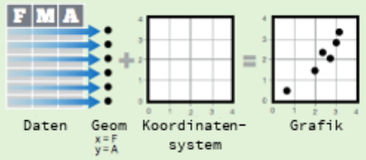
Daten veranschaulichen mit ggplot2

Schummelzettel

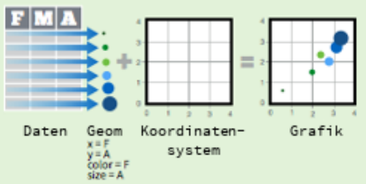
RStudio

Grundlagen

ggplot2 basiert auf der **„Grammatik von Grafiken"**, einem Konzept das besagt, dass jede Grafik durch die selben wenigen Komponenten erstellt werden kann: ein **Datensatz**, ein **Koordinatensystem** und eine Menge an **„Geomen"**— visuelle Markierungen der Datenpunkte.



Um Datenwerte abzubilden werden Variablen im Datensatz mit den ästhetischen Eigenschaften des Geoms (z. B. **Größe**, **Farbe**, **x**- und **y**-Koordinate) verknüpft.



Ein Graph wird mittels **ggplot()** oder **qplot()** erzeugt.

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

Startet die unsichtbare Grundebene einer Grafik, die mittels weiterer Ebenen erstellt wird. Diese erste Ebene hat keine Standardeinstellung, aber mehr Bedienungselemente als qplot().

Daten

neue Ebene oder Elemente mittels + einfügen

Ebene = Geom & Standard-Statistik & Ebenen-spezifische Zuordnung

zusätzliche Elemente

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

Eine **geom_*()** oder **stat_*()** Funktion fügt eine neue Ebene ein. Diese verfügt über ein Geom, ästhetische Eigenschaften, eine statistische Standardeinstellung und Möglichkeiten der Positionierung.

ästhetische Eigenschaften — Daten — Geom

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

Erstellt eine komplette Grafik mit dem vorhandenen Datensatz, dem Geom und der Zuordnung. Beinhaltet viele nützliche Standardeinstellungen.

last_plot()

Zeigt die zuletzt erstellte Grafik an.

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

Speichert die zuletzt erstellte Grafik als Datei "plot.png" mit der Größe 5 ft x 5 ft (engl. Maß für Fuß) im Arbeitsverzeichnis. Der Dateityp wird durch die Dateiendung bestimmt.

Geom - Datenwerte werden mittels Geom auf eine Ebene abgebildet. Dessen ästhetischen Eigenschaften beschreiben Variablen. Der Rückgabewert jeder Funktion ist eine Ebene der Grafik.

Eine Variable

Stetig

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

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

a + geom_density(kernel = "gaussian"**)**
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

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

a + geom_freqpoly()
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

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

Diskret

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

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

Grafische Primitive

c <- ggplot(map, aes(long, lat))

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

d <- ggplot(economics, aes(date, unemploy))

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

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

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

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

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

Zwei Variablen

X und Y stetig

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

f + geom_blank()
(praktisch um Intervallgrenzen zu erweitern)

f + geom_jitter()
x, y, alpha, color, fill, shape, size

f + geom_point()
x, y, alpha, color, fill, shape, size

f + geom_quantile()
x, y, alpha, color, linetype, size, weight

f + geom_rug(sides = "bl"**)**
alpha, color, linetype, size

f + geom_smooth(model = lm**)**
x, y, alpha, color, fill, linetype, size, weight

f + geom_text(aes(label = cty)**)**
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

X diskret und Y stetig

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

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

g + geom_boxplot()
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + geom_dotplot(binaxis = "y", stackdir = "center"**)**
x, y, alpha, color, fill

g + geom_violin(scale = "area"**)**
x, y, alpha, color, fill, linetype, size, weight

X und Y diskret

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

h + geom_jitter()
x, y, alpha, color, fill, shape, size

Stetige bivariate Verteilung

i <- ggplot(movies, aes(year, rating))

i + geom_bin2d(binwidth = c(5, 0.5)**)**
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + geom_density2d()
x, y, alpha, colour, linetype, size

i + geom_hex()
x, y, alpha, colour, fill size

Stetige Funktion

j <- ggplot(economics, aes(date, unemploy))

j + geom_area()
x, y, alpha, color, fill, linetype, size

j + geom_line()
x, y, alpha, color, linetype, size

j + geom_step(direction = "hv"**)**
x, y, alpha, color, linetype, size

Abweichungen & Ausreißer

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

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

k + geom_errorbar()
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

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

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

Landkarten

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

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

Drei Variablen

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

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

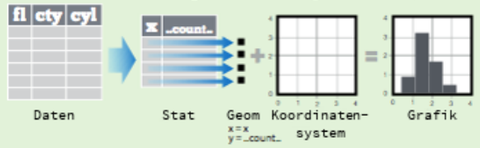
m + geom_raster(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE**)**
x, y, alpha, fill (schnell)

m + geom_tile(aes(fill = z)**)**
x, y, alpha, color, fill, linetype, size (langsam)

Übersetzt von Lucia Gjeltema • rtpanalysts.org

Stat – Alternativen für neue Ebenen

Einige Grafiken stellen eine **Transformation** des ursprünglichen Datensatzes dar. **„Stat"** wird verwendet um solch eine visuelle Statistik zu erzeugen, z. B. **a + geom_bar(stat = "bin")**



Jede Stat generiert weitere Variablen, die mit ästhetischen Eigenschaften verknüpft werden. Diese Variablen haben eine **..Name..** Syntax.

Sowohl **geom_*()** als auch **stat_*()** Funktionen kombinieren eine Stat mit einem Geom um eine Ebene zu erstellen, d.h. das Resultat von **stat_bin(geom="bar")** ist identisch zu **geom_bar(stat="bin")**

Stat Funktion | Ebenen-spezifische Zuordnung | Variable mittels Transformation erstellt

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

Geom für Ebene | Parameter für Stat

1D Verteilungen

a + stat_bin(binwidth = 1, origin = 10**)**
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + stat_bindot(binwidth = 1, binaxis = "x"**)**
x, y, | ..count.., ..ncount..
a + stat_density(adjust = 1, kernel = "gaussian"**)**
x, y, | ..count.., ..density.., ..scaled..

2D Verteilungen

f + stat_bin2d(bins = 30, drop = TRUE**)**
x, y, fill | ..count.., ..density..
f + stat_binhex(bins = 30**)**
x, y, fill | ..count.., ..density..
f + stat_density2d(contour = TRUE, n = 100**)**
x, y, color, size | ..level..

3 Variablen

m + stat_contour(aes(z = z)**)**
x, y, z, order | ..level..
m+ stat_spoke(aes(radius= z, angle = z)**)**
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + stat_summary_hex(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..
m + stat_summary2d(aes(z = z), bins = 30, fun = mean**)**
x, y, z, fill | ..value..

Vergleiche

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

Funktionen

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

allgemeine Anwendung

ggplot() + stat_function(aes(x = -3:3), fun = dnorm, n = 101, args = list(sd=0.5)**)**
x | ..y..
f + stat_identity()
ggplot() + stat_qq(aes(sample=1:100), distribution = qt, dparams = list(df=5)**)**
sample, x, y | ..x.., ..y..
f + stat_sum()
x, y, size | ..size..
f + stat_summary(fun.data = "mean_cl_boot"**)**
f + stat_unique()

Skalen & Maßstäbe

Skalen beeinflussen wie die Daten mit den visuellen ästhetischen Eigenschaften verknüpft werden. Auch benutzerdefinierte Skalen sind möglich.

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

scale_ | ästhetische Eigenschaft anpassen | voreingestellte Skala verwenden | spezifische Argumente einstellen

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")**)**

Wertebereich der Grafik auswählen | Titel der Legende/ Achse | Beschriftung der Legende/ Achse | Unterteilungen der Legende/ Achse

Allgemeine Skalen

Mit jeder ästhetischen Eigenschaft kombinierbar: alpha, color, fill, linetype, shape, size

scale_*_continuous() - bildet stetige Datenwerte ab
scale_*_discrete() - bildet diskrete Datenwerte ab
scale_*_identity() - bildet Datenwerte ab
scale_*_manual(values = c()**)** - bildet Datenwerte mit manuell ausgewählten Einstellungen ab

Maßstäbe für x- und y-Achse

Zu verwenden mit ästhetischen Eigenschaften von x oder y (hier gezeigt für x)

scale_x_date(labels = date_format("%m/%d") , breaks = date_breaks("2 weeks")**)** - x als Datum. Siehe ?strptime für Anzeigeformat.

scale_x_datetime() - x als Datum und Zeit. Verwendet die gleichen Parameter wie scale_x_date().

scale_x_log10() - x in logarithmischer Darstellung mit Basis 10

scale_x_reverse() - verkehrte x-Achse
scale_x_sqrt() - Darstellung von x als Wurzel

Farben und Füllung

Diskret

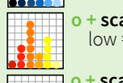
n <- b + geom_bar(aes(fill = fl)**)**

n + scale_fill_brewer(palette = "Blues"**)**
für Farbpaletten: library(RcolorBrewer) display.brewer.all()

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

Stetig

o <- a + 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)**)**
auch: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

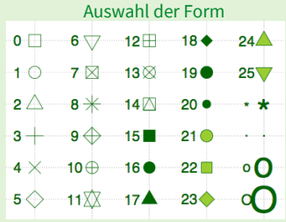
Formen

p <- f + geom_point(aes(shape = fl)**)**

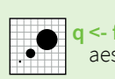
p + scale_shape(solid = FALSE**)**

p + scale_shape_manual(values = c(3:7)**)**
siehe Grafik rechts für manuelle Auswahl

Werte für manuelle Auswahl der Form



Größe



q <- f + geom_point(aes(size = cyl)**)**

q + scale_size_area(max = 6**)**
Wert entspricht der Fläche des Kreises, nicht dem Radius

Koordinatensysteme

r <- b + geom_bar()

r + coord_cartesian(xlim = c(0, 5)**)**
xlim, ylim
Kartesisches Koordinatensystem mit Standardeinstellung

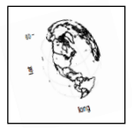
r + coord_fixed(ratio = 1/2**)**
ratio, xlim, ylim
Kartesische Koordinaten mit festem Verhältnis von x- zu y-Einheiten

r + coord_flip()
xlim, ylim
Vertauschte kartesische Kooordinaten

r + coord_polar(theta = "x", direction=1 **)**
theta, start, direction
Polarkoordinaten

r + coord_trans(xtrans = "sqrt"**)**
xtrans, ytrans, limx, limy
Transformierte kartesische Koordinaten. Die Parameter xtrans und ytrans sind Fensterfunktionen (engl. window function).

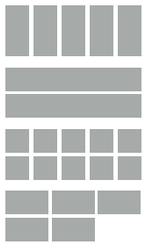
z + coord_map(projection = "ortho", orientation=c(41, -74, 0)**)**
projection, orientation, xlim, ylim
Kartenprojektionen vom „mapproj" Paket (mercator (Standardeinstellung), azequalarea, lagrange, usw.)



Facetten

Facetten terteilen eine Grafik in mehrere Untergrafiken auf, basierend auf den Werten einiger diskreter Variablen.

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



t + facet_grid(. ~ fl)
In Spalten aufteilen, basierend auf fl
t + facet_grid(year ~ .)
In Zeilen aufteilen, basierend auf year
t + facet_grid(year ~ fl)
In Zeilen und Spalten aufteilen
t + facet_wrap(~ fl)
Aufteilung in rechteckige Anordnung

Variable Achsengrenzen in den einzelnen Facetten

t + facet_grid(y ~ x, scales = "free")
Unterschiedliche x- und y-Achsengrenzen in den individuellen Facetten
- **"free_x"** - die x-Achse passt sich an
- **"free_y"** - die y-Achse passt sich an

Anpassen der **Beschriftung** der einzelnen Facetten

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

fl: c	fl: d	fl: e	fl: p	fl: r

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

α^{c} α^{d} α^{e} α^{p} α^{r}

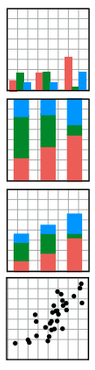
t + facet_grid(. ~ fl, labeller = label_parsed)

c	d	e	p	r

Anpassen der Positionen

Positionen können adaptiert werden um Geomen, die normalerweise an der gleichen Stelle wären, neu anzuordnen.

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



s + geom_bar(position = "dodge")
Elemente nebeneinander anordnen

s + geom_bar(position = "fill")
Elemente übereinander anordnen und ihre Höhe normalisieren

s + geom_bar(position = "stack")
Elemente übereinander anordnen

f + geom_point(position = "jitter")
Weißes Rauschen zu x- und y-Positionen hinzufügen damit Elemente nicht übereinander gedruckt werden

Umformulierung als Funktion mit den Parametern Breite (engl. **width**) und Höhe (engl. **height**) ist möglich

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

Beschriftung

t + ggtitle("Neuer Bildtitel"**)**
Titel über der Grafik
t + xlab("Neuer x-Achsentitel"**)**
Beschriftung der x-Achse
t + ylab("Neuer y-Achsentitel"**)**
Beschriftung der y-Achse
t + labs(title =" Titel neu", x = „x neu", y = "y neu"**)**
Alle oben genannten Beschriftungen zusammen

Skalen-Funktionen werden verwendet um Legenden-beschriftungen zu ändern

Legende

t + theme(legend.position = "bottom"**)**
Legende anordnen mit "bottom", "top", "left", "right"
t + guides(color = "none"**)**
Art der Legende auswählen für ästhetische Eigenschaften: colorbar, legend oder none (keine Legende)
t + scale_fill_discrete(name = "Titel", labels = c("A", "B", "C"**)**
Titel und Beschriftung der Legende via Skalen-Funktion

Themen

r + theme_bw()
Weißer Hintergrund mit Rasterlinien

r + theme_classic()
Weißer Hintergrund, keine Rasterlinien

r + theme_grey()
Grauer Hintergrund (Standardeinstellung)

r + theme_minimal()
Minimal-Thema

ggthemes – Paket mit weiteren ggplot2-Themen

Zoom

Ohne Abschneiden (bevorzugt)

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

Mit Abschneiden (entfernt Daten außerhalb)

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

Übersetzt von Lucia Gjeltema • rtpanalysts.org