Data Visualization

Kylie Ariel Bemis

9/7/2017

Review

What are the components of a plot, according to the layered grammer of graphics?

Components of a plot

- Default dataset and set of mappings from variables to aesthetics
- One or more layers, each having:
  - (Optional) A dataset
  - (Optional) A set of aesthetic mappings
  - A geometric object
  - A statistical transformation
  - A position adjustment
- A scale for each mapped aesthetic
- A coordinate system
- A facet specification

In ggplot2 (full specification)

```
ggplot(data = <DATASET>,
       mapping = aes(<MAPPINGS>)) +
  layer(geom = <GEOM>,
        stat = <STAT>,
        position = <POSITION>) +
  <SCALE_FUNCTION>() +
  <COORDINATE_FUNCTION>() +
  <FACET_FUNCTION>()
```

In ggplot2 (typical specification)

```
ggplot(data = <DATASET>,
       mapping = aes(<MAPPING>)) +
  <GEOM_FUNCTION>(stat = <STAT>,
                  position = <POSITION>) +
  <COORDINATE_FUNCTION>() +
  <FACET_FUNCTION>()
```

or

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

Geom + stat define many common types of statistical plots

What are these geom + stat combinations?

- geom = “point”, stat = “identity”
- geom = “bar”, stat = “count”
- geom = “bar”, stat = “bin”
- geom = “point”, stat = “qq”
- geom = “boxplot”, stat = “boxplot”

Geom + stat define many common types of statistical plots

What are these geom + stat combinations?

- geom = “point”, stat = “identity”
  - Scatterplot
- geom = “bar”, stat = “count”
  - Bar plot
- geom = “bar”, stat = “bin”
  - Histogram
- geom = “point”, stat = “qq”
  - Quantile-quantile plot
- geom = “boxplot”, stat = “boxplot”
  - Boxplot

In `ggplot2`, each geom has a default stat

What are default stats for these geoms?

- `geom_point`
- `geom_line`
- `geom_bar`
- `geom_boxplot`

In `ggplot2`, each geom has a default stat

What are default stats for these geoms?

- `geom_point`
    - `stat_identity`
- `geom_line`
    - `stat_identity`
- `geom_bar`
    - `stat_count`
- `geom_boxplot`
    - `stat_boxplot`

In `ggplot2`, geoms are shortcuts

Each geom function represents a set of defaults for:

- A geometric object
- A statistical transformation
- A position adjustment

That means some ggplot2 ‘geoms’ represent the same geometric object w/ different statistical transformations.

What are some examples of this?

In ggplot2, geoms are shortcuts

Each geom function represents a set of defaults for:

- A geometric object
- A statistical transformation
- A position adjustment

That means some ggplot2 ‘geoms’ represent the same geometric object w/ different statistical transformations. What are examples of this?

- geom = “bar” : geom_bar, geom_histogram
- geom = “point” : geom_point, geom_qq

In `ggplot2`, geoms are shortcuts

Are there any ggplot2 ‘geoms’ that represent the same geometric object and statistical transformation but with different position adjustments?

In `ggplot2`, geoms are shortcuts

Are there any ggplot2 ‘geoms’ that represent the same geometric object and statistical transformation but with different position adjustments?

- geom = “point” : geom_point, geom_jitter

How can we inspect what these defaults are?

1. Check the help page (`?geom_point`, `?geom_qq`)
2. Check the body and formal arguments of the function

```
geom_histogram
```

```
## function (mapping = NULL, data = NULL, stat = "bin", pos
##     ..., binwidth = NULL, bins = NULL, na.rm = FALSE, sh
##     inherit.aes = TRUE)
## {
##     layer(data = data, mapping = mapping, stat = stat, g
##         position = position, show.legend = show.legend,
##         params = list(binwidth = binwidth, bins = bins,
##             pad = FALSE, ...))
## }
## <environment: namespace:ggplot2>
```

In summary

Minimally, when plotting with `ggplot2`:

- Initialize a with plot `ggplot()` and (optionally):
  - A default dataset
  - A default set of aesthetic mappings
- Add at least one layer with a geometric object and default statistical transformation with `geom_xxx()` and (optionally):
  - A dataset
  - A set of aesthetic mappings

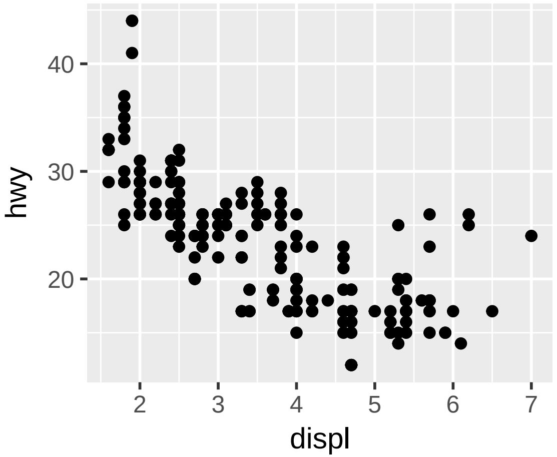
We can even do each step incrementally

With ggplot2, plots can be assigned to a variable to be modified and plotted later.

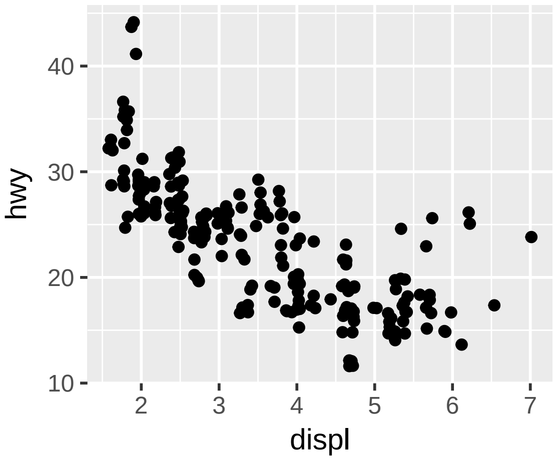
A sequence of ggplot2 functions won't be plotted (yet) if assigned to a variable.

```r
g <- ggplot(data=mpg)
```

```r
g + geom_point(mapping=aes(x=displ, y=hwy))
```



```r
g + geom_jitter(mapping=aes(x=displ, y=hwy))
```



We can also use stat functions with their default geoms

Sometimes it makes more sense to think about the statistical transformation you want to display visually first.

Because each 'stat' also have a default 'geom' in `ggplot2`, you can use the `stat_xxx` functions for plotting as well.

What do you think the default geoms are for these stats?

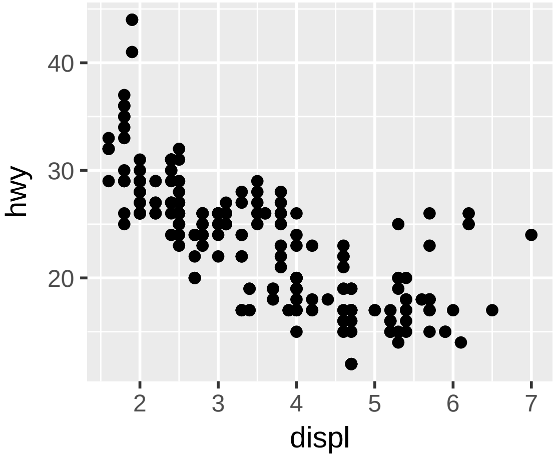
- `stat_identity`
- `stat_count`
- `stat_bin`
- `stat_boxplot`
- `stat_qq`

We can also use stat functions with their default geoms

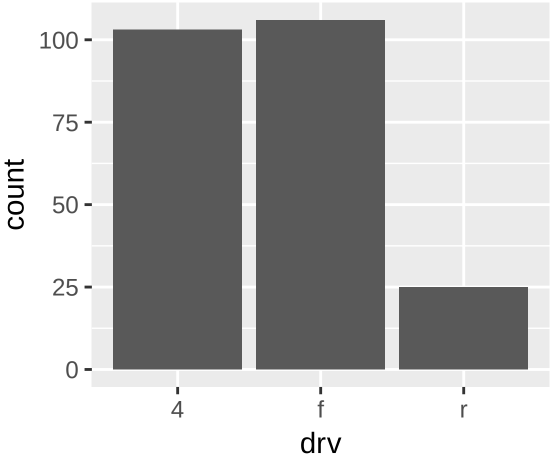
What do you think the default geoms are for these stats?

- `stat_identity`
  - `geom_point`
- `stat_count`
  - `geom_bar`
- `stat_bin`
  - `geom_bar`
- `stat_qq`
  - `geom_point`
- `stat_boxplot`
  - `geom_boxplot`

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

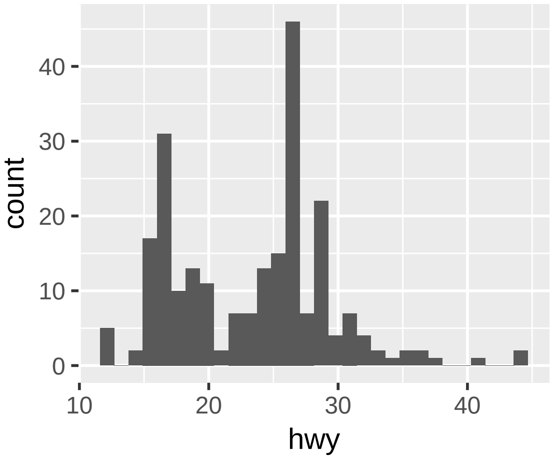


```r
ggplot(data=mpg) + stat_count(mapping=aes(x=drv))
```

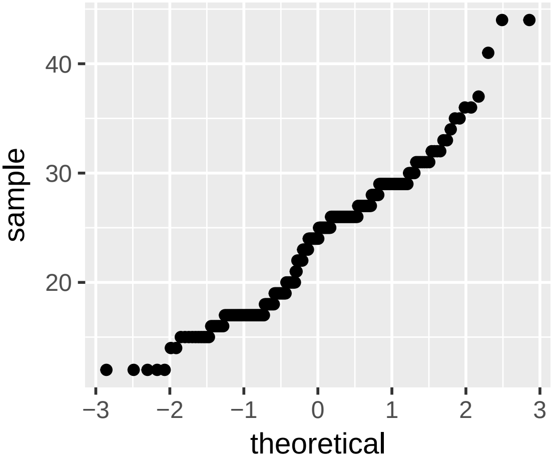


```r
ggplot(data=mpg) + stat_bin(mapping=aes(x=hwy))
```

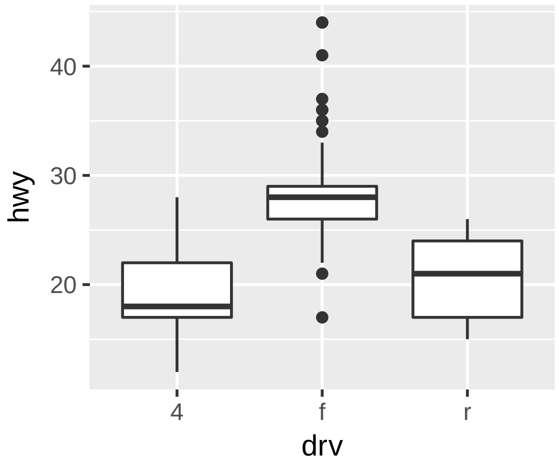
```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



```r
ggplot(data=mpg) + stat_qq(mapping=aes(sample=hwy))
```



```r
ggplot(data=mpg) + stat_boxplot(mapping=aes(x=drv, y=hwy))
```



When to use `stat_xxx` instead of `geom_xxx` functions?

- Using the `stat_xxx` functions to create specify plots is not as common in `ggplot2` as using the `geom_xxx` functions
- However, in some cases, the statistical transformation you want to visualize is more readily obvious than the geometric object
- Use whichever is more intuitive for you

Can we use position_xxx functions to specify a plot?

Do position adjustments have default geometric objects and statistical transformaions associated with them?

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

Can we use position_xxx functions to specify a plot?

Do position adjustments have default geometric objects and statistical transformaions associated with them?

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

```
## Error: Don't know how to add position_identity() to a pl
```

No, position adjustments do not have obvious geometric objects or statistical transformations associated with them.

Most plot types can be defined by a geom + stat.

Computed variables

Many statistical transformations calculate values to be plotted.

A basic example are histograms and bar plots.

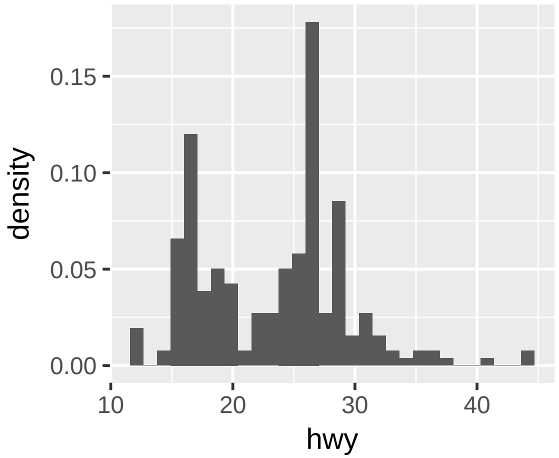
By default, both `geom_histogram` and `geom_bar` plot counts.

What if we want to plot the density of each bin?

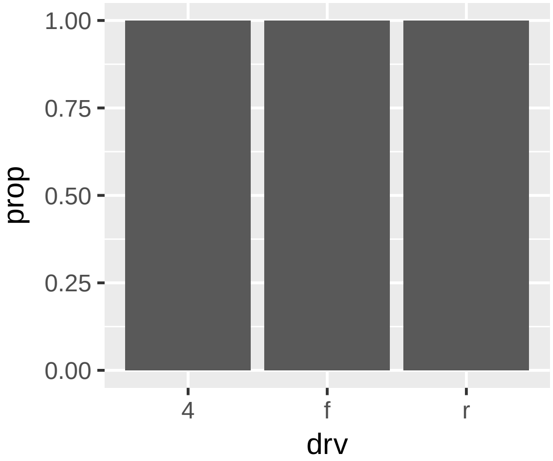
What if we want to plot the proportion for each group?

```r
ggplot(data=mpg) +
  geom_histogram(mapping=aes(x=hwy, y=..density..))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



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


1.00
0.75
0.50
0.25
0.00
prop
4
f
r
drv

What went wrong?

How do we find out exactly what is calculated?

Check the documentation:

```
?geom_bar
?stat_count
```

What went wrong?

“Bar charts”

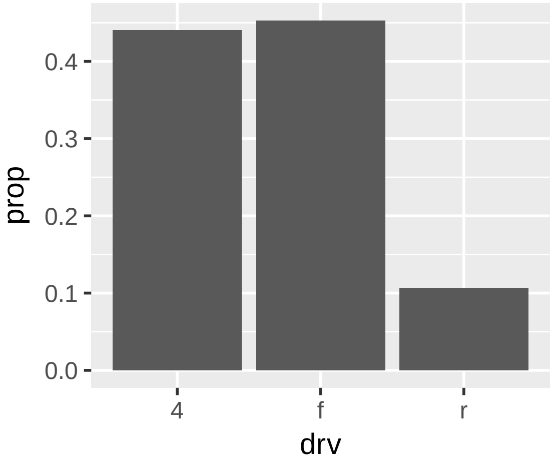
Description

> *There are two types of bar charts: geom_bar makes the height of the bar proportional to the number of cases in each group. . .*

Computed variables

- count
  - number of points in bin
- prop
  - groupwise proportion

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


prop
0.4
0.3
0.2
0.1
0.0
4
f
r
drv

Does it matter what `group` is as long as it's a constant?

```r
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=drv, y=..prop.., group=2))
```

Will this work?

```r
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=drv, y=..prop.., group="foo"))
```

Can we replicate the original behavior while specifying `y` explicitly?

```r
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=drv)) +
  ggtitle("Implicit y")
```

versus

```r
ggplot(data=mpg) +
  geom_bar(mapping=aes(x=drv, y=..count..)) +
  ggtitle("Explicit y")
```

More on calculated variables

- What does `stat_smooth` calculate?
  - What geom does it use?
  - Why doesn't it use `geom_line`?

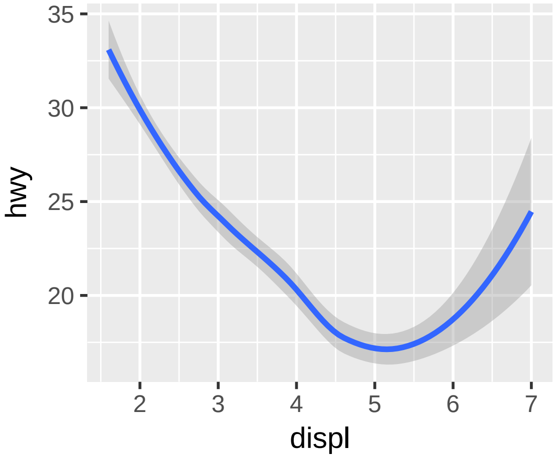
```r
ggplot(data=mpg) + stat_smooth(aes(x=displ,
                                   y=hwy),
                               geom="smooth")
```

versus

```r
ggplot(data=mpg) + stat_smooth(aes(x=displ,
                                   y=hwy),
                               geom="line")
```

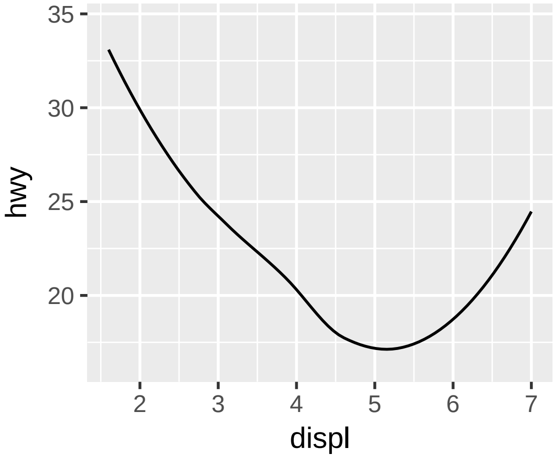
`geom="smooth"`: computed variable (standard error) is plotted

```
## `geom_smooth()` using method = 'loess'
```



`geom="line"`: `geom_line` doesn't know how to plot standard error

```
## `geom_smooth()` using method = 'loess'
```



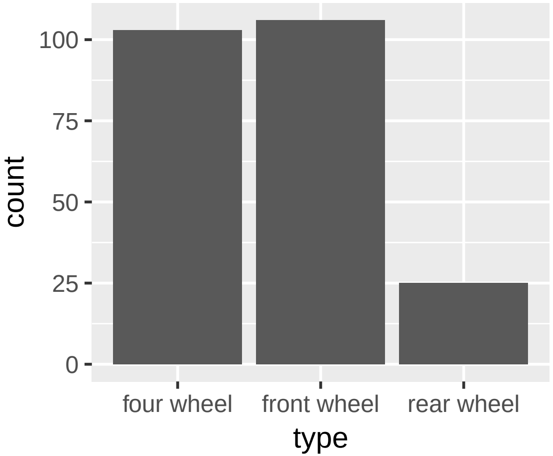
What if we want to calculate our own variables?

```r
drive <- data.frame(type = c("four wheel",
                             "front wheel",
                             "rear wheel"),
                    count = c(sum(mpg$drv == "4"),
                              sum(mpg$drv == "f"),
                              sum(mpg$drv == "r")))
```

What does `sum()` do?

How would the code above change if there were missing values?

```r
ggplot(drive) + geom_bar(mapping=aes(x=type, y=count),
                                   stat="identity")
```



100
75
50
25
0
count
four wheel
front wheel
rear wheel
type

How would we change the code below to plot proportions instead?

```r
drive <- data.frame(type = c("four wheel",
                             "front wheel",
                             "rear wheel"),
                    count = c(sum(mpg$drv == "4"),
                              sum(mpg$drv == "f"),
                              sum(mpg$drv == "r")))
ggplot(drive) + geom_bar(mapping=aes(x=type, y=count),
                              stat="identity")
```

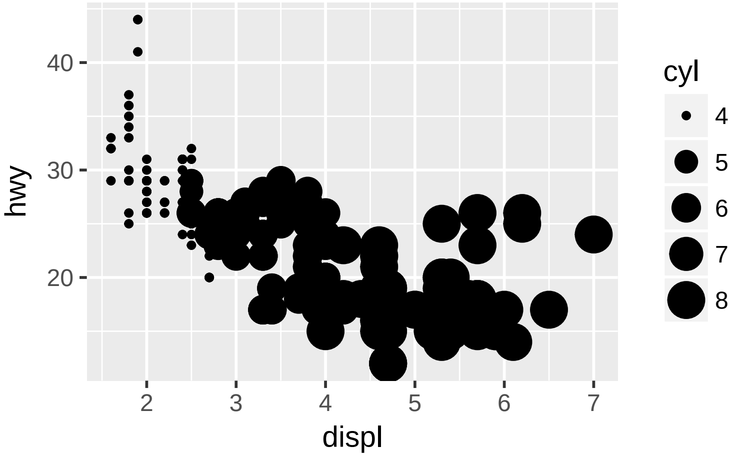
Hint: `nrow()` gives the number of rows in a `data.frame`

Bonus: Can you simplify the code by finding a 'geom' function that uses geom="bar" with stat="identity" by default?

Mapping aesthetics to derived variables

Suppose we want to plot engine size vs highway mileage while visualizing the number of cylinders for each car.

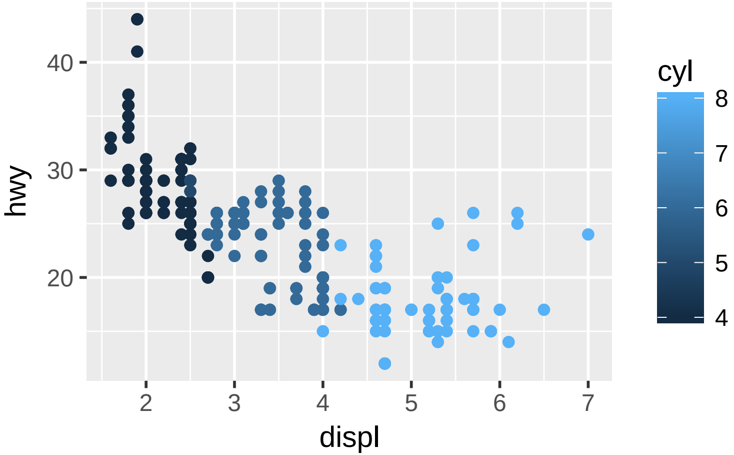
```
ggplot(data=mpg) + geom_point(aes(x=displ,
                                  y=hwy,
                                  size=cyl))
```



This is plot is kind of busy and difficult to read though.

What if we mapped number of cylinders to color?

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

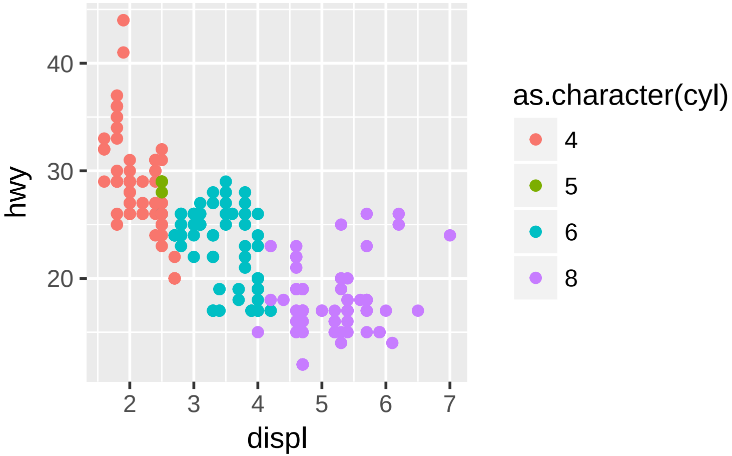


This isn't what we wanted: `ggplot2` is treating `cyl` as continuous.

Make the variable categorical

```
ggplot(data=mpg) + geom_point(aes(x=displ,
                                  y=hwy,
                                  color=as.character(cyl)))
```

What does `as.character()` do?

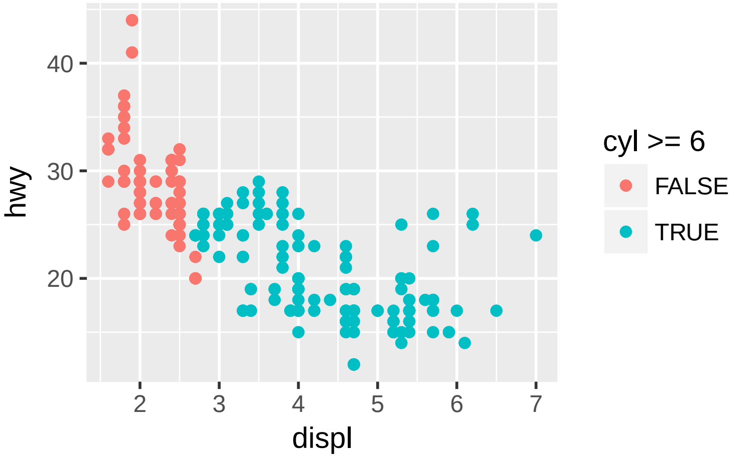


hwy
40
30
20
2
3
4
5
6
7
displ
as.character(cyl)
4
5
6
8

Mapping aesthetics to derived variables (cont'd)

What if we only want to distinguish cars with less than 6 cylinders and cars with 6 cylinders or more?

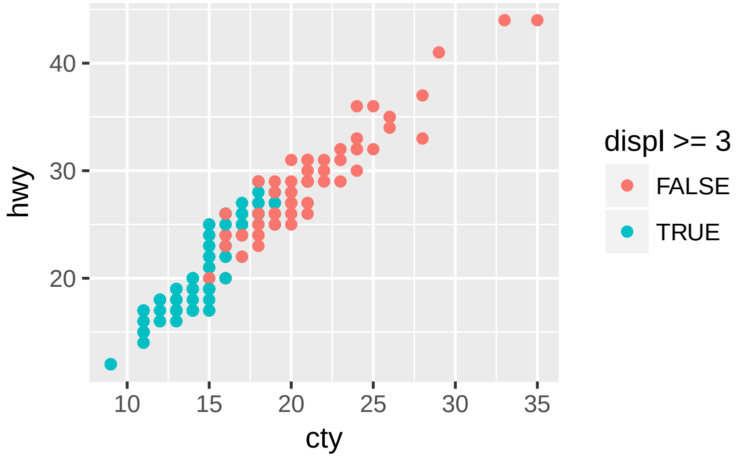
```
ggplot(data=mpg) + geom_point(aes(x=displ,
                                  y=hwy,
                                  color=cyl >= 6))
```


hwy
40
30
20
2
3
4
5
6
7
displ
cyl >= 6
FALSE
TRUE

Mapping aesthetics to derived variables (cont'd)

What if we want to plot highway vs city mileage while distinguishing whether each car has an engine size smaller or larger than 3 L?

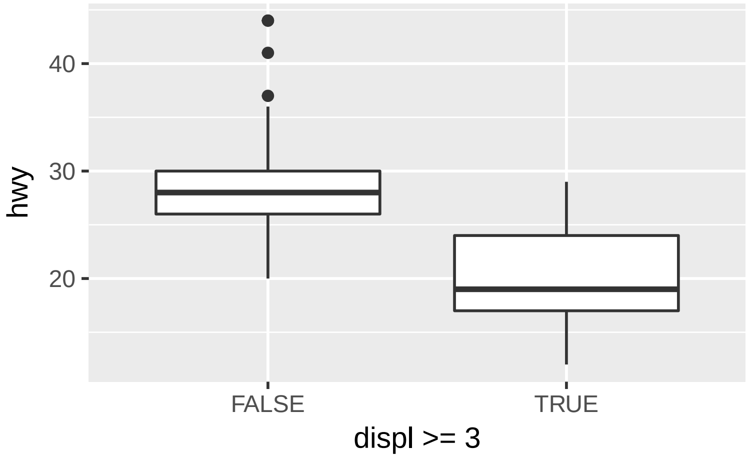
```r
ggplot(data=mpg) + geom_point(aes(x=cty,
                                  y=hwy,
                                  color=displ >= 3))
```


hwy
40
30
20
10
15
20
25
30
35
cty
displ >= 3
FALSE
TRUE

Mapping aesthetics to derived variables (cont'd)

What if we want to make a boxplot of highway mileage for cars with engine sizes smaller and larger than 3 L?

```
ggplot(data=mpg) + geom_boxplot(aes(x=displ >= 3,
                                    y=hwy))
```


40
30
20
hwy
FALSE
TRUE
displ >= 3

Summary: mapping aesthetics to calculated variables

- Aesthetics can be mapped to expressions that calculate new variables
- Aesthetics can be mapped to variables calculated by stats functions

Position adjustments

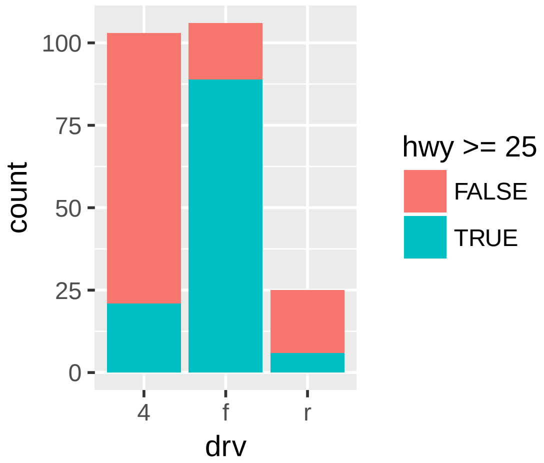
Last Tuesday we discussed a useful position adjustment: “jitter”

What are other useful position adjustments?

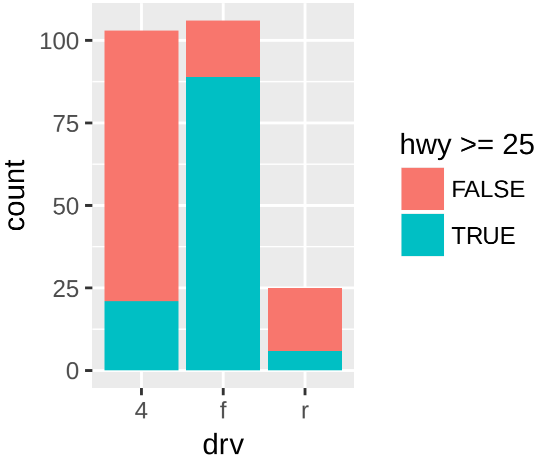
```
ggplot(mpg) + geom_bar(aes(x=drv,
                           fill=hwy >= 25),
                       position="stack")
```

```
ggplot(mpg) + geom_bar(aes(x=drv,
                           fill=hwy >= 25),
                       position="dodge")
```

```
position="stack"
```



```
position="dodge"
```



Scales

Each aesthetic has a scale associated with it.

Usually, a sensible default is used and we don't have to specify it manually:

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

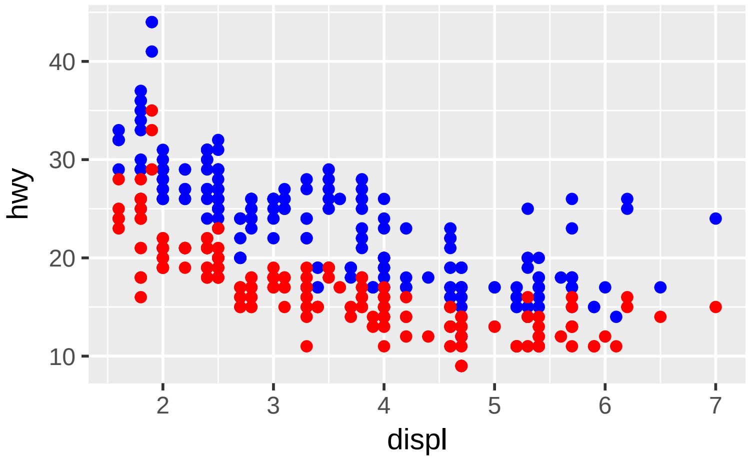
versus

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

Sometimes it is useful to set a scale manually.

Suppose we want to plot both highway and city mileage

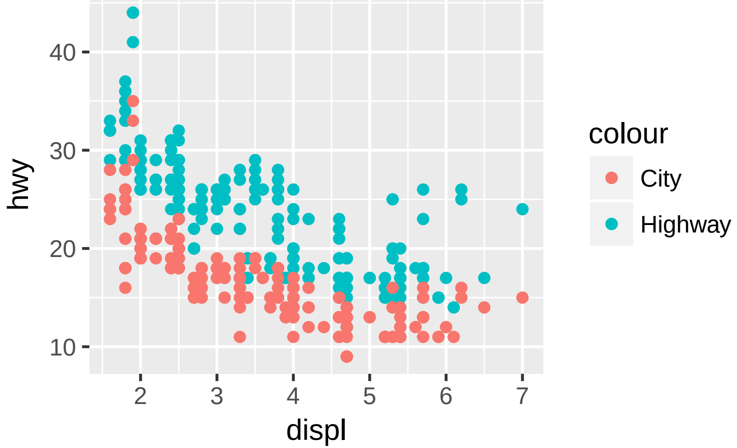
```r
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy), color="blue") +
  geom_point(mapping=aes(x=displ, y=cty), color="red")
```



Why is there no legend?

To generate a legend, an aesthetic must be mapped

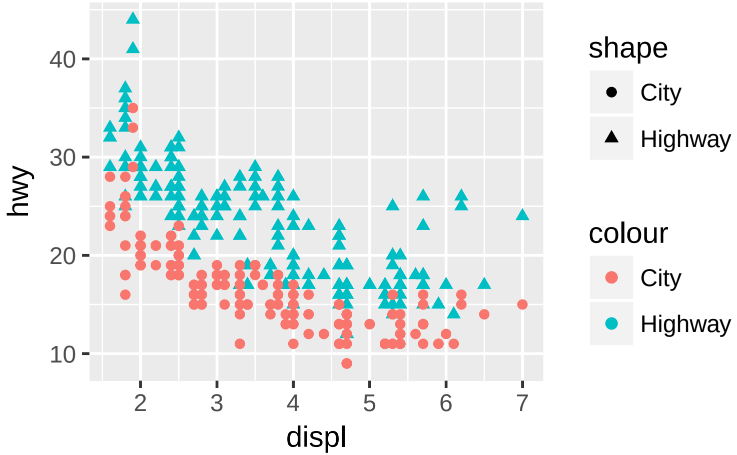
```r
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy, color="Highway"))
  geom_point(mapping=aes(x=displ, y=cty, color="City"))
```



40
30
20
10
hwy
2
3
4
5
6
7
displ
colour
City
Highway

What if we want to use both color and shape?

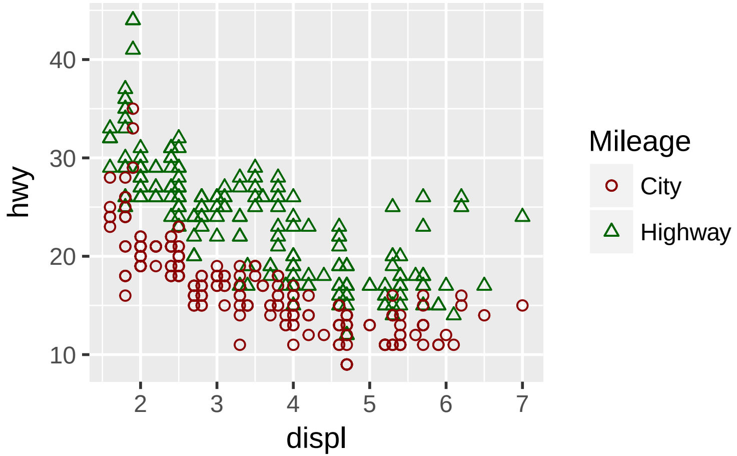
```r
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy,
                         color="Highway",
                         shape="Highway")) +
  geom_point(mapping=aes(x=displ, y=cty,
                         color="City",
                         shape="City"))
```



But this results in two legends.

Need to specify scales manually

```r
ggplot(data=mpg) +
  geom_point(mapping=aes(x=displ, y=hwy,
                         color="Highway",
                         shape="Highway")) +
  geom_point(mapping=aes(x=displ, y=cty,
                         color="City",
                         shape="City")) +
  scale_colour_manual(name="Mileage",
                      values=c("darkred", "darkgreen")) +
  scale_shape_manual(name="Mileage",
                     values=c(1, 2))
```


40
30
20
10
hwy
2
3
4
5
6
7
displ
Mileage
City
Highway

Setting scales manually

- Allows us to control the legend
- Allows us to specify custom levels for each aesthetic

Coordinate systems

Each plot uses a coordinate system.

Usually, a sensible default is used and we don't have to specify it manually:

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

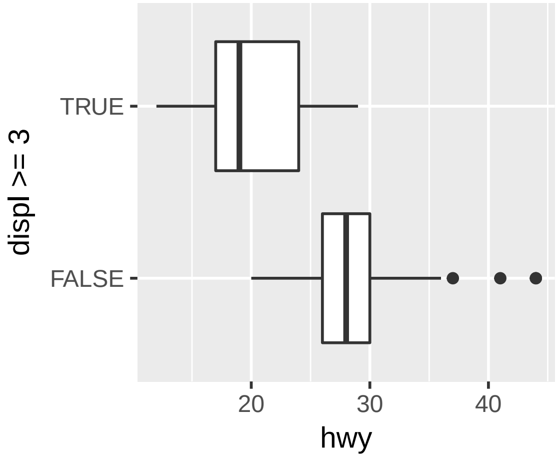
versus

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

Sometimes we want to specify a different coordinate system.

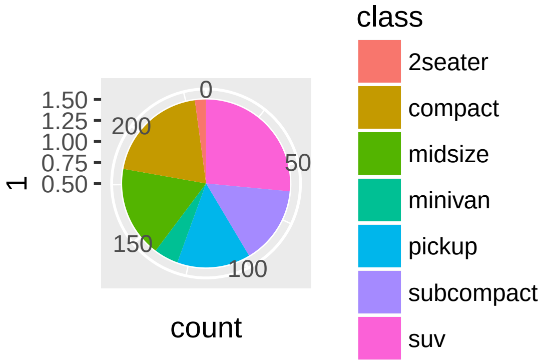
Coordinate systems: flipped

```r
ggplot(data = mpg, aes(x = displ >= 3, y = hwy)) +
  geom_boxplot() +
  coord_flip()
```


TRUE
FALSE
displ >= 3
20
30
40
hwy

Coordinate systems: polar

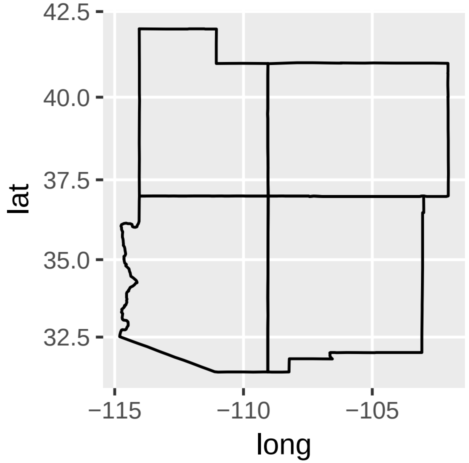
```r
ggplot(data = mpg, aes(x = 1, fill=class)) +
  geom_bar(width=1) +
  coord_polar(theta="y")
```



1.50
1.25
1.00
0.75
0.50
1
0
50
100
150
200
count
class
2seater
compact
midsize
minivan
pickup
subcompact
suv

Coordinate systems: maps

```r
library(maps)
four_corners <- map_data("state",
                         region=c("arizona",
                                  "new mexico",
                                  "utah",
                                  "colorado"))
ggplot(four_corners) +
  geom_polygon(mapping=aes(x=long,
                           y=lat,
                           group=group),
               fill=NA,
               color="black") +
  coord_map()
```



42.5
40.0
37.5
35.0
32.5
lat
−115
−110
−105
long

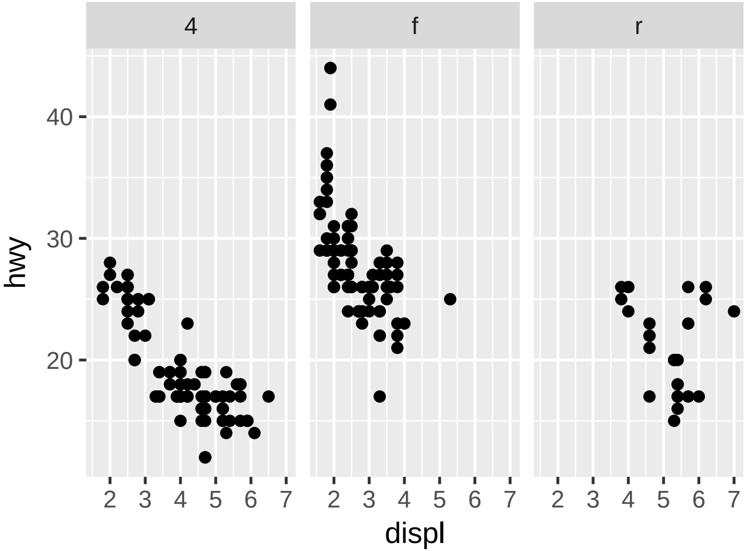
Faceting

Faceting, also called “trellis” or “lattice” graphics, use subplots to visualize different subsets of a dataset, usually by conditioning on levels of a variable in the dataset.

Faceting by 1 categorical variable

Condition on drive type:

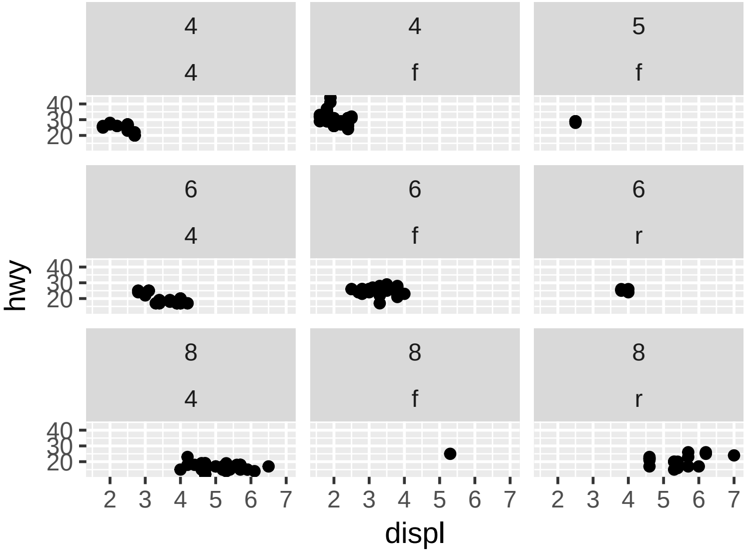
```r
ggplot(mpg, aes(x=displ, y=hwy)) +
  geom_point() +
  facet_wrap(~ drv)
```


4
f
r
hwy
40
30
20
2 3 4 5 6 7
2 3 4 5 6 7
2 3 4 5 6 7
displ

Faceting by 2 categorical variables

Condition on drive type and number of cylinders:

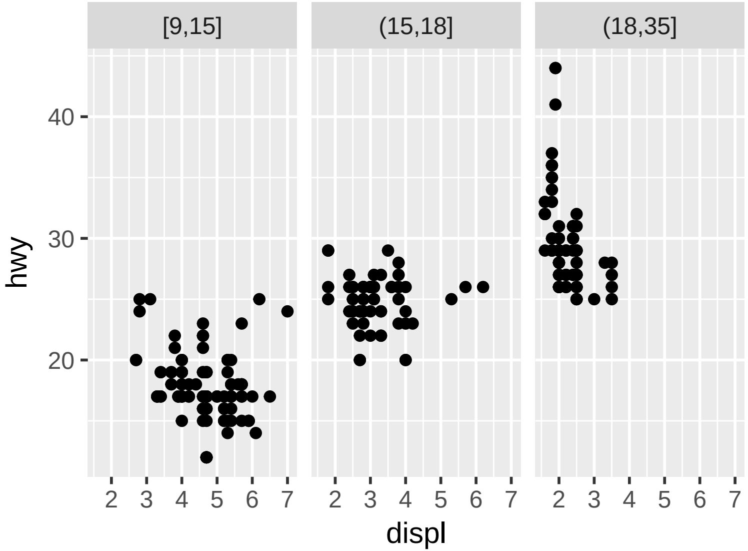
```
ggplot(mpg, aes(x=displ, y=hwy)) +
  geom_point() +
  facet_wrap(cyl ~ drv)
```


4
4
4
f
5
f
6
4
6
f
6
r
8
4
8
f
8
r
hwy
40
30
20
displ
2
3
4
5
6
7

Faceting by a continuous variable

Condition on intervals of city mileage:

```r
ggplot(mpg, aes(displ, hwy)) +
  geom_point() +
  facet_wrap(~ cut_number(cty, 3))
```


[9,15]
(15,18]
(18,35]
hwy
40
30
20
2
3
4
5
6
7
displ