Week 11: Correlations

Avalon C.S. Owens, Eric R. Scott

11/16/2018

Outline

```
library(ggplot2)
```

- Calculating correlation coefficient (Pearson and Spearman)
- Doing correlation test (Pearson and Spearman)
- Bi-variate scatter plots
- Anscombe's quartet
- Using `stat_summary()` in `ggplot2`

Correlation Coefficients

Correlation Coefficients

- `cor()` takes two vectors and returns a correlation coefficient
- Default is Pearson
- Get Spearman correlation coefficient with `method = "spearman"`

```
cor(trees$Girth, trees$Height)
```

```
## [1] 0.5192801
```

```
cor(trees$Girth, trees$Height, method = "spearman")
```

```
## [1] 0.4408387
```

Correlation Coefficients

- Order of two vectors doesn't matter.

```
cor(trees$Height, trees$Girth)
```

```
## [1] 0.5192801
```

Correlation Test

Correlation Test

- `cor.test()` takes two vectors and does a correlation test.
- "`alternative =`" controls tails (`"two.sided"` is default)

```
cor.test(trees$Height, trees$Girth)
```

```
##
##  Pearson's product-moment correlation
##
## data:  trees$Height and trees$Girth
## t = 3.2722, df = 29, p-value = 0.002758
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.2021327 0.7378538
## sample estimates:
##       cor
## 0.5192801
```

Correlation Test

- `method = "spearman"` for Spearman correlation
- Safe to ignore message: "Cannot compute exact p-value with ties"

```
cor.test(trees$Height, trees$Girth, method = "spearman")
```

```
## Warning in cor.test.default(trees$Height, trees$Girth, method =
## "spearman"): Cannot compute exact p-value with ties
```

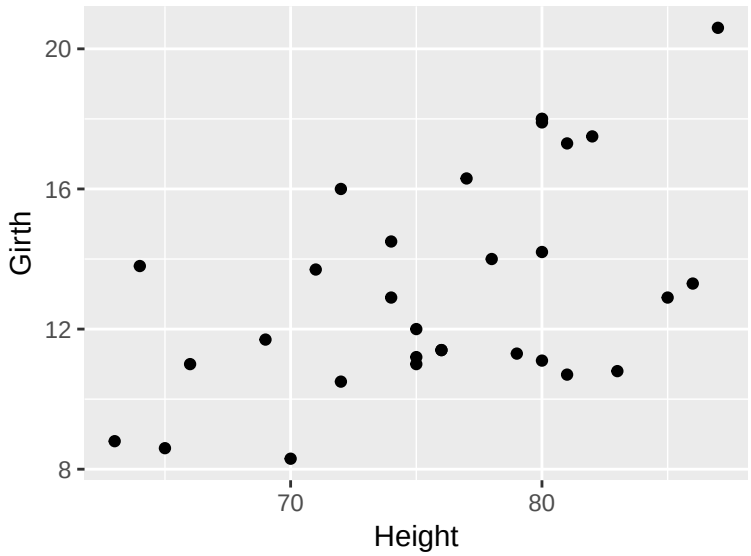
```
##
##  Spearman's rank correlation rho
##
## data:  trees$Height and trees$Girth
## S = 2773.4, p-value = 0.01306
## alternative hypothesis: true rho is not equal to 0
## sample estimates:
##       rho
## 0.4408387
```

Bi-variate Scatter Plots

Scatterplots in ggplot2

- Both `x` and `y` `aes`thetics need to be continuous
- `geom_point()` plots points

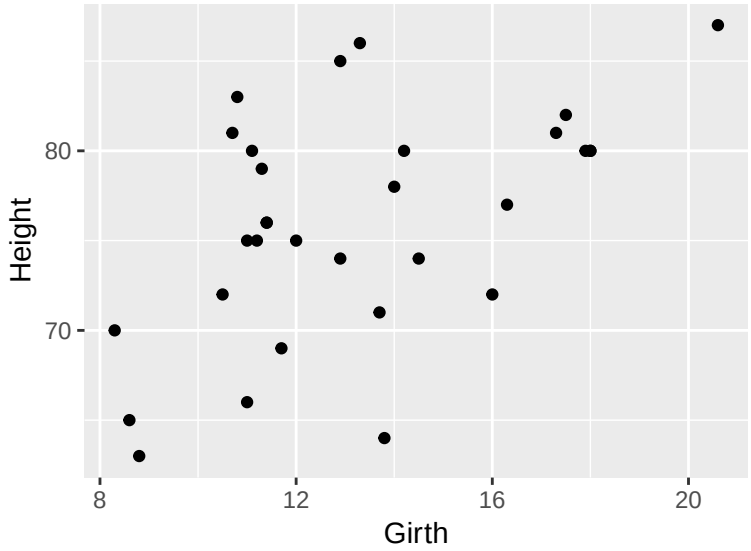
```
ggplot(trees, aes(x = Height, y = Girth)) + geom_point()
```



Scatterplots in ggplot2

- OK to use either variable for the x-axis for correlations.
- You can put `aes()` either inside of `ggplot()` or inside of a `geom`

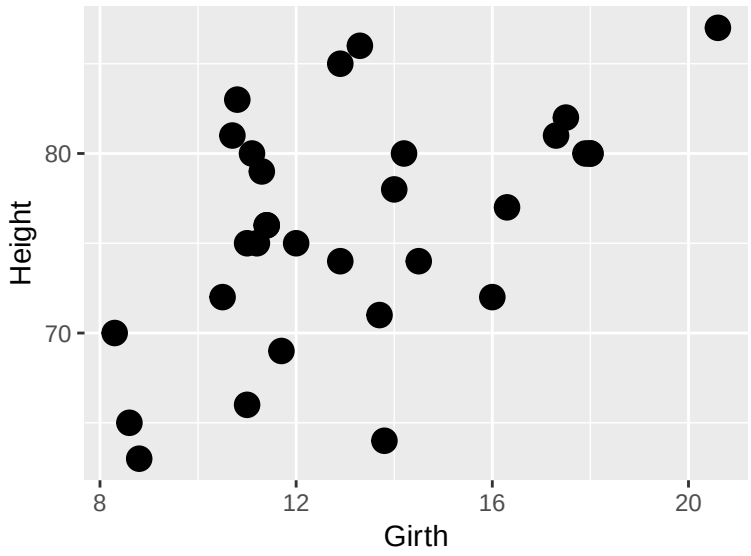
```
ggplot(trees) + geom_point(aes(x = Girth, y = Height))
```



Scatterplots in ggplot2

- Use `size =` outside of `aes()` to change the size of all points

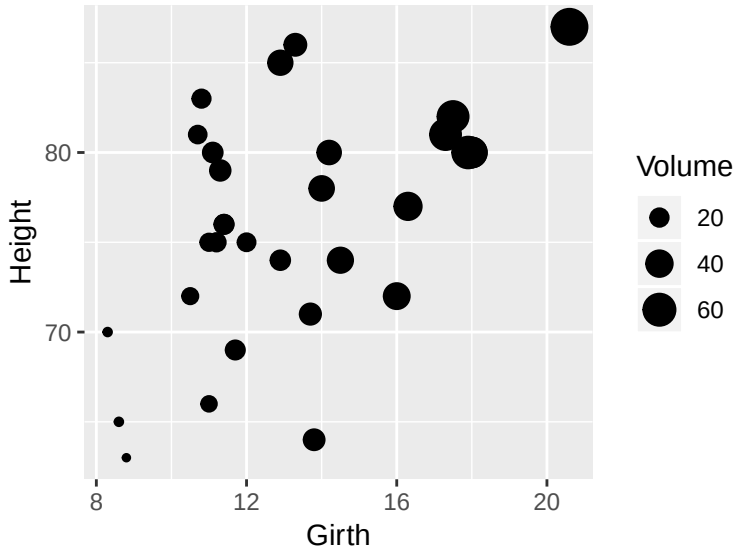
```
ggplot(trees) + geom_point(aes(x = Girth, y = Height), size = 4)
```



Scatterplots in ggplot2

- Put `size =` *inside* of `aes()` to map a variable to point size.
- You probably won't have a use for this in this class.

```
ggplot(trees) + geom_point(aes(x = Girth, y = Height, size = Volume))
```



Customizing plots

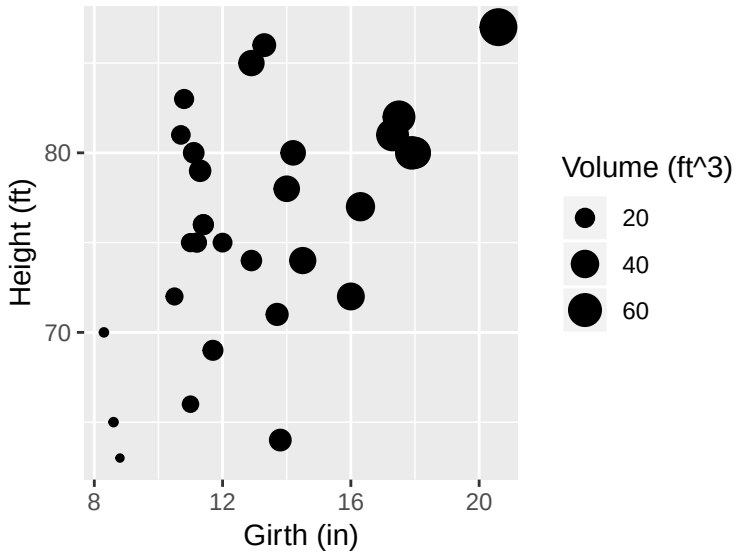
Save previous plot

```
p <- ggplot(trees) + geom_point(aes(x = Girth, y = Height, size = Volume))
```

Finishing up your scatter plot

- Add units to axis labels
- (and to size legend)

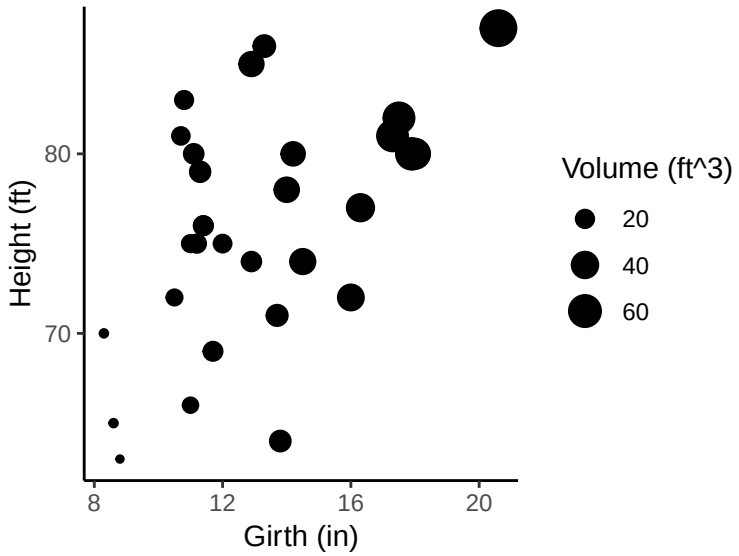
```
p2 <- p +
  labs(x = "Girth (in)", y = "Height (ft)") +
  scale_size_continuous("Volume (ft^3)")
p2
```



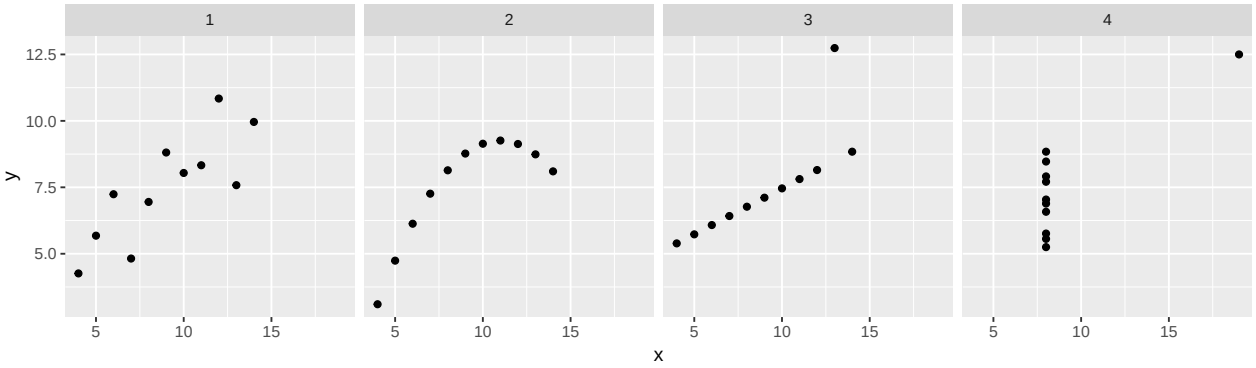
Finishing up your scatter plot

- Try out a different `theme_*()`
- Type theme_ and browse the dropdown menu to try some out.

```
p2 + theme_classic()
```



Anscombe's quartet



Anscombe's Quartet

- Anscombe's Quartet is a group of 4 bi-variate datasets
- Similar mathematical properties despite very different shapes
- Data visualization is important!
- Available as `anscombe` in R
- Even more mathematically similar and visually different datasets in the `datasauRus` package!

Anscombe's Quartet

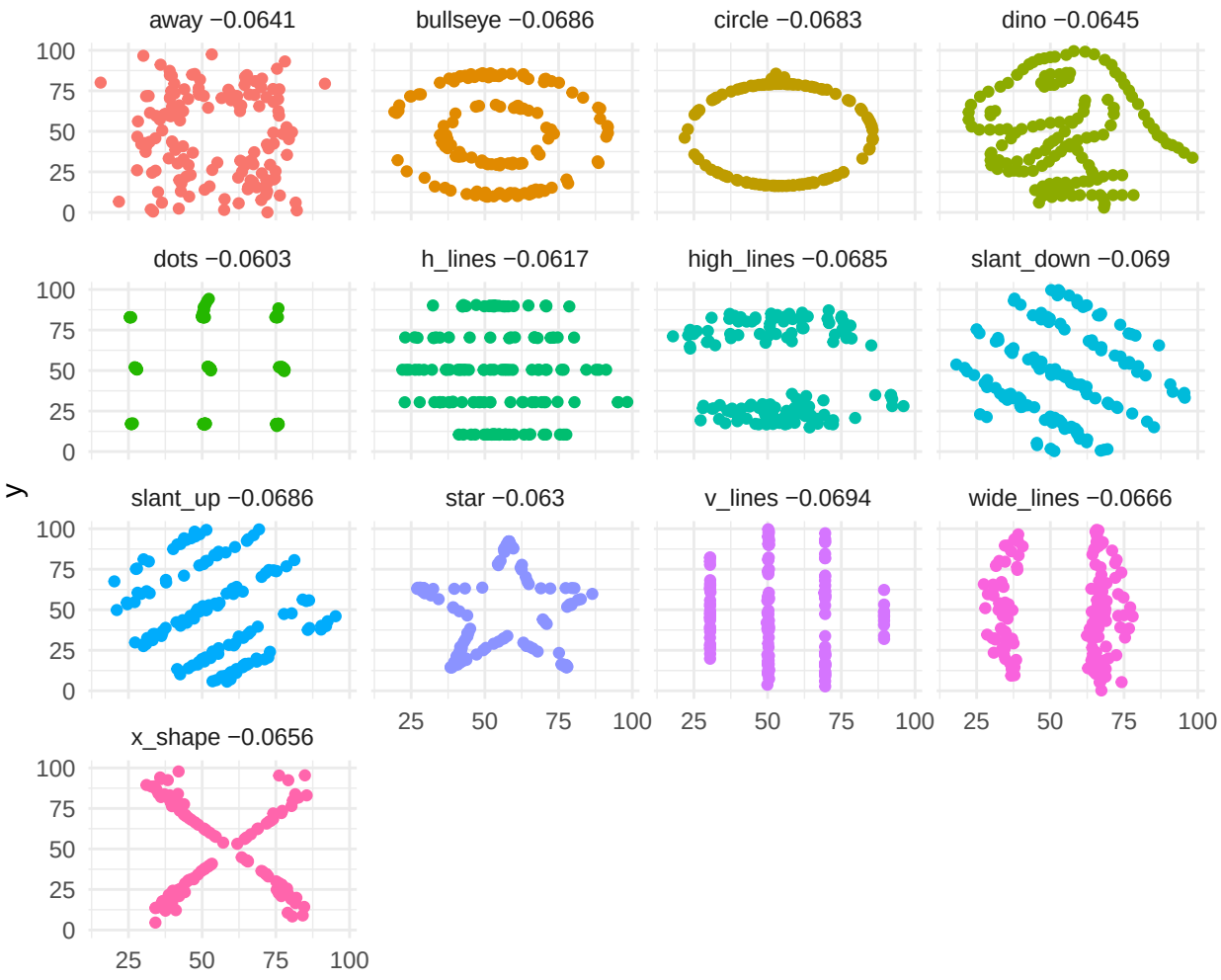
For your assigned set of x and y variables:

- Calculate the Pearson Spearman correlation coefficients
- Do a correlation test (just get the p-value, not all 6 steps)
- Make a bi-variate scatter plot
- Compare with another student

```
anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

datasauRus

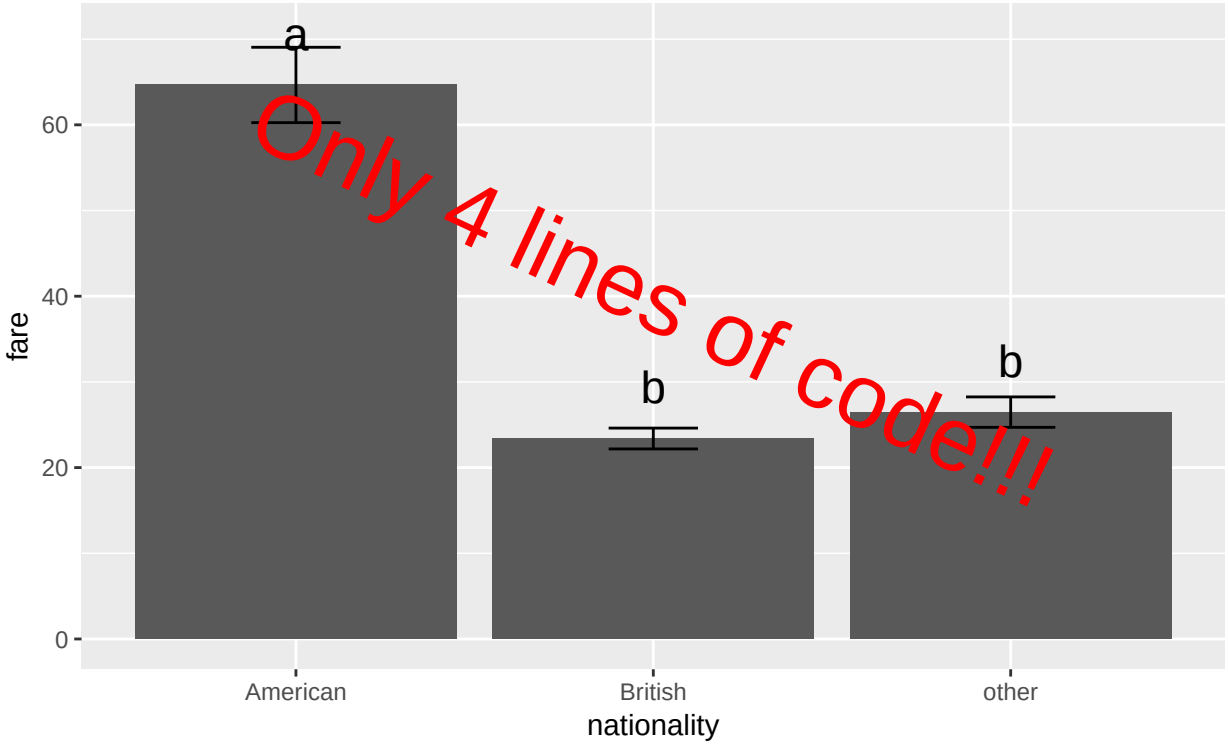


What now?

Work on homework?

OR

Go through a *way* quicker way to make bar plots and add significance letters?



Making complex plots simple with `stat_summary()`

What's a `stat`?

- "geoms" add mapping to geometric features (e.g points, lines, bars)
- "stats" do some transformation.
- E.g. `geom_boxplot()` uses `stat = "boxplot"` by default. Takes dataframe and calculates quartiles, fences, outliers.
- `geom_point()` uses `stat = "identity"` by default. Simply passes x and y unchanged.

Adding a stat

- Some geoms can take different stats, but usually not a good idea to change the default.
- You *can* add a `stat_*()` instead of adding a `geom_*()`
- E.g. instead of:

```
p + geom_boxplot(stat = "boxplot")
```

- You could do:

```
p + stat_boxplot(geom = "boxplot")
```

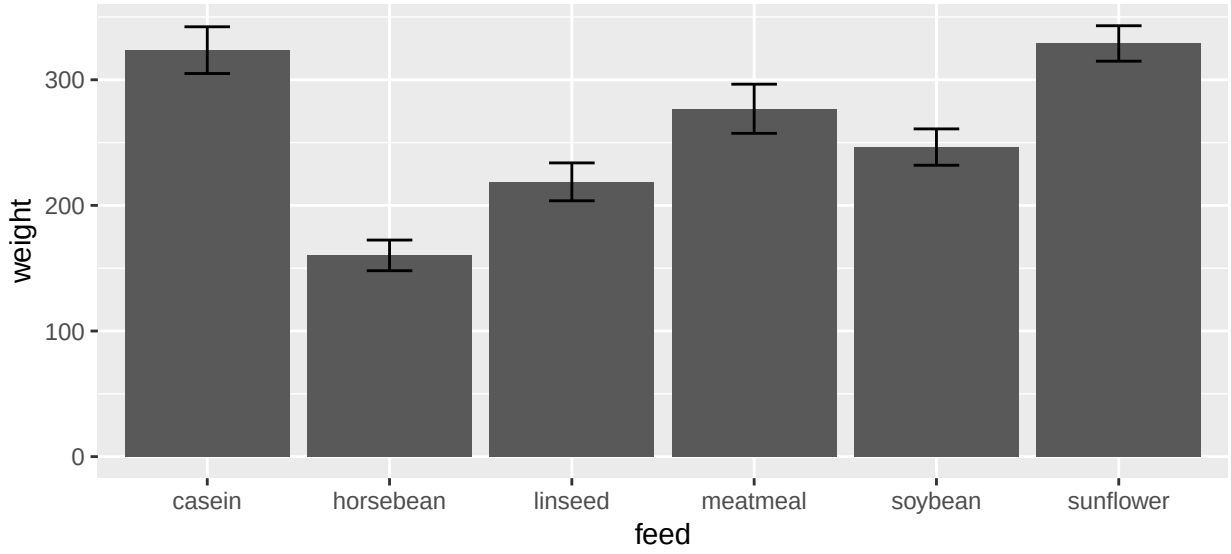
    - These are identical, no reason to *actually* do this.

stat_summary() is flexible!

- Tell it how you want it to summarize your data frame and what geom you want it to use

- Remember bar plots? First we summarized the data using `group_by(...) %>% summarize(...)`, then we plotted the summary data frame.
- With `stat_summary()` instead:

```
ggplot(chickwts, aes(x = feed, y =  weight)) +
  stat_summary(geom = "bar", fun.y = "mean") + # makes the bars
  stat_summary(geom = "errorbar", fun.data = "mean_se", width = 0.25) # makes the errorbars
```



stat_summary()

```
ggplot(chickwts, aes(x = feed, y =  weight)) +
  stat_summary(geom = "bar", fun.y = "mean") # use x = feed, but y = mean(weight)
```

- `geom = "bar"` tells it to use `geom_bar()`
- `fun.y` is telling it how to get the y-values that `geom_bar()` needs. Here, we used `mean()`
- Try changing `fun.y` to something else like `min`, `max`, or `median`.

stat_summary()

```
ggplot(chickwts, aes(x = feed, y =  weight)) +
  stat_summary(geom = "bar", fun.y = "mean") +
  stat_summary(geom = "errorbar", fun.data = "mean_se")
# use x = feed, but ymin = mean(weight) - SEM, ymax = mean(weight) + SEM
```

- `geom = "errorbar"` tells it to use `geom_errorbar()`
- `mean_se` is a function from `ggplot2` that calculates ymin, and ymax aesthetics needed for errorbars.
- `fun.data` because `mean_se` returns a dataframe, not just the y-values

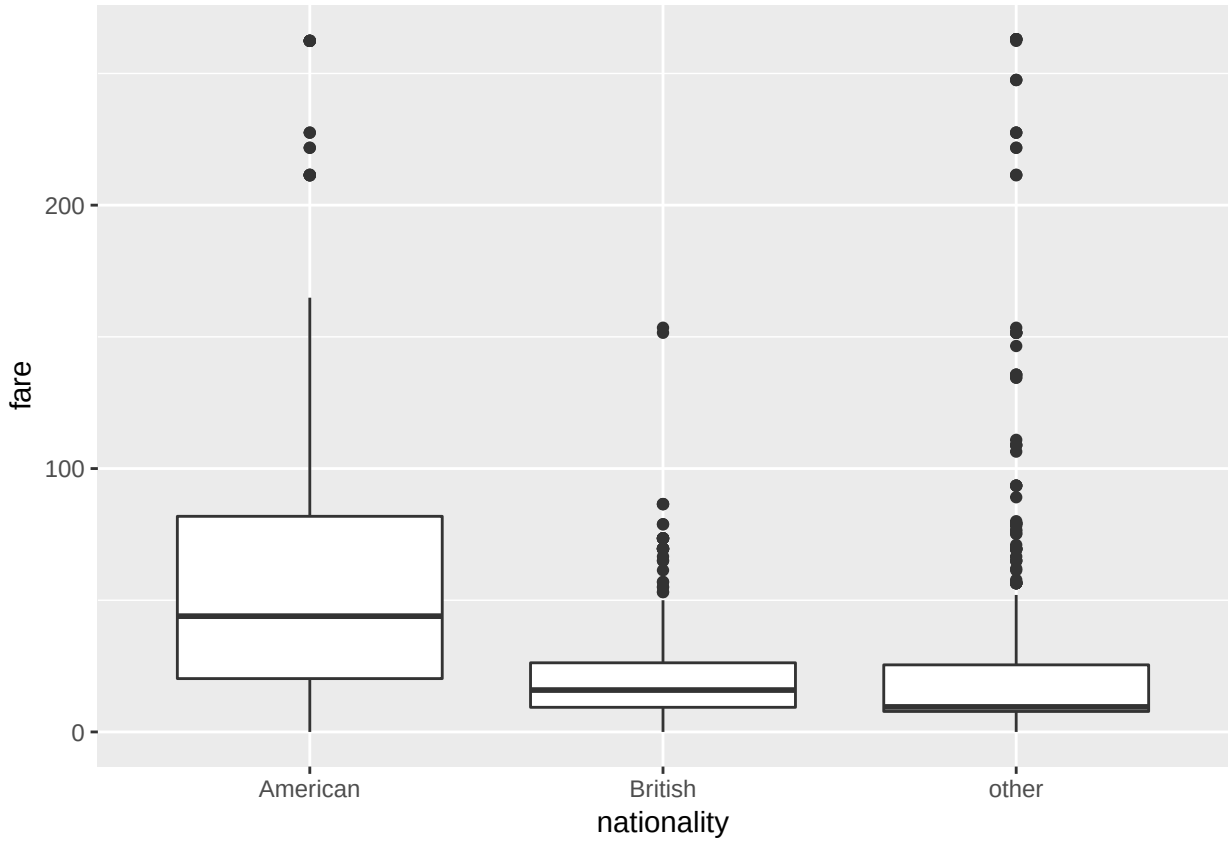
Boxplots with letters using `stat_summary()`

- We can use `stat_summary()` to add letters to boxplots without making a new data frame

```
titanic <- read.csv("titanic.csv") %>% filter(fare < 400)
p <- ggplot(titanic, aes(x = nationality, y = fare)) + geom_boxplot()
```

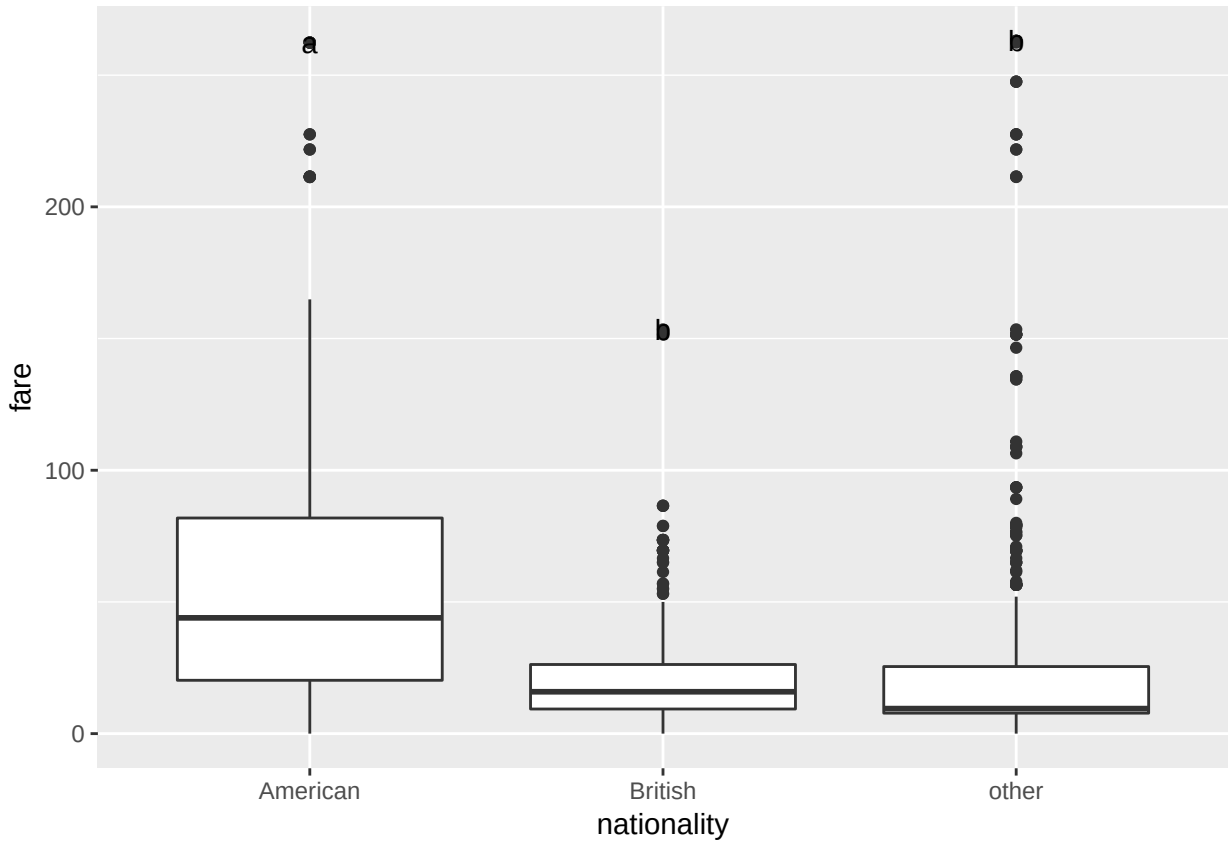
Base boxplot

```
p
```



Adding letters with `stat_summary()`

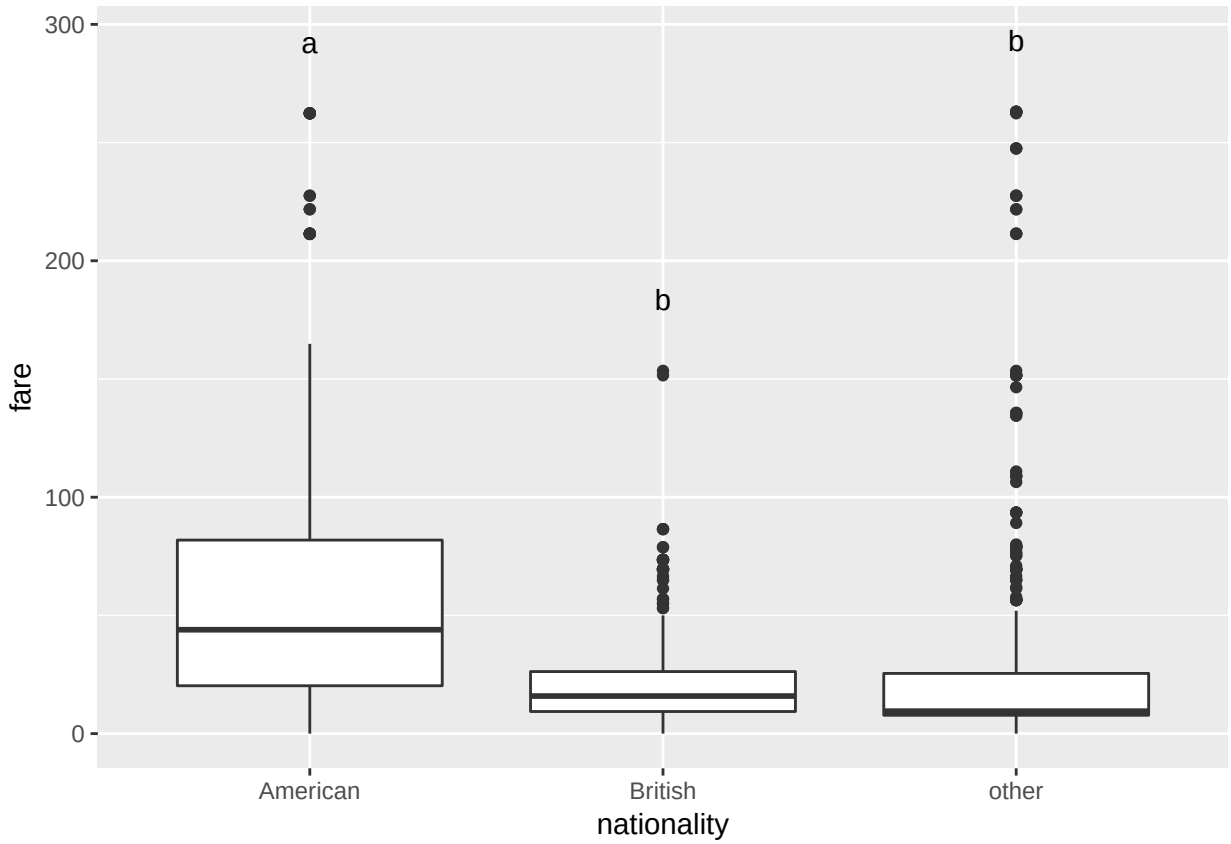
```
p + stat_summary(geom = "text", fun.y = "max", label = c("a", "b", "b"))
```



Adjust letter position

- Nudge the letters up with `position = position_nudge()`

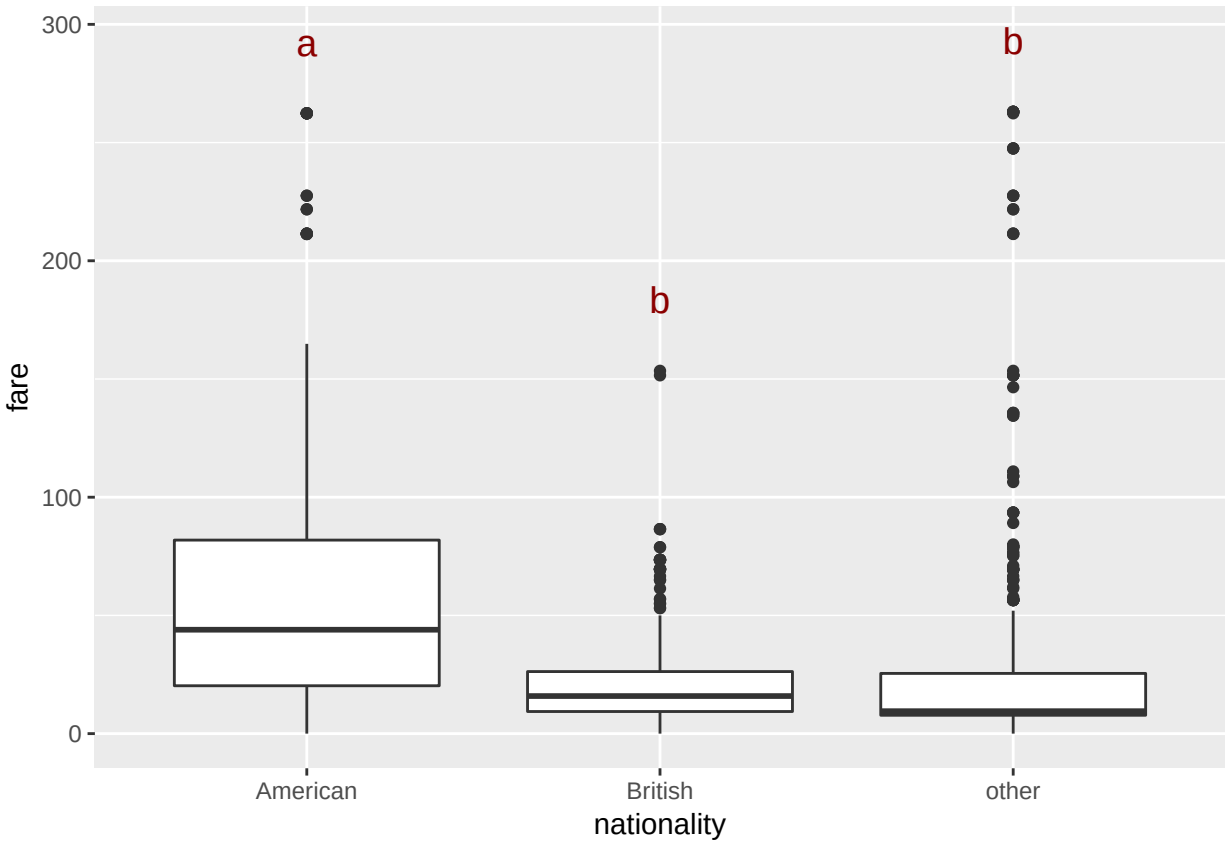
```
p + stat_summary(geom = "text", fun.y = "max", label = c("a", "b", "b"),
                 position = position_nudge(y = 30)) #in units of y-axis
```



Change the look of letters

- additional arguments get passed to `geom_text()`

```
p + stat_summary(geom = "text", fun.y = "max", label = c("a", "b", "b"),
                 position = position_nudge(y = 30), #in units of y-axis
                 size = 5, color = "darkred")
```



Using `stat_summary()` in practice

- Making a summary table first, then plotting may be more *transparent*, easier for others to understand
- *But*, `stat_summary()` can make code cleaner, easier to read
- Complex plots with original data with multiple summary annotations (e.g. means, errorbars, significance letters) are made a **lot** simpler with `stat_summary()`