GGPLOT2
Cheatsheet

Ruoxi Liu (rl3155) Ziyu Fang (zf2253)

October 25, 2021

1 Basic Info

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.

base code

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

case1

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

qplot(x = cty, y = hwy, data = mpg, geom =
"point")
Creates a complete plot with given data, geom,
and mappings. Supplies many useful defaults.

last_plot() Returns the last plot

ggsave("plot.png", width = 5, height = 5)
Saves last plot as 5x5 file named "plot.png" in
working directory. Matches file type to file
extension.
```

2 Geoms

GRAPHICAL PRIMITIVES

base code

```
a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))
```

case1

```
a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1,
xend=long+1,curvature=z))
x, xend, y, yend, alpha, angle, color,
curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round",
linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax=
long + 1, ymax = lat + 1))
xmax, xmin, ymax, ymin, alpha, color, fill,
linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900,
ymax=unemploy + 900))
x, ymax, ymin, alpha, color, fill, group,
linetype, size
```

LINE SEGMENTS

```
b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))
```

GRAPHICAL PRIMITIVES

ONE VARIABLE

```
c <- ggplot(mpg, aes(hwy));
c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size,
weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly()
x, y, alpha, color, group, linetype, size

c + geom_histogram(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight

c2 + geom_qq(aes(sample = hwy))
x, y, alpha, color, fill, linetype, size, weight
```

TWO VARIABLES

```
e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family,
fontface, hjust, lineheight, 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_rug(sides = "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 = cty), nudge_x = 1,
nudge_y = 1, check_overlap = TRUE)
x, y, label, alpha, angle, color, family,
fontface, hjust, lineheight, size, vjust
```

GRAPHICAL PRIMITIVES

discrete x and 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, ymax, ymin, alpha,
color, fill, group, linetype, shape, size,
weight

f + geom_dotplot(binaxis = "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 and discrete y

```
g <- ggplot(diamonds, aes(cut, color))

g + geom_count()
x, y, alpha, color, fill, shape, size, stroke
```

continuous bivariate distribution

```
h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 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
```

STATS

Stats

An alternative way to build a layer A stat builds new variables to plot (e.g., count, prop).

example

```
Visualize a stat by changing the default stat of
a geom function, geom_bar(stat="count") or by
using a stat function, stat_count(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.
```

base code

```
c + stat_bin(binwidth = 1, origin = 10)
c + stat_count(width = 1)
c + stat_density(adjust = 1, kernel =
"gaussian")

e + stat_bin_2d(bins = 30, drop = T)
e + stat_bin_hex(bins=30)
e + stat_density_2d(contour = TRUE, n = 100)

e + stat_ellipse(level = 0.95, segments = 51,
type = "t")
l + stat_contour(aes(z = z))
l + stat_summary_hex(aes(z = z), bins = 30, fun
= max)
l + stat_summary_2d(aes(z = z), bins = 30, fun =
mean)

f + stat_boxplot(coef = 1.5)
f + stat_ydensity(kernel = "gaussian", scale =
"area")
e + stat_ecdf(n = 40)

e + stat_quantile(quantiles = c(0.1, 0.9),
formula = y ~ log(x), method = "rq")
e + stat_smooth(method = "lm", formula = y ~ x,
se=T, level=0.95)
ggplot() + stat_function(aes(x = -3:3), n = 99,
fun = dnorm, args = list(sd=0.5))
e + stat_identity(na.rm = TRUE)
ggplot() + stat_qq(aes(sample=1:100), dist = qt,
dparam=list(df=5))

e + stat_sum()
e + stat_summary(fun.data = "mean_cl_boot")
h + stat_summary_bin(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.

example

```
n + scale_fill_manual(values = c("skyblue",
"royalblue", "blue", "navy"), limits = c("d",
"e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))
```

GENERAL PURPOSE SCALES

```
scale_*_continuous() - map cont' values to
visual ones
scale_*_discrete() - map discrete values to
visual ones scale_*_identity() - use data values
as visual ones
scale_*_manual(values = c()) - map discrete
values to manually chosen visual ones
scale_*_date(date_labels = "\%m/\%d"),
date_breaks = "2 weeks") - treat data values as
dates.
scale_*_datetime() - treat data x values as date
times. Use same arguments as scale_x_date().
```

COLOR AND FILL SCALES (DISCRETE)

```
n <- d + geom_bar(aes(fill = fl))
n + scale_fill_brewer(palette = "Blues")
For palette choices:
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 = ..x..))
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_gradientn(colours=topo.colors(6))

p <- e + geom_point(aes(shape = fl, size = cyl))
p + scale_shape() + scale_size()
p + scale_shape_manual(values = c(3:7))
p + scale_radius(range = c(1,6))
p + scale_size_area(max_size = 6)
```

Others

Coordinate Systems

```
r <- d + geom_bar()
r + coord_cartesian(xlim = c(0, 5))
r + coord_fixed(ratio = 1/2)
r + coord_flip()
r + coord_polar(theta = "x", direction=1 )
r + coord_trans(ytrans = "sqrt")
 + coord_quickmap()
 + coord_map(projection = "ortho", orientation=c(41,
-74, 0))
```

Position Adjustments

```
s <- ggplot(mpg, aes(fl, fill = drv))
s + geom_bar(position = "dodge")
s + geom_bar(position = "fill")
e + geom_point(position = "jitter")
e + geom_label(position = "nudge")
s + geom_bar(position = "stack")
s + geom_bar(position = position_dodge(width = 1))
```

Themes

```
r + theme_classic()
r + theme_light()
r + theme_linedraw()
r + theme_minimal()
```

Faceting

```
t <- ggplot(mpg, aes(cty, hwy)) + geom_point()
t + facet_grid(cols = vars(fl))
t + facet_grid(rows = vars(year))
t + facet_grid(rows = vars(year), cols = vars(fl))
t + facet_wrap(vars(fl))
t + facet_grid(rows = vars(drv), cols = vars(fl),
scales = "free")
t + facet_grid(cols = vars(fl), labeller =
label_both)
t + facet_grid(rows = vars(fl), labeller =
label_bquote(alpha ^ .(fl)))
```

Labels

```
t + labs( x = "New x axis label", y = "New y axis
label", title ="Add a title above the plot",  Use
scale functions subtitle = "Add a subtitle below
title", caption = "Add a caption below plot")
t + annotate(geom = "text", x = 8, y = 9, label =
"A")
```

Legends

```
n + theme(legend.position = "bottom")
n + guides(fill = "none")
n + scale_fill_discrete(name = "Title", labels =
c("A", "B", "C", "D", "E"))
```

Zooming

```
t + coord_cartesian(xlim = c(0, 100), ylim = c(10,
20))
t + xlim(0, 100) + ylim(10, 20)
t + scale_x_continuous(limits = c(0, 100)) +
scale_y_continuous(limits = c(0, 100))
```