The bootstrap for sampling distributions

Assessing assumptions

- Our t-tests assume normality of variable being tested
- but, Central Limit Theorem says that normality matters less if sample is "large"
- in practice "approximate normality" is enough, but how do we assess whether what we have is normal enough?
- so far, use histogram/boxplot and make a call, allowing for sample size.

What actually has to be normal

- is: **sampling distribution of sample mean**
- the distribution of sample mean over *all possible samples*
- but we only have *one* sample!
- Idea: assume our sample is representative of the population, and draw samples from our sample (!), with replacement.
- This gives an idea of what different samples from the population might look like.
- Called *bootstrap*, after expression "to pull yourself up by your own bootstraps".

Blue Jays attendances

```
jays$attendance
```

```
##  [1] 48414 17264 15086 14433 21397 34743 44794 14184
##  [9] 15606 18581 19217 21519 21312 30430 42917 42419
## [17] 29306 15062 16402 19014 21195 33086 37929 15168
## [25] 17276
```

- A bootstrap sample:

```
s <- sample(jays$attendance, replace = TRUE)
s
```

```
##  [1] 21195 34743 21312 44794 16402 19014 34743 21195
##  [9] 17264 18581 19014 19217 34743 19217 14433 15062
## [17] 16402 15062 34743 15062 15086 15168 15086 48414
## [25] 30430
```

Getting mean of bootstrap sample

- A bootstrap sample is same size as original, but contains repeated values (eg. 15062) and missing ones (42917).
- We need the mean of our bootstrap sample:

```
mean(s)
```

```
## [1] 23055.28
```

- This is a little different from the mean of our actual sample:

```
mean(jays$attendance)
```

```
## [1] 25070.16
```

- Want a sense of how the sample mean might vary, if we were able to take repeated samples from our population.
- Idea: take lots of *bootstrap* samples, and see how *their* sample means vary.

Setting up bootstrap sampling

- Begin by setting up a dataframe that contains a row for each bootstrap sample. I usually call this column `sim`. Do just 4 to get the idea:

```
tibble(sim = 1:4)
```

```
## # A tibble: 4 x 1
##     sim
##   <int>
## 1     1
## 2     2
## 3     3
## 4     4
```

Drawing the bootstrap samples

- Then set up to work one row at a time, and draw a bootstrap sample of the attendances in each row:

```
tibble(sim = 1:4) %>%
  rowwise() %>%
  mutate(sample = list(sample(jays$attendance, replace = TRUE)))
```

```
## # A tibble: 4 x 2
##     sim sample
##   <int> <list>
## 1     1 <dbl [25]>
## 2     2 <dbl [25]>
## 3     3 <dbl [25]>
## 4     4 <dbl [25]>
```

- Each row of our dataframe contains *all* of a bootstrap sample of 25 observations drawn with replacement from the attendances.

Sample means

- Find the mean of each sample:

```
tibble(sim = 1:4) %>%
  rowwise() %>%
  mutate(sample = list(sample(jays$attendance, replace = TRUE))) %>%
  mutate(my_mean = mean(sample))
```

```
## # A tibble: 4 x 3
##     sim sample     my_mean
##   <int> <list>       <dbl>
## 1     1 <dbl [25]>  28472.
## 2     2 <dbl [25]>  28648.
## 3     3 <dbl [25]>  23329.
## 4     4 <dbl [25]>  24808.
```

- These are (four simulated values of) the bootstrapped sampling distribution of the sample mean.

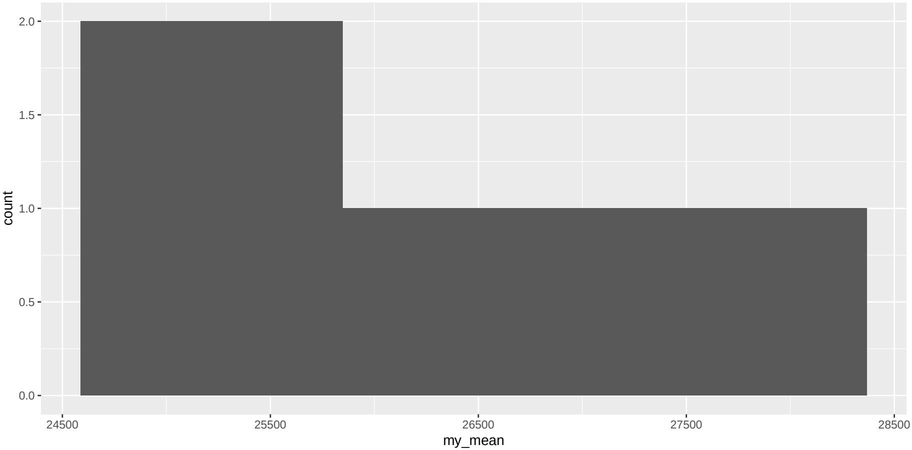
Make a histogram of them

- rather pointless here, but to get the idea:

```
tibble(sim = 1:4) %>%
  rowwise() %>%
  mutate(sample = list(sample(jays$attendance, replace = TRUE))) %>%
  mutate(my_mean = mean(sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 3) -> g
```

The (pointless) histogram

```
g
```



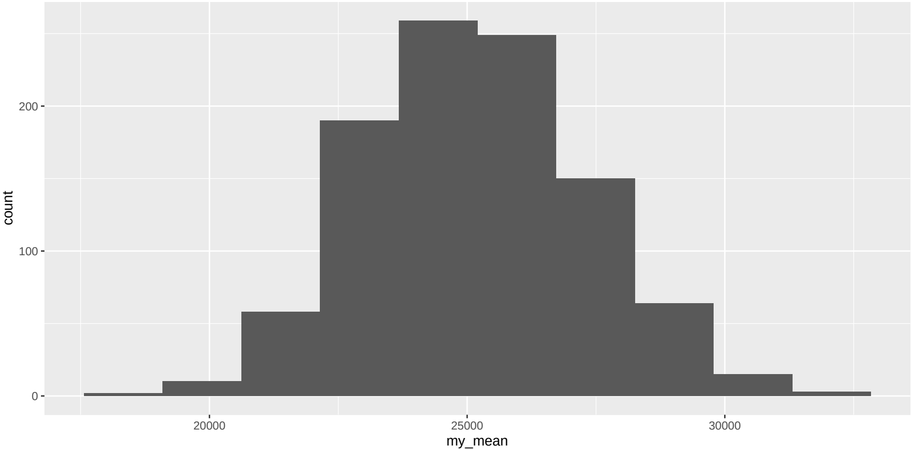
Now do again with a decent number of bootstrap samples

- say 1000, and put a decent number of bins on the histogram also:

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(sample(jays$attendance, replace = TRUE))) %>%
  mutate(my_mean = mean(sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 10) -> g
```

The (better) histogram

```
g
```



Comments

- This is very close to normal
- The bootstrap says that the sampling distribution of the sample mean is close to normal, even though the distribution of the data is not
- A sample size of 25 is big enough to overcome the skewness that we saw
- This is the Central Limit Theorem in practice
- It is surprisingly powerful.
- Thus, the t-test is actually perfectly good here.

Comments on the code

- You might have been wondering about this:

```
tibble(sim = 1:4) %>%
  rowwise() %>%
  mutate(sample = list(sample(jays$attendance, replace = TRUE)))
```

```
## # A tibble: 4 x 2
##     sim sample
##   <int> <list>
## 1     1 <dbl [25]>
## 2     2 <dbl [25]>
## 3     3 <dbl [25]>
## 4     4 <dbl [25]>
```

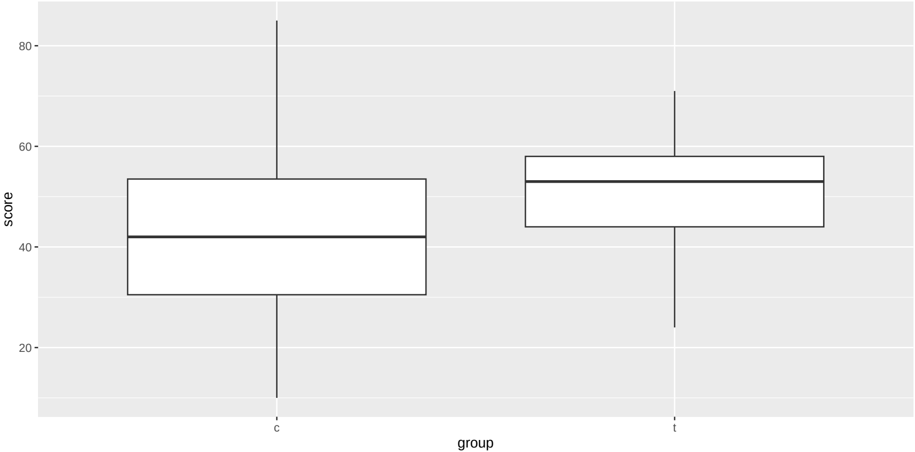
- how did we squeeze all 25 sample values into one cell?
  - sample is a so-called "list-column" that can contain anything.
- why did we have to put `list()` around the `sample()`?
  - because `sample` produces a collection of numbers, not just a single one
  - the `list()` signals this: "make a list-column of samples".

Two samples

- Assumption: *both* samples are from a normal distribution.
- In practice, each sample is "normal enough" given its sample size, since Central Limit Theorem will help.
- Use bootstrap on each group independently, as above.

Kids learning to read

```
ggplot(kids, aes(x=group, y=score)) + geom_boxplot()
```



Getting just the control group

- Use `filter` to select rows where something is true:

```
kids %>% filter(group=="c") -> controls
controls
```

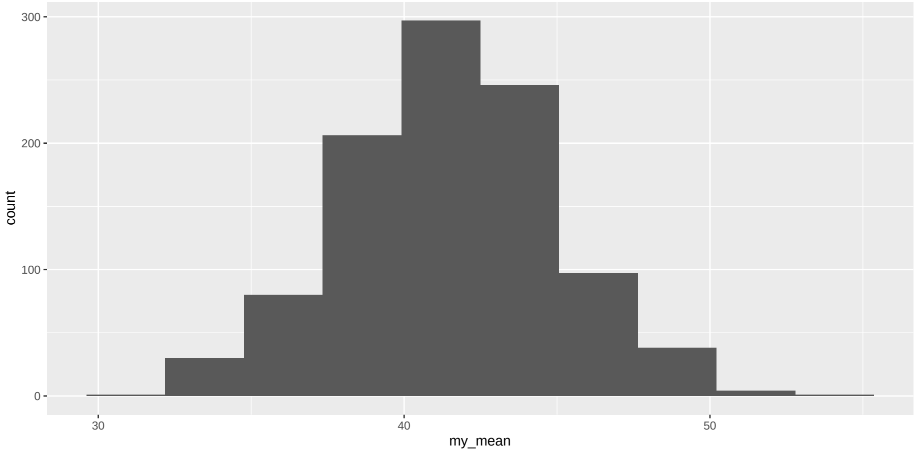
```
## # A tibble: 23 x 2
##    group score
##    <chr> <dbl>
##  1 c        42
##  2 c        33
##  3 c        46
##  4 c        37
##  5 c        43
##  6 c        41
##  7 c        10
##  8 c        42
##  9 c        55
```

Bootstrap these

```
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(sample(controls$score, replace = TRUE))) %>%
  mutate(my_mean = mean(sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 10) -> g
```

Plot

```
g
```

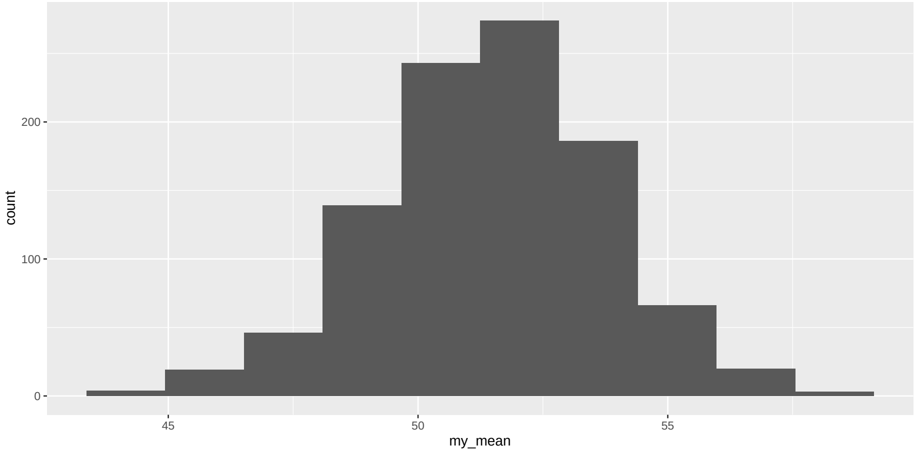


… and the treatment group:

```
kids %>% filter(group=="t") -> treats
tibble(sim = 1:1000) %>%
  rowwise() %>%
  mutate(sample = list(sample(treats$score, replace = TRUE))) %>%
  mutate(my_mean = mean(sample)) %>%
  ggplot(aes(x = my_mean)) + geom_histogram(bins = 10) -> g
```

Histogram

```
g
```



Comments

- sampling distributions of sample means both look pretty normal
- as we thought, no problems with our two-sample t at all.