Lecture 12
Resampling (bootstrap and permutation)

CHOONG-WAN WOO | COCOAN lab | http://cocoanlab.github.io

What we learned...

- Sampling distribution
- Standard error of the mean
- Confidence interval
- One-sample t-test
- Paired t-test
- Independent samples t-test

Monte Carlo simulation

Stan Ulam

Nicholas Metropolis

- Developed by **Stan Ulam** and **Nicholas Metropolis**
- As a part of the Manhattan Project
    - Developing an atomic weapon
    - Needed to compute the average distance that a neutron would travel in a substance before the collision
    - Ulam realized that he could use a simulation using random numbers, just like a casino game
    - Ulam's uncle had gambled at the Monte Carlo casino in Monaco, and they named this method with this Monte Carlo

Monte Carlo simulation: Four basic steps

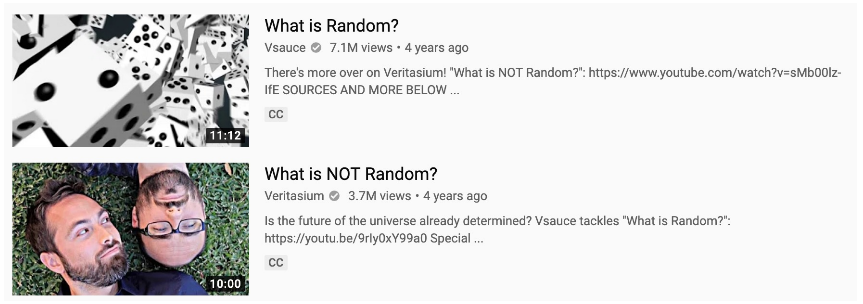
- Four basic steps:
  1. Define a domain of possible values
  2. Generate **random** numbers within that domain from a probability distribution
  3. Perform a computation using the **random** numbers
  4. Combine the results across many repetitions

What is *random* in statistics?

- A process is *random* if it is unpredictable
- It doesn’t mean that it is not deterministic
    - E.g., tossing a coin
    - If we know the initial condition perfectly, we could predict the outcome of the coin flip
    - However, in practice, many factors that influence the coin flip make it unpredictable

- Really difficult to make *truly random* numbers
- Usually use “pseudo-random” numbers

Recommended youtube video



What is Random?
Vsauce 7.1M views • 4 years ago
There's more over on Veritasium! "What is NOT Random?": https://www.youtube.com/watch?v=sMb00lz-IfE SOURCES AND MORE BELOW ...

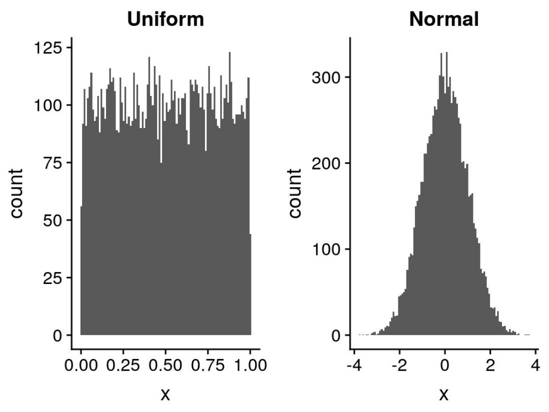
What is NOT Random?
Veritasium 3.7M views • 4 years ago
Is the future of the universe already determined? Vsauce tackles "What is Random?": https://youtu.be/9rly0xY99a0 Special ...

https://www.youtube.com/watch?v=9rIy0xY99a0

Generating random numbers using R and Matlab

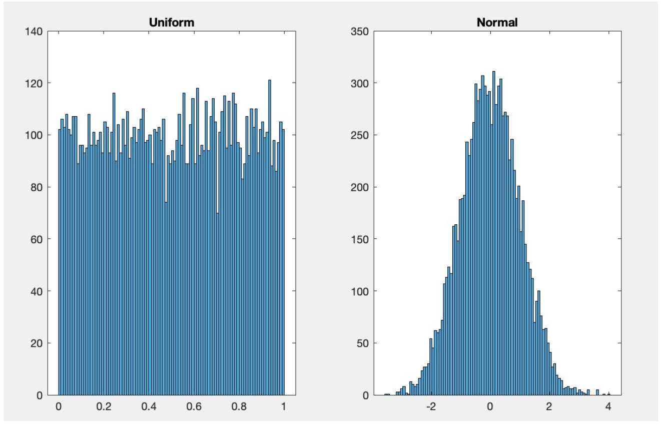
- In R

```
p1 <-
  tibble(
    x = runif(10000)
  ) %>%
  ggplot((aes(x))) +
  geom_histogram(bins = 100) +
  labs(title = "Uniform")

p2 <-
  tibble(
    x = rnorm(10000)
  ) %>%
  ggplot(aes(x)) +
  geom_histogram(bins = 100) +
  labs(title = "Normal")

plot_grid(p1, p2, ncol = 3)
```



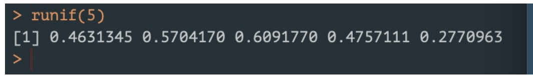
- In Matlab

```
x1 = rand(10000,1);
subplot(1,2,1);
histogram(x1, 100);
title('Uniform')

x2 = randn(10000,1);
subplot(1,2,2);
histogram(x2, 100);
title('Normal')
```

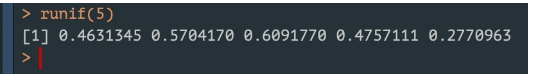


Random seeds

- Random seed
  - If you use the same random seed, the random number will be same
  - It is possible because all stat languages uses *pseudo-random* numbers
  - In R

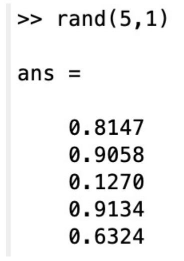


```
> runif(5)
[1] 0.4631345 0.5704170 0.6091770 0.4757111 0.2770963
>
```

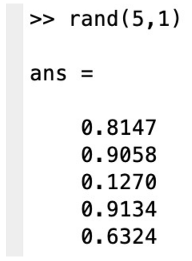


```
> runif(5)
[1] 0.4631345 0.5704170 0.6091770 0.4757111 0.2770963
>
```

  - In Matlab



```
>> rand(5,1)

ans =

    0.8147
    0.9058
    0.1270
    0.9134
    0.6324
```



```
>> rand(5,1)

ans =

    0.8147
    0.9058
    0.1270
    0.9134
    0.6324
```

  - `set.seed` in R and `rng` in Matlab can be used to set the random seed
  - How to shuffle the random seed? `set.seed(Sys.time())` in R, `rng('shuffle')` in Matlab

    → set the random seed based on the current time

Example of Monte Carlo simulation

- Example: how much time to allow for an in-class quiz?

- Finishing time distribution: Normal model
- What we want to know is the distribution of the *longest* finishing time

```r
# sample maximum value 5000 times and compute 99th percentile
nRuns <- 5000
sampSize <- 150

sampleMax <- function(sampSize = 150) {
  samp <- rnorm(sampSize, mean = 5, sd = 1)
  return(max(samp))
}

maxTime <- replicate(nRuns, sampleMax())

cutoff <- quantile(maxTime, 0.99)
sprintf("99th percentile of maxTime distribution: %.2f", cutoff)
```

```
## [1] "99th percentile of maxTime distribution: 8.81"
```

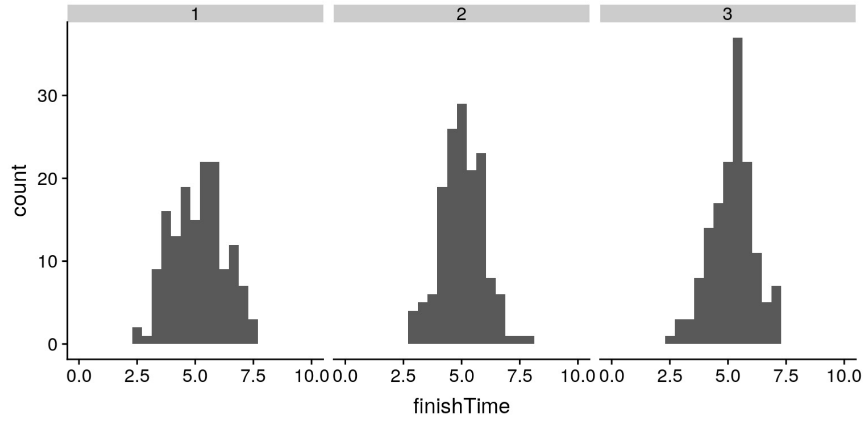


Figure 8.2: Simulated finishing time distributions.

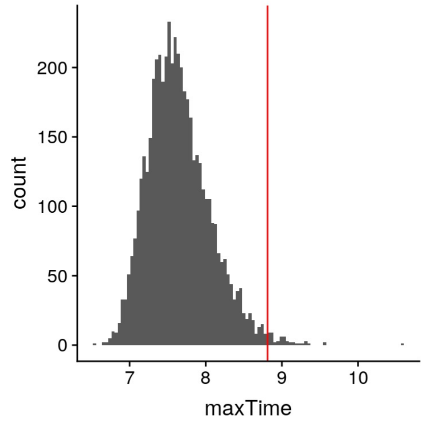
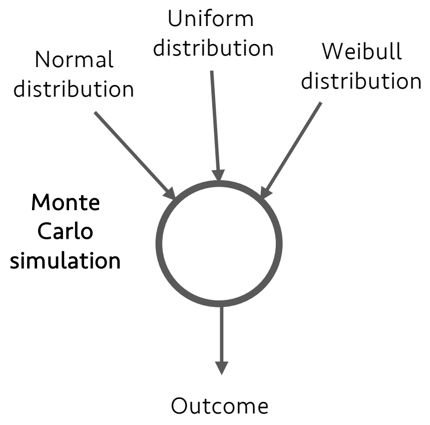


Figure 8.3: Distribution of maximum finishing times across simulations.

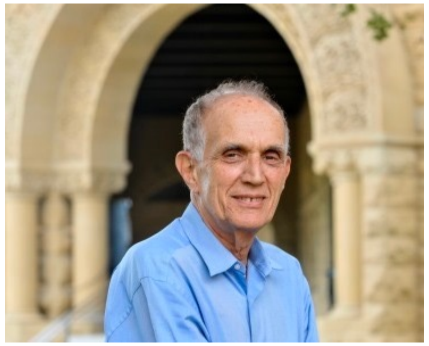


Using simulation for statistics: The bootstrap

- *Bootstrap*: resampling methods to quantify the uncertainty about statistical estimates

- The basic idea: Use the sample data themselves to get the distribution of the statistical estimates
  - Resample with replacement
    - The same data point could represent multiple times within one of the samples
    - We then compute our statistic of interest on each of the bootstrap samples, and use the distribution of those estimates.

Bradley Efron

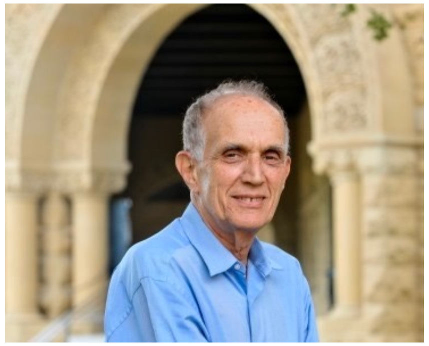


Using simulation for statistics: The bootstrap

- *Bootstrap*: resampling methods to quantify the uncertainty about statistical estimates

- The basic idea: Use the sample data themselves to get the distribution of the statistical estimates
  - Resample with replacement
    - The same data point could represent multiple times within one of the samples
    - We then compute our statistic of interest on each of the bootstrap samples, and use the distribution of those estimates.
  - We don't need the normality assumption for using bootstrap!
  - Usage: standard error of the mean, confidence interval, null-hypothesis testing, etc.
  - Conceived by Bradley Efron

Bradley Efron



Computing SEM with bootstrap

```
# perform the bootstrap to compute SEM and compare to parametric method

nRuns <- 2500
sampleSize <- 32

heightSample <-
  NHANES_adult %>%
  sample_n(sampleSize)

bootMeanHeight <- function(df) {
  bootSample <- sample_n(df, dim(df)[1], replace = TRUE)
  return(mean(bootSample$Height))
}

bootMeans <- replicate(nRuns, bootMeanHeight(heightSample))

SEM_standard <- sd(heightSample$Height) / sqrt(sampleSize)
sprintf("SEM computed using sample SD: %f", SEM_standard)
```

Based on a bootstrap test

```
## [1] "SEM computed using sample SD: 1.595789"
```

```
SEM_bootstrap <- sd(bootMeans)
sprintf("SEM computed using SD of bootstrap estimates: %f", SEM_bootstrap)
```

Based on normality assumption

```
## [1] "SEM computed using SD of bootstrap estimates: 1.586913"
```

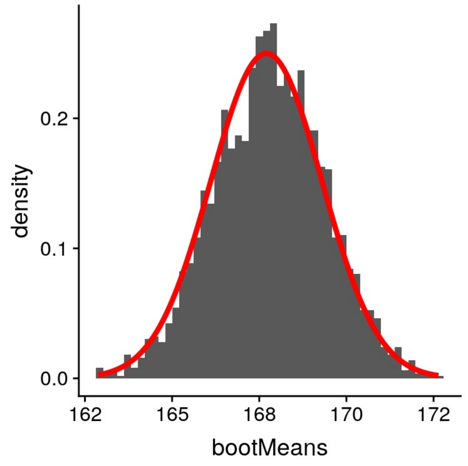
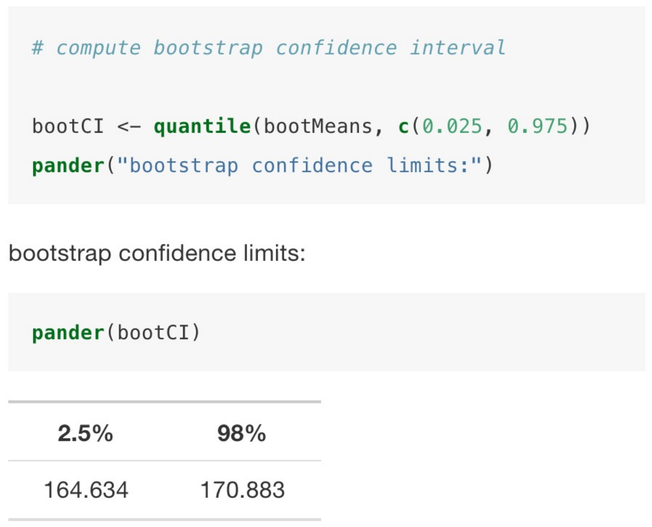


Figure 8.4: An example of bootstrapping to compute the standard error of the mean. The histogram shows the distribution of means across bootstrap samples, while the red line shows the normal distribution based on the sample mean and standard deviation.

For the mean, you can see that the distribution is fairly close to the theoretical estimate based on the normality assumption

Computing Confidence interval with bootstrap

Based on a bootstrap test



```r
# compute bootstrap confidence interval

bootCI <- quantile(bootMeans, c(0.025, 0.975))
pander("bootstrap confidence limits:")
```

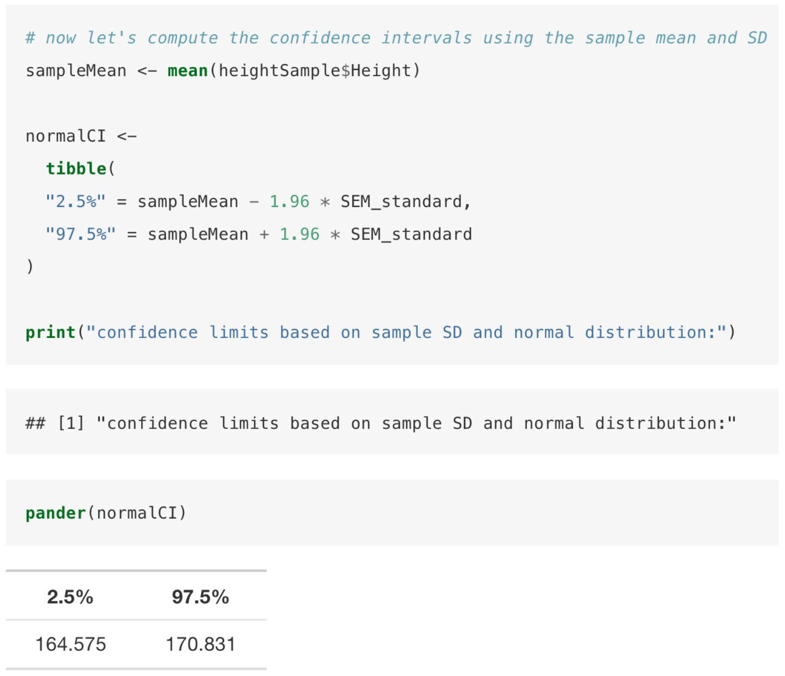
bootstrap confidence limits:

```r
pander(bootCI)
```

2.5%	98%
164.634	170.883

- We usually don’t use bootstrap tests for mean (central limit theorem). But why not?
- The bootstrap would more often be used to generate standard errors for estimates of other statistics where we know or suspect that the normal distribution is not appropriate.

Based on a normality assumption



```r
# now let's compute the confidence intervals using the sample mean and SD
sampleMean <- mean(heightSample$Height)

normalCI <-
  tibble(
  "2.5%" = sampleMean - 1.96 * SEM_standard,
  "97.5%" = sampleMean + 1.96 * SEM_standard
)

print("confidence limits based on sample SD and normal distribution:")
```

```
## [1] "confidence limits based on sample SD and normal distribution:"
```

```r
pander(normalCI)
```

2.5%	97.5%
164.575	170.831

FYI, Matlab function for bootstrap tests on mean values:

https://github.com/cocoanlab/cocoanCORE/blob/master/Statistics/boot_mean_wani.m

Permutation tests

- Using simulation to calculate p-value
- Basic idea:
  - Generate simulated data under the null hypothesis
  - Then, ask how extreme the observed data are in comparison to those simulated data (null)

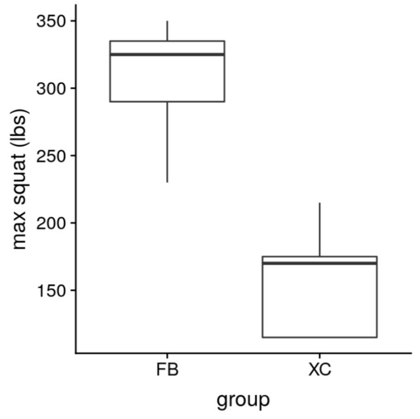
- The key question then:
  - How can we generate data for which the null hypothesis is true?
  - The general answer: we can randomly rearrange the data in a specific way that makes the data look like they would if the null was really true

Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$
$$H_A : \mu_{FB} > \mu_{XC}$$

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215



t-test using t.test() in R

```
# compute and print t statistic comparing two groups

tt <-
  t.test(
    squat ~ group,
    data = squatDf,
    alternative = "greater",
    var.equal = TRUE
  )

sprintf("p(t > %0.2f, df = 8) = %0.5f", tt$statistic, tt$p.value)
```

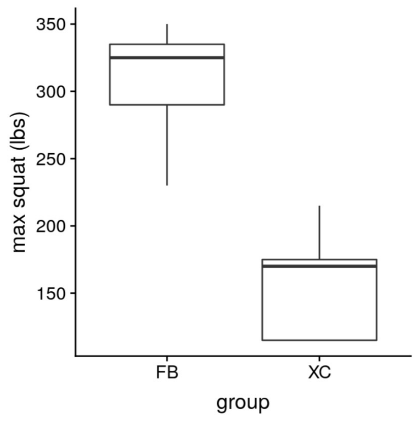
```
## [1] "p(t > 5.14, df = 8) = 0.00044"
```

Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$
$$H_A : \mu_{FB} > \mu_{XC}$$

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215



Now, shuffle the label!

```
# create a scrambled version of the group membership variable

dfScram <-
  squatDf %>%
  mutate(
    scrambledGroup = sample(group)
  ) %>%
  select(-group)

pander(dfScram)
```

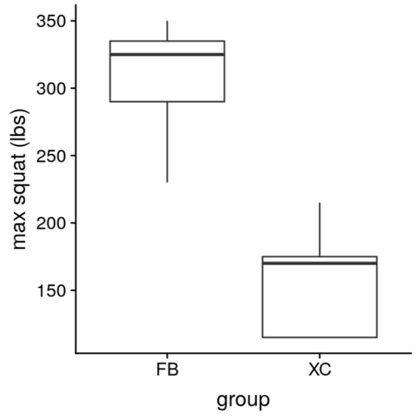
Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$

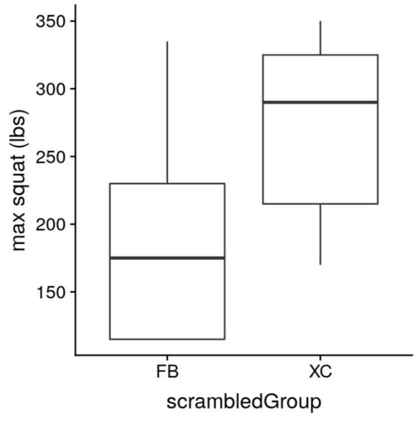
$$H_A : \mu_{FB} > \mu_{XC}$$

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215



Now, shuffle the label!

squat	scrambledGroup
335	FB
350	XC
230	FB
290	XC
325	XC
115	FB
115	FB
170	XC
175	FB
215	XC

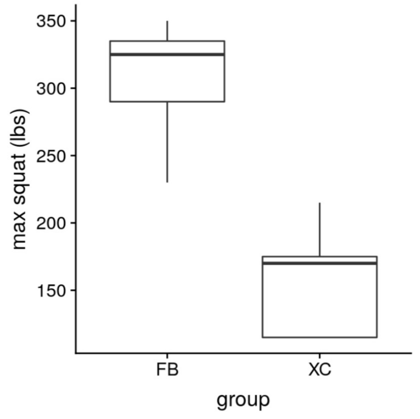


Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$
$$H_A : \mu_{FB} > \mu_{XC}$$

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215



Run this many times (e.g., 10,000 iterations)

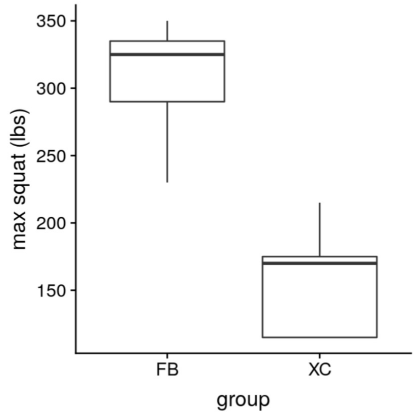
```
# shuffle data 10,000 times and compute distribution of t values

nRuns <- 10000

shuffleAndMeasure <- function(df) {
  dfScram <-
    df %>%
    mutate(
      scrambledGroup = sample(group)
    )
  tt <- t.test(
    squat ~ scrambledGroup,
    data = dfScram,
    alternative = "greater",
    var.equal = TRUE
  )
  return(tt$statistic)
}

shuffleDiff <- replicate(nRuns, shuffleAndMeasure(squatDf))
```

Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$
$$H_A : \mu_{FB} > \mu_{XC}$$

group	squat
FB	335
FB	350
FB	230
FB	290
FB	325
XC	115
XC	115
XC	170
XC	175
XC	215



The results from the permutation test:

```
# compute p value using randomization
pvalRandomization <- mean(shuffleDiff >= tt$statistic)

sprintf(
  'p(t > %0.2f, df = 8) using randomization = %0.5f',
  tt$statistic,
  pvalRandomization
)
```

```
## [1] "p(t > 5.14, df = 8) using randomization = 0.00330"
```

Example of permutation tests

- Comparing mean squatting ability of football players (FB) with cross-country runners (XC)

$$H_0 : \mu_{FB} \leq \mu_{XC}$$
$$H_A : \mu_{FB} > \mu_{XC}$$

t-test using t.test() in R

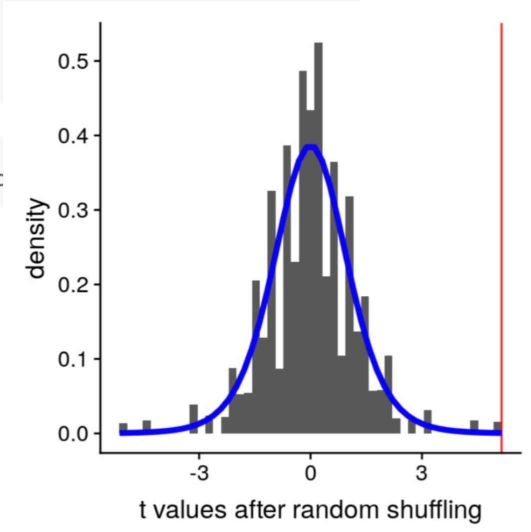
```
# compute and print t statistic comparing two groups

tt <-
  t.test(
    squat ~ group,
    data = squatDf,
    alternative = "greater",
    var.equal = TRUE
  )

sprintf("p(t > %0.2f, df = 8) = %0.5f", tt$statistic, tt$p.value)
```

```
## [1] "p(t > 5.14, df = 8) = 0.00044"
```

The results from the permutation test:

```
# compute p value using randomization
pvalRandomization <- mean(shuffleDiff >= tt$statistic)

sprintf(
  'p(t > %0.2f, df = 8) using randomization = %0.5f',
  tt$statistic,
  pvalRandomization
)
```

```
## [1] "p(t > 5.14, df = 8) using randc
```



Examples of permutation tests

- Don't require an assumption about the normal distribution
- Allow us to compute p-values for statistics even if we don't know its distribution

- One main assumption: *Exchangeability*
  - All of the observations are distributed in the same way; we can interchange them without changing the overall distribution
  - When can this break down? There are dependencies among observations in the data
    - E.g., data from individuals in different families; some would be closer to each other than to others
  - In general, if the data were obtained by random sampling, then the assumption of exchangeability should hold.