

# Resampling statistics and multiple testing

---

STEPHANIE J. SPIELMAN, PHD

BIO5312, FALL 2017



# While you wait, install the following R packages:

---

1. devtools

2. coin

3. modelr

4. broom

# Computer-intensive methods

---

1. Monte Carlo simulation
2. Permutation/randomization test
3. Bootstrap
  1. Here, for computing CI and SE

# Why use simulation

---

Test statistics have a true sampling distribution under specific conditions

- $t$ -statistics from data have a  $t$  sampling distribution when data is normal and/or  $N$  is sufficiently large
- $\chi^2$  statistics from data have a  $\chi^2$  sampling distribution when  $N$  is sufficiently large

We can use **simulation** to **approximate** an analytically unknown/untractable sampling distribution for given conditions

# Monte Carlo simulation

---

Involves randomly sampling data from a probability distribution

### Random sample of N=15 for  $N(3.5, 4)$

```
> rnorm(15, 3.5, 2)
 [1]  5.8166897  3.3402803  2.1469112  3.3038842  5.0005214  2.9234959
 [7]  4.6663932  3.1889110  6.2384465  5.8085365  1.7456411  0.3697735
[13] -1.4639567  5.8856386  8.2788721
```

# Using simulation for hypothesis testing

---

1. Decide your test statistic
  - Make your own quantity or use a "standard" quantity ( $t$ ,  $\chi^2$ , etc.)
2. Generate  $S$  independent datasets under conditions of interest
3. Compute test statistic  $T_s$  for each simulated dataset
  - $T_1, T_2, T_3, \dots, T_S \rightarrow$  Sampling distribution = null
4. Compute P-value by finding your test statistic in this  $T_s$  distribution

**Use computers to imitate the process of repeated sampling from a population to approximate the null distribution of the test statistic.**

# Example simulation study

---

I have a sample of 10 ducks: 4 are blue, 4 are green, and 2 are purple. Are duck colors evenly distributed?

Ho: Duck colors are evenly distributed 1:1:1.

Ha: Duck colors are not evenly distributed 1:1:1

Duck color	# Observed	# expected
blue	4	3.33
green	4	3.33
purple	2	3.34

$\chi^2$  assumption violated!

# Performing the simulation study

---

1. Choose my test statistic, here  $\chi^2$
2. Simulate a duck group and compute the  $\chi^2$  statistic
3. Repeat **a lot** of times (1e4, 1e5)



# Simulating a single group of ducks

---

```
### Simulate a group of ducks
> duck1 <- sample(c("blue", "green", "purple"), 10, replace=T)

### compute chi-squared statistic from duck group
> table(duck1)
duck1
  blue  green purple
    4     3     3

> e <- 10/3
> ((4-e)^2)/e + ((3-e)^2)/e + ((3-e)^2)/e
[1] 0.2
```

**Do this  $1e5$  times to get  $1e5\chi^2$  test statistics**

# The full duck simulation

---

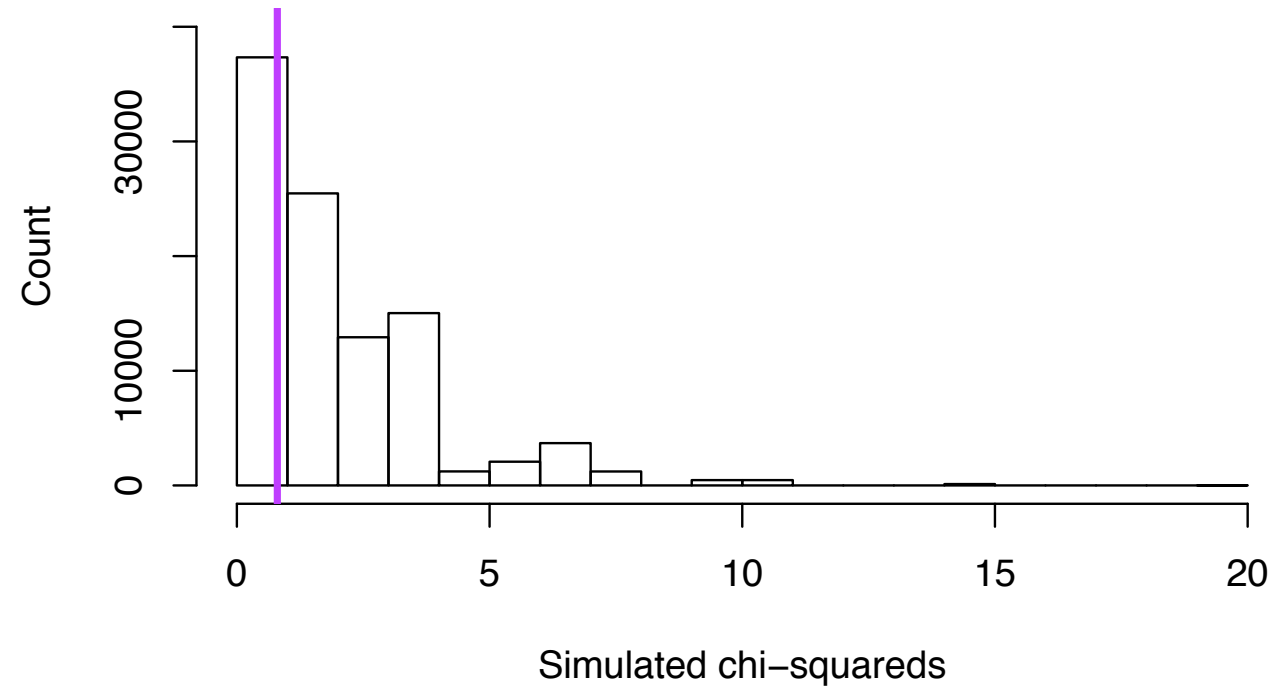
```
all.chisq <- c()
e <- 10/3
for (x in 1:1e5){
  duck <- sample(c("blue", "green", "purple"), 10, replace=T)
  b <- sum(duck == "blue")
  g <- sum(duck == "green")
  p <- sum(duck == "purple")
  chisqu <- ((b-e)^2)/e + ((g-e)^2)/e + ((p-e)^2)/e
  all.chisq <- c(all.chisq, chisqu)
}
```

```
length(all.chisq)
[1] 100000
```

# Duck simulation testing

Duck color	# Observed	# expected
blue	4	3.33
green	4	3.33
purple	2	3.34

$\chi^2 = 0.8$



```
## Compute the probability of being as or more extreme as test statistic  
> sum(all.chisq >= 0.8)/length(all.chisq)  
[1] 0.78705
```

# Duck simulation results, conclusions

---

With  $P=0.787$ , we fail to reject the null hypothesis. We have no evidence that duck color distributions differ from 1:1:1.

# Permutation (randomization) test

---

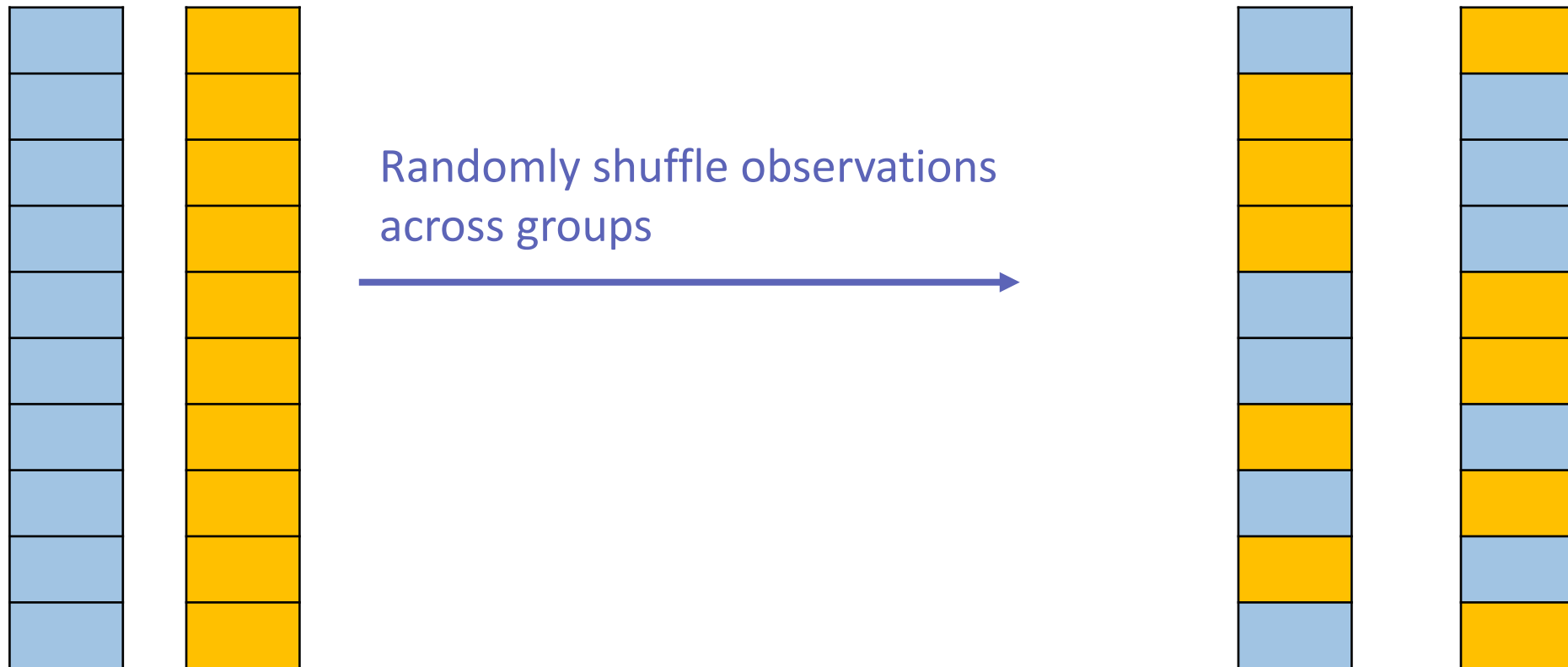
A computer-intensive non-parametric approach for comparing samples

Approach:

1. Choose a statistic that measures the effect you are looking for.
2. Construct the sampling distribution that this statistic would have if the effect were *not* present
3. Locate the observed statistic (i.e. from your data) on this distribution. Area to the tail = Pvalue.

# Permutation tests randomize observed data

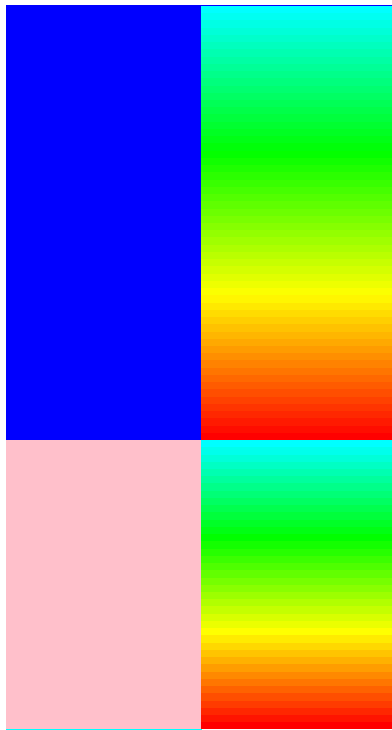
---



# Permutation when null is true

---

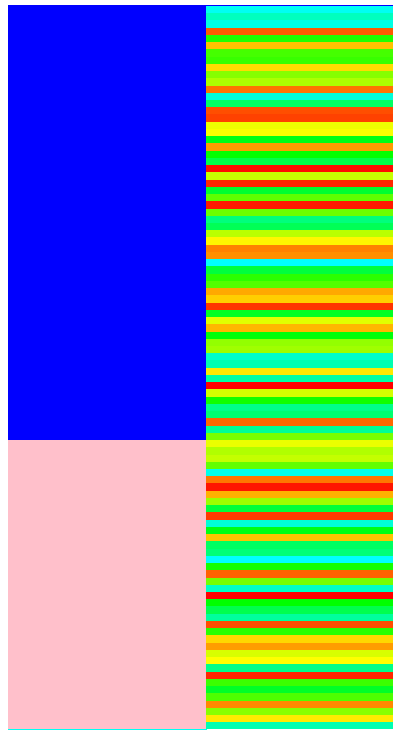
Data



gender

outcome

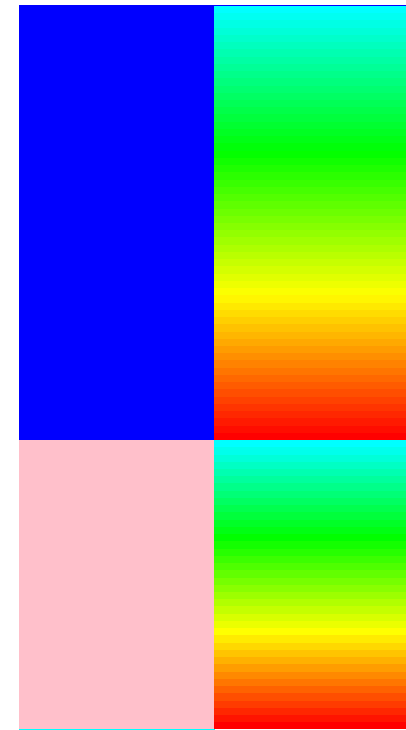
Shuffling outcomes



gender

outcome

Shuffling outcomes (ordered)



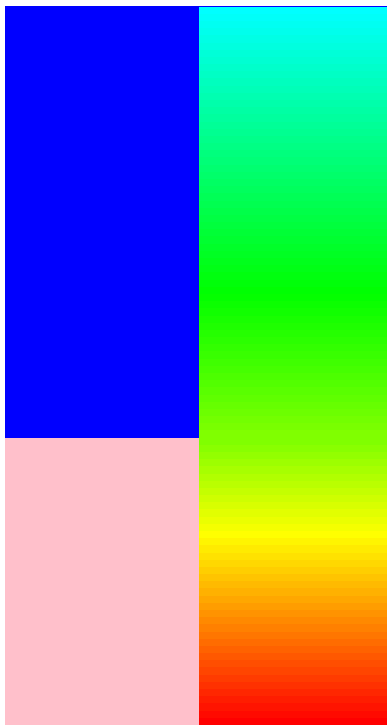
gender

outcome

# Permutation when null is false

---

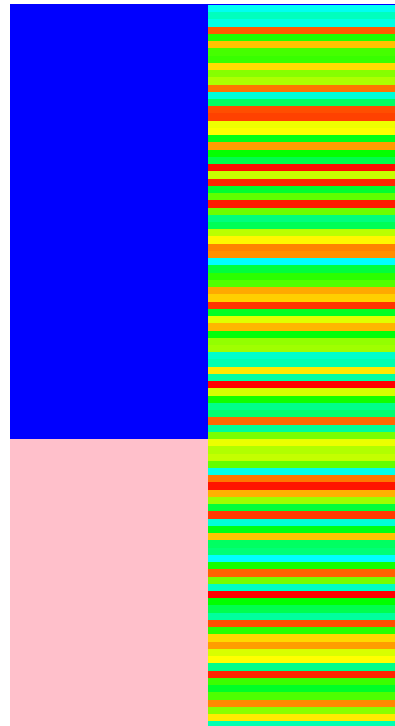
Data



gender

outcome

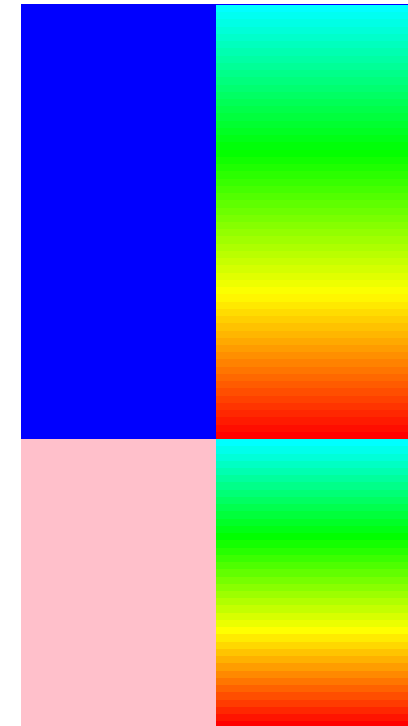
Shuffling outcomes



gender

outcome

Shuffling outcomes (ordered)



gender

outcome



# Example permutation test

---

I am measuring the length of bumblebees between two species with the following data (mm):

- Species A: 4.5, 4.6, 4.1, 5.2
- Species B: 5.1, 4.7, 5.4, 4.8, 4.9

Do the bumblebee species tend to have the same lengths?

Ho: Bumblebee species A and B have the same lengths on average

Ha: Bumblebee species A and B have different lengths on average.

# Some permutations of my data

---

**Species A: 4.5, 4.6, 4.1, 5.2**

**Species B: 5.1, 4.7, 5.4, 4.8, 4.9**

Species A: 4.5, 5.1, 4.1, 5.2

Species B: 4.6, 4.7, 5.4, 4.8, 4.9

Species A: 4.6, 4.7, 4.8, 4.9

Species B: 4.5, 4.1, 5.2, 5.4, 5.1

# Testing the bee data

---

1. Choose my test statistic, here  $t$ , and compute on data
2. Create **a lot** of permuted datasets
  1. Compute  $t$  for each dataset to construct null sampling distribution
3. Compute P-value as probability of being as or more extreme than  $t$  calculated from data

## Performing step 2

---

1. Generate your permuted data with `modelr::permute()`
2. Extract your permutations with `purrr::map()`
3. See the full permutation output with `broom::tidy()` or `broom::glance()`

# Exciting detour: the broom package

---

broom tidies\* up output from linear models and hypothesis tests

- Vignette: <https://cran.r-project.org/web/packages/broom/vignettes/broom.html>

## Functions:

- `tidy()`
- `glance()`
- `augment()`

\*groan.

# Using broom in hypothesis testing

---

```
> versicolor <- iris %>% filter(Species == "versicolor")
> setosa <- iris %>% filter(Species == "setosa")

> my.test <- t.test(versicolor$Sepal.Length, setosa$Sepal.Length)

> tidy(my.test)
```

	estimate	estimate1	estimate2	statistic	p.value	parameter	conf.low
1	0.93	5.936	5.006	10.52099	3.746743e-17	86.538	0.7542926
	conf.high			method	alternative		
1	1.105707	Welch	Two Sample t-test	two.sided			

# Using broom in hypothesis testing while piping

---

```
> iris %>%  
  filter(Species != "virginica") %>%  
  do(tidy(t.test( Sepal.Length~Species, data=. )))
```

	estimate	estimate1	estimate2	statistic	p.value	parameter	conf.low
1	-0.93	5.006	5.936	-10.52099	3.746743e-17	86.538	-1.105707
	conf.high			method	alternative		
1	-0.7542926			Welch Two Sample t-test	two.sided		

# Step One: compute $t$ for my data

---

```
> bees <- tibble(species=c(rep("a", 4), rep("b", 5)),  
                  lengths=c(4.5, 4.6, 4.1, 5.2, 5.1, 4.7, 5.4, 4.8, 4.9))
```

```
> t.test(lengths ~species, data=bees)  
Welch Two Sample t-test
```

data: lengths by species

$t = -1.4673$ ,  $df = 4.7391$ ,  $p\text{-value} = 0.2053$

alternative hypothesis: true difference in means is not equal to 0

95 percent confidence interval:

-1.0568836 0.2968836

sample estimates:

mean in group a	mean in group b
4.60	4.98



# Step Two: generate permuted data

---

```
library(tidyverse)
library(broom)
library(modelr)

### Make 10000 permutations with permute(), from modelr
> set.seed(567)
> N <- 1e4
> bee.perms <- permute(bees, N, lengths) ## dataframe number perms, column to permute
> head(bee.perms)
# A tibble: 10,000 x 2
      perm      .id
  <list> <chr>
1 <S3: permutation> 00001
2 <S3: permutation> 00002
3 <S3: permutation> 00003
4 <S3: permutation> 00004
5 <S3: permutation> 00005
6 <S3: permutation> 00006
```

# Step Two: Compute $t$ for each permuted dataset

---

```
> bee.ttests <- map(bee.perms$perm, ~t.test(lengths~species, data=..))
> head(bee.ttests)
[[1]]
```

Welch Two Sample t-test

```
data: lengths by species
t = -1.5636, df = 6.6843, p-value = 0.1639
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9602378  0.2002378
sample estimates:
mean in group a mean in group b
      4.60      4.98
```

```
[[2]]
```

Welch Two Sample t-test

```
data: lengths by species
t = -1.8074, df = 6.3713, p-value = 0.1178
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -0.9923401  0.1423401
sample estimates:
mean in group a mean in group b
      4.575      5.000
```

# Step Two: Compute $t$ for each permuted dataset

```
> bee.ttests <- map(bee.perms$perm, ~t.test(lengths~species, data=..))  
> tidied.bees <- map_df(bee.ttests, tidy, .id = "id")  
> head(tidied.bees)
```

	id	estimate	estimate1	estimate2	statistic	p.value	parameter	conf.low
1	1	-0.380	4.600	4.98	-1.5635521	0.1639062	6.684311	-0.9602378
2	2	-0.425	4.575	5.00	-1.8074200	0.1178304	6.371305	-0.9923401
3	3	0.160	4.900	4.74	0.6064784	0.5637055	6.865079	-0.4663247
4	4	-0.065	4.775	4.84	-0.2156013	0.8386887	4.518027	-0.8655244
5	5	-0.245	4.675	4.92	-0.8733769	0.4214981	5.123589	-0.9609000
6	6	-0.335	4.625	4.96	-1.3358210	0.2245809	6.799964	-0.9315602

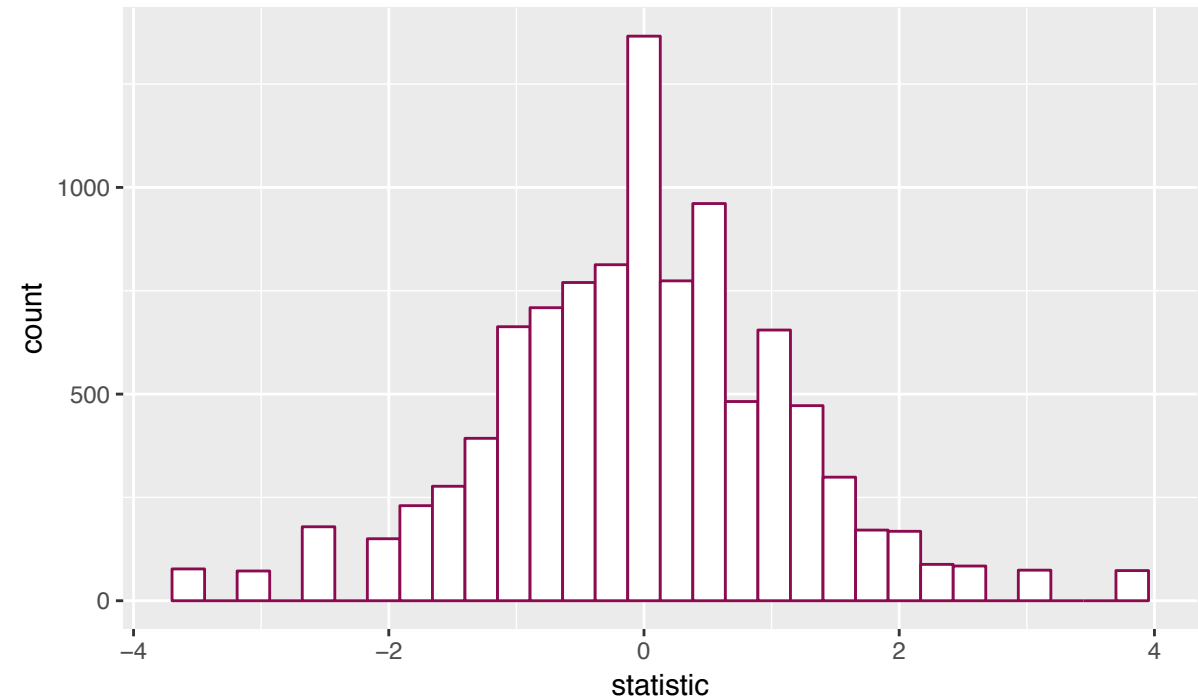
  

	conf.high	method	alternative
1	0.2002378	Welch Two Sample t-test	two.sided
2	0.1423401	Welch Two Sample t-test	two.sided
3	0.7863247	Welch Two Sample t-test	two.sided
4	0.7355244	Welch Two Sample t-test	two.sided
5	0.4709000	Welch Two Sample t-test	two.sided
6	0.2615602	Welch Two Sample t-test	two.sided

# Visualize the null constructed with permutation

---

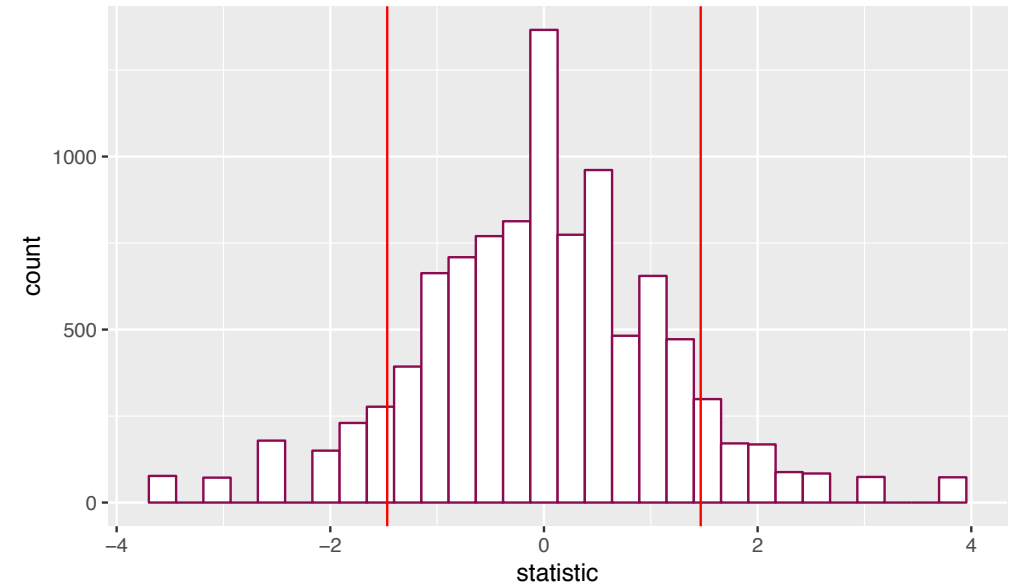
```
ggplot(tidied.bees, aes(x = statistic)) +  
  geom_histogram(fill="white", color="deeppink4")
```



# Visualize the null constructed with permutation

---

```
ggplot(tidied.bees, aes(x = statistic)) +  
  geom_histogram(fill="white", color="deeppink4") +  
  geom_vline(xintercept = -1.4673, color="red") +  
  geom_vline(xintercept = 1.4673, color="red")
```



# Step Three: calculate our Pvalue

---

```
> t.from.data <- -1.4673  
  
> sum(tidied.bees$statistic >= abs(t.from.data)) -> upper.tail  
  
> sum(tidied.bees$statistic <= -1*abs(t.from.data)) -> lower.tail  
  
> (upper.tail + lower.tail) / nrow(tidied.bees)  
[1] 0.1942
```

**With  $P=0.1942$ , we fail to reject the null hypothesis. We have no evidence that bumblebee species a and b have different lengths.**

Pop Quiz: P-values for a permutation test can never be 0. Why?

# Note on computations shown here

---

```
bee.ttests <- map(bee.perms$perm, ~t.test(lengths~species, data=.))
```

```
bee.results <- map(bee.perms$perm, ~<only certain functions> )
```

# Note the R package coin...

---

Pre-dated the tidyverse but **very** convenient for simple tests

```
library(coin)
### Permutation test for two arbitrary samples
independence_test(b ~ a, data)
```

```
independence_test(lengths ~ species, data = bees)
```

Asymptotic General Independence Test

```
data: lengths by species (a, b)
Z = -1.4337, p-value = 0.1517
alternative hypothesis: two.sided
```



# coin for contingency tables

---

```
> sparrows %>% group_by(Sex, Survival) %>% tally() -> sex.survival
```

	Sex	Survival	n
1	Female	Alive	21
2	Female	Dead	28
3	Male	Alive	51
4	Male	Dead	36

```
> xtabs(n ~ Sex + Survival, data = sex.survival) -> sex.survival.table
```

	Survival	
Sex	Alive	Dead
Female	21	28
Male	51	36

```
> independence_test(sex.survival.table)
```

Asymptotic General Independence Test

data: Survival by Sex (Female, Male)

Z = -1.7617, p-value = 0.07813

alternative hypothesis: two.sided

# Exercise break

---

# Bootstrapping

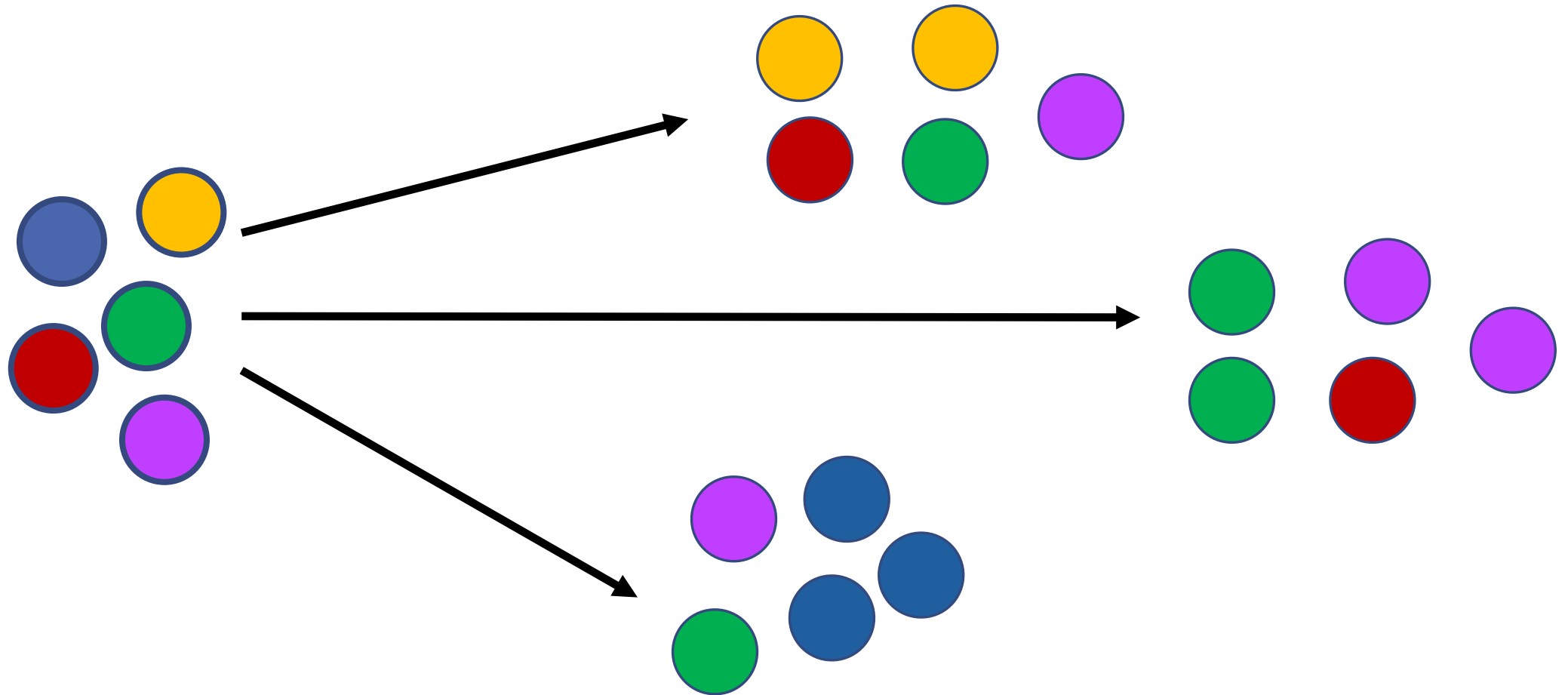
---

Bootstrapping uses resampling from the data with replacement to approximate the sampling distribution of an estimate

Use bootstrapping to calculate CI and standard error

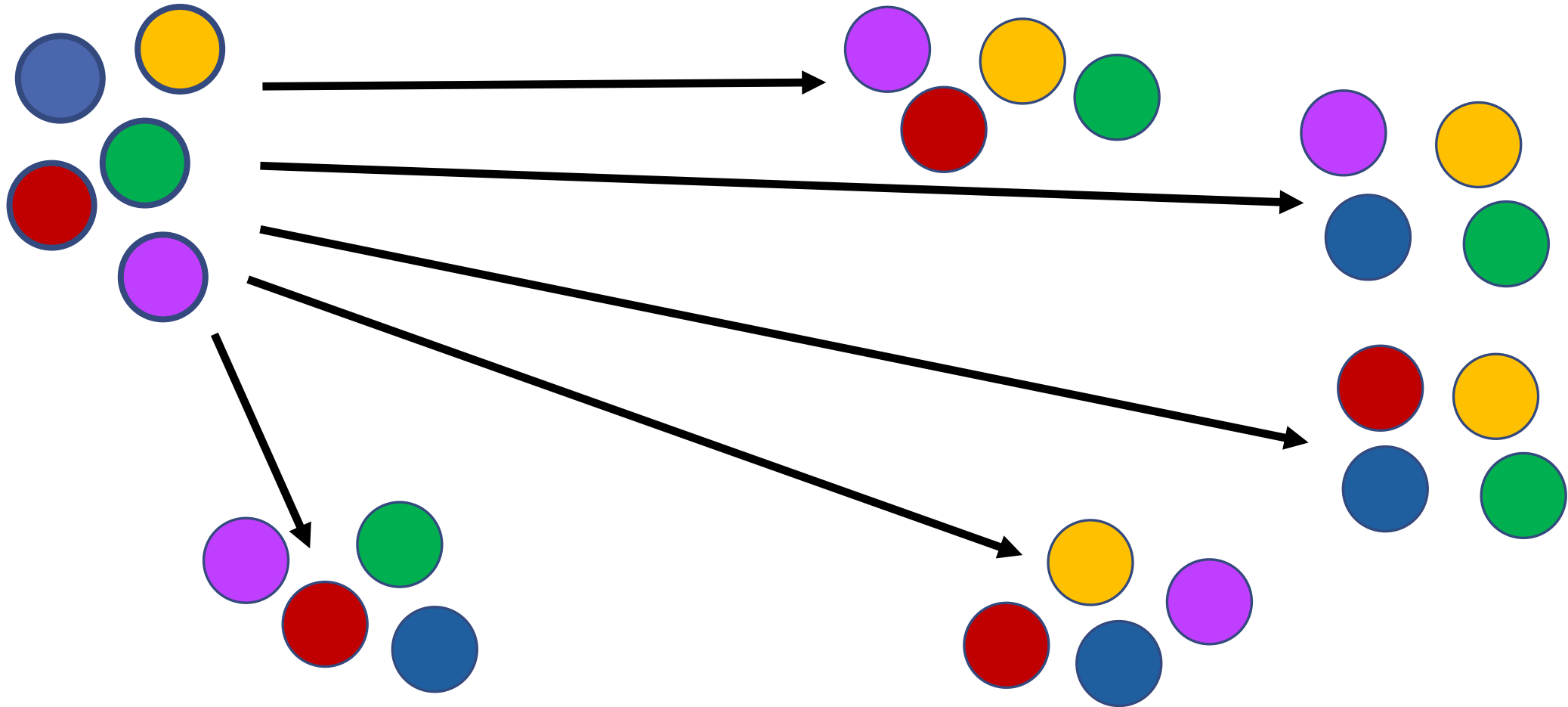
# What is bootstrapping?

---



As aside, this is jackknifing

---



# WA acidic rain estimation, the "regular way"

---

```
> rain <- tibble(pH = c(4.73, 5.28, 5.06, 5.16, 5.25, 5.11, 4.79))
```

```
> mean(rain$pH)
[1] 5.054286
```

```
## 95% CI as estimate +/- t_0.025*SE
> se <- sd(rain$pH)/sqrt(nrow(rain))
> df <- nrow(rain) - 1
> qt(0.025,df) * se
[1] -0.1992792
```

Estimate for WA rain acidity is 5.05 +/- 0.199

**BUT our assumptions for this approach were not met, so we should use the bootstrap**

# Using the bootstrap to estimate rain pH

---

```
> set.seed(199)
> devtools::install_github('jtleek/slipper')

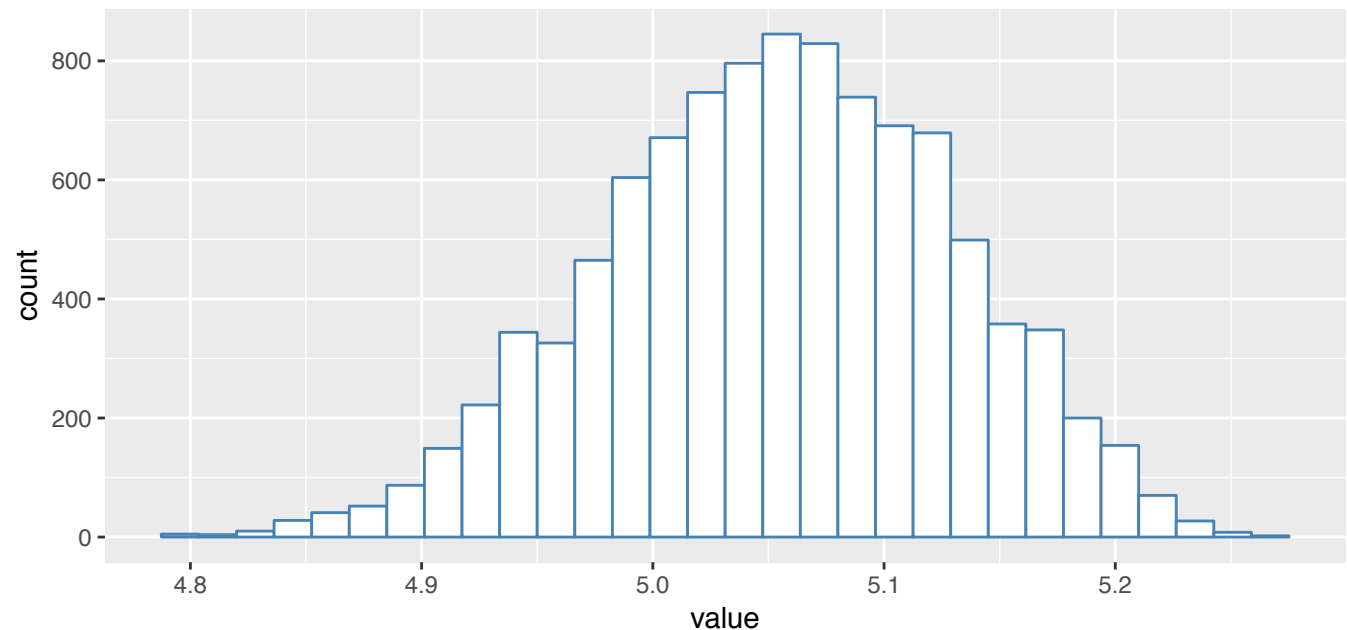
> rain.boot <- slipper(rain, mean(pH), B=10000)
> head(rain.boot)
      type    value
1 observed 5.054286
2 bootstrap 5.090000
3 bootstrap 5.115714
4 bootstrap 4.984286
5 bootstrap 5.015714
6 bootstrap 5.164286

> rain.boot %>% filter(type == "bootstrap") %>% summarize(mean(value))
  mean(value)
1    5.054452
```

# Visualizing the bootstrap sampling distribution of the mean

---

```
> rain.boot %>%  
  filter(type == "bootstrap") %>%  
  ggplot(aes(x = value)) + geom_histogram(fill = "white", color = "steelblue")
```

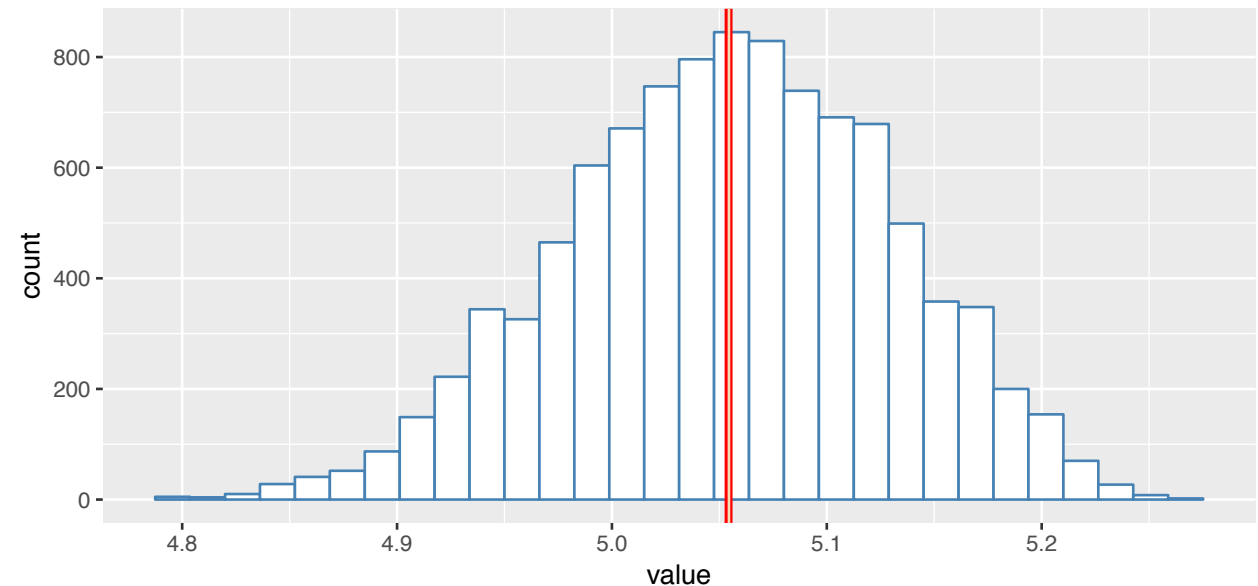




# Visualizing the bootstrap sampling distribution of the mean

---

```
> rain.boot %>%  
  filter(type == "bootstrap") %>%  
  ggplot(aes(x = value)) + geom_histogram(fill = "white", color = "steelblue") +  
  geom_vline(xintercept=5.054286, color="red", size=1.5) +  
  geom_vline(xintercept=5.054452, color="wheat")
```



# All together, computing the bootstrap estimates

---

```
rain %>%
  slipper(mean(pH), B=10000) %>%
  filter(type=="bootstrap") %>%
  summarize(mean = mean(value),
            ci_low = quantile(value, 0.025),
            ci_high = quantile(value, 0.975))

  mean    ci_low  ci_high
1 5.053425 4.894286 5.192857
```

**Our bootstrap estimate: pH has a mean of 5.053 with a 95% CI of [4.89, 5.19]**

"Regular" bootstrap was 5.05 with 95% CI of [4.85, 5.24]

# What might we want to estimate from the bee data?

---

Estimate for *difference of means*

Confidence interval for *difference of means*

# Generate bootstraps when data frame has multiple samples

---

```
> bees %>% filter(species == "a") -> bees.a
> bees.a %>%
  slipper(mean(lengths), B=1e4) %>%
  mutate(species = "a") -> a.boot
  type value species
1 observed 4.600      a
2 bootstrap 4.575      a
3 bootstrap 4.200      a
4 bootstrap 4.650      a
...
```

```
> bees %>% filter(species == "b") -> bees.b
> bees.b %>%
  slipper(mean(lengths), B=1e4) %>%
  mutate(species = "b") -> b.boot
  type value species
1 observed 4.98      b
2 bootstrap 5.10      b
3 bootstrap 4.84      b
4 bootstrap 4.88      b
...
```

# con't

---

```
full.boot <- rbind(a.boot, b.boot)
```

```
head(full.boot)
```

	type	value	species
1	observed	4.600	a
2	bootstrap	4.750	a
3	bootstrap	4.450	a

```
full.boot %>%
```

```
  filter(type == "bootstrap") %>%
```

```
  group_by(species) %>%      ### Add unique ID per group line
```

```
  mutate(n = 1:n()) %>%    ### ""
```

```
  spread(species, value)
```

	type	n	a	b
	<fctr>	<int>	<dbl>	<dbl>
1	bootstrap	1	4.750	5.10
2	bootstrap	2	4.450	4.84
3	bootstrap	3	4.500	4.88

# con't

---

```
full.boot %>%
  filter(type == "bootstrap") %>%
  group_by(species) %>%      ### Add unique ID per group line
  mutate(n = 1:n()) %>%     ### ""
  spread(species, value) %>%
  mutate(value = a - b) %>%
  summarize(mean = mean(value),
            ci_low = quantile(value,0.025),
            ci_high = quantile(value,0.975))

# A tibble: 1 x 3
   mean ci_low ci_high
  <dbl> <dbl>   <dbl>
1 -0.3816805 -0.815    0.07
```

**The bootstrap difference in mean bee lengths is -0.382 with a 95% bootstrap CI of [-0.815, 0.07]**

**Pop quiz! Recall our permutation test gave P=0.19. Is the bootstrap consistent?**

# Exercise break

---

# Multiple testing

---

<https://xkcd.com/882/>

Run tests until you get a significant result = **no.**

- Data dredging
- P-hacking
- Data fishing
- Data snooping



# False positive rate increases with more tests

---

## Single test

$$P(\text{false positive}) = \alpha$$

$$P(\text{not false positive}) = 1 - \alpha$$

## N tests

$$P(\text{no false positives}) = (1 - \alpha)^N$$

$$P(\text{at least 1 false positive}) = 1 - (1 - \alpha)^N$$

For  $N=20$  and  $\alpha=0.05$ ,  $P(\text{at least 1 false positive}) = 65\%$

For  $N=100$  and  $\alpha=0.05$ ,  $P(\text{at least 1 false positive}) = 99\%$

# Example: Many tests on iris

---

```
versicolor <- iris %>% filter(Species == "versicolor")
virginica<- iris %>% filter(Species == "virginica")
setosa <- iris %>% filter(Species == "setosa")

> t.test(versicolor$Sepal.Length, virginica$Sepal.Length)$p.value
[1] 1.866144e-07
> t.test(versicolor$Sepal.Length, setosa$Sepal.Length)$p.value
[1] 3.746743e-17
> t.test(virginica$Sepal.Length, setosa$Sepal.Length)$p.value
[1] 3.966867e-25

> t.test(versicolor$Sepal.Width, virginica$Sepal.Width)$p.value
[1] 0.001819483
> t.test(versicolor$Sepal.Width, setosa$Sepal.Width)$p.value
[1] 2.484228e-15
> t.test(virginica$Sepal.Width, setosa$Sepal.Width)$p.value
[1] 4.570771e-09
```

```
> t.test(versicolor$Petal.Length, virginica$Petal.Length)$p.value
[1] 4.900288e-22
> t.test(versicolor$Petal.Length, setosa$Petal.Length)$p.value
[1] 9.934433e-46
> t.test(virginica$Petal.Length, setosa$Petal.Length)$p.value
[1] 9.269628e-50

> t.test(versicolor$Petal.Width, virginica$Petal.Width)$p.value
[1] 2.111534e-25
> t.test(versicolor$Petal.Width, setosa$Petal.Width)$p.value
[1] 2.717008e-47
> t.test(virginica$Petal.Width, setosa$Petal.Width)$p.value
[1] 2.437136e-48
```

# Two broad types of error rates

---

## **Family-wise error rate (FWER)**

- Probability of having at least one false positive among all tests
- Probability of rejecting at least one true null
- Corrections control  $\text{FWER} \leq \alpha$

## **False discovery rate (FDR)**

- Expected proportion of false positives among rejected hypotheses
- Rate that false discoveries occur
- $\text{FP} / (\text{FP} + \text{TP})$
- Corrections control  $\text{FDR} \leq \alpha$

# Controlling the FWER

---

**Bonferroni** correction is the most common

- For  $m$  tests, change  $\alpha \rightarrow \alpha/m$  and assess significance

My 6 tests gave  $P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

- $0.05 \rightarrow 0.05/6 \rightarrow$  My new  $\alpha$  is 0.0083
- $P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

Can alternatively multiply P-values by  $m$  and use original  $\alpha$  (0.05)

- raw  $P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$
- **corrected**  $P = \{0.06, 0.18, 0.024, 0.162, 0.0036, \text{2.88 1.0}\}$

# The Holm (BH) method uses a *stepwise* procedure to control FWER

---

Uses a *stepwise* (step-down) procedure to control FWER

1. Order P-values from  $m$  tests from smallest to largest and rank  $k$
2. For a given  $\alpha$ , find the **smallest**  $k$  where  $P_k > \alpha/(m+1-k)$
3. Reject  $H_0$  for all  $P_1 \dots P_{k-1}$ 
  1. Fail to reject  $H_0$  for all  $P_k \dots P_m$

# Holm (Bonferroni-Holm) FWER correction


$P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

Raw P	$\alpha/(m-k+1)$	
0.0006	$0.05/(6-1+1) = 0.0083$	P is smaller, continue
0.004	$0.05/(6-2+1) = 0.01$	P is smaller, continue
0.01	$0.05/(6-3+1) = 0.0125$	P is smaller, continue
0.027	$0.05/(6-4+1) = 0.0125$	P is greater, <b>STOP</b>
0.03		
0.048		

$P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

# Holm FWER correction with corrected P

$$P_{\text{corrected}} = m * p_k, k=1$$
$$= \max[ p_{(k-1)}, p_k * (m-k+1) ], \quad k>1$$



Raw P	Holm-corrected P
0.0006	0.0006 * 6 = <b>0.0036</b>
0.004	0.004 * (6-2+1) = 0.02 max(0.0036, 0.02) = <b>0.02</b>
0.01	0.01 * (6-3+1) = 0.04 max(0.02, 0.04) = <b>0.04</b>
0.027	0.027 * (6-4+1) = 0.081 max(0.04, 0.081) = <b>0.081</b>
0.03	0.03 * (6-5+1) = 0.06 max(0.081, 0.06) = <b>0.081</b>
0.048	0.048 * (6-6+1) = 0.048 max(0.081, 0.048) = <b>0.081</b>

# Controlling error with FDR

---

FWER control procedures are **highly conservative**

- Bonferroni is the most severe
- Guarantees low false positive rate at the cost of a high false negative rate

FDR control procedures are much more powerful

- You end up with more significant results, at the cost of a higher false positive rate
- Invented as a response to the severe conservatism of Bonferroni, etc.



# Controlling FDR with Benjamini-Hochberg ("BH")

---

This is also a stepwise ("step-up") procedure

1. Determine the FDR you are willing to live with with,  $Q$
2. Order P-values from  $m$  tests from smallest to largest and rank
3. For a given  $Q$ , find the **largest**  $k$  where  $P_k \leq Q * k/m$
4. Reject all  $H_0$  for  $P_1 \dots P_k$ 
  1. Fail to reject  $H_0$  for  $P_{k+1} \dots P_m$

# BH FDR procedure

---

$P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

Raw P	$\alpha * k/m$
0.0006	
0.004	
0.027	
0.03	
0.01	
0.048	$0.05 * 6/6 = 0.05$

**P** is smaller, **STOP**

$P = \{0.01, 0.03, 0.004, 0.027, 0.0006, 0.048\}$

# FDR correction with corrected P

$$P_{\text{corrected}} = p_k, k=m$$
$$= \min[ p_{(k+1)}, p_k^*(m/k) ], \quad k < m$$

Raw P	FDR-corrected P
0.0006	$0.0006 * 6/1 = 0.0036$ $\min(0.012, 0.0036) = \mathbf{0.0036}$
0.004	$0.004 * 6/2 = 0.012$ $\min(0.02, 0.012) = \mathbf{0.012}$
0.01	$0.01 * 6/3 = 0.02$ $\min(0.036, 0.02) = \mathbf{0.02}$
0.027	$0.027 * 6/4 = 0.0405$ $\min(0.036, 0.0405) = \mathbf{0.036}$
0.03	$0.03 * 6/5 = 0.036$ $\min(0.048, 0.036) = \mathbf{0.036}$
0.048	<b>0.048</b>

# R makes it super easy

---

```
p.values <- c(0.0006, 0.004, 0.01, 0.027, 0.03, 0.048)
```

```
### Holm is default
```

```
p.adjust(p.values)
```

```
[1] 0.0036 0.0200 0.0400 0.0810 0.0810 0.0810
```

```
### Specify Bonferroni
```

```
p.adjust(p.values, method = "bonferroni")
```

```
[1] 0.0036 0.0240 0.0600 0.1620 0.1800 0.2880
```

```
### Specify as bonf for when you can't remember if 2 r's or 2 n's
```

```
p.adjust(p.values, method = "bonf")
```

```
[1] 0.0036 0.0240 0.0600 0.1620 0.1800 0.2880
```

```
### Specify FDR
```

```
p.adjust(p.values, method = "bonf")
```

```
[1] 0.0036 0.0120 0.0200 0.0360 0.0360 0.0480
```

# Use sum() to determine number of significant results

---

```
p.values <- c(0.0006, 0.004, 0.01, 0.027, 0.03, 0.048)
alpha <- 0.05
```

```
### Holm
sum( p.adjust(p.values) <= alpha )
[1] 3
```

```
### Bonferroni
sum( p.adjust(p.values, method = "bonf") <= alpha)
[1] 2
```

```
### FDR
sum( p.adjust(p.values, method = "fdr") <= alpha)
[1] 6
```

# Running multiple tests in R

---

```
pairwise.t.test(response, grouping)
```

```
pairwise.wilcox.test(response, grouping)
```

```
> pairwise.t.test(iris$Sepal.Width, iris$Species)
```

Pairwise comparisons using t tests with pooled SD

data: iris\$Sepal.Width and iris\$Species

	setosa	versicolor
versicolor	< 2e-16	-
virginica	9.1e-10	0.0031

P value adjustment method: holm

# Specify correction with "p.adj"

---

```
> pairwise.t.test(iris$Sepal.Width, iris$Species, p.adj = "fdr" )
```

Pairwise comparisons using t tests with pooled SD

data: iris\$Sepal.Width and iris\$Species

	setosa	versicolor
versicolor	< 2e-16	-
virginica	6.8e-10	0.0031

P value adjustment method: fdr

# The triumphant return of broom!

---

```
> pairwise.t.test(iris$Sepal.Width, iris$Species, p.adj = "fdr") %>% tidy()
```

	group1	group2	p.value
1	versicolor	setosa	5.497468e-17
2	virginica	setosa	6.808435e-10
4	virginica	versicolor	3.145180e-03

### How many are significant at 0.05?

```
pairwise.t.test(iris$Sepal.Width, iris$Species, p.adj = "fdr") %>%  
  tidy() %>%  
  mutate(sig = p.value <= 0.05) %>%  
  group_by(sig) %>% tally()  
  sig      n  
<lgl> <int>  
1  TRUE      3
```



# The thrilling age of GWAS

## SCIENTIFIC REPORTS

OPEN

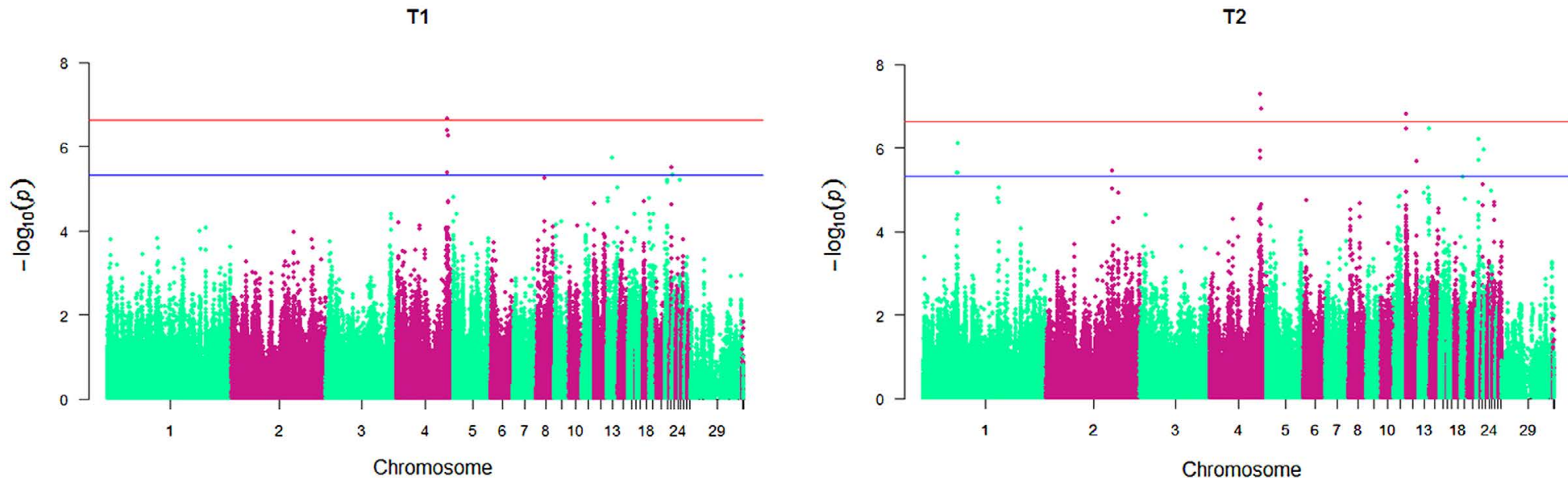
### Genome-wide association study of aggressive behaviour in chicken

Zhenhui Li<sup>1,2</sup>, Ming Zheng<sup>1,2</sup>, Bahareldin Ali Abdalla<sup>1,2</sup>, Zhe Zhang<sup>1,2</sup>, Zhenqiang Xu<sup>1,2,3</sup>, Qiao Ye<sup>1,2</sup>, Haiping Xu<sup>1,2</sup>, Wei Luo<sup>1,2</sup>, Qinghua Nie<sup>1,2</sup> & Xiquan Zhang<sup>1,2</sup>

Abbreviation	Phenotype Description	Classification	Mean $\pm$ SD
T1	Number of fighting times during the whole recording period (16 days)	Fighting times	14.69 $\pm$ 11.24
T2	Number of fighting times in days with frequencies not less than 4 times per day		4.64 $\pm$ 8.63
T3	Number of days for chicken showed fighting	Fighting days	7.28 $\pm$ 3.53
T4	Number of days for chicken showed fighting with frequencies not less than 4 times per day		0.89 $\pm$ 3.53

Table 1. Four aggressive-behaviour phenotypes measured traits in male chickens.

# The age of GWAS



**Figure 1. Manhattan plots of genome-wide association study on chicken aggressive-behaviour measured traits from T1 to T4 for all the SNPs.** The associated values (in terms of  $-\log_{10}P$ ) are shown by chromosomes. The blue highlighted line indicates genome-wide association ( $P = 4.6E-6$ ), and the red highlighted line indicates significance with a  $P$ -value threshold of the 5% Bonferroni genome-wide significance ( $P = 2.3E-7$ ).

# Exercise break

---