3

José Maurício Schneedorf Ferreira da Silva

SCRIPTRPLOT:

UM SCRIPT EM R PARA PRODUÇÃO DE GRÁFICOS ELEGANTES PARA *GGPLOT2*

DOI: 10.31560/pimentacultural/978-85-7221-396-7.3

INTRODUÇÃO

Ensino reprodutível (*ER*, "*reproducible teaching*") é uma abordagem ativa para ensino e aprendizagem que utiliza linguagem de programação para a produção de textos, objetos didáticos, ou sua combinação. Tem suas raízes na *pesquisa reproduzível*, ressurgida na Universidade de Standford pelo cientista da computação Donald E. Knuth (Knuth, 1984), que, ao combinar a linguagem de formatação documental *Tex* de sua autoria com a linguagem de programação *Pascal*, possibilitou a saída de um único documento contendo texto estilizado e trechos de códigos reproduzíveis pelo programa *WEB*, isso há quase uma década antes do advento da internet. Essa junção culminou na compilação de um *documento dinâmico* final e estabeleceu o conceito de *literate programming*, ou programação letrada. A ideia teve sequência na mesma instituição anos mais tarde (Claerbout; Karrenbach, 1992), visando a reprodutibilidade de processamento e filtragem de dados de exploração sísmica, permitindo que profissionais distintos da equipe pudessem reproduzir cálculos, análises e figuras por meio da execução das linhas do código-fonte.

Essa possibilidade de reprodução de um experimento completo, desde sua concepção até sua publicitação e publicação técnico-científica com auxílio de uma linguagem de programação, vem sendo largamente discutida na academia nos últimos anos para a consolidação de uma Ciência Aberta (*Pesquisa Reprodutível*) (Hernandez; Colom, 2025), bem como de uma Educação Aberta (*ER*) (Dogucu, 2024), faces de um mesmo monolito concebido a partir de 1) dados originais, 2) código de análise/produção, 3) documentação e 4) compartilhamento irrestrito (Bean, 2023).

APLICAÇÕES GRÁFICAS PARA PESQUISA & ENSINO-APRENDIZAGEM

Entre os produtos pertinentes ao binômio de *PR* e *ER* encontram-se os gráficos, representações visuais de linguagem algébrica para dados. Ao lado de tabelas e mapas, gráficos constituem a principal argamassa de visualização de dados para manuscritos e livros-texto de abordagem técnico-científica, outrora erigidos em papel milimetrado e publicados com auxílio de papel vegetal sulfurizado. Atualmente, há muitos recursos digitais para a construção de gráficos e sua publicitação, utilizáveis em computadores, em dispositivos móveis e em nuvem. Por envolver comumente a necessidade de cálculos e análises estatísticas na academia, os aplicativos são inseridos em suítes gráficas e de análise de dados. Boa parte desses programas exigem licença paga, tais como Prisma Graphpad, Origin Lab, MathWorks Matlab, Wolfram Mathematica, Microsoft Power Bi e Grafiti SigmaPlot.

Também há diversas alternativas gratuitas, entre as quais destacam-se Gnu Octave – uma saída de código aberto ao *Matlab* –, Maxima – uma linguagem de programação para computação algébrica e gráficos, bem como sua interface de usuário WxMaxima –, Jupyter Notebooks – uma suíte para edição e visualização on-line de trechos de códigos, bem como o programa de computação estatística R, agregado à sua interface RStudio. Como uma plataforma, *R & RStudio* são amplamente utilizados na pesquisa e ensino-aprendizagem, tanto por instituições públicas como por grandes empresas e *big techs* (Meta, Google, Microsoft, SpaceX).

R E PACOTES GRÁFICOS

R (Team, 2000) é uma linguagem de programação de código aberto desenvolvida como um projeto acadêmico no departamento de Estatística da Universidade de Auckland (NZ) em 1992 (Ihaka, 2017), originada a partir da linguagem *S* dos Laboratórios Bell (Giorgi; Ceraolo; Mercatelli, 2022) e que opera como uma linguagem orientada a objeto por comandos de *prompt*. Cinco anos depois, o *R* foi adquirido por um grupo internacional de desenvolvedores, R Core Team (Team, 2000), hoje responsável por sua manutenção e aprimoramento. Atualmente possui extensibilidade garantida pelos seus quase 22 mil pacotes oficiais compartilhados por usuários para uma infinidade de problemas, desde estatística até composição musical, passando por virtualmente todas as áreas do conhecimento. Como atua por expressões envolvendo sintaxe de código, não é de uso simples, embora sua curva de aprendizado seja considerada suave. Juntos, *R & RStudio* combinam uma plataforma robusta para importação de dados, análise, visualização, criação de relatórios, desenvolvimento de funções, pacotes e projetos, entre outros.

Há mais de uma dezena de pacotes gráficos para o *R*, dentre os quais destacam-se *graphics*, *lattice* e *ggplot2*, os dois primeiros da instalação original do programa. Esses pacotes apresentam abordagens distintas para a visualização de dados. O pacote base *graphics* opera pelo uso de funções sequenciais, tanto para os elementos gráficos, como para análise e cálculos concomitantes aos dados envolvidos. Dessa forma, um produto construído em *graphics* para múltiplas variáveis requer linhas de comando individuais para cada conjunto, o que tende a tornar o código extenso e repetitivo. Contornando isso, há o pacote *lattice*, desenvolvido para implementar o sistema *Trellis* e voltado à produção de gráficos envolvendo dados multivariados.

O *lattice* permite, em uma única linha de comando, a representação multivariada global dos dados ou em painéis, bem com a adição de linhas de tendência e cálculos correlatos. Embora seja mais robusto para a representação de grande volume de dados, em comparação ao sistema *graphics*, opera com sintaxe algorítmica mais elaborada, o que limita sua apropriação ao aprendiz iniciante.

O sistema *ggplot2*, criado em 2007 por Hadley Wickham – cientista-chefe da *Posit*, que mantém e desenvolve o *RStudio* atualmente –, agrega o potencial gráfico de ambos os sistema anteriores, combinando a característica aditiva de linhas de comandos intuitivos (*graphics*) ao potencial analítico para grande volume de dados multivariados (*lattice*). De fato, a percepção visual para um gráfico construído com *ggplot2* tem tido melhor recepção em relação ao uso do pacote base *graphics*, mesmo entre estudantes, como reportado num estudo comparativo entre 29.534 alunos de um curso aberto em Pesquisa Reproduzível (Myint *et al.*, 2020).

O *ggplot2* (Wickham; Sievert, 2009) teve por objetivo implementar a *gramática de gráficos* (Wilkinson, 2011), ideia similar à de aplicativos de imagem (Gimp, Canva), e que aborda a construção de um gráfico por camadas sobrepostas. Essa decomposição semântica define 7 camadas justapostas por sinais de adição ("+"), e engloba:

Sequência de camadas para a gramática de gráficos com "ggplot2":

1. Data: linhas individuais de valores (formato Long);

2. Aesthetics: variáveis visuais – mapeamento de eixos, cor, preenchimento, forma, tamanho...;

3. Geometries: pontos, linhas, barras, colunas, boxplot...;

4. Theme: cores, fontes, formatações, demarcações, legendas...;

5. Coordinates: sistema de coordenadas – focalização e zoom do canvas;

6. Facets: visualização de subconjunto de dados em painéis;

7. Statistics: transformação de dados para plotagem.

GGPLOT2 E FERRAMENTAS CORRELATAS

O pacote *ggplot2* faz parte de um *ecossistema* de análise e visualização de dados denominado *Tidyverse*, cujos recursos são carregados por pacote homônimo. O *tidyverse* agrega, num conjunto de pacotes interconectados, a importação e organização dos dados (*tidying*), sua transformação, visualização, modelagem e comunicação dos resultados. Para uso do *ggplot2*, e nas palavras de seu mantenedor, "você fornece os dados, diz ao ggplot2 como mapear as variáveis para a estética, quais primitivas gráficas usar, e ele cuida dos detalhes"[1] (Wickham *et al*., 2016, tradução nossa). E no espírito pleno e mesmo inconsciente do ensino reprodutível, sua documentação primária pode ser acessada livremente pela internet, por meio de obra periodicamente revisada por seu criador.

Uma vez instalada, a estrutura gramatical da biblioteca permite a produção de gráficos de qualidade de publicação técnico-científica, como ilustrado em diversos sites na rede. Além disso, vários pacotes têm sido contemplados para situações específicas de uso de *ggplot2* em produção científica, tais como *ggpubfigs* (Steenwyk; Rokas, 2021) para a extensão de paletas de cor, *ggmap* para a visualização espacial, *ggfortify* para a visualização de modelos estatísticos, *GGally* para

1 No original: You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

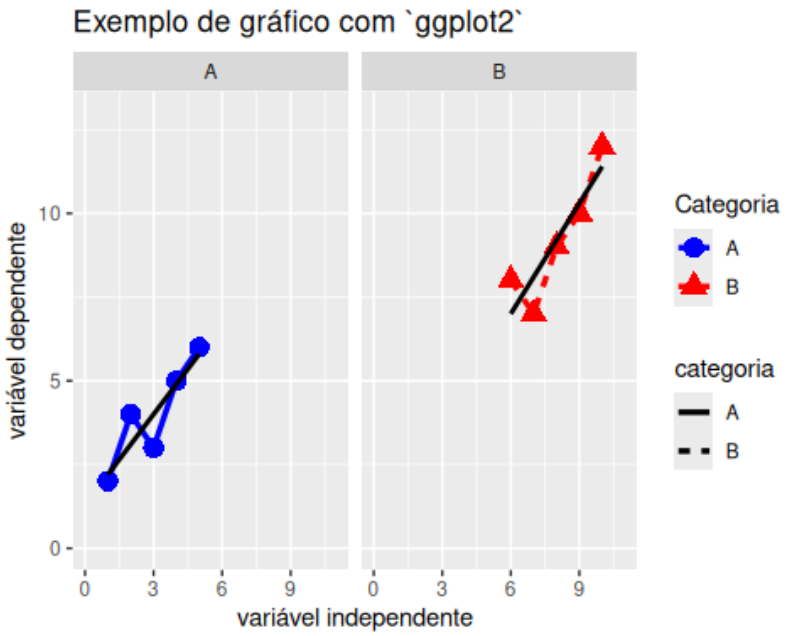
a visualização estatística complementar de gráficos, como também *ggbio* e *ggtree*, ambos do repositório *Biocondutor*, para a visualização de dados biológicos (Tyner; Briatte; Hofmann, 2017).

Por outro lado, ainda que *ggplot2* seja intuitivo e com uma curva de aprendizagem relativamente suave, a simples observação de que depende de uma sintaxe de linguagem de programação é suficiente para afugentar iniciantes com experiências plenas nos aplicativos de cliques de mouse, tais como os elencados na Seção 2. Para ilustrar essa situação, segue um trecho de código reproduzível em ambientes R e seu produto de saída contendo as 7 camadas da *gramática de gráficos* de Wilkinson (2011).

```
library(ggplot2)
dados <- data.frame( x = 1:10, y = c(2, 4, 3, 5, 6, 8, 7, 9,
10, 12), categoria = factor(rep(c("A", "B"), each = 5))) #
dados simulados
ggplot(dados, aes(x = x, y = y, color = categoria, shape =
categoria)) + # camada de mapeamento estético dos dados
geom_line(aes(linetype = categoria), size = 1.2) +
geom_point(size = 4) +
geom_smooth(method = "lm", se = FALSE, color =
"black") +
# camada geométrica com linhas, pontos e curva de ten-
dência scale_color_manual(values = c("A" = "blue", "B"
= "red")) +
scale_shape_manual(values = c("A" = 16, "B" = 17)) +
# camada de cores e formas personalizadas coord_carte-
sian(xlim = c(0, 11), ylim = c(0, 13)) +
# camada de coordenadas
facet_wrap(~ categoria, ncol = 2) +
# camadas de painel (gráficos por categorias) +
theme_grey() +
```


```
labs(title = "Exemplo de gráfico com 'ggplot2'", x = "va-
riável independente",
y = "variável dependente", color = "Categoria",
shape = "Categoria") # camada de temas e rótulos
```

Figura 1 – Um exemplo de gráfico produzido com *ggplot2* apresentando as 7 camadas da gramática de gráficos



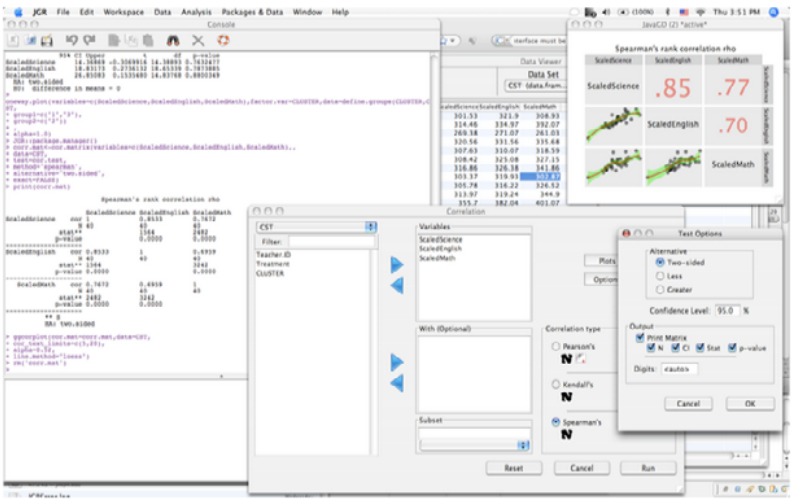
Fonte: elaborado pelo autor com base em Wilkinson, 2011.

INTERFACES GRÁFICAS PARA GGPLOT2

Pelo trecho de código apresentado para o gráfico da Figura 1, fica evidente uma aculturação parcimoniosa para apropriação da sintaxe

desejada à produção gráfica com *ggplot2*. Contornando esse problema, surgiram algumas poucas interfaces gráficas no intuito de facilitar a visualização de dados com cliques de mouse agregados (ou não) aos códigos-fonte dos gráficos. Uma das primeiras concebidas envolveu o pacote Deducer (Fellows, 2012), uma implementação de interface baseada em *Java* para construção gráfica (*plot builder*) em *ggplot2* (Figura 2).

Figura 2 – A interface gráfica *Deducer* para *ggplot2*

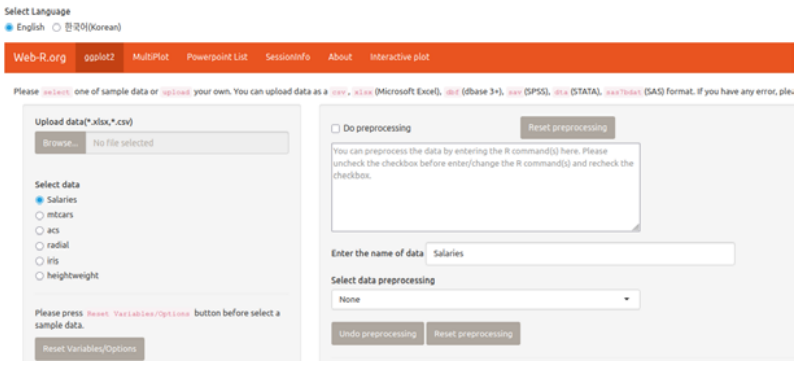


Fonte: Deducer, 2022.

Embora *Deducer* contemple a interface gráfica como um pacote do próprio R, outras *GUIs* para uso remoto surgiram para lubrificar o caminho de produção com *ggplot2*. Um exemplo é Learn ggplot2 (Moon, 2017), um aplicativo web desenvolvido em 2016 com o *framework shiny*, também um pacote do *R* (Figura 3).

SUMÁRIO

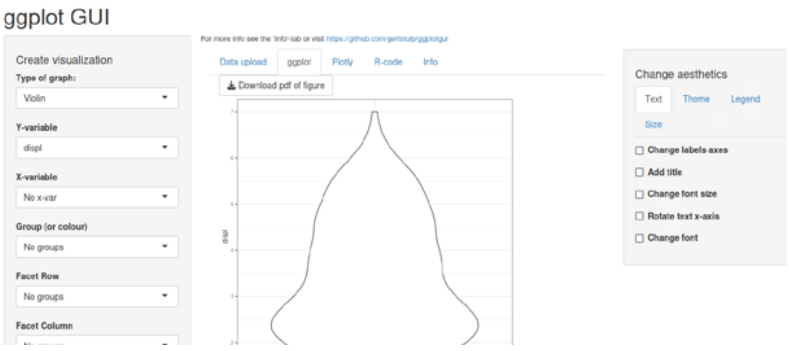
Figura 3 – Tela inicial do aplicativo web *Learn ggplot2*



Fonte: Moon, 2017.

Pela facilidade de implementação para visualização de dados com uso de shiny, outras iniciativas foram posteriormente contempladas. Um exemplo é PlotsOfData (Postma; Goedhart, 2019), um aplicativo web baseado em *shiny* e que permite a inserção e sumarização de dados, além da construção de gráficos com a biblioteca (Figura 4).

Figura 4 – Exemplo de tela do aplicativo web PlotsOfData para gráficos *ggplot2* em ambiente *shiny*



Fonte: Postma; Goedhart, 2019.

SUMÁRIO

Outro exemplo de aplicativo web, e que também faz uso do *framework shiny*, é o GGPlot GUI, cuja tela inicial é apresentada na Figura 5.

Figura 5 – GGPlot GUI, outra iniciativa de cliques de mouse para facilitar o uso de *ggplot2*

Fonte: GGPlot Gui, 2025.

Ainda que essas iniciativas sejam sedutoras por possibilitarem a construção de um gráfico *ggplot2* de natureza complexa no que tange à lógica de programação envolvida, e sob as expensas de cliques de mouse para inserção/seleção de dados, bem como do universo de elementos gráficos constituídos pelo pacote, há um viés envolvido no processo de ensino e aprendizagem: não são passíveis de reprodutibilidade, tal como auferida em Ciência e Educação Abertas. Isso decorre do fato de que cliques de mouse, salvo a produção e compartilhamento de vídeo detalhado sobre cada gráfico construído, não guardam memória, uma vez que não operam por linhas de comando de texto.

E essa é uma das principais características vantajosas do uso de linhas de comando sobre cliques de mouse, visto que permitem

o compartilhamento facilitado de ideias e registros juntamente ao código-fonte comentado, visando à compilação de um produto específico num simples editor de texto. Assim, estabelece-se um impasse técnico-didático: se por um lado o uso do mouse pode facilitar (e muito) a construção de gráficos, tal como ocorre com planilhas eletrônicas (por exemplo, Microsoft Excel, Libreoffice Calc), por outro a sintaxe de programação é plena em reproduzir uma sequência de comandos indubitáveis à reprodução e adaptação do objeto pretendido.

E indo um pouco mais além, com o advento de Markdown, uma linguagem de marcação de texto para formatação web (*HTML*), plataformas como *R & RStudio* permitem implementar a *programação letrada* de Knuth (Knuth, 1984), com texto estilizado e gráficos de *ggplot2* num mesmo texto de literatura científica (Cockett *et al.*, 2024; Rimal, 2021). Exemplo disso constitui o uso do pacote quarto para a criação de artigos técnico-científicos em *Rmarkdown*, tal como detalhado em Bauer e Landesvatter (2023) e ilustrado na redação deste capítulo.

SCRIPTRPLOT

No meio termo entre os *recursos educacionais abertos* (Adedoyin *et al.*, 2025) para a produção de gráficos com *ggplot2* por "*ambientes clicáveis*" de agradabilidade garantida e a obrigatória inserção de linhas de comando plenamente reproduzíveis, adaptáveis e compartilháveis, oferece-se neste capítulo o *ScriptRplot*. O objeto, uma corruptela para *script* com *R* para a produção de *plots*, permite a produção de gráficos de *ggplot2* por comandos de texto. Diferente da sintaxe convencional de programação de *R* e do sistema gráfico do pacote, o *ScriptRplot* oferece ao usuário os ajustes necessários à plotagem em linguagem natural. Não obstante, também fornece a linha completa de comandos para a compilação do gráfico pretendido.

O código-fonte para o *ScriptRplot*, bem como algumas instruções de uso e exemplos ilustrativos, encontra-se disponível no site Bioquanti (Schneedorf, 2023). O website foi idealizado inicialmente para cobrir diversos temas em *Bioquímica Quantitativa* sob pressupostos de ensino reprodutível e pelo uso de três softwares de distribuição livre, incluindo a plataforma *R & RStudio*. O site em atualização conta também com materiais voltados às áreas do ensino básico, envolvendo Matemática, Ciências da Natureza, Ciências Humanas e Linguagens, e com um aplicativo portável para estudo de gráficos em *JavaScript* e *JSPlotly*.

CAMADAS E ESTRUTURA PARA OS ELEMENTOS GRÁFICOS

O *ScriptRPlot* foi concebido itemizando as 7 camadas propostas na *gramática de gráficos*, mas dispersas numa sequência autoral itemizada para a confecção do gráfico dada abaixo:

1. Data load - carregamento de dados;
2. Reshape – conversão facultativa de dados de formato Wide para Long exigido pelo pacote;
3. Preamble – transformações prévias no conjuto de dados (atualmente, somente uma conversão facultativa de variável numérica em categórica);
4. Dataset – atribuição de variáveis dependente e independente;
5. Grouping – atribuição de variável de grupo;
6. Error bar – barra de erros e formatação;
7. Symbol – tipo e atribuições para símbolos;
8. Line – tipo e atribuições para linhas;
9. Curve – linhas matemáticas de tendência e formatação;

10. Other plots – atribuições facultativas para barras, histogramas, gráfico de matriz de dados, simulação de curvas matemáticas, e apresentação de paineis;

11. Axis –atribuição dos eixos (primário e secundário);

12. Linhas – adição e formatação de linhas verticais e horizontais;

13. Zoom – nível de cobertura dos eixos do gráfico;

14. Additional colour scale – paletas adicionais de cores;

15. Additional expression – expressões aditivas facultativas;

16. Legend – formatação de legendas;

17. Caption – formatação de rótulos (títulos, subtítulos);

18. Text – inserção e posicionamento de texto.

Para cada item elencado acima, o *ScriptRplot* oferece um rápido comentário sobre as possibilidades de inserção, tal como representado na Figura 6.

Figura 6 – Um excerto de alternativas possíveis para edição de um componente gráfico presente no *ScriptRplot*

```
26
27  ########## PREAMBULE #########
28  tmp=as.factor(tmp) # se necessário, converte uma variável numérica em discreta
29
30  #############  DATASET  #############
31  x= DA         # variável independente
32  y= Fluor      # variável dependente
33  error= NA # variável error; NA
34  group= tmp # group: NA; NULL # útil quando só houver uma variável discreta
35  mapping = aes(x = DA, y = Fluor) # aes(x=x,y=y); não pode ter x=x e x=y pra subconjunto (er
    geom_smooth); se boxplot, bar ou col - aes(x=grupo, y = sinal); se bar ou histogram para
    somatórias (count), apenas x ou y
36  theme_plot=theme_grey() # grey,bw,minimal,linedraw,classic,dark,void,light
37
38  ##################  GROUPING  ###############
39  # Main geoms, unique for overall data our separated by groups
40  allpoints = geom_blank # blank; point; col; bar; line (será adicional/alternativo ao line
    interno); smooth (também adicional); path (útil pra cv, pois conecta pontos na ordem em que
    aparecem); count; boxplot; smooth; path; area; polygon; rug; step; tile; violin; pointrange
```

Fonte: elaborado pelo autor, 2025.

Dessa forma, é possível experimentar variações diversas para a plotagem, ao mesmo tempo em que se permite uma apropriação paulatina e até mesmo lúdica da sintaxe exigida pelo *ggplot2*, embora essa não seja necessária à compilação do gráfico, bastando-se apenas a inserção sequencial e facultativa de seus elementos. Uma vez configurado textualmente os elementos gráficos, o *ScriptRplot* segue convertendo as variáveis do usuário em variáveis para interpretação do compilador do *R*. As variáveis convertidas a partir dos componentes gráficos são apresentadas na Figura 7.

Figura 7 – Quadro de componentes gráficos para o *ScriptRplot* junto à gramática de *gglot2*

Elemento	Objeto	Comando
Mapping	main	ggplot(data=dataset, mapping)
Título e Legenda	titles	theme(plot.title = ..., plot.caption = ...)
Rótulos	labels	labs(x=..., y=..., title=..., ...)
Barras de erro	error_bar	geom_errorbar(... aes(ymin=y-error_down, ymax=y+error_up))
Legenda	plot_legend	theme(legend.title=..., legend.position=..., ...)
Guias	legend_guide	guides(colour=..., size=..., ...)
Anotação de texto	annotation	annotate("text", x=..., y=..., label=..., parse=TRUE)
Valores dos eixos	axis_values	theme(axis.text.x=..., axis.text.y=...)
Rótulos dos eixos	axis_labels	theme(axis.title.x=..., axis.title.y=...)
Zoom	axis_lim	coord_cartesian(xlim=c(x_min,x_max), ylim=c(y_min,y_max))
Linha horizontal	line_add_horiz	geom_hline(yintercept=..., ...)
Linha vertical	line_add_vert	geom_vline(xintercept=..., ...)
Linha inclinada	line_add_incl	geom_abline(intercept=..., slope=..., ...)
Tipo 1 - Todos os pontos	type_allpoints	allpoints(shape=..., colour=..., ...)
Tipo 2 - Agrupamento	type_group	grouping(stroke=..., aes(shape=..., ...))
Resumo estatístico	summary_stats	stat_summary(fun.data=..., fun=..., ...)
Linha 1 - Todos os pontos	line_all	geom_line(size=..., alpha=..., ...)
Linha 2 - Agrupamento	line_group	geom_line(size=..., aes(...))
Curva 1 - Todos os pontos	curve_all	geom_smooth(data=subset(...), method=..., aes(...))
Curva 2 - Agrupamento	curve_group	geom_smooth(data=subset(...), method=..., aes(...))
Barra/Histograma	bar_hist	hist_bar(stat=..., bins=..., aes(...))
Simulação	simul	simulation(fun=..., geom=..., ...)
Comando Principal	p_main	main + theme_plot + ... + bar_hist + ...
Complementos	p	p_main + axis_inversion + ... + simul + legend_inversion

Fonte: elaborado pelo autor, 2025.

A partir das atribuições acima, o usuário define as condições necessárias para a construção do gráfico, embora desprovido da obrigatoriedade do conhecimento de sua sintaxe. As variáveis permitem a construção de até dois gráficos sobrepostos, incluindo atribuições individuais para tipos, símbolos, linhas e curvas de tendência. Ainda que passível de uma discussão acerca de seu mérito, isso permite a visualização simultânea para os dados globais do *dataset* junto a grupos separados de suas variáveis, bem como a apreciação simultânea de dois tipos gráficos sobrepostos, como dispersão de pontos junto à *boxplot*, habilidade inexistente nos programas concorrentes listados neste texto.

Entre as curvas de tendência, o *script* utiliza bibliotecas da instalação original para ajuste linear, curva de suavização local (*lowess - locally weighted scatterplot smoothing*), e ajuste não-linear aos dados, esse a partir de funções arbitrárias inseridas pelo usuário.

DIAGRAMA DE PLOTAGEM

Sucedendo-se a estrutura para configuração dos elementos gráficos, o *script* reúne as informações num *diagrama de plotagem*, o qual estrutura a sequência para a compilação do código. Uma descrição sucinta do diagrama está apresentada abaixo.

DIAGRAMA DE PLOTAGEM PARA O SCRIPTRPLOT:

```
# Mapping
main=ggplot(data=dataset,mapping)

# Title & Caption
titles=theme(plot.title = element_text(face = “bold”,hjust=0.5))+theme(plot.caption = element_text(hjust = figure_caption_x_position))
```


```
# Labels
labels=labs(x=axis_label_x,y=axis_label_y,title=figure_
title,subtitle=figure_subtitle,caption=figure_caption)

# Error bars
error_bar=geom_errorbar(size=error_thick,colour=er-
ror_colour,  alpha=error_alpha,width=error_width,aes(y-
min=y-error_down, ymax=y+error_up))

# Legend
plot_legend=theme(legend.title  =  element_blank(),le-
gend.position = legend_position,legend.background = ele-
ment_rect(linetype = 1, size = 0.2, colour = 1),legend.text
= element_text(size=legend_fontsize))

# Guides
legend_guide=guides(colour=legend_colour,size=le-
gend_size,linetype=legend_line,shape=legend_shape,-
fill=legend_fill, alpha = legend_alpha)

# Text
annotation=annotate("text", x = annotation_x,y=annota-
tion_y,label=annotation_text,parse=TRUE)

# Axis values
axis_values= theme(axis.text.x =element_text(size=axis_
values_x_fontsize,angle  =  axis_values_x_angle))+the-
me(axis.text.y =element_text(size=axis_values_x_fontsi-
ze,angle = axis_values_y_angle))

# Axis label values
axis_labels=theme(axis.title.x = element_text(size = axis_
label_x_fontsize))+theme(axis.title.y = element_text(size
= axis_label_y_fontsize))

# Zoom
axis_lim=coord_cartesian(xlim=c(x_min,x_max),ylim=-
c(y_min,y_max))
```


```
# Added lines (horizontal, vertical, sloping)
line_add_horiz=geom_hline(yintercept = line_horiz,linetype=line_type,size=line_add_size,colour=line_colour,alpha=line_alpha) line_add_vert=geom_vline(xintercept = line_vert,linetype=line_type,size=line_add_size,colour=line_colour,alpha=line_alpha) line_add_incl=geom_abline(intercept=line_incl_intercept,slope=line_incl_slope,linetype=line_type,size=line_add_size,colour=line_colour,alpha=line_alpha)

# 1st Type – Allpoints
type_allpoints=allpoints(shape=symbol_shape_all,colour=symbol_colour_all,size=symbol_size,fill=symbol_fill_all,stroke=symbol_stroke,alpha=symbol_alpha)

# 2nd Type – Grouping
type_group=grouping(stroke=symbol_stroke,stat=symbol_stat,position=symbol_position,aes(shape=symbol_shape_group,colour=symbol_colour_group,size=symbol_size,fill=symbol_fill_group,alpha=symbol_alpha))

# Statistical Summary
summary_stats = stat_summary(fun.data = stat_fun_dat, fun = stat_fun, geom = stat_geom, colour = stat_colour, fill = stat_fill, size = stat_size)

# 1st Line – Allpoints
line_all=geom_line(size=line_size,alpha=line_alpha,linetype=line_type_all,color=line_colour_all,na.rm=TRUE)

# 2nd Line – Grouping
line_group=geom_line(size=line_size,alpha=line_alpha,-na.rm=TRUE, aes(linetype=line_type_group,color=line_colour_group))
```


```
# 1st Curve – Allpoints
curve_all=geom_smooth(data=subset(dataset,curve_subset), method=curve_method,formula = curve_formula, method.args = list(start = curve_nonlinear_seeds),se=curve_se,span=curve_span,size=curve_size, colour=curve_colour_all,linetype=curve_type_all,na.rm=TRUE, n = curve_npoints, aes(weight = curve_weight))

# 2nd Curve – Grouping
curve_group=geom_smooth(data=subset(dataset,curve_subset),na.rm=TRUE, method=curve_method,formula = curve_formula,method.args = list(start = curve_nonlinear_seeds),se=curve_se,span=curve_span,size=curve_size, n = curve_npoints, aes(weight = curve_weight,colour=curve_colour_group,linetype=curve_type_group))

# Bar/Histogram
bar_hist=hist_bar(stat=count_stat,bins=count_bins,binwidth=count_binwidth,position=count_position,aes(colour=symbol_colour_group,fill=symbol_fill_group))

# Simulation
simul = simulation(fun=simul_fun, geom = simul_geom, colour = simul_colour, size = simul_size, n= simul_n,xlim=c(simul_x_min,simul_x_max))
```

LINHA DE COMANDO FINAL

Observa-se, pela sequência individual acima, que os elementos gráficos inseridos pelo usuário em linguagem natural são passados para variáveis que estabelecem a linha lógica de construção das camadas gráficas de *ggplot2*. Uma vez mais, contudo, o usuário

pode observar e aprender com a sintaxe original para cada função do pacote. Por fim, cada variável convertida é agrupada numa única linha de comando aditiva para as diversas configurações do gráfico. Essa linha está representada abaixo.

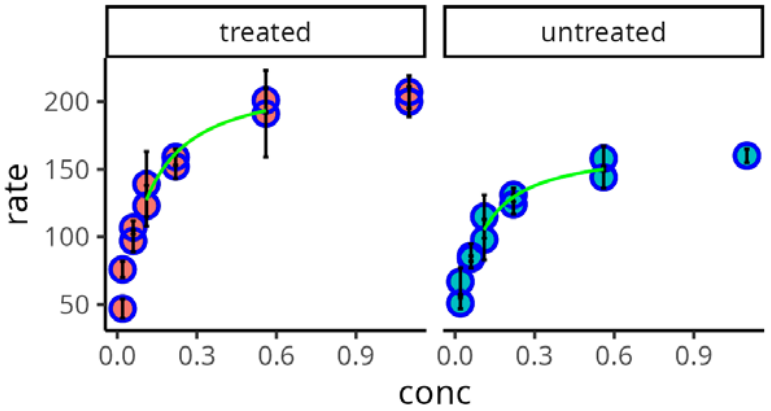
LINHA DE COMANDO FINAL PARA O SCRIPTRPLOT:

```
main+theme_plot+titles+labels+error_bar+plot_legend+annotation+axis_values+axis_labels+axis_lim+faceting+type_allpoints+type_group+line_all+line_group+curve_all+curve_group+symbol_jitter+bar_hist+line_add_horiz+line_add_vert+line_add_incl+axis_inversion+axis_transf_x+axis_transf_y+legend_guide+add_ggplot_expression+add_colour_scale+secondary_axis+summary_stats+simul+legend_inversion.
```

Dessa forma, possibilita-se, junto ao *ScriptRplot*, a apreciação de *32 conjuntos de informações* facultativamente combinados em um gráfico, cada um com suas formatações específicas. Não obstante, o conjunto aparentemente significativo de elementos, o *script*, ainda está muito aquém de todas as possibilidades de construção com o sistema *ggplot2*. Em contrapartida, a execução do gráfico como um todo depende somente da inserção de seus elementos iniciais, tais como o *dataset*, o tipo de gráficos desejados, de símbolos, de linhas e as cores, provendo o *script* com a sequência necessária à visualização final do gráfico. Uma vez inseridas essas informações, basta executar o *ScriptRplot* como qualquer outro *script* do *R*. Exemplificando tal simplicidade, por seleção do texto (Ctrl+A), seguido da compilação do *script* (Ctrl+Enter).

Um exemplo de gráfico elaborado com o ScriptRplot, visualizado na página correlata do website Bioquanti, é ilustrado na Figura 8.

Figura 8 – Saída gráfica do *ScriptRplot* (versão 1.0) para os dados atividade enzimática sob ação do antibiótico puromicina[2]



Fonte: elaborado pelo autor, 2025.

UMA PALAVRA SOBRE INICIATIVAS PÚBLICAS EM LITERACIA DIGITAL E EDUCAÇÃO

O *ScriptRplot* de que trata este capítulo pretende coadunar com algumas das expectativas da nova Educação 4.0 e 5.0 (Ahmad *et al.*, 2023), fac-símiles transpostas da Indústria 4.0 e 5.0 (Leng *et al.*, 2022) e distintas apenas na aplicação das últimas a valores sociais e da comunidade global. Especificamente, o aprendizado

2 Observe que o gráfico apresenta a inserção de 1) dados multivariados, com 2) símbolos e 3) espessura definida, 4) tipos distintos, 5) preenchimento e 6) cores distintas para cada subconjunto, 7) barras de erro, 8) disposição em painel, 9) configuração e tema específicos e 10) ajuste não linear em subconjuntos para cada nível dos dados e por equação introduzida pelo usuário.

personalizado, imersivo e autônomo para ferramentas de literacia digital e programação aplicadas ao cotidiano, tal como pretendido pela Educação 2030: O Futuro da Educação e das Competências (OECD, 2015). A proposta instituída pela OECD (Organização para a Cooperação e Desenvolvimento Econômico) em 2015, prevê a composição de uma matriz conceitual de aprendizagem para as novas competências (conhecimentos, habilidades, atitudes e valores) exigidas ao século XXI. Em terras brasilianas, também a Lei 14.533 (Brasil, 2023), que instituiu a Política Nacional de Educação Digital (PNED).

Pela PNED, prevê-se a aplicação de recursos estatais que visem eixos como de Inclusão Digital, contando com estratégias para "treinamento de competências digitais", bem como o "desenvolvimento e acesso a plataformas e repositórios de recursos digitais". Em paralelo, eixo de Educação Digital Escolar, o qual tem por objeto "garantir a inserção da educação digital nos ambientes escolares, em todos os níveis e modalidades, a partir do estímulo ao letramento digital e informacional e à aprendizagem de computação, de programação, de robótica e de outras competências digitais" (Brasil, 2023, n.p.).

Com a recente chancela do governo federal frente às diretrizes do *Novo Ensino Médio* pelo Conselho Nacional de Educação, destaca-se também a formação técnica e integração tecnológica para uso de ferramentas digitais e tecnologias emergentes, como programação e softwares educacionais. Parte dessa iniciativa também está espelhada na recente Cúpula de Líderes do G20, reunida no Rio de Janeiro em novembro de 2024, que articula, em seu Relatório do G20 sobre Educação, a integração tecnológica nas práticas educacionais e o compartilhamento de conteúdo educacional digital (Group of Twenty, 2024).

A ELABORAÇÃO DESTE CAPÍTULO

A formatação de texto (*RMarkdown*), a inserção e a referenciação cruzada de figuras e do diagrama em tabela (pacote kable, versão 1.4.0), dos hiperlinks, dos trechos de código renderizáveis para gráficos *ggplot2* (versão 3.4.4), bem como o gerenciamento bibliográfico e a conversão do *documento dinâmico* final para compilação a um modelo de arquivo *DOCX* previamente configurado, foram exclusivamente conduzidos com a linguagem de programação *R* (versão 4.3.3, fev/2024) em ambiente de desenvolvimento integrado *RStudio* (versão 2024.09.1 Build 394), e pacote *quarto* (versão 1.4.4) como sistema de publicação científica.

AGRADECIMENTOS

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior (CAPES) – Código de Financiamento 001.

REFERÊNCIAS

ADEDOYIN, O. B. *et al.* Open educational resources: Qualitative evaluation of faculty interest in motivating students to create and use. **Innovations in Education and Teaching International**, [*s. l.*], v. 62, n. 1, p. 216-232, 2025.

AHMAD, S. *et al.* Education 5.0: requirements, enabling technologies, and future directions. **arXiv preprint arXiv:2307.15846**, [*s. l.*], 2023.

BAUER, P. C.; LANDESVATTER, C. **Writing a reproducible paper with RStudio and quarto**. [*s. l.*], 2023.

SUMÁRIO

BEAN, B. L. Teaching Reproducibility to First Year College Students: Reflections From an Introductory Data Science Course. **Journal on Empowering Teaching Excellence**, [*s. l.*], v. 7, n. 2, p. 5, 2023.

BRASIL. **Lei n. 14.533, de 11 de janeiro de 2023**. Institui a Política Nacional de Educação Digital e altera as Leis nºs 9.394, de 20 de dezembro de 1996 (Lei de Diretrizes e Bases da Educação Nacional), 9.448, de 14 de março de 1997, 10.260, de 12 de julho de 2001, e 10.753, de 30 de outubro de 2003. Brasília, DF: Presidência da República, 2023. Disponível em: https://www.planalto.gov.br/ccivil_03/_Ato2023-2026/2023/Lei/L14533.htm. Acesso em: 17 maio 2025.

CLAERBOUT, J. F.; KARRENBACH, M. Electronic documents give reproducible research a new meaning. *In*: **SEG TECHNICAL PROGRAM EXPANDED ABSTRACTS 1992**. [*S. l.*]: Society of Exploration Geophysicists, 1992. p. 601-604.

COCKETT, R. *et al.* Continuous Tools for Scientific Publishing. **scipy**, [*s. l.*], p. 121-136, 2024.

DEDUCERManual. Deducer, 2022. Disponível em: https://www.deducer.org/pmwiki.php?n=Main.DeducerManual?from=Main.HomePage. Acesso em: 22 mar. 2025.

DOGUCU, M. Reproducibility in the Classroom. **Annual Review of Statistics and Its Application**, [*s. l.*], v. 12, 2024.

FELLOWS, I. Deducer: a data analysis GUI for R. **Journal of statistical Software**, [*s. l.*], vol. 49, p. 1–15, 2012.

GIORGI, F. M.; CERAOLO, C.; MERCATELLI, D. The R language: an engine for bioinformatics and data science. **Life**, [*s. l.*], v. 12, n. 5, p. 648, 2022.

GROUP OF TWENTY. **G20 Rio de Janeiro Leaders' Declaration**. G20, 2024. Disponível em: https://g20.org/wp-content/uploads/2024/11/G20-Rio-de-Janeiro-Leaders-Declaration-EN.pdf. Acesso em: 22 mar. 2025.

HERNANDEZ, J. A.; COLOM, M. Reproducible research policies and software/data management in scientific computing journals: a survey, discussion, and perspectives. **Frontiers in Computer Science**, [*s. l.*], v. 6, p. 1491823, 2025.

IHAKA, R. The r project: A brief history and thoughts about the future. **Univ. Auckl**, [*s. l.*], v. 4, p. 22, 2017.

SUMÁRIO

KNUTH, D. E. Literate Programming. **Comput. J.**, v. 27, n. 2, p. 97-111, 1984. DOI: https://doi.org/10.1093/comjnl/27.2.97. Disponível em: https://academic.oup.com/comjnl/article-abstract/27/2/97/343244. Acesso em: 22 mar. 2025.

LENG, J. *et al.* Industry 5.0: Prospect and retrospect. **Journal of Manufacturing Systems**, [*s. l.*], v. 65, p. 279-295, 2022.

MOON, K.-W. **Learn *ggplot2* using shiny app**. [*S. l.*]: Springer, 2017.

MYINT, L. *et al.* Comparison of beginning R students' perceptions of peer-made plots created in two plotting systems: a randomized experiment. **Journal of Statistics Education**, [*s. l.*], v. 28, n. 1, p. 98-108, 2020.

OECD. **Future of Education and Skills 2030/2040**. Organisation for Economic Co-operation and Development, 2015. Disponível em: https://www.oecd.org/en/about/projects/future-of-education-and-skills-2030.html. Acesso em: 17 maio 2025.

POSTMA, M.; GOEDHART, J. PlotsOfData—A web app for visualizing data together with their summaries. **PLoS biology**, [*s. l.*], v. 17, n. 3, 2019.

RIMAL, Y. Reproducible Academic Writing and Interactive Data Visualization Using R Markdown (R Programming Flex-Dashboard: Flex_Dashboard Packages). *In*: **Rising Threats in Expert Applications and Solutions**: Proceedings of FICR-TEAS 2020. [*S. l.*]: Springer, 2021. p. 603-615.

SCHNEEDORF, J. M. Bioquanti, um website interativo para ensino-aprendizagem em Bioquímica. **Revista de Ensino de Bioquímica**, [*s. l.*], v. 21, n. 1, p. 110-124, 2023.

STEENWYK, J. L.; ROKAS, A. ggpubfigs: colorblind-friendly color palettes and *ggplot2* graphic system extensions for publication-quality scientific figures. **Microbiology Resource Announcements**, [*s. l.*], v. 10, n. 44, p. 10-1128, 2021.

TEAM, R. C. R language definition. **Vienna, Austria: R foundation for statistical computing**, [*s. l.*], v. 3, n. 1, p. 116, 2000.

TYNER, S. C.; BRIATTE, F.; HOFMANN, H. Network Visualization with *ggplot2*. **The R Journal**, [*s. l.*], 2017.

WICKHAM, H.; SIEVERT, C. ***ggplot2*: elegant graphics for data analysis**. [*S. l.*]: springer New York, 2009. vol. 10

WICKHAM, H. *et al*. **Overview**. ggplot2, 2016. Disponível em: https://ggplot2.tidyverse.org/. Acesso em: 17 maio 2025.

WILKINSON, L. The grammar of graphics. *In*: **Handbook of computational statistics**: concepts and methods. [*S. l.*]: Springer, 2011. p. 375-414.