

Visualização de dados com ggplot2

Prof. Walmes Zeviani

walmes@ufpr.br

Laboratório de Estatística e Geoinformação

Departamento de Estatística

Universidade Federal do Paraná

Recursos gráficos do R

Visão geral

O **landscape** de recursos para visualização de dados no R pode ser representado por uma divisão em 4 territórios.

1. O pacote `graphics` e derivados.
2. O pacote `lattice` e derivados.
3. O pacote `ggplot2` e derivados.
4. Pacotes para gráficos interativos.

O pacote `graphics`

- ▶ Contém os recursos mais primitivos: com funções de alto e baixo nível.
- ▶ Pouco suporte para mapeamento em variáveis visuais retinais.
- ▶ Vários pacotes que complementam suas funcionalidades: `plotrix` e `gplots`.
- ▶ Usado em métodos gráficos de saídas de análises: dendrogramas, biplots, etc.

O pacote lattice

- ▶ Desenvolvido por Deepayan Sarkar.
- ▶ Plotagem multi-painel e mapeamento em variáveis visuais retinais.
- ▶ Já vem com a instalação básica do R.
- ▶ Também é utilizado na implementação de métodos gráficos.

O pacote `ggplot2`

- ▶ Desenvolvido por Hadley Wickham.
- ▶ Baseado na *Grammar of Graphics*.
- ▶ Plotagem multi-painel e mapeamento em variáveis visuais retinais.
- ▶ A importância da `ggplot2` está no modelo mental mais claro.

Pacotes para recursos interativos

A visualização interativa é voltada para exibição na WEB. Alguns dos pacotes para isso são estes:

- ▶ `plotly`.
- ▶ `highcharter`.
- ▶ `googleVis`.
- ▶ `rCharts`.
- ▶ `leaflet`.
- ▶ `iplots`.
- ▶ `rgl`.
- ▶ `animation`.

Mais informação

Para mais detalhes sobre os recursos gráficos, siga esse link: https://www.stat.ubc.ca/~jenny/STAT545A/block90_baseLatticeGgplot2.html.

Para uma descrição completada comparação entre `lattice` e `ggplot2`, siga esse link: <https://learnr.wordpress.com/2009/08/26/ggplot2-version-of-figures-in-lattice-multivariate-data-visualization-with-r-final-p/>

Confira no **R Graph Gallery** a variedade de gráficos confeccionados com o R.

Galerias e tutoriais

The R Graph Gallery

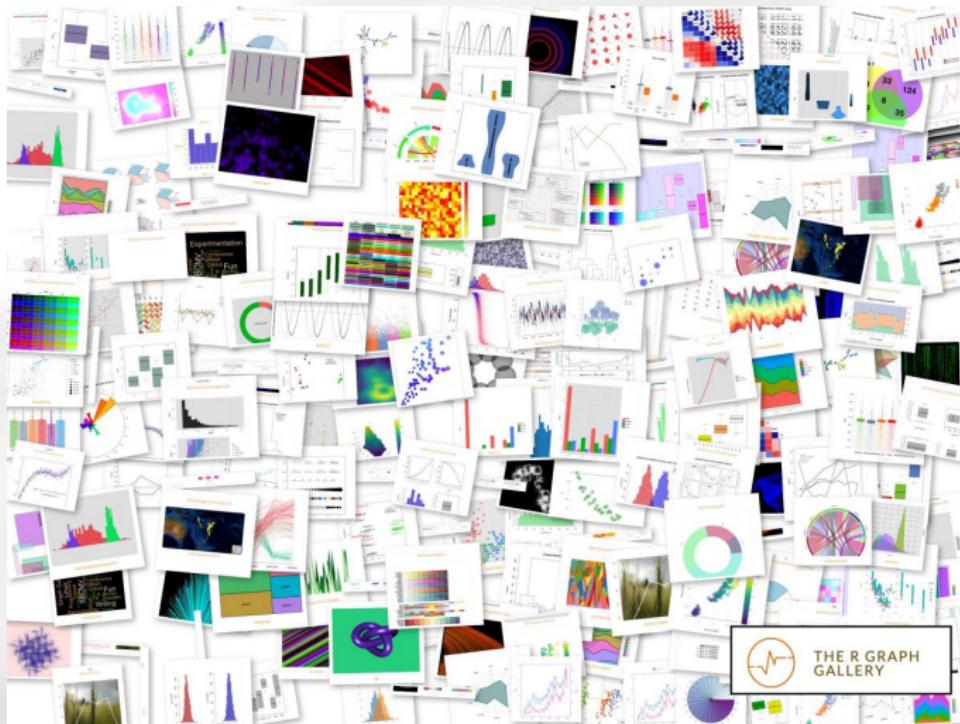


Figura 1. Gráficos da galeria de gráficos do R. Fonte: Mara Averick.

Galerias

- ▶ [https://www.r-graph-gallery.com/portfolio/ggplot2-package/.](https://www.r-graph-gallery.com/portfolio/ggplot2-package/)
- ▶ [http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html.](http://r-statistics.co/Top50-Ggplot2-Visualizations-MasterList-R-Code.html)
- ▶ [https://r4stats.com/examples/graphics-ggplot2/.](https://r4stats.com/examples/graphics-ggplot2/)
- ▶ [http://girke.bioinformatics.ucr.edu/GEN242/pages/mydoc/Rgraphics.html.](http://girke.bioinformatics.ucr.edu/GEN242/pages/mydoc/Rgraphics.html)

Extensões do ggplot2

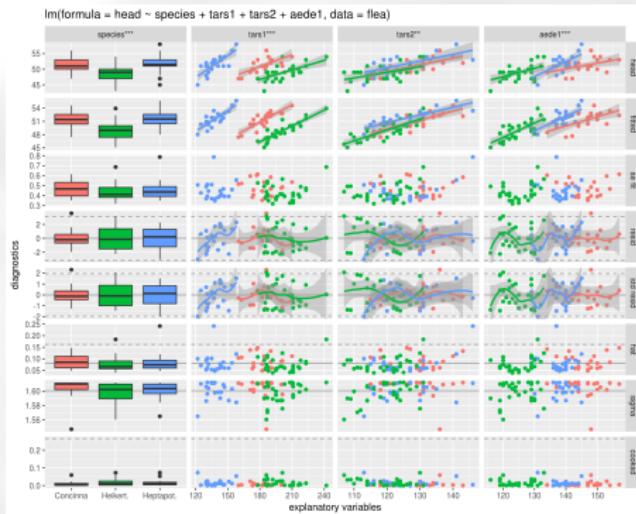


Figura 2. Gráficos da galeria de extensões do ggplot2. Fonte: GGally.

- ▶ <http://www.ggplot2-exts.org/gallery/>.
- ▶ <https://mode.com/blog/r-ggplot-extension-packages>.

Mapas

- ▶ https://rstudio-pubs-static.s3.amazonaws.com/176768_ec7fb4801e3a4772886d61e65885fbdd.html.
- ▶ <https://www.curso-r.com/blog/2017-05-04-mapas-tematicos-3-minutos/>.
- ▶ <http://girke.bioinformatics.ucr.edu/GEN242/pages/mydoc/Rgraphics.html>.

Tutoriais em português

- ▶ [https://rpubs.com/mnunes/ggplot2.](https://rpubs.com/mnunes/ggplot2)
- ▶ [https://analisereal.com/2015/09/19/introducao-ao-ggplot2/.](https://analisereal.com/2015/09/19/introducao-ao-ggplot2/)
- ▶ [https://timogrossenbacher.ch/2016/12/beautiful-thematic-maps-with-ggplot2-only/.](https://timogrossenbacher.ch/2016/12/beautiful-thematic-maps-with-ggplot2-only/)
- ▶ [http://recologia.com.br/tag/graficos/.](http://recologia.com.br/tag/graficos/)
- ▶ [http://rstudio-pubs-static.s3.amazonaws.com/24563_3b7b0a6414824e3b91769a95309380f1.html.](http://rstudio-pubs-static.s3.amazonaws.com/24563_3b7b0a6414824e3b91769a95309380f1.html)
- ▶ [http://eduardogutierrezes.com/inteligencia-geografica-gerando-mapas-em-r/.](http://eduardogutierrezes.com/inteligencia-geografica-gerando-mapas-em-r/)
- ▶ [https://pt.stackoverflow.com/questions/332053/r-mapa-de-cidades-brasileiras.](https://pt.stackoverflow.com/questions/332053/r-mapa-de-cidades-brasileiras)

Um overview do ggplot2

A ficha técnica

`ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics`

A system for 'declaratively' creating graphics, based on "The Grammar of Graphics". You provide the data, tell 'ggplot2' how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details.

Version:	3.1.0
Depends:	R (\geq 3.1)
Imports:	digest , grid , gttable (\geq 0.1.1), lazyeval , MASS , mgcv , plyr (\geq 1.7.1), reshape2 , rlang (\geq 0.2.1), scales (\geq 0.5.0), stats , tibble , viridisLite , withr (\geq 2.0.0)
Suggests:	covr , dplyr , ggplot2movies , hexbin , Hmisc , lattice , mapproj , maps , maptools , multcomp , munsell , nlme , testthat (\geq 0.11.0), yddirr , quantreg , knitr , rgeos , rpart , rmarkdown , sf (\geq 0.3-4), svglite (\geq 1.2.0.9001)
Enhances:	sp
Published:	2018-10-25
Author:	Hadley Wickham [aut, cre], Winston Chang [aut], Lionel Henry [aut], Thomas Lin Pedersen [aut], Kohske Takahashi [aut], Claus Wilke [aut], Kara Woo [aut], RStudio [cph]
Maintainer:	Hadley Wickham <hadley at rstudio.com>
BugReports:	https://github.com/tidyverse/ggplot2/issues
License:	GPL-2 file LICENSE
URL:	http://ggplot2.tidyverse.org , https://github.com/tidyverse/ggplot2
NeedsCompilation:	no
Citation:	ggplot2 citation info
Materials:	README NEWS
In views:	Graphics , Phylogenetics , TeachingStatistics
CRAN checks:	ggplot2 results

Figura 3. Ficha técnica do ggplot2.

Gramática dos gráficos

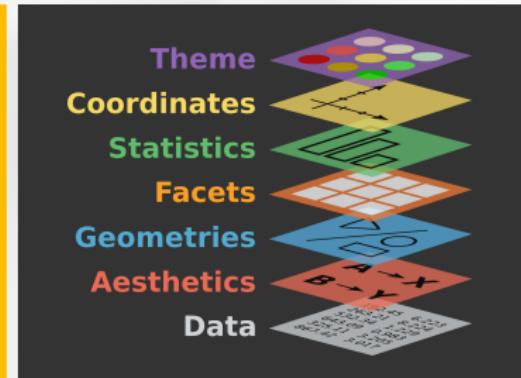
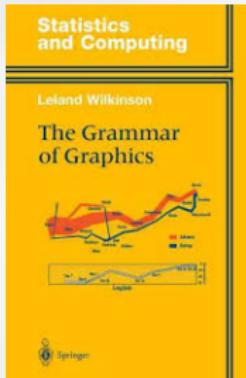


Figura 4. Leland Wilkinson (esq.) autor de “The grammar of graphics” (meio) e as camadas da gramática de gráficos que são usadas no ggplot2.

Gramática dos gráficos

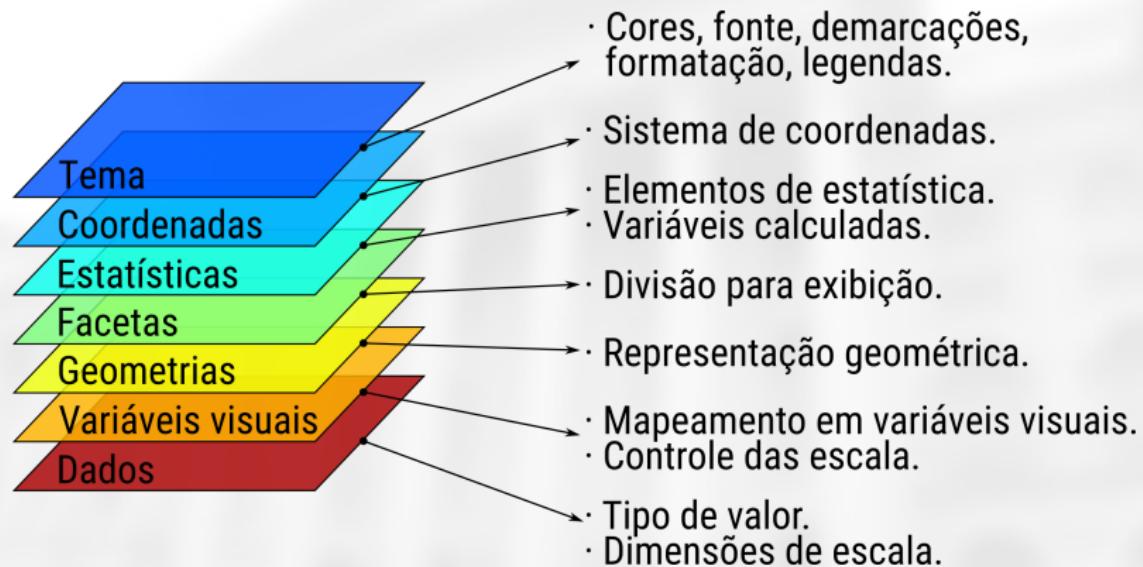
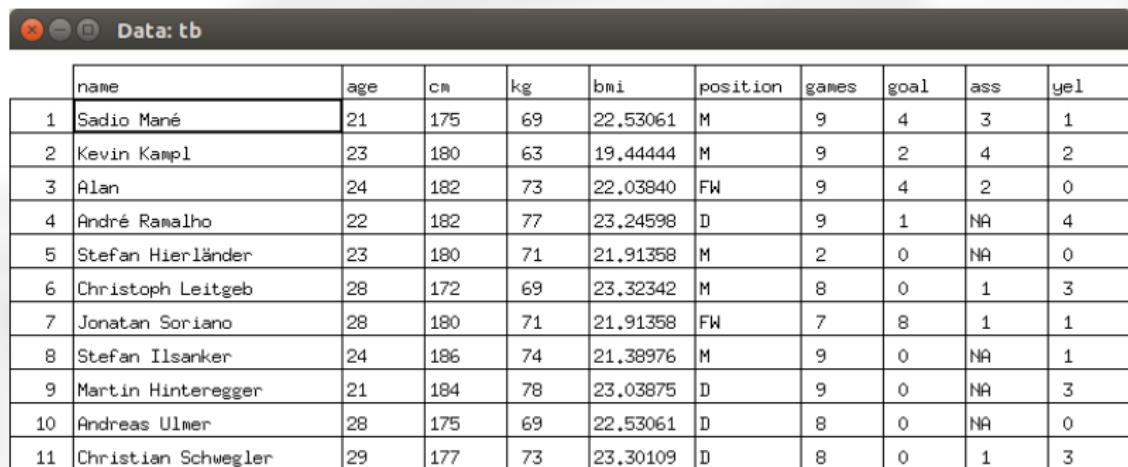


Figura 5. As camadas na gramática dos gráficos.

Camada 1: dados



The screenshot shows a data viewer window with a dark header bar containing three icons (close, minimize, maximize) and the text "Data: tb". The main area is a table with 11 rows and 11 columns. The columns are labeled: name, age, cm, kg, bmi, position, games, goal, ass, and yel. The data represents player statistics:

	name	age	cm	kg	bmi	position	games	goal	ass	yel
1	Sadio Mané	21	175	69	22.53061	M	9	4	3	1
2	Kevin Kampl	23	180	63	19.44444	M	9	2	4	2
3	Alan	24	182	73	22.03840	FW	9	4	2	0
4	André Ramalho	22	182	77	23.24598	D	9	1	NA	4
5	Stefan Hierländer	23	180	71	21.91358	M	2	0	NA	0
6	Christoph Leitgeb	28	172	69	23.32342	M	8	0	1	3
7	Jonatan Soriano	28	180	71	21.91358	FW	7	8	1	1
8	Stefan Ilsanker	24	186	74	21.38976	M	9	0	NA	1
9	Martin Hinteregger	21	184	78	23.03875	D	9	0	NA	3
10	Andreas Ulmer	28	175	69	22.53061	D	8	0	NA	0
11	Christian Schwegler	29	177	73	23.30109	D	8	0	1	3

Figura 6. A camada dos dados.

Camada 1: dados

Deve se estar atento ao tipo de valor/objeto.

- ▶ Quantitativa: variável numérica discreta ou contínua.
- ▶ Qualitativa: variável nominal ou ordinal.
- ▶ Cronológica: variável de data ou data-tempo.
- ▶ Geográfica: objeto como polígonos, por exemplo.

Camada 2: mapeamento em variáveis visuais

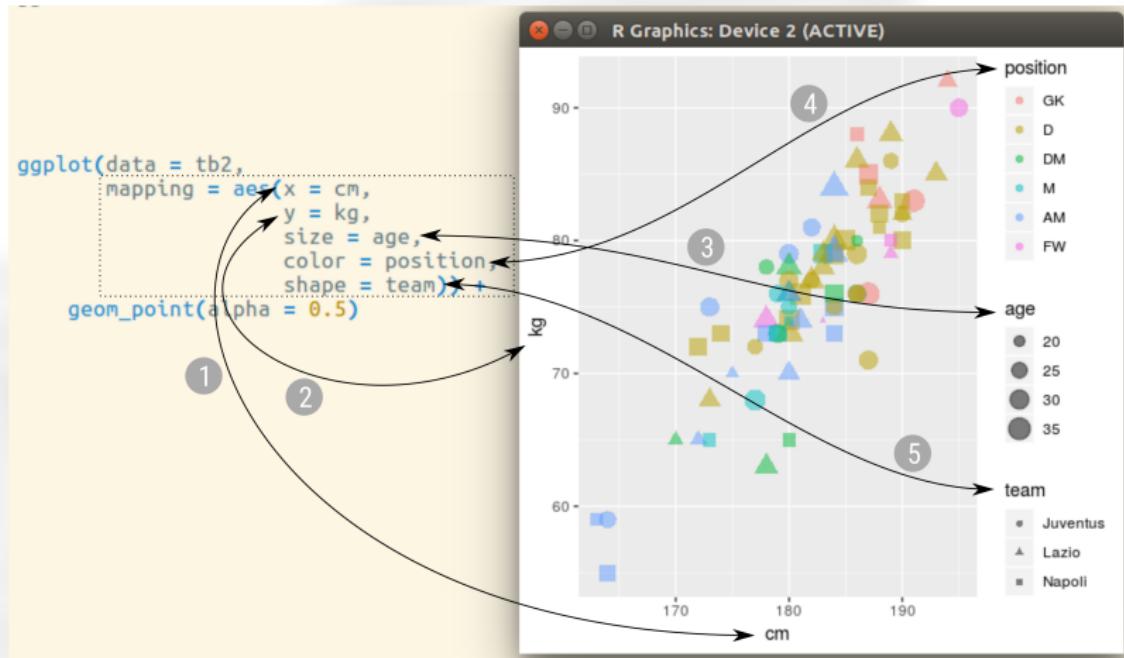


Figura 7. A camada de mapeamento dos valores em variáveis visuais.

Camada 2: mapeamento em variáveis visuais

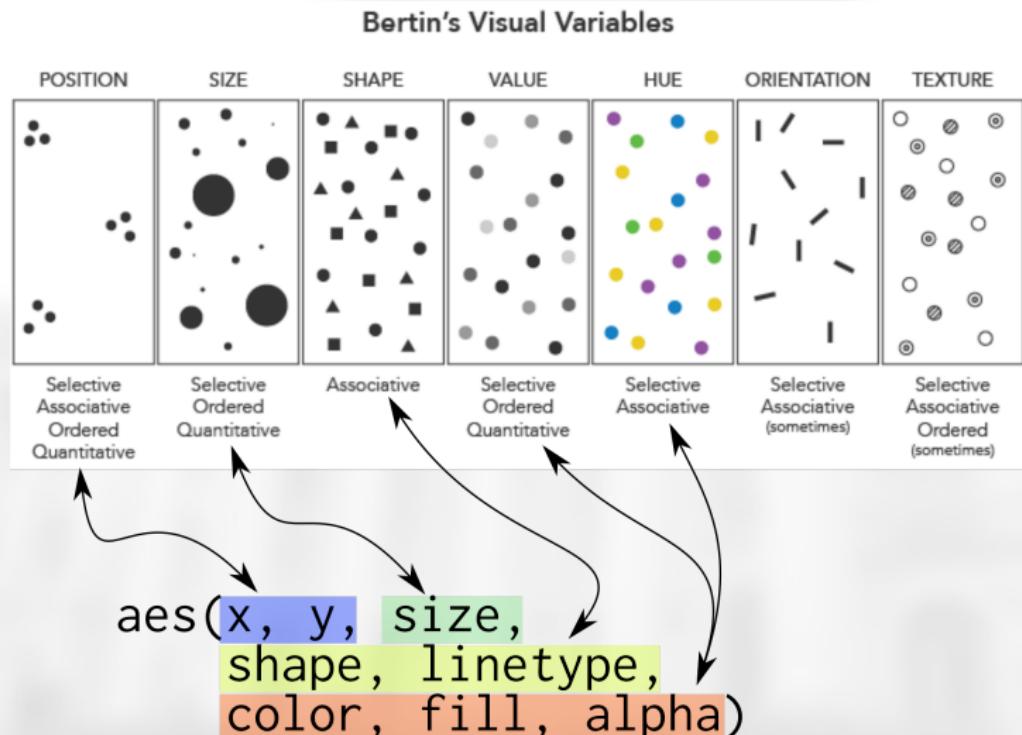


Figura 8. Variáveis visuais disponíveis no ggplot2.

Camada 2: mapeamento em variáveis visuais

Funções para controle de escala:

```
scale_alpha_:
  continuous date datetime discrete identity manual ordinal
scale_colour_:
  brewer continuous date datetime discrete distiller gradient
  gradient2 gradientn grey hue identity manual ordinal
scale_continuous_:
  identity
scale_discrete_:
  identity manual
scale_fill_:
  brewer continuous date datetime discrete distiller gradient
  gradient2 gradientn grey hue identity manual ordinal
scale_linetype_:
  continuous discrete identity manual
scale_shape_:
  continuous discrete identity manual ordinal
scale_size_:
  area continuous date datetime discrete identity manual ordinal
scale_x_:
  continuous date datetime discrete log10 reverse sqrt time
scale_y_:
  continuous date datetime discrete log10 reverse sqrt time
```

Camada 3: geometrias

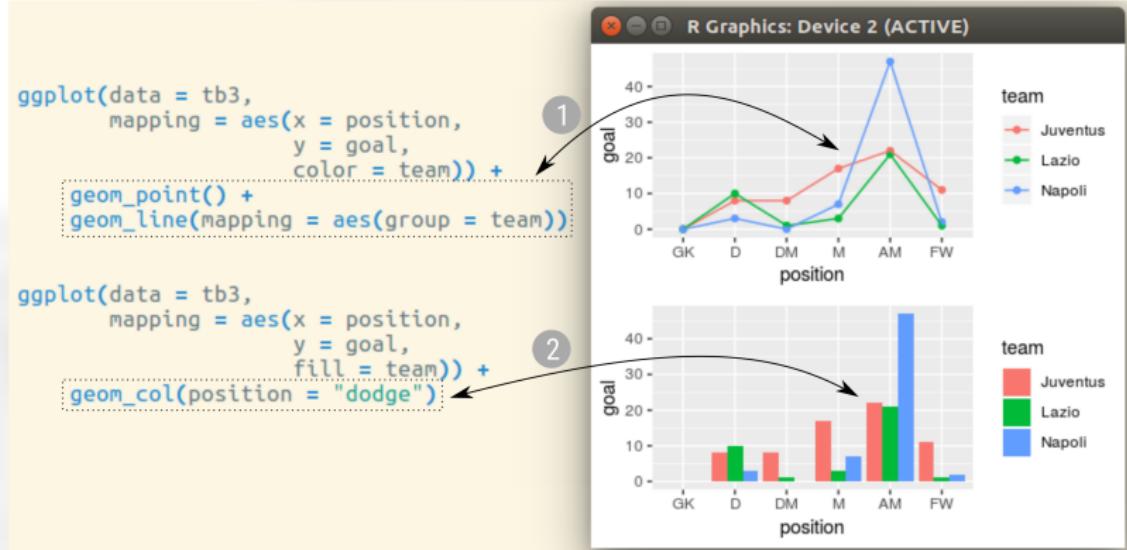


Figura 9. A camada de elementos geométricos.

Camada 3: geometrias

Funções disponíveis para a camada de geometria:

geom_abline	geom_freqpoly	geom_rect
geom_area	geom_hex	geom_ribbon
geom_bar	geom_histogram	geom_rug
geom_bin2d	geom_hline	geom_segment
geom_blank	geom_jitter	geom_sf
geom_boxplot	geom_label	geom_sf_label
geom_col	geom_line	geom_sf_text
geom_contour	geom_linerange	geom_smooth
geom_count	geom_map	geom_spoke
geom_crossbar	geom_path	geom_step
geom_curve	geom_point	geom_text
geom_density	geom_pointrange	geom_tile
geom_density2d	geom_polygon	geom_violin
geom_density_2d	geom_qq	geom_vline
geom_dotplot	geom_qq_line	update_geom_defaults
geom_errorbar	geom_quantile	
geom_errorbarh	geom_raster	

Camada 4: divisão em facetas

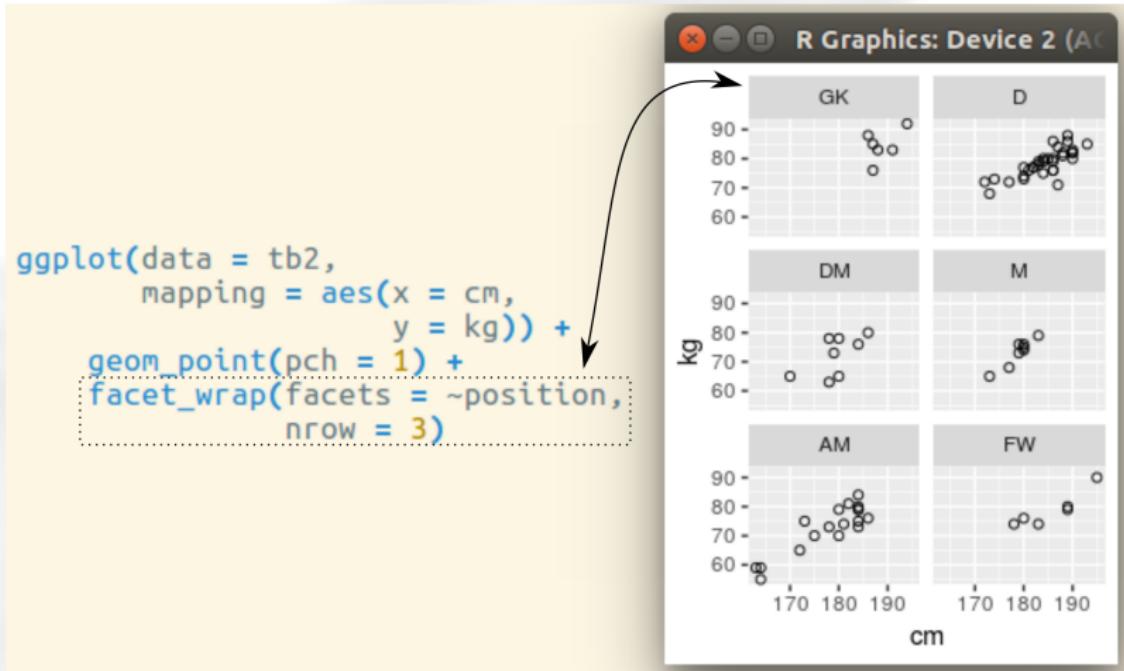


Figura 10. A camada da divisão em facetas.

Camada 4: divisão em facetas

Funções disponíveis para divisão em facetas:

```
facet_grid  
facet_null  
facet_wrap
```

Camada 5: estatística

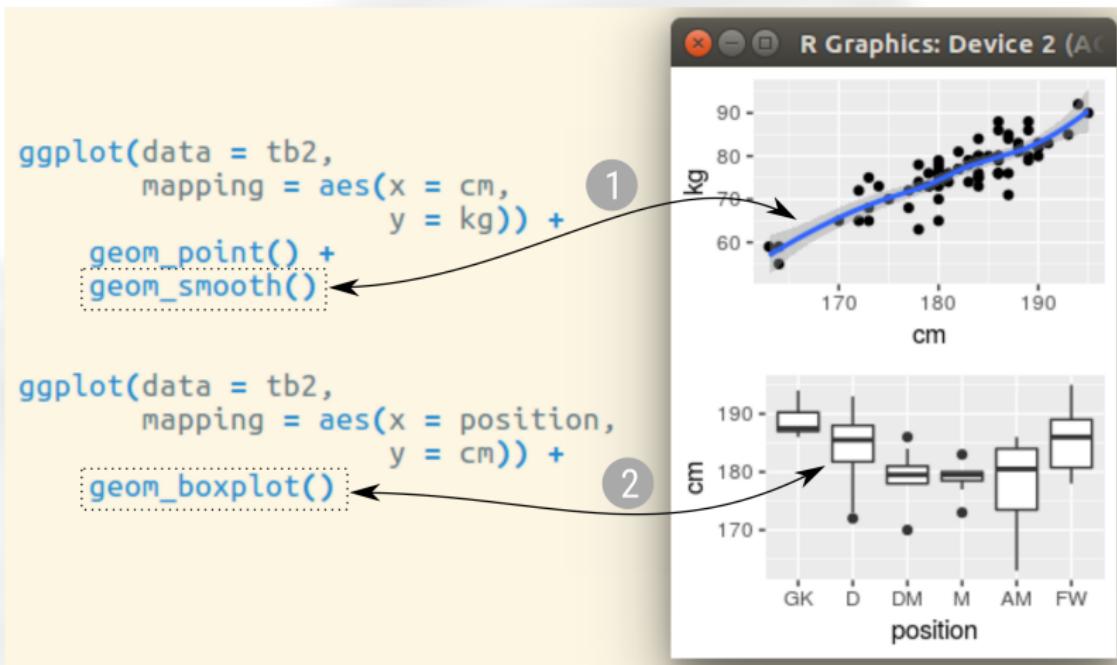


Figura 11. A camada de elementos de estatística

Camada 5: estatística

Funções disponíveis para a camada estatística:

```
stat_bin           stat_ecdf          stat_sum  
stat_bin2d         stat_ellipse        stat_summary  
stat_bin_2d        stat_function      stat_summary2d  
stat_binhex        stat_identity     stat_summary_2d  
stat_bin_hex       stat_qq            stat_summary_bin  
stat_boxplot       stat_qq_line       stat_summary_hex  
stat_contour       stat_quantile     stat_unique  
stat_count         stat_sf            stat_ydensity  
stat_density        stat_sf_coordinates update_stat_defaults  
stat_density2d      stat_smooth       stat_spoke
```

Camada 6: coordenadas

```
ggplot(data = tb4,  
       mapping = aes(x = 0,  
                      y = goal,  
                      fill = position)) +  
  geom_col(color = "black")  
  
ggplot(data = tb4,  
       mapping = aes(x = 0,  
                      y = goal,  
                      fill = position)) +  
  geom_col(color = "black") +  
  coord_polar(theta = "y")
```

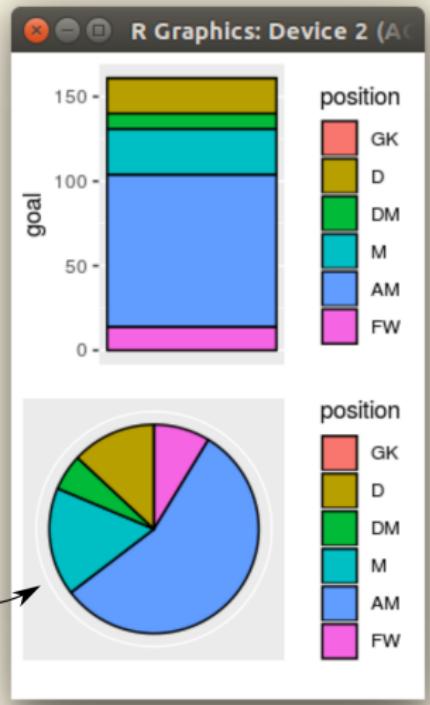


Figura 12. A camada do sistema de coordenadas.

Camada 6: coordenadas

Funções disponíveis para a camada de coordenadas:

```
coord_cartesian  
coord_equal  
coord_fixed  
coord_flip  
coord_map  
coord_munch  
coord_polar  
coord_quickmap  
coord_sf  
coord_trans
```

Camada 7: tema

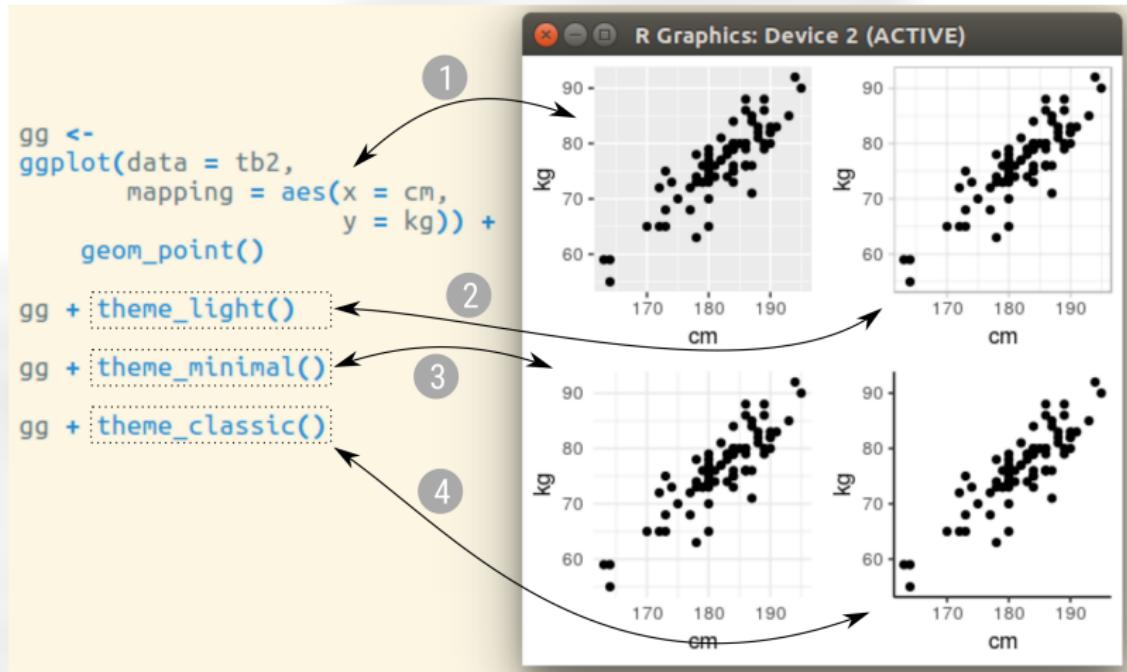


Figura 13. A camada de tema.

Camada 7: tema

Funções disponíveis para a camada de tema:

```
theme_bw  
theme_classic  
theme_dark  
theme_get  
theme_gray  
theme_grey  
theme_light  
theme_linedraw  
theme_minimal  
theme_replace  
theme_set  
theme_test  
theme_update  
theme_void
```

TODO WALMES FIXME e quais as funções que sobram?

ls_ggplot %>%

str_subset("[[:upper:]]", negate = TRUE)

1

2

3

Data Visualization with ggplot2 :: CHEAT SHEET



Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.



Complete the template below to build a graph.

```
ggplot(data = <DATA> +  
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>),  
  stat = <STAT>, position = <POSITIONS>) +  
  <COORDINATE_FUNCTION> +  
  <FACET_FUNCTION> +  
  <SCALE_FUNCTION> +  
  <THEME_FUNCTION>)
```

ggplot(data = mpg, aes(x = cyl, y = hwy)) begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | **data** | **geom**
plot() = (x, y) | **hwy** = (y-axis)
Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggssave("plot.png", width = 5, height = 5) Saves last plot as 5" x 5" file named "plot.png" in working directory. Matches file type to file extension.



Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

```
a <- ggplot(economics, aes(date, unemploy))  
b <- geom_point(aes(x = date, y = lat))  
  
# A geom, blank()  
# (useful for expanding limits)  
a + geom_curve(aes(yend = lat + 1,  
  xend = long + 1, curvature = 0.5), x = long, y = lat,  
  alpha = 0.5, color = "red", linetype = 1, size = 2)  
  
# A geom_path([in]eed = "butt", linejoin = "round",  
# lineheight = 1)  
x, y, alpha, color, group, linetype, size  
  
# A geom_polygon(aes(group = group))  
x, y, alpha, fill, group, linetype, size  
  
# B geom_rect(aes(xmin = long, ymin = lat - 1, xmax =  
# long + 1, ymax = lat + 1)) x, y, alpha, fill, group,  
# linetype, size, stroke  
  
# A geom_ribbon(aes(ymin = unemploy - 900,  
# ymax = unemploy + 900)) x, y, alpha, fill, group,  
# linetype, size
```

LINE SEGMENTS

```
common aesthetics: x, y, alpha, color, linetype, size  
b + geom_abline(aes(intercept=0, slope=1))  
b + geom_hline(aes(yintercept = 0))  
b + geom_vline(aes(xintercept = 0))  
  
b + geom_segment(aes(xend=long+1, yend=long+1))  
b + geom_spoke(aes(angle = 1:115, radius = 1))
```

ONE VARIABLE continuous

```
c <- ggplot(mpg, aes(hwy)); c <- ggplot(mpg)  
  
c + geom_area(stat = "bin")  
x, y, alpha, color, fill, linetype, size  
c + geom_density(binwidth = "gaussian")  
x, y, alpha, fill, group, linetype, size, weight  
c + geom_dotplot(binwidth = "y", stackdir =  
  "center") x, y, alpha, color, fill, group  
  
c + geom_freqpoly(x = y, alpha, color, group,  
  linetype, size)  
  
c + geom_histogram(binwidth = 5) x, y, alpha,  
  color, fill, linetype, size, weight  
  
c + geom_quasirandom(scale = hwy) x, y, alpha,  
  color, fill, group, linetype, size, weight
```

discrete

```
d <- ggplot(mpg, aes(frt))  
d + geom_bar()  
x, alpha, color, fill, linetype, size, weight
```

TWO VARIABLES

```
continuous x, continuous y  
e <- ggplot(economics, aes(label = city, nudge_x = 1,  
  alpha = 0.5, check_overlap = TRUE)) x, y, label,  
  linheight, size, vjust  
e + geom_jitter(height = 2, width = 2)  
x, y, alpha, color, fill, shape, size  
e + geom_point() x, y, alpha, color, fill, shape,  
  size, stroke  
e + geom_quantile() x, y, alpha, color, group,  
  linetype, size, weight  
e + geom_rectsides("bl") x, y, alpha, color,  
  linetype, size  
e + geom_smooth(method = lm) x, y, alpha,  
  color, fill, group, linetype, size, weight  
e + geom_text(aes(label = city, nudge_x = 1,  
  nudge_y = 1, check_overlap = TRUE)) x, y, label,  
  alpha, angle, color, family, fontface, hjust,  
  linheight, size, vjust
```

discrete x, continuous y

```
f <- ggplot(mpg, aes(class, hwy))  
  
f + geom_col() x, y, alpha, color, fill, group,  
  linetype, size  
f + geom_boxplot() x, y, lower, middle, upper,  
  ymin, ymax, alpha, color, fill, group, linetype,  
  shape, size, weight  
f + geom_dodgeplot(binwidth = "y") stackdir =  
  "center" x, y, alpha, color, fill, group  
  
f + geom_violin(scale = "area") x, y, alpha, color,  
  fill, group, linetype, size, weight
```

discrete x, discrete y

```
g <- ggplot(diamonds, aes(cut, color))  
  
g + geom_count() x, y, alpha, color, fill, shape,  
  size, stroke
```

THREE VARIABLES

```
seals <- with(seals, sort(delta_long^2 + delta_lat^2))  
l <- ggplot(seals, aes(long))  
  
l + geom_contour(aes(z = 1))  
x, y, alpha, colour, group, linetype,  
  size, weight  
l + geom_raster(aes(fill = z), hjust = 0.5, vjust = 0.5,  
  interpolate = FALSE) x, y, alpha, fill  
l + geom_tile(aes(fill = z)) x, y, alpha, color, fill,  
  linetype, size, width
```

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))  
  
h + geom_bin2d(binwidth = c(0.125, 500))  
x, y, alpha, color, fill, linetype, size, weight  
h + geom_density2d() x, y, alpha, colour, group, linetype, size  
h + geom_hex() x, y, alpha, colour, fill, size
```

continuous function

```
i <- ggplot(economics, aes(date, unemploy))  
  
i + geom_area() x, y, alpha, color, fill, linetype, size  
i + geom_line() x, y, alpha, color, group, linetype, size  
i + geom_step(direction = "hv")  
x, y, alpha, color, group, linetype, size
```

visualizing error

```
di <- diamonds %>% filter(x == "A" | x == "B")  
fit <- lm(price ~ cut * x, data = di)  
j <- ggplot(di, aes(grp, fit, ymin = fit - se, ymax = fit + se))  
  
j + geom_crossbar(fatten = 2)  
x, y, ymin, ymax, alpha, color, fill, group, linetype,  
  size  
j + geom_errorbar(x, y, ymin, ymax, alpha, color,  
  group, linetype, size, width) (also  
  geom_errorbarh())  
j + geom_linerange(x, y, ymin, alpha, color, group, linetype, size  
j + geom_pointrange(x, y, ymin, ymax, alpha, color, fill, group, linetype,  
  size, width)
```

maps

```
maps <- data.frame(murder = USArrests$Murder,  
  state = tolower(state.name[USArrests]$State),  
  map = map_data("state"))  
map <- map_data("state")  
k <- ggplot(maps, aes(state))  
k + geom_map(aes(map_id = state), map = map)  
map_id, alpha, color, fill, linetype, size
```

Stats

An alternative way to build a layer

A stat builds new variables to plot (e.g., count, prop).

```
graph LR
    data --> stat[stat]
    stat --> geom[geom]
    geom --> plot[coordinate plot]
    style: font-family: monospace;
    style: font-size: 0.8em;
```

Visualizes a stat by changing the default stat of a geom function, `geom_bar(stat="bar")` or by using a stat function, `stat_bin(geom="bar")`, which calls a default geom to make a layer (equivalent to a geom function). Use `.name..` syntax to map stat variables to aesthetics.

`geom` to use `stat function` `geom mappings`

```
i + stat_density2d(aes(fill = ..level..),
                    geom = "polygon")
variable created by stat
```

```
c + stat_bin(binwidth = 1, origin = 10)
x, y | .count..., .n..., .density..., .ndensity...
c + stat_count(w = 1) x + y, | .count..., .prop...
c + stat_density(alpha = 1, kernel = "gaussian")
x, y | .density..., .scaled...
```

```
e + stat_bin_dof(binwidth = 5, drop = T)
x, y | .count..., .n..., .density...
e + stat_bin_hex(bins=30) x, y, fill | .count..., .density...
e + stat_density2d(contour = TRUE, n = 100)
x, y, color, size | .level...
e + stat_ellipse(level = 0.95, segments = 51, type = "l")
x + stat_contours(pes(x) x, y, z, order | .level...
l + stat_summary_hex(pes(x) y, bins = 30, fun = max)
x, y, z, fill | .value...
l + stat_summary_2d(pes(x) z, bins = 30, fun = mean)
x, y, z, fill | .value...
f + stat_bxpplot(coef = 1.5) x, y | .lower...
, .middle..., .upper..., .width..., .ymin..., .ymax...
f + stat_idensity(kernel = "gaussian", scale = "area") x, y | .density..., .scaled..., .count..., .n...
, .violinwidth..., .width...
e + stat_ecdf(n = 40) x, y | .x...
e + stat_quantile(quartiles = c(0.1, 0.9), formula = y - log(x), method = "q1") x, y | .quartile...
e + stat_smooth(method = "lm", formula = y ~ x, se = T, level = 0.95) x, y | .se..., .x..., .ymin..., .ymax...
ggplot() + stat_function(fun = dnorm, args = list(sd = 0.5)) x, y | .x...
e + stat_identity(na.rm = TRUE)
```

```
ggplot() + stat_qq(q=qnorm(sample(1:100), dist = qt,
dparam = list(df = 5)), sample, x, y | .sample..., .theoretical...
e + stat_summ(x, y, size | .n..., .prop...
e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary(fun.y = "mean", geom = "bar")
e + stat_unique()
```

Scales

Scales map data values to the visual values of an aesthetic. To change a mapping, add a new scale.

`[n <- d + geom_bar(aes(fill = n))]`

`scale` `aesthetic` `prepackaged` `scale to use` `scale-specific arguments`

`n + scale_fill_manual`

`value = c("skyblue", "cyanblue", "blue", "navyblue", "darkblue", "purple", "darkpurple", "brown", "darkbrown", "tan", "darktan", "gray", "darkgray", "black", "darkblack")`

`label = c("1", "2", "3", "4", "5", "6", "7", "8", "9", "10")`

`limits = c(1, 10)`

`breaks = c(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)`

`guide = guide_colorbar`

`title = title` `to title use in legend` `label` `label to use in legend/axis` `breaks to use in legend/axis`

GENERAL PURPOSE SCALES

Use with most aesthetics

`scale_x_continuous` - map cont'ous values to visual ones

`scale_x_discrete` - map discrete values to visual ones

`scale_y_identity` - use data values as visual ones

`scale_y_log10` - map discrete values to visual ones

`scale_x_date` - map discrete dates to visual ones

`scale_x_datetime` - treat data x values as date times. Use `date_labels` as `scale_x_date(...)`. See `strptime` for label formats.

X & Y LOCATION SCALES

Use with x/y aesthetics (shown here)

`scale_x_log10` - Plot x on log10 scale

`scale_x_reverse` - Reverse direction of x axis

`scale_x_sqrt` - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)

`n + geom_bar(aes(fill = n))`

`n + stat_bin_hex(palette = "Blue")`

`RColorBrewer::display.brewer.all()`

`n + scale_fill_grey(start = 0.2, end = 0.8, na.value = "red")`

COLOR AND FILL SCALES (CONTINUOUS)

`o <- c + geom_dotplot(aes(..., fill = ..))`

`o + scale_fill_distiller(palette = "Blues")`

`o + scale_fill_gradient(low = "red", high = "yellow")`

`o + scale_fill_gradient2(low = "red", high = "blue", mid = "white", midpoint = 25)`

`o + scale_fill_hexbin(colours = topo.colors(6))`

`o + scale_fill_hexbin(pal = rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal())`

SHAPE AND SIZE SCALES

`p + e + geom_point(aes(shape = fill, size = cy))`

`p + scale_shape(..., scale = size)`

`p + scale_shape_manual(values = c(3:7))`

`p + scale_size(range = c(1,6))`

`p + scale_size_area(max_size = 6)`

Coordinate Systems

`r <- d + geom_bar()`

`t + coord_cartesian(xlim = c(0, 5))`

`xlim, ylim` `the default cartesian coordinate system` `range` `ratio`

`r + coord_flip()`

`Cartesian coordinates with fixed aspect ratio`

`r + coord_polar(theta = "x", direction = 1)`

`theta, start, direction`

`r + coord_trans(trans = "sqrt")`

`trans, xtrans, ytrans`

`r + coord_sf(..., crs = 4326)`

`crs` `set spatial coordinates. Set xtrans and ytrans to the name of a window function.`

`#+ coord_quickmap()`

`#+ coord_map(projection = "ortho", orientation = c(-141, -74, 0))`

`projection, orientation, xlim, ylim`

`Map projections from the mapproj package`

`"free_x": x axis limits adjust`

`"free_y": y axis limits adjust`

`Set scales to let axis limits vary across facets`

`t + facet_grid(cols = vars(f1), rows = vars(f2), scales = "free")`

`t + facet_grid(rows = vars(driv), cols = vars(f1), scales = "free")`

`x and y axis limits adjust to individual facets`

`"free_x": x axis limits adjust`

`"free_y": y axis limits adjust`

`Set tabeller to adjust facet labels`

`t + facet_grid(cols = vars(f1), labeller = label_both)`

`f1: n | n | n | n | n | n`

`t + facet_grid(rows = vars(f1), labeller = label_bquote(alpha ^ .(f1)))`

`alpha^n | alpha^n | alpha^n | alpha^n | alpha^n | alpha^n`

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

`s <- ggplot(mpg, aes(f1, fill = driv))`

`s + geom_bar(position = "dodge")`

`s + geom_bar(position = "fill")`

`Stack elements on top of one another, normalize height`

`d + position_nudge(x = 10, y = 10)`

`Random nudge to X and Y position of each element to avoid overlapping`

`s + geom_bar(position = "nudge")`

`Nudge labels away from points`

`s + geom_bar(position = "stack")`

`Stack elements on top of one another`

Each position adjustment can be recast as a function with `width` and `height` arguments

`s + geom_bar(position = position_dodge(width = 1))`

Themes

`r + theme_bw()`

`black background with grid lines`

`r + theme_gray()`

`grayscale theme`

`r + theme_minimal()`

`Minimal theme`

`r + theme_dark()`

`dark for contrast`

`r + theme_void()`

`Empty theme`

`r + theme_classic()`

`r + theme_linedraw()`

`r + theme_ticks()`

`n + theme(base.position = "bottom")`

Place legend at "bottom", "top", "left", or "right"

`n + guides(fill = "none")`

Set guide type for each aesthetic: colorbar, legend, or none (no legend)

`n + scale_fill_discrete(name = "Title", labels = "New Labels", legend.outside = TRUE)`

Set legend title and labels with a scale function.

`zooming`

`Without clipping (preferred)`

`+ coord_cartesian(xlim = c(0, 100), ylim = c(10, 20))`

`With clipping (removes unseen data points)`

`+ xlim(0, 100) + ylim(10, 20)`

`+ scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 20))`

Figura 15. Cartão de referência dos recursos do ggplot2.

Walmes Zeviani · UFPR

Visualização de dados com ggplot2

35