Technische
Hochschule
Wildau [FH]
*Technical University
of Applied Sciences*

Einführung in Graphiken in R
mit `ggplot2`

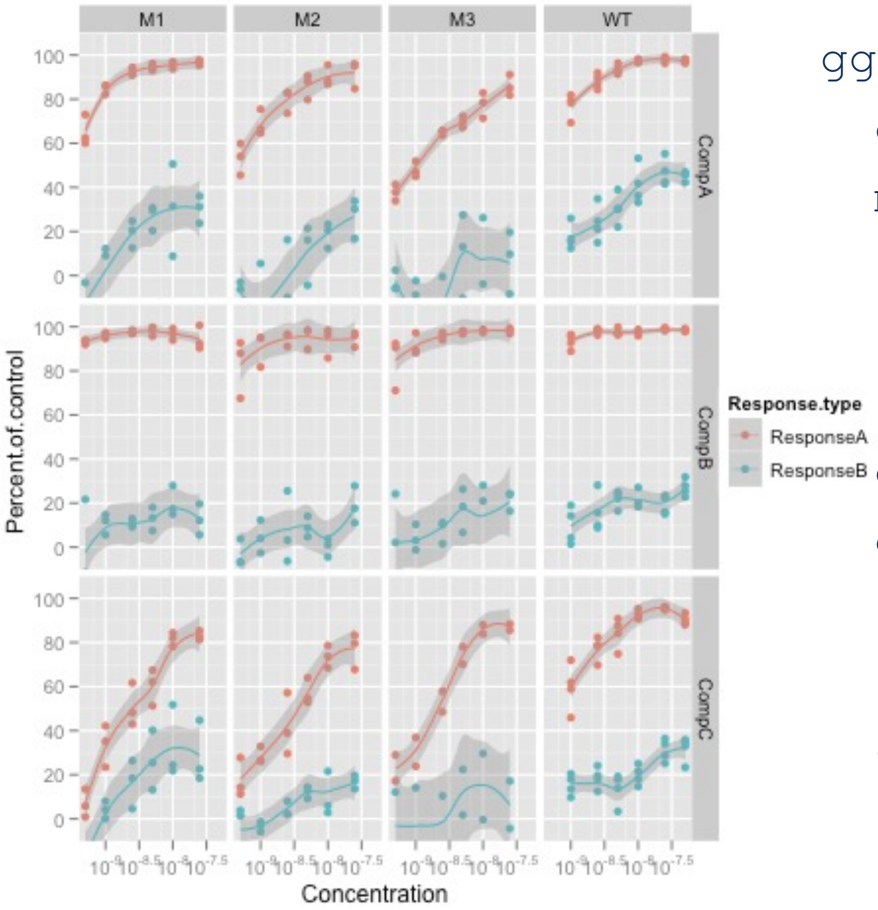
Prof. Dr. Rainer Stollhoff

Vgl.
R for Data Science, Grolemund & Wickham,
http://r4ds.had.co.nz/exploratory-data-analysis.html

- Die folgenden Übersichten und Dokumentation stammen aus dem *Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC , https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf*

- Der Code wurde mit Hilfe des ggplot2 Package erstellt: https://CRAN.R-project.org/package=ggplot2

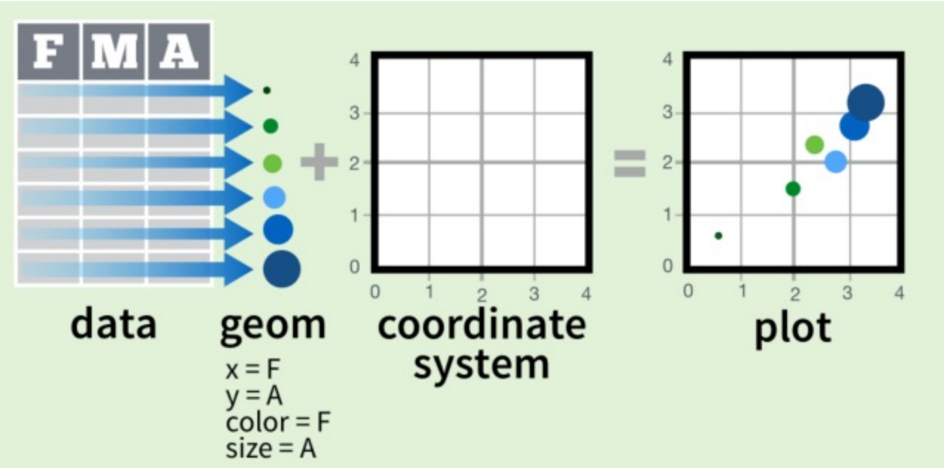
Grammar of Graphics - ggplot

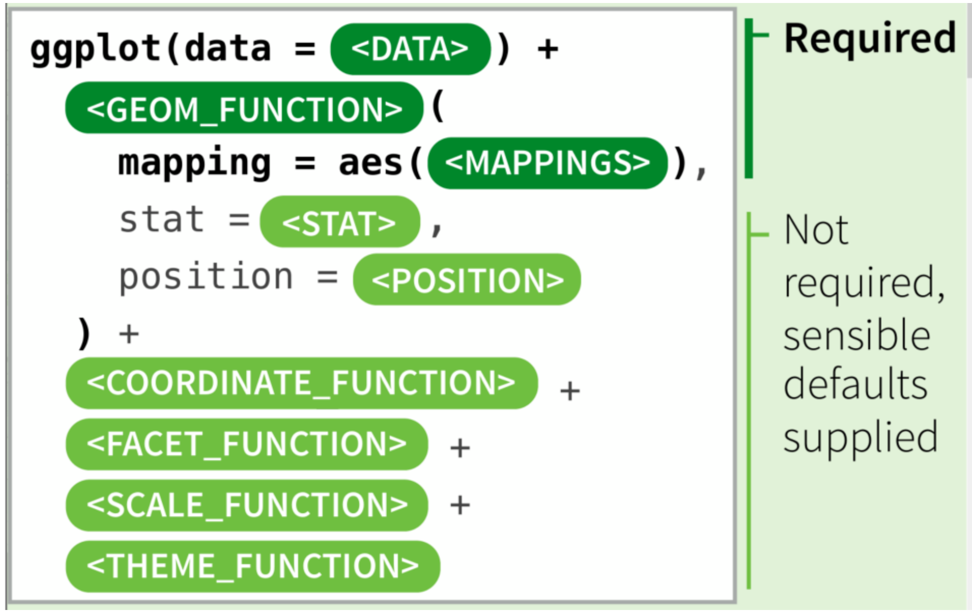


```
ggplot(
  data=screening_data,
  mapping = aes(
    x=Concentration,
    y= Percent.of.control
    colour=Response.type)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10() +
  facet_grid(Compound ~ Cell.line) +
  coord_cartesian(ylim=c(-10, 110))
```

In der Grammar of Graphics ähnelt eine Beschreibung der Graphik in (englischer) Sprache dem R-Code zum Erzeugen der Graphik

Grammar





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

Required

Not required, sensible defaults supplied

- **Daten** werden geplottet
  - mit **Abbildungsparametern** wie
    - **x- und y-Achsen**,
    - Farben, Formen, Größen
- durch eine **geometrische Funktion**
  - aggregiert zu bestimmten **Statistiken**
  - angezeigt an unterschiedlichen **Positionen**
- modifiziert durch
  - **besondere Koordinatensystemee**
  - **Anordnung als Gruppenplot**
  - **Achsenskalierung**
  - …

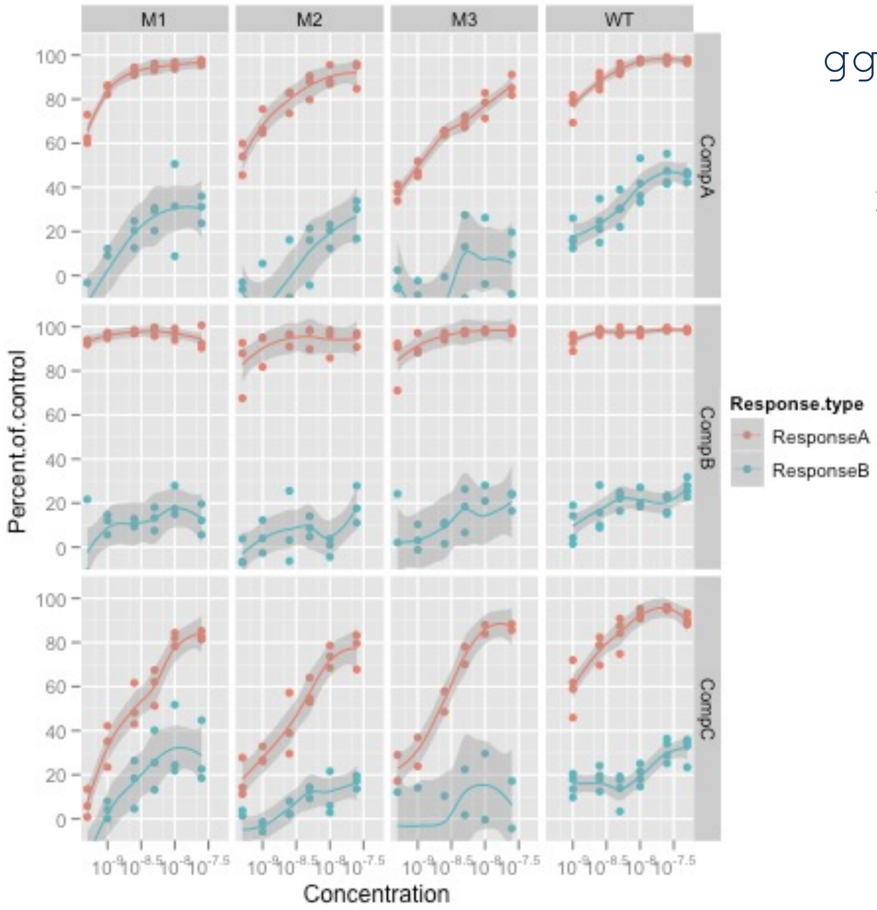
Data visualization with ggplot2 : : CHEAT SHEET , *Posit Software, PBC* , *https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf*

Grammar

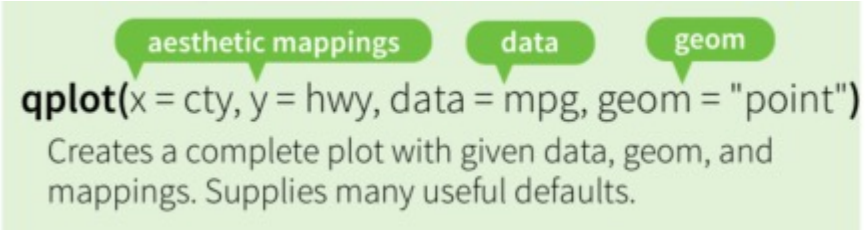
```
ggplot(
  data=screening_data,
  mapping = aes(
    x=Concentration,
    y= Percent.of.control
    colour=Response.type)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10() +
  facet_grid(Compound ~
Cell.line) +
  coord_cartesian(ylim=c(-10,
110))
```

- **Daten** werden geplottet
  - mit **Abbildungsparametern** wie
    - **x- und y-Achsen**,
    - Farben, Formen, Größen
- durch eine **geometrische Funktion**
  - aggregiert zu bestimmten **Statistiken**
  - angezeigt an unterschiedlichen **Positionen**
- modifiziert durch
  - **besondere Koordinatensystemee**
  - **Anordnung als Gruppenplot**
  - **Achsenskalierung**

Grammar



```
ggplot(
  data=screening_data,
  mapping = aes(
    x=Concentration,
    y= Percent.of.control
    colour=Response.type)) +
  geom_point() +
  geom_smooth() +
  scale_x_log10() +
  facet_grid(Compound ~ Cell.line) +
  coord_cartesian(ylim=c(-10, 110))
```

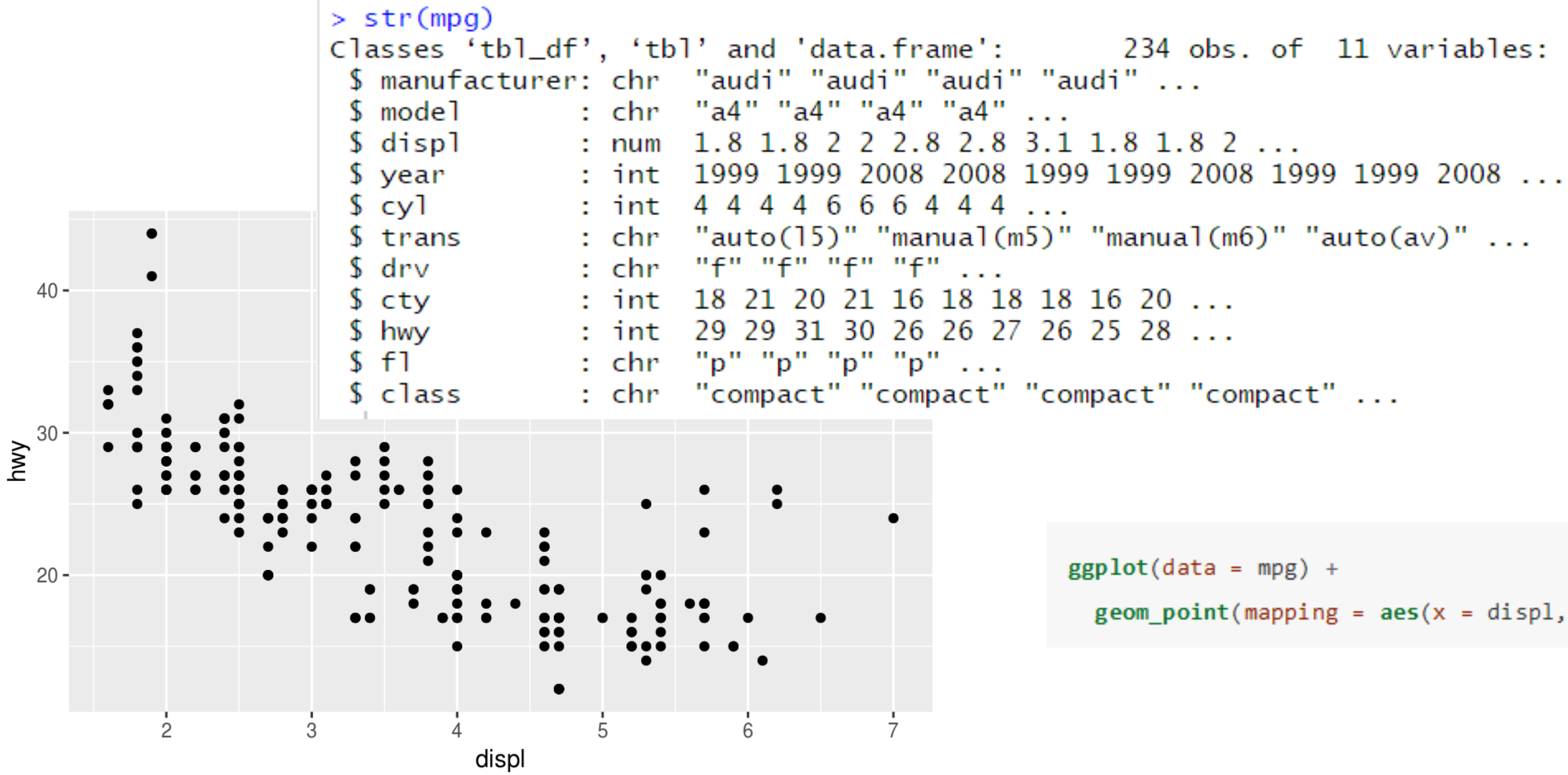

aesthetic mappings
data
geom

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

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC , https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

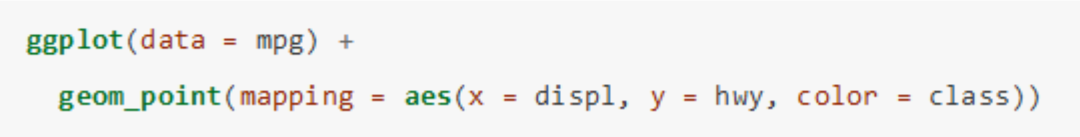
Basics



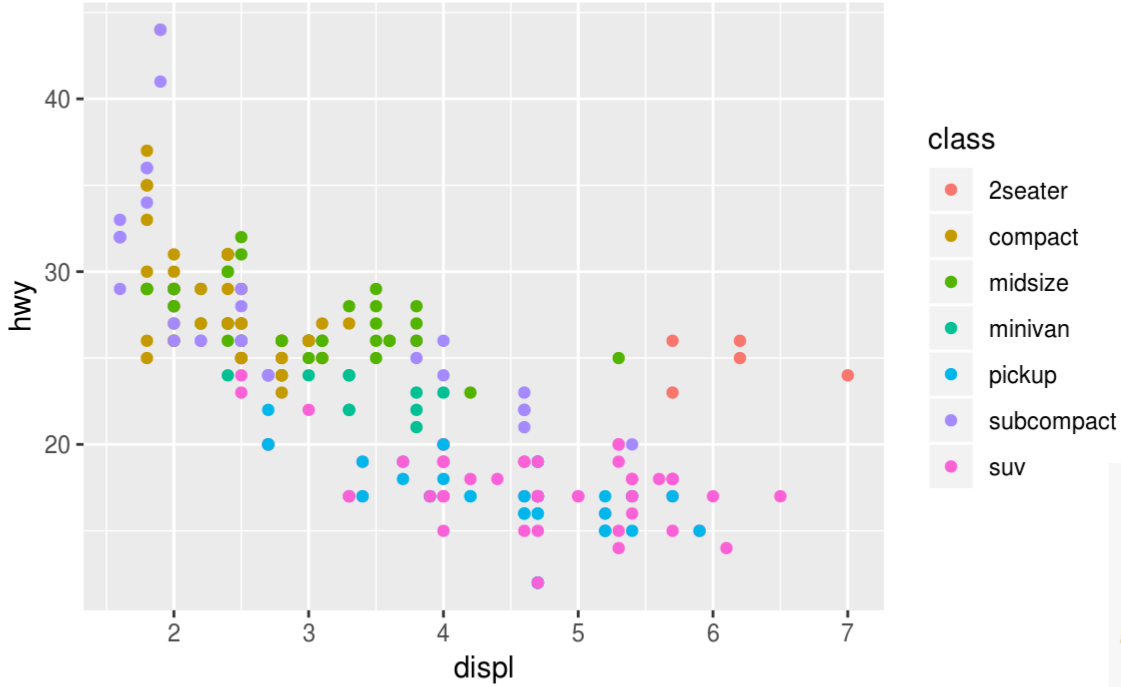
```
> str(mpg)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame':	234 obs. of  11 variables:
 $ manufacturer: chr  "audi" "audi" "audi" "audi" ...
 $ model       : chr  "a4" "a4" "a4" "a4" ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : chr  "auto(l5)" "manual(m5)" "manual(m6)" "auto(av)" ...
 $ drv         : chr  "f" "f" "f" "f" ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : chr  "p" "p" "p" "p" ...
 $ class       : chr  "compact" "compact" "compact" "compact" ...
```

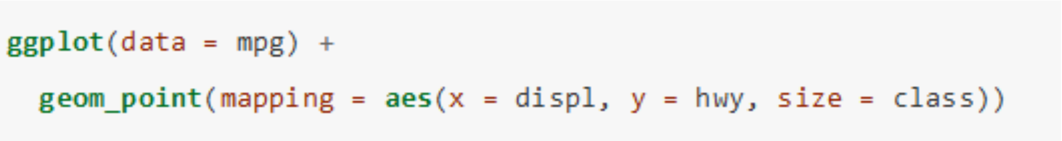
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

Aesthetics

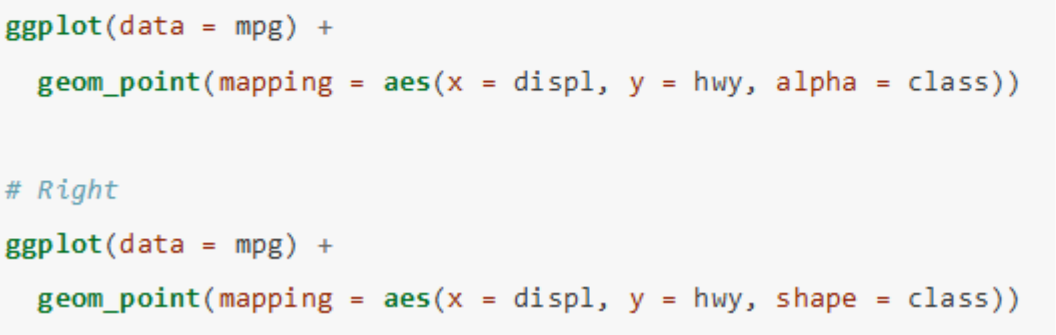


```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



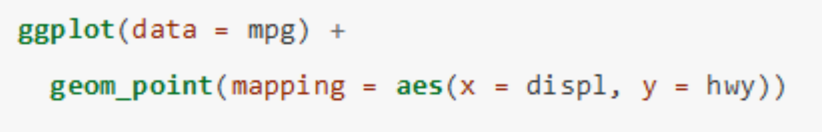


```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, size = class))
```

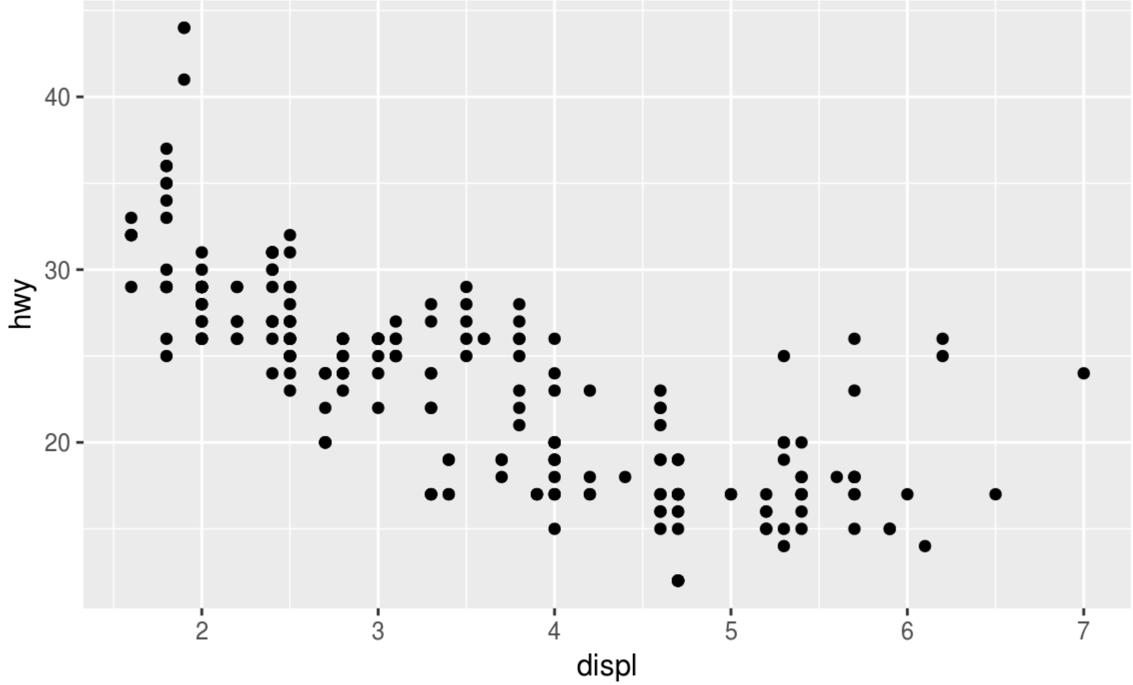


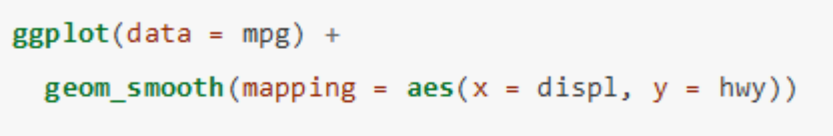
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, alpha = class))

# Right
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, shape = class))
```

Geometric functions/objects

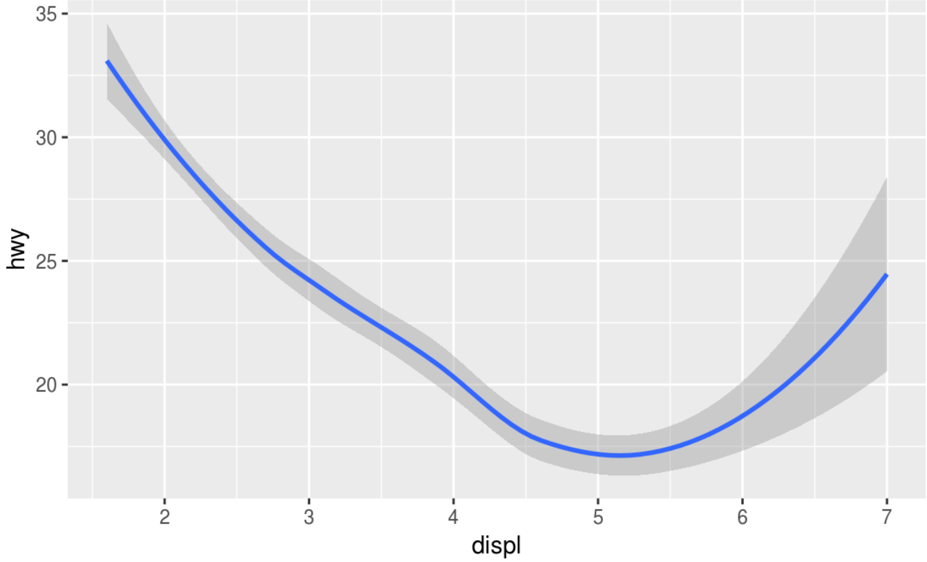


```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy))
```

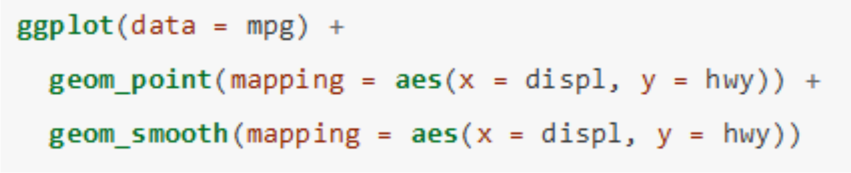




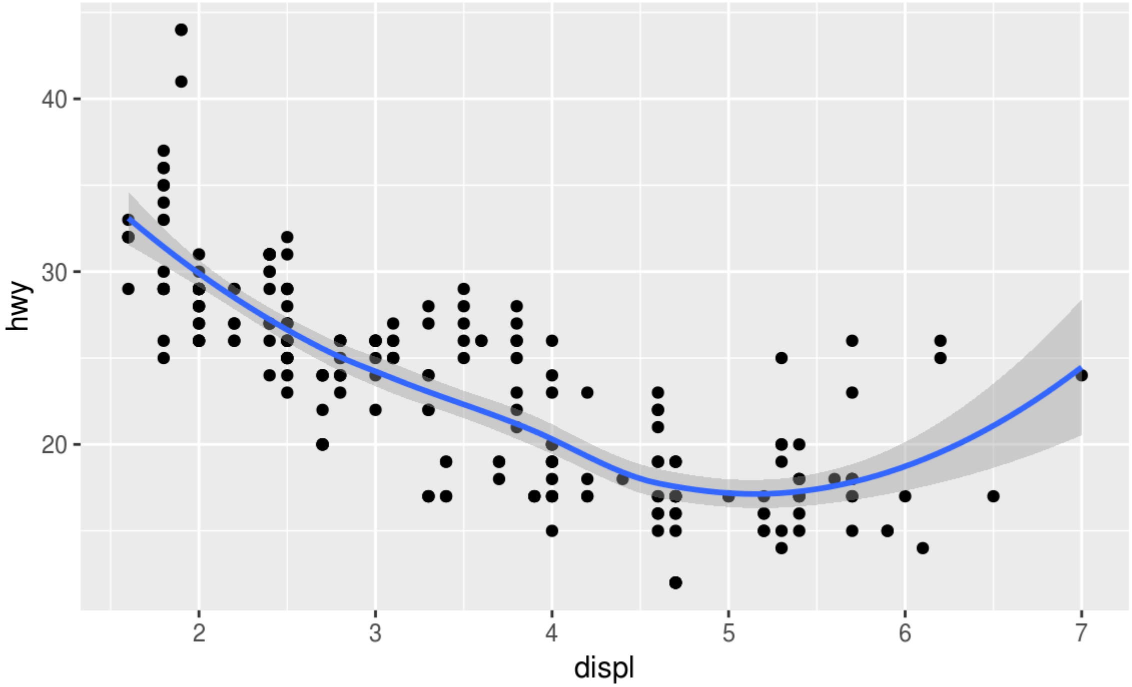
```
ggplot(data = mpg) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```

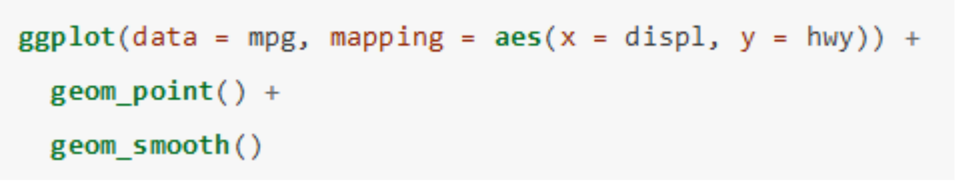


Geometric functions/objects



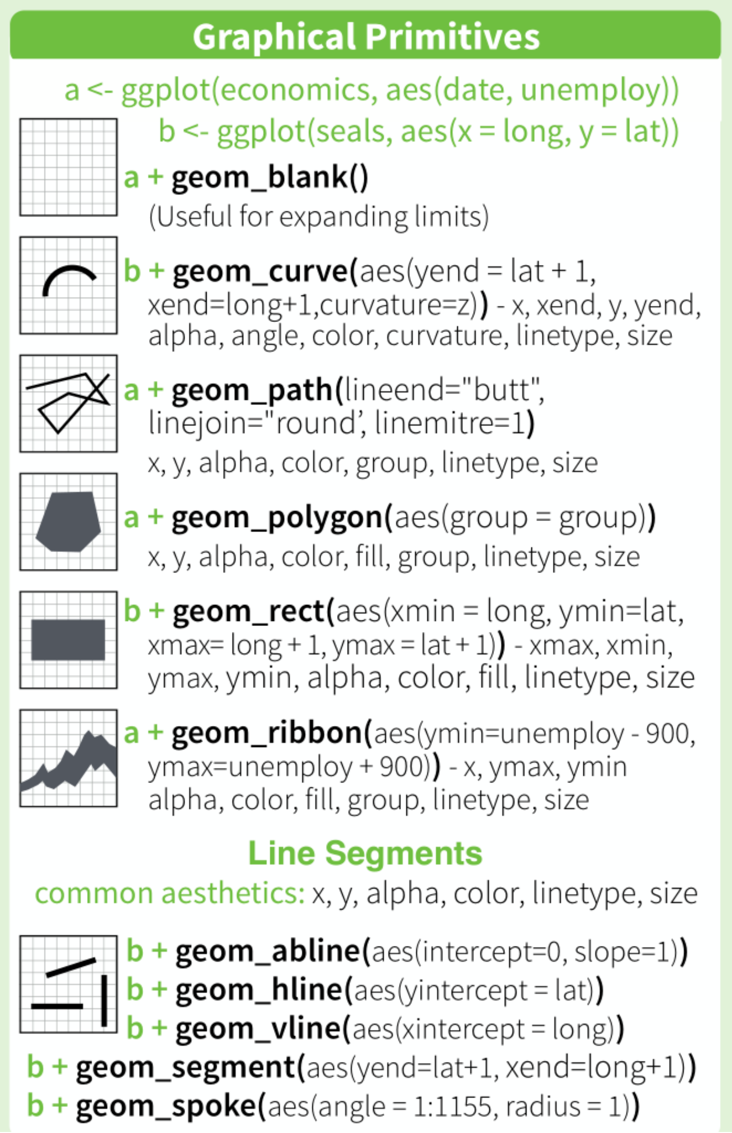
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  geom_smooth(mapping = aes(x = displ, y = hwy))
```





```
ggplot(data = mpg, mapping = aes(x = displ, y = hwy)) +
  geom_point() +
  geom_smooth()
```

Geometric functions/objects



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

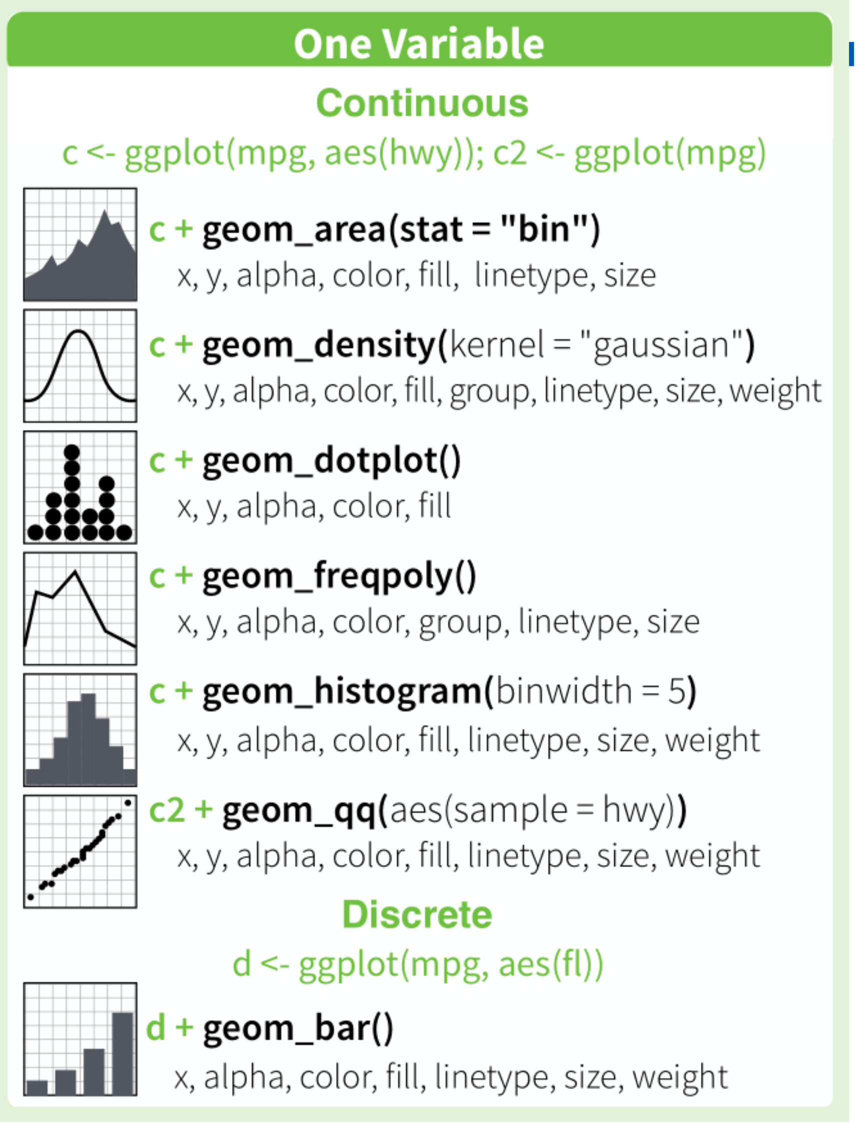
b + **geom_rect(**aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)**)** - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

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

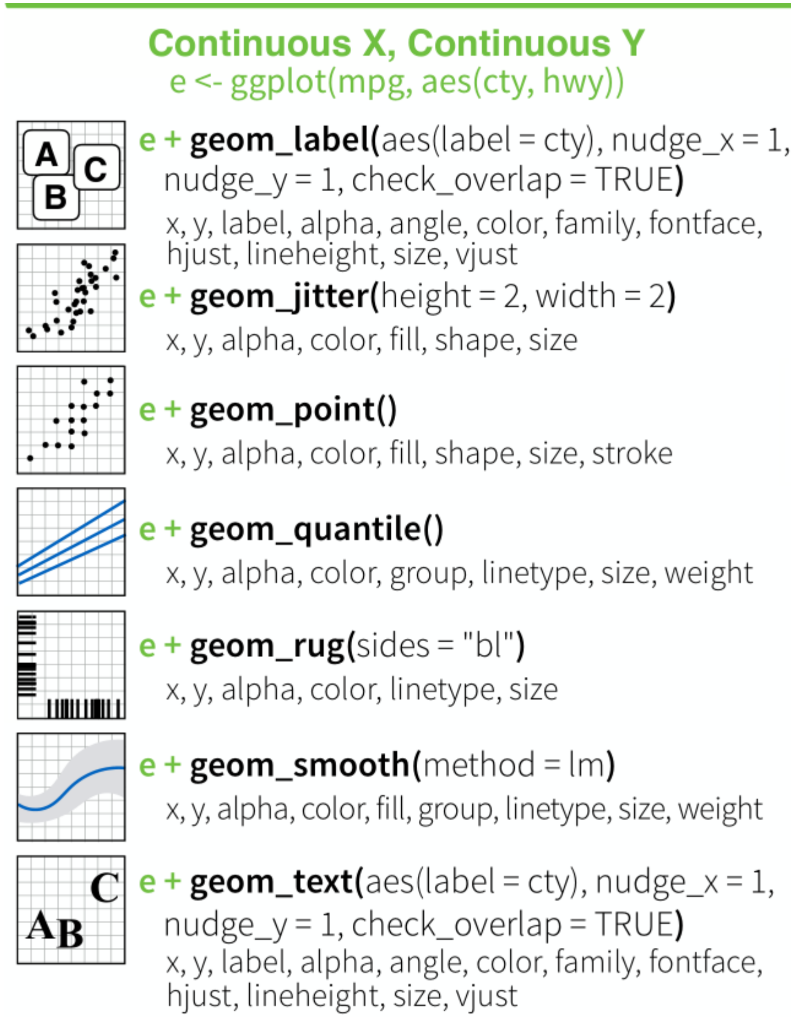
Discrete

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

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

Data visualization with ggplot2 : : CHEAT SHEET , *Posit Software, PBC* ,
https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

Geometric functions/objects



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

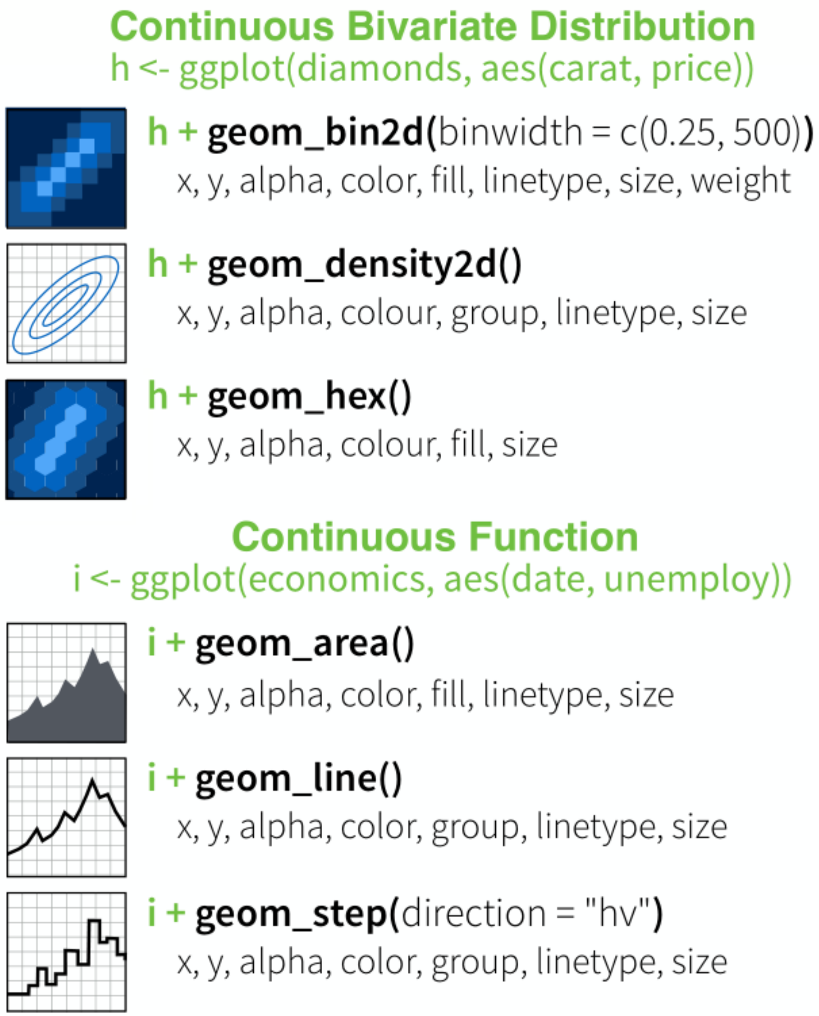
e + **geom_point()**
x, y, alpha, color, fill, shape, size, stroke

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



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))

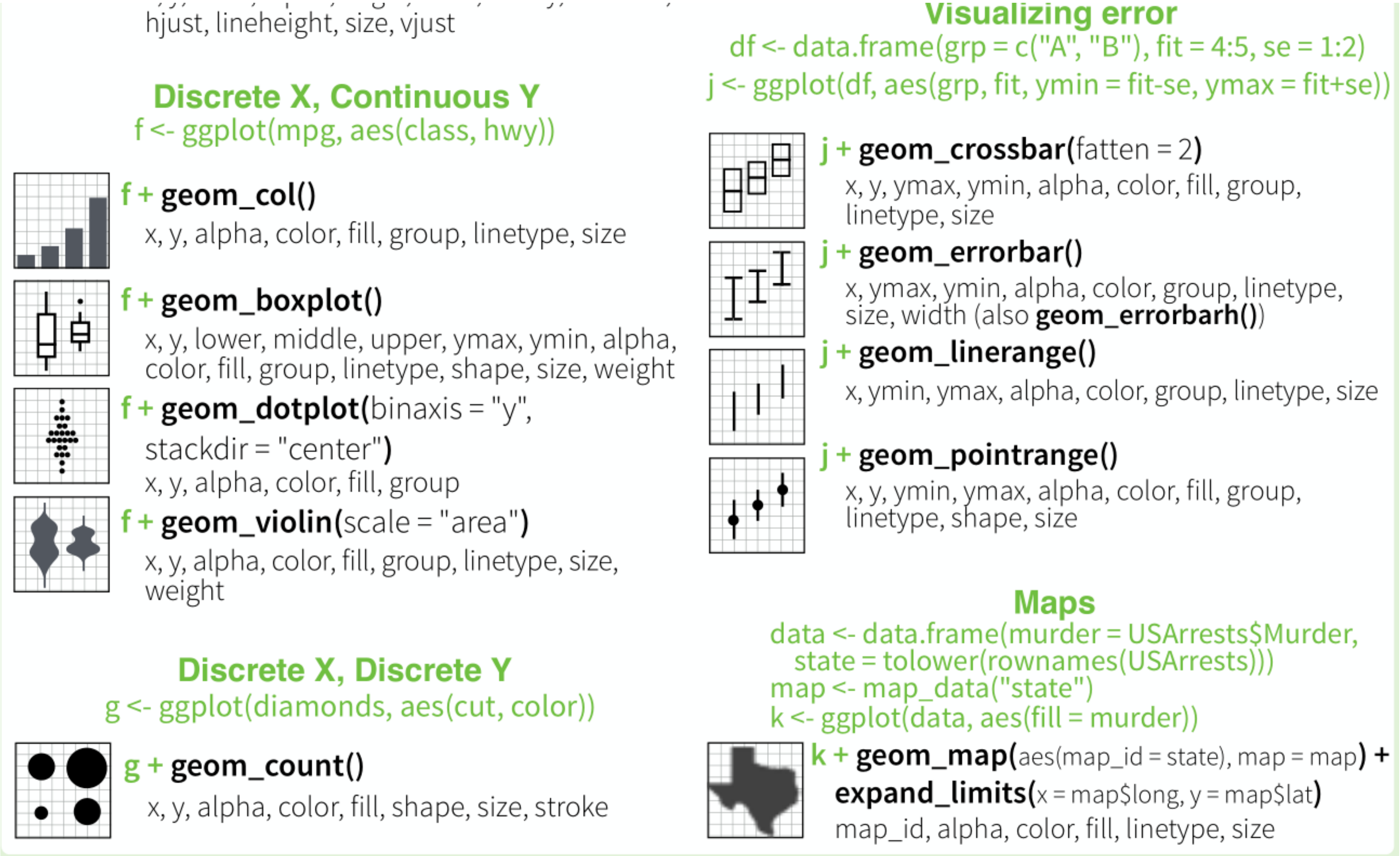
i + **geom_area()**
x, y, alpha, color, fill, linetype, size

i + **geom_line()**
x, y, alpha, color, group, linetype, size

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

*Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC ,
https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf*

Geometric functions/objects



hjust, lineheight, 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, 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

Visualizing error

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 (also **geom_errorbarh()**)

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

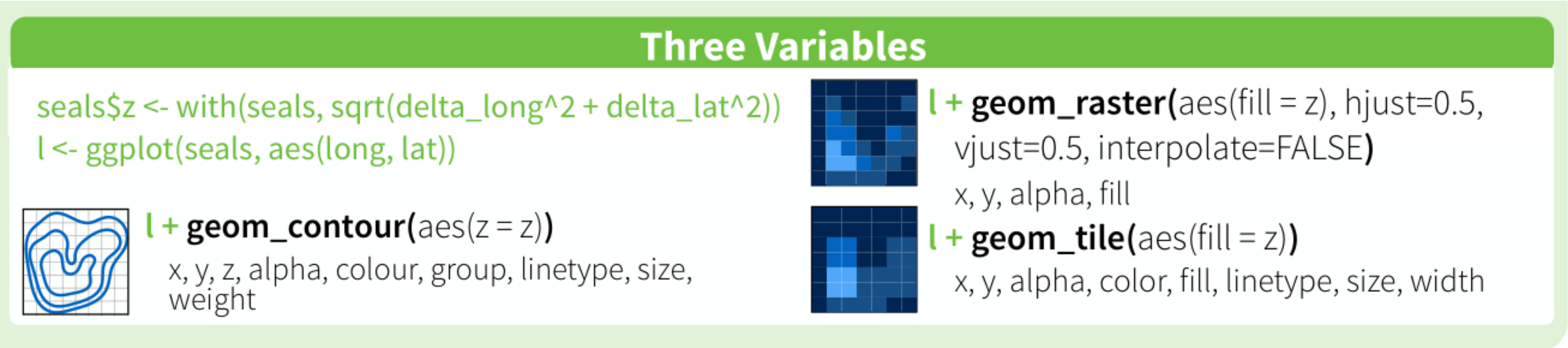
Maps

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

Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC ,
https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

Geometric functions/objects



Three Variables

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

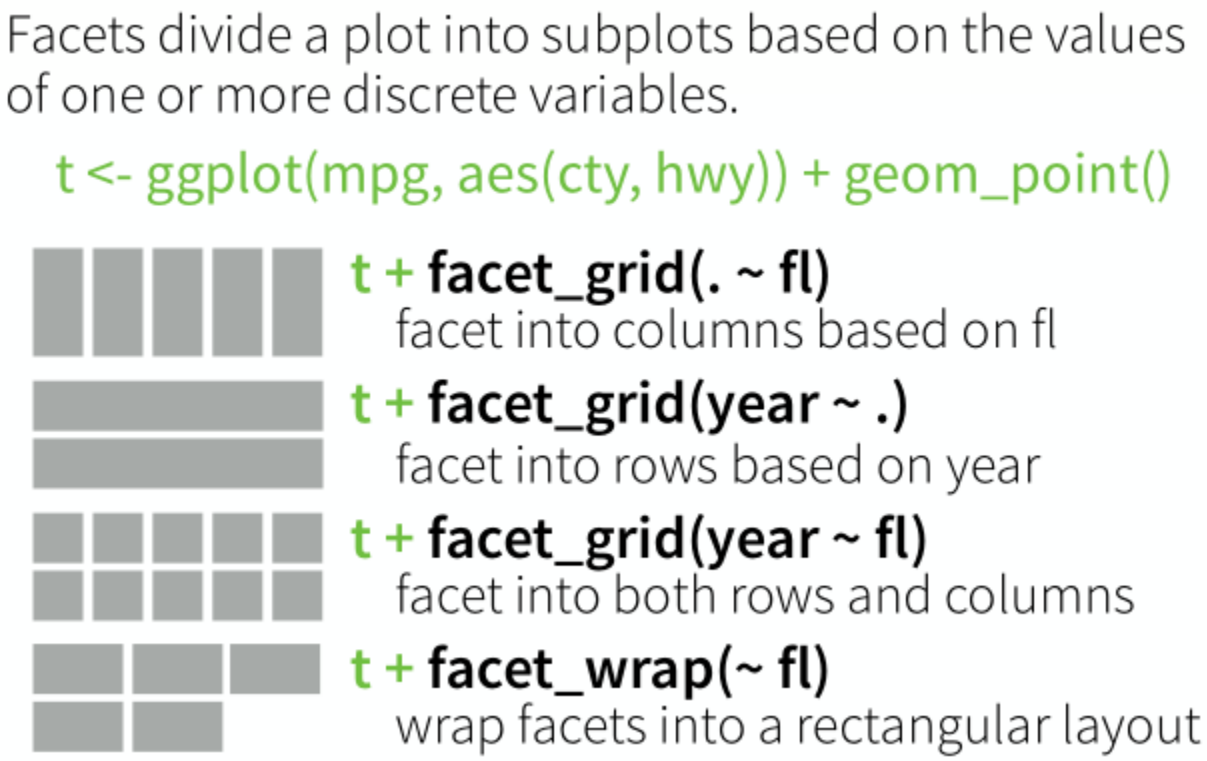
l + **geom_contour(**aes(z = z)**)**
x, y, z, alpha, colour, group, linetype, size, weight

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

Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC ,
https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

Facets



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

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

t + facet_grid(. ~ fl)
facet into columns based on fl

t + facet_grid(year ~ .)
facet into rows based on year

t + facet_grid(year ~ fl)
facet into both rows and columns

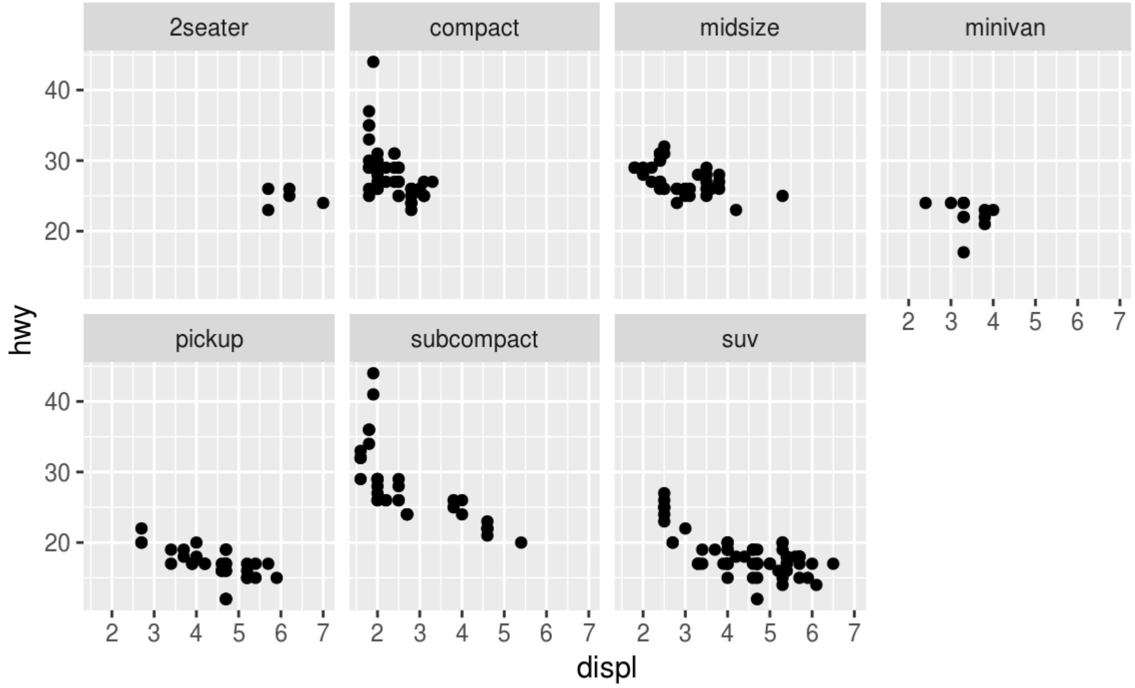
t + facet_wrap(~ fl)
wrap facets into a rectangular layout

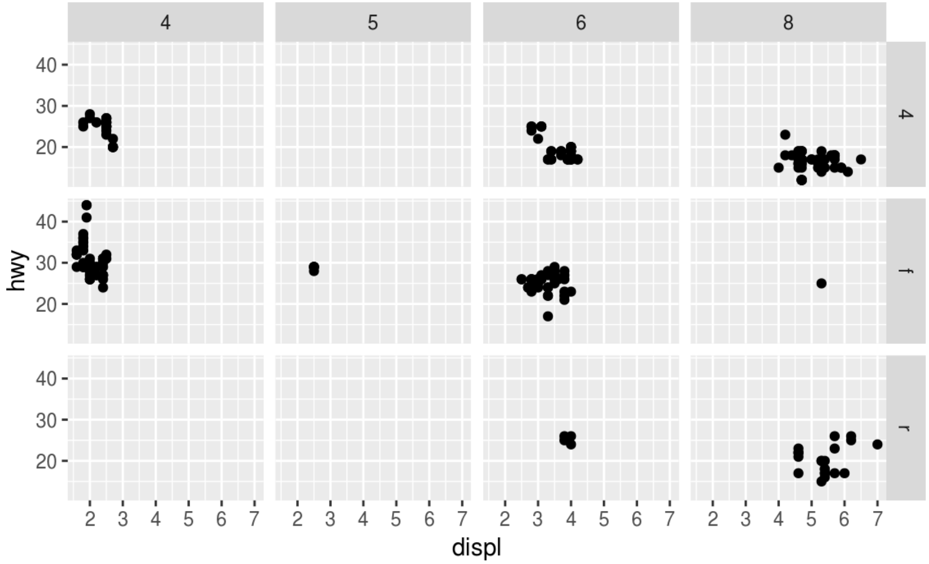
Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC , https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

Facets

```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_wrap(~ class, nrow = 2)
```

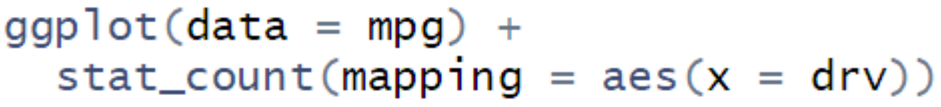
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy)) +
  facet_grid(drv ~ cyl)
```



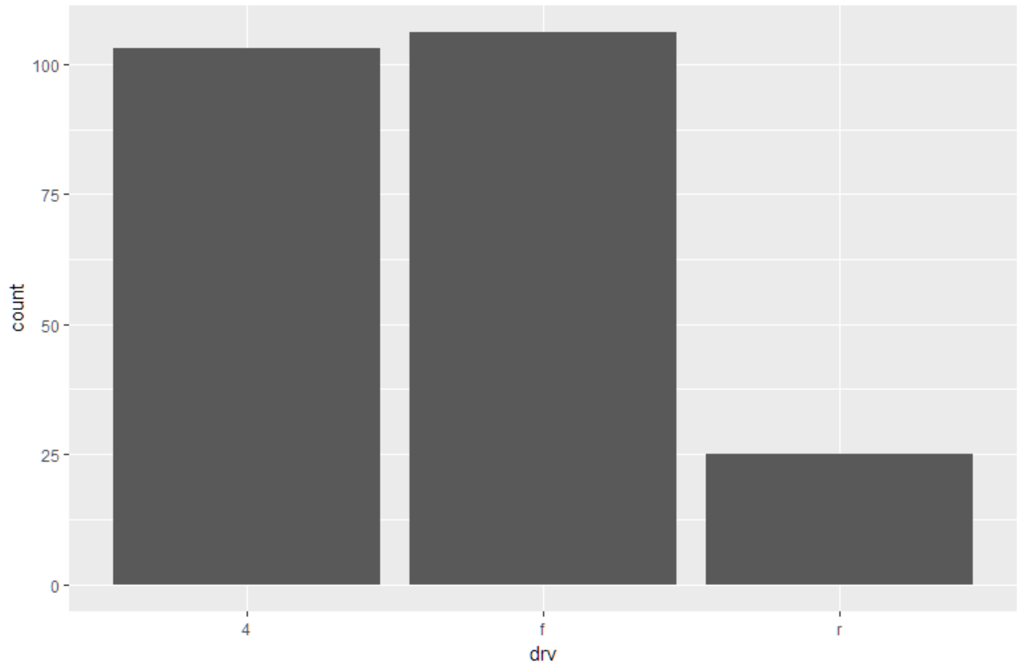


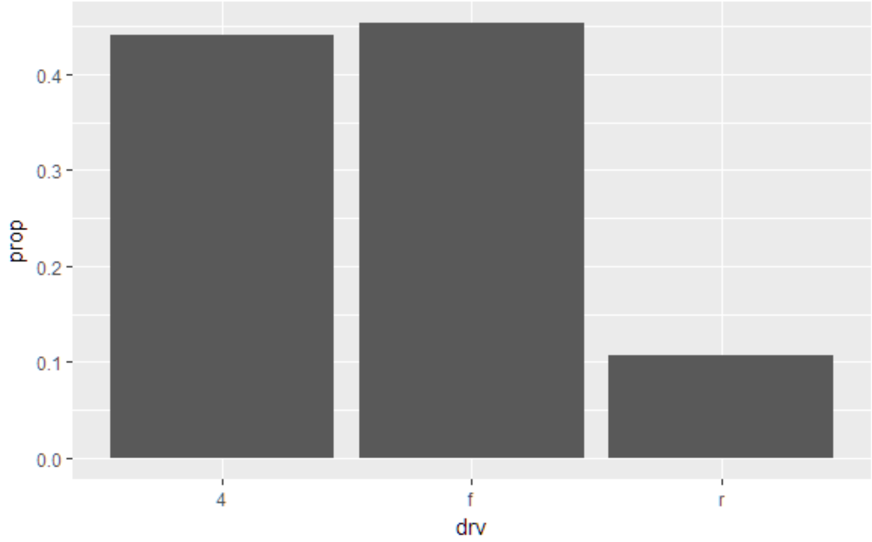
Statistics

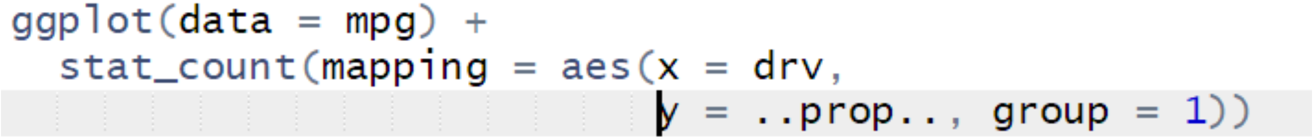
`Stat_count()` verwendet als default die Zählstatistik



```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv))
```



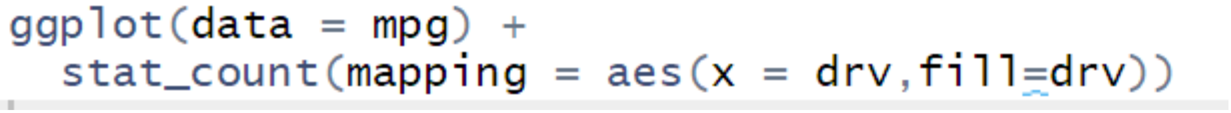




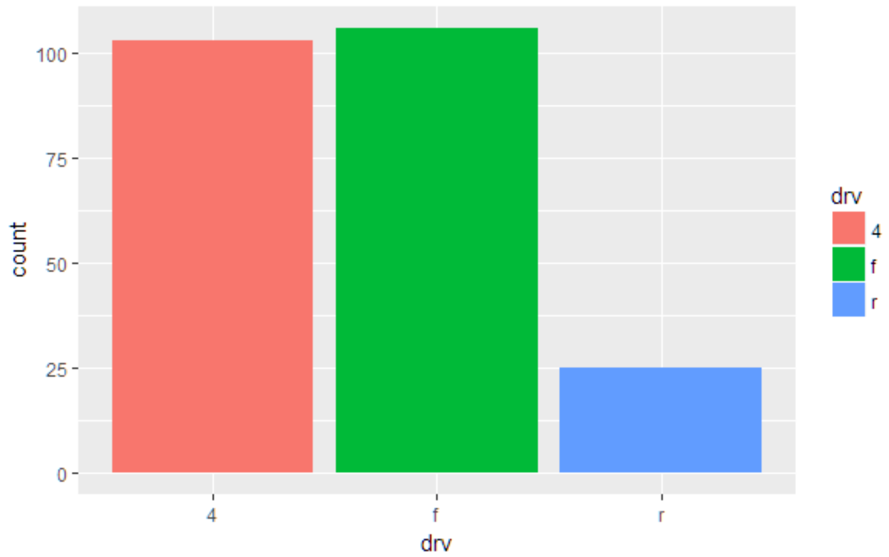
```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv,
                           y = ..prop.., group = 1))
```

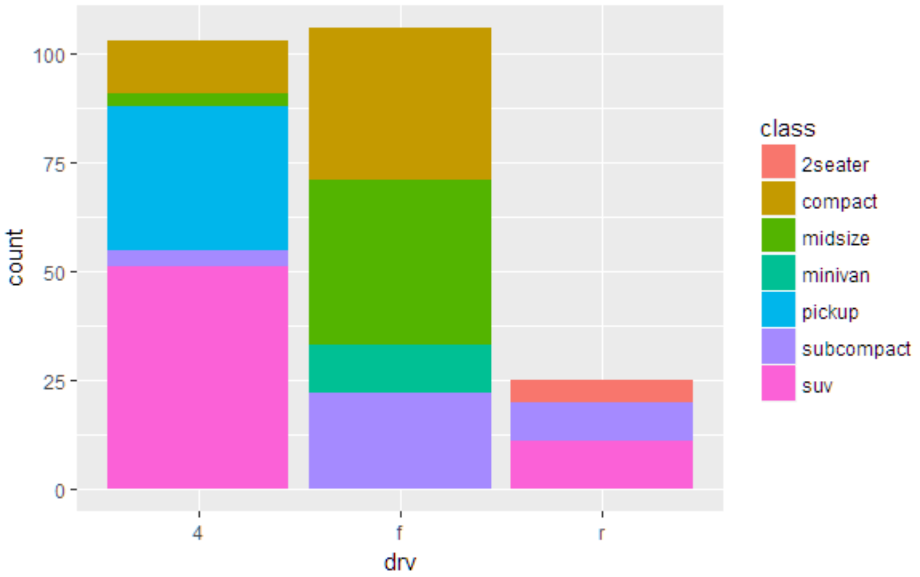
Alternative: Relativer Anteil

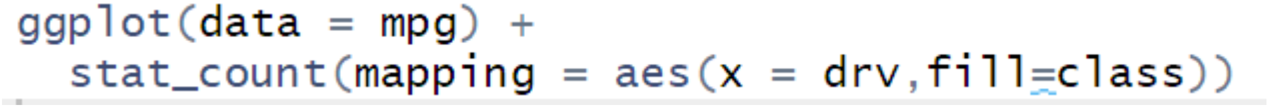
Statistics



```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv,fill=drv))
```

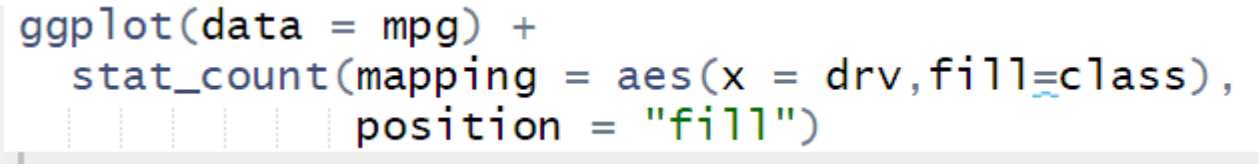




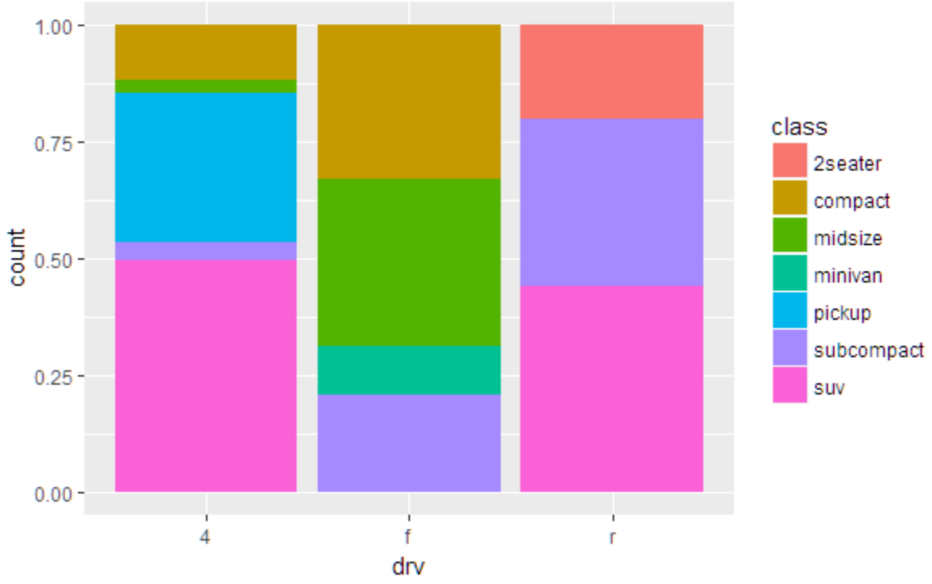


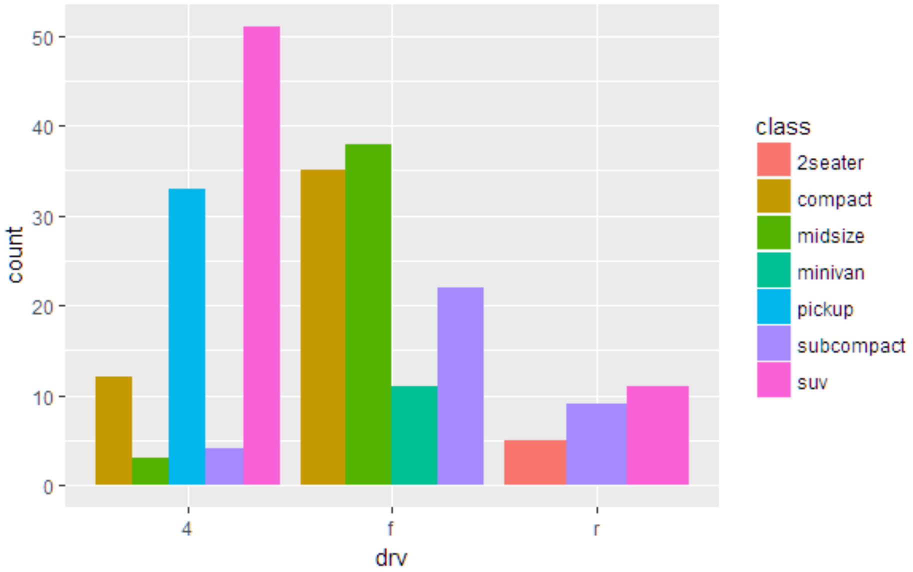
```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv,fill=class))
```

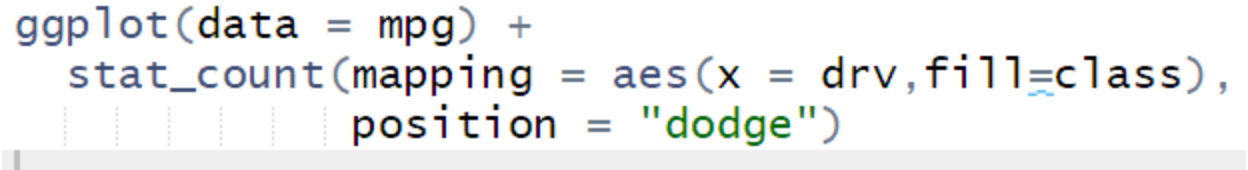
Statistics



```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv,fill=class),
             position = "fill")
```

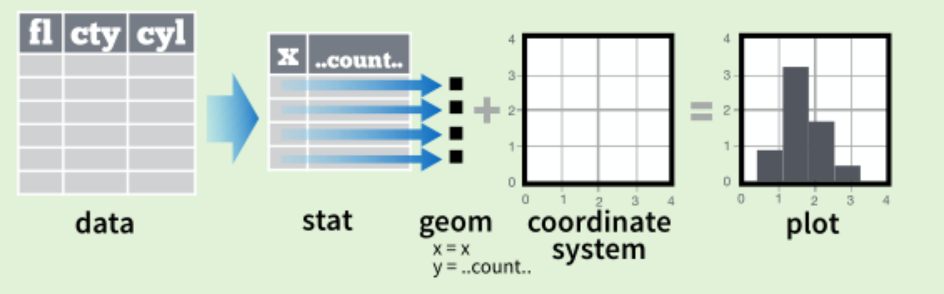




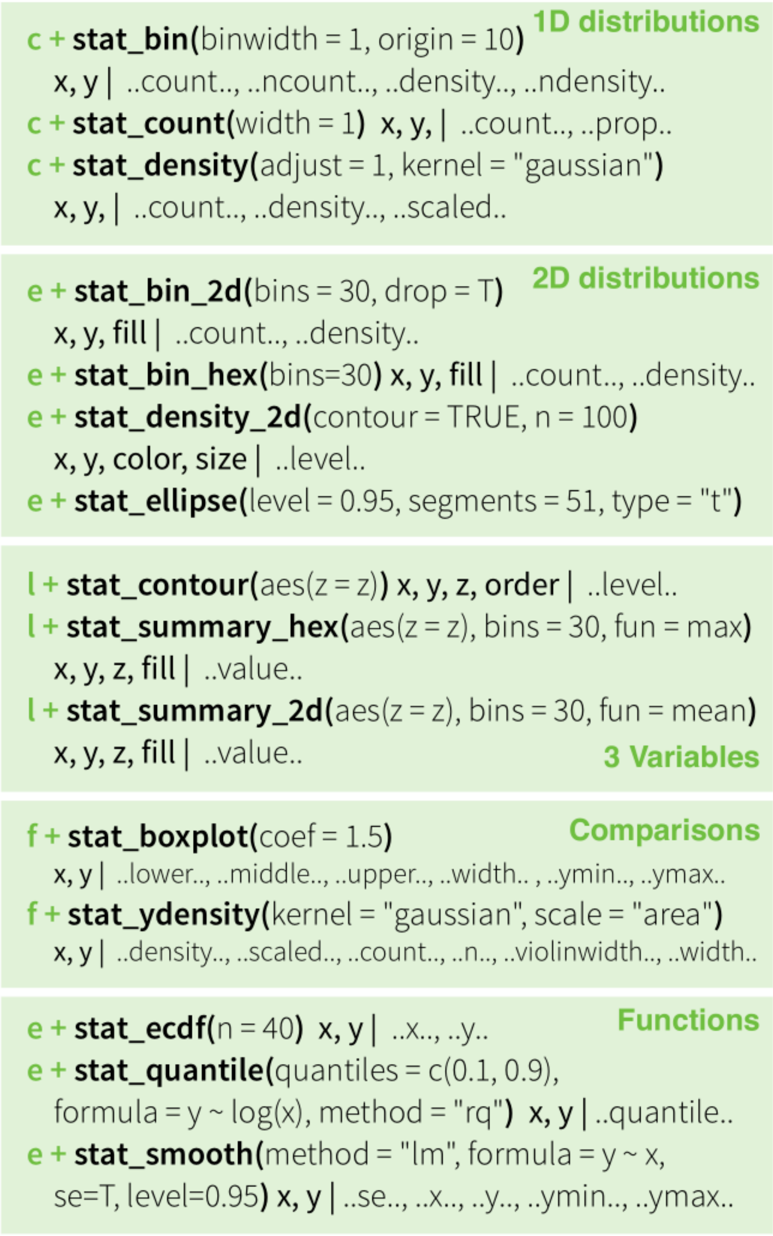


```
ggplot(data = mpg) +
  stat_count(mapping = aes(x = drv,fill=class),
             position = "dodge")
```

Statistics



```
geom_bar(stat="count")
stat_count(geom="bar")
```



1D distributions

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..

2D distributions

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 | ..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..

3 Variables

Comparisons

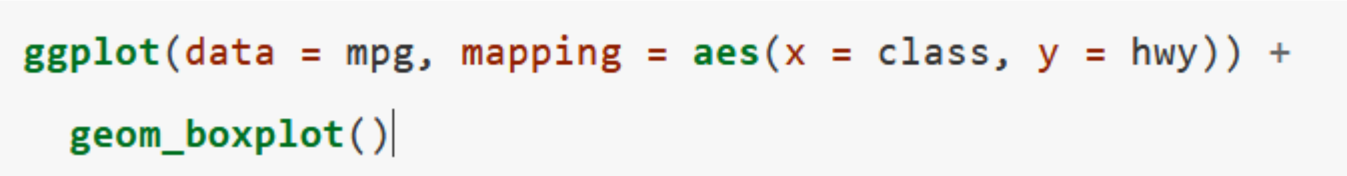
f + **stat_boxplot(**coef = 1.5**)**
x, y | ..lower.., ..middle.., ..upper.., ..width.. , ..ymin.., ..ymax..
f + **stat_ydensity(**kernel = "gaussian", scale = "area"**)**
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

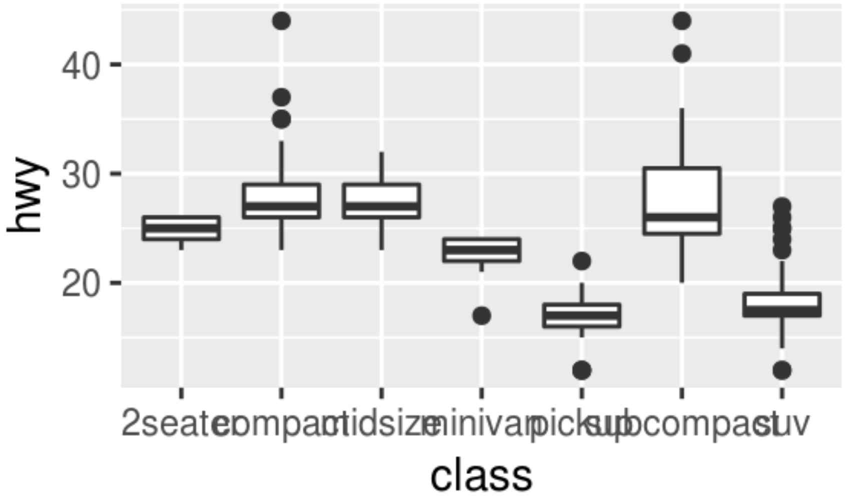
e + **stat_ecdf(**n = 40**)** x, y | ..x.., ..y..
e + **stat_quantile(**quantiles = c(0.1, 0.9),
formula = y ~ log(x), method = "rq"**)** x, y | ..quantile..
e + **stat_smooth(**method = "lm", formula = y ~ x,
se=T, level=0.95**)** x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

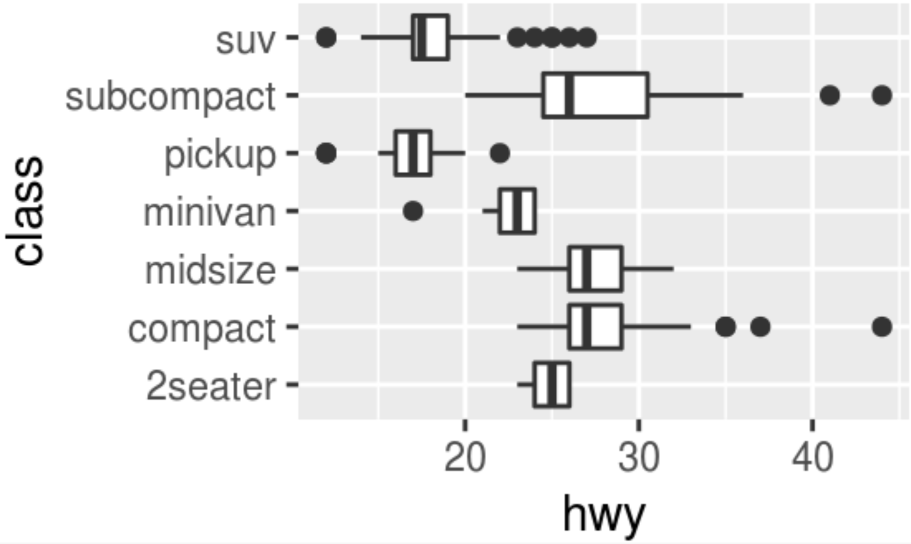
Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC ,
https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf

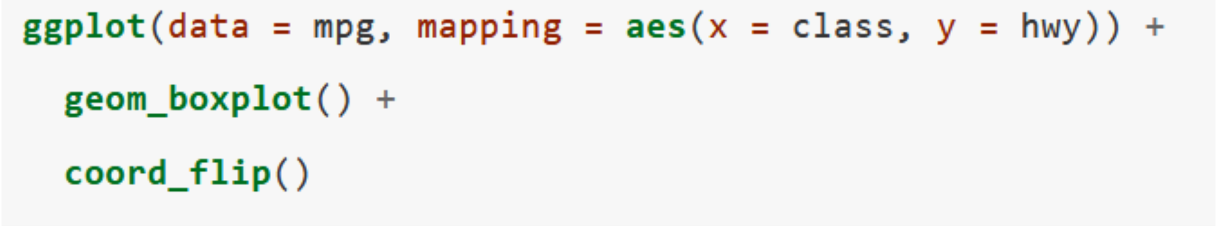
Koordinatensysteme



```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot()
```

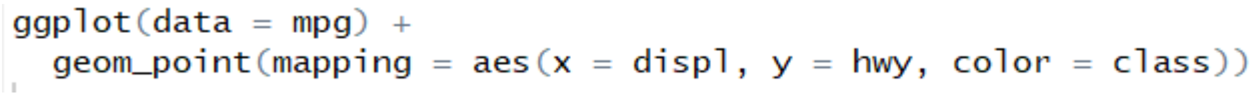




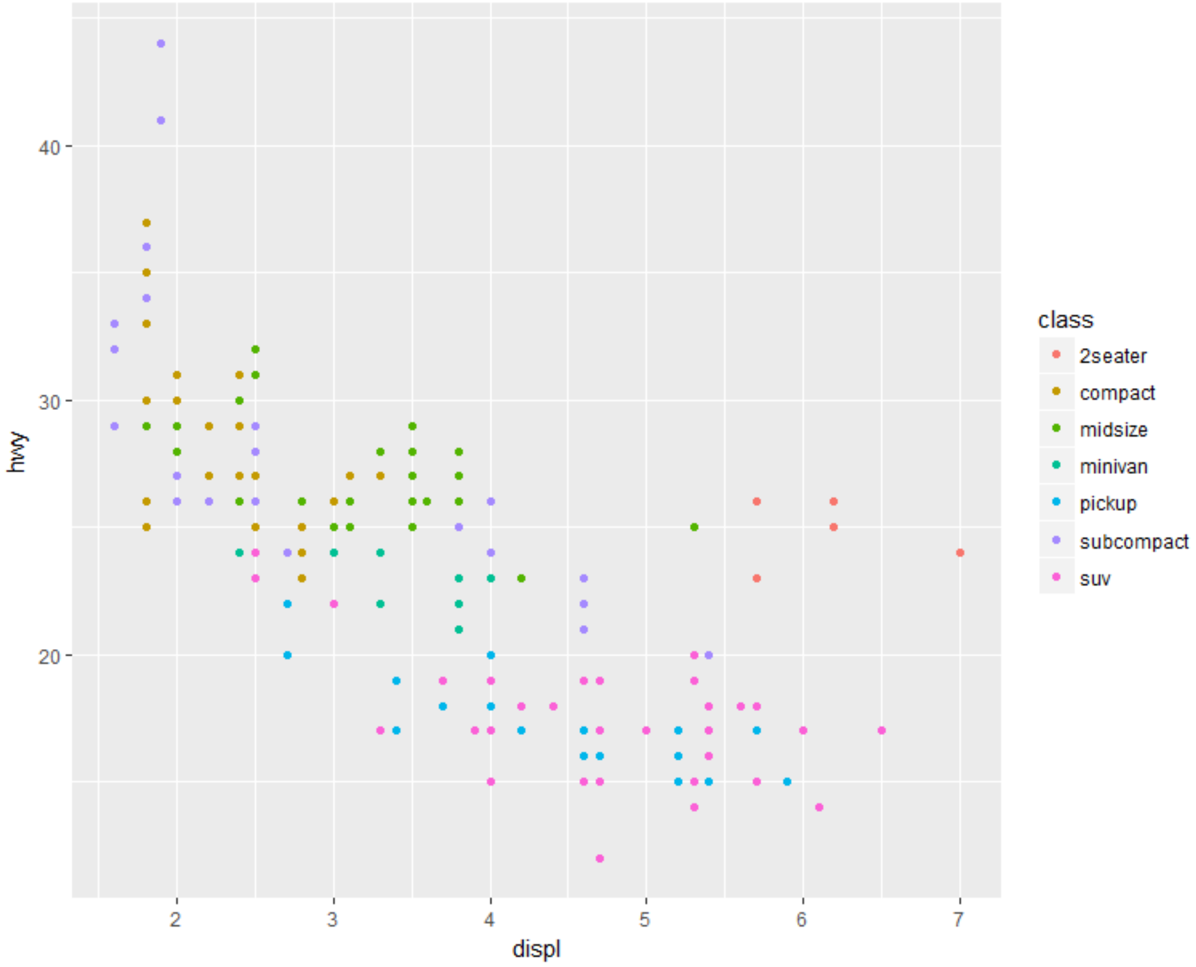


```
ggplot(data = mpg, mapping = aes(x = class, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```

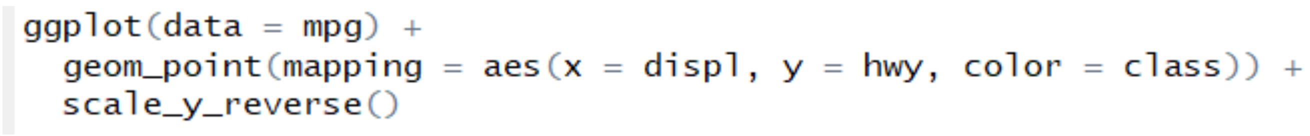
Skalen



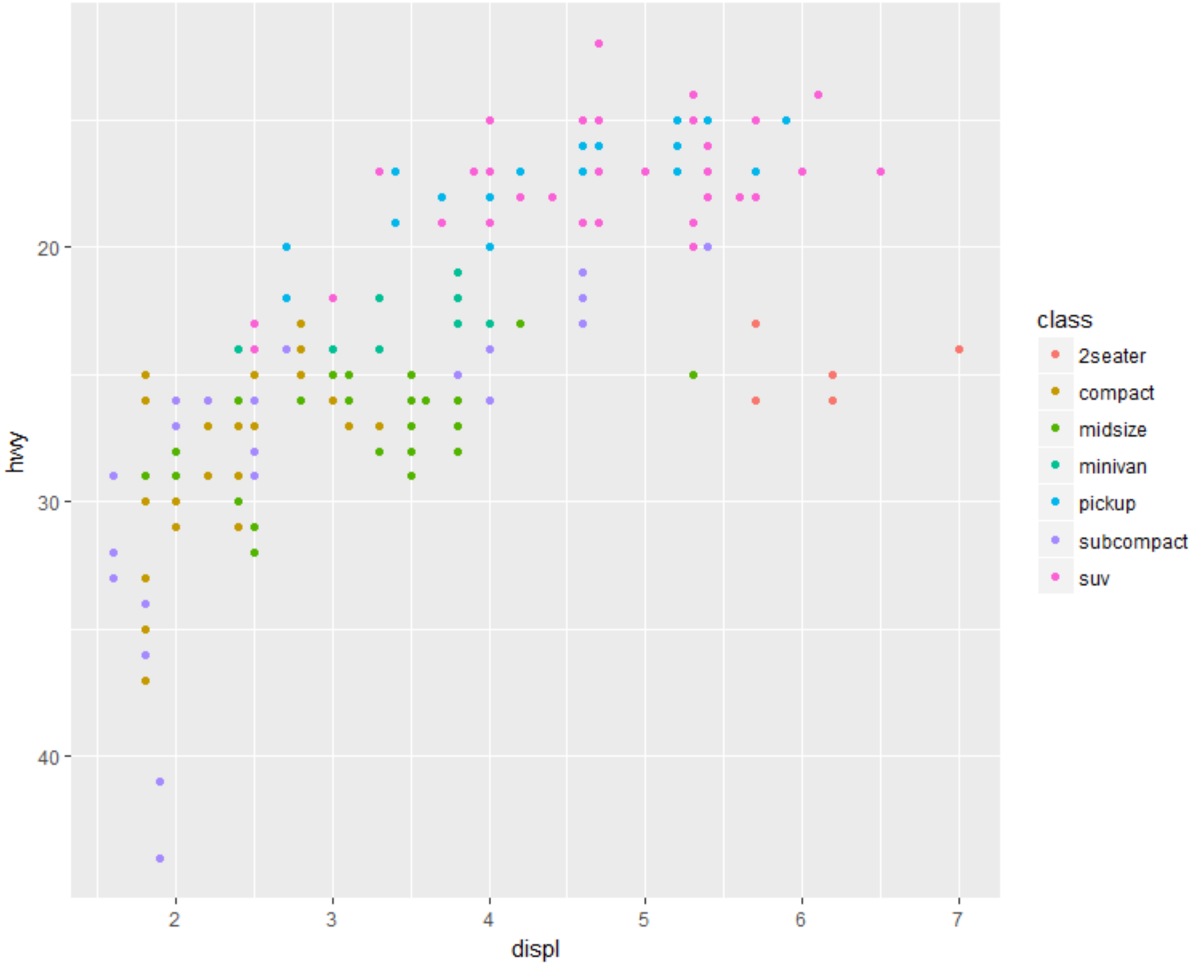
```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class))
```



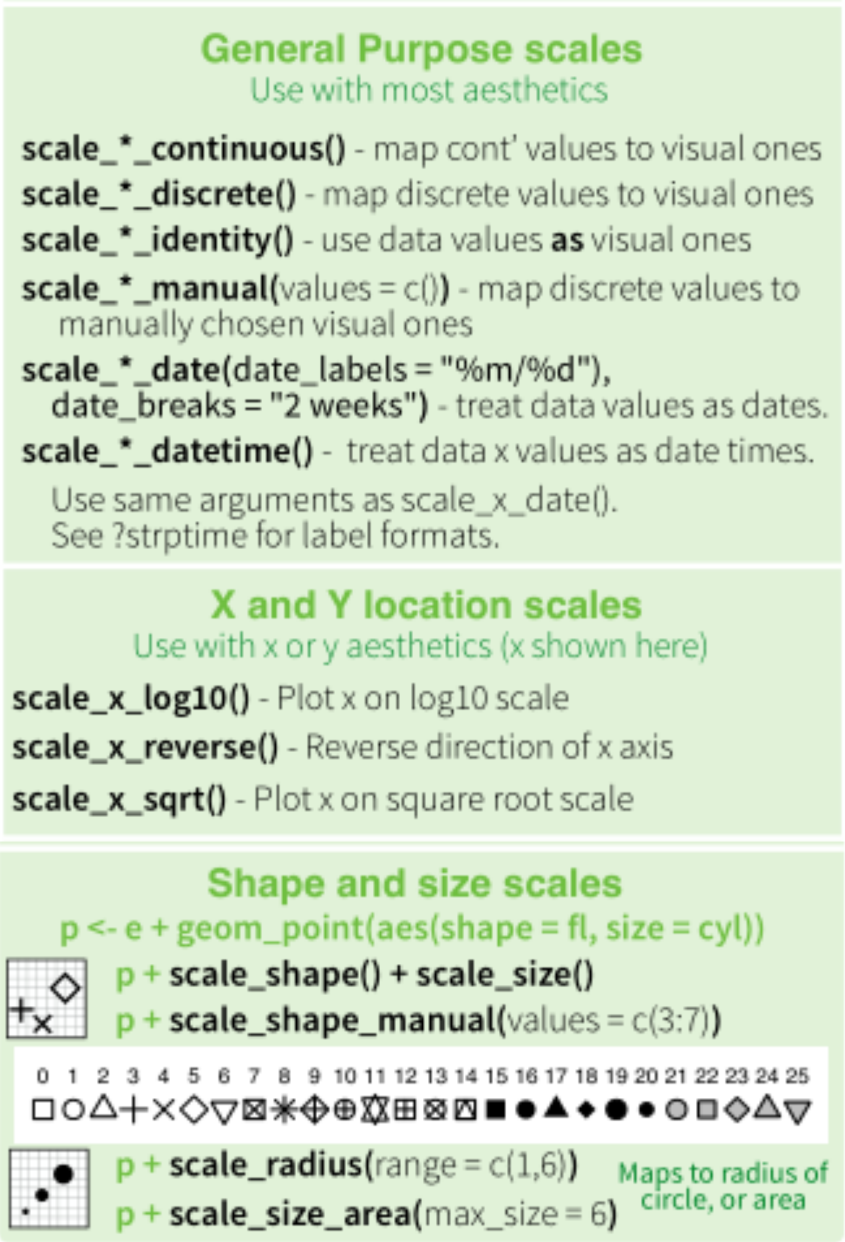
Skalen



```
ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ, y = hwy, color = class)) +
  scale_y_reverse()
```



Skalen & Koordinatensysteme



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 label formats.

X and 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

Shape and size scales

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

p + **scale_shape() + scale_size()**

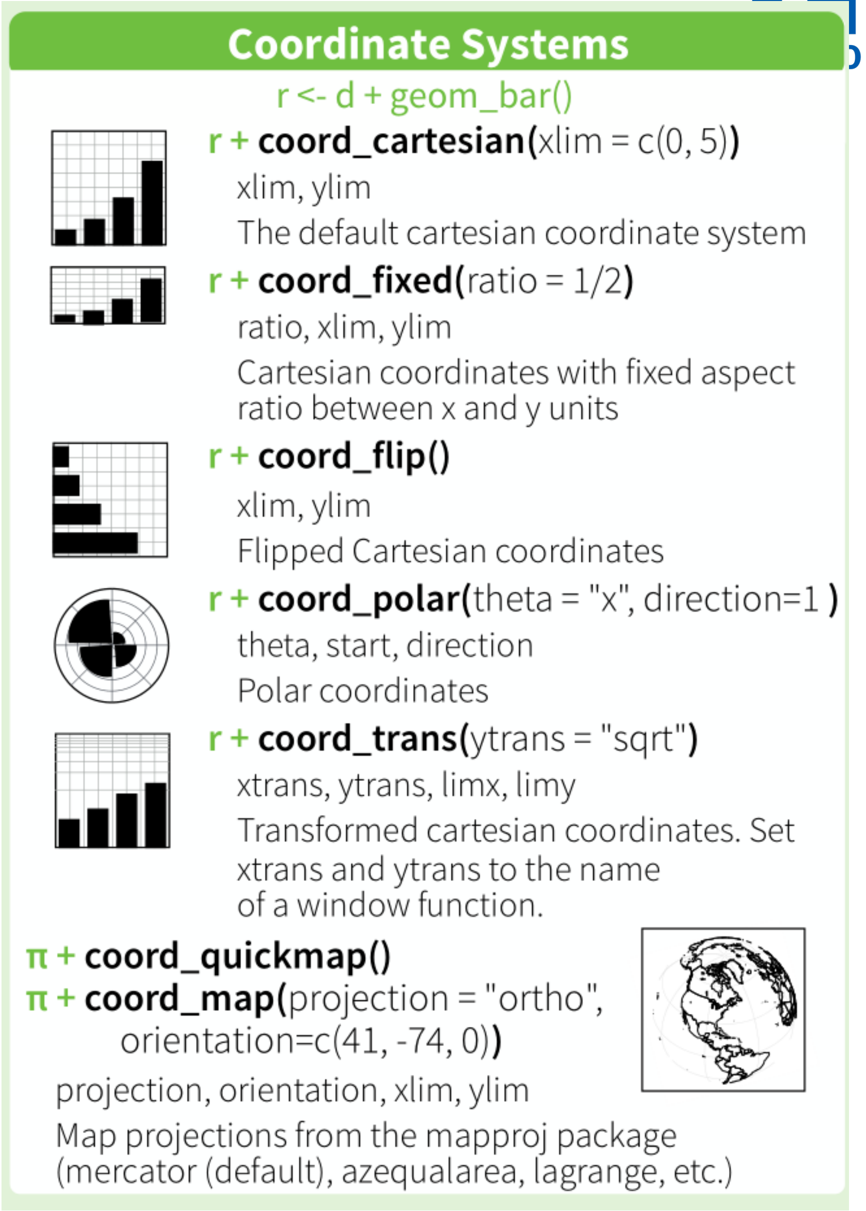
p + **scale_shape_manual(**values = c(3:7)**)**

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
□	○	△	+	×	◇	▽	⊠	✳	◈	⊕	✡	⊞	⊠	◪	■	●	▲	◆	●	•	○	□	◇	△	▽

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

p + **scale_size_area(**max_size = 6**)**

Maps to radius of circle, or area



Coordinate Systems

r <- d + geom_bar()

r + **coord_cartesian(**xlim = c(0, 5)**)**

xlim, ylim

The default cartesian coordinate system

r + **coord_fixed(**ratio = 1/2**)**

ratio, xlim, ylim

Cartesian coordinates with fixed aspect ratio between x and y units

r + **coord_flip()**

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, orientation, xlim, ylim

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

Data visualization with ggplot2 : : CHEAT SHEET , Posit Software, PBC, https://posit.co/wp-content/uploads/2022/10/data-visualization-1.pdf