heRcules: Um repositório de scripts R anotados em Português para a análise de dados científicos

Hercules Rezende Freitas

https://doi.org/10.1590/SciELOPreprints.3389

Submetido em: 2021-12-22
Postado em: 2022-01-03 (versão 1)
(AAAA-MM-DD)

heRcules: um repositório de scripts R anotados em Português para a análise de dados científicos

heRcules: a repository for annotated R scripts in Portuguese for scientific data analysis

Hércules Rezende Freitas, Ph.D.[1]

[1]Laboratório de Erros Inatos do Metabolismo, Instituto de Bioquímica Leopoldo de Meis, Centro de Ciências da Saúde, Universidade Federal do Rio de Janeiro/UFRJ - Ilha do Fundão, Rio de Janeiro/RJ, Brasil.

Resumo

A análise de dados é uma etapa fundamental do desenvolvimento de projetos científicos. Antes mesmo de iniciar o trabalho, o pesquisador precisa planejar seus experimentos e análises de forma clara, garantindo uma abordagem robusta e protegida dos vieses mais elementares. O presente documento reporta a criação do repositório "heRcules," que dará acesso público a modelos de *scripts* em linguagem R para a análise de dados científicos, com ênfase nas disciplinas das Ciências Biológicas e da Saúde. Nesse primeiro modelo, reportado aqui, estão inclusos *scripts* para tarefas essenciais no planejamento, análise, visualização e teste de hipóteses, incluindo cálculo de tamanho amostral, cálculo de poder estatístico, importação de planilhas, criação de vetores e data frames, estatística descritiva, exportação de arquivos, criação de gráficos (base R e ggplot2), testes de *outliers,* testes de normalidade, testes de hipóteses e criação de notebooks com o R markdown. O repositório está depositado na plataforma GitHub (https://github.com/drhrf/heRcules.git)[1], o que garantirá acesso aos recursos de forma eficiente, gratuita e colaborativa.

Palavras-chave: R; Análise de dados; Tamanho amostral; Gráficos; GitHub; Teste de hipóteses.

Abstract

Data analysis is a fundamental step in the development of scientific projects. Before starting a project, the researcher needs to plan their experiments and analyzes clearly, ensuring a robust approach, protected from the most elementary biases. This document reports the creation of the "heRcules" repository, which will provide public access to script models in R language for the analysis of scientific data, with an emphasis on disciplines of the Biological and Health Sciences. The model presented here provides scripts for essential tasks in planning, analysis, visualization, and hypothesis testing procedures, including sample size calculation, statistical power calculation, spreadsheet import, vector and data frame creation, descriptive statistics, file export, plot creation (base R and ggplot2), outlier tests, normality tests, hypothesis tests and notebook creation with R markdown. The "heRcules" repository is deposited on GitHub, which will ensure efficient, free, and collaborative access to these resources.

Keywords: R; Data analysis, Sample size; Plots; GitHub; Hypotheses tests.

Introdução

Pesquisadores modernos possuem acesso a uma miríade de recursos para o planejamento, análise e representação visual de dados científicos. Muitos desses recursos, porém, são funcionalmente limitados e/ou exigem compra de licenças a preços frequentemente proibitivos. Como alternativa, é possível utilizar linguagens de programação como R (https://www.r-project.org/)[1] e Python (https://www.python.org/)[1], que são estruturadas para não somente suportar todas as tarefas executáveis nos recursos pagos, mas fornecer ao usuário uma infinitude de outras ferramentas, inclusive a possibilidade de desenvolver os próprios programas e funções.

Apesar do poder dessas linguagens, a ausência de interfaces amigáveis acaba por afastar um grande número de potenciais usuários, mantendo-os reféns de ferramentas limitadas. Nesse contexto, nota-se a emergência de um movimento voltado para facilitar o acesso de usuários não familiarizados com linguagens de programação. A grande maioria desses novos recursos, porém, é publicado em Inglês, o que acaba por não resolver o problema para a população monolíngue. Recentemente, alguns esforços têm sido feitos para remediar a situação, a exemplo do guia "Introdução ao R (https://vanderleidebastiani.github.io/tutoriais/Introducao_ao_R.html)"[1], elaborado por V. J. Debastiani (DEBASTIANI, 2021).

Com o objetivo de ampliar o acesso gratuito a ferramentas de computação estatística na língua Portuguesa, o presente trabalho reporta a criação do repositório heRcules (https://github.com/drhrf/heRcules.git)[1], que servirá como uma plataforma aberta para acesso a modelos de análise de dados na linguagem R. O primeiro modelo do repositório, apresentado abaixo, contém o código necessário para o planejamento amostral, o cálculo de estatística descritiva, a geração de visualizações e os testes de hipóteses. Os códigos disponíveis são completamente anotados em língua Portuguesa para facilitar a compreensão e reutilização pelo usuário inexperiente.

Recomendações

Ao usuário completamente inexperiente na linguagem R, recomenda-se a leitura do guia "Introdução ao R (https://vanderleidebastiani.github.io/tutoriais/Introducao_ao_R.html)"[1], mencionado anteriormente. Para utilização do ambiente de forma mais amistosa, é desejável que o usuário obtenha não somente o software R (https://www.r-project.org/)[1] (R CORE TEAM, 2013), mas também um ambiente de desenvolvimento integrado (IDE), que permitirá maior flexibilidade no uso da linguagem. O RStudio (https://www.rstudio.com/products/rstudio/download/)[1] é, de longe, a IDE mais utilizada para programação em R. Ambos os recursos são gratuitos e podem ser utilizados em diversas plataformas UNIX, Windows e MacOS.

O modelo abaixo foi construído de forma a representar um conjunto típico de dados produzidos por experimentos científicos nas áreas de Ciências Biológicas e da Saúde, mas pode ser facilmente modificado para atender a outros modelos e demandas. Adicionalmente, os blocos de código disponíveis aqui podem ser utilizados de forma independente para satisfazer alguma demanda específica. Para aplicá-los a um novo conjunto de dados, basta ao usuário formatar os próprios dados experimentais no modelo proposto e substituir o nome das variáveis dentro de cada bloco de código.

Modelo

Pacotes

Os pacotes abaixo foram utilizadas no presente modelo (Bloco 1). Alguns pacotes importados aqui possuem funções similares e não precisam se repetir no caso de uma análise de fato. Se o pacote desejado não estiver disponível no sistema, deve-se instalá-lo com o comando "install.packages("nome_do_pacote")."

Bloco 1. Pacotes utilizados no presente modelo.

```
library(psych)          #REVELLE, 2021
library(dplyr)          #WICKHAM et al., 2021
library(pastecs)        #GROSJEAN & IBANEZ, 2018
library(fBasics)        #WUERTZ et al., 2020
library(skimr)          #WARING et al., 2021
library(ggplot2)        #WICKHAM, 2016
library(xlsx)           #DRAGULESCU & ARENDT, 2020
library(reshape2)       #WICKHAM, 2007
library(rstatix)        #KASSAMBARA, 2021
library(kableExtra)     #ZHU, 2021
library(pwr)            #CHAMPELY, 2020
library(effsize)        #TORCHIANO, 2020
library(stats)          #R CORE TEAM, 2013
```

Dados do modelo

Os dados utilizados no presente modelo são adaptações de resultados obtidos em experimentos reais. A importação de objetos é um processo simples em R, e exige do usuário apenas uma linha de código. O caso mais habitual para pesquisas em Biociências é a importação de dados contidos em planilhas, normalmente salvas no formato .xlsx. Para esse tipo de documento, basta que o pesquisador execute o seguinte código: "**df <- readxl::read_excel('caminho_do_arquivo_no_computador.xlsx')**." Esse comando importará para o ambiente R a primeira aba da planilha e atribuirá ao objeto "df" as informações contidas nela.

A Tabela 1, criada com o pacote "kableExtra," apresenta os dados utilizados no presente modelo. Aqui, foi simulado um experimento com grupos controle, controle positivo e dois tratamentos (I e II).

Tabela 1. Dados utilizados no modelo.

```
db <- df #Atribui "df" a "db"

k <- kbl(caption = "Modelo I",
         x = db) #Cria tabela kbl() e atribui a "k"

kable_classic(full_width = F,
              html_font = "Cambria",
              kable_input = k) #Define o tema da tabela | #Resultado:
```

Modelo I

Controle	Positivo	Tratamento_I	Tratamento_II
95.26185	0.6711409	97.93014	76.47201
92.13382	0.1011122	88.39161	75.64001
97.84615	0.3436426	96.36364	76.49453
98.84755	1.6676875	92.36735	75.33878

Controle	Positivo	Tratamento_I	Tratamento_II
90.50705	0.6223255	88.26834	75.45820
87.31900	0.6634388	97.31882	78.72481
90.77282	1.2961154	87.79540	76.70423
89.36114	1.7559411	95.44376	76.20736
95.00713	0.5677962	87.45515	76.08017
97.33021	0.8238413	91.02690	75.49445
90.82158	1.0506822	92.61274	76.43213
89.81938	0.8963051	91.05311	78.48522
97.75434	1.3193839	91.24195	75.94288
98.18071	1.8930254	92.28904	76.91323

Estimativa de poder estatístico e tamanho amostral

As estimativas de tamanho amostral (Bloco 2) e poder estatístico (Bloco 4) para a análise de variância (ANOVA) podem ser realizadas utilizando o mesmo código, sendo apenas necessário substituir o valor desejado por “NULL.” Também é possível criar um gráfico para representar a relação entre as variável da estimativa amostral (Blocos 3 e 5).

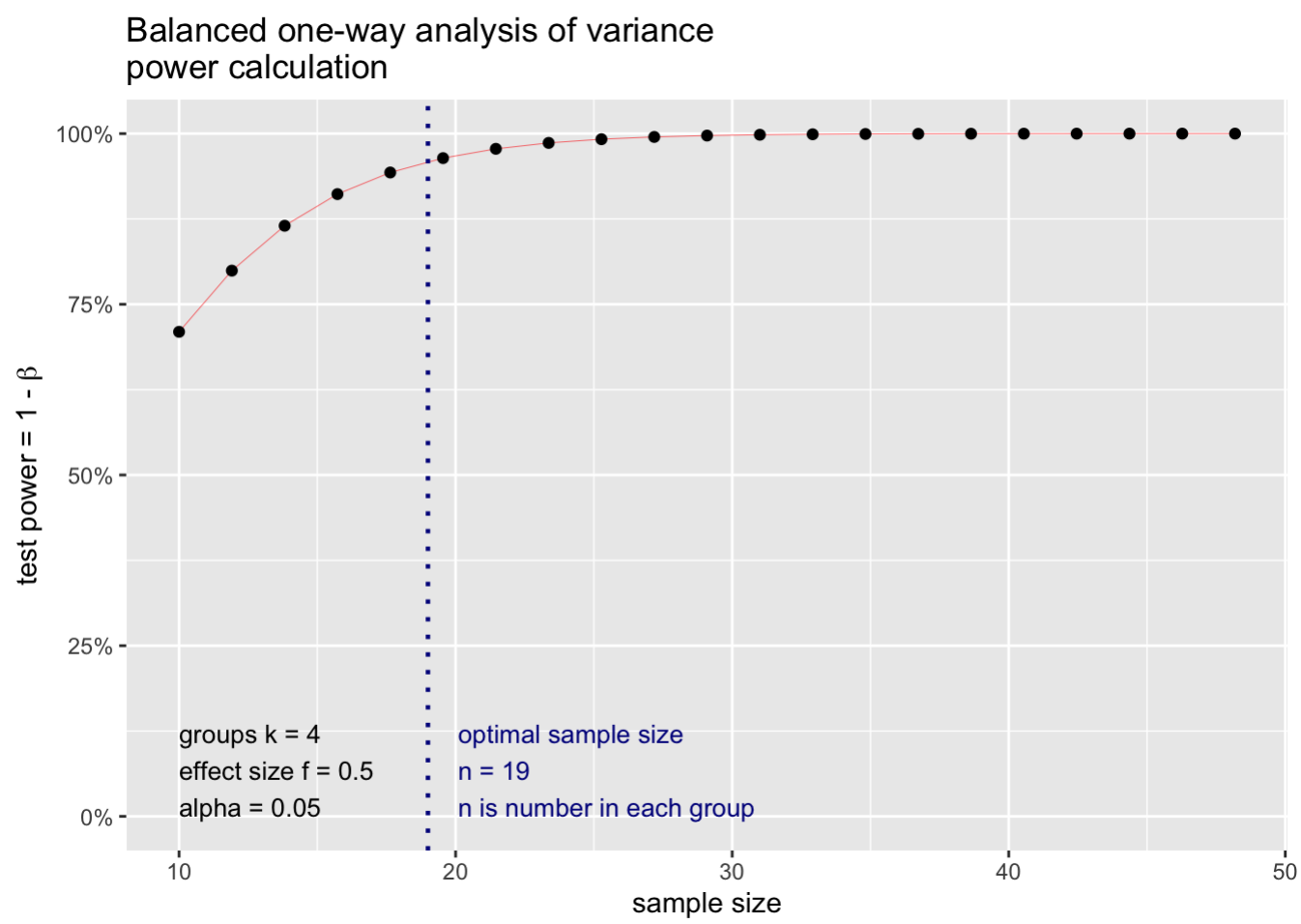
Bloco 2. Estimativa de tamanho amostral para análise de variância.

```
avn <- pwr.anova.test(k = 4, #Número de grupos experimentais
                      n = NULL, #Número de observações (tamanho amostral)
                      f = 0.5, #Tamanho de efeito f
                      sig.level = 0.05, #alfa (prob. de falso positivo)
                      power = .95) #1-beta (1 - prob. de falso negativo)

avn #Resultado:
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 4
##               n = 18.18244
##               f = 0.5
##       sig.level = 0.05
##           power = 0.95
##
## NOTE: n is number in each group
```

Bloco 3. Gráfico da estimativa de tamanho amostral para análise de variância.

```
plot.power.htest(avn) #Resultado:
```



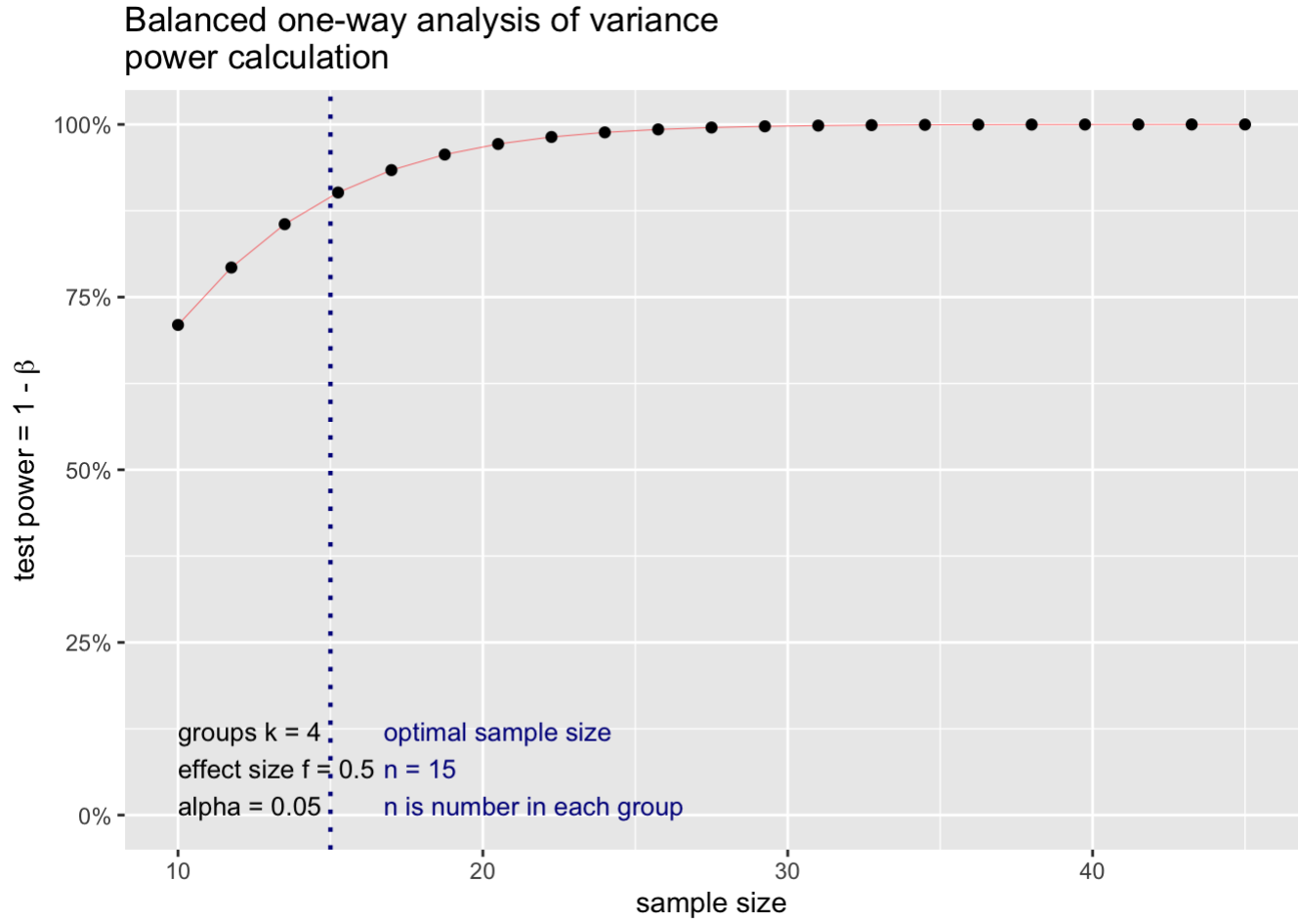
Bloco 4. Estimativa de poder estatístico para análise de variância.

```
avp <- pwr.anova.test(k = 4, #Número de grupos experimentais
                      n = 15, #Número de observações (tamanho amostral)
                      f = 0.5, #Tamanho de efeito f
                      sig.level = 0.05, #alfa (prob. de falso positivo)
                      power = NULL) #1-beta (1 - prob. de falso negativo)

avp #Resultado:
```

```
##
##      Balanced one-way analysis of variance power calculation
##
##               k = 4
##               n = 15
##               f = 0.5
##       sig.level = 0.05
##           power = 0.8957212
##
## NOTE: n is number in each group
```

Bloco 5. Gráfico da estimativa de poder estatístico para análise de variância.

```
plot.power.htest(avp) #Resultado:
```



O procedimento para estimar o poder estatístico e o tamanho amostral de testes t é similar ao realizado para a ANOVA (Bloco 6). Também similar é a geração de um gráfico representando a relação entre poder estatístico e tamanho amostral (Bloco 7).

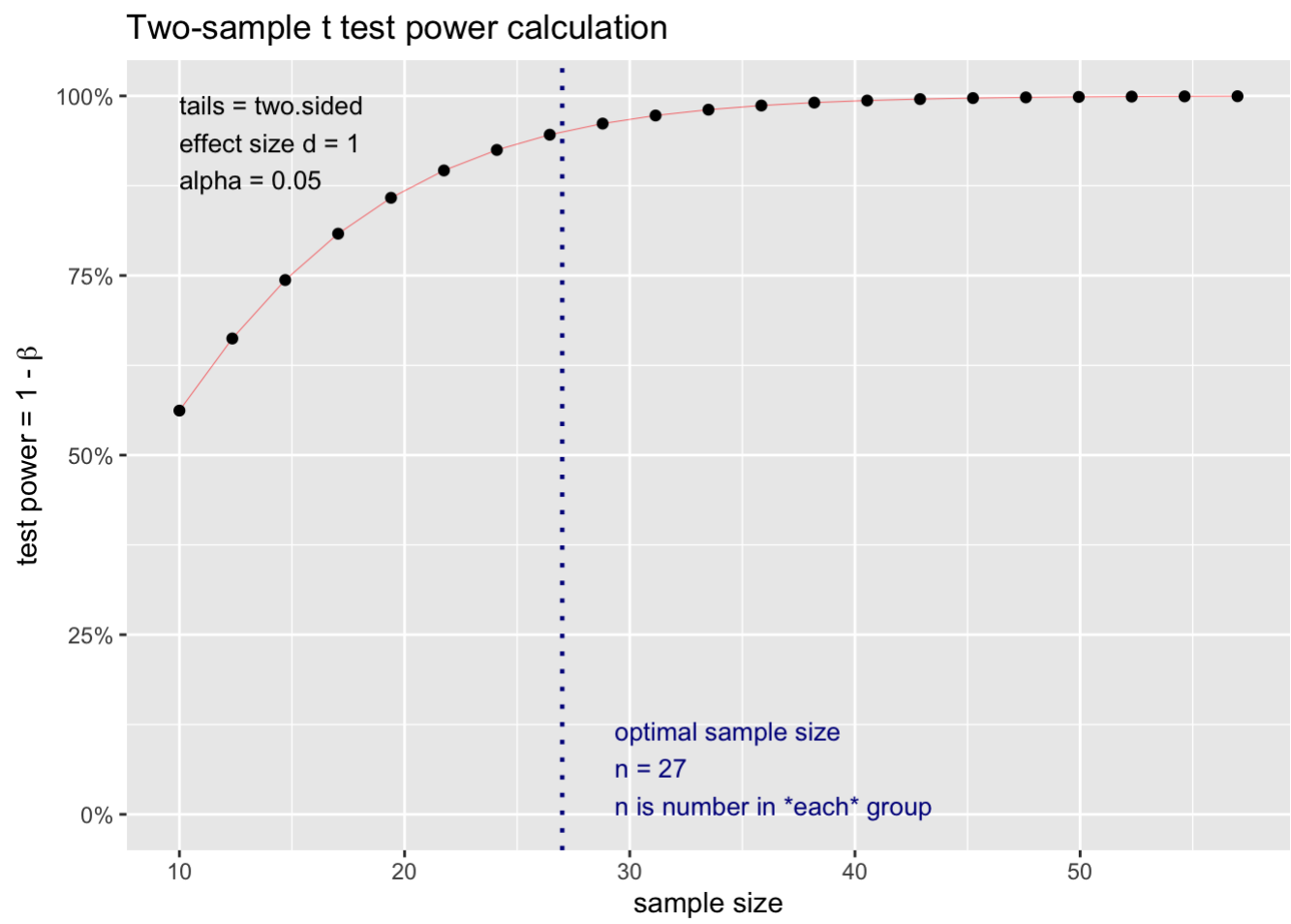
Bloco 6. Estimativa de tamanho amostral para testes t.

```
tt <- pwr.t.test(n = NULL, #Número de observações (tamanho amostral)
                 d = 1.0, #Tamanho de efeito d
                 sig.level = 0.05, #alfa (prob. de falso positivo)
                 power = .95, #1-beta (1 - prob. de falso negativo)
                 type = "two.sample", #Duas amostras, não pareadas
                 alternative = "two.sided") #Duas caudas

tt #Resultado:
```

```
##
##      Two-sample t test power calculation
##
##               n = 26.9892
##               d = 1
##       sig.level = 0.05
##           power = 0.95
##     alternative = two.sided
##
## NOTE: n is number in *each* group
```

Bloco 7. Gráfico da estimativa de tamanho amostral para testes t.

```
plot.power.htest(tt) #Resultado:
```



O pacote “pwr” oferece, ainda, a possibilidade de se estimar o tamanho de efeito (Cohen D) sem a necessidade de calculadoras ou aplicativos externos. No presente modelo, pode-se observar um exemplo da análise de tamanho de efeito entre os grupos “Controle” e “Tratamento_II” (ver Tabela 1 em Dados do modelo[1]), que resulta em um d = 5.96, com intervalo de confiança entre 4.15 e 7.78 (Bloco 8).

Bloco 8. Estimativa de tamanho de efeito (Cohen D).

```
cohen.d(df$Controle, df$Tratamento_II) #Resultado:
```

```
##
## Cohen's d
##
## d estimate: 5.969867 (large)
## 95 percent confidence interval:
##    lower    upper
## 4.155318 7.784415
```

Estatística descritiva e exportação de dados

Uma das primeiras etapas de análise dos dados coletados é a geração de estatísticas descritivas. Dependendo do projeto, o pesquisador precisará de mais ou menos informações sobre seus resultados, mas é procedimento comum no universo das Ciências Biológicas e da Saúde obter ao menos uma medida de tendência central (e.g. média) e uma medida de dispersão (e.g. desvio padrão) do conjunto de dados. No presente modelo, são apresentadas quatro alternativas de código para o cálculo de estatísticas descritivas. A utilização desses códigos, novamente, fica ao critério do pesquisador.

A primeira opção de comando, summary( ) de análise é gerada pelo ambiente nativo do R, e produz as seguintes informações: Mínimo, 1º quartil, Mediana, Média, 3º quartil e Máximo (Bloco 9).

Bloco 9. Estatística descritiva com o comando summary( ).

```
summary(df) #Resultado:
```

```
##     Controle        Positivo       Tratamento_I   Tratamento_II
##  Min.   :87.32   Min.   :0.1011   Min.   :87.46   Min.   :75.34
##  1st Qu.:90.57   1st Qu.:0.6326   1st Qu.:89.05   1st Qu.:75.72
##  Median :93.57   Median :0.8601   Median :91.77   Median :76.32
##  Mean   :93.64   Mean   :0.9766   Mean   :92.11   Mean   :76.46
##  3rd Qu.:97.65   3rd Qu.:1.3136   3rd Qu.:94.74   3rd Qu.:76.65
##  Max.   :98.85   Max.   :1.8930   Max.   :97.93   Max.   :78.72
```

A segunda opção de comando, stat.desc( ), utiliza o pacote “pastecs” e fornece as seguintes informações: Número de valores, Número de valores nulos, Número de valores ausentes, Mínimo, Máximo, Intervalo mínimo-máximo, Soma, Mediana, Média, Erro padrão da média, Intervalo de confiança 95%, Variância, Desvio padrão e Coeficiente de variação (Bloco 10).

Bloco 10. Estatística descritiva com o comando stat.desc( ).

```
#Para visualizar todos os resultados, acesse a versão em .html

stat.desc(df) #Resultado:
```

	Controle <dbl>	**Positivo** <dbl>	**Tratamento_I** <dbl>	**Tratamento_II** <dbl>
nbr.val	1.400000e+01	14.0000000	1.400000e+01	1.400000e+01
nbr.null	0.000000e+00	0.0000000	0.000000e+00	0.000000e+00
nbr.na	0.000000e+00	0.0000000	0.000000e+00	0.000000e+00
min	8.731900e+01	0.1011122	8.745515e+01	7.533878e+01
max	9.884755e+01	1.8930254	9.793014e+01	7.872481e+01
range	1.152854e+01	1.7919132	1.047499e+01	3.386034e+00
sum	1.310963e+03	13.6724383	1.289558e+03	1.070388e+03
median	9.357047e+01	0.8600732	9.176549e+01	7.631975e+01
mean	9.364019e+01	0.9766027	9.211128e+01	7.645629e+01
SE.mean	1.052423e+00	0.1450803	9.448416e-01	2.757487e-01

1-10 of 14 rows Previous **1** 2 Next

O comando skim( ), do pacote “skimr,” possui a habilidade adicional de fornecer um histograma simples dos dados, além de outras informações relevantes, como: Número de linhas, Número de colunas, Tipo de dado nas colunas (ex.: numérico), Variáveis agrupadas (categóricos), Nome da variável, Valores faltando, Taxa de completude (indica ausência de valores), Média, Desvio padrão (sd), Percentil 0 (p0), Percentil 25 (p25), Percentil 50 (p50), Percentil 75 (p75), Percentil 100 (p100) e os Histogramas (Bloco 11).

Bloco 11. Estatística descritiva com o comando skim( ).

```
skim(df)
```

Data summary

Name	df
Number of rows	14
Number of columns	4
_____	
Column type frequency:	
numeric	4
_____	
Group variables	None

Variable type: numeric

skim_variable	**n_missing**	**complete_rate**	**mean**	**sd**	**p0**	**p25**	**p50**	**p75**	**p100**	**hist**
Controle	0	1	93.64	3.94	87.32	90.57	93.57	97.65	98.85	▃▇▁▂▇
Positivo	0	1	0.98	0.54	0.10	0.63	0.86	1.31	1.89	▃▇▇▂▅
Tratamento_I	0	1	92.11	3.54	87.46	89.05	91.77	94.74	97.93	▇▇▇▁▆
Tratamento_II	0	1	76.46	1.03	75.34	75.72	76.32	76.65	78.72	▇▇▅▁▂

O pacote “psych,” por sua vez, possui o comando describe( ), que fornece dados importantes sobre o enviesamento e a curtose das distribuições contidas no conjunto de dados. São as informações geradas por describe( ): Variáveis, Número de observações, Média, Desvio padrão (sd), Mediana, Média truncada, Desvio mediano absoluto (mad), Mínimo, Máximo, Intervalo mínimo-máximo, Enviesamento (skew), Curtose (kurtosis) e Erro padrão da média (se) (Bloco 12).

Bloco 12. Estatística descritiva com o comando describe( ).

```
#Para visualizar todos os resultados, acesse a versão em .html

describe(df)
```

	vars <int>	n <dbl>	mean <dbl>	sd <dbl>	median <dbl>	trimmed <dbl>	mad <dbl>	min <dbl>	max <dbl>
Controle	1	14	93.6401940	3.9378060	93.5704721	93.7330140	5.5677772	87.3190025	98.847546
Positivo	2	14	0.9766027	0.5428409	0.8600732	0.9731917	0.5399030	0.1011122	1.893025
Tratamento_I	3	14	92.1112810	3.5352735	91.7654950	92.0143864	5.0935035	87.4551540	97.930142
Tratamento_II	4	14	76.4562879	1.0317571	76.3197468	76.3603701	0.7249656	75.3387776	78.724812

4 rows | 1-10 of 14 columns

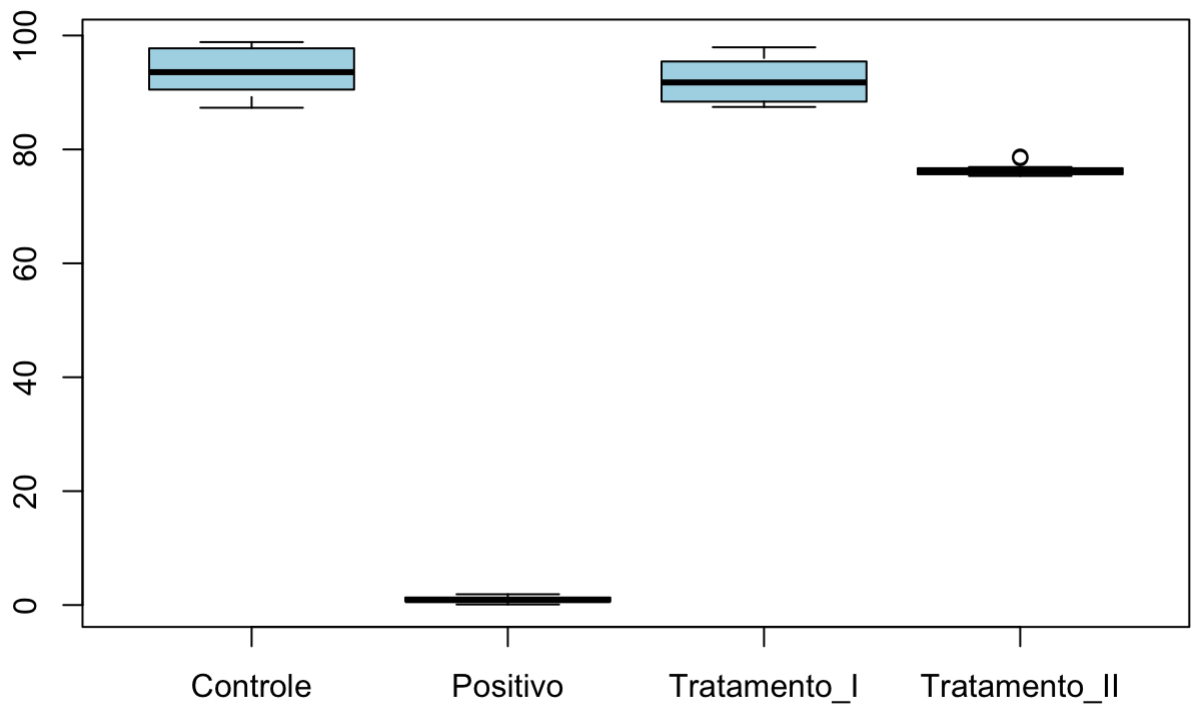
Os dados produzidos pelos comandos acima podem, ainda, ser exportados no formato .xlsx utilizando o pacote “xlsx.” Para isso, primeiro é necessário atribuir o comando a um objeto. Utilizando “describe(df)” como exemplo, tem-se: “**tabela <- describe(df)**,” onde tabela é o novo objeto contendo as informações geradas por describe( ). Depois, basta executar o comando “**write.xlsx(x = tabela, file = ‘caminho_onde_o_arquivo_será_salvo.xlsx’)**.” Não existe a necessidade, porém, de exportar todos os dados para um arquivo, já que eles podem ser visualizados e/ou copiados do console.

Geração e exportação de gráficos

Uma das principais vantagens do ambiente R é a capacidade produzir visualizações altamente personalizáveis e com qualidade de publicação. O ambiente nativo do R (base) é capaz de produzir gráficos a partir de um conjuto de dados com um simples comando “**plot(df)**” (lembre-se que “df” é o nome atribuído ao objeto que contém as informações do conjunto de dados do modelo). Apesar disso, o gráfico gerado pode não atender exatamente ao objetivo do pesquisador, tornando necessário o fornecimento de um código mais específico. No Bloco 13, por exemplo, um conjunto de diagramas de caixa (*boxplots*) é gerado a partir de todas as variáveis do modelo.

Bloco 13. Diagramas de caixa (*boxplots*) das variáveis do modelo.

```
boxplot(df$Controle,
        df$Positivo,
        df$Tratamento_I,
        df$Tratamento_II, #Determina as variáveis de df para o boxplot
        col = "lightblue", #Determina a cor de preenchimento
        names = c("Controle",
                  "Positivo",
                  "Tratamento_I",
                  "Tratamento_II")) #Determina os nomes de cada grupo | #Resultado:
```



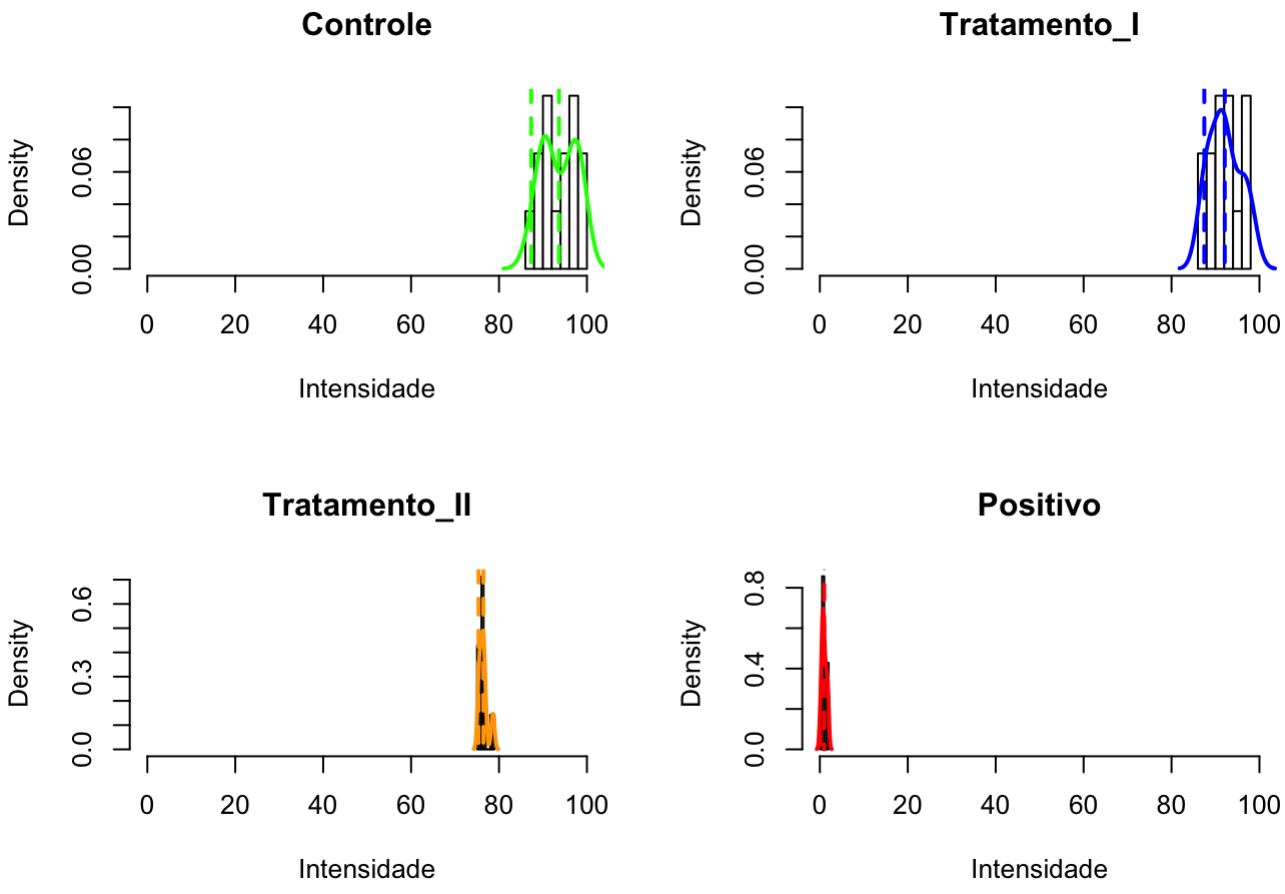
Em R, também é possível gerar vários gráficos de forma simultânea. Esses gráficos serão organizados automaticamente na janela de visualização de acordo com a segmentação imposta pelo comando “**par(mfrow = c(x, y))**,” onde “x” é o número de linhas desejadas e “y” é o número de colunas. O Bloco 14 apresenta um exemplo onde, para cada grupo do modelo, foi gerado um histograma sobreposto pela linha de

densidade da distribuição, além de uma linha pontilhada indicando a média e outra, indicando o valor mínimo. Todos os gráficos foram apresentados simultaneamente em uma matriz do tipo 2x2.

Bloco 14. Matriz de gráficos com sobreposição de elementos.

```
par(mfrow = c(2, 2)) #Quatro linhas, uma coluna

hist(df$Controle, #Histograma do grupo controle
     freq = F, #Sem contagem de frequências
     xlim = c(0,100), #Limites do eixo x
     col = "white", #Cor desejada para o gráfico
     main = "Controle", #Título
     xlab = "Intensidade") #Legenda do eixo x
lines(density((df$Controle)), #Gráfico de densidade
      col = "green", #Cor desejada para o gráfico
      lwd = 2) #Espessura da linha
lines(x = c(mean(df$Controle), mean(df$Controle)), #xy da Linha
      y = c(mean(df$Controle), 0), #xy da Linha
      col = "green", #Cor desejada para o gráfico
      lty = 2, #Tipo de linha (pontilhada)
      lwd = 2) #Espessura da linha
lines(x = c(min(df$Controle), min(df$Controle)), #xy da Linha
      y = c(min(df$Controle), 0), #xy da Linha
      col = "green", #Cor desejada para o gráfico
      lty = 2, #Tipo de linha (pontilhada)
      lwd = 2) #Espessura da linha

hist(df$Tratamento_I, #Histograma do grupo controle
     freq = F,
     xlim = c(0,100),
     col = "white",
     main = "Tratamento_I",
     xlab = "Intensidade")
lines(density((df$Tratamento_I)),
      col = "blue",
      lwd = 2)
lines(x = c(mean(df$Tratamento_I), mean(df$Tratamento_I)),
      y = c(mean(df$Tratamento_I), 0),
      col = "blue",
      lty = 2,
      lwd = 2)
lines(x = c(min(df$Tratamento_I), min(df$Tratamento_I)),
      y = c(min(df$Tratamento_I), 0),
      col = "blue",
      lty = 2,
      lwd = 2)

hist(df$Tratamento_II, #Histograma do grupo controle
     freq = F,
     xlim = c(0,100),
     col = "white",
     main = "Tratamento_II",
     xlab = "Intensidade")
lines(density((df$Tratamento_II)),
      col = "orange",
      lwd = 2)
lines(x = c(mean(df$Tratamento_II), mean(df$Tratamento_II)),
      y = c(mean(df$Tratamento_II), 0),
      col = "orange",
      lty = 2,
      lwd = 2)
lines(x = c(min(df$Tratamento_II), min(df$Tratamento_II)),
      y = c(min(df$Tratamento_II), 0),
      col = "orange",
      lty = 2,
      lwd = 2)

hist(df$Positivo, #Histograma do grupo controle
     freq = F,
     xlim = c(0,100),
     col = "white",
     main = "Positivo",
     xlab = "Intensidade")
lines(density((df$Positivo)),
      col = "red",
      lwd = 2)
lines(x = c(mean(df$Positivo), mean(df$Positivo)),
      y = c(mean(df$Positivo), 0),
      col = "red",
      lty = 2,
      lwd = 2)
lines(x = c(min(df$Positivo), min(df$Positivo)),
      y = c(min(df$Positivo), 0),
      col = "red",
```

```
      lty = 2,
      lwd = 2) #Resultado:
```



Apesar de bastante versátil, o pacote base do R possui limitações quando se trata de gráficos mais complexos. Para solucionar o problema, alguns pacotes podem ser importados para o ambiente, tornando-o uma das melhores ferramentas para geração de figuras científicas disponíveis na atualidade. O pacote mais utilizado no momento é o "ggplot2," que constrói gráficos em um sistema de camadas, onde cada bloco de comando insere um elemento à figura.

A forma como o "ggplot2" interpreta os dados do modelo é um pouco diferente daquela utilizada nos gráficos de base R. Por isso, é necessário, em certos casos, modificar a estrutura dos dados de maneira a convertê-la do formato "largo" (*wide*) para o formato "longo" (*long*). Essa transformação pode ser realizada com a função melt( ) do pacote "reshape2" (Tabela 2). Compare a Tabela 2, abaixo, com a Tabela 1, em Dados do modelo[1], que ilustra o formato original dos dados.

Tabela 2. Convertendo do formato *wide* para o formato *long* com a função melt( ).

```
dtt <- melt(df, id.vars = 0) #formato wide para o formato long

dbb <- dtt #Atribui "dtt" a "dbb"

kk <- kbl(caption = "Modelo I - long",
          x = dbb) #Cria tabela kbl() e atribui a "kk"

kable_classic(full_width = F,
              html_font = "Cambria",
              kable_input = kk) #Define o tema da tabela | #Resultado:
```

Modelo I - long

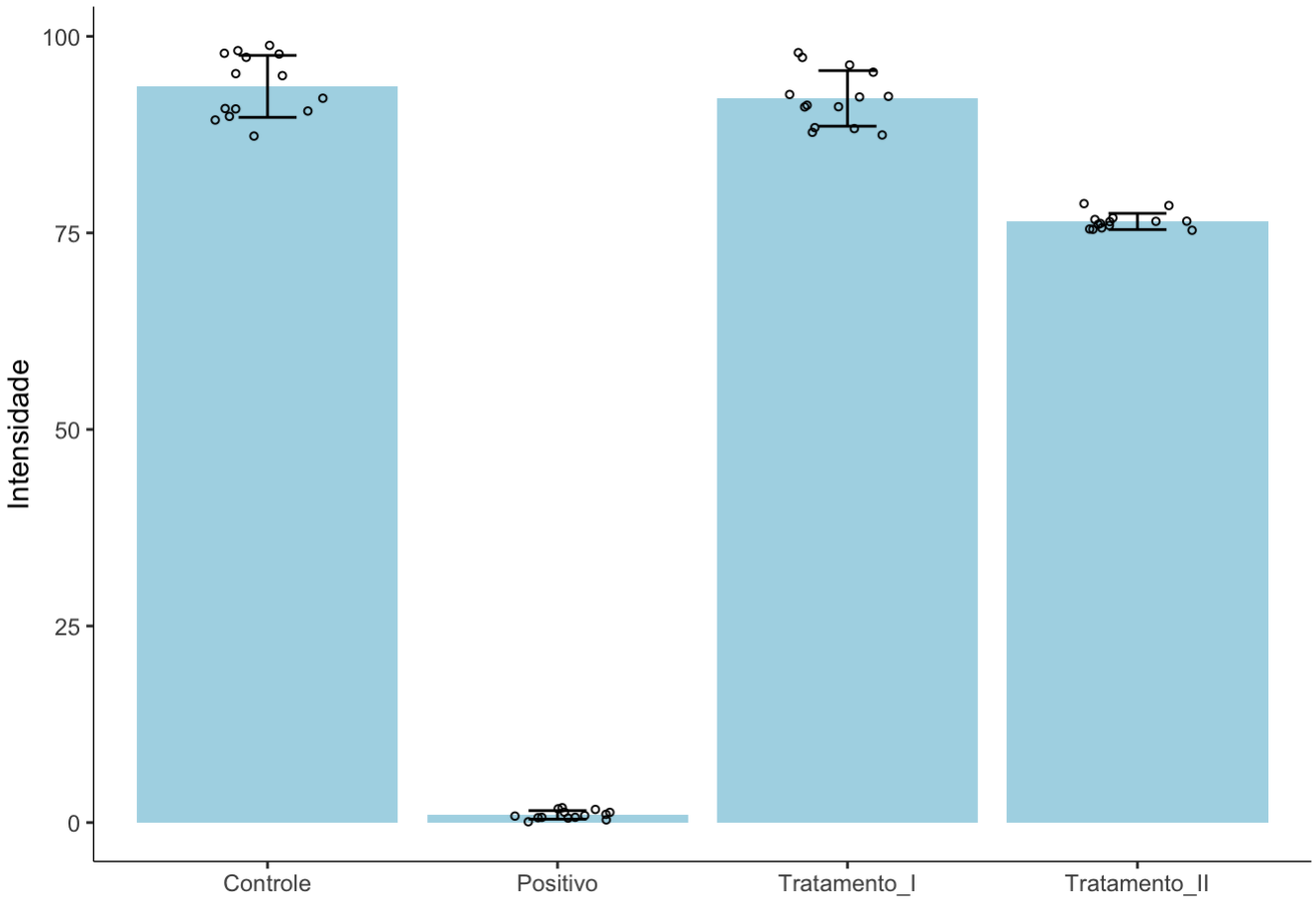
variable	value
Controle	95.2618454
Controle	92.1338155
Controle	97.8461538
Controle	98.8475457
Controle	90.5070494
Controle	87.3190025
Controle	90.7728169
Controle	89.3611418
Controle	95.0071287
Controle	97.3302083
Controle	90.8215810
Controle	89.8193797
Controle	97.7543408
Controle	98.1807070
Positivo	0.6711409
Positivo	0.1011122
Positivo	0.3436426
Positivo	1.6676875
Positivo	0.6223255
Positivo	0.6634388

variable	value
Positivo	1.2961154
Positivo	1.7559411
Positivo	0.5677962
Positivo	0.8238413
Positivo	1.0506822
Positivo	0.8963051
Positivo	1.3193839
Positivo	1.8930254
Tratamento_I	97.9301423
Tratamento_I	88.3916084
Tratamento_I	96.3636364
Tratamento_I	92.3673454
Tratamento_I	88.2683394
Tratamento_I	97.3188209
Tratamento_I	87.7953985
Tratamento_I	95.4437578
Tratamento_I	87.4551540
Tratamento_I	91.0268956
Tratamento_I	92.6127377
Tratamento_I	91.0531072
Tratamento_I	91.2419459
Tratamento_I	92.2890440
Tratamento_II	76.4720108
Tratamento_II	75.6400126
Tratamento_II	76.4945338
Tratamento_II	75.3387776
Tratamento_II	75.4582028
Tratamento_II	78.7248117
Tratamento_II	76.7042270
Tratamento_II	76.2073594
Tratamento_II	76.0801705
Tratamento_II	75.4944509
Tratamento_II	76.4321343
Tratamento_II	78.4852230
Tratamento_II	75.9428846
Tratamento_II	76.9132319

Depois de transformado, o modelo pode ser utilizado para a criação de um ggplot, que é o elemento básico de um gráfico com o pacote “ggplot2.” No Bloco 15, abaixo, calculou-se a média e desvio padrão de cada variável, que depois foram utilizados na geração de um ggplot do tipo gráfico de barras (“geom_bar”), contendo os pontos individuais de cada grupo (“geom_jitter”) e as correspondentes barras de erro (“geom_errorbar”).

Bloco 15. Geração de gráfico de barras com ggplot2.

```
#dtt$variable <- as.factor(dtt$variable)

dttt <- dtt %>%
  group_by(variable) %>% #Agrupa as variáveis do modelo
  summarise(sd = sd(value, #Calcula os desvios padrão
                    na.rm = TRUE), #Elimina valores ausentes
            len = mean(value)) #Calcula as médias

ggplot(dtt, aes(x = variable,
                y = value)) + #Cria o gráfico
  geom_bar(stat = "summary",
           fun = "mean",
           fill = "lightblue") + #Determina o tipo (barras)
  geom_jitter(position = position_jitter(0.2),
              color = "black",
              size = 1,
              shape = 21) + #Adiciona pontos individuais
  labs(x = " ",
       y = "Intensidade") + #Nomeia os eixos
  theme(panel.background = element_blank(),
        axis.line.y = element_line("black", size = .25),
        axis.line.x = element_line("black", size = .25)) + #Muda o tema
  geom_errorbar(aes(x = variable,
                    y = len,
                    ymin = len-sd,
                    ymax = len+sd),
                data = dttt,
                width = 0.2) #Adiciona as barras de erro | #Resultado:
```



O próprio RStudio possui botão “Export” disponível em sua interface, facilitando a rápida obtenção de figuras, .pdfs e até de cópias de qualquer item apresentado na subjanela “Plots.” O mesmo procedimento pode ser feito para um ggplot, que também traz em seu pacote a possibilidade de personalizar as características da figura sendo salva (geralmente para qualidade de publicação). Exportar um gráfico do ggplot pode ser feito com o comando “**ggsave(filename = "nome_do_arquivo.tiff", path = (‘caminho_onde_o_arquivo_será_salvo.xlsx’), dpi = 300)**,” onde “.tiff” é o formato da figura (que poderia ser .pdf, .png, .jpeg ou outros), “path” é o local no computador onde a figura será salva e “dpi” (*dots per inch*) é a resolução da figura.

Valores extremos, normalidade e testes de hipótese

Um dos grandes problemas que permeiam as Biociências modernas é a escolha inadequada de testes de hipótese na comparação das variáveis de um determinado modelo. Isso provoca baixa reprodutibilidade e robustez nas conclusões produzidas pelos estudos. Uma das maneiras de contornar o problema usando o R é tirar proveito da vasta documentação e pacotes disponíveis, que permitem identificar a melhor abordagem possível na estimativa de tamanho amostral e poder estatístico, na identificação de valores extremos (*outliers*) e na escolha de testes de hipótese. Infelizmente, muitos desses recursos só estão disponíveis em Inglês, sendo pouco acessíveis ao usuário inexperiente. Contribuir para a solução desse problema é um dos objetivos da criação do repositório heRcules (https://github.com/drhrf/heRcules.git)[1].

O pacote “rstatix,” através do comando identify_outliers( ), permite a identificação de *outliers* e valores extremos (conotações diferentes no comando). O procedimento é similar ao realizado no Bloco 15 para a determinação de média e desvio padrão do modelo em formato *long* (Bloco 16). Nota-se que, no presente modelo, só foram econtrados *outliers* no grupo “Tratamento_II” (é possível vê-los no *boxplot* do Bloco 13). Cabe ressaltar que o operador pipe (%>%), utilizado aqui, depende do pacote “dplyr,” também importado no presente modelo (veja Pacotes[1]).

Bloco 16. Identificação de *outliers* e valores extremos em R.

```
dtt %>%
  group_by(variable) %>% #Agrupa as variáveis do modelo
  identify_outliers(value) #Calcula outliers | #Resultado:
```

variable <fct>	value <dbl>	is.outlier <lgl>	is.extreme <lgl>
Tratamento_II	78.72481	TRUE	FALSE
Tratamento_II	78.48522	TRUE	FALSE

2 rows

A avaliação de normalidade também é outro componente importante para a realização de testes de hipótese que a assumem *a priori*. Uma possibilidade é a realização do teste de Shapiro-Wilk (SHAPIRO & WILK, 1965), que no R usa o comando shapiroTest( ), do pacote “fBasics.” Nota-se, no Bloco 17, que o Tratamento_II não aparenta ser distribuído normalmente no modelo (P = 0.02081). Cabe ao pesquisador tomar a decisão quanto à possibilidade de prosseguir ou não com os testes de hipótese após essa informação.

Bloco 17. Teste de normalidade por Shapiro-Wilk em R.

```
shapiroTest(x = df$Controle) #Resultado:
```

```
##
## Title:
##  Shapiro - Wilk Normality Test
##
## Test Results:
##   STATISTIC:
##     W: 0.8995
##   P VALUE:
##     0.111
##
## Description:
##  Sun Dec 26 12:35:25 2021 by user:
```

```
shapiroTest(x = df$Positivo) #Resultado:
```

```
##
## Title:
##  Shapiro - Wilk Normality Test
##
## Test Results:
##   STATISTIC:
##     W: 0.9557
##   P VALUE:
##     0.6527
##
## Description:
##  Sun Dec 26 12:35:25 2021 by user:
```

```
shapiroTest(x = df$Tratamento_I) #Resultado:
```

```
##
## Title:
##  Shapiro - Wilk Normality Test
##
## Test Results:
##   STATISTIC:
##     W: 0.9219
##   P VALUE:
##     0.2343
##
## Description:
##  Sun Dec 26 12:35:25 2021 by user:
```

```
shapiroTest(x = df$Tratamento_II) #Resultado:
```

```
##
## Title:
##  Shapiro - Wilk Normality Test
##
## Test Results:
##   STATISTIC:
##     W: 0.8479
##   P VALUE:
##     0.02081
##
## Description:
##  Sun Dec 26 12:35:25 2021 by user:
```

Existem diversas opções para a realização de análise de variância (ANOVA) em R. Uma delas é feita através da geração de um modelo ANOVA com o comando aov( ), que depende do pacote “stats.” O procedimento de criação do modelo segue o formato “**aov(variável dependente ~ variável independente)**,” cujos resultados podem ser apresentados utilizando o comando summary( ), tal como indicado no Bloco 18.

Bloco 18. Criação e descrição do modelo ANOVA em R.

```
res.aov <- aov(dtt$value ~ dtt$variable) #Cria o modelo ANOVA

summary(res.aov) #Resultado:
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## dtt$variable  3  80962   26987    3676 <2e-16 ***
## Residuals    52    382       7
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

É comum ser necessário, após a realização da ANOVA, a utilização de algum teste *post-hoc*. O teste de significância honesta de Tukey (TUKEY, 1949) é um dos recursos disponíveis para essa demanda (é importante ressaltar que o pesquisador sempre deve se certificar da adequação do teste aos seus resultados experimentais). O Bloco 19, abaixo, ilustra a utilização desse teste na resultante do ANOVA gerado pelo Bloco 18. O comando TukeyHSD( ) depende do pacote “stats.”

Bloco 19. Teste *post-hoc* de Tukey em R.

```
TukeyHSD(res.aov,
         conf.level = .95) #Testa e apresenta os resultados:
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = dtt$value ~ dtt$variable)
##
## $`dtt$variable`
##                                 diff        lwr        upr     p adj
## Positivo-Controle          -92.663591 -95.381553 -89.945630 0.0000000
## Tratamento_I-Controle       -1.528913  -4.246874   1.189048 0.4489958
## Tratamento_II-Controle     -17.183906 -19.901867 -14.465945 0.0000000
## Tratamento_I-Positivo       91.134678  88.416717  93.852640 0.0000000
## Tratamento_II-Positivo      75.479685  72.761724  78.197647 0.0000000
## Tratamento_II-Tratamento_I -15.654993 -18.372954 -12.937032 0.0000000
```

Finalmente, alguns modelos experimentais podem exigir a comparação entre apenas dois grupos, situação onde um modelo de ANOVA não é apropriado. Nesse caso, é comum ser apropriada a realização de um teste da família de testes *t*. Abaixo, encontram-se exemplos de comparações múltiplas entre pares de grupos do modelo, ilustrando a utilização do teste *t* no ambiente do R (Bloco 20). Tal como o teste de Tukey e a ANOVA, o comando t.test( ) depende do pacote “stats.”

Bloco 20. Múltiplos testes *t* em R.

```
t.test(df$Controle, df$Positivo) #Testa e apresenta os resultados:
```

```
##
##  Welch Two Sample t-test
##
## data:  df$Controle and df$Positivo
## t = 87.223, df = 13.494, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  90.37698 94.95020
## sample estimates:
##  mean of x  mean of y
## 93.6401940  0.9766027
```

```
t.test(df$Tratamento_I, df$Positivo) #Testa e apresenta os resultados:
```

```
##
##  Welch Two Sample t-test
##
## data:  df$Tratamento_I and df$Positivo
## t = 95.338, df = 13.613, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  89.07896 93.19040
## sample estimates:
##  mean of x  mean of y
## 92.1112810  0.9766027
```

```
t.test(df$Tratamento_II, df$Positivo) #Testa e apresenta os resultados:
```

```
##
##  Welch Two Sample t-test
##
## data:  df$Tratamento_II and df$Positivo
## t = 242.24, df = 19.685, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  74.82906 76.13031
## sample estimates:
##  mean of x  mean of y
## 76.4562879  0.9766027
```

```
t.test(df$Controle, df$Tratamento_I) #Testa e apresenta os resultados:
```

```
##
##  Welch Two Sample t-test
##
## data:  df$Controle and df$Tratamento_I
## t = 1.081, df = 25.703, p-value = 0.2897
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.379910  4.437736
## sample estimates:
## mean of x mean of y
##  93.64019  92.11128
```

```
t.test(df$Controle, df$Tratamento_II) #Testa e apresenta os resultados:
```

```
##
##  Welch Two Sample t-test
##
## data:  df$Controle and df$Tratamento_II
## t = 15.795, df = 14.777, p-value = 1.168e-10
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  14.86194 19.50587
## sample estimates:
## mean of x mean of y
##  93.64019  76.45629
```

Conclusão

O crescimento no número de usuários que procura linguagens como R e Python para análise de dados científicos, em detrimento de ferramentas mais limitadas e/ou pagas, é um sinal de progresso a favor da reprodutibilidade na ciência. O objetivo do presente trabalho é marcar o início de um esforço direcionado a fornecer recursos estruturados, e em língua portuguesa, a pesquisadores em processo de transição para a linguagem R.

Contribuição dos autores

Todas as etapas do presente trabalho, da concepção à publicação, foram realizadas por HRF.

ORCID

0000-0003-1584-9157

Declaração de conflito de interesses

O autor declara não haver conflitos de interesse.

Fontes de financiamento

Conselho Nacional de Desenvolvimento Científico e Tecnológico, número 152071/2020-2.

Referências

DEBASTIANI, V. J. (2020). Introdução ao R. [S. l.]. Disponível em: https://vanderleidebastiani.github.io/tutoriais/Introducao_ao_R.html. (https://vanderleidebastiani.github.io/tutoriais/Introducao_ao_R.html.) Acesso em: 21 dez. 2021.

R CORE TEAM (2013). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL http://www.R-project.org/. (http://www.R-project.org/.)

REVELLE, W. (2021) psych: Procedures for Personality and Psychological Research, Northwestern University, Evanston, Illinois, USA, https://CRAN.R-project.org/package=psych (https://CRAN.R-project.org/package=psych) Version = 2.1.9,.

WICKHAM, H.; FRANÇOIS, R.; HENRY, L. & MÜLLER, K. (2021). dplyr: A Grammar of Data Manipulation. R package version 1.0.7. https://CRAN.R-project.org/package=dplyr (https://CRAN.R-project.org/package=dplyr).

GROSJEAN, P. & IBANEZ, F. (2018). pastecs: Package for Analysis of Space-Time Ecological Series. R package version 1.3.21. https://CRAN.R-project.org/package=pastecs (https://CRAN.R-project.org/package=pastecs).

WUERTZ, D.; SETZ, T. & CHALABI, Y. (2020). fBasics: Rmetrics - Markets and Basic Statistics. R package version 3042.89.1. https://CRAN.R-project.org/package=fBasics (https://CRAN.R-project.org/package=fBasics).

WARING, E.; QUINN, M.; MCNAMARA, A.; LA RUBIA, E. A.; ZHU, H. & ELLIS, S. (2021). skimr: Compact and Flexible Summaries of Data. R package version 2.1.3. https://CRAN.R-project.org/package=skimr (https://CRAN.R-project.org/package=skimr) .

WICKHAM, H. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

DRAGULESCU, A. & ARENDT, C. (2020). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.5. https://CRAN.R-project.org/package=xlsx (https://CRAN.R-project.org/package=xlsx).

WICKHAM, H. (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/. (http://www.jstatsoft.org/v21/i12/.)

KASSAMBARA, A. (2021). rstatix: Pipe-Friendly Framework for Basic Statistical Tests. R package version 0.7.0. https://CRAN.R-project.org/package=rstatix (https://CRAN.R-project.org/package=rstatix).

ZHU, H. (2021). kableExtra: Construct Complex Table with 'kable' and Pipe Syntax. R package version 1.3.4. https://CRAN.R-project.org/package=kableExtra (https://CRAN.R-project.org/package=kableExtra).

CHAMPELY, S. (2020). pwr: Basic Functions for Power Analysis. R package version 1.3-0. https://CRAN.R-project.org/package=pwr (https://CRAN.R-project.org/package=pwr).

TORCHIANO, M. (2020). _effsize: Efficient Effect Size Computation_. doi: 10.5281/zenodo.1480624 (URL: https://doi.org/10.5281/zenodo.1480624), (https://doi.org/10.5281/zenodo.1480624),) R package version 0.8.1, <URL: https://CRAN.R-project.org/package=effsize (https://CRAN.R-project.org/package=effsize).

SHAPIRO, A. S. S.; & WILK, M. B. (1965). An Analysis of Variance Test for Normality (Complete Samples). Biometrika, 52(3/4), 591–611. Link: https://doi.org/https://doi.org/10.2307/2333709. (https://doi.org/https://doi.org/10.2307/2333709.)

TUKEY, J. W. (1949). Comparing individual means in the analysis of variance. *Biometrics*, 99-114. Link: https://doi.org/10.2307/3001913 (https://doi.org/10.2307/3001913).

---

1. Hiperlinks.↩

Este preprint foi submetido sob as seguintes condições:

- Os autores declaram que estão cientes que são os únicos responsáveis pelo conteúdo do preprint e que o depósito no SciELO Preprints não significa nenhum compromisso de parte do SciELO, exceto sua preservação e disseminação.
- Os autores declaram que os necessários Termos de Consentimento Livre e Esclarecido de participantes ou pacientes na pesquisa foram obtidos e estão descritos no manuscrito, quando aplicável.
- Os autores declaram que a elaboração do manuscrito seguiu as normas éticas de comunicação científica.
- Os autores declaram que os dados, aplicativos e outros conteúdos subjacentes ao manuscrito estão referenciados.
- O manuscrito depositado está no formato PDF.
- Os autores declaram que a pesquisa que deu origem ao manuscrito seguiu as boas práticas éticas e que as necessárias aprovações de comitês de ética de pesquisa, quando aplicável, estão descritas no manuscrito.
- Os autores concordam que caso o manuscrito venha a ser aceito e postado no servidor SciELO Preprints, a retirada do mesmo se dará mediante retratação.
- Os autores concordam que o manuscrito aprovado será disponibilizado sob licença Creative Commons CC-BY.
- O autor submissor declara que as contribuições de todos os autores e declaração de conflito de interesses estão incluídas de maneira explícita e em seções específicas do manuscrito.
- Os autores declaram que o manuscrito não foi depositado e/ou disponibilizado previamente em outro servidor de preprints ou publicado em um periódico.
- Caso o manuscrito esteja em processo de avaliação ou sendo preparado para publicação mas ainda não publicado por um periódico, os autores declaram que receberam autorização do periódico para realizar este depósito.
- O autor submissor declara que todos os autores do manuscrito concordam com a submissão ao SciELO Preprints.