1. Qual a região com maior média?

   3.2. Qual a região com o maior desvio padrão?