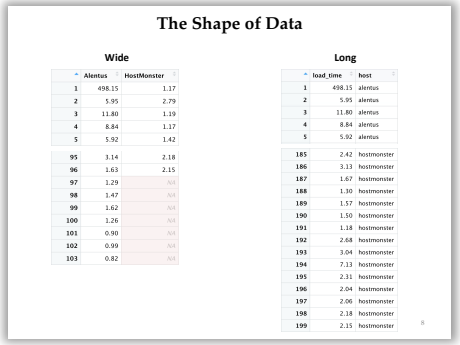
Business Analytics Using
Computational Statistics

Week 5
Bootstrapped Tests

Week 6
Permutation Tests

Week 7
Comparing Groups of Data

Reshaping Data



The Shape of Data

Wide

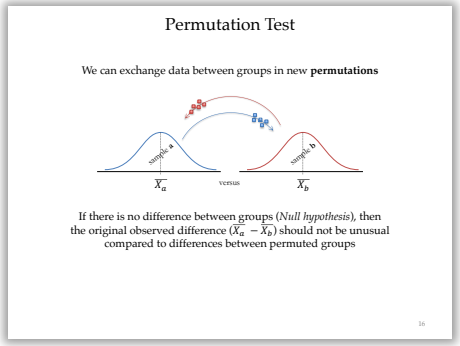
	Alentus	HostMonster
1	498.15	1.17
2	5.95	2.79
3	11.80	1.19
4	8.84	1.17
5	5.92	1.42
95	3.14	2.18
96	1.63	2.15
97	1.29	NA
98	1.47	NA
99	1.62	NA
100	1.26	NA
101	0.90	NA
102	0.99	NA
103	0.82	NA

Long

	load_time	host
1	498.15	alentus
2	5.95	alentus
3	11.80	alentus
4	8.84	alentus
5	5.92	alentus
185	2.42	hostmonster
186	3.13	hostmonster
187	1.67	hostmonster
188	1.30	hostmonster
189	1.57	hostmonster
190	1.50	hostmonster
191	1.18	hostmonster
192	2.68	hostmonster
193	3.04	hostmonster
194	7.13	hostmonster
195	2.31	hostmonster
196	2.04	hostmonster
197	2.06	hostmonster
198	2.18	hostmonster
199	2.15	hostmonster

8

Permutation Test



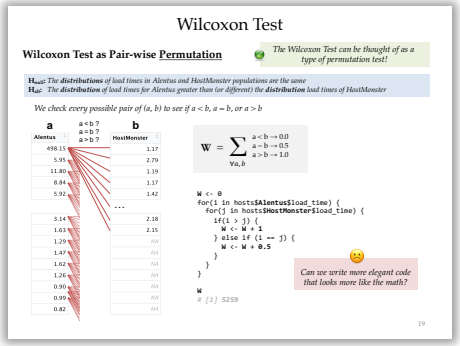
Permutation Test

We can exchange data between groups in new **permutations**

If there is no difference between groups (*Null hypothesis*), then the original observed difference ($\overline{X_a} - \overline{X_b}$) should not be unusual compared to differences between permuted groups

16

Wilcoxon Test



Wilcoxon Test

Wilcoxon Test as Pair-wise Permutation

The Wilcoxon Test can be thought of as a type of permutation test!

H_{null}: *The* ***distributions*** *of load times in Alentus and HostMonster populations are the same*
H_{alt}: *The* ***distribution*** *of load times for Alentus greater than (or different) the* ***distribution*** *load times of HostMonster*

We check every possible pair of (a, b) to see if a < b, a = b, or a > b

a	a < b ? a = b ? a > b ?	b
Alentus		HostMonster
498.15		1.17
5.95		2.79
11.80		1.19
8.84		1.17
5.92		1.42
	...	
3.14		2.18
1.63		2.15
1.29		NA
1.47		NA
1.62		NA
1.26		NA
0.90		NA
0.99		NA
0.82		NA

$$W = \sum_{\forall a,b} \begin{cases} a < b \rightarrow 0.0 \\ a = b \rightarrow 0.5 \\ a > b \rightarrow 1.0 \end{cases}$$

```
W <- 0
for(i in hosts$Alentus$load_time) {
  for(j in hosts$HostMonster$load_time) {
    if(i > j) {
      W <- W + 1
    } else if (i == j) {
      W <- W + 0.5
    }
  }
}
W
# [1] 5259
```

Can we write more elegant code that looks more like the math?

19

Review of Error

Mean usage of **new** device

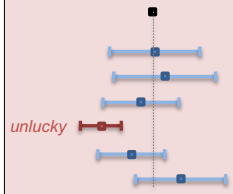
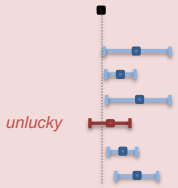
Mean usage of **previous** device

$$t = \frac{(\bar{x} - \mu_o)}{s/\sqrt{n}}$$

Your colleague, a data analyst in your organization, is working on a hypothesis test where he has sampled product usage information from customers who are using a new smartwatch. He wishes to test whether the mean (x_i) usage time is higher than the usage time of the company's previous smartwatch released two years ago (μ_o):

> H_{null}: The mean usage time of the new smartwatch is the same or less than for the previous smartwatch.
>
> H_{alt}: The mean usage time is greater than that of our previous smartwatch.

After collecting data from just n=50 customers, he informs you that he has found diff=0.3 and sd=2.9. Your colleague believes that we cannot reject the null hypothesis at alpha of 5%.

	If H_{null} is really *True*	If H_{null} is really *False*
If evidence says *reject H_{null}*	**Type I Error** Probability: α *"Significance Level"* 	**Correct!** Probability: (1-β) *"Power of the Test"*
If evidence says *cannot reject H_{null}*	**Correct!** Probability: 1-α *"Confidence Level"*	**Type II Error** Probability: β 

i. Would this scenario create systematic or random error (or both or neither)?
ii. Which part of the t-statistic or significance (diff, sd, n, alpha) would be affected?
iii. Will it increase or decrease our *power* to reject the null hypothesis?
iv. Which kind of error (Type I or Type II) becomes more likely because of this scenario?

a. You discover that your colleague wanted to target the general population of Taiwanese users of the product. However, he only collected data from a pool of young consumers, and missed many older customers who you suspect might use the product *much less* every day. → Type II
b. You find that 20 of the respondents are reporting data from the wrong wearable device, so they should be removed from the data. These 20 people are just like the others in every other respect. → Type I
c. A very annoying professor visiting your company has criticized your colleague's "95% confidence" criteria, and has suggested relaxing it to just 90%. → Type I
d. Your colleague has measured usage times on five weekdays and taken a daily average. But you feel this will underreport usage for younger people who are very active on weekends, whereas it over-reports usage of older users. → Type II

Review of Bootstrapping

```
hyp_mean <- 7.6
sample_mean <- mean(times)
observed_diff <- sample_mean - hyp_mean
sample_n <- length(times)
sample_sd <- sd(times)
```

Parametric t-test and p-value:

```
t_test <- t.test(times, mu=hyp_mean, alternative = "greater", conf.level = 0.99)
t_value <- t_test$statistic
# t = 2.5608, df = 1686, p-value = 0.005265
```

Parametric power:

```
power.t.test(n=sample_n, delta = observed_diff, sd = sample_sd, sig.level = 0.01,
             alternative = "one.sided", type = "one.sample")

#          power = 0.5918705
#    alternative = one.sided
```

Bootstrapping the Null and Alt distributions

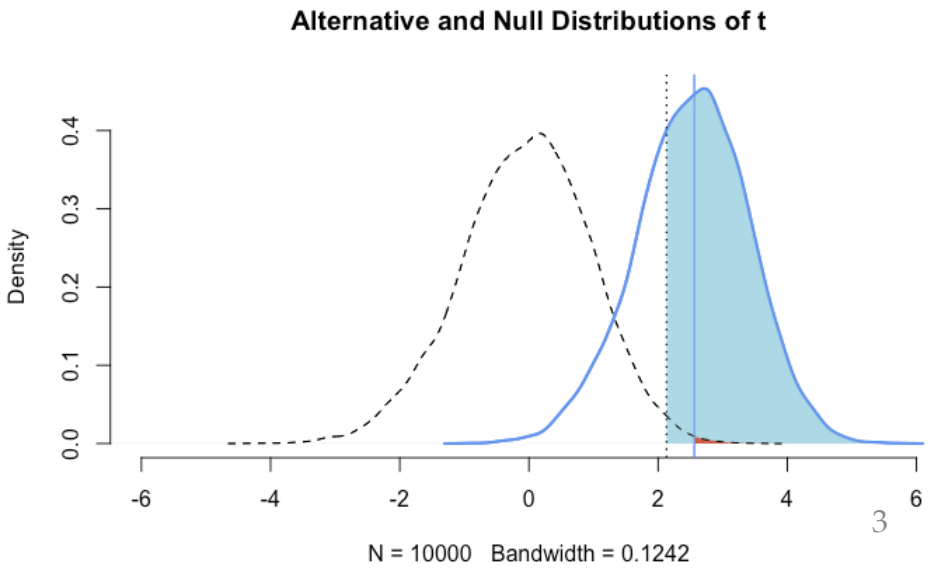
```
set.seed(439387348)
boot_t_stats <- replicate(10000, bootstrap_null_alt(times, hyp_mean))

t_alt <- boot_t_stats[1,]
t_null <- boot_t_stats[2,]
cutoff_99 <- quantile(t_null, probs=0.99)
[1] 0.6834
```

Bootstrapped p-value

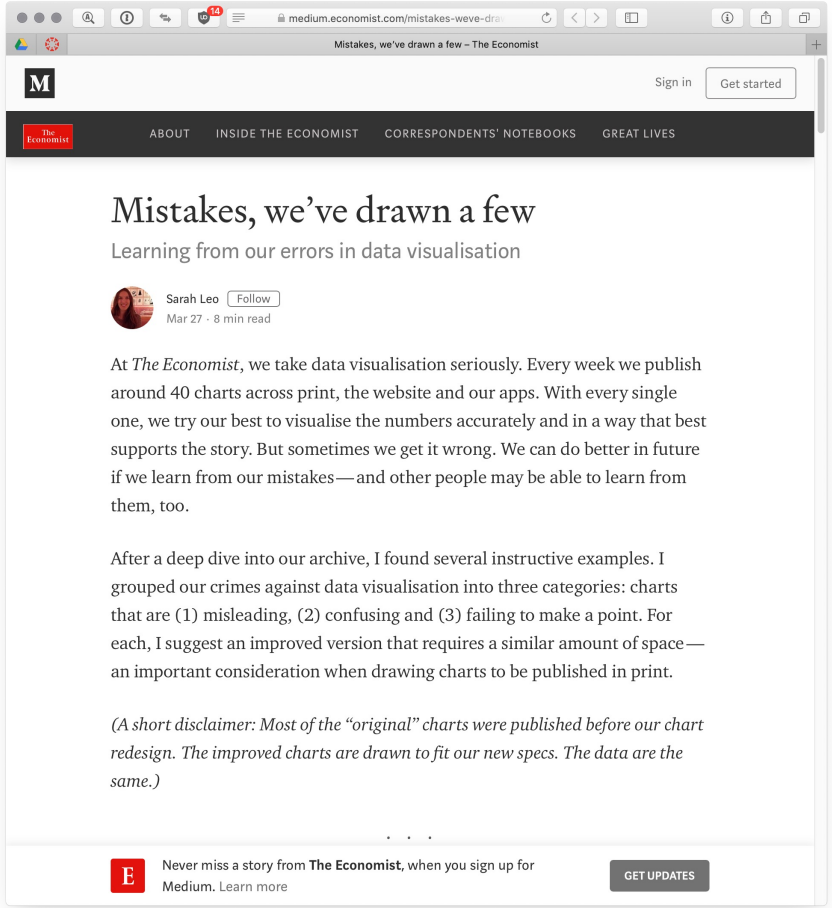
```
null_probs <- ecdf(t_null)
one_tailed_pvalue <- 1 - null_probs(t_value)
[1] 0.0022
```

Bootstrapped power

```
alt_probs <- ecdf(t_alt)
t_power <- 1 - alt_probs(cutoff_99)
[1] 0.6834
```



Data Visualization



Mistakes, we've drawn a few – The Economist

Sign in | Get started

ABOUT | INSIDE THE ECONOMIST | CORRESPONDENTS' NOTEBOOKS | GREAT LIVES

Mistakes, we've drawn a few

Learning from our errors in data visualisation

Sarah Leo Follow
Mar 27 · 8 min read

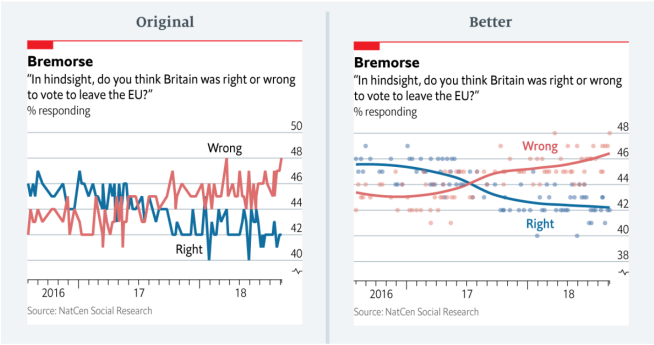
At *The Economist*, we take data visualisation seriously. Every week we publish around 40 charts across print, the website and our apps. With every single one, we try our best to visualise the numbers accurately and in a way that best supports the story. But sometimes we get it wrong. We can do better in future if we learn from our mistakes—and other people may be able to learn from them, too.

After a deep dive into our archive, I found several instructive examples. I grouped our crimes against data visualisation into three categories: charts that are (1) misleading, (2) confusing and (3) failing to make a point. For each, I suggest an improved version that requires a similar amount of space—an important consideration when drawing charts to be published in print.

(A short disclaimer: Most of the "original" charts were published before our chart redesign. The improved charts are drawn to fit our new specs. The data are the same.)

Never miss a story from **The Economist**, when you sign up for Medium. Learn more

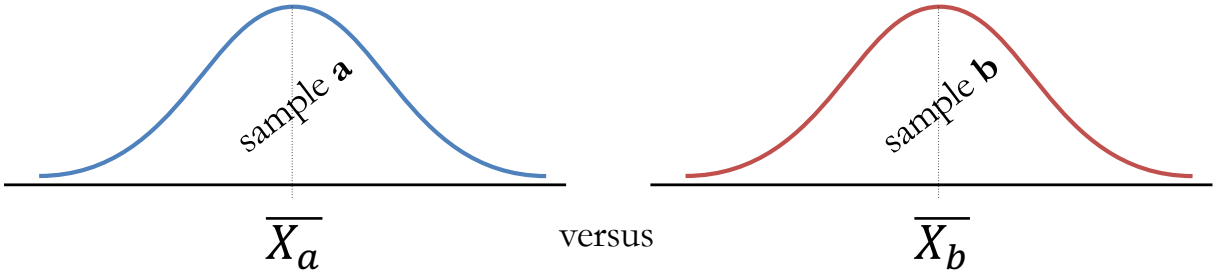
GET UPDATES



Instead of plotting the individual polls with a smoothed curve to show the trend, we connected the actual values of each individual poll. This happened, primarily, because our in-house charting tool does not plot smoothed lines. Until fairly recently, we were less comfortable with statistical software (like R) that allows more sophisticated visualisations. Today, all of us are able to plot a polling chart like the redesigned one above.

Comparing Populations

*Do two populations have the same **mean**?*



*Do two populations have the same **distribution**?*

🤔

When might we ask each these questions?

Are these questions different?

Example: A Frustrated Web Site Owner

"I received an email from an irritated customer which indicated that my website www.SigmaZone.com was very slow. The most obvious solution to a slow site would be to notify my web hosting company, Alentus. I did some preliminary testing and it did appear that my site was not only slower than big sites such as Google and Corel, but was actually much slower than Alentus' home page. I was more than a little irked that the company I was paying for hosting, Alentus, had a much faster load time than my site…

If I chose to do nothing and my site was truly slow, then my customers would suffer. If I chose to do something, but in reality my site was not slow, then I would be expending a lot of effort which wasn't needed. To act on a single data point would be folly. The differences in speed could have been due to my local internet connection, the day or time of day I ran the test (maybe Alentus was doing a server update), or perhaps Alentus was under a Denial of Service attack. I needed more data…

My company happens to have another website with HostMonster which we use for FTP access. So, I loaded three pages that were hosted on Alentus to HostMonster.

I had three different employees collect page load times over a period of 2 weeks…"

load_times

Filter

	Alentus	HostMonster
1	498.15	1.17
2	5.95	2.79
3	11.80	1.19
4	8.84	1.17
5	5.92	1.42
6	7.99	1.60
7	1.80	1.30
8	1.60	1.44
.	.	.
95	3.14	2.18
96	1.63	2.15
97	1.29	NA
98	1.47	NA
99	1.62	NA
100	1.26	NA
101	0.90	NA
102	0.99	NA
103	0.82	NA

<u>Claim 1:</u>
Imagine that Alentus claims that its **average** load time is not worse than its competitor HostMonster

<u>Claim 2:</u>
Imagine that Alentus claims that its **overall** load times are not worse than its competitor HostMonster

source: http://www.sigmazone.com/Aritcle_HypothesisTestBackStory.htm (retrieved March 2019)

Exploring our Data

```
# Load and inspect data
page_loads <- read.csv(file="page_loads.csv")

class(page_loads)
# [1] "data.frame"

View(page_loads)

page_loads$Alentus
```

```
 [1] 498.15   5.95  11.80   8.84   5.92   7.99   1.80   1.60   2.49   1.06   0.75   2.01   3.04   2.04   1.79
 [16]   2.47   2.03   2.28   7.97   6.38 424.01  19.36  21.99   9.15   9.08   3.18   3.12   1.57   1.27   4.06
  .      .      .
 [91]   1.67   1.97   1.70   0.86   3.14   1.63   1.29   1.47   1.62   1.26   0.90   0.99   0.82
```

```
page_loads$HostMonster
```

```
 [1]  1.17  2.79  1.19  1.17  1.42  1.60  1.30  1.44  5.74  1.43  1.36  2.01 11.77  4.24  4.59  4.95  3.21  2.73
 [19]  3.37  5.82  2.62  2.64  2.48  2.64  2.74  3.17  3.94  1.46  2.20  1.97  2.52  3.70  2.38  2.07  2.18  1.31
  .     .     .
 [91]  7.13  2.31  2.04  2.06  2.18  2.15    NA    NA    NA    NA    NA    NA    NA
```

	Alentus	HostMonster
1	498.15	1.17
2	5.95	2.79
3	11.80	1.19
4	8.84	1.17
5	5.92	1.42
95	3.14	2.18
96	1.63	2.15
97	1.29	NA
98	1.47	NA
99	1.62	NA
100	1.26	NA
101	0.90	NA
102	0.99	NA
103	0.82	NA

```
# Describe and visualize data
plot(density(page_loads$Alentus), lwd=2)
plot(density(page_loads$HostMonster))
# Error in density.default(page_loads$HostMonster)
#  'x' contains missing values


mean(page_loads$Alentus)
[1] 20.64718

mean(page_loads$HostMonster)
[1] NA
```

☹️

These problems will keep happening as we analyze our data further

The Shape of Data

Wide

	Alentus	HostMonster
1	498.15	1.17
2	5.95	2.79
3	11.80	1.19
4	8.84	1.17
5	5.92	1.42
95	3.14	2.18
96	1.63	2.15
97	1.29	NA
98	1.47	NA
99	1.62	NA
100	1.26	NA
101	0.90	NA
102	0.99	NA
103	0.82	NA

Long

	load_time	host
1	498.15	alentus
2	5.95	alentus
3	11.80	alentus
4	8.84	alentus
5	5.92	alentus
185	2.42	hostmonster
186	3.13	hostmonster
187	1.67	hostmonster
188	1.30	hostmonster
189	1.57	hostmonster
190	1.50	hostmonster
191	1.18	hostmonster
192	2.68	hostmonster
193	3.04	hostmonster
194	7.13	hostmonster
195	2.31	hostmonster
196	2.04	hostmonster
197	2.06	hostmonster
198	2.18	hostmonster
199	2.15	hostmonster

Wide Data – commonly used in data / results *reporting*

⚠ ***Column headers are not <u>variables</u>***

Variable: Company Name
Values: "Alentus" / "HostMonster"

	Alentus	HostMonster
1	498.15	1.17
2	5.95	2.79
3	11.80	1.19
4	8.84	1.17
5	5.92	1.42
95	3.14	2.18
96	1.63	2.15
97	1.29	NA
98	1.47	NA
99	1.62	NA
100	1.26	NA
101	0.90	NA
102	0.99	NA
103	0.82	NA

⚠ ***Each row is not about a single <u>case</u> / subject***

⚠ ***NA's where observations are missing***

✓ ***Compact representation of data***

Basketball Data

Team	Points	Assists	Rebounds
A	88	12	22
B	91	17	28
C	99	24	30
D	94	28	31

Read more: https://www.statology.org/long-vs-wide-data/

Long Data – commonly used in data *analysis*

✅ Every column is a variable

Variable: Company Name
Values: "Alentus" / "HostMonster"

	load_time	host
1	498.15	alentus
2	5.95	alentus
3	11.80	alentus
4	8.84	alentus
5	5.92	alentus
185	2.42	hostmonster
186	3.13	hostmonster
187	1.67	hostmonster
188	1.30	hostmonster
189	1.57	hostmonster
190	1.50	hostmonster
191	1.18	hostmonster
192	2.68	hostmonster
193	3.04	hostmonster
194	7.13	hostmonster
195	2.31	hostmonster
196	2.04	hostmonster
197	2.06	hostmonster
198	2.18	hostmonster
199	2.15	hostmonster

✅ Every row is an observation

✅ Can avoid *NA*s if observation is entirely missing

⚠ Lots of repeated values

⚠ Lengthy representation

Basketball Data

Team	Variable	Value
A	Points	88
A	Assists	12
A	Rebounds	22
B	Points	91
B	Assists	17
B	Rebounds	28
C	Points	99
C	Assists	24
C	Rebounds	30
D	Points	94
D	Assists	28
D	Rebounds	31

Reshaping Data

Manual Coding

```
alentus     <- na.omit(page_loads$Alentus)
hostmonster <- na.omit(page_loads$HostMonster)

loads_long <- data.frame(
  load_time = c(alentus, hostmonster),
  host = c(rep("alentus", length(alentus)),
           rep("hostmonster", length(hostmonster)))
)
```

data.frame *– tabular (tables)* ***data structure***

Using External Packages

CRAN: The Comprehensive R Archive Network

https://cran.r-project.org/

```
# install.packages("reshape2")
library(reshape2)
loads_long <- melt(page_loads, na.rm = TRUE,
                   variable.name = "host",
                   value.name    = "load_time")

# install.packages("tidyr")
library(tidyr)
loads_long <- gather(page_loads, na.rm = TRUE,
                     key   = "host",
                     value = "load_time")
```

🤔

Why are ***different packages*** *doing similar things?*

Which one should I ***pick****?!?*

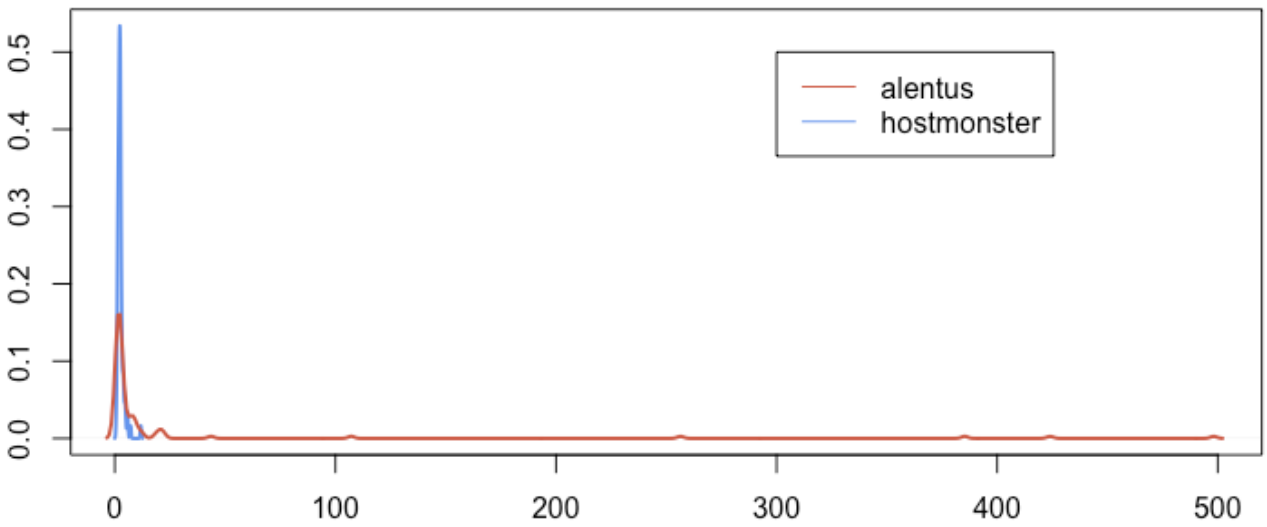
Is there value in ***coding this ourselves*** *if other packages can do it for us?*

Visualizing and Describing the Data

*We can **split** long data based on **groupings***

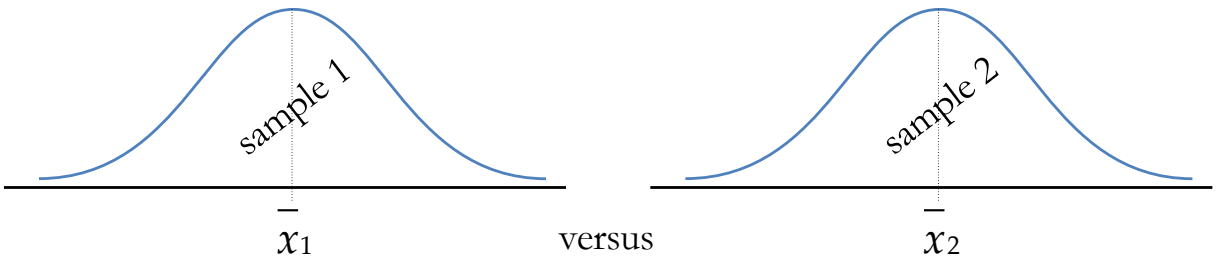
```
hosts <- split(x = loads_long, f = loads_long$host)
```

```
plot(density(hosts$HostMonster$load_time), col="cornflowerblue", lwd=2, xlim=c(0, 500))
lines(density(hosts$Alentus$load_time), col="coral3", lwd=2)
legend(300, 0.5, lty=1, c("alentus", "hostmonster"), col=c("coral3", "cornflowerblue"))
```



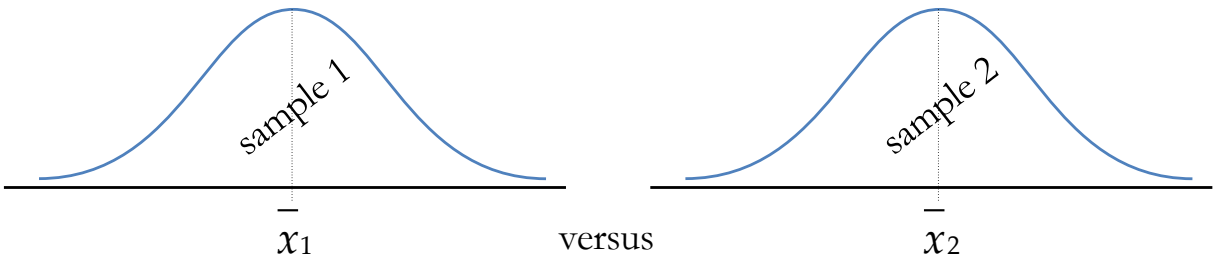

```
summary(hosts$Alentus$load_time)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#   0.41    1.55    2.24   20.65    6.58  498.15

summary(hosts$HostMonster$load_time)
#   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
#  0.730   1.745   2.240   2.523   2.692  11.770       7
```

t-Tests for Comparing Two Sample Means



Comparing means of independent samples			Comparing means of dependent (paired) samples
When population standard deviations are equal	**When population standard deviations are <u>not</u> equal**		
$t = \dfrac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\left(\dfrac{s_p^2}{n_1} + \dfrac{s_p^2}{n_2}\right)}}$ $df = n_1 + n_2 - 2$ $s_p^2 = \dfrac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}$ *pooled standard deviation*	$t = \dfrac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\left(\dfrac{s_1^2}{n_1} + \dfrac{s_2^2}{n_2}\right)}}$ $df = \dfrac{\left[\left(s_1^2/n_1\right) + \left(s_2^2/n_2\right)\right]^2}{\dfrac{\left(s_1^2/n_1\right)}{n_2 - 1} + \dfrac{\left(s_2^2/n_2\right)}{n_2 - 1}}$	$z = \dfrac{(\bar{x}_1 - \bar{x}_2)}{\sqrt{\left(\dfrac{s_1^2}{n_1} + \dfrac{s_2^2}{n_2}\right)}}$ *assumes* $n > 30$	$t = \dfrac{\bar{d}}{s_d/\sqrt{n}}$ $\bar{d} = \left(\sum x_1 - x_2\right)/n$ $s_d = \sqrt{\dfrac{\sum d_i^2 - n\bar{d}^2}{n - 1}}$

Two sample t-statistics in R



Comparing means of independent samples		Comparing means of dependent (paired) samples
When population standard deviations are equal	**When population standard deviations are <u>not</u> equal**	
Drug Effectiveness Study *(compare mean drug vs. placebo effectiveness)*	*Automobile Mileage Study* *(compare mean mileage of manual vs. automatic cars)*	*Athletic Training Study* *(compare mean performance before and after training)*

`drug    = c(15, 10, 13, 7,  9, 8, 21, 9, 14, 8)`
`placebo = c(15, 14, 12, 8, 14, 7, 16, 10, 15, 12)`
 	
`t.test(drug, placebo, alt="less", var.equal=TRUE)`
 	
`Two Sample t-test`
 	
`data:  drug and placebo`
`t = -0.5331, df = 18, p-value = 0.3002`
`alternative hypothesis: true difference in means is less than 0`
`95 percent confidence interval:`
`  -Inf 2.027436`
`sample estimates:`
`mean of x mean of y`
`  11.4  12.3`
	`mt.manual = mtcars[mtcars$am==1, ]`
	`mt.auto = mtcars[mtcars$am==0, ]`
	
	`t.test(mt.manual$mpg, mt.auto$mpg, var.equal=FALSE)`
	
	`Welch Two Sample t-test`
	
	`data:  mt.manual$mpg and mt.auto$mpg`
	`t = 3.7671, df = 18.332, p-value = 0.001374`
	`alternative hypothesis: true difference in means is not equal to 0`
	`95 percent confidence interval:`
	`  3.209684 11.280194`
	`sample estimates:`
	`mean of x mean of y`
	`  24.39231  17.14737`
		`before = c(12.9, 13.5, 12.8, 15.6, 17.2,`
		`  19.2, 12.6, 15.3, 14.4, 11.3)`
		`after = c(12.7, 13.6, 12.0, 15.2, 16.8,`
		`  20.0, 12.0, 15.9, 16.0, 11.1)`
		
		`t.test(before, after, paired=TRUE)`
		`Paired t-test`
		
		`data:  before and after`
		`t = -0.2133, df = 9, p-value = 0.8358`
		`alternative hypothesis: true difference in means is not equal to 0`
		`95 percent confidence interval:`
		`  -0.5802549  0.4802549`
		`sample estimates:`
		`mean of the differences`
		`  -0.05`

Student's Two-Sample t-Test

```
t.test(hosts$Alentus$load_time, hosts$HostMonster$load_time,
       alt="greater", var.equal=TRUE)


      Two Sample t-test

data:  hosts$Alentus$load_time and hosts$HostMonster$load_time
t = 2.2848, df = 197, p-value = 0.0117
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 5.014645      Inf
sample estimates:
mean of x mean of y
20.647184  2.522917
```

Assumptions of Student's Two-Sample t-Test

1. Both populations are normal
2. Variance of two populations are the same (*homoscedacticity*)

Welch's Two-Sample t-Test

```
t.test(hosts$Alentus$load_time, hosts$HostMonster$load_time,
       alt="greater", var.equal=FALSE)

      Welch Two Sample t-test

data:  hosts$Alentus$load_time and hosts$HostMonster$load_time
t = 2.367, df = 102.07, p-value = 0.00991
alternative hypothesis: true difference in means is greater than 0
95 percent confidence interval:
 5.413982      Inf
sample estimates:
mean of x mean of y
20.647184  2.522917
```

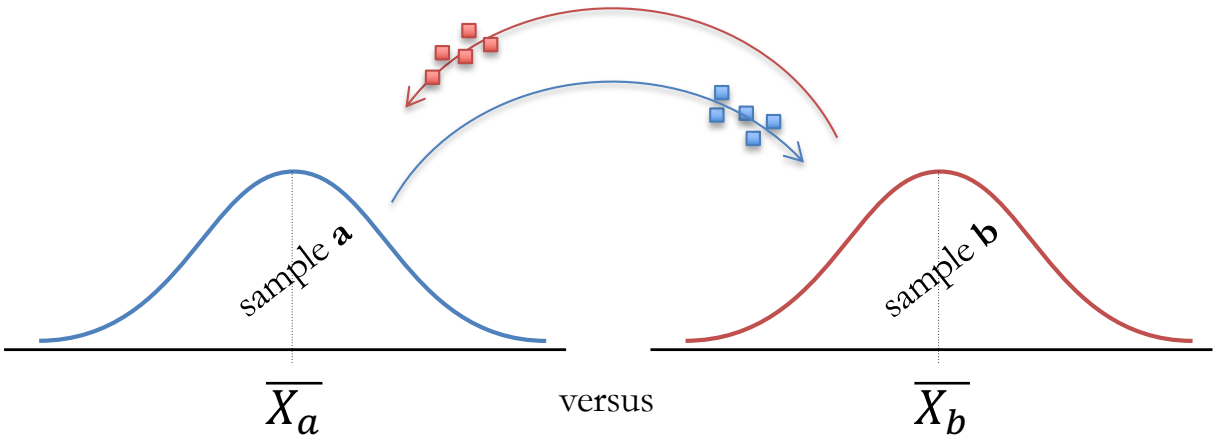
Assumptions of Welch's Two Sample t-Test

1. Both populations are normal
2. ~~Variance of two populations are the same~~

T-Tests are usually fairly robust to a bit of non-normality

Permutation Test

We can exchange data between groups in new **permutations**



If there is no difference between groups (*Null hypothesis*), then the original observed difference ($\overline{X_a} - \overline{X_b}$) should not be unusual compared to differences between permuted groups

Demonstration of Permutation Logic

Two Samples

```
a <- c(91:95) # [1] 91 92 93 94 95
b <- c(1:5)   # [1]  1  2  3  4  5
```

Observed Difference

```
observed_diff <- mean(a) - mean(b)
# 90
```

90

Permutation: *switch elements between groups*

```
values <- c(a, b)
[1] 91 92 93 94 95  1  2  3  4  5

groups <- c(rep('a', 5), rep('b', 5))
[1]  a  a  a  a  a  b  b  b  b  b

set.seed(42)
permuted <- sample(values, replace = FALSE)
[1] 91 95  5  3 92 94  1  4  2 93

grouped  <- split(permuted, groups)
$a
   [1] 91 95  5  3 92
$b
   [1] 94  1  4  2 93

mean(grouped$a) - mean(grouped$b)
[1] 18.4
```

Replacing <u>without</u> replacement
just switches positions!

Splitting with original labels
makes the "switch" complete

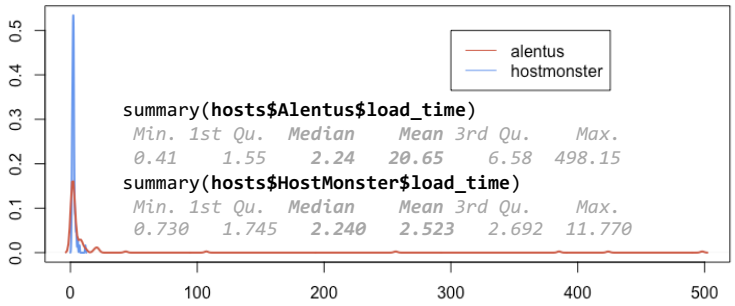
18.4

Observed vs. Permuted difference of groups are quite different!

🤔

What if we repeated the permutation many times?

Example: Alentus vs. HostMonster Load Times



H_{null}: *The mean of the two groups is the same*
H_{alt}: *The mean of Alentus is larger (or different) than the mean of HostMonster*

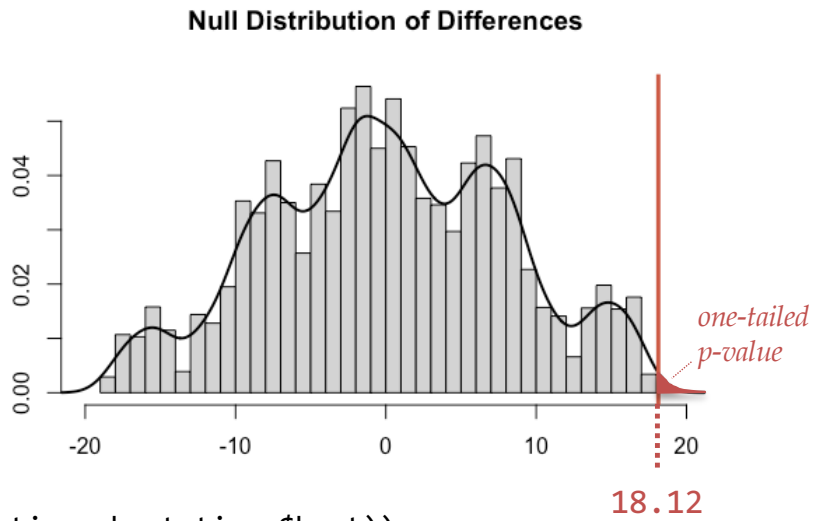
Observed Difference

```
observed_diff <- mean(hosts$Alentus$load_time) - mean(hosts$HostMonster$load_time)
[1] 18.12427
```

Permutations: *switch elements between groups*

```
permute_diff <- function(values, groups) {
  permuted <- sample(values, replace = FALSE)
  grouped <- split(permuted, groups)
  permuted_diff <- mean(grouped[[1]]) - mean(grouped[[2]])
}

nperms <- 10000
```

Number of permutations




```
permuted_diffs <- replicate(nperms, permute_diff(host_times$load_time, host_times$host))

hist(permuted_diffs, breaks = "fd", probability = TRUE)
lines(density(permuted_diffs), lwd=2)
```

```
p_1tailed <- sum(permuted_diffs > observed_diff) / nperms
[1] 0  --> never in 10,000 permutations!

p_2tailed <- sum(abs(permuted_diffs) > observed_diff) / nperms
[1] 0.0022  --> 0.22% of 10,000 permutations
```

One-tailed p-value
Percent of permutations where
diff $> \overline{X_a} - \overline{X_b}$, out of M

Two-tailed p-value
Percent of permutations where
absolute diff $> \overline{X_a} - \overline{X_b}$, out of M

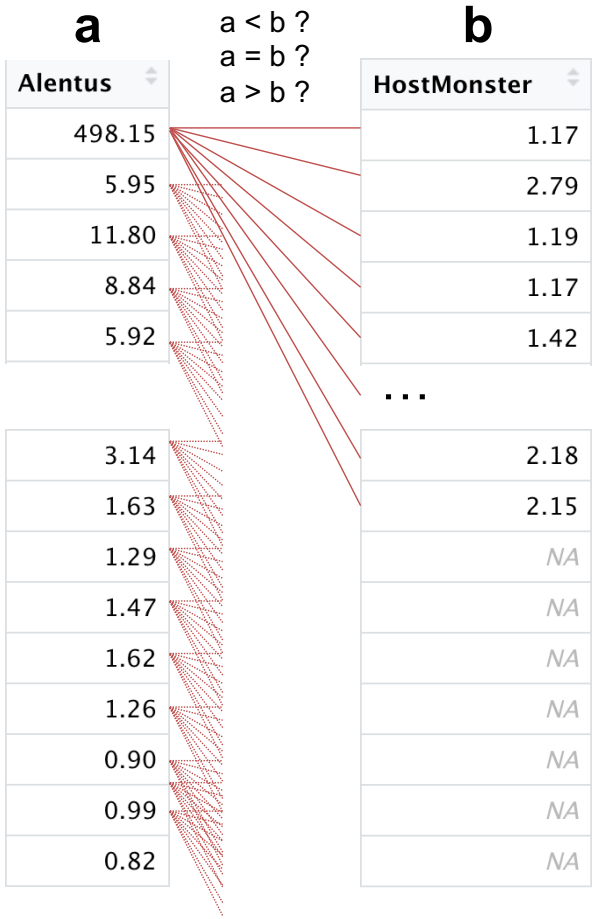
Wilcoxon Test

Wilcoxon Test as Pair-wise <u>Permutation</u>

The Wilcoxon Test can be thought of as a type of permutation test!

H_{null}: *The **distributions** of load times in Alentus and HostMonster populations are the same*
H_{alt}: *The **distribution** of load times for Alentus greater than (or different) the **distribution** load times of HostMonster*

We check every possible pair of (a, b) to see if a < b, a = b, or a > b



$$W = \sum_{\forall a, b} \begin{cases} a < b \rightarrow 0.0 \\ a = b \rightarrow 0.5 \\ a > b \rightarrow 1.0 \end{cases}$$

```
W <- 0
for(i in hosts$Alentus$load_time) {
  for(j in hosts$HostMonster$load_time) {
    if(i > j) {
      W <- W + 1
    } else if (i == j) {
      W <- W + 0.5
    }
  }
}

W
# [1] 5259
```

Can we write more elegant code that looks more like the math?

A functional and vectorized solution can represent the logic more ***cleanly*** *and* ***confidently***

```
gt_eq <- function(a, b) {
  ifelse(a > b, 1, 0) + ifelse(a == b, 0.5, 0)
}

W <- sum(outer(hosts$Alentus$load_time, hosts$HostMonster$load_time, FUN = gt_eq))
# [1] 5259
```

```
outer(1:4, 3:5, FUN = multiply)
     [,1] [,2] [,3]
[1,]    3    4    5
[2,]    6    8   10
[3,]    9   12   15
[4,]   12   16   20

outer(1:4, 3:5, FUN = "*")
outer(1:4, 3:5, FUN = "+")
```

```
multiply <- function(a, b) {
  # browser()
  a * b
}
```

```
Called from: FUN(X, Y, ...)
Browse[1]> a
 [1] 1 2 3 4 1 2 3 4 1 2 3 4
Browse[1]> b
 [1] 3 3 3 3 4 4 4 4 5 5 5 5
```

?outer **FUN** is called with these two extended **vectors** as arguments. It must be a **vectorized function** (or the name of one) expecting at least **two arguments** and returning a value with the **same length** as the first (and the second) vector.

Wilcoxon Test as "Rank Sum Test"

*We can **rank** the order of values across groups and find out **which group had bigger ranks**!*

```
loads_long
```

```
           host load_time
1       Alentus    498.15
2       Alentus      5.95
3       Alentus     11.80
.           .         .
101     Alentus      0.90
102     Alentus      0.99
103     Alentus      0.82
104 HostMonster      1.17
105 HostMonster      2.79
106 HostMonster      1.19
.           .         .
197 HostMonster      2.06
198 HostMonster      2.18
199 HostMonster      2.15
```

1. **Rank** the load times from 1 – 199 (ties use x.5)

```
time_ranks <- rank(loads_long$load_time)
 [1] 199.0 170.0 186.0 180.0 169.0 178.0  66.0  49.0 118.5  16.5   4.0  79.5 139.5  82.5  65.0
 [16] 113.0  81.0 102.5 177.0 171.0 198.0 189.0 193.0 183.0 181.0 146.0 141.0  46.0  27.0 161.0
 [31] 136.0  69.0 129.0 182.0 153.0 131.0 196.0 172.0 155.0 192.0 179.0 187.0  46.0 137.0  11.0
 [46]  98.0  33.0   8.0 100.0  43.5  43.5  42.0 176.0   1.0 108.5  10.0  20.0  96.5  35.5  14.5
 [61]  71.0  31.5   2.5 158.0 115.5  49.0 151.0 163.0 174.0  89.5  53.5  14.5 197.0 175.0 190.0
 [76] 194.0 188.0 191.0 184.0  88.0  18.0  16.5 147.0 160.0 150.0 195.0  68.0  12.0  64.0  25.0
 [91]  55.5  75.5  57.0   7.0 143.5  52.0  28.0  40.0  51.0  26.0   9.0  13.0   5.0  21.5 134.0
[106]  24.0  21.5  35.5  49.0  29.5  38.0 167.0  37.0  34.0  79.5 185.0 162.0 164.0 166.0 148.0
[121] 132.0 149.0 168.0 125.0 126.5 115.5 126.5 133.0 145.0 158.0  39.0  96.5  75.5 120.5 154.0
[136] 108.5  87.0  94.0  31.5  77.0 101.0  59.5 111.0  94.0  67.0  62.5   6.0   2.5 156.0 118.5
[151]  70.0  59.5  85.0  73.0  99.0  78.0  62.5  59.5  74.0 102.5  85.0 122.0 105.5 135.0 123.0
[166] 120.5 158.0  72.0 104.0  91.5 128.0 112.0 138.0 143.5 124.0 107.0  53.5  59.5  19.0 115.5
[181] 165.0 152.0  89.5 115.5 110.0 142.0  55.5  29.5  46.0  41.0  23.0 130.0 139.5 173.0 105.5
[196]  82.5  85.0  94.0  91.5
```

2. Gather and **sum** the **ranks** of each group

```
ranked_groups <- split(time_ranks, loads_long$host)
$Alentus
  [1] 199.0 170.0 186.0 180.0 169.0 178.0  66.0  49.0 118.5  16.5   4.0  79.5
 [76] 194.0 188.0 191.0 184.0  88.0  18.0  16.5 147.0 160.0 150.0 195.0  68.0
 [92] 75.5  57.0   7.0 143.5  52.0  28.0  40.0  51.0  26.0   9.0  13.0   5.0

$HostMonster
 [1]  21.5 134.0  24.0  21.5  35.5  49.0  29.5  38.0 167.0  37.0  34.0  79.5
[65]  72.0 104.0  91.5 128.0 112.0 138.0 143.5 124.0 107.0  53.5  59.5  19.0
[85]  29.5  46.0  41.0  23.0 130.0 139.5 173.0 105.5  82.5  85.0  94.0  91.5

U1 <- sum(ranked_groups$Alentus)
[1] 10615
```

3. Adjust the rank sum proportionally

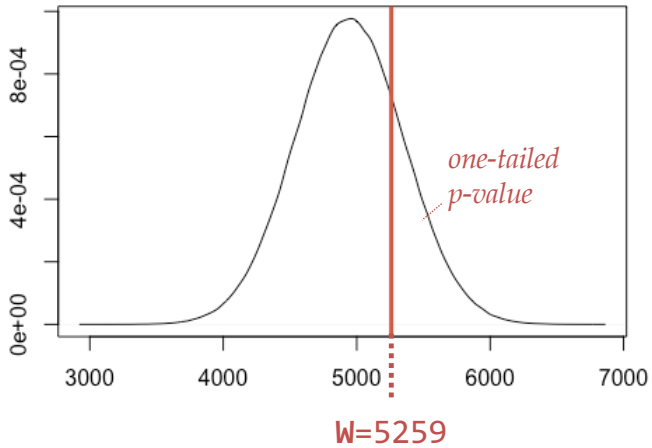
```
n1 <- nrow(hosts$Alentus)
W <- U1 - (n1 * (n1 + 1))/2
[1] 5259
```

This is the basics of computing W as Rank Sum
*There are **complications**:*
(a) handling ties; (b) size of samples;
(c) approximating the distributions, …

p-values of the Wilcoxon Two-Sample Test

`pwilcox()` function helps us find probabilities for Wilcox distribution

```
n1 <- nrow(hosts$Alentus)      # 103
n2 <- nrow(hosts$HostMonster) #  96

wilcox_p_1tail <- 1 - pwilcox(W, n1, n2)
# [1] 0.2189435

wilcox_p_2tail <- 2 * wilcox_p_1tail
# [1] 0.437887
```



*Wilcoxon Test is **not** a test of **medians**!*
(although it seems to behave like that at times)

Using built-in Wilcox test in R

```
wilcox.test(load_time ~ host, data = loads_long, alternative = "greater")

      Wilcoxon rank sum test with continuity correction
data:  load_time by host
W = 5259, p-value = 0.2192
alternative hypothesis: true location shift is greater than 0


wilcox.test(load_time ~ host, data = loads_long, alternative = "two.sided")

      Wilcoxon rank sum test with continuity correction
data:  load_time by host
W = 5259, p-value = 0.4385
alternative hypothesis: true location shift is not equal to 0
```

Computational Methods Have Complications…

?wilcox.test

Note

The literature is not unanimous about the definitions of the Wilcoxon rank sum and Mann-Whitney tests. The two most common definitions correspond to the sum of the ranks of the first sample with the minimum value subtracted or not: `R` subtracts and S-PLUS does not, giving a value which is larger by *m(m+1)/2* for a first sample of size *m*. (It seems Wilcoxon's original paper used the unadjusted sum of the ranks but subsequent tables subtracted the minimum.)

`R`'s value can also be computed as the number of all pairs `(x[i], y[j])` for which `y[j]` is not greater than `x[i]`, the most common definition of the Mann-Whitney test.

Different software may compute the Wilcoxon test in different ways…

1. Use a built-in method unless adapting the test for a different use

2. Always report which software you used!

R uses the permutation method!
(i.e., it does not use sum of ranks)