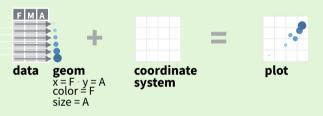
Data Visualization with ggplot2:: CHEAT SHEET

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.

required ggplot (data = <DATA>) + <GEOM_FUNCTION> (mapping = aes(<MAPPINGS> stat = <STAT>, position = <POSITION>) + required, <COORDINATE FUNCTION>+ defaults <FACET FUNCTION> supplied <SCALE FUNCTION> >

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

aesthetic mappings | data | geom

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

<THEME FUNCTION>

ggsave("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 <- ggplot(seals, aes(x = long, y = lat))

> 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

common aesthetics: x, y, alpha, color, linetype, size



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))

ONE VARIABLE continuous

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

discrete d <- ggplot(mpg, aes(fl))



d + geom_bar() x, alpha, color, fill, linetype, size, weight

TWO VARIABLES

continuous x, continuous y

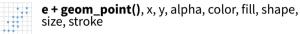
e <- ggplot(mpg, aes(cty, hwy))



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_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 vicet lineheight, size, viust

discrete x, continuous y

f <- ggplot(mpg, aes(class, hwy))



f + geom_col(), x, y, alpha, color, fill, group,



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, 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()



i + geom_step(direction = "hv") x, y, alpha, color, group, linetype, size

 $df \leftarrow data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)$ i <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))</pre>



j + geom_crossbar(fatten = 2) x, y, ymax, ymin, alpha, color, fill, group, linetype,



j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also geom_errorbarh())



x, ymin, ymax, alpha, color, group, linetype, size



j + geom_pointrange() x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

data <- data.frame(murder = USArrests\$Murder, state = tolower(rownames(USArrests))) map <- map_data("state") k <- ggplot(data, aes(fill = murder))



k + geom_map(aes(map_id = state), map = map)
+ expand_limits(x = map\$long, y = map\$lat), map_id, alpha, color, fill, linetype, size

THREE VARIABLES

seals\$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))l <- ggplot(seals, aes(long, lat))



l + geom contour(aes(z = z))x, y, z, alpha, colour, group, linetype, size, weight



l + geom_raster(aes(fill = z**)**, hjust=0.5, vjust=0.5, interpolate=FALSE) x, y, alpha, fill

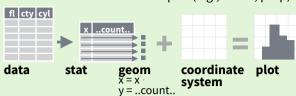


l + geom_tile(aes(fill = z)), x, y, alpha, color, fill, linetype, size, width



Stats An alternative way to build a layer

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



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.



geom to use stat function geommappings

i + stat_density2d(aes(fill = ..level..), geom = "polygon")

variable created by stat

c + stat_bin(binwidth = 1, origin = 10) **x, y** | ...count..., ..ncount..., ..density..., ..ndensity...

c + stat_count(width = 1) x, y, | ...count..., ...prop...

c + stat_density(adjust = 1, kernel = "gaussian") **x, y,** | ...count..., ...density..., ...scaled..

e + stat_bin_2d(bins = 30, drop = T) **x, y, fill** ...count.., ..density..

e + stat_bin_hex(bins=30) x, y, fill | ..count.., ..density...

e + stat_density_2d(contour = TRUE, n = 100) x, y, color, size 1...level...

e + stat ellipse(level = 0.95, segments = 51, type = "t")

l + stat_contour(aes(z = z)) x, y, z, order | ..level..

 $l + stat_summary_hex(aes(z = z), bins = 30, fun = max)$ x, y, z, fill | ..value..

 $l + stat_summary_2d(aes(z = z), bins = 30, fun = mean)$ x, y, z, fill | ..value..

f + stat_boxplot(coef = 1.5) x, y | ..lower..., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..

f + stat_ydensity(kernel = "gaussian", scale = "area") **x, y** ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

e + stat_ecdf(n = 40) **x, y** | ..x.., ..y..

e + stat_quantile(quantiles = c(0.1, 0.9), formula = $y \sim log(x)$, method = "rq") **x, y** | ..quantile..

e + stat_smooth(method = "lm", formula = y ~ x, se=T, level=0.95) **x, y** | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

ggplot() + stat_function(aes(x = -3:3), n = 99, fun = dnorm, args = list(sd=0.5)) x | ..x.., ..y..

e + stat_identity(na.rm = TRUE)

 $\label{eq:ggplot() + stat_qq(aes(sample=1:100), dist = qt, dparam=list(df=5)) sample, x, y \mid ...sample..., ...theoretical..}$

e + stat_sum() x, y, size | ..n.., ..prop..

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.



GENERAL PURPOSE SCALES

Use with most aesthetics

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(). See ?strptime for

X & Y LOCATION SCALES

Use with x or y aesthetics (x shown here)

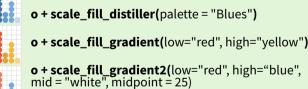
scale_x_log10() - Plot x on log10 scale scale_x_reverse() - Reverse direction of x axis scale_x_sqrt() - Plot x on square root scale

COLOR AND FILL SCALES (DISCRETE)



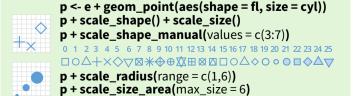
COLOR AND FILL SCALES (CONTINUOUS)

o <- c + geom_dotplot(aes(fill = ..x..))</pre>



o + scale_fill_gradientn(colours=topo.colors(6)) Also: rainbow(), heat.colors(), terrain.colors(), cm.colors(), RColorBrewer::brewer.pal()

SHAPE AND SIZE SCALES



Coordinate Systems

r <- d + geom_bar()



xlim, ylim Flipped Cartesian coordinates

r + coord_polar(theta = "x", direction=1) theta, start, direction Polar coordinates

r + coord_trans(ytrans = "sqrt")

xtrans, ytrans, limx, limy Transformed cartesian coordinates. Set xtrans and ytrans to the name of a window function.



π + coord_quickmap()

π + coord map(projection = "ortho", orientation=c(41, -74, 0))projection, orienztation,

Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Position Adjustments

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





s + geom_bar(position = "stack") Stack elements on top of one another

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

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

Themes



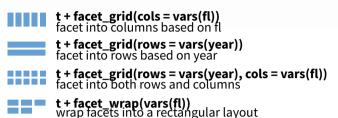


Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.



t <- ggplot(mpg, aes(cty, hwy)) + geom_point()



Set **scales** to let axis limits vary across facets

t + facet_grid(rows = vars(drv), cols = vars(fl), scales = "free")

x and y axis limits adjust to individual facets

"free_x" - x axis limits adjust "free_y" - y axis limits adjust

Set labeller to adjust facet labels

t + facet grid(cols = vars(fl), labeller = label both) fl: c fl: d fl: e fl: p fl: r t + facet grid(rows = vars(fl). labeller = label_bquote(alpha ^ .(fl))) α^c α^d α^e α^p α^r

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", to update legend caption = "Add a caption below plot", <AES> = "New <AES> legend title")

t + annotate(geom = "text", x = 8, y = 9, label = "A")

geom to place manual values for geom's aesthetics

Legends

n + theme(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"

n + guides(fill = "none")
Set legend type for each aesthetic: colorbar, legend, or
none (no legend)

n + scale_fill_discrete(name = "Title", labels = c("A", "B", "C", "D", "E")) Set legend title and labels with a scale function.

Zooming



Without clipping (preferred)

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

With clipping (removes unseen data points)

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

 $t + scale_x_continuous(limits = c(0, 100)) + scale_y_continuous(limits = c(0, 100))$

