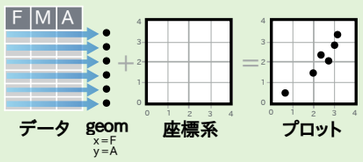
ggplot2を使ったデータのビジュアライゼーション チートシート

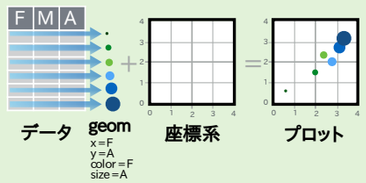
RStudio

基本

ggplot2は「グラフィクス文法」という考えに基づいており、どのグラフもデータセット、geomセット、座標系という同じコンポーネント群から作られる。



表示の際には、データセット内の変数をgeomの審美的属性(色、サイズなど)およびX、Y座標に従ってマッピングする。



グラフの作成はggplot()またはqplot()で行う。

ggplot(data = mpg, **aes(**x = cty, y = hwy**))**

ggplotは、デフォルトの表示は特になく、レイヤーを加えることでqplot()よりも細かい調整ができる。

データ

```
ggplot(mpg, aes(hwy, cty)) +
  geom_point(aes(color = cyl)) +
  geom_smooth(method = "lm") +
  coord_cartesian() +
  scale_color_gradient() +
  theme_bw()
```

+を使ってレイヤーや要素を追加

レイヤー = geom + stat + レイヤー特有マッピング

追加要素

geom_*()関数やstat_*()関数で新しいレイヤーを追加することで、geom、審美的属性、表示計算と位置調整を行うことができる。

審美的属性　データ　geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point"**)**

qplotは、与えられたデータ、geo、マッピング情報を使いプロットを行う。表示はデフォルトで調整済。

last_plot()

最後に行ったプロットを返す。

ggsave("plot.png", width = 5, height = 5)

例：最後に行ったプロットを5×5(インチ)で「plot.png」に保存。ファイル形式は拡張子で判断。

Geoms - データポイントの表現にはgeomを使う。またgeomの審美的属性(aes)を使って変数を表現する。それぞれの関数がレイヤーとなる。

グラフのプリミティブ

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

a + geom_blank()
(軸を拡大するのに使う)

a + geom_curve(aes(yend = lat + delta_lat, xend = long + delta_long, curvature = z**))**
x, xend, y, yend, alpha, angle, color, curvature, linetype, size

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

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

a + geom_rect(aes(xmin = long, ymin = lat, xmax=long + delta_long, ymax = lat + delta_lat)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

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

a + geom_segment(aes(yend=lat + delta_lat, xend = long + delta_long)**)**
x, xend, y, yend, alpha, color, linetype, size

a + geom_spoke(aes(yend = lat + delta_lat, xend = long + delta_long)**)**
x, y, angle, radius, alpha, color, linetype, size

1変数

連続値の場合

c <- ggplot(mpg, aes(hwy))

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

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
a + geom_freqpoly(aes(y = ..density..))

c + geom_histogram(binwidth = 5**)**
x, y, alpha, color, fill, linetype, size, weight
a + geom_histogram(aes(y = ..density..))

離散値の場合

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

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

2変数

Xが連続値、Yが連続値の場合

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

Xが離散値、Yが連続値の場合

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

f + geom_bar(stat = "identity")
x, y, alpha, color, fill, linetype, size, weight

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

Xが離散値、Yが離散値の場合

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

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

2変数の連続分布

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

連続関数

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

誤差の表示

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

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

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

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

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

マップ

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

3変数

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

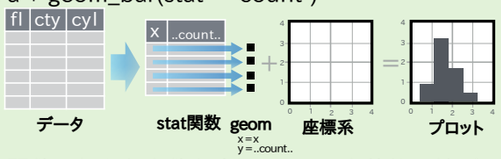
l + geom_contour(aes(z = z)**)**
x, y, z, alpha, colour, group, linetype, size, weight

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

Stat - レイヤーを作る別の方法

オリジナルのデータセットを変換してプロットする場合もある。変換してプロットするにはstat関数を使う。
(例) a + geom_bar(stat = "count")



stat関数は新たな変数を追加しそれに審美的属性がマップされる。これらの変数は..name..という共通の書き方をする。

stat関数はgeom関数と組み合わせレイヤーを構成。
(例)stat_bin(geom="bar")とgeom_bar(stat="bin")は同じ。

stat関数 / レイヤーマッピング

i + **stat_density2d**(aes(fill = ..level..), geom = "polygon", n = 100)

変換により作られた変数 / レイヤーのgeom / statのパラメーター

1次元分布

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..

2次元分布

e + **stat_bin_2d**(bins = 30, drop = TRUE)
x, y, fill | ..count.., ..density..
e + **stat_density_2d**(contour = TRUE, n = 100)
x, y, color, size | ..level..
e + **stat_ellipse**(level = 0.95, segments = 51, type = "t")

3変数

l + **stat_contour**(aes(z = z))
x, y, z, order | ..level..
l + **stat_summary_hex**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..
l + **stat_summary_2d**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

比較

f + **stat_boxplot**(coef = 1.5)
x, y | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..
f + **stat_ydensity**(adjust = 1, kernel = "gaussian", scale = "area")
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

関数

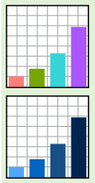
e + **stat_ecdf**(n = 40)
x, y | ..x.., ..y..
e + **stat_quantile**(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")
x, y | ..quantile..
e + **stat_smooth**(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

汎用目的

ggplot() + **stat_function**(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5))
x | ..x.., ..y..
e + **stat_identity**(na.rm = TRUE)
ggplot() + **stat_qq**(aes(sample=1:100), distribution = qt, dparams = list(df=5))
sample, x, y | ..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**()

スケール

スケールはデータ値を審美的属性の表示値にどうマッピングするかを調整する。カスタムスケールを追加することでマッピングを変えられる。



n <- b + **geom_bar**(aes(fill = fl))
n

scale_ / 審美的属性 / 既製のスケールが使える / スケール特有の引数

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"))

マッピングに使う値の範囲 / 凡例や軸のタイトル / 凡例や軸のラベル / 凡例や軸での分割の位置

汎用目的スケール

どの審美的属性にも使える
alpha, color, fill, linetype, shape, size

scale_*_continuous() - 連続値を表示値にマッピング
scale_*_discrete() - 離散値を表示値にマッピング
scale_*_identity() - データ値を表示値に使う
scale_*_manual(values = c()) - 離散値をマニュアル選択した表示値にマッピング

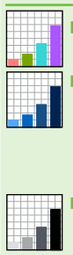
XおよびY位置のスケール

XまたYの審美的属性と共に使用(例ではX)

scale_x_date(date_labels = "%m/%d", date_breaks = "2 weeks") - xの値を日付として扱う。ラベルのフォーマットは?strptime参照。
scale_x_datetime() - xの値を日時として扱う。引数はscale_x_date()と同じ。
scale_x_log10() - xをlog10スケールでプロット
scale_x_reverse() - x軸を逆向きにする
scale_x_sqrt() - xを平方根スケールでプロット

色およびfillスケール

離散値



n <- d + **geom_bar**(aes(fill = fl))

n + **scale_fill_brewer**(palette = "Blues")
パレットの選択:
library(RColorBrewer)
display.brewer.all()

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

連続値

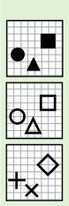
o <- c + **geom_dotplot**(aes(fill = ..x..))

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 = terrain.colors(6))
他に rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

shapeのスケール

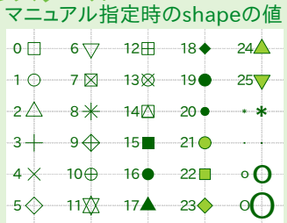


p <- e + **geom_point**(aes(shape = fl, size = cyl))

p + **scale_shape**(solid = FALSE)

p + **scale_shape_manual**(values = c(3:7))
shapeの値は右図参照

マニュアル指定時のshapeの値



sizeのスケール

p + **scale_radius**(range=c(1,6))
p + **scale_size**()

p + **scale_size_area**(max=6)
円の領域にマップする値(半径ではない)

座標系

r <- d + geom_bar()

r + **coord_cartesian**(xlim = c(0, 5))
xlim, ylim
デフォルトの直交座標系

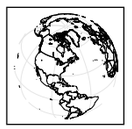
r + **coord_fixed**(ratio = 1/2)
ratio, xlim, ylim
X軸とY軸のアスペクト比の決まった直交座標系

r + **coord_flip**()
xlim, ylim
直交座標系をフリップ

r + **coord_polar**(theta = "x", direction=1)
theta, start, direction
極座標系

r + **coord_trans**(ytrans = "sqrt")
xtrans, ytrans, limx, limy
直交座標系を変換。xtransとytransでウィンドウ関数の名前を指定する。

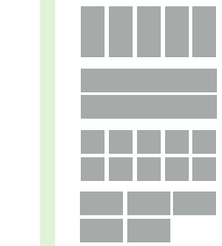
π + **coord_map**(projection = "ortho", orientation=c(41, -74, 0))
projection, orientation, xlim, ylim
mapprojパッケージのマッププロジェクション(mercator (デフォルト),azequalarea,lagrange等)



ファセット

ファセットは1つ以上の離散変数の値に従いサブプロットに分けて表示する。

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



t + **facet_grid**(. ~ fl)
flの値で列方向にファセット作成
t + **facet_grid**(year ~ .)
yearの値で行方向にファセット作成
t + **facet_grid**(year ~ fl)
行、列方向共にファセット作成
t + **facet_wrap**(~ fl)
長方形になるようファセットをラップ

ファセット間で軸の上下限が変えられるようスケールをセット

t + **facet_grid**(drv ~ fl, **scales = "free"**)
各ファセットでのX軸、Y軸の上下限の調整
- **"free_x"** - x軸の上下限調整
- **"free_y"** - y軸の上下限調整

ファセットのラベルの調整にはllabellerを使用

t + **facet_grid**(. ~ fl, **labeller = label_both**)

f : c	f : d	f : e	f : p	f : r

t + **facet_grid**(fl ~ ., **labeller = label_bquote**(alpha ^ .(fl)))

α^c	α^d	α^e	α^p	α^r

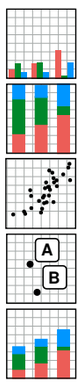
t + **facet_grid**(. ~ fl, **labeller = label_parsed**)

c	d	e	p	r

位置調整

位置調整をして同じスペースを占めるgeomをどのように調整するかを決める

s <- ggplot(mpg, aes(fl, fill = drv))



s + **geom_bar**(**position = "dodge"**)
サイド・バイ・サイドに並べる
s + **geom_bar**(**position = "fill"**)
要素を積み上げ、高さを揃える
e + **geom_point**(**position = "jitter"**)
プロットの重なりを避けるためX位置およびY位置にランダムノイズを加える
e + **geom_label**(**position = "nudge"**)
ラベルを点から離す
s + **geom_bar**(**position = "stack"**)
要素を積み上げる

各位置調整は、幅や高さの指定を加えて関数として指定することも可能

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

ラベル

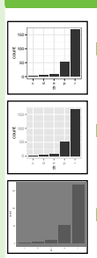
t + **ggtitle**("New Plot Title")
図の上にメインタイトルを表示
t + **xlab**("New Xlabel")
X軸ラベルを変更
t + **ylab**("New Ylabel")
Y軸ラベルを変更
t + **labs**(title =" New title", x = "New x", y = "New y")
上記すべてを行う

凡例ラベルをアップデートするには scale関数を使う

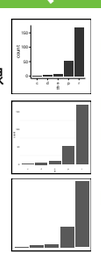
凡例

n + **theme**(legend.position = "bottom")
凡例の位置を指定("bottom", "top","left", "right")
n + **guides**(fill = "none")
各審美的属性について凡例タイプをセット、または凡例なしの指定
n + **scale_fill_discrete**(name = "Title", labels = c("A", "B", "C", "D", "E"))
スケール関数と共に凡例のタイトルとラベルをセット

テーマ



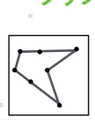
r + **theme_bw**()
背景白+グリッド線
r + **theme_gray**()
背景グレー(デフォルト)
r + **theme_dark**()
背景暗め



r + **theme_classic**()
r + theme_light()
r + theme_linedraw()
r + **theme_minimal**()
最小テーマ
r + **theme_void**()
空のテーマ

ズーム

クリッピングなし(推奨)
t + **coord_cartesian**(xlim = c(0, 100), ylim = c(10, 20))



クリッピングあり(見えない点は除去される)
t + **xlim**(0, 100) + **ylim**(10, 20)
t + **scale_x_continuous**(limits = c(0, 100)) + **scale_y_continuous**(limits = c(0, 100))