Getting back into R - and how to handle messy data

Advanced Research Methods and Skills

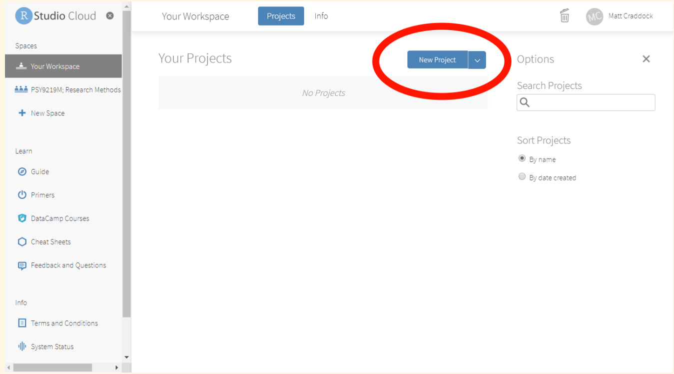
2021/03/16

Topics previously covered

- Data visualization
  - Using **ggplot2**
- Data manipulation
  - Using **dplyr**
- Basic statistics
  - t-tests, correlations
  - regression
  - ANOVA

Themes for today

- Refamiliarizing yourself with R
- Basic data transformations
- Missing data



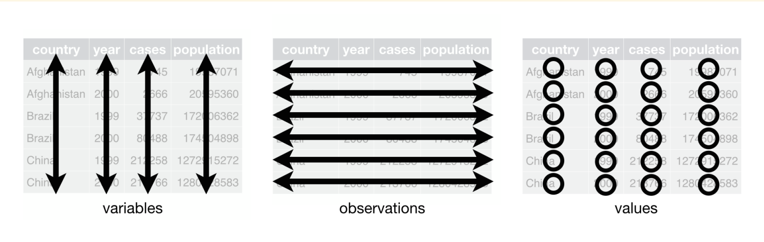
Tabular data

```
## # A tibble: 16 x 4
##    Participant Viewpoint Block     RT
##          <int> <chr>     <chr>  <dbl>
##  1           1 Different First   517.
##  2           1 Different Second  399.
##  3           1 Same      First   550.
##  4           1 Same      Second  420.
##  5           2 Different First   454.
##  6           2 Different Second  349.
##  7           2 Same      First   632.
##  8           2 Same      Second  357.
##  9           3 Different First   374.
## 10           3 Different Second  357.
## 11           3 Same      First   556.
## 12           3 Same      Second  425.
## 13           4 Different First   496.
## 14           4 Different Second  443.
## 15           4 Same      First   537.
## 16           4 Same      Second  487.
```

Tables of data are what you're most commonly dealing with in R.

This one confirms to the **tidy data** principles:

One row per observation, one column per variable



Different types of file

The most common file formats you'll deal with are either Excel files or text files, but you may also find dealing with SPSS files useful.

Fortunately, R has several functions and packages for importing data!

File formats	File extension	Functions	Package
SPSS	.sav	**read_sav()**	library(haven)
Excel	.xls, .xlsx	**read_excel()**	library(readxl)
Text	.csv, .txt, .*	**read_csv()**, **read_delim()**	library(readr)

Data wrangling

dplyr is a really useful package for manipulation of data tables.

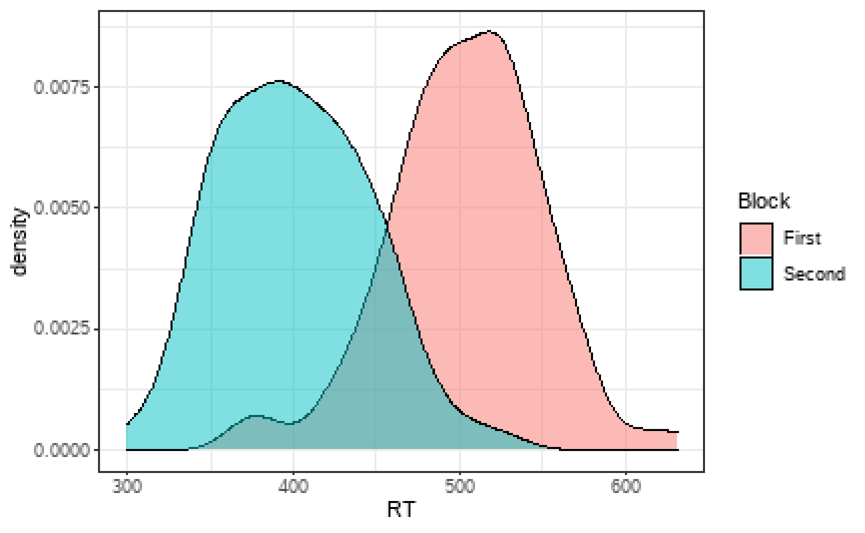
Function	Effect
select()	Include or exclude variables (columns)
arrange()	Change the order of observations (rows)
filter()	Include or exclude observations (rows)
mutate()	Create new variables (columns)
group_by()	Create groups of observations
summarise()	Aggregate or summarise groups of observations (rows)

Quickly plotting your data

Plotting data is really important for all aspects of data analysis.

The **ggplot2** package provides a framework for doing this:

```
ggplot(example_rt_df,
       aes(x = RT, fill = Block)) +
  geom_density(alpha = 0.5) +
  theme_bw()
```



Running statistics

The humble t-test...

```
t.test(RT~Block, data = example_rt_df)
```

```
##
##     Welch Two Sample t-test
##
## data:  RT by Block
## t = 16.158, df = 197.85, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##   90.60909 115.80141
## sample estimates:
##  mean in group First mean in group Second
##             503.8649             400.6597
```

Running statistics

The factorial ANOVA... (the aov_ez() function from the *afex* package)

```
aov_ez(dv = "RT",
       within = c("Block", "Viewpoint"),
       id = "Participant",
       data = example_rt_df)
```

```
## Anova Table (Type 3 tests)
##
## Response: RT
##            Effect    df     MSE          F   ges p.value
## 1           Block 1, 49 1969.79 270.37 ***  .569   <.001
## 2       Viewpoint 1, 49 2172.63       0.15 <.001    .699
## 3 Block:Viewpoint 1, 49 2267.93       0.04 <.001    .850
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

Reminder!

RStudio.cloud has a selection of built-in exercises covering a lot of the topics we've done before.

If you need a bit of practice, try going through them again!

Also **USE THE DROP-INS**, use the discussion board, and use the Teams channel!

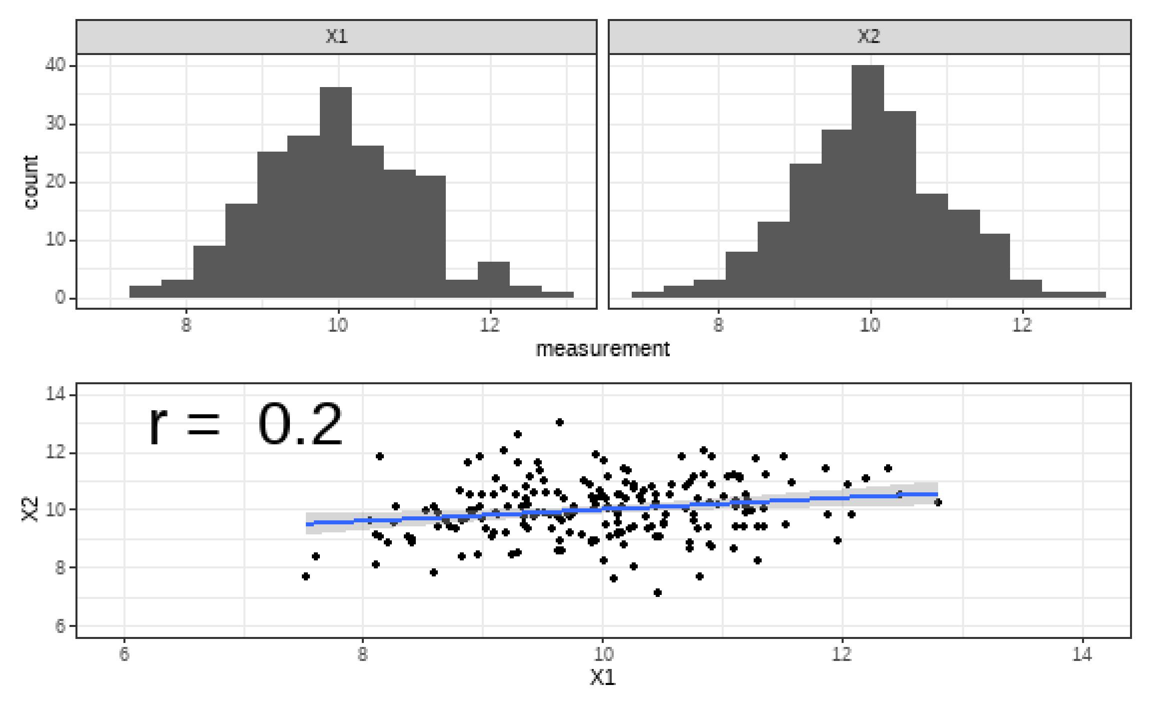
Each week, Matt will go over the exercises for that week, twice, half-way through the scheduled drop-in times - so at 1000 and 1200!

How to handle "messy" or otherwise awkward data

The ideal data

In an ideal world all our data would be beautifully normal:

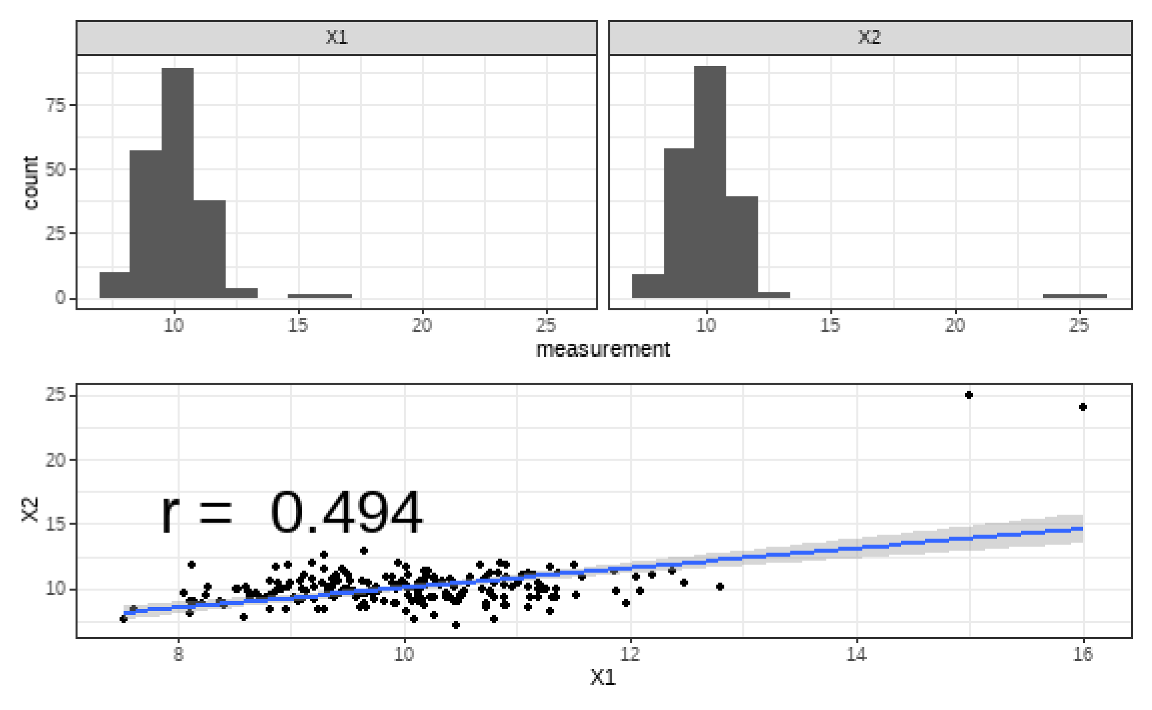
```
## `geom_smooth()` using formula 'y ~ x'
```



The real data

But reality is rarely so kind. Data can be all kinds of messy. It can have *outliers*...

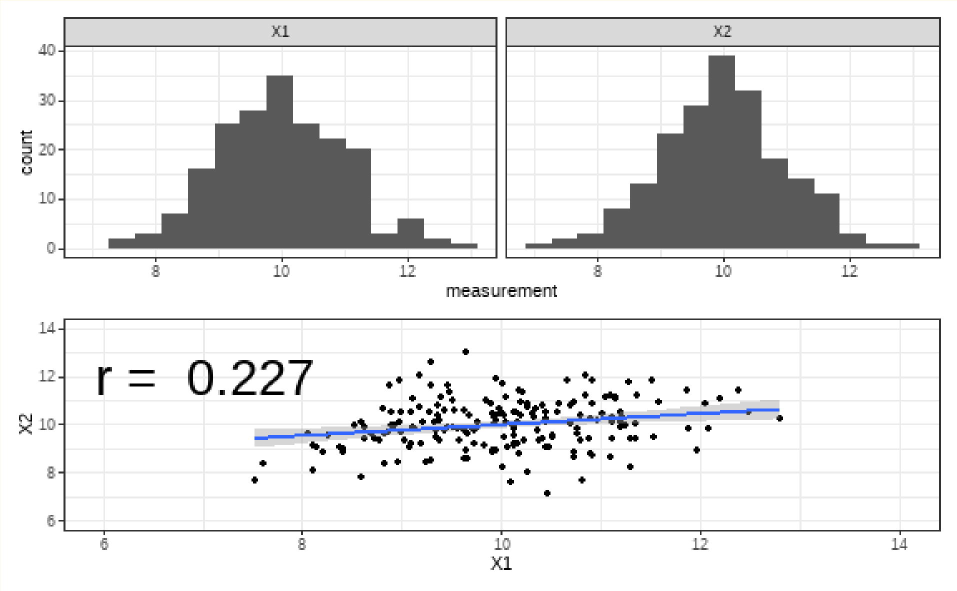
```
## `geom_smooth()` using formula 'y ~ x'
```



The real data

Data can be *missing*...

```
## `geom_smooth()` using formula 'y ~ x'
```

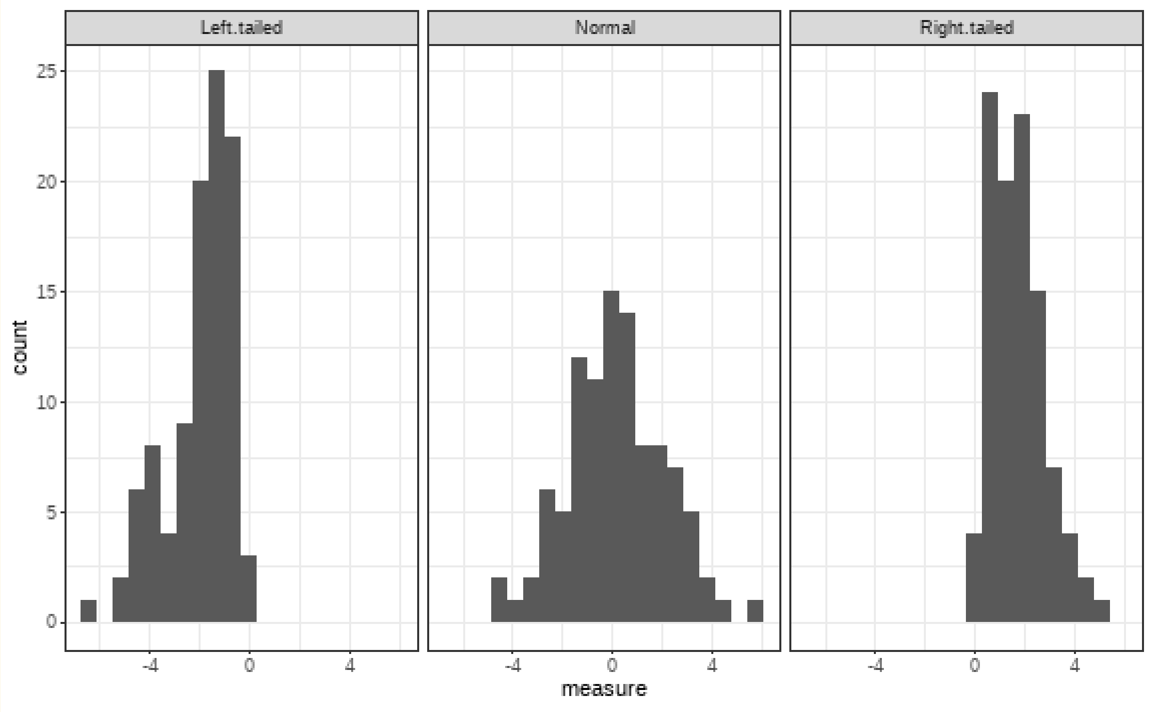



```
##             X1        X2
## 1           NA  9.426102
## 2           NA 10.037950
## 3           NA 10.095301
## 4           NA  9.376683
## 5           NA 11.819401
## 6    10.532486        NA
## 7    10.764667        NA
## 8    10.124343  9.856845
## 9     8.412414  8.829608
## 10    9.299633  8.516352
```

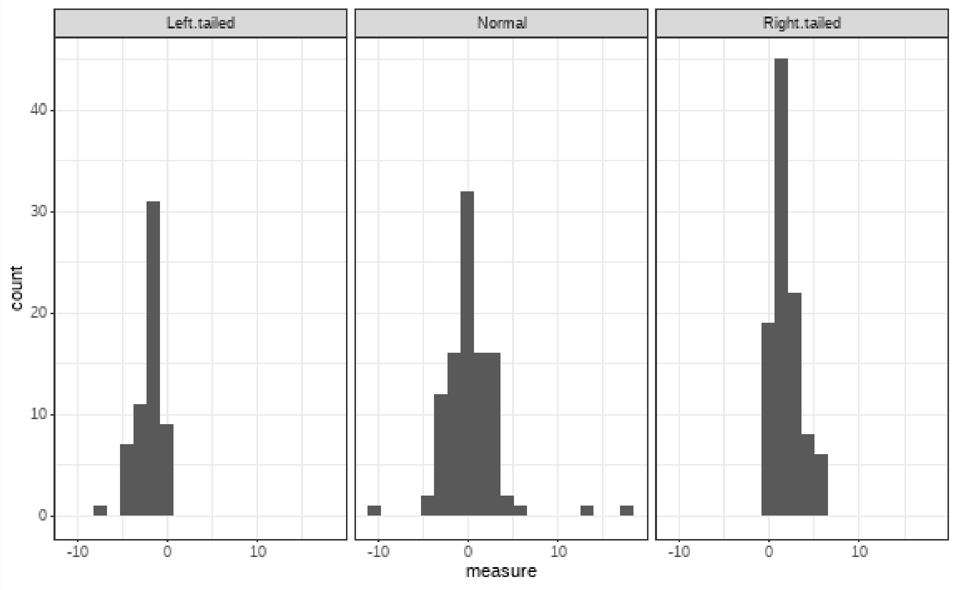
Complete cases = 193

The real data

Data can be *skewed*...



The real data



There can be any combination of these things...

All of these pose problems for estimating the properties of our data, the relationships between variables, and the phenomena we are investigating.

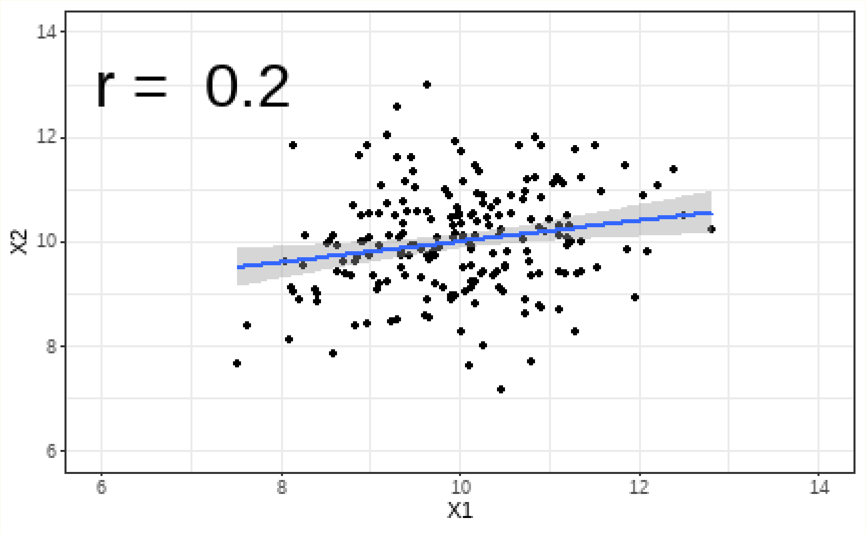
Handling outliers

What is an outlier?

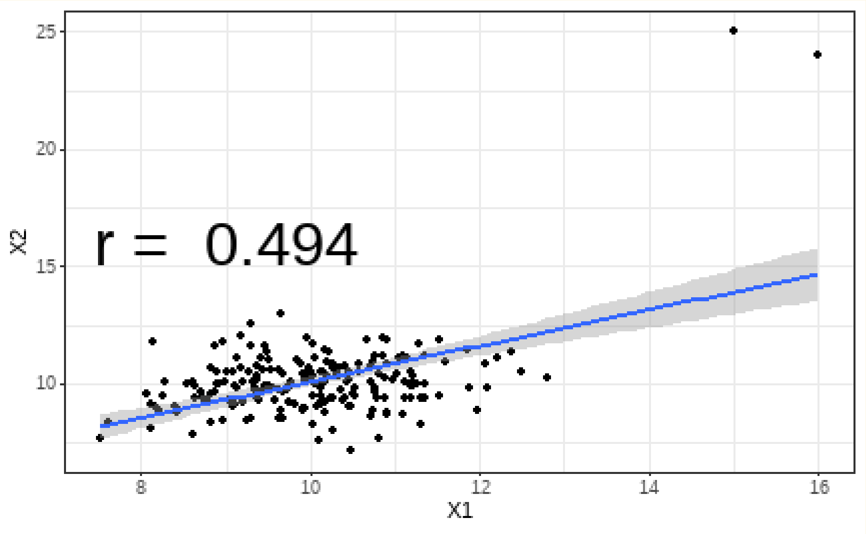
Two out of the 200 pairs of x-y values were replaced.

The resulting coefficient (approx *r* = .49) is *way-off* the true coefficient for these data.

```
## `geom_smooth()` using formula 'y ~ x'
```




```
## `geom_smooth()` using formula 'y ~ x'
```



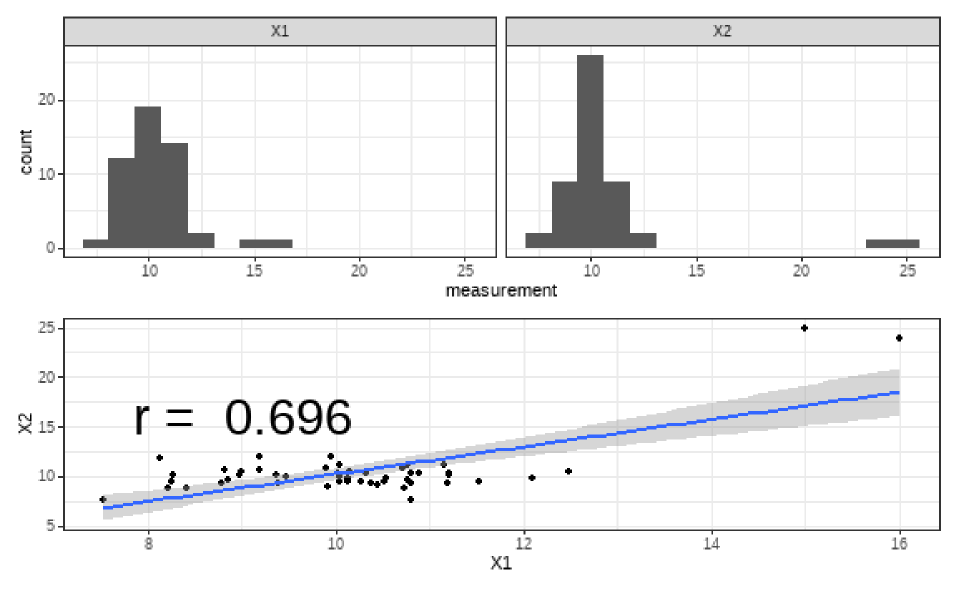
What is an outlier?

The problem gets even worse with smaller sample sizes.

Here there are 50 datapoints with two outliers, rather than 200.

The correlation coefficient becomes even *more* biased than it was previously.

```
## `geom_smooth()` using formula 'y ~ x'
```



What should we do with outliers?

Three common approaches:

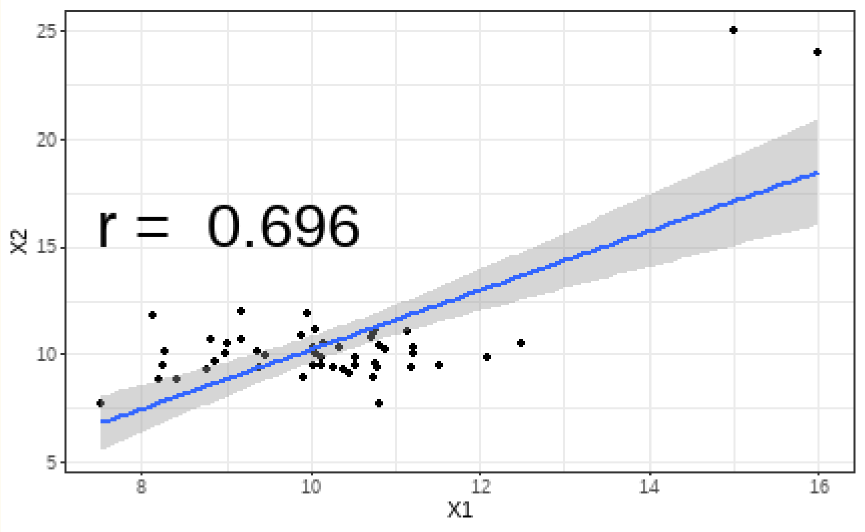
1. **Remove them**
    - If you're sure these reflect an error, not genuine data, then removal is a possibility.
2. **Transformation**
    - *rescaling* or *transforming* your data may help reduce the influence of outliers. (We'll come back to this!)
3. **Replace them**
    - Replacing the outliers with values $\pm$ 2-3 standard deviations away from the mean.

Identifying and replacing outliers

Identifying outliers

Plotting your data can be an excellent way to spot outliers: here they're *very* obvious!

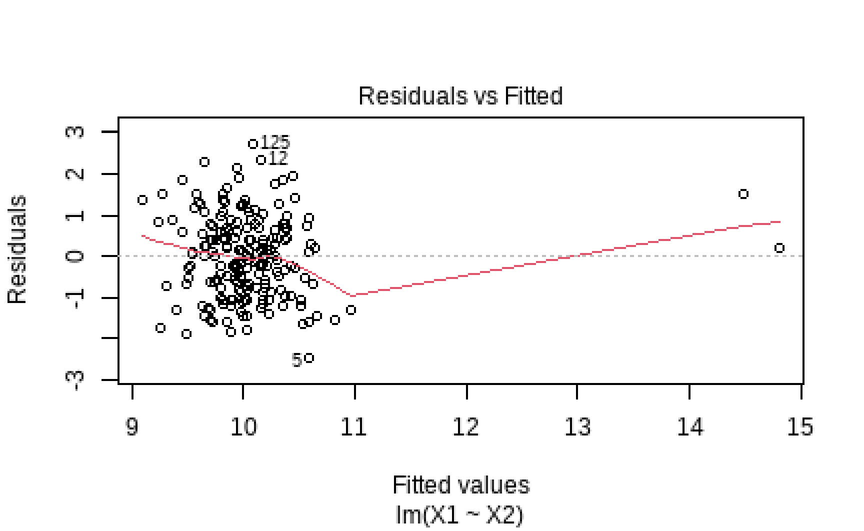
```
## `geom_smooth()` using formula 'y ~ x'
```

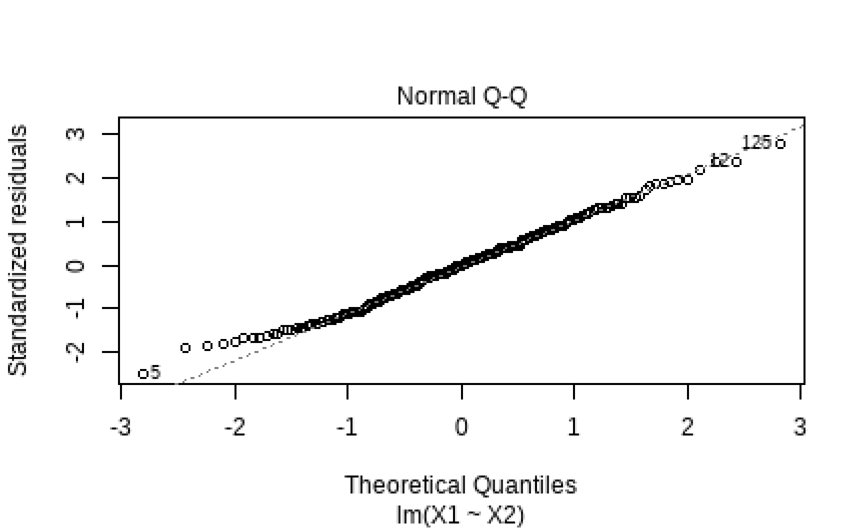


Identifying outliers

Plotting the residuals of your linear model will also help you identify troublesome observations.

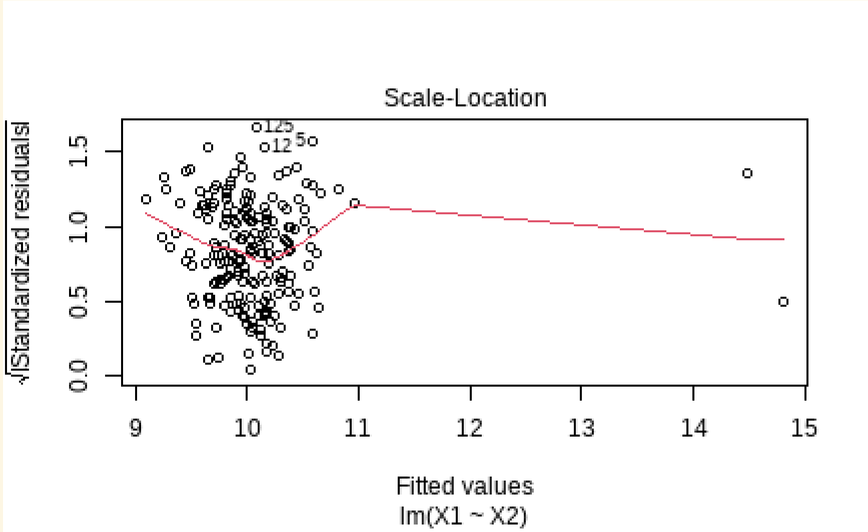
```
plot(lm(X1 ~ X2, data = temp_df_out))
```

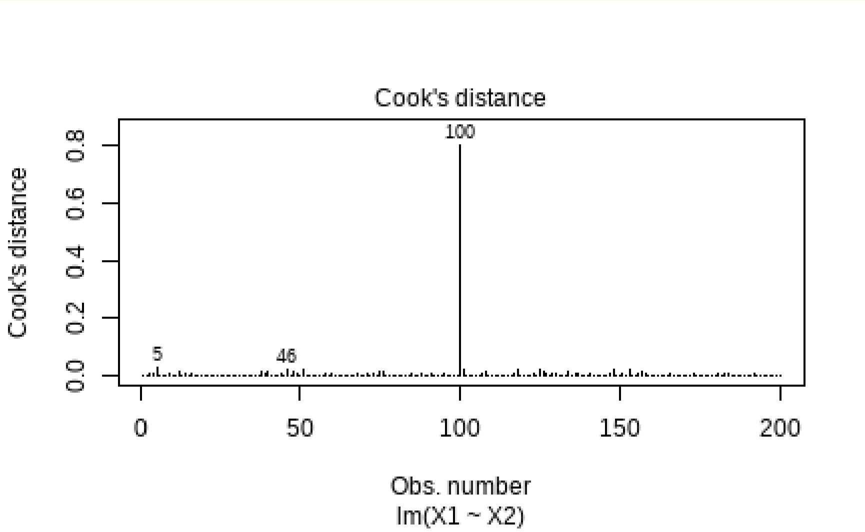




Identifying outliers

Plotting the residuals of your linear model will also help you identify troublesome observations.



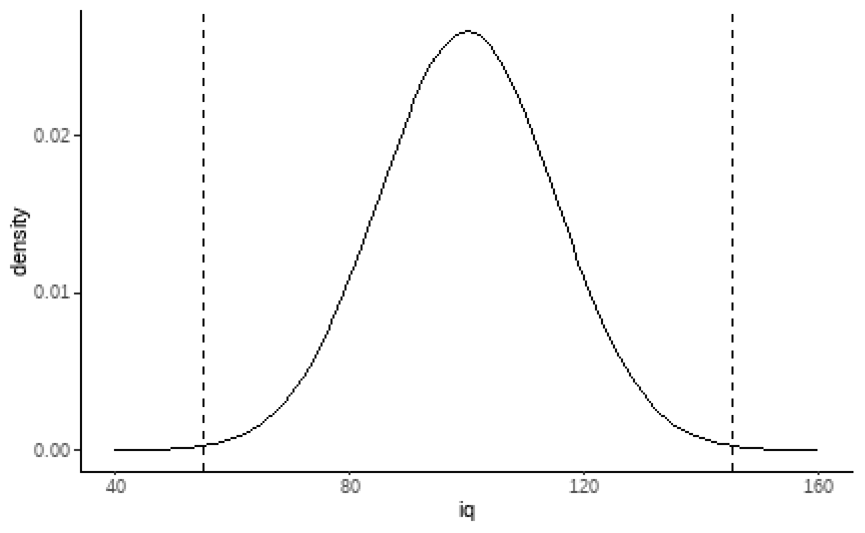


Identifying outliers

Sometimes a *threshold* is used to determine whether an observation is an outlier.

Sometimes this is driven by common sense: e.g. a value of 120 for a participant's age is **extremely** unlikely to be genuine.

Sometimes this is *data-driven*: e.g. values more than $\pm$ 3 standard deviations away from the mean are *unusual*.



Scaling by the mean and standard deviation

The data | Manually scale | scale()

```
example <- c(rnorm(15), 35)
hist(example)
```

Scaling by the mean and standard deviation

The data | **Manually scale** | scale()

```r
example
```

```
##  [1]  0.46022243  1.49799547 -0.82338515 -0.28118514  1.42268324 -0.79391589
##  [7]  0.36079445 -1.16870954 -0.02524575  0.71011527 -0.01553508 -0.19555281
## [13] -0.20691027 -1.09108919 -0.62968784 35.00000000
```

```r
(example - mean(example)) / sd(example)
```

```
##  [1] -0.19076686 -0.07282520 -0.33664732 -0.27502694 -0.08138435 -0.33329818
##  [7] -0.20206673 -0.37589302 -0.24593974 -0.16236684 -0.24483613 -0.26529493
## [13] -0.26658569 -0.36707156 -0.31463385  3.73463736
```

Scaling by the mean and standard deviation

The data | Manually scale | scale()

```
# you don't need to use t() - that transposes the values so they fit on the slide :)
# Just use scale()
t(scale(example))
```

```
##             [,1]       [,2]       [,3]       [,4]        [,5]       [,6]
## [1,] -0.1907669 -0.0728252 -0.3366473 -0.2750269 -0.08138435 -0.3332982
##             [,7]      [,8]       [,9]      [,10]      [,11]      [,12]
## [1,] -0.2020667 -0.375893 -0.2459397 -0.1623668 -0.2448361 -0.2652949
##            [,13]      [,14]      [,15]    [,16]
## [1,] -0.2665857 -0.3670716 -0.3146339 3.734637
## attr(,"scaled:center")
## [1] 2.138787
```

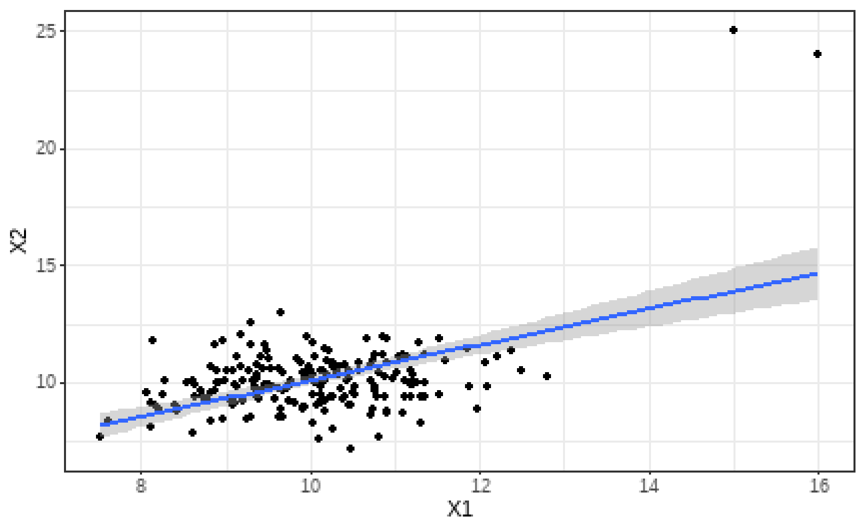
Removing values above a threshold

The **filter()** function from dplyr can be used to remove outliers easily!

With outliers | Without outliers

```
temp_df_out %>%
  ggplot(aes(x = X1, y = X2)) +
  geom_point() +
  theme_bw() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```



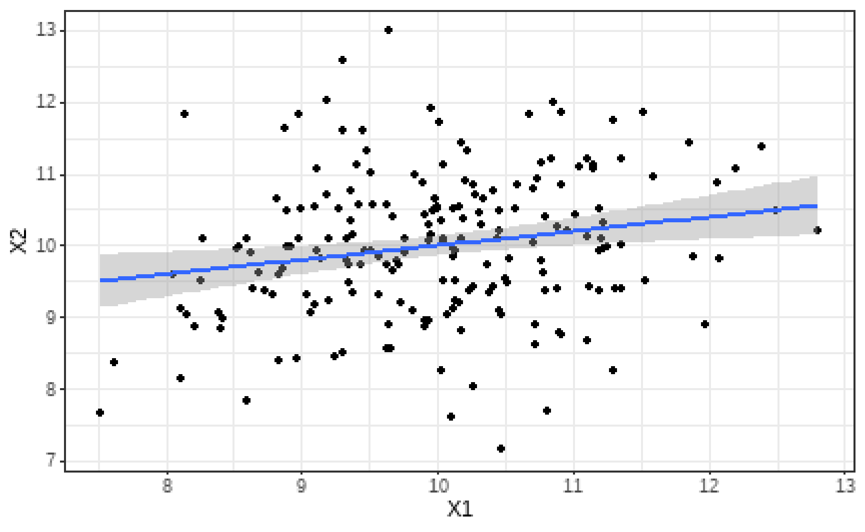
Removing values above a threshold

The **filter()** function from dplyr can be used to remove outliers easily!

With outliers | Without outliers

```
temp_df_out %>%
  dplyr::filter(X1 < 15) %>%
  ggplot(aes(x = X1, y = X2)) +
  geom_point() + theme_bw() +
  stat_smooth(method = "lm")
```

```
## `geom_smooth()` using formula 'y ~ x'
```

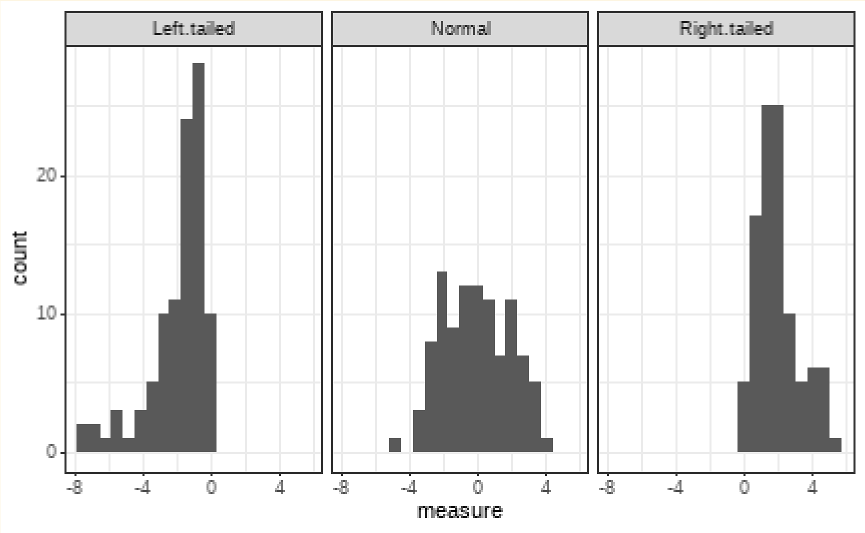


Data transformation

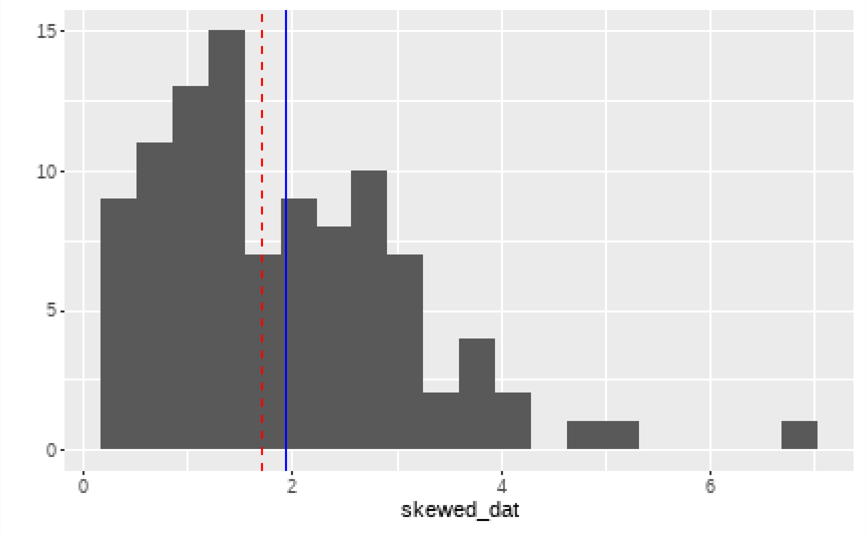
Skewed data

Skewed data is data that *leans* in a particular direction.

These are often described by the direction of the "long-tail" - so a left-tailed distribution means a distribution with a long tail on the left, rather than most values on the left.



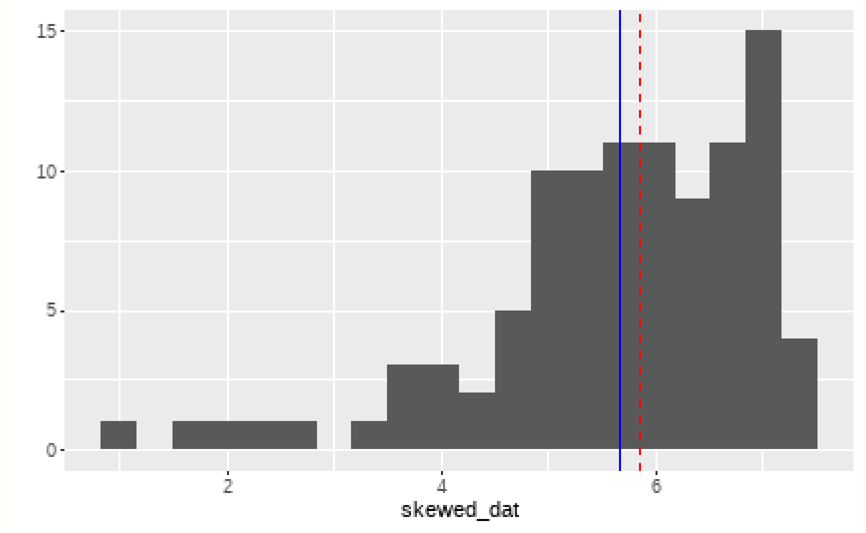
Skewed data



This data is *right-tailed*. This is sometimes also called *positively skewed*. For this type of data, the mean (blue line) is usually higher than the median (red, dashed line).

This type of skew is relatively common with data that is *bounded* at zero. e.g. reaction time data, the distribution of wages

Left-tailed skew



Data skewed the opposite way - many high scores but few low scores - has a long *left* tail. This is also called *negative skew*. The mean (blue solid line) is usually less than the median (red dashed line).

Transformation of skewed data

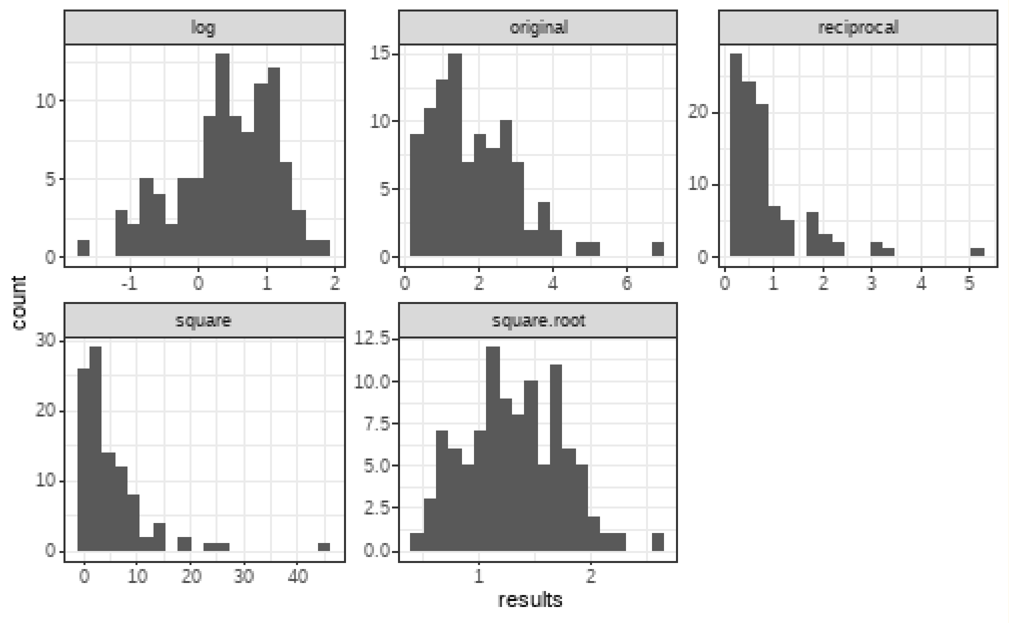
One way to handle skew is to transform the data to a different *scale*.

Transformation type	code
Log	log(X)
Square root	sqrt(X)
Reciprocal	1 / X
Square	x^2

(See Section 5.8.2 in Field et al., DSUR)

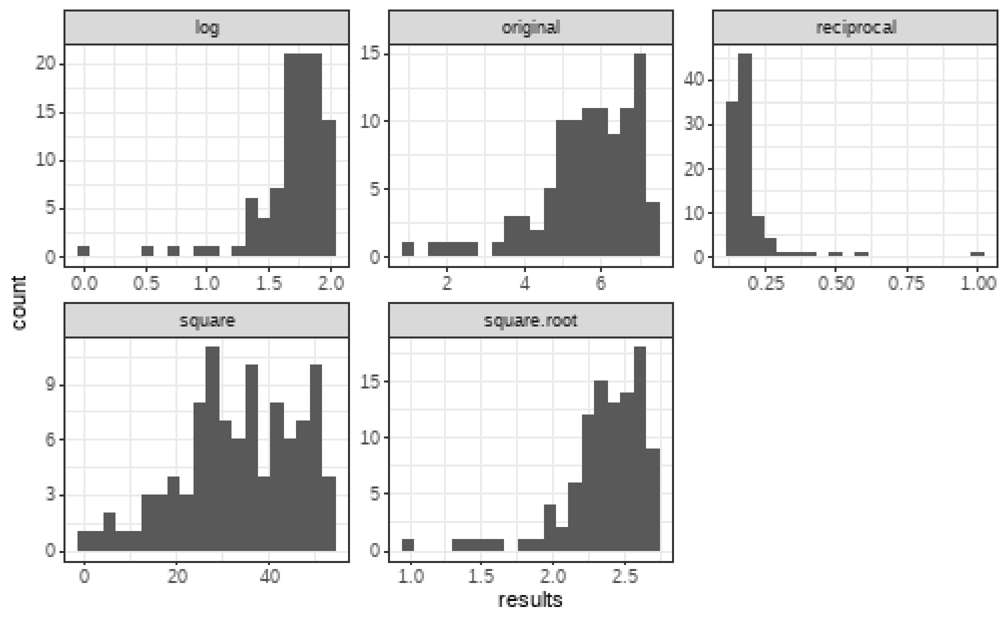
Transformation of skewed data

Right-tailed   Left-tailed



Transformation of skewed data

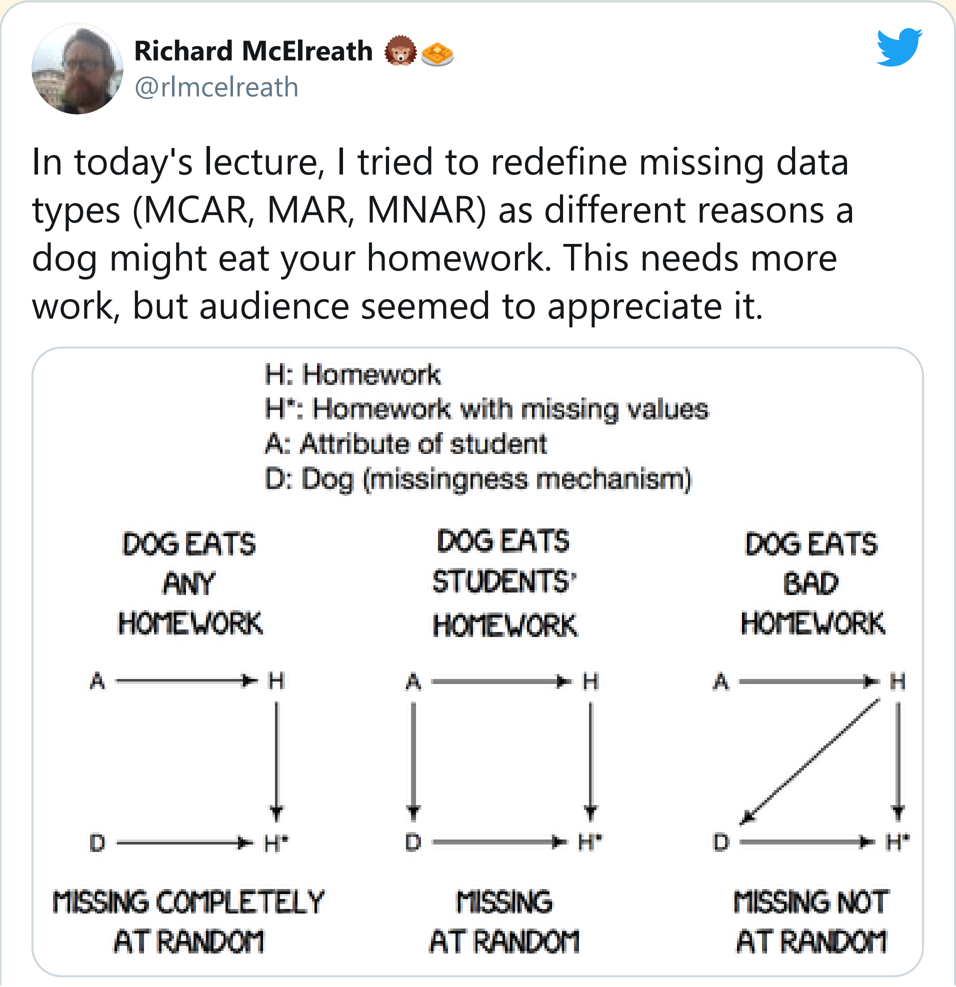
Right-tailed | Left-tailed



Handling missing data

Types of missing data

- Missing Completely At Random
  - Missingness does not depend on anything
- Missing At Random
  - Missingness depends on the observed data
- Missing Not At Random
  - Missingness depends on the missing data



Missing Completely At Random

If you have missing data, MCAR is the best kind of missing data.

There is nothing *systematic* about which data is missing.

For example, all your participants filled out three different questionnaires. Unfortunately, your dog chewed through a pile of them, and half of your participants now have only two questionnaires.

```
##           X1        X2       X3
## 1         NA 10.564509 12.00366
## 2         NA 11.134840 12.46372
## 3         NA  8.197005 11.06698
## 4         NA  9.704088 14.19018
## 5         NA  8.706921 13.03839
## 6   8.344476  9.537686 12.89438
## 7  11.263166  7.791435 13.76203
## 8   8.669327  8.276546 12.92337
## 9   8.377079  9.164257 11.07893
## 10  9.288548  6.947327 12.80245
```

Missing At Random

```
##           X1        X2       X3 age
## 1  11.047154  9.130396 12.86073  19
## 2   8.509500 10.760188 12.07602  20
## 3  11.619125  9.636584 13.04400  19
## 4  11.212559  7.600109 11.90494  20
## 5  10.951217  9.341155 13.35871  19
## 6   9.286955 10.227833 12.48842  19
## 7   9.555602  8.320163       NA  36
## 8  10.394811  7.726169 15.13988  19
## 9   8.376943  8.289583       NA  26
## 10 10.528843  8.812628 13.50900  20
```

Confusingly, Missing At Random (MAR) data is not missing (completely) at random.

For example, for some reason, people older than 21 typically failed to complete the third questionnaire.

This data is MAR - whether the data in the third column is missing is related to the value of the fourth column.

Missing Not At Random

```
##               X1        X2       X3 age
## 1  11.095201  8.235348       NA  19
## 2  11.159246  9.198542       NA  19
## 3  11.026114 11.395065       NA  29
## 4  10.723638 10.066800       NA  19
## 5  11.399608 11.517779       NA  19
## 6  10.051001  9.103957 12.61627  18
## 7   9.073119  7.654592 13.10197  42
## 8  10.110520 10.048114 12.38895  29
## 9   9.729318 10.017504       NA  19
## 10 11.901271  9.132584 13.14818  52
```

The final, most troubling type of missing data is data that is Missing Not At Random (MNAR).

For example, imagine that the questionnaire relates to depression; people who score high for depression are less likely to complete the final questionnaire.

In this case, the values that are missing for the third questionnaire depends on the value of the responses to that questionnaire, so this data is MNAR.

Dealing with missing data

List-wise deletion: Cases with missing data are completely **removed** from **all** analysis.

Pair-wise deletion: Cases with missing data are only **removed** from **comparisons where one or more variables are missing**.

By default, functions such as **mean()** return NA if any value in the input is NA/missing.

```
mean(temp_df_missing$X1)
```

```
## [1] NA
```

```
mean(temp_df_missing$X1, na.rm = TRUE)
```

```
## [1] 10.01075
```

```
sum(complete.cases(temp_df_missing))
```

```
## [1] 193
```

Single Imputation

Replace missing values with a simple "best-guess". e.g. Using the mean or the median for the condition.

```r
orig_data <- 1:12
orig_data
```

```
##  [1]  1  2  3  4  5  6  7  8  9 10 11 12
```

```r
mean(orig_data)
```

```
## [1] 6.5
```

```r
missing_one <- orig_data
missing_one[6] <- NA
missing_one
```

```
##  [1]  1  2  3  4  5 NA  7  8  9 10 11 12
```

```r
mean(missing_one,  na.rm = TRUE)
```

```
## [1] 6.545455
```

Single Imputation

Replace missing values with a simple "best-guess". e.g. Using the mean or the median for the condition.

```
replace_one <- missing_one
replace_one[6] <- mean(missing_one, na.r
mean(replace_one)
```

```
## [1] 6.545455
```

```
mean(orig_data)
```

```
## [1] 6.5
```

Problem: the mean and median are biased by the missing data. And replacing a missing value with one of these values tends to artificially reduce variability.

Multiple Imputation

In **multiple** imputation, we replace missing values with estimates based on a *model* of the data that incorporates uncertainty about what the value should be.

We create a model based on the data that is not missing, and use its predictions to guess the values that the missing data has.

We do this multiple times and then take an average or *pool* the results to fill in the gap.

Packages such as **mice** and **Amelia** can do this for us, and help us identify patterns of missingness.

Alternative approaches to missing data, skew, and other oddities

Generalized Linear Models (as opposed to General Linear Models) allow modelling of data of many different types without necessitating transformations.

For example, counts can be modelled using Poisson regression, and categorical outcomes can be modelled with logistic regression.

Multilevel or *mixed*-models can handle all of these things and much more besides; they are perfectly capable of handling missing data.

We'll cover both logistic regression and multilevel models later in the course!

Next week

Look into power and effect sizes:

See Field et al, Discovering Statistics Using R, pages 56-59, Sections on:

```
- Type I and Type II error (2.6.3)
- effect sizes (2.6.4)
- statistical power (2.6.5)
```

Cohen, J. (1992). A power primer. Psychological Bulletin, 112(1), 155-159.
http://dx.doi.org/10.1037/0033-2909.112.1.155