Introduction to R Graphics

Supplemental Lecture

Dr. Greg Chism

ggplot2

Remember the tidyverse?

- A consistent framework for describing data visualizations.
- Helps think about and plan graphics outside of R… but implemented deeply in R’s in ggplot2 package.
- May also be familiar if you’ve worked in Tableau – Wilkinson now works for Tableau

Grammar of graphics components

data

aesthetic mapping

geometric object

statistical transformations

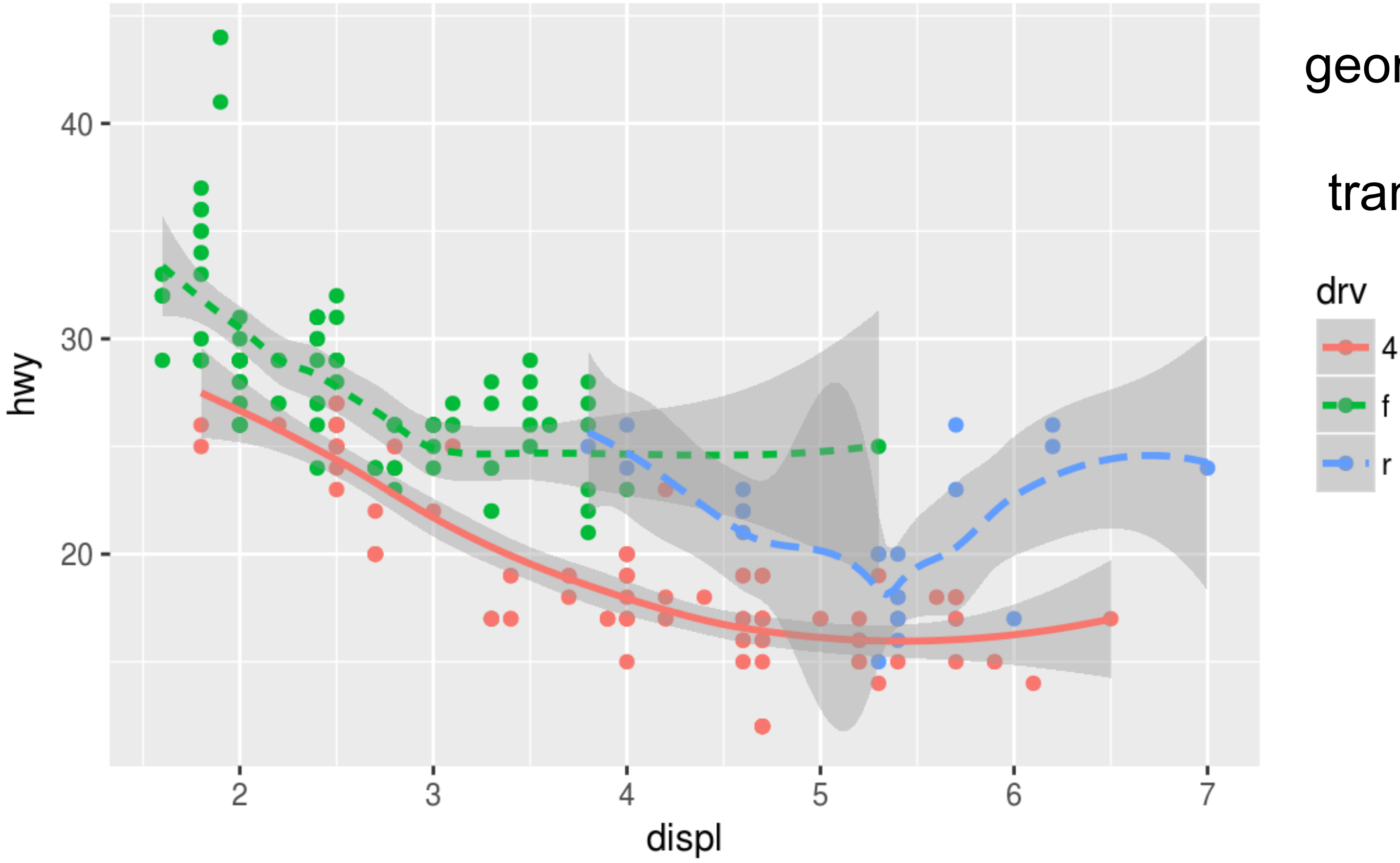
scales

coordinate system

position adjustments

faceting

Anatomy of a ggplot



data

aesthetic mapping

geometric object

statistical transformations

scales

coordinate system

position adjustments

faceting

ggplot components

Or minimally,

```
ggplot(data=<DATA>)+
  <GEOM_FUNCTION>(mapping =
aes(<MAPPINGS>))
```

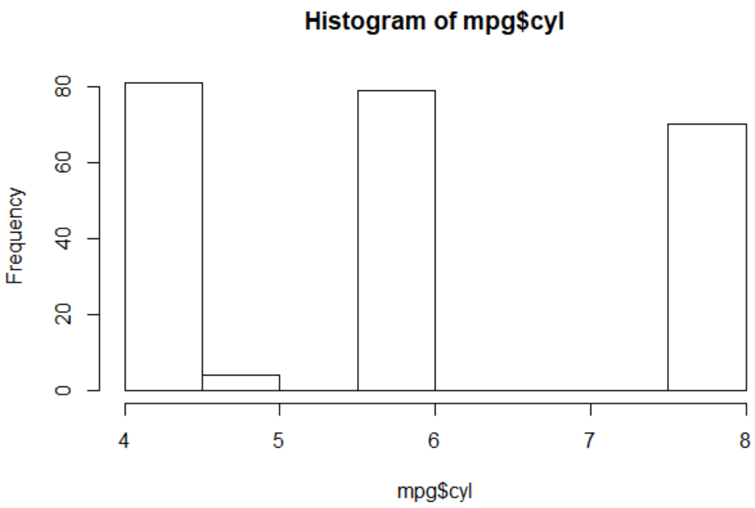
e.g., using mpg dataset

```
ggplot(mpg)+
  geom_point(aes(displ, hwy, color=class))
```

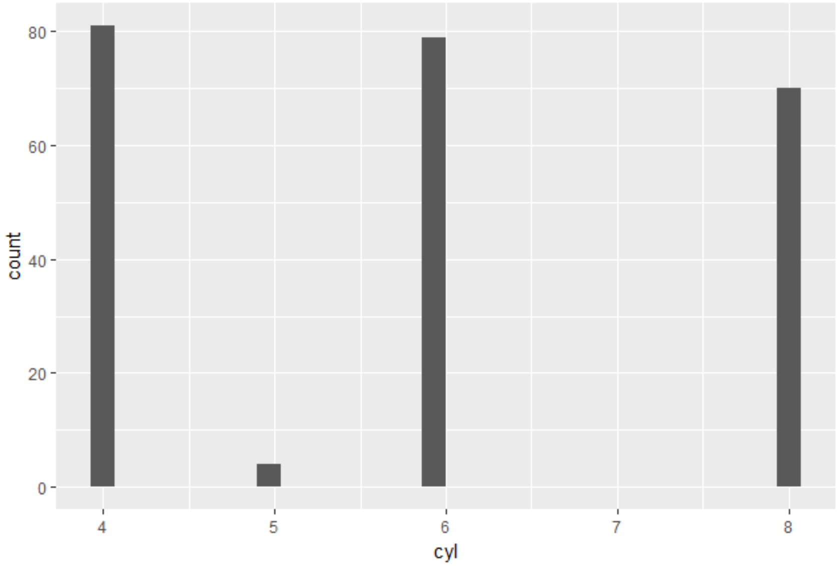
* Here "+" means "and"

Versus base / qplot shorthand

```
hist(mpg$cyl)
```



```
ggplot(mpg)+geom_histogram(aes(x=cyl))
ggplot(mpg, aes(cyl))+geom_histogram()
qplot(mpg$cyl, geom="histogram")
```



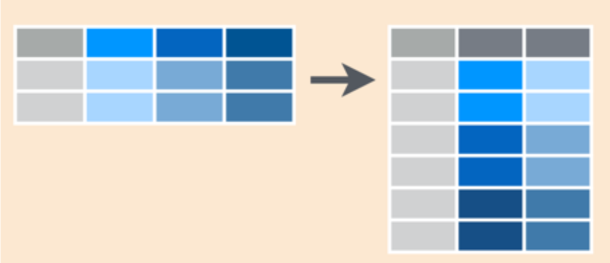
data

ggplot likes "long", well structured data.frames

ggplot "**stats**" can make quick transformations

dplyr will help with complicated transformations

tidyr will help go from wide to long

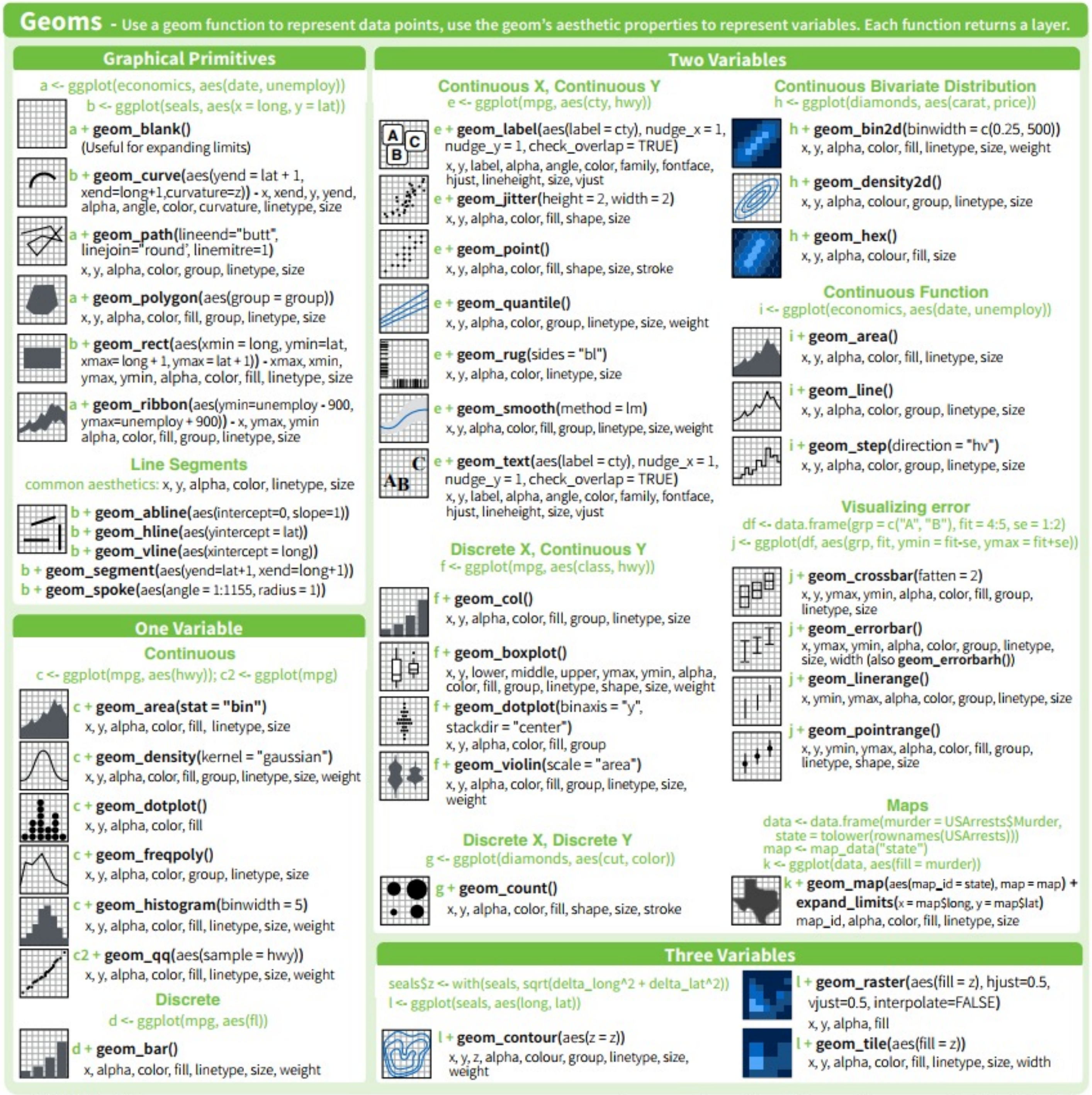


Extensions allow ggplot to understand other kinds of data (e.g. maps, network data)

geoms

…and many more in related packages.

Like other key packages, a cheat sheet is built into R.



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_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

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

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

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

Three Variables

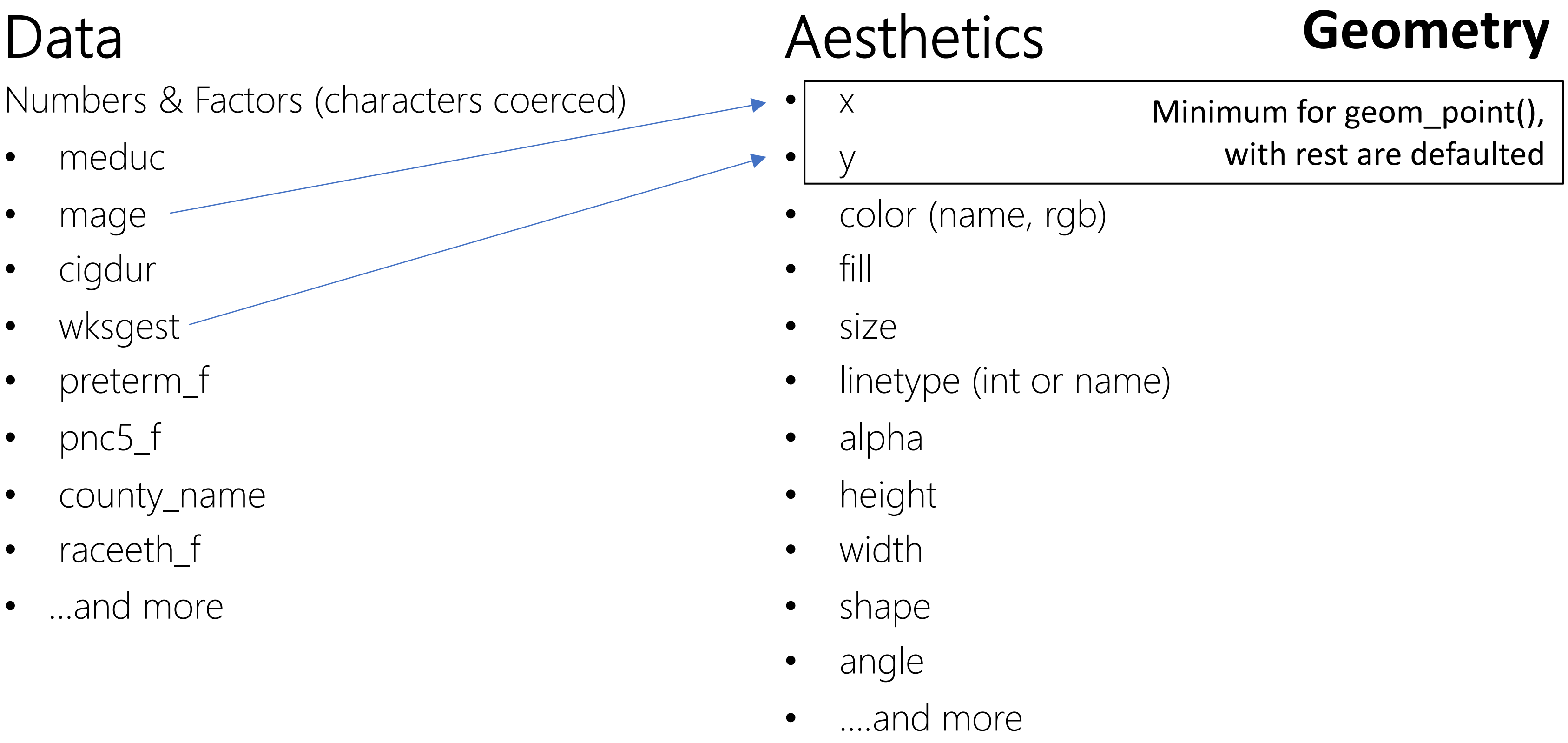
seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
l <- ggplot(seals, aes(long, lat))

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

Aesthetic Mappings



Data

Numbers & Factors (characters coerced)

- meduc
- mage
- cigdur
- wksgest
- preterm_f
- pnc5_f
- county_name
- raceeth_f
- ...and more

Aesthetics

- x
- y
- color (name, rgb)
- fill
- size
- linetype (int or name)
- alpha
- height
- width
- shape
- angle
- ....and more

Geometry

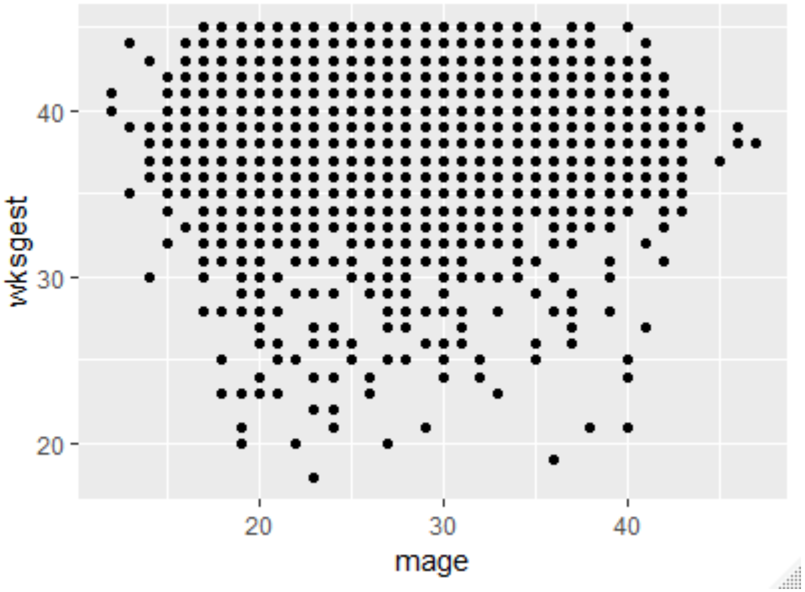
Minimum for geom_point(),
with rest are defaulted

https://cran.r-project.org/web/packages/ggplot2/vignettes/ggplot2-specs.html

Basic geometries and mappings

1. Let’s create a **scatterplot** of wksgest and mage using ggplot and geom_point.

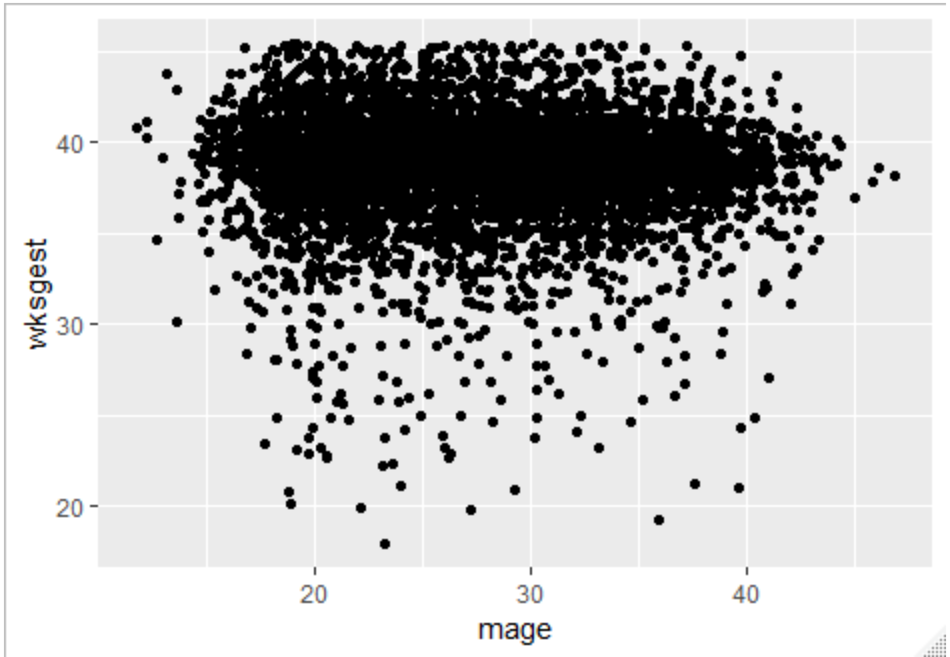
```
ggplot(births_10k, aes(mage, ksgest))+geom_point()
```



Basic geometries and mappings

D'oh! **Overplotting**! Use the geom_jitter() geometry instead.

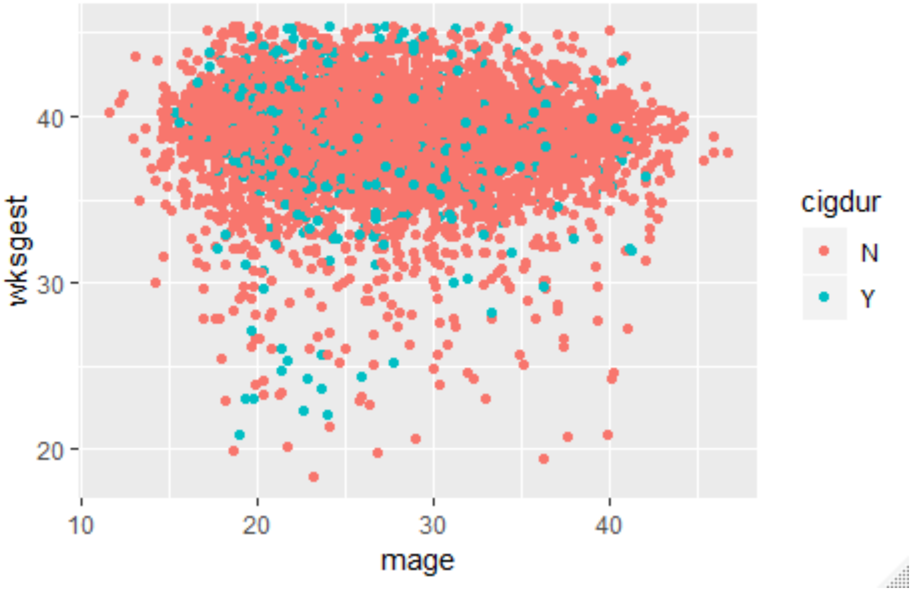
```
ggplot(births_10k, aes(mage, ksgest))+geom_jitter()
```



Basic geometries and mappings

Let's try **colors**. Map cigdur to color. That's it!

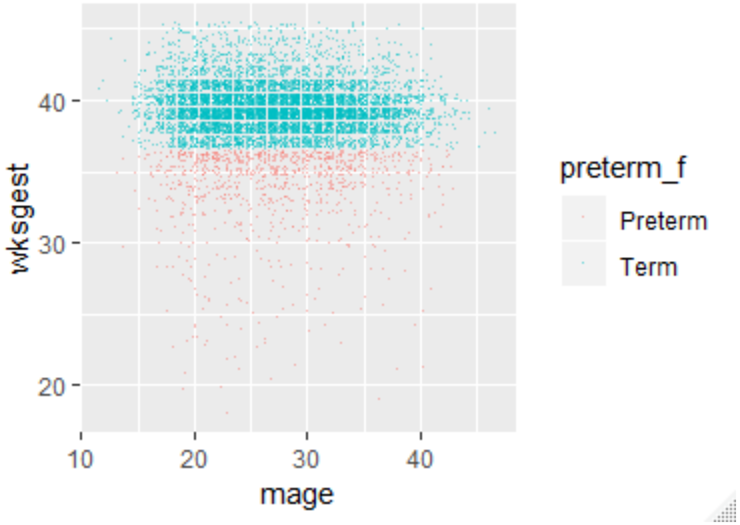
```
ggplot(births_10k, aes(mage, wksgest, color=cigdur))+
  geom_jitter()
```



Basic geometries and mappings

Fancier: change color to preterm_f, change the point character to a “.” and alpha to 0.5. Note global aes!

```
ggplot(births_10k, aes(mage, wksgest, color=preterm_f))+
  geom_jitter(pch=“.”, alpha=0.1)
# ^ Typical chained spacing in last two examples, like dplyr
```

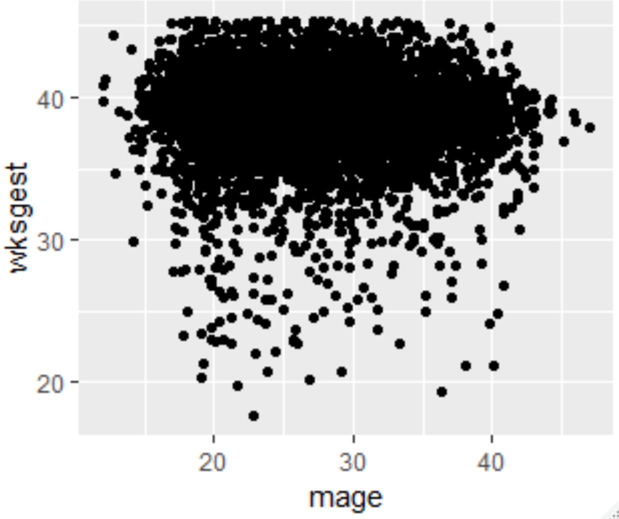


Aesthetic inheritance

Subsequent geometric layers will inherit the aesthetic mappings from the original ggplot() call unless they're overwritten.

Meaning these are equivalent:

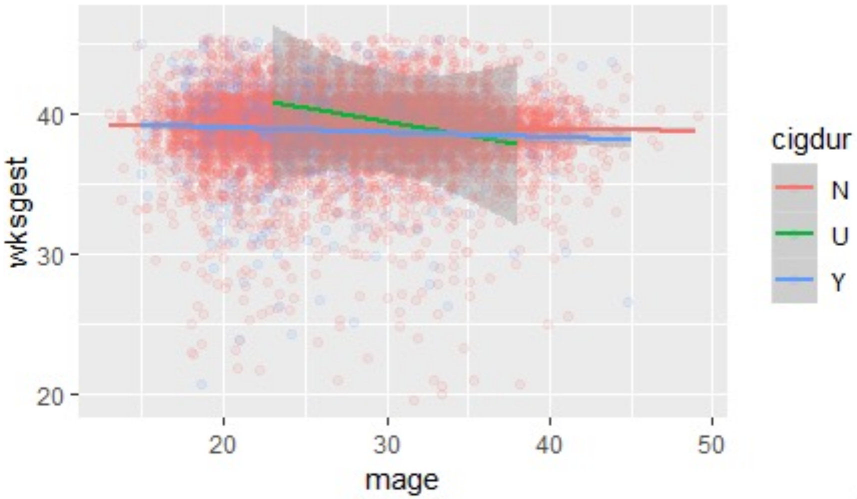
```
ggplot(births_10k, aes(mage, wksgest))+
  geom_jitter()

ggplot(births_10k)+
  geom_jitter(aes(mage, wksgest))
#^ equivalent
```



Aesthetic inheritance 2

…but there’s good reason to be intentional about this – like when we want multiple geometries to use the same mappings

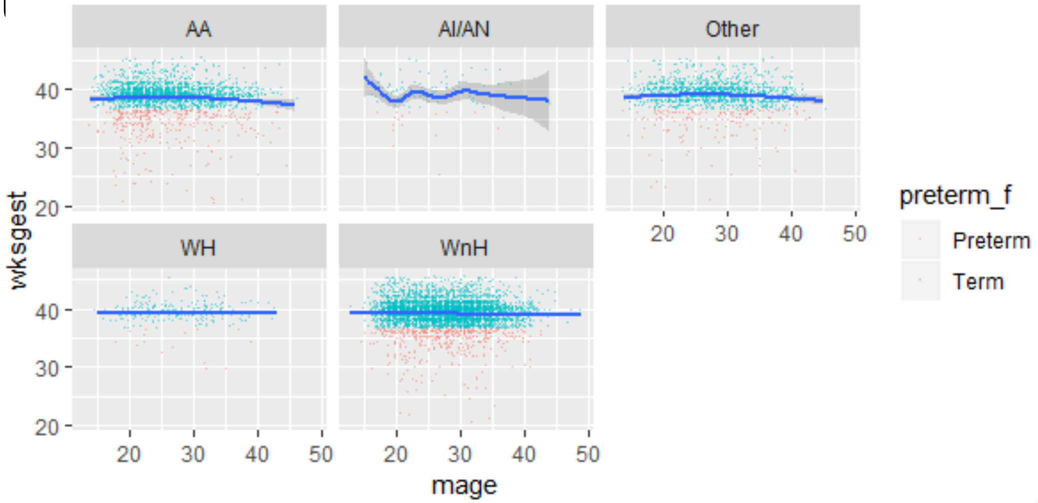
```
ggplot(births_10k, aes(mage, wksgest, color=cigdur))+
  geom_jitter(alpha=0.1) +
  geom_smooth(method="lm")
```



Facets (wrap/grid):small multiples

- Facets take an R formula object (e.g . ~ x, y ~ x) and split your graph into small multiples based on that.
- Can also “free the scales” so they aren’t shared across plots with scales = “free_x” and/or “free_y”

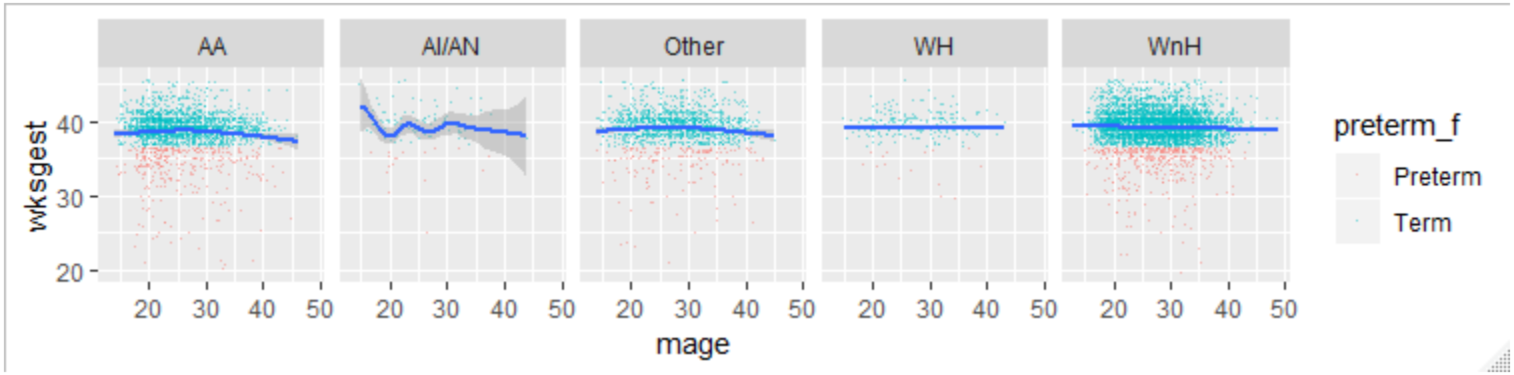
```
ggplot(births_10k, aes(mage, wksgest))+
  geom_jitter(aes(color=preterm_f), pch=".", alpha=0.5)+
  geom_smooth()+ facet_wrap(~ racecth_f)
```



Facets (grid/wrap):small multiples

- Facets take an R formula object (e.g . ~ x, y ~ x) and split your graph into small multiples based on that.
- Can also “free the scales” so they aren’t shared across plots with scales = “free_x” and/or “free_y”

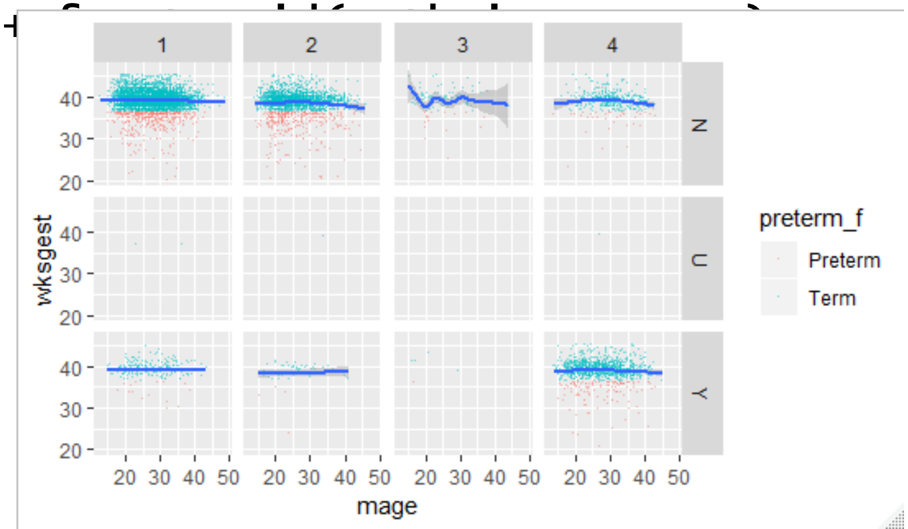
```
ggplot(births_10k, aes(mage, wksgest))+
  geom_jitter(aes(color=preterm_f), pch=".", alpha=0.5)+
  geom_smooth()+ facet_grid( ~ raceeth_f)
```



Facets (grid/wrap):small multiples

- Facets take an R formula object (e.g . ~ x, y ~ x) and split your graph into small multiples based on that.
- Can also “free the scales” so they aren’t shared across plots with scales = “free_x” and/or “free_y”

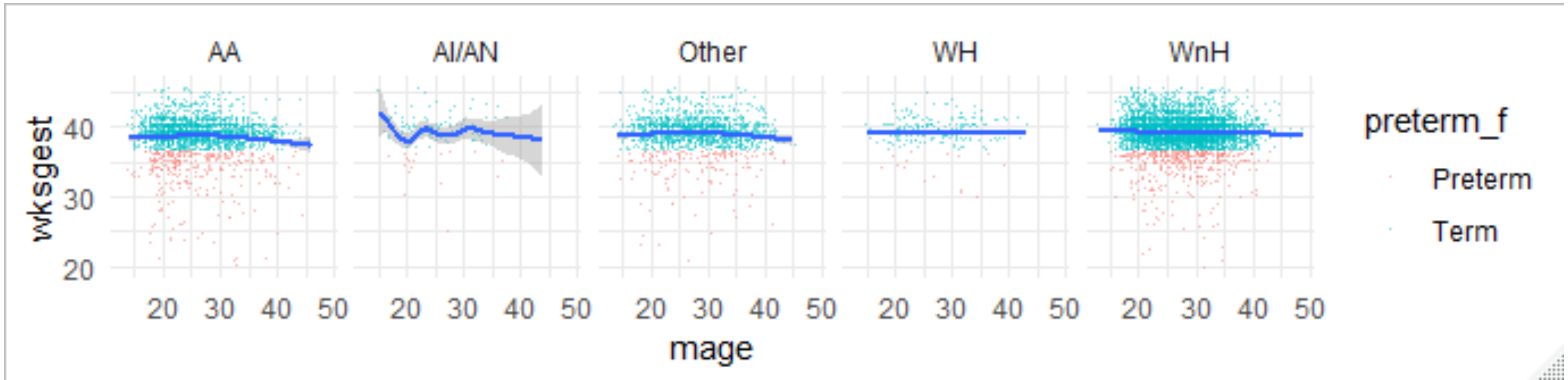
```
ggplot(births_10k, aes(mage, wksgest))+
  geom_jitter(aes(color=preterm_f), pch=".", alpha=0.5)+
  geom_smooth()+
```



Themes

- Change the theme with theme_NAME(), e.g. theme_minimal().
- Can define your own themes, or tweak existing ones.
- See https://cran.r-project.org/web/packages/ggthemes/vignettes/ggthemes.html for more themes. More on ggplot extensions later!

```
ggplot(births_10k, aes(mage, wksgest))+
  geom_jitter(aes(color=preterm_f), pch=".", alpha=0.5)+
  geom_smooth()+ facet_grid( ~ raceeth_f)+
  theme_minimal()
```



Layer=data+stats+geoms

- Hadley's (package author) underlying theory from grammar of graphics: layer=data+stats+geoms. Often stat or geom imply the other, so each has a default parameter of the other.
  - http://vita.had.co.nz/papers/layered-grammar.pdf

- Excellent Stack Overflow Review
  - https://stackoverflow.com/questions/38775661/what-is-the-difference-between-geoms-and-stats-in-ggplot2
- Default stats for geoms: http://sape.inf.usi.ch/quick-reference/ggplot2/geom

Stat variables

- geoms, behind the scenes, often calculate a new dataframe to actually plot on screen.
- stat_<thing> calculates a new dataframe explicitly
- That dataframe has some “secret” (documented) variable names, accessible by special inline variables

..count.. ..ncount.. ..density.. ..ndensity..

..count.. ..prop.. Etc.

- What if I want to refer to those new variables elsewhere in the ggplot call? (rare use – dplyr instead!)