Aula 2: A Linguagem R

João Paulo Lazzarini Cyrino

Agosto de 2020

A Linguagem R

R é uma linguagem de programação desenvolvida por estatísticos para estatística. Foi desenvolvida no ano 2000 por Ross Ihaka e por Robert Gentleman, professores do Departamento de Estatística da Universidade de Auckland (Nova Zelândia).

É uma linguagem bastante poderosa para finalidades científicas e acabou por ser uma das mais utilizadas nos departamentos de várias áreas acadêmicas. Além disso, ela é bastante utilizada na indústria para tarefas relacionadas a estatística, ciência de dados e inteligência artificial.

Há atulamente um debate sobre se devemos utilizar uma outra linguagem, denominada Python. A linguagem Python é mais genérica, podendo ter mais aplicações que R. No entanto, para finalidades acadêmicas, R parece trazer mais facilidades:

- R possui um ambiente de trabalho denominado RStudio, que inclui uma série de facilidades.
- A ferramenta RMarkdown permite criar documentos (como este), enquanto estamos programando. Ou seja, podemos escrever o artigo enquanto conduzimos o estudo.
- R é muito simples para conduzir testes estatísticos, com muitas funcionalidades vindas de fábrica.
- R possui recursos para visualização (gráficos) muito superiores e mais intuitivos.

Se você estiver realmente interessado em linguística computacional, principalmente fora do escopo puramente acadêmico, recomendamos aprender Python também. Para as finalidades exclusivamente científicas, R será uma grande parceira e te ajudará imensamente com suas descobertas e teses/dissertações/artigos, etc.

Instalando R e RStudio

Atualmente é impossível dissociar R de seu ambiente de trabalho, a plataforma RStudio. Por essa razão, precisamos instalar os dois programas: a linguagem R e a plataforma RStudio.

Para baixar a linguagem R você pode clicar em um dos links abaixo, de acordo com seu sistema:

- Windows: https://cran.r-project.org/bin/windows/base/
- Mac OSX: https://cran.r-project.org/bin/macosx/

Assim que baixar, você deverá instalá-la antes de partir para baixar e instalar o RStudio. Uma vez instalada a linguagem R, você pode ir ao site https://rstudio.com/products/rstudio/download/#download e baixar a plataforma RStudio. Novamente, assim que baixar, instale a plataforma no seu computador.

A instalação tando da linguagem R como do RStudio devem ser fáceis e correrão sem maiores dificuldades. Em caso de problemas, você pode simplesmente digitar o problema no Google e certamente haverá uma solução para ele! Uma das grandes vantagens da linguagem R é sua **comunidade**: tanto os usuários como os mantenedores da linguagem são pessoas muito interessadas em ajudar, o que faz com que haja inúmeros recursos sobre a linguagem online.

Importante! Como aprender a programar?

Talvez seja sua primeira vez tentando aprender uma linguagem de programação. R é um tanto peculiar no sentido de ser mais objetiva e te levar a perder menos tempo com a criação de um programa e te permitir dedicar somente aos cálculos e análises do seu estudo. Mas, apesar dessa facilidade, isso não significa que será necessariamente fácil aprender a utilizar R e muito menos ter uma familiaridade total com a linguagem a ponto de se sentir plenamente confortável utilizando-a.

Programar é uma habilidade que se conquista com **muita** prática. Não adianta muito fazer cursos e apenas acompanhar o raciocínio do professor. É necessário tentar você mesmo e é necessário também querer explorar sozinho formas diferentes de resolver o problema.

Recomenda-se fortemente que, além de replicar mais de uma vez os códigos desse curso e estudá-los linha por linha, e além de fazer as atividades propostas, você também tente implementar esses conhecimentos em sua pesquisa atual. A forma como R funciona permite que você consiga integrar, com facilidade, o seu processo de pesquisa com o de programação. Por isso, gastar tempo praticando R a partir dos dados de sua pesquisa irá ser benéfico tanto para seu aprendizado de R como para sua pesquisa em si.

Outra coisa importante: acostume-se com não saber muita coisa e ter que pesquisar no google ou em documentos de referência como fazê-la. Nunca dominaremos totalmente os recursos todos da linguagem R, mas é importante aprender a aprender sobre eles. E também: **não se preocupe em memorizar todos os comandos, funções, etc. Isso pode ser bastante frustrante! Deixe que a prática se encarregue disso**.

Programando em R (o básico)

Aqui vamos cobrir alguns básicos de programação em R. Essa seção não é, nem de perto, um bom curso básico de R, mas é o suficiente para que você consiga acompanhar o que será feito nas próximas aulas. Ao final dela aponto alguns sites com mais cursos e recursos para você estender seu conhecimento.

Aprender uma linguagem de programação consiste, basicamente, em aprender:

1. Tipos e estruturas de dados: o que podemos calcular/computar
2. Operadores e Funções: os comandos que nos permitem realizar as operações com os dados
3. Estruturas de Controle: como controlar o fluxo do algoritmo

Aqui, no entanto, aprenderemos apenas os itens 1 e 2. Estruturas de controle em R são um tanto quanto desnecessárias a nível básico. No entanto, apresentaremos alguns elementos desse asssunto ao longo desse curso, conforme necessário. Mais importante que estruturas de controle é como visualizar gráficos, e isso está incluido brevemente aqui.

Tipos de Dados e Operações Básicas

Em R estamos, normalmente, manipulando ou números ou categorias. Categorias costumam ser descritas por texto, como por exemplo "Categoria A", ou "Singular", ou mesmo algo mais codificado como "sg.pl". Por essa razão, em R manipulamos dados de dois tipos, basicamente: `numeric` e `character`.

A função `class` nos permite descobrir qual o tipo de um dado. Veja só o que acontece quando digitamos os seguintes comandos no console do R: (aviso - tudo aquilo que aparece depois de # é um comentário e não precisa ser digitado no console. Você pode colocar comentários no seu código à vontade para tornar as coisas mais fáceis de entender, a linguagem simplesmente os ignora na hora de executar seu programa)

```
class(12) # descobrir em que tipo de dado o número 12 se enquadra
```

```
## [1] "numeric"
```

```
class("Abacate") # descobrir em que tipo de dado o texto "Abacate" se enquadra
```

```
## [1] "character"
```

Nota-se que dados do tipo `character` sempre são declarados utilizando aspas (podem ser duplas ou simples). Veja só o que acontece quando pedimos `class("12")` ao invés de `class(12)`:

```
class("12")
```

```
## [1] "character"
```

Podemos armazenar nossos dados em variáveis, o que torna nosso trabalho muito prático. Fazemos isso utilizando os operadores `<-` e `->`:

```
x <- 12 # x agora vale 12
12 -> x # um outro jeito de escrever que x agora vale 12
```

Nomes de variáveis se diferenciam de caracteres por não terem aspas. Para não confundir com números, elas também não podem começar por números. Sugerimos utilizar nomes de variáveis que sejam fáceis de entender. Quando digitamos o nome da variável no console, ele nos retornará o seu valor:

```
x
```

```
## [1] 12
```

É interessante que no RStudio há uma janela (Environment) que mostra todas as variáveis criadas ao longo do trabalho e os seus respectivos valores. Isso ajuda a não se perder no fluxo de trabalho.

Podemos fazer operações com dados numéricos como fazemos em matemática. Temos os operadores `+`, para soma, `-`, para subtração, `*` para multiplicação e `/` para divisão. Além disso podemos elevar um número n a uma potência p com `n**p` ou `n^p`. Também podemos tomar a raiz quadrada de um número n com a função `sqrt(n)`. Alguns exemplos abaixo:

```
soma <- 23 + 7 # armazenando o resultado de 23 + 7 na variável soma.
# Vamos fazer isso com outras operações também
subtr <- 30 - 23
mult <- 2 * 56
div <- 30/8.4 # repare que o . é o separador decimal
quadrado <- 8^2
raiz <- sqrt(16)
```

Podemos criar um vetor de números que nos retornará os resultados das operações acima. Fazemos isso assim:

```
c(soma, subtr, mult, div, quadrado, raiz)
```

```
## [1]  30.000000   7.000000 112.000000   3.571429  64.000000   4.000000
```

Vetores são estruturas de dados de uma dimensão. Eles aceitam dados de apenas um tipo (ou dados numéricos, ou de caracteres). Podemos armazená-los em variáveis da mesma forma que os demais dados. Vejamos algumas operações interessantes com vetores:

```
# armazenando o vetor com as operações matemáticas na variável v
v <- c(soma,subtr,mult,div,quadrado,raiz)
v # retornará o vetor
```

```
## [1]  30.000000   7.000000 112.000000   3.571429  64.000000   4.000000
```

```
v[1] # retorna o primeiro elemento do vetor
```

```
## [1] 30
```

```
v[3] <- 12 # insere o número 12 no terceiro elemento do vetor
```

```
v[1:4] # retorna os elementos de 1 a 4 no vetor
```

```
## [1] 30.000000  7.000000 12.000000  3.571429
```

```r
length(v) # retorna o tamanho do vetor
```

```
## [1] 6
```

```r
v[7] <- 7 # adiciona um novo elemento 7 no vetor
length(v) # novo tamanho do vetor
```

```
## [1] 7
```

```r
v[length(v)+1] <- 8 # adiciona o número 8 ao final do vetor.
# Note como nos referimos à nova posição do vetor utilizando a função length() + 1.
# Encadear funções é muito comum em programação.
v
```

```
## [1] 30.000000  7.000000 12.000000  3.571429 64.000000  4.000000  7.000000
## [8]  8.000000
```

```r
# nomes para cada posição do vetor
names(v) <- c('soma', 'sub', '12', 'div', 'quadrado', 'raiz', '7', '8')
v # veja como aparecem os nomes
```

```
##      soma       sub        12       div  quadrado      raiz         7         8
## 30.000000  7.000000 12.000000  3.571429 64.000000  4.000000  7.000000  8.000000
```

```r
v['quadrado'] # pedindo o elemento do vetor pelo nome
```

```
## quadrado
##       64
```

Manipular vetores é algo frequente quando estamos lidando com dados. Por enquanto, mais do que decorar as operações, se preocupe em entender o que está acontecendo. É interessante que podemos aplicar operações a todos os elementos dos vetores:

```r
par <- c(2,4,6,8,10)  # vetor de números pares
par^2 # mostra os membros do vetor par elevados ao quadrado
```

```
## [1]   4  16  36  64 100
```

```r
par + 1 # adiciona 1 aos membros do vetor par
```

```
## [1]  3  5  7  9 11
```

É importante entender que as operações sobre vetores geram um novo vetor e não alteram o vetor antigo. Se quisermos alterar o vetor já existente, devemos sobrescrevê-lo em sua respectiva variável:

```r
par <- par * 2 # vetor par agora ficou armazenado com seus valores multiplicados por 2
par # exibir o novo conteúdo de par
```

```
## [1]  4  8 12 16 20
```

Podemos construir uma tabelinha com dois vetores de comprimento igual através da função `cbind`:

```r
par <- c(2,4,6,8,10,12,14) # vetor par
impar <- par - 1 # vetor impar subtraindo 1 de cada valor de par
tabela <- cbind(impar,par) # tabela com duas colunas
tabela
```

```
##      impar par
## [1,]     1   2
## [2,]     3   4
## [3,]     5   6
## [4,]     7   8
```

```
## [5,]    9  10
## [6,]   11  12
## [7,]   13  14
```

Essa tabela é também denominada `matrix` (matriz) em R. Note que cada coluna tem um nome emprestado da variável que formou essa coluna. Podemos alterar os nomes das colunas com a função `colnames`:

```
colnames(tabela) <- c('x', 'x+1') # novos nomes para as colunas
tabela
```

```
##       x x+1
## [1,]  1   2
## [2,]  3   4
## [3,]  5   6
## [4,]  7   8
## [5,]  9  10
## [6,] 11  12
## [7,] 13  14
```

Podemos obter valores da tabela com a notação `[linha,coluna]`. Veja abaixo:

```
tabela[3,2] # obter o valor da terceira linha, segunda coluna.
```

```
## x+1
##   6
```

```
tabela[4,'x'] # obter o valor da quarta linha, coluna entitulada 'x'
```

```
## x
## 7
```

```
tabela[,'x+1'] # obter todos os valores da coluna 'x+1'
```

```
## [1]  2  4  6  8 10 12 14
```

```
tabela[2,] # obter todos os valores da linha 2
```

```
##   x x+1
##   3   4
```

```
nrow(tabela) # número de linhas da tabela
```

```
## [1] 7
```

```
ncol(tabela) # número de colunas da tabela
```

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

Um terceiro tipo de estrutura de dados, que reune as propriedades da matriz com algumas outras funcionalidades é o quadro de dados, ou `data.frame`. Trata-se, simplesmente, de uma tabela em que cada coluna pode ser de um tipo diferente de dado. Podemos converter nossa tabela em um `data.frame` da seguinte forma:

```
tabela <- data.frame(tabela) # converte tabela em data.frame e a armazena novamente em tabela
tabela
```

```
##    x x.1
## 1  1   2
## 2  3   4
## 3  5   6
## 4  7   8
## 5  9  10
## 6 11  12
```

```
## 7 13  14
```

O quadro de dados é o que normalmente manipulamos em R. Podemos utilizar as mesmas funções que utilizamos na matriz para obter dados dele. Também podemos chamar cada coluna com a sintaxe `nome_da_tabela$nome_da_coluna`:

```
tabela$x # pedindo a coluna x
```

```
## [1]  1  3  5  7  9 11 13
```

```
# perceba que R mudou o nome da coluna x+1 da matriz para x.1 para evitar conflitos
tabela$x.1
```

```
## [1]  2  4  6  8 10 12 14
```

Se quisermos criar uma nova coluna, podemos:

```
tabela$x.2 <- tabela$x + 2 # nova coluna x.2 somando 2 aos valores da coluna x
tabela # vamos ver como ficou a tabela agora
```

```
##    x x.1 x.2
## 1  1   2   3
## 2  3   4   5
## 3  5   6   7
## 4  7   8   9
## 5  9  10  11
## 6 11  12  13
## 7 13  14  15
```

Parece bastante coisa, né? Mas acredite: manipular tabelas, matrizes e vetores em R é muito mais prático que em muitas linguagens de programação! Fica até mesmo mais fácil do que em Excel, depois que pegamos a prática. Na próxima aula aprenderemos a utilizar o pacote *tidyverse* para manipular dados, o que torna as coisas ainda mais fáceis.

Por enquanto apenas vimos operações para manipular dados numéricos. Há operações para manipular texto também, mas essas serão introduzidas ao longo do curso, conforme necessário.

Há um terceiro tipo importante de dado (há outros, mas eles derivam de alguma forma desses três), que é o tipo lógico. Esse tipo só permite os valores `TRUE` ou `FALSE`, correspondendo a verdadeiro e falso. Com isso podemos fazer operações lógicas em R, o que é de grande utilidade.

Uma dessas operações são as operações de igualdade, como `==` (igual a) e `!=` (diferente). Com elas podemos comparar valores.

```
1 != 2 # verifica se 1 é diferente de 2
```

```
## [1] TRUE
```

```
1 == 2 # verifica se 1 é igual a 2
```

```
## [1] FALSE
```

Podemos aplicar operações lógicas para valores de tabelas. Elas nos retornará um vetor ou matriz (conforme o caso) de valores lógicos:

```
tabela == 5 # uma matriz com TRUE onde havia 5 na tabela
```

```
##          x   x.1   x.2
## [1,] FALSE FALSE FALSE
## [2,] FALSE FALSE  TRUE
## [3,]  TRUE FALSE FALSE
## [4,] FALSE FALSE FALSE
```

```
## [5,] FALSE FALSE FALSE
## [6,] FALSE FALSE FALSE
## [7,] FALSE FALSE FALSE
```

curiosamente, matematicamente os valores de TRUE podem ser somados como se fossem 1 e os de FALSE como se fossem 0. Dessa forma, podemos somar os valores verdadeiros da matriz obtendo a contagem de números 5 na tabela. Isso pode ser feito com a função `sum`, que soma os valores de uma matriz ou vetor:

```
sum(tabela == 5) # conta quantas vezes 5 aparece na tabela
```

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

Da mesma forma, podemos utilizar a função de média `mean` para calcular a proporção de 5 na tabela. Isso será amplamente utilizado nas nossas análises.

```
mean(tabela == 5) # proporção de números 5 na tabela
```

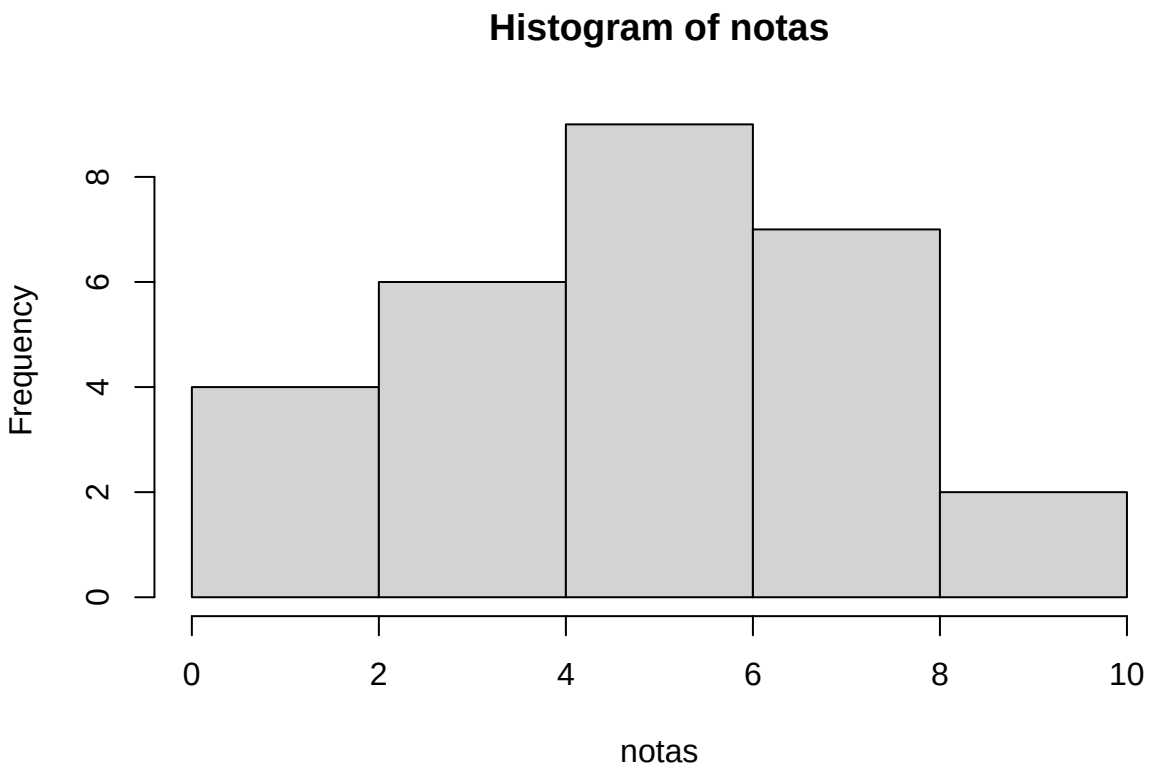
```
## [1] 0.0952381
```

Apresentamos, aqui algumas operações básicas em R. Novamente, não se preocupe tanto em decorar, mas em tentar replicar e explorá-las (tentar fazer cálculos com outros valores, matrizes diferentes, etc). Essa prática é o que traz segurança no uso da ferramenta.

Gráficos Básicos

Na próxima aula veremos uma forma mais eficiente e flexível de como plotar gráficos em R. Aqui mostramos algumas funções gráficas fundamentais que podem ser úteis para visualização rápida de dados. A primeira função é `hist`, que plota histogramas. Histogramas são gráficos de distribuição de frequências e são importantes para visualizar o que está ocorrendo com uma variável numérica.

```
# notas de uma turma
notas <- c(3,3,4,2,5,5,5,6,6,7,8,9,8,8,8,10,5,5,6,2,2,1,3,3,4,5,7,7)
hist(notas) #histograma das notas
```



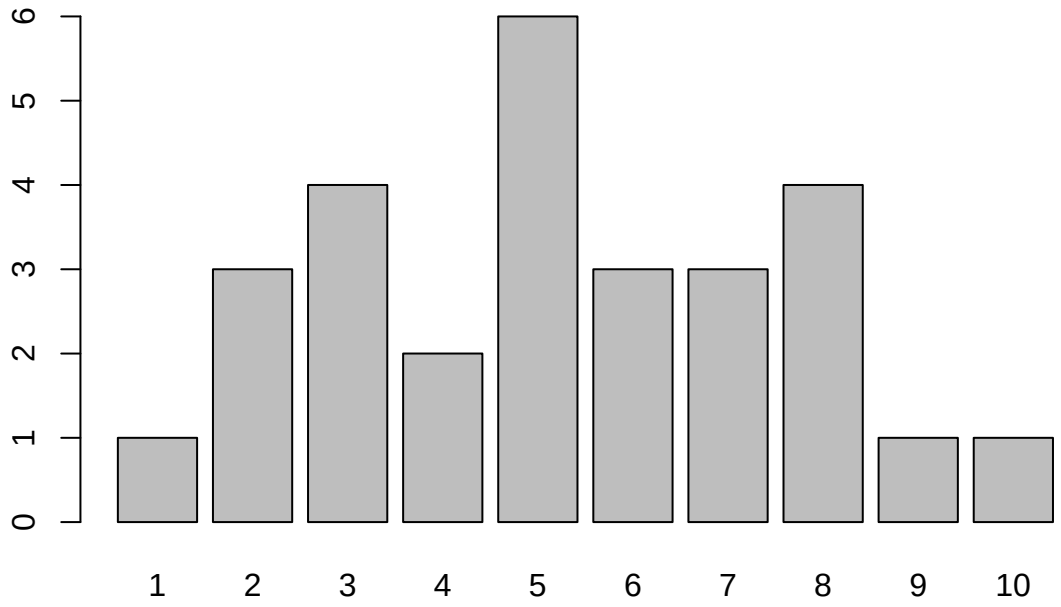
Podemos criar gráficos de barras a partir de tabelas de frequência. Para criar uma tabela de frequências a partir de um vetor como, por exemplo, *notas*, utilizamos a função `table`:

```
table(notas)
```

```
## notas
##  1  2  3  4  5  6  7  8  9 10
##  1  3  4  2  6  3  3  4  1  1
```

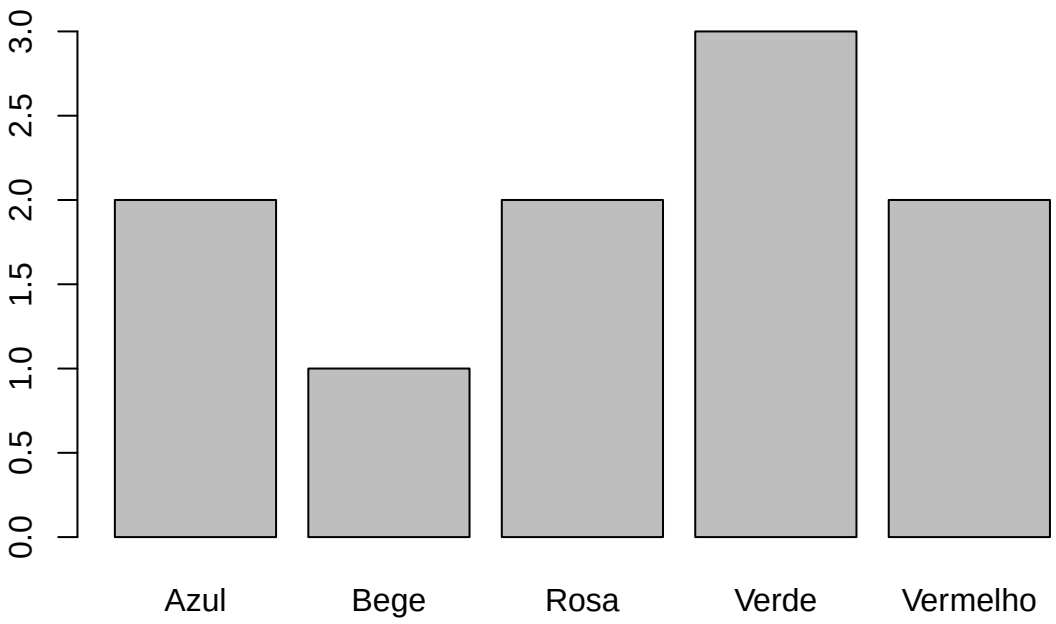
O gráfico de barras pode ser construido da seguinte forma:

```
barplot(table(notas))
```



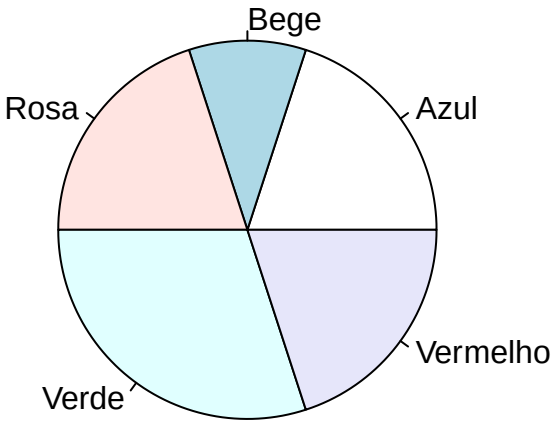
Gráficos de barras são ótimos para visualizar variáveis categóricas:

```
# Cores preferidas dos alunos de uma turma
cores <- c('Azul','Azul','Rosa','Verde','Verde','Vermelho','Verde','Vermelho','Rosa','Bege')
t <- table(cores) # tabela de frequência das cores armazenada em t
barplot(t)
```



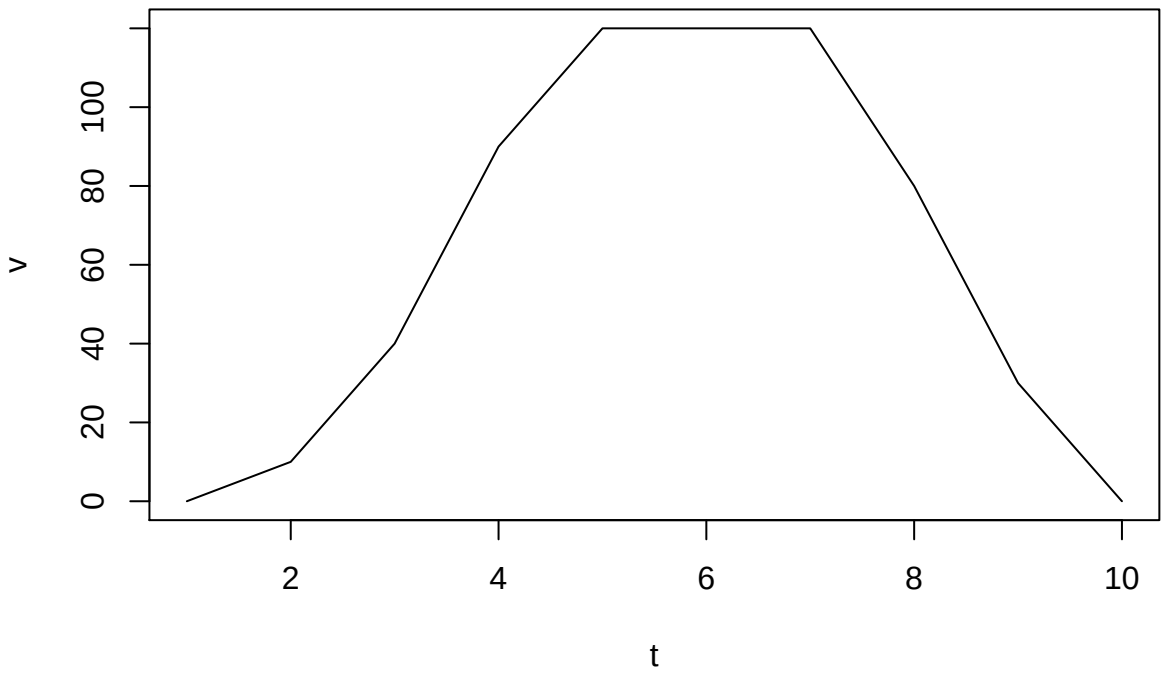
Gráficos de pizza também podem ser plotados:

```
pie(t)
```

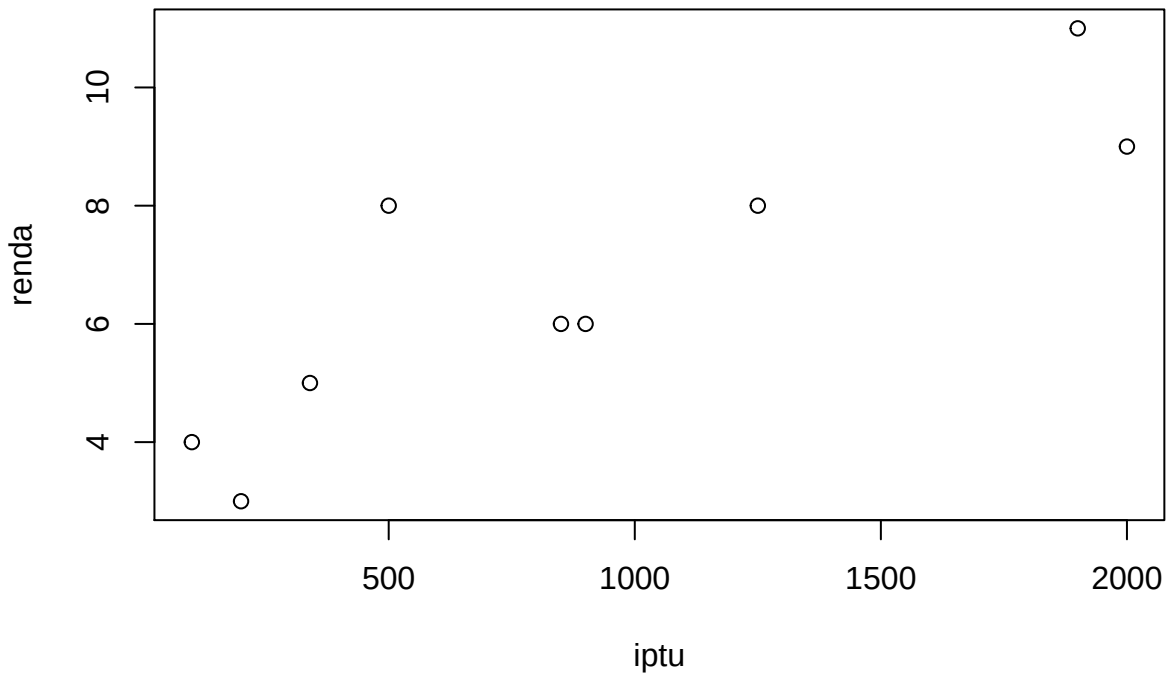


Para mais de uma variável, podemos plotar gráficos de dispersão e de linhas com a função `plot`:

```
v <- c(0,10,40,90,120,120,120,80,30,0) # velocidades de um carro
t <- c(1,2,3,4,5,6,7,8,9,10) # instantes de medição
plot(t,v, type='l') # plotar t no eixo x e v no eixo y, tipo de gráfico linha ('l')
```




```
iptu <- c(1250,900,850,2000,340,200,1900,500,100) # iptu anual do imóvel
renda <- c(8,6,6,9,5,3,11,8,4) # renda familiar do morador em número de salários mínimos
plot(iptu,renda) # gráfico de dispersão iptu x renda
```



Há várias opções que podem ser incluídas em cada um desses comandos de gráficos, seja para adicionar títulos ou cores, seja para sobrepor gráficos. Sugerimos, no entanto, que se utilize a técnica da próxima aula para plotar gráficos e deixe esses comandos mais para uma visualização rápida.

Onde encontrar mais recursos

Isso foi uma introdução bastante básica da linguagem R. Caso deseje se aprofundar mais na linguagem em si, recomendamos alguns sites com tutoriais:

- Curso-R http://material.curso-r.com/
- Estatística em R para linguistas (Profa. Lívia Oushiro Unicamp) https://rpubs.com/oushiro/iel?fbclid=IwAR1R2ZU3WXQCZnvgAQO4NtkmLnCTh229oMM64koBPymHd30ccFgjbUKFyDk
- Tutorial R/RStudio https://edisciplinas.usp.br/pluginfile.php/4883125/mod_resource/content/1/Tutorial.pdf
- R-Tutor (em inglês) <http://www.r-tutor.com/

Importando planilhas e juntando

Importar planilhas do excel é algo que faremos com frequência ao trabalhar com R. O RStudio facilita isso com a funcionalidad Import Dataset, na janela Environment. Basta clicar no botão *Import Dataset* e escolher a opção *From Excel*.

Juntando Planilhas

Na aula passada baixamos duas planilhas do WALS: uma sobre flexão verbal (exercício para casa) e outra sobre tipo de marcação na oração. Se você fez tudo certo deve ter planilhas como as que estão nos arquivos *clause_marking.xls* e *verb_inflection.xls* na pasta dessa aula.

Como os dados do WALS versam sobre uma única variável, normalmente, juntar planilhas para verificar correlaçoes entre essas variáveis é uma habilidade bastante importante. Podemos fazer isso com o R contanto que haja uma variável que identifique cada linha e que seja comum às duas planilhas.

No caso dos dados do WALS, essa variável corresponde à língua, já que cada observação é referente a uma língua. Dessa forma, queremos que as observações da língua x de uma planilha se alinhem às observações dessa mesma língua x de uma segunda planilha.

Vamos carregar *clause_marking.xls* e *verb_inflection.xls*. Após carregar, podemos executar a função `str` para verificar a estrutura de cada tabela:

```
str(clause_marking)
```

```
## tibble [236 x 4] (S3: tbl_df/tbl/data.frame)
##  $ id           : chr [1:236] "abk" "ace" "aco" "ain" ...
##  $ locus_marking: chr [1:236] "Head marking" "Head marking" "Head marking" "No marking" ...
##  $ language     : chr [1:236] "Abkhaz" "Acehnese" "Acoma" "Ainu" ...
##  $ ref          : chr [1:236] "Hewitt 1979" "Durie 1985" "Miller 1965" "Tamura 2000;Shibatani 1990;Pa
```

```
str(verb_inflection)
```

```
## tibble [145 x 2] (S3: tbl_df/tbl/data.frame)
##  $ id                     : chr [1:145] "abk" "aco" "ash" "ain" ...
##  $ inflectional_categories: chr [1:145] "10-11 categories per word" "4-5 categories per word" "8-9 ca
```

Ambas as tabelas possuem uma variável denominada *id*, com cada valor constituído por três letras. Trata-se do código de cada língua. Podemos utilizar essa variável *id* para juntar as duas tabelas.

Podemos ver também que as tabelas tem um número de línguas diferente: uma com 236 e outra com 145. Podemos optar por juntar somente as observações que possuem um *id* comum às duas tabelas:

```
cm.vi <- merge(clause_marking,verb_inflection,by='id') # Juntando as duas tabelas nos ids comuns
str(cm.vi) # 136 observações
```

```
## 'data.frame':    136 obs. of  5 variables:
##  $ id                     : chr  "abk" "aco" "aeg" "ain" ...
##  $ locus_marking          : chr  "Head marking" "Head marking" "No marking" "No marking" ...
##  $ language               : chr  "Abkhaz" "Acoma" "Arabic (Egyptian)" "Ainu" ...
##  $ ref                    : chr  "Hewitt 1979" "Miller 1965" "Gary and Gamal-Eldin 1982" "Tamura 200
##  $ inflectional_categories: chr  "10-11 categories per word" "4-5 categories per word" "6-7 categori
```

Vemos que a tabela resultante possui 136 observações. Ou seja, há 136 observações de línguas comuns às duas tabelas. Esse tipo de junção é conhecido por *inner join* no jargão de banco de dados.

Podemos juntar todas as entradas das duas tabelas. Dessa forma, aquelas línguas sem observação em um ou outro estudo ocorrerão com o valor NA (não disponível):

```
cm.vi.tudo <- merge(clause_marking,verb_inflection,by='id',all=TRUE) #  opção all=TRUE
str(cm.vi.tudo) # Estrutura da nova tabela
```

```
## 'data.frame':    245 obs. of  5 variables:
##  $ id                     : chr  "-ik" "abk" "ace" "aco" ...
##  $ locus_marking          : chr  "Dependent marking" "Head marking" "Head marking" "Head marking" ..
##  $ language               : chr  "Ik" "Abkhaz" "Acehnese" "Acoma" ...
##  $ ref                    : chr  "Tucker 1973;Tucker 1972;Tucker 1971" "Hewitt 1979" "Durie 1985" "Mi
##  $ inflectional_categories: chr  NA "10-11 categories per word" NA "4-5 categories per word" ...
```

```
# Visualizar a tabela (colunas 1,2,5). Veja que haverá observações com valores <NA>.
cm.vi.tudo[,c(1,2,5)]
```

```
##      id     locus_marking   inflectional_categories
```

```
## 1   -ik Dependent marking                     <NA>
## 2   abk      Head marking 10-11 categories per word
## 3   ace      Head marking                     <NA>
## 4   aco      Head marking   4-5 categories per word
## 5   aeg        No marking   6-7 categories per word
## 6   ain        No marking   4-5 categories per word
## 7   ala      Head marking 10-11 categories per word
## 8   amb Dependent marking                     <NA>
## 9   ame      Head marking   6-7 categories per word
## 10  amp Dependent marking   4-5 categories per word
## 11  ane      Head marking                     <NA>
## 12  apu      Head marking   6-7 categories per word
## 13  arm             <NA>   2-3 categories per word
## 14  arp      Head marking   4-5 categories per word
## 15  ash             <NA>   8-9 categories per word
## 16  asm      Head marking   4-5 categories per word
## 17  atk      Head marking   8-9 categories per word
## 18  awp    Double marking   8-9 categories per word
## 19  awt Dependent marking                     <NA>
## 20  aym    Double marking   8-9 categories per word
## 21  bag      Head marking   6-7 categories per word
## 22  bbw      Head marking                     <NA>
## 23  bej Dependent marking   4-5 categories per word
## 24  bel    Double marking   6-7 categories per word
## 25  bma Dependent marking   6-7 categories per word
## 26  brh Dependent marking   2-3 categories per word
## 27  bri        No marking                     <NA>
## 28  brm Dependent marking   2-3 categories per word
## 29  brr      Head marking   4-5 categories per word
## 30  brs    Double marking   4-5 categories per word
## 31  bsq    Double marking   4-5 categories per word
## 32  bur    Double marking   8-9 categories per word
## 33  buu        No marking                     <NA>
## 34  bzi        No marking                     <NA>
## 35  ccp    Double marking                     <NA>
## 36  cha Dependent marking   6-7 categories per word
## 37  chk    Double marking   4-5 categories per word
## 38  cho      Head marking                     <NA>
## 39  chy    Double marking                     <NA>
## 40  cin      Head marking                     <NA>
## 41  cjo      Head marking                     <NA>
## 42  cku      Head marking 10-11 categories per word
## 43  cnl      Head marking   4-5 categories per word
## 44  coo      Head marking                     <NA>
## 45  cre      Head marking   6-7 categories per word
## 46  ctm      Head marking                     <NA>
## 47  cyv        No marking   6-7 categories per word
## 48  dag      Head marking   8-9 categories per word
## 49  dig             <NA>   4-5 categories per word
## 50  diy Dependent marking                     <NA>
## 51  diz Dependent marking                     <NA>
## 52  dji    Double marking                     <NA>
## 53  djp Dependent marking                     <NA>
## 54  dni      Head marking   6-7 categories per word
```

```
## 55  dum        No marking                     <NA>
## 56  dyi Dependent marking                     <NA>
## 57  eka      Head marking   4-5 categories per word
## 58  eng Dependent marking   2-3 categories per word
## 59  epe Dependent marking   4-5 categories per word
## 60  eve Dependent marking   6-7 categories per word
## 61  ewe        No marking                     <NA>
## 62  fij      Head marking   6-7 categories per word
## 63  fin Dependent marking   2-3 categories per word
## 64  fre        No marking   4-5 categories per word
## 65  fur Dependent marking                     <NA>
## 66  gar             <NA>   2-3 categories per word
## 67  geo    Double marking   8-9 categories per word
## 68  ger Dependent marking   2-3 categories per word
## 69  gku        No marking                     <NA>
## 70  gmz Dependent marking                     <NA>
## 71  goo    Double marking   6-7 categories per word
## 72  grb Dependent marking   6-7 categories per word
## 73  grk    Double marking   4-5 categories per word
## 74  grr Dependent marking                     <NA>
## 75  grw    Double marking   4-5 categories per word
## 76  gua      Head marking   4-5 categories per word
## 77  hai        No marking   8-9 categories per word
## 78  hat        No marking   2-3 categories per word
## 79  hau    Double marking   6-7 categories per word
## 80  heb Dependent marking   4-5 categories per word
## 81  hin    Double marking   2-3 categories per word
## 82  hix      Head marking   4-5 categories per word
## 83  hmo        No marking   2-3 categories per word
## 84  hua    Double marking                     <NA>
## 85  hun Dependent marking   4-5 categories per word
## 86  hve      Head marking   4-5 categories per word
## 87  hzb             <NA>   6-7 categories per word
## 88  iau        No marking                     <NA>
## 89  ice Dependent marking                     <NA>
## 90  ijo        No marking                     <NA>
## 91  ika        No marking                     <NA>
## 92  imo    Double marking 10-11 categories per word
## 93  ind        No marking   4-5 categories per word
## 94  ing    Double marking 10-11 categories per word
## 95  jak      Head marking   4-5 categories per word
## 96  jel        No marking                     <NA>
## 97  jiv    Double marking                     <NA>
## 98  jpn Dependent marking   4-5 categories per word
## 99  juh        No marking                     <NA>
## 100 kay Dependent marking   4-5 categories per word
## 101 kch Dependent marking   2-3 categories per word
## 102 ket      Head marking   2-3 categories per word
## 103 kew Dependent marking   6-7 categories per word
## 104 kha Dependent marking   2-3 categories per word
## 105 kho Dependent marking   6-7 categories per word
## 106 khs             <NA>   4-5 categories per word
## 107 kim      Head marking                     <NA>
## 108 kio      Head marking   8-9 categories per word
```

```
## 109 kis        No marking   2-3 categories per word
## 110 kiu      Head marking                      <NA>
## 111 kkr Dependent marking                      <NA>
## 112 kmb        No marking                      <NA>
## 113 knd Dependent marking   2-3 categories per word
## 114 knm Dependent marking                      <NA>
## 115 knp    Double marking                      <NA>
## 116 knr    Double marking                      <NA>
## 117 koa    Double marking 12-13 categories per word
## 118 kob        No marking                      <NA>
## 119 kor Dependent marking   6-7 categories per word
## 120 kri Dependent marking                      <NA>
## 121 krk      Head marking   8-9 categories per word
## 122 kro        No marking   4-5 categories per word
## 123 ksg              <NA>   2-3 categories per word
## 124 kut    Double marking   4-5 categories per word
## 125 lai    Double marking   8-9 categories per word
## 126 lan      Head marking   6-7 categories per word
## 127 lav      Head marking   4-5 categories per word
## 128 lez Dependent marking   2-3 categories per word
## 129 lkt      Head marking 10-11 categories per word
## 130 lmh        No marking                      <NA>
## 131 lug        No marking                      <NA>
## 132 luv      Head marking   8-9 categories per word
## 133 maa              <NA>   6-7 categories per word
## 134 mai Dependent marking                      <NA>
## 135 mal Dependent marking   4-5 categories per word
## 136 mao Dependent marking                      <NA>
## 137 map    Double marking   8-9 categories per word
## 138 mar    Double marking   8-9 categories per word
## 139 mau      Head marking   4-5 categories per word
## 140 may        No marking     0-1 category per word
## 141 mei Dependent marking   4-5 categories per word
## 142 mis        No marking                      <NA>
## 143 mlk             Other                      <NA>
## 144 mnd Dependent marking     0-1 category per word
## 145 mot    Double marking                      <NA>
## 146 mpa    Double marking                      <NA>
## 147 mrh      Head marking                      <NA>
## 148 mrt Dependent marking   2-3 categories per word
## 149 mss    Double marking   4-5 categories per word
## 150 mun    Double marking   6-7 categories per word
## 151 mxc        No marking   4-5 categories per word
## 152 myi    Double marking   6-7 categories per word
## 153 nah Dependent marking     0-1 category per word
## 154 nan              <NA>   4-5 categories per word
## 155 nar Dependent marking                      <NA>
## 156 nat      Head marking                      <NA>
## 157 nbd Dependent marking                      <NA>
## 158 nca        No marking   6-7 categories per word
## 159 nep Dependent marking                      <NA>
## 160 ngi Dependent marking   4-5 categories per word
## 161 niv      Head marking   8-9 categories per word
## 162 nkk      Head marking                      <NA>
```

```
## 163 nmb      Head marking   8-9 categories per word
## 164 nsg      Head marking   4-5 categories per word
## 165 ntj Dependent marking                      <NA>
## 166 ntu    Double marking   2-3 categories per word
## 167 nug    Double marking                      <NA>
## 168 nuu        No marking                      <NA>
## 169 nyn    Double marking                      <NA>
## 170 ond      Head marking   8-9 categories per word
## 171 orh        No marking   6-7 categories per word
## 172 ori        No marking                      <NA>
## 173 otm      Head marking   8-9 categories per word
## 174 pai    Double marking   4-5 categories per word
## 175 pau    Double marking                      <NA>
## 176 pip      Head marking   4-5 categories per word
## 177 prh        No marking   8-9 categories per word
## 178 prs    Double marking   4-5 categories per word
## 179 pso Dependent marking   4-5 categories per word
## 180 pur Dependent marking                      <NA>
## 181 qim Dependent marking   8-9 categories per word
## 182 qui    Double marking                      <NA>
## 183 ram    Double marking   2-3 categories per word
## 184 rap    Double marking   8-9 categories per word
## 185 rus Dependent marking   4-5 categories per word
## 186 sah      Head marking                      <NA>
## 187 san        No marking     0-1 category per word
## 188 sdw        No marking                      <NA>
## 189 shk Dependent marking   6-7 categories per word
## 190 siu    Double marking                      <NA>
## 191 sla      Head marking   8-9 categories per word
## 192 sml Dependent marking                      <NA>
## 193 snm      Head marking   4-5 categories per word
## 194 spa    Double marking   4-5 categories per word
## 195 squ    Double marking   8-9 categories per word
## 196 sue        No marking                      <NA>
## 197 sup        No marking   2-3 categories per word
## 198 swa      Head marking   4-5 categories per word
## 199 syu        No marking                      <NA>
## 200 tag    Double marking   2-3 categories per word
## 201 tau    Double marking                      <NA>
## 202 tgp      Head marking                      <NA>
## 203 tha        No marking   2-3 categories per word
## 204 tib Dependent marking   4-5 categories per word
## 205 tiw      Head marking   4-5 categories per word
## 206 tkl      Head marking                      <NA>
## 207 tlf      Head marking                      <NA>
## 208 tli      Head marking                      <NA>
## 209 tmc      Head marking                      <NA>
## 210 ton    Double marking                      <NA>
## 211 tru Dependent marking   8-9 categories per word
## 212 tsh Dependent marking                      <NA>
## 213 tuk        No marking   6-7 categories per word
## 214 tun      Head marking                      <NA>
## 215 tur Dependent marking   6-7 categories per word
## 216 uhi Dependent marking                      <NA>
```

```
## 217 ung      Head marking                     <NA>
## 218 usa      Head marking                     <NA>
## 219 vie        No marking     0-1 category per word
## 220 wao        No marking                     <NA>
## 221 war      Head marking   4-5 categories per word
## 222 was      Head marking                     <NA>
## 223 wch    Double marking   2-3 categories per word
## 224 wem    Double marking                     <NA>
## 225 wic      Head marking 12-13 categories per word
## 226 win    Double marking                     <NA>
## 227 wra Dependent marking   4-5 categories per word
## 228 wrl    Double marking                     <NA>
## 229 wrn      Head marking                     <NA>
## 230 wry    Double marking                     <NA>
## 231 wsk    Double marking                     <NA>
## 232 xok Dependent marking                     <NA>
## 233 yag             Other 10-11 categories per word
## 234 yaq    Double marking   4-5 categories per word
## 235 yes      Head marking                     <NA>
## 236 yim      Head marking                     <NA>
## 237 yko Dependent marking                     <NA>
## 238 yli      Head marking                     <NA>
## 239 yor    Double marking   6-7 categories per word
## 240 ypk    Double marking                     <NA>
## 241 yuc      Head marking                     <NA>
## 242 yur      Head marking   6-7 categories per word
## 243 zqc    Double marking   6-7 categories per word
## 244 zul      Head marking   4-5 categories per word
## 245 zun    Double marking                     <NA>
```

Arquivo RDS

Caso queiramos salvar a tabela nova para usar posteriormente em R podemos salvá-la em um tipo de arquivo de dados específico da linguagem R: o arquivo RDS.

```
saveRDS(cm.vi.tudo, file="cm_vi_tudo.rds")
saveRDS(cm.vi, file = "cm_vi.rds")
```

Os arquivos ficarão disponíveis na sua pasta de trabalho. Para carregá-los no seu projeto, basta clicar neles no navegador do RStudio. Existe o comando `readRDS` também, para carregar o arquivo manualmente.

```
cm.vi.tudo <- readRDS(file="cm_vi_tudo.rds")
cm.vi <- readRDS(file="cm_vi.rds")
```

Concluindo

Apresentamos aqui alguns dos fundamentos da linguagem R para trabalhar ao longo desse curso. Lembrando sempre que: o mais importante é praticar e não decorar as funções. Tome seu tempo para replicar os códigos ensinados nesta aula e tentar explorar, por conta própria, as funções com outros dados.