

# Simulation and the bootstrap

# packages

```
library(tidyverse)
```

# Simulation

- Sometimes you know the exact mathematical answer to a problem, eg:
  - ▶  $X_1, \dots, X_n \sim N(\mu, \sigma^2)$ , what is distribution of  $\bar{X}$ ? (Ans:  $N(\mu, \sigma^2/n)$ .)
- More often, though, you don't:
  - ▶ if  $X \sim \text{Bin}(2, 0.5)$ ,  $Y \sim \text{Bin}(3, 0.2)$ , what is dist of  $Z = X + Y$ ?
- Simulation: generate random  $X$  and  $Y$ , calculate sum, repeat many times. Gives you (approx) dist of  $X + Y$  without any mathematics!

## Random numbers in R

- R knows about a lot of distributions, eg: norm binom pois exp gamma t chisq (type ?distributions in Console to see more)
- to generate random numbers from a distribution, put r on front of these; inputs are number of random values to generate, and parameters of distribution to simulate from.
- Examples:

```
rmnorm(5, 100, 15)
```

```
[1] 111.96072 89.80331 113.74231 90.74167 89.86461
```

and

```
rmnorm(10, 3.5)
```

```
[1] 5 2 5 5 3 5 1 3 4 0
```

## Our problem: simulating once

- if  $X \sim \text{Bin}(2, 0.5)$ ,  $Y \sim \text{Bin}(3, 0.2)$ , what is dist of  $Z = X + Y$ ?

```
x <- rbinom(1, 2, 0.5)
y <- rbinom(1, 3, 0.2)
x
```

```
[1] 1
```

```
y
```

```
[1] 1
```

```
x + y
```

```
[1] 2
```

To simulate many times: - set up dataframe with space for each simulated value - work rowwise - do one simulation per row

# Simulating many times

```
tibble(sim = 1:10000) %>%  
  rowwise() %>%  
  mutate(x = rbinom(1, 2, 0.5),  
         y = rbinom(1, 3, 0.2),  
         z = x + y) -> d  
  
d
```

# A tibble: 10,000 x 4

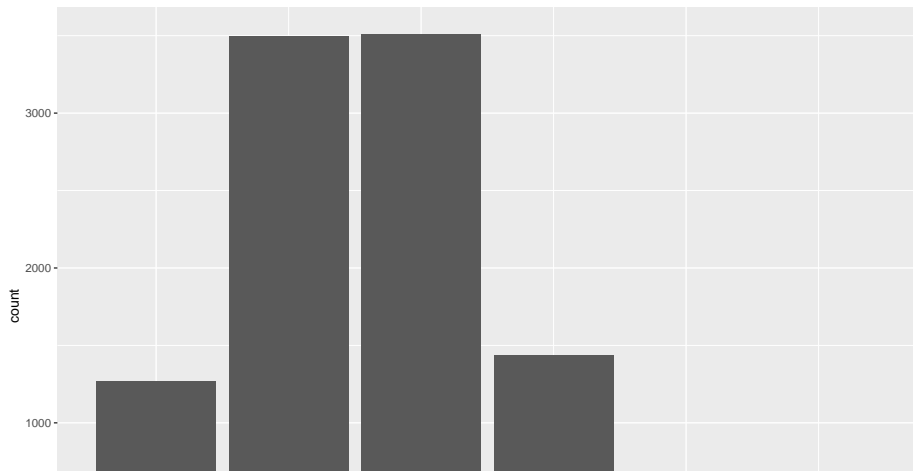
# Rowwise:

	sim	x	y	z
	<int>	<int>	<int>	<int>
1	1	1	0	1
2	2	2	1	3
3	3	1	0	1
4	4	0	0	0
5	5	1	0	1
6	6	2	0	2

## Distribution of sum

Make a bar chart rather than a histogram because distribution of  $Z$  is discrete:

```
ggplot(d, aes(x = z)) + geom_bar()
```



(Simulated) probability that the sum is at least 4:

```
d %>% count(z >= 4)
```

```
# A tibble: 2 x 2
# Rowwise:
  `z >= 4`      n
  <lgl>      <int>
1 FALSE      9715
2 TRUE       285
```

Only this much:

```
285/10000
```

```
[1] 0.0285
```

A sum of 5 is possible though very unlikely.



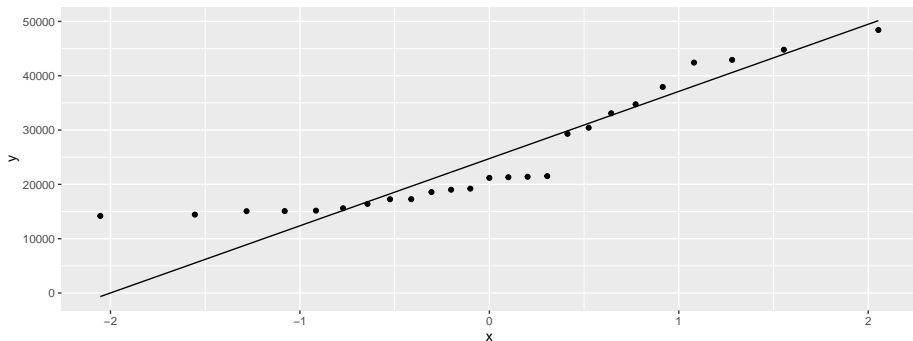
# The bootstrap

Source: Hesterberg et al

- Sampling distribution of a statistic is distribution of that statistic over “all possible samples” from population of interest.
- “Plug-in principle”: sample mean estimates population mean, sample variance estimates population variance, etc.
- Also, sample is estimate of population (precisely, proportion of sample values  $\leq x$  estimates probability of drawing value  $\leq x$  from population, for any  $x$ ).
- As long as your sample is representative, sampling *from the sample* (!) is an estimate of sampling from the population. Called a *bootstrap sample*.
- Sample from sample *with* replacement, or else you get original sample back.

## Blue Jays attendances:

```
ggplot(jays, aes(sample = attendance)) +  
  stat_qq() + stat_qq_line()
```



- $t$  procedure for the mean may not be a good idea because the distribution is skewed.
- Previously: hand-waving with sample size.

## What *actually* matters

- It's not the distribution of the *data* that has to be approx normal (for a *t* procedure).
- What matters is the *sampling distribution of the sample mean*.
- If the sample size is large enough, the sampling distribution will be normal enough even if the data distribution is not.
  - ▶ This is why we had to consider the sample size as well as the shape.
- But how do we know whether this is the case or not? We only have *one* sample.
- Use the bootstrap to simulate sampling distribution.

# Simulating the sampling distribution of sample statistic

- Sample from our sample *with replacement*.
- Calculate statistic
- Repeat many times (simulation).
- This gives an idea of how our statistic might vary in repeated samples: that is, its sampling distribution.
- Called the **bootstrap distribution** of the statistic.
- If the bootstrap distribution is approx normal, infer that the true sampling distribution also approx normal, therefore inference about the mean such as  $t$  is good enough.
- If not, we should be more careful.

## Bootstrapping the Blue Jays attendances

- Sampling with replacement is done like this (the default sample size is as long as the original data):

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

```
[1] 14184 14433 15062 15086 15168 15606 16402 17264 17276 18581  
[13] 21195 21312 21397 21519 29306 30430 33086 34743 37929 42419  
[25] 48414
```

```
sort(boot)
```

```
[1] 14184 14433 14433 15086 15086 15168 15168 16402 17264 17276  
[13] 17276 17276 18581 19014 21312 21519 33086 37929 42419 42419  
[25] 48414
```

```
mean(boot)
```

```
[1] 23764.76
```

## A whole bunch

- We are now doing a simulation. I like 10,000 samples when testing for normality:

```
tibble(sim = 1:10000) %>%  
  rowwise() %>%  
  mutate(boot_sample = list(sample(jays$attendance, replace =  
    mutate(my_mean = mean(boot_sample)) -> samples  
samples
```

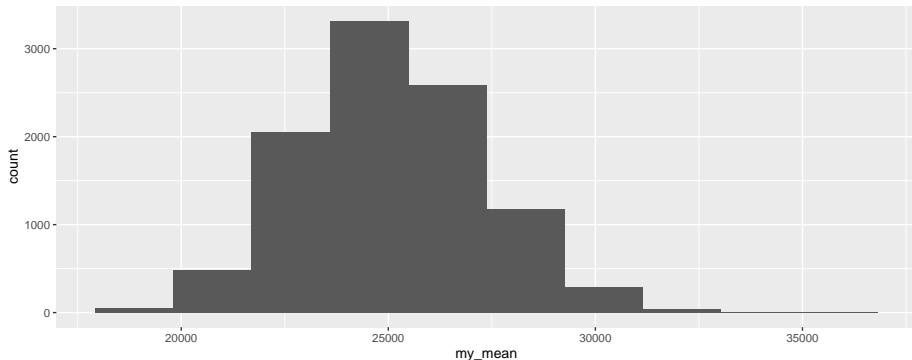
# A tibble: 10,000 x 3

# Rowwise:

	sim	boot_sample	my_mean
	<int>	<list>	<dbl>
1	1	<dbl [25]>	23055.
2	2	<dbl [25]>	25513.
3	3	<dbl [25]>	25563.
4	4	<dbl [25]>	29198.
5	5	<dbl [25]>	23615.

# Sampling distribution of sample mean

```
ggplot(samples, aes(x=my_mean)) + geom_histogram(bins=10)
```

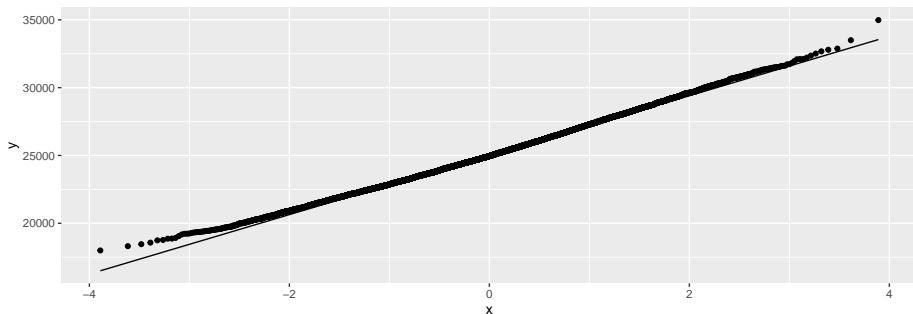


- Is that a slightly long right tail?

## Normal quantile plot

might be better than a histogram:

```
ggplot(samples, aes(sample = my_mean)) +  
  stat_qq()+stat_qq_line()
```



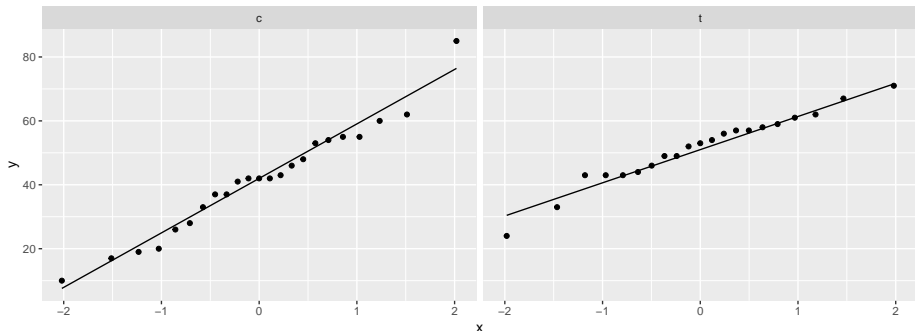
- a very very slight right-skewness, but very close to normal.
- hence the  $t$ -test is fine for the Blue Jays attendances.



# Kids learning to read

- Normal quantile plots, one for each sample:

```
ggplot(kids, aes(sample = score)) +  
  stat_qq() + stat_qq_line() +  
  facet_wrap(~ group)
```



- These both look close to normal.

## Control group

- Pull out control group children
- Obtain bootstrap sampling distribution of scores

```
kids %>% filter(group == "c") -> controls
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(controls$score, replace = TRUE)))
  mutate(my_mean = mean(my_sample)) -> samples1
samples1
```

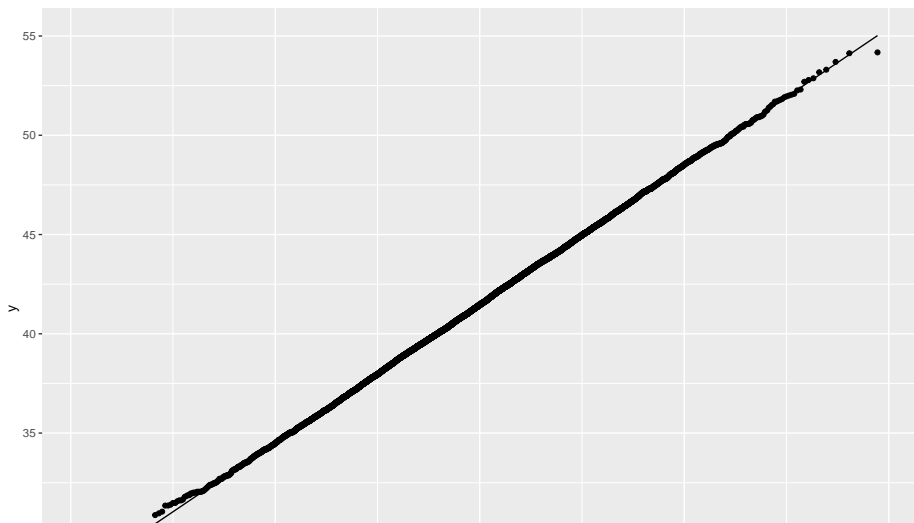
# A tibble: 10,000 x 3

# Rowwise:

	sim	my_sample	my_mean
	<int>	<list>	<dbl>
1	1	<dbl [23]>	43.7
2	2	<dbl [23]>	38.9
3	3	<dbl [23]>	44.3
4	4	<dbl [23]>	40.8

# The bootstrap sampling distribution

```
ggplot(samples1, aes(sample = my_mean)) +  
  stat_qq() + stat_qq_line()
```

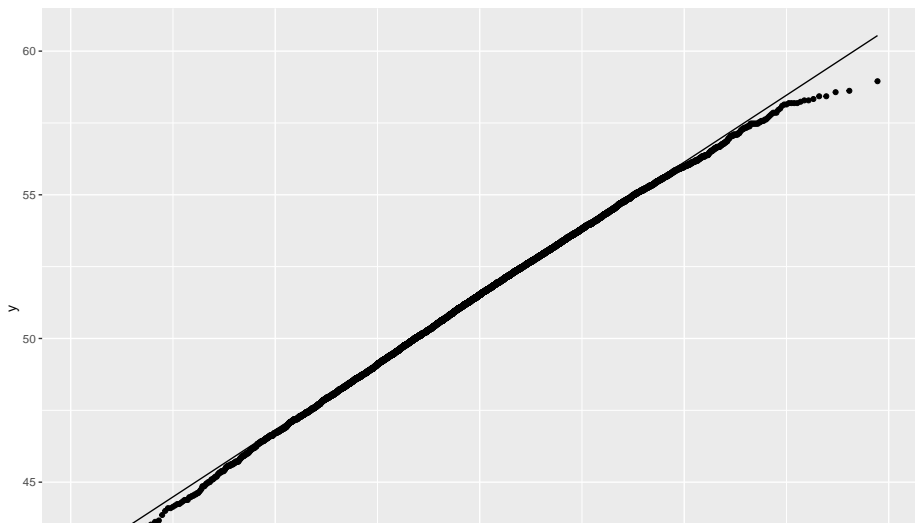


## Same, for treatment group

```
kids %>% filter(group == "t") -> treated
tibble(sim = 1:10000) %>%
  rowwise() %>%
  mutate(my_sample = list(sample(treated$score, replace = TRUE)))
  mutate(my_mean = mean(my_sample)) -> samples2
```

# The bootstrap sampling distribution

```
ggplot(samples2, aes(sample = my_mean)) +  
  stat_qq() + stat_qq_line()
```



## Pain relief

- With matched pairs, assumption is of normality of *differences*:

```
pain %>% mutate(diff = drug_a - drug_b) -> pain  
pain
```

```
# A tibble: 12 x 4
```

	subject	drug_a	drug_b	diff
	<dbl>	<dbl>	<dbl>	<dbl>
1	1	2	3.5	-1.5
2	2	3.6	5.7	-2.1
3	3	2.6	2.9	-0.300
4	4	2.6	2.4	0.200
5	5	7.3	9.9	-2.6
6	6	3.4	3.3	0.100
7	7	14.9	16.7	-1.80
8	8	6.6	6	0.600
9	9	2.3	3.8	-1.5
10	10	2	4	-2

# Bootstrap sampling distribution of differences

```
tibble(sim = 1:10000) %>%  
  rowwise() %>%  
  mutate(my_sample = list(sample(pain$diff, replace = TRUE)))  
  mutate(my_mean = mean(my_sample)) -> samples  
samples
```

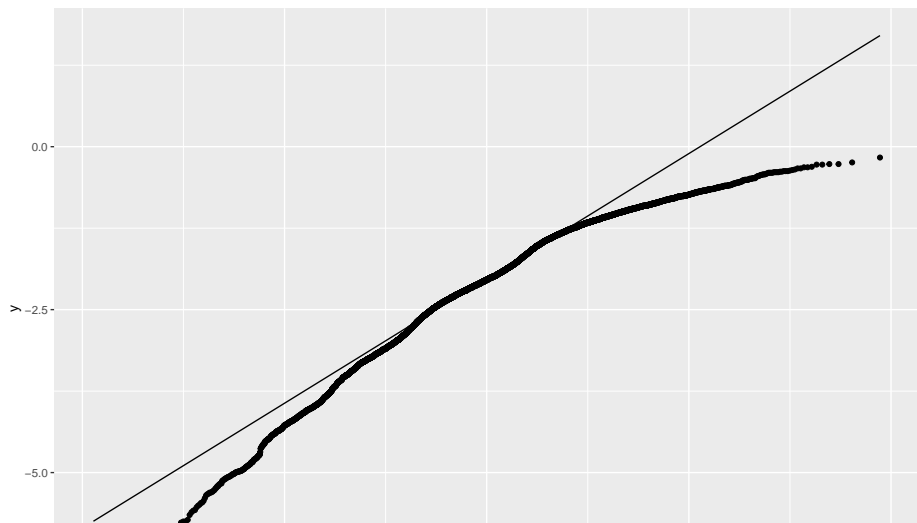
# A tibble: 10,000 x 3

# Rowwise:

	sim	my_sample	my_mean
	<int>	<list>	<dbl>
1	1	<dbl [12]>	-1.82
2	2	<dbl [12]>	-3.58
3	3	<dbl [12]>	-2.66
4	4	<dbl [12]>	-2.9
5	5	<dbl [12]>	-0.975
6	6	<dbl [12]>	-2.25
7	7	<dbl [12]>	-2.32

## Assess it for normality

```
ggplot(samples, aes(sample = my_mean)) +  
  stat_qq() + stat_qq_line()
```





## Histogram with many bins

```
ggplot(samples, aes(x = my_mean)) + geom_histogram(bins = 60)
```

