An Incomplete Solutions Guide to the NIST/SEMATECH e-Handbook of Statistical Methods

examples and case studies using the tidyverse and ggplot2

Ray Hoobler

2018-07-16

Contents

Preface

Exploratory Data Analysis (EDA) is a philosophy on how to work with data, and for many applications, the workflow is better suited for most working scientist and engineers. As a scientist, we are trained to formulate a hypothesis and design a series of experiments that will allow us to test the hypothesis effectively. Unfortunately, most data doesn't from carefully controlled trials, but from observations. Statisticians will readily jump into describing the difference in as much detail as you would like.

For most of us, we need tools to characterize an instrument or a process. The philosophy of EDA provides the framework to do this work.

Unfortunately, most textbooks still focus on traditional statistical techniques and even while it is essential to understand the underlying assumptions and fundamentals, I would argue that most of the work we do as scientist and engineers are not well suited for rigorous statistical analysis. In many cases, the need to disseminate information to a broad audience is best served by the methods espoused by EDA. The NIST e-Handbook Engineering Statistics is a welcome deviation from the norm.

In the Spring of 2018, I adopted this text as the basis of a one-semester, graduate course that focused applied statistical techniques. The audience for this course were working scientist, and the course was a core course in a Professional Science Master's (PSM).

Unfortunately, the one drawback of the NIST Handbook is the use of Dataplot as the primary software package for analysis. The authors have provided examples using the R statistical language; however, most–if not all–of these scripts are written using base R which is unfortunate. Modern R now incorporates many packages for streamlining the EDA process. This book attempts to capture my efforts to use these methods and share them with students in the course. The two packages that I primarily used were **tidyverse** and **ggplot2**.

Before going further, I should clarify one thing—I'm a hack. I classify learning as three levels: novice, hack, expert.

Novice: basic knowledge of how to use a tool with a desire to learn. Hack: Basic to intermediate knowledge of how to use a tool accompanied by resources to produce a finished product. Expert: Extensive knowledge of how to use a tool; can produce a finished product with few outside resources.

I'm sure other factors can be added to each category, but these capture the spirit of how I approach learning.

The number of resources available to learn R is numerous, and the first I would strongly recommend is R for Data Science. This text is an introduction to the tidyverse. The tidyverse is not just a collection of R packages, but a philosophy on how to work with data. It makes data analysis almost fun!

The other primary resource available for EDA is ggplot2. Like the tidyverse, ggplot2 is not just a package of tools, but a philosophy built around the Grammar of Graphics.

I encourage the reader to explore the references related to these two packages and their underlying design philosophies.

This book will show how I have worked through the exercises and case studies presented in the NIST handbook using methods found in the tidyverse and ggplot2. I have found this framework to be incredibly satisfying and one I was eager to share beyond my class.

If you find this material useful, please send me an email.

Structure of the book

Content was built around the e-book NIST/SEMATECH e-Handbook of Statistical Methods.

At the begining of each exercise or case study, I've included a link back to the specific page of the e-Handbook. The e-Handbook can be downloaded in full from the NIST site. The compressed file is over 100Mb (not 43Mb) as stated.

Software information and conventsions

Follow "best practices" of the *tityverse*

The R session information for this book is shown below:

```r
sessionInfo()
```

```
## R version 3.4.3 (2017-11-30)
## Platform: x86_64-apple-darwin15.6.0 (64-bit)
## Running under: macOS High Sierra 10.13.5
##
## Matrix products: default
## BLAS: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/3.4/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
##  [1] multcompView_0.1-7 multcomp_1.4-8     TH.data_1.0-8
##  [4] MASS_7.3-49        survival_2.42-3    mvtnorm_1.0-7
##  [7] lme4_1.1-17        Matrix_1.2-14      broom_0.4.4
## [10] magrittr_1.5       bindrcpp_0.2.2     forcats_0.3.0
## [13] stringr_1.3.0      dplyr_0.7.4        purrr_0.2.4
## [16] readr_1.1.1        tidyr_0.8.0        tibble_1.4.2
## [19] ggplot2_2.2.1      tidyverse_1.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.16     lubridate_1.7.4  lattice_0.20-35  zoo_1.8-1
##  [5] assertthat_0.2.0 rprojroot_1.3-2  digest_0.6.15    psych_1.8.3.3
##  [9] utf8_1.1.3       R6_2.2.2         cellranger_1.1.0 plyr_1.8.4
## [13] backports_1.1.2  evaluate_0.10.1  httr_1.3.1       pillar_1.2.1
## [17] rlang_0.2.0      lazyeval_0.2.1   readxl_1.0.0     rstudioapi_0.7
## [21] minqa_1.2.4      nloptr_1.0.4     rmarkdown_1.9    labeling_0.3
## [25] splines_3.4.3    foreign_0.8-69   munsell_0.4.3    compiler_3.4.3
## [29] modelr_0.1.1     xfun_0.1         pkgconfig_2.0.1  mnormt_1.5-5
## [33] htmltools_0.3.6  tidyselect_0.2.4 bookdown_0.7     codetools_0.2-15
```

```
## [37] crayon_1.3.4     grid_3.4.3       nlme_3.1-137     jsonlite_1.5
## [41] gtable_0.2.0     scales_0.5.0     cli_1.0.0        stringi_1.1.7
## [45] reshape2_1.4.3   xml2_1.2.0       sandwich_2.4-0   tools_3.4.3
## [49] glue_1.2.0       hms_0.4.2        parallel_3.4.3   yaml_2.1.18
## [53] colorspace_1.3-2 rvest_0.3.2      knitr_1.20       bindr_0.1.1
## [57] haven_1.1.1
```

Acknowledgements

This book was created using the **bookdown** package (Xie, 2018), which was built on top of R Markdown and **knitr** (Xie, 2015).

Ray James Hoobler
Salt Lake City, Utah May 2018

Chapter 1

Exploratory Data Analysis

1.1 A EDA Example

An EDA/Graphics Example

The Anscombe dataset is an excelent place to start as it will allow us to start using R immediately. The anscombe dataset is part of the **datasets** package and is automatically loaded with RStudio.

```
anscombe
```

```
##    x1 x2 x3 x4    y1   y2    y3    y4
## 1  10 10 10  8  8.04 9.14  7.46  6.58
## 2   8  8  8  8  6.95 8.14  6.77  5.76
## 3  13 13 13  8  7.58 8.74 12.74  7.71
## 4   9  9  9  8  8.81 8.77  7.11  8.84
## 5  11 11 11  8  8.33 9.26  7.81  8.47
## 6  14 14 14  8  9.96 8.10  8.84  7.04
## 7   6  6  6  8  7.24 6.13  6.08  5.25
## 8   4  4  4 19  4.26 3.10  5.39 12.50
## 9  12 12 12  8 10.84 9.13  8.15  5.56
## 10  7  7  7  8  4.82 7.26  6.42  7.91
## 11  5  5  5  8  5.68 4.74  5.73  6.89
```

1.2 But first… let's start working in the tidyverse

The tidyverse is discribed as

> an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

You can install the **tidyverse** package with

```
install.packages("tidyverse")
```

Once installed, simply load the package:

```
library(tidyverse)
```

Additional details can be found at tidyverse.org

If you have only created charts and graphs using spreadsheets, you will assume the data is ready to plot. It might be nice to have the x1 and y1 values closer together in the table, but we could still select the individual columns and plot the datasets.

We're going to jump right in with with the idea of *tidy data*. That each row should be a single observation.

As mentioned in the introduction, this text assumes a basic knowledge of the tidyverse. In this example, we will selct the x data from the data frame, rename the colunn labels, use the `gather()` function to tidy the data. We will then repeat the process for the y data, removing the group names from the data set. The last step is to combine these two data frames into a single data frame we will use for plotting. I'm sure there are more efficient ways to do this; however, the code used to do this manipulation is typical when working with non-tidy data. An added benifit is that hte code is readable.

```
x_anscombe <- anscombe %>%  # results will be storred into a new object x_anscombe; we start with the o
  dplyr::select(x1, x2, x3, x4) %>%  # select the columns we want to work with
  rename(group1 = x1, group2 = x2, group3 = x3, group4 = x4) %>% # rename the values using a generic he
  gather(key = group, value = x_values, group1, group2, group3, group4) # gather the columns into rows

x_anscombe
```

```
##     group x_values
## 1  group1       10
## 2  group1        8
## 3  group1       13
## 4  group1        9
## 5  group1       11
## 6  group1       14
## 7  group1        6
## 8  group1        4
## 9  group1       12
## 10 group1        7
## 11 group1        5
## 12 group2       10
## 13 group2        8
## 14 group2       13
## 15 group2        9
## 16 group2       11
## 17 group2       14
## 18 group2        6
## 19 group2        4
## 20 group2       12
## 21 group2        7
## 22 group2        5
## 23 group3       10
## 24 group3        8
## 25 group3       13
## 26 group3        9
## 27 group3       11
## 28 group3       14
## 29 group3        6
## 30 group3        4
## 31 group3       12
## 32 group3        7
## 33 group3        5
## 34 group4        8
## 35 group4        8
```

```
## 36 group4        8
## 37 group4        8
## 38 group4        8
## 39 group4        8
## 40 group4        8
## 41 group4       19
## 42 group4        8
## 43 group4        8
## 44 group4        8
```

```
y_anscombe <- anscombe %>%
  dplyr::select(y1, y2, y3, y4) %>%
  gather(key = group, value = y_values, y1, y2, y3, y4) %>% # I don't need to rename the columns as I w
  dplyr::select(y_values)

y_anscombe
```

```
##    y_values
## 1      8.04
## 2      6.95
## 3      7.58
## 4      8.81
## 5      8.33
## 6      9.96
## 7      7.24
## 8      4.26
## 9     10.84
## 10     4.82
## 11     5.68
## 12     9.14
## 13     8.14
## 14     8.74
## 15     8.77
## 16     9.26
## 17     8.10
## 18     6.13
## 19     3.10
## 20     9.13
## 21     7.26
## 22     4.74
## 23     7.46
## 24     6.77
## 25    12.74
## 26     7.11
## 27     7.81
## 28     8.84
## 29     6.08
## 30     5.39
## 31     8.15
## 32     6.42
## 33     5.73
## 34     6.58
## 35     5.76
## 36     7.71
## 37     8.84
```

```
## 38     8.47
## 39     7.04
## 40     5.25
## 41    12.50
## 42     5.56
## 43     7.91
## 44     6.89
```

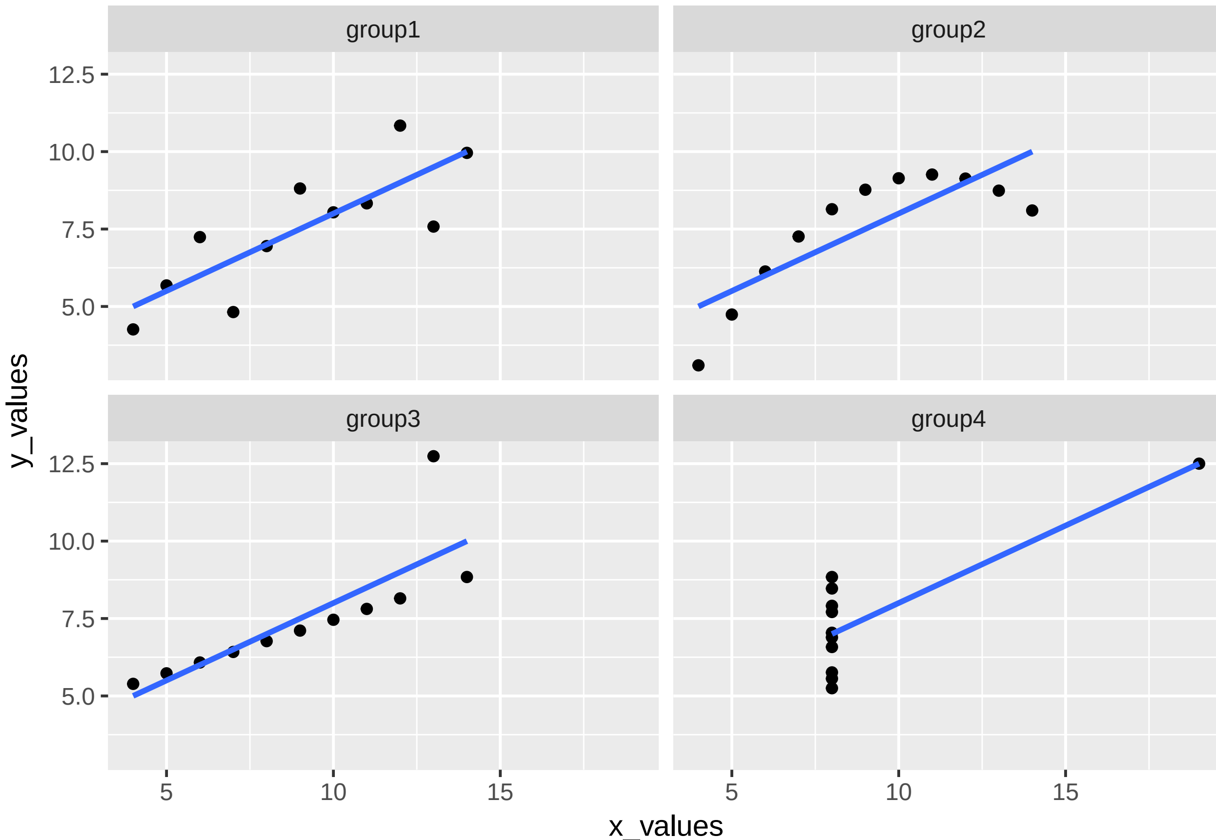
```
anscombe_tidy <- bind_cols(x_anscombe, y_anscombe)
anscombe_tidy
```

```
##     group x_values y_values
## 1  group1       10     8.04
## 2  group1        8     6.95
## 3  group1       13     7.58
## 4  group1        9     8.81
## 5  group1       11     8.33
## 6  group1       14     9.96
## 7  group1        6     7.24
## 8  group1        4     4.26
## 9  group1       12    10.84
## 10 group1        7     4.82
## 11 group1        5     5.68
## 12 group2       10     9.14
## 13 group2        8     8.14
## 14 group2       13     8.74
## 15 group2        9     8.77
## 16 group2       11     9.26
## 17 group2       14     8.10
## 18 group2        6     6.13
## 19 group2        4     3.10
## 20 group2       12     9.13
## 21 group2        7     7.26
## 22 group2        5     4.74
## 23 group3       10     7.46
## 24 group3        8     6.77
## 25 group3       13    12.74
## 26 group3        9     7.11
## 27 group3       11     7.81
## 28 group3       14     8.84
## 29 group3        6     6.08
## 30 group3        4     5.39
## 31 group3       12     8.15
## 32 group3        7     6.42
## 33 group3        5     5.73
## 34 group4        8     6.58
## 35 group4        8     5.76
## 36 group4        8     7.71
## 37 group4        8     8.84
## 38 group4        8     8.47
## 39 group4        8     7.04
## 40 group4        8     5.25
## 41 group4       19    12.50
## 42 group4        8     5.56
## 43 group4        8     7.91
```

```
## 44 group4        8     6.89
```

While this may seem like a lot of work to make a new table—which is much harder to read—this method allows us to exploit the **gramar of graphics** used by the **ggplot2** package.

```r
ggplot(anscombe_tidy, aes(x_values, y_values)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  facet_wrap(~group)
```



It may not be immediately obvious from the plots, but the slope and intercept for each line are identical. We can calculate these values for each dataset using the linear model function, `lm()`.

```r
lm(y1 ~ x1, data = anscombe)
```

```
##
## Call:
## lm(formula = y1 ~ x1, data = anscombe)
##
## Coefficients:
## (Intercept)           x1
##      3.0001       0.5001
```

```r
lm(y2 ~ x2, data = anscombe)
```

```
##
## Call:
## lm(formula = y2 ~ x2, data = anscombe)
##
## Coefficients:
```

```
## (Intercept)           x2
##       3.001        0.500
```

```r
lm(y3 ~ x3, data = anscombe)
```

```
##
## Call:
## lm(formula = y3 ~ x3, data = anscombe)
##
## Coefficients:
## (Intercept)           x3
##      3.0025       0.4997
```

```r
lm(y4 ~ x4, data = anscombe)
```

```
##
## Call:
## lm(formula = y4 ~ x4, data = anscombe)
##
## Coefficients:
## (Intercept)           x4
##      3.0017       0.4999
```

The calculated slope and intercept are the same (at least to three significant figures); the use of EDA allows us to differentiate the data quickly.

1.3 Common graphical analysis used in the e-Handbook

Four techniques are routinely used in the e-Handbook for preliminary EDA. These four charts are routinely displayed as a “4-plot.” Each technique will be presented in the following sub-sections.

- Run sequence plot
- Lag plot
- Histogram
- Normal probility plot

1.4 Case studies from chapter 1 of the NIST/SEMATECH e-Handbook

1.4.1 Normal random numbers

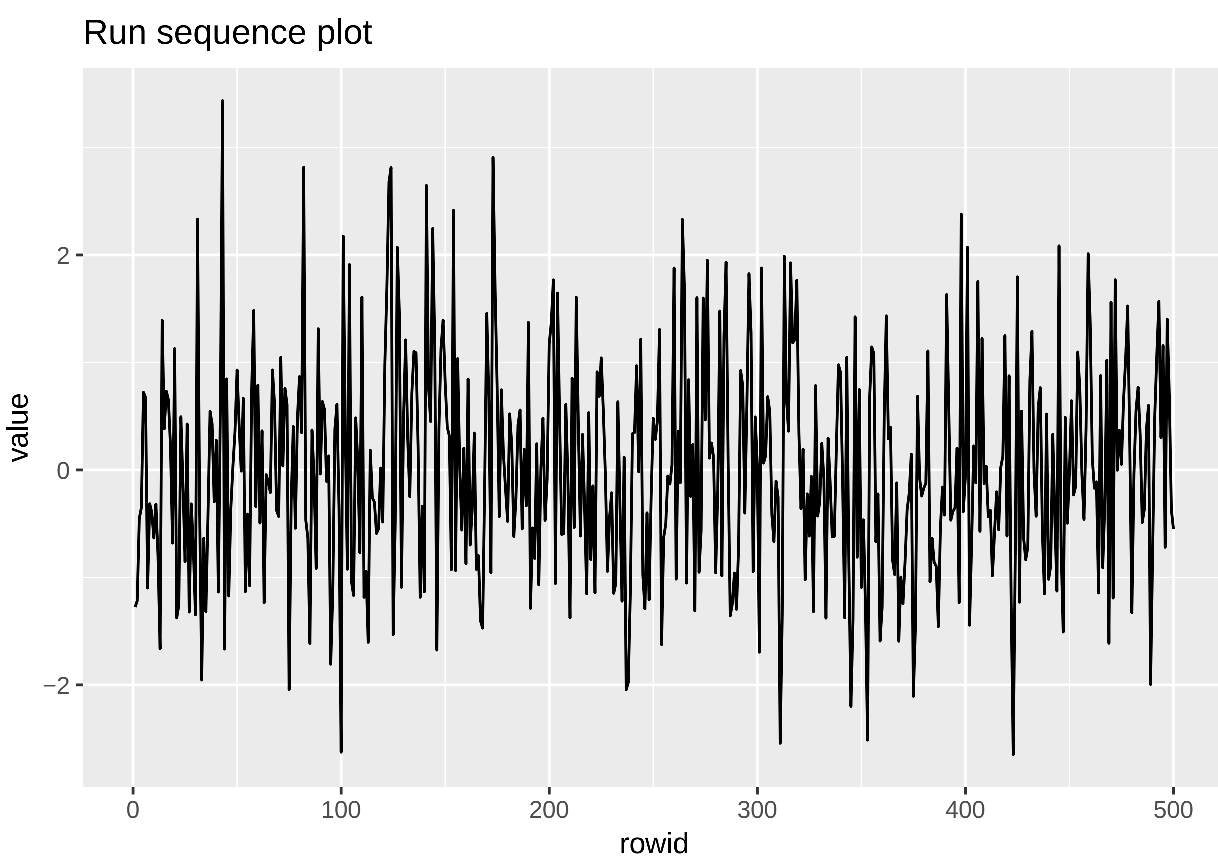
Normal Random Numbers

```r
normal_random_numbers <- scan("NIST data/RANDN.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

normal_random_numbers
```

```
## # A tibble: 500 x 2
##    rowid  value
##    <int>  <dbl>
##  1     1 -1.28
```

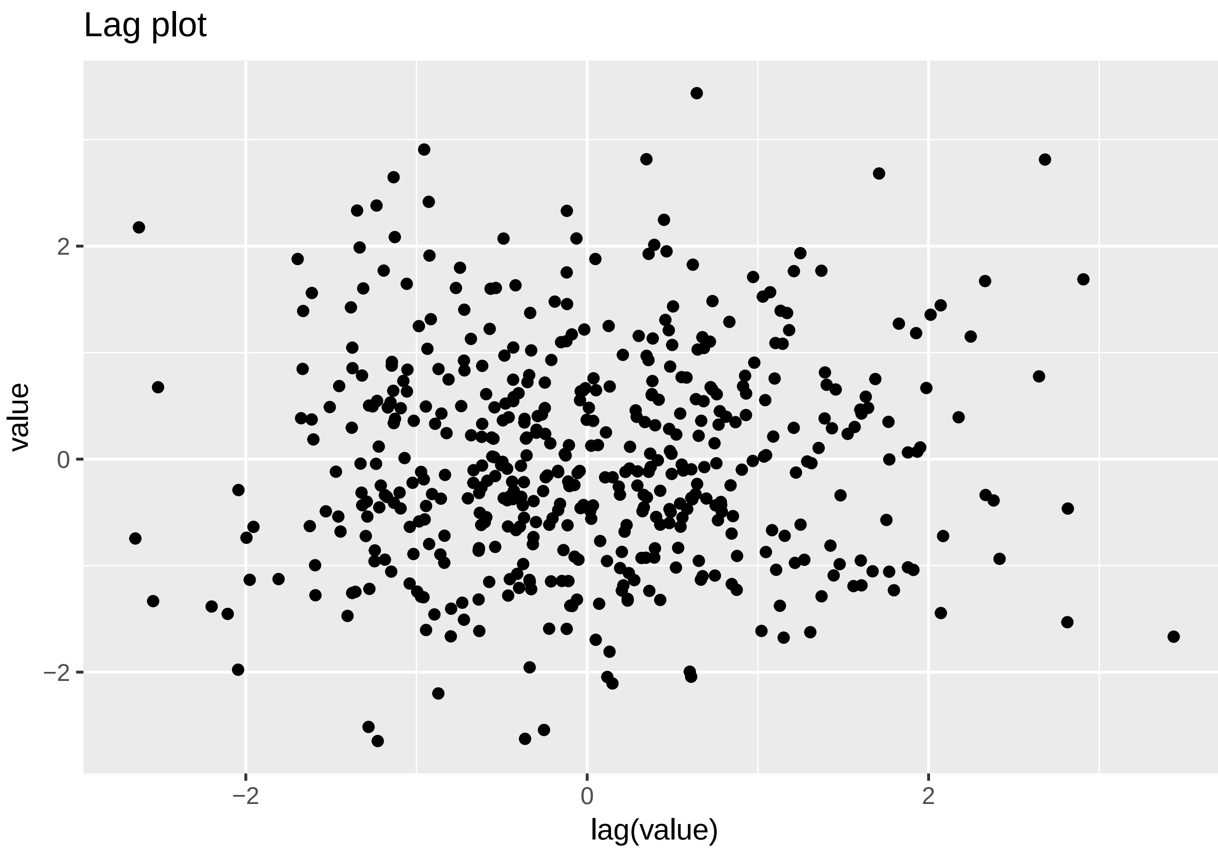
```
## 2      2 -1.22
## 3      3 -0.453
## 4      4 -0.350
## 5      5  0.723
## 6      6  0.676
## 7      7 -1.10
## 8      8 -0.314
## 9      9 -0.394
## 10    10 -0.633
## # ... with 490 more rows
```

```
ggplot(normal_random_numbers, aes(rowid, value)) +
  geom_line() +
  labs(title = "Run sequence plot")
```



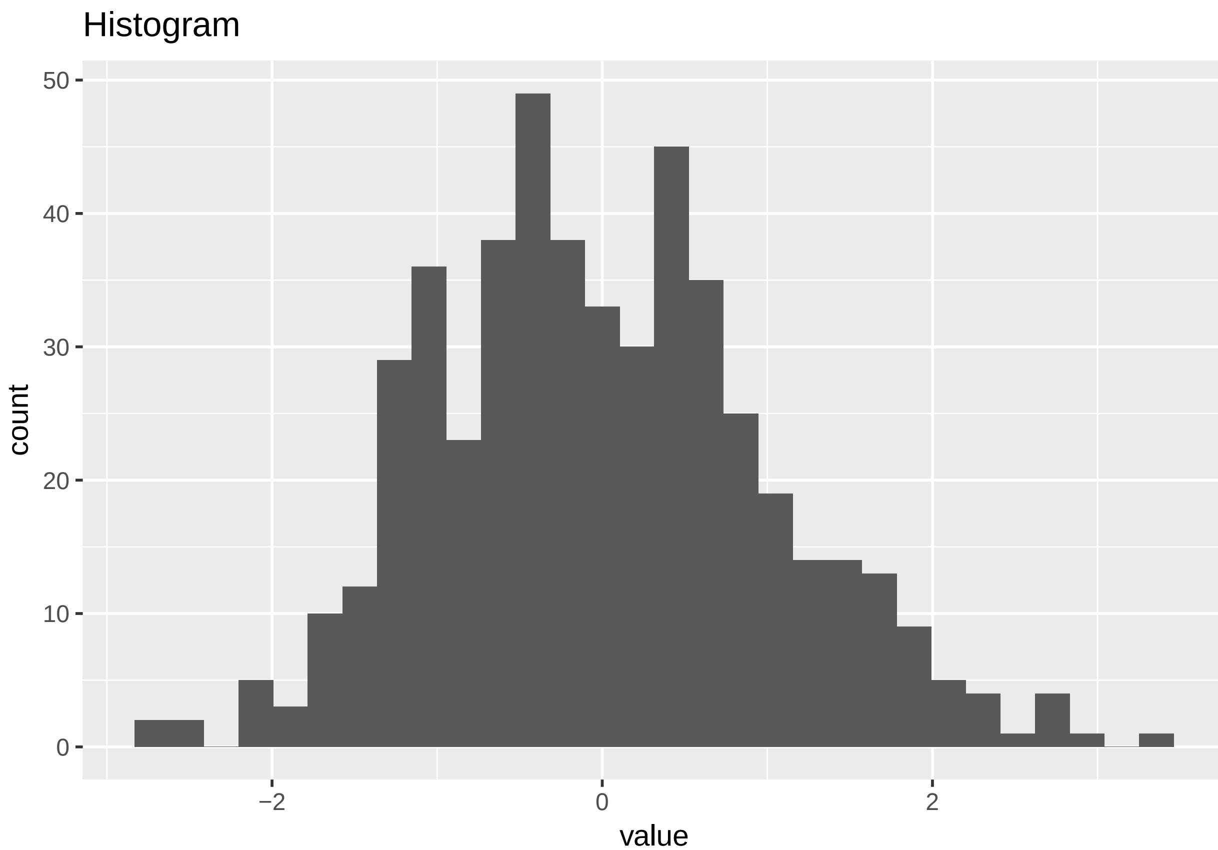

```
ggplot(normal_random_numbers, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

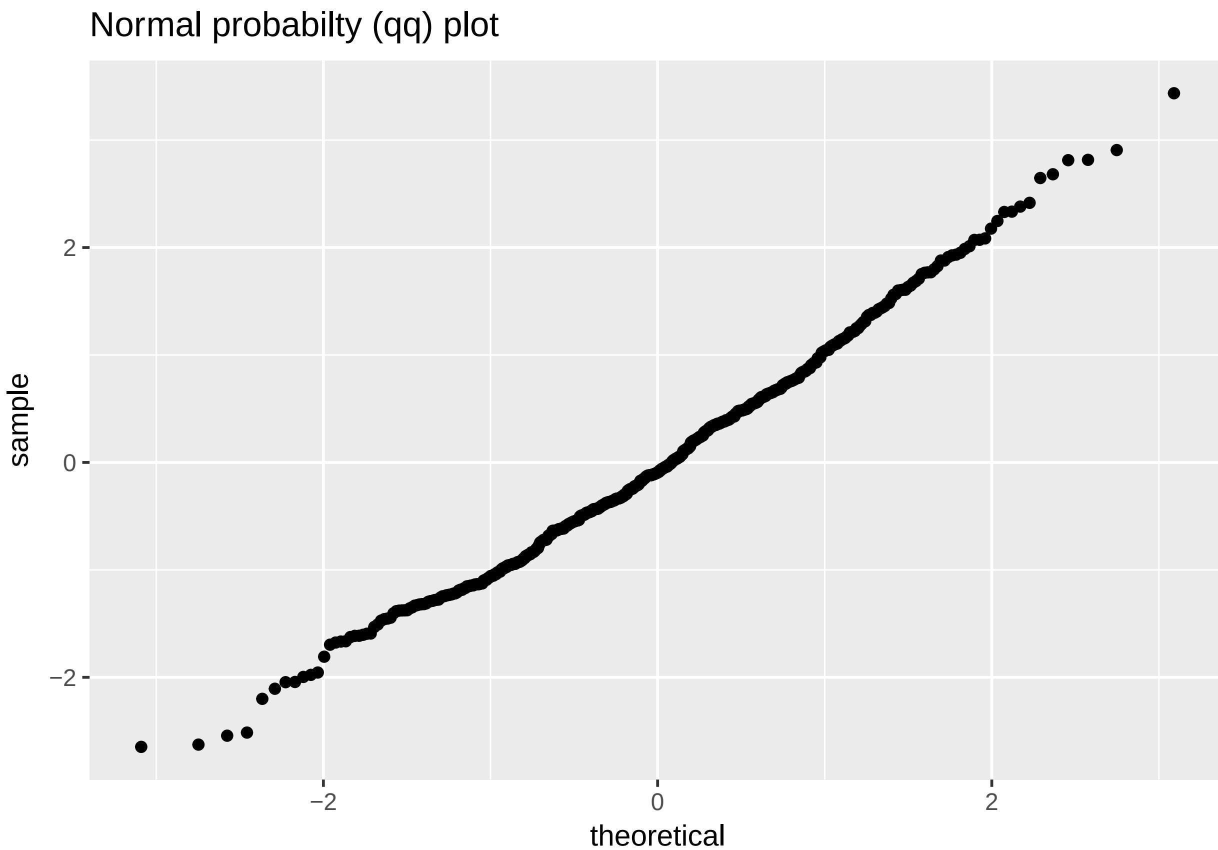



```
ggplot(normal_random_numbers, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(normal_random_numbers, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.2 Uniform random numbers

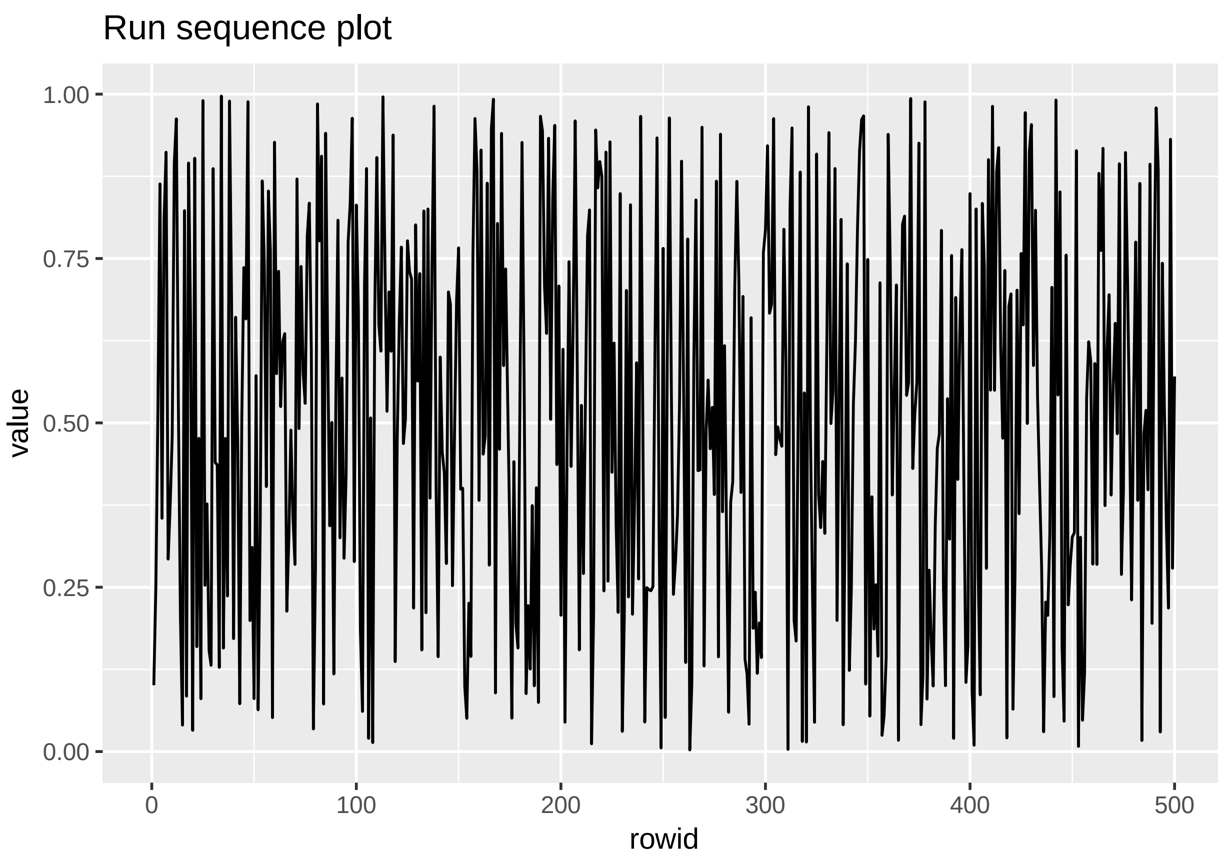
Uniform Random Numbers

```
uniform_random_numbers <- scan("NIST data/RANDU.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

uniform_random_numbers
```

```
## # A tibble: 500 x 2
##    rowid value
##    <int> <dbl>
##  1     1 0.101
##  2     2 0.253
##  3     3 0.520
##  4     4 0.863
##  5     5 0.355
##  6     6 0.810
##  7     7 0.912
##  8     8 0.293
##  9     9 0.375
## 10    10 0.481
## # ... with 490 more rows
```

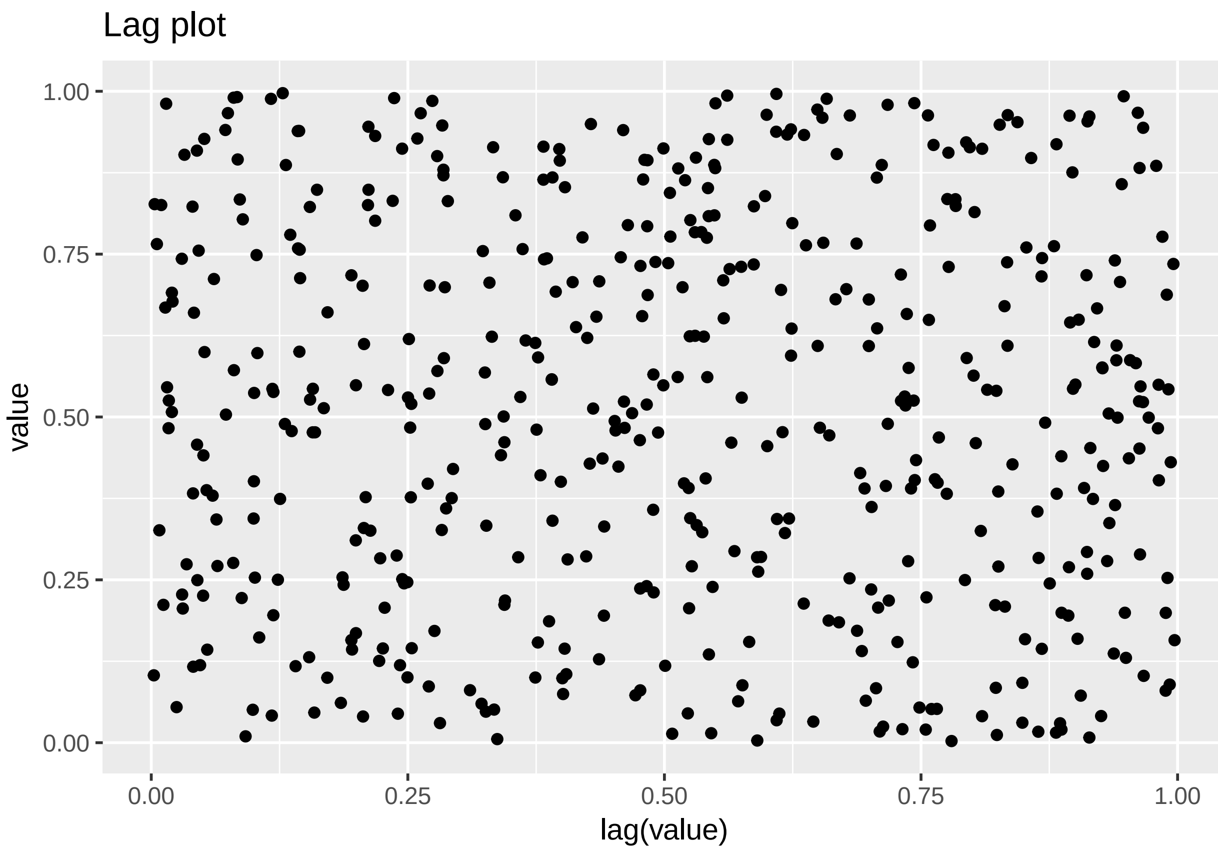
```
ggplot(uniform_random_numbers, aes(rowid, value)) +
  geom_line() +
```

```
  labs(title = "Run sequence plot")
```



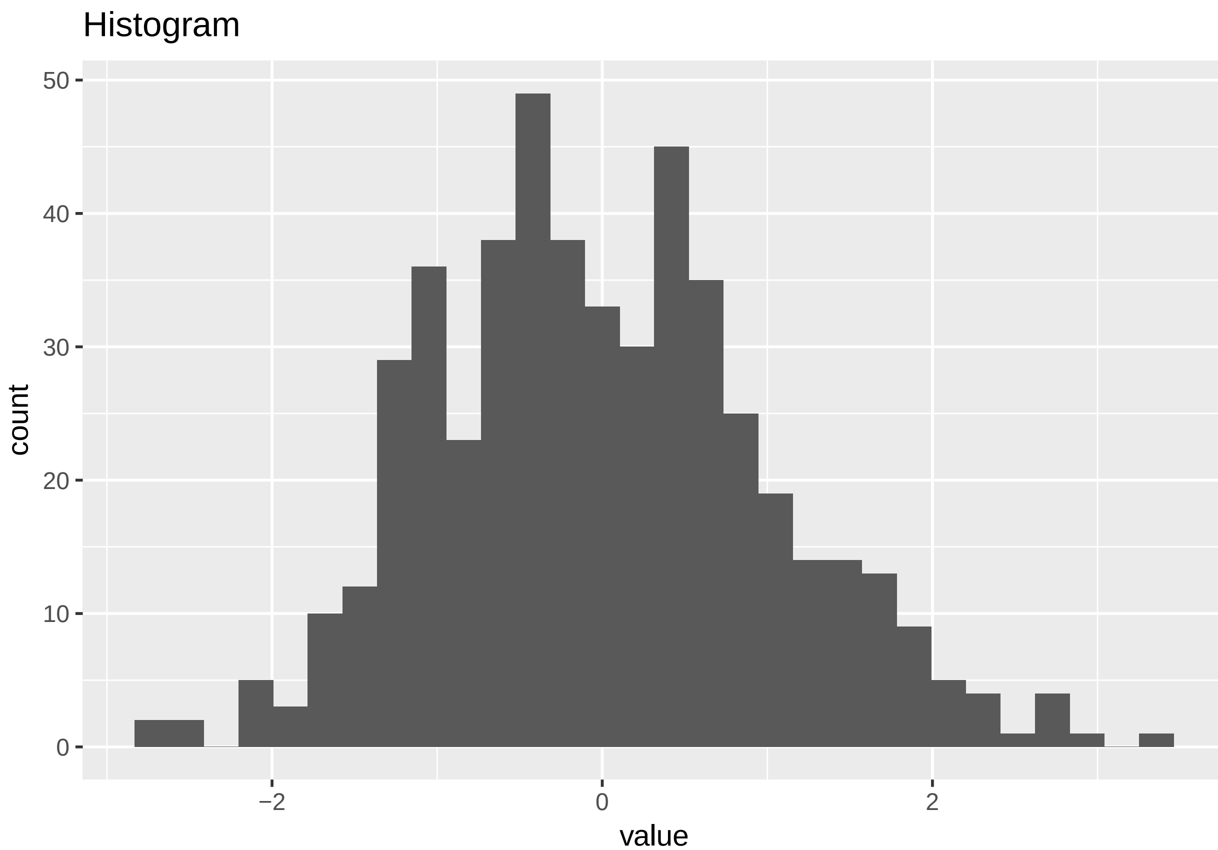

```
ggplot(uniform_random_numbers, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

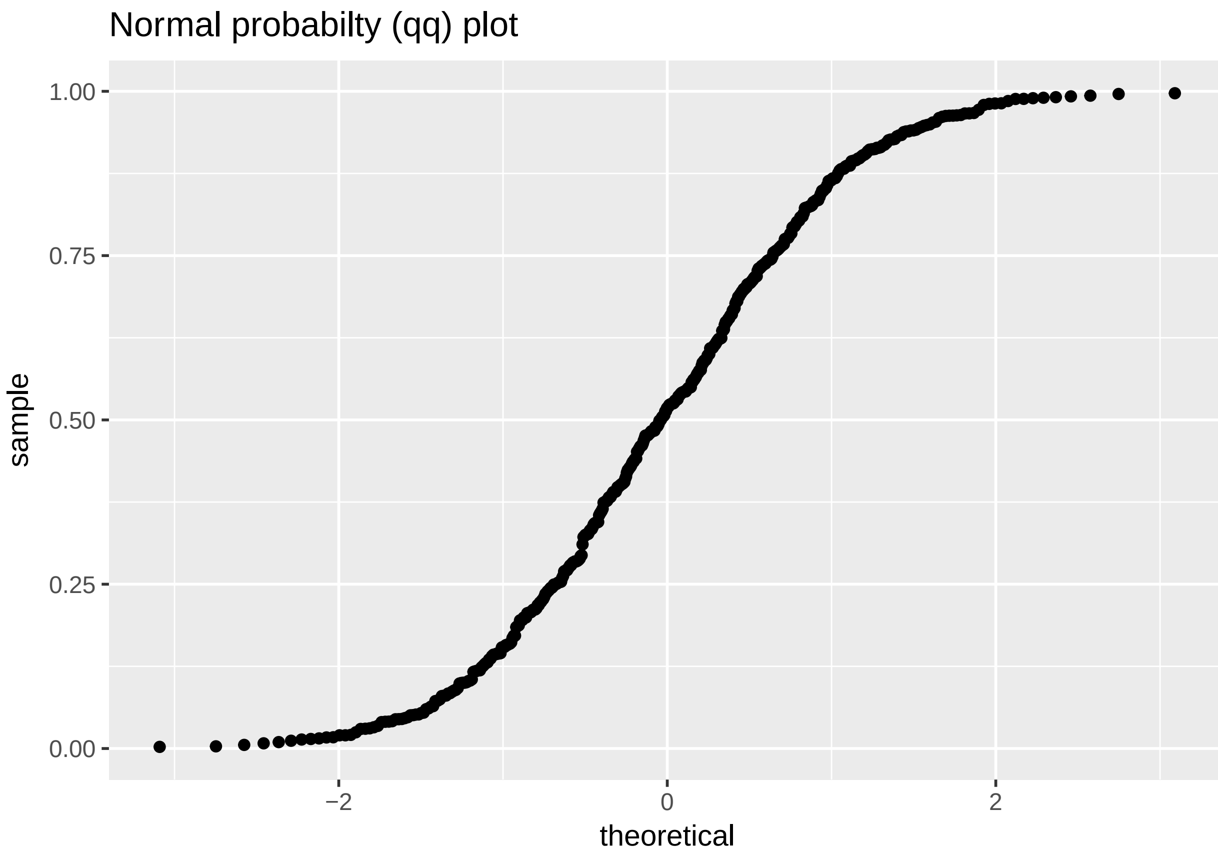



```r
ggplot(normal_random_numbers, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(uniform_random_numbers, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.3 Random walk

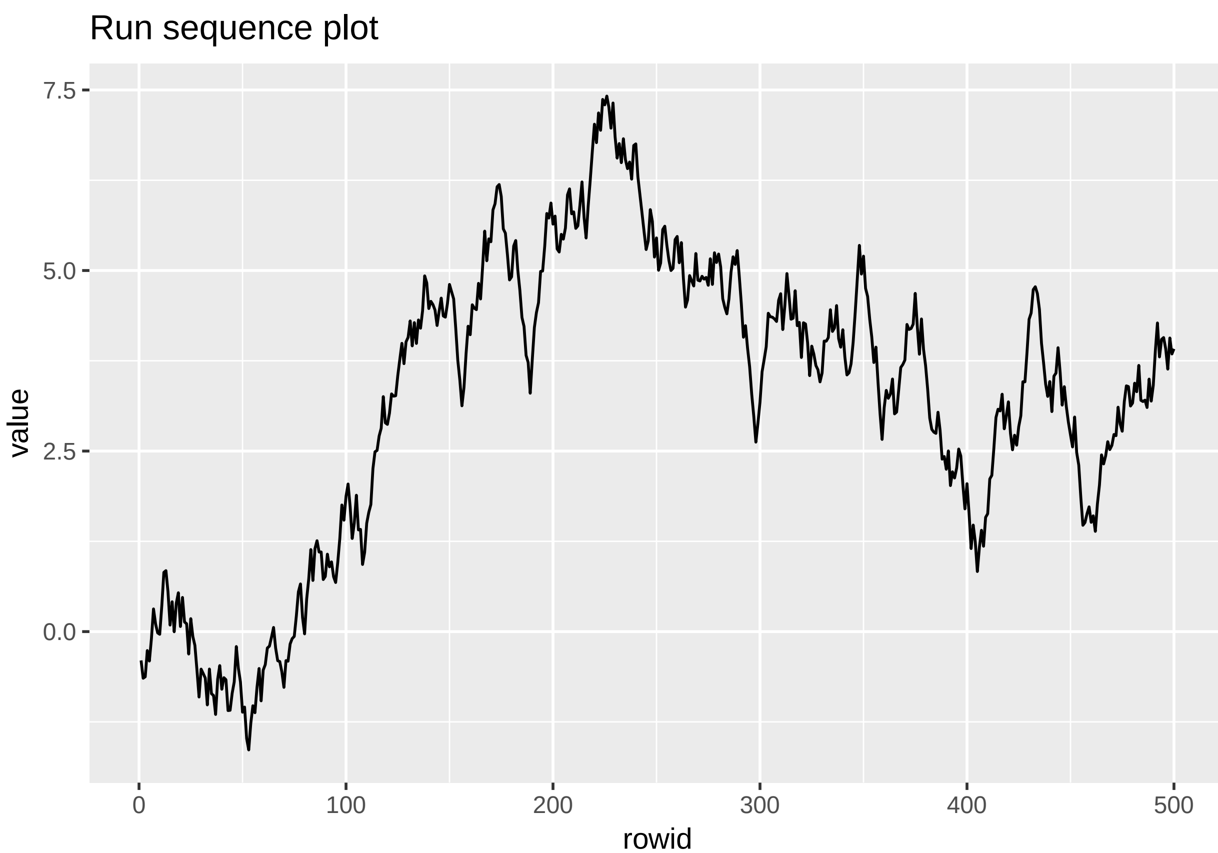
Random Walk

```
random_walk <- scan("NIST data/RANDWALK.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

random_walk
```

```
## # A tibble: 500 x 2
##    rowid   value
##    <int>   <dbl>
##  1     1 -0.399
##  2     2 -0.646
##  3     3 -0.626
##  4     4 -0.262
##  5     5 -0.407
##  6     6 -0.0976
##  7     7  0.314
##  8     8  0.107
##  9     9 -0.0177
## 10    10 -0.0371
## # ... with 490 more rows
```

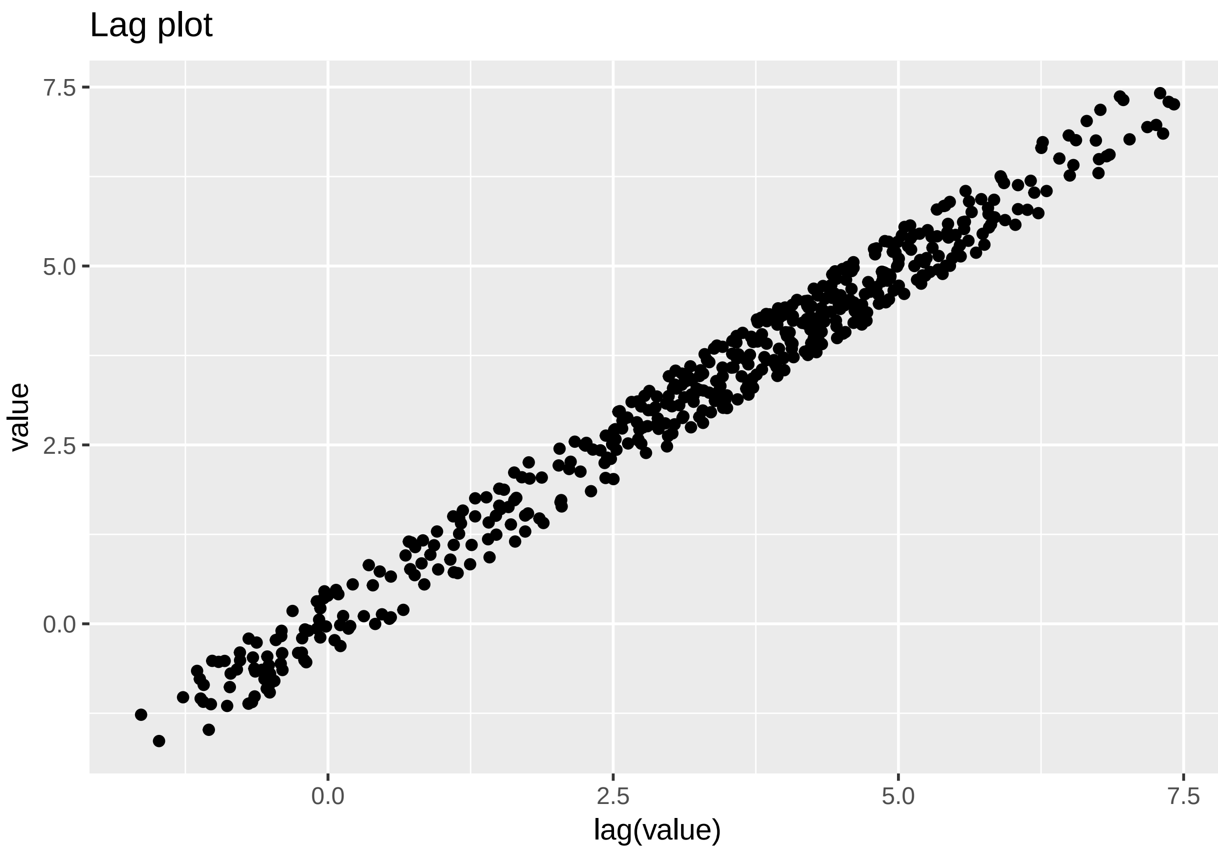
```
ggplot(random_walk, aes(rowid, value)) +
  geom_line() +
```

```
  labs(title = "Run sequence plot")
```



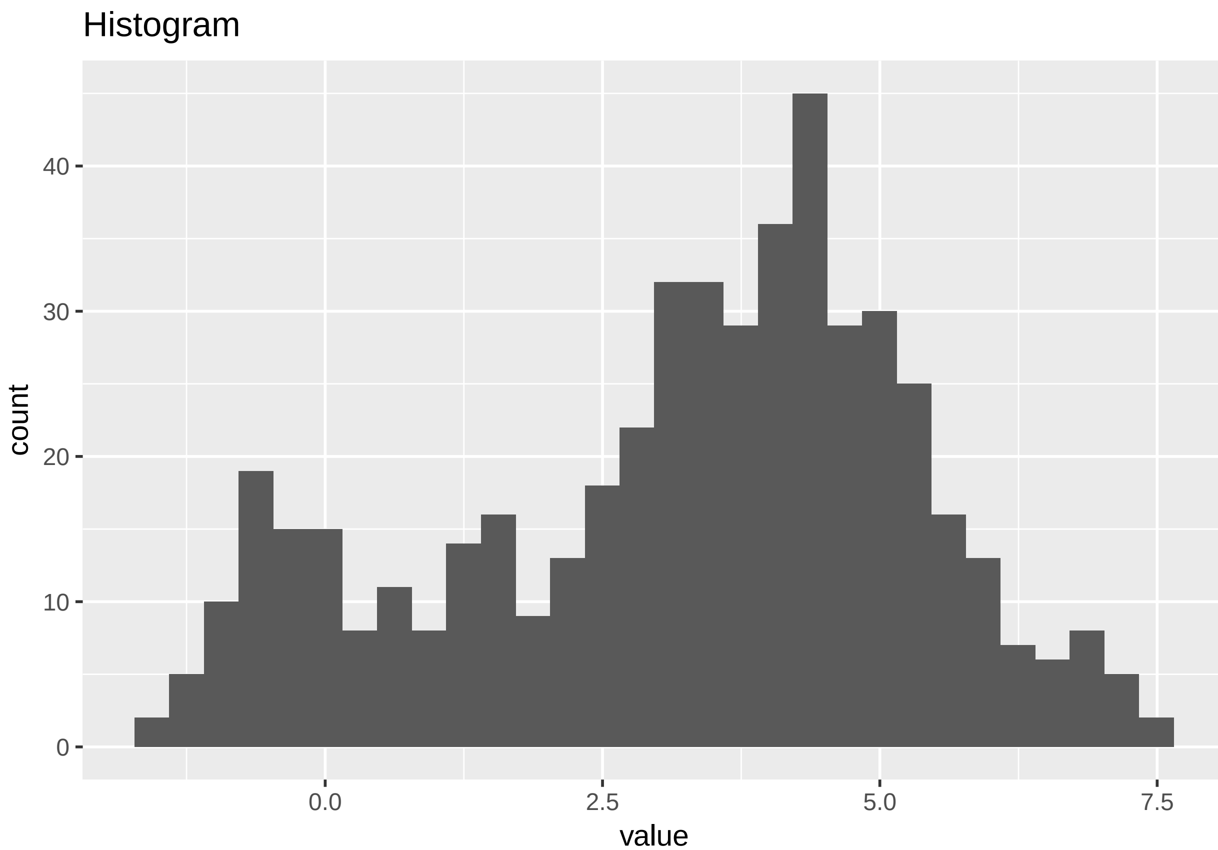

```
ggplot(random_walk, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

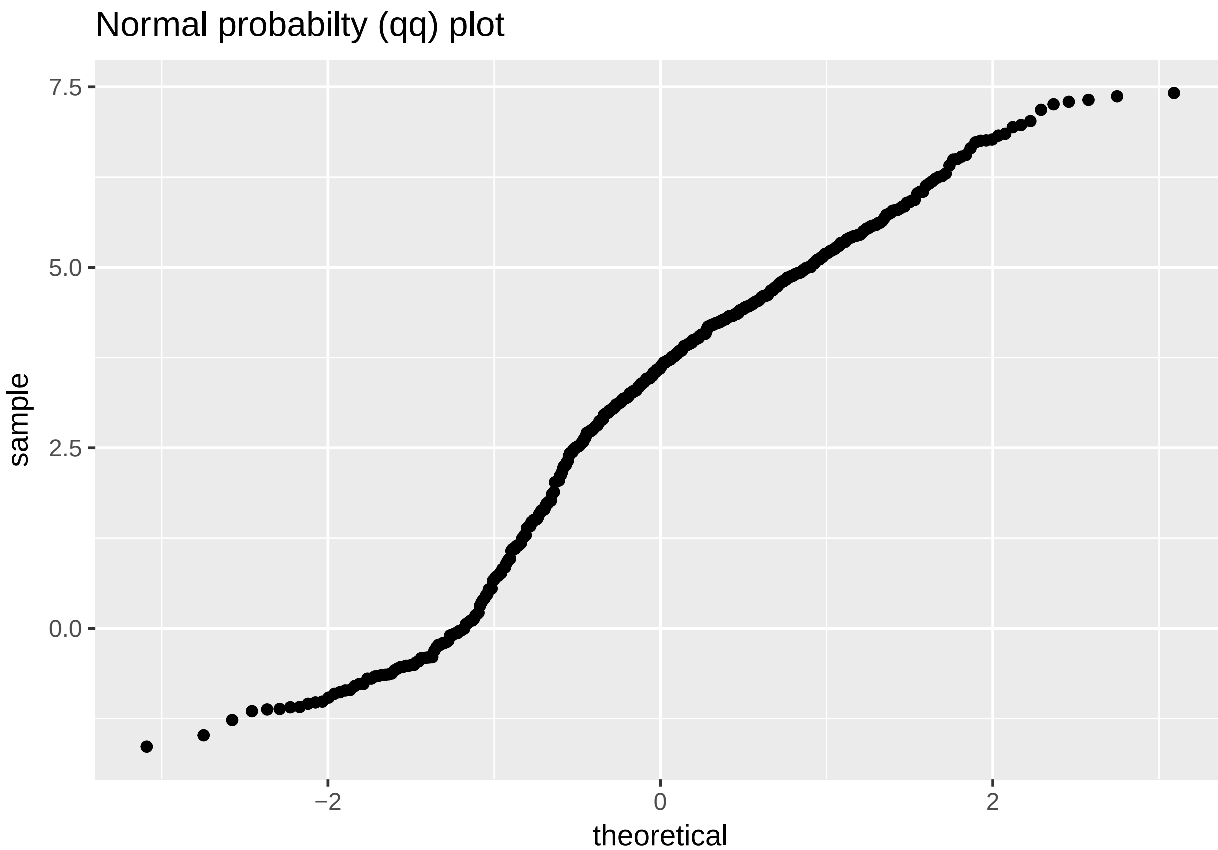



```
ggplot(random_walk, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggplot(random_walk, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.4 Beam deflections

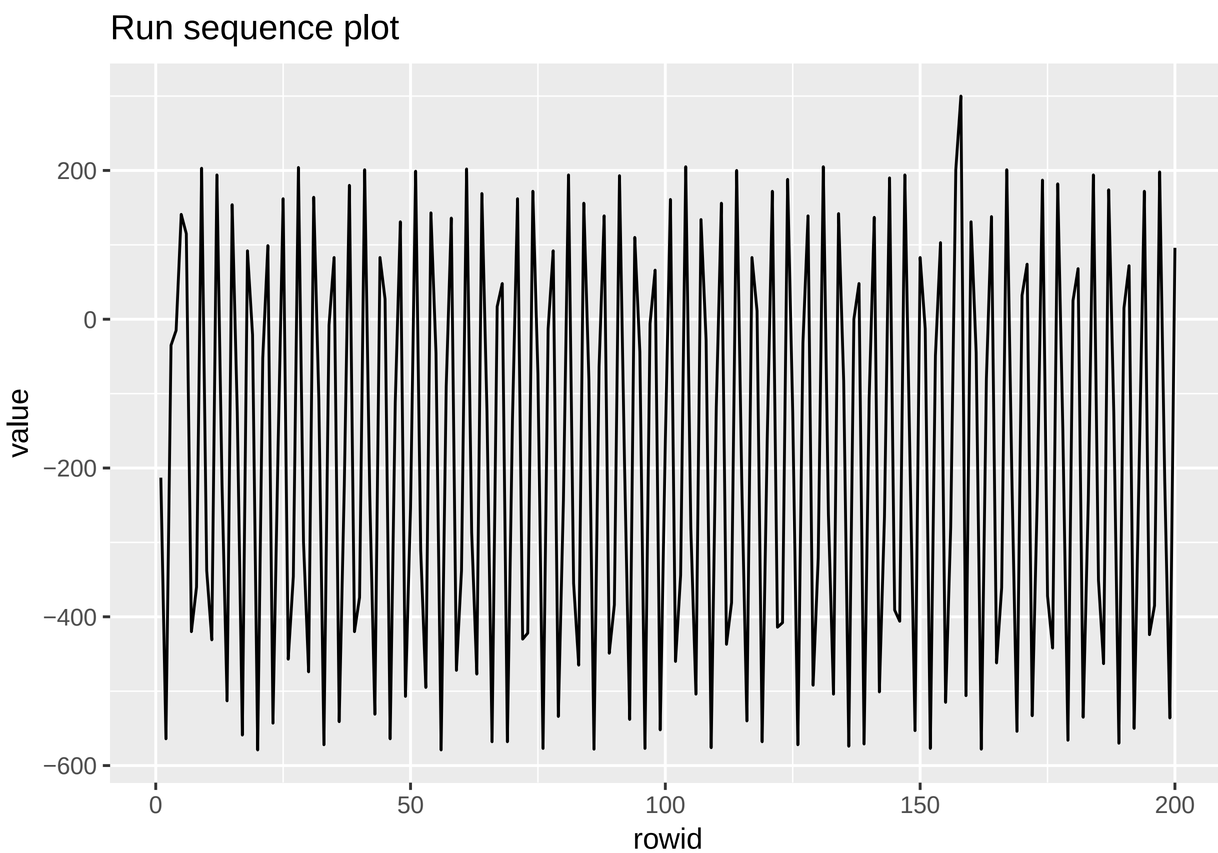
Beam Deflections

```
beam_deflections <- scan("NIST data/LEW.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

beam_deflections
```

```
## # A tibble: 200 x 2
##    rowid value
##    <int> <dbl>
##  1     1 -213.
##  2     2 -564.
##  3     3  -35.
##  4     4  -15.
##  5     5  141.
##  6     6  115.
##  7     7 -420.
##  8     8 -360.
##  9     9  203.
## 10    10 -338.
## # ... with 190 more rows
```

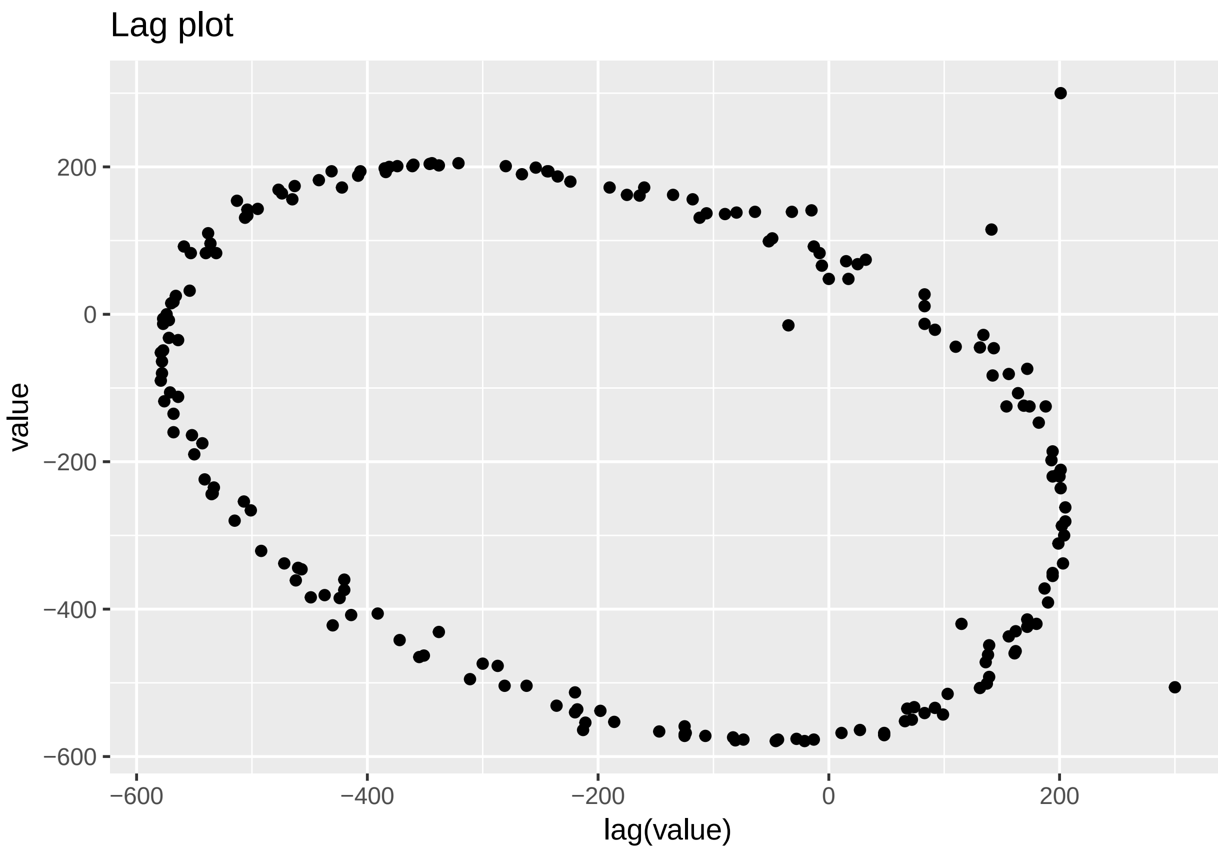
```
ggplot(beam_deflections, aes(rowid, value)) +
  geom_line() +
```

```
  labs(title = "Run sequence plot")
```



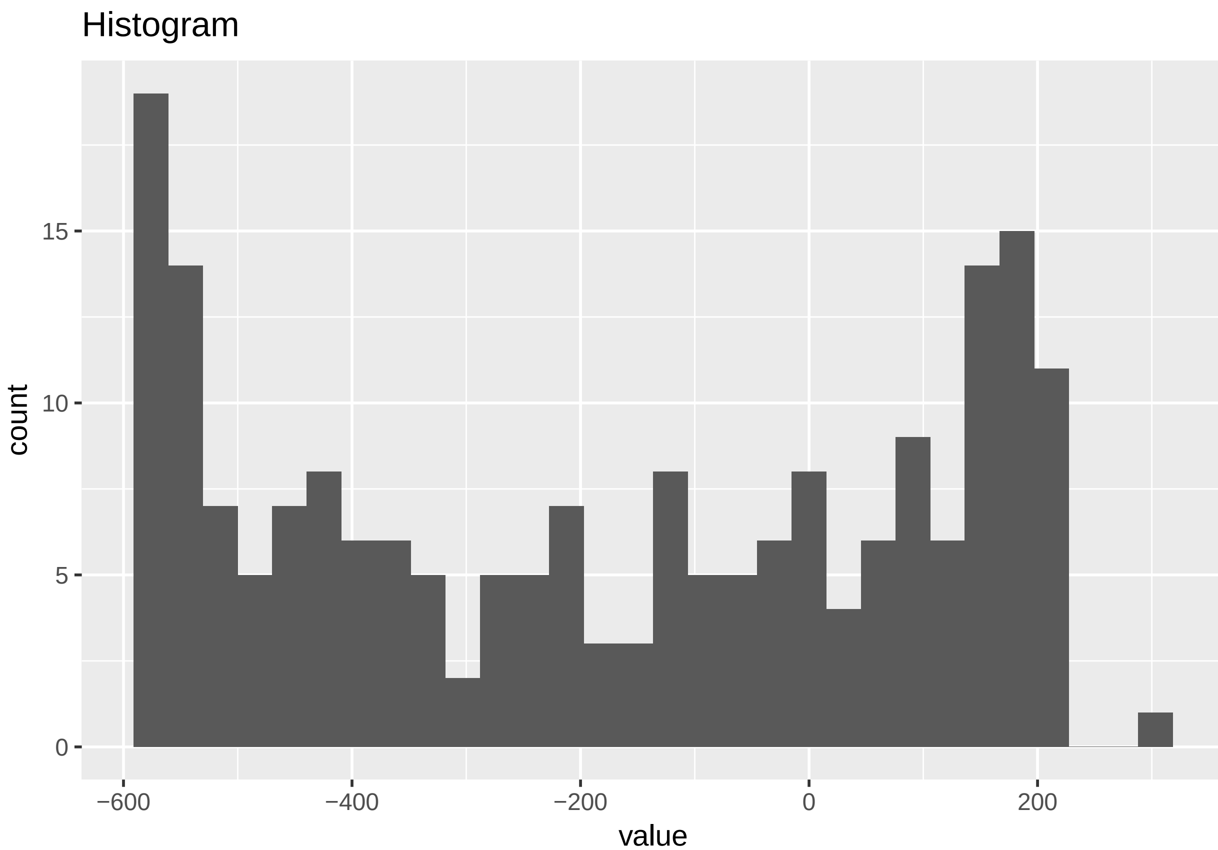

```
ggplot(beam_deflections, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

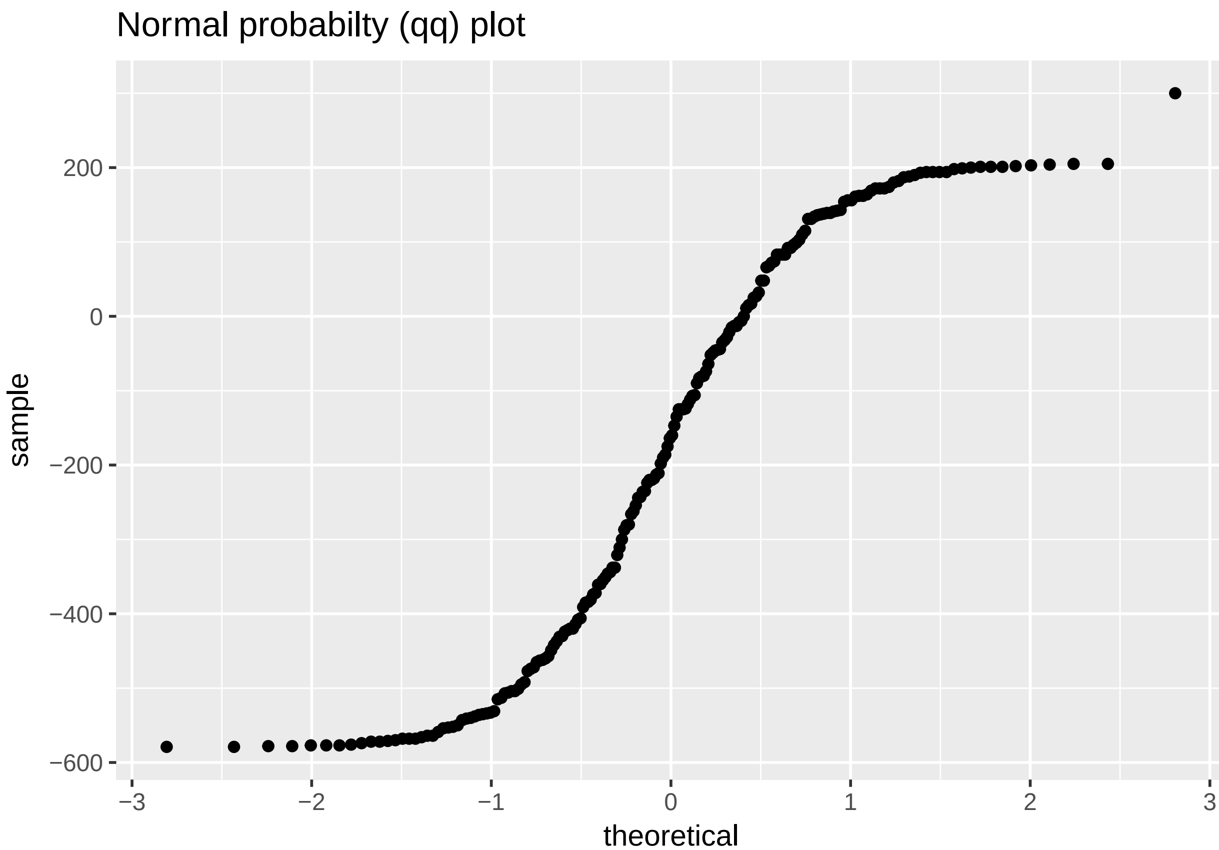



```
ggplot(beam_deflections, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggplot(beam_deflections, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.5 Filter transmitance

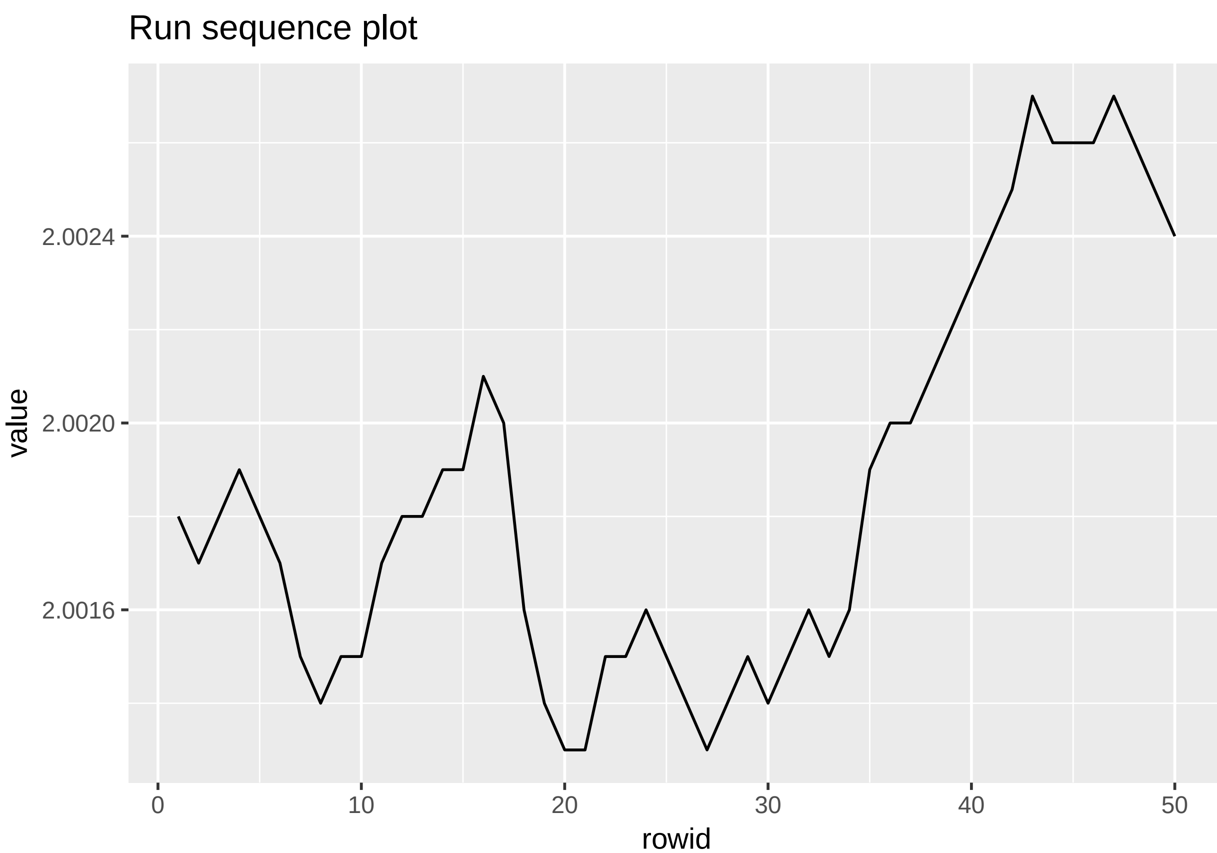
Filter Transmitance

```
filter_transmitance <- scan("NIST data/MAVRO.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

filter_transmitance
```

```
## # A tibble: 50 x 2
##    rowid value
##    <int> <dbl>
##  1     1  2.00
##  2     2  2.00
##  3     3  2.00
##  4     4  2.00
##  5     5  2.00
##  6     6  2.00
##  7     7  2.00
##  8     8  2.00
##  9     9  2.00
## 10    10  2.00
## # ... with 40 more rows
```

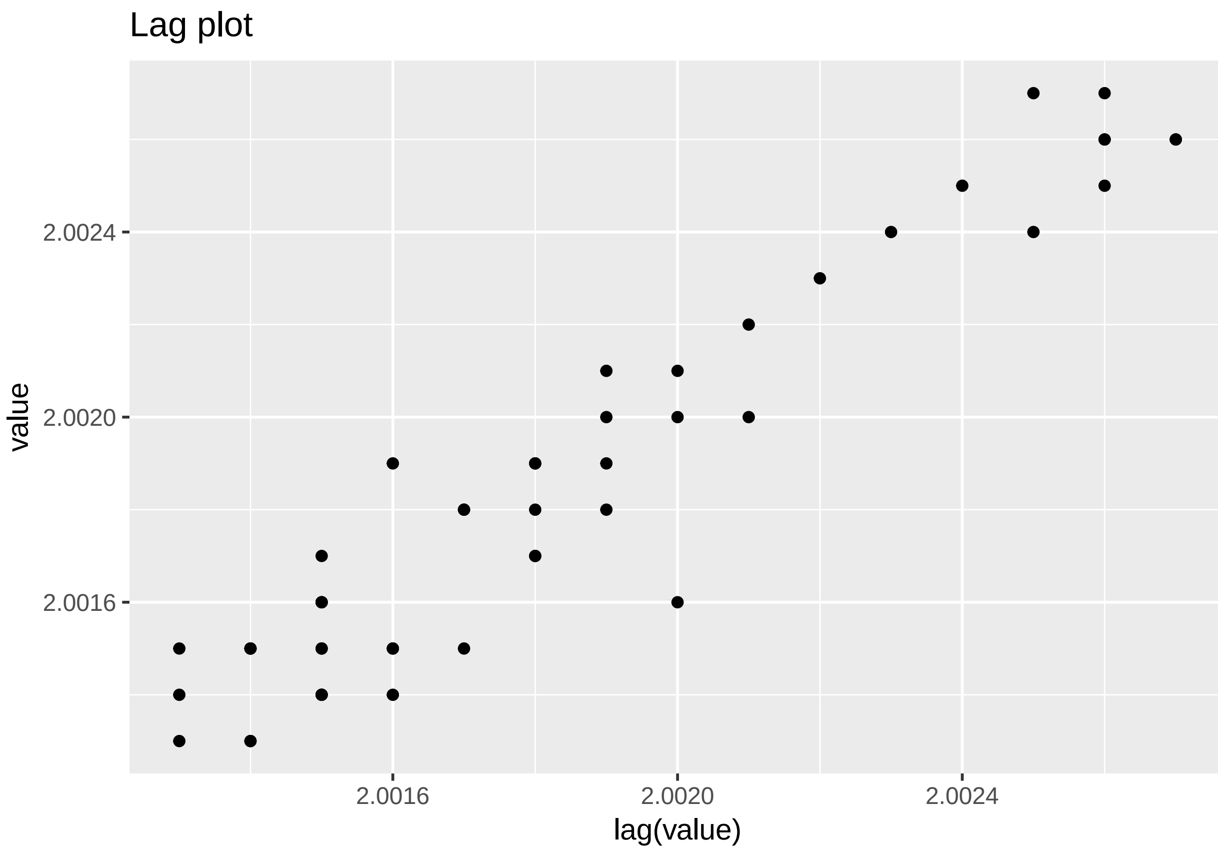
```
ggplot(filter_transmitance, aes(rowid, value)) +
  geom_line() +
```

```
  labs(title = "Run sequence plot")
```



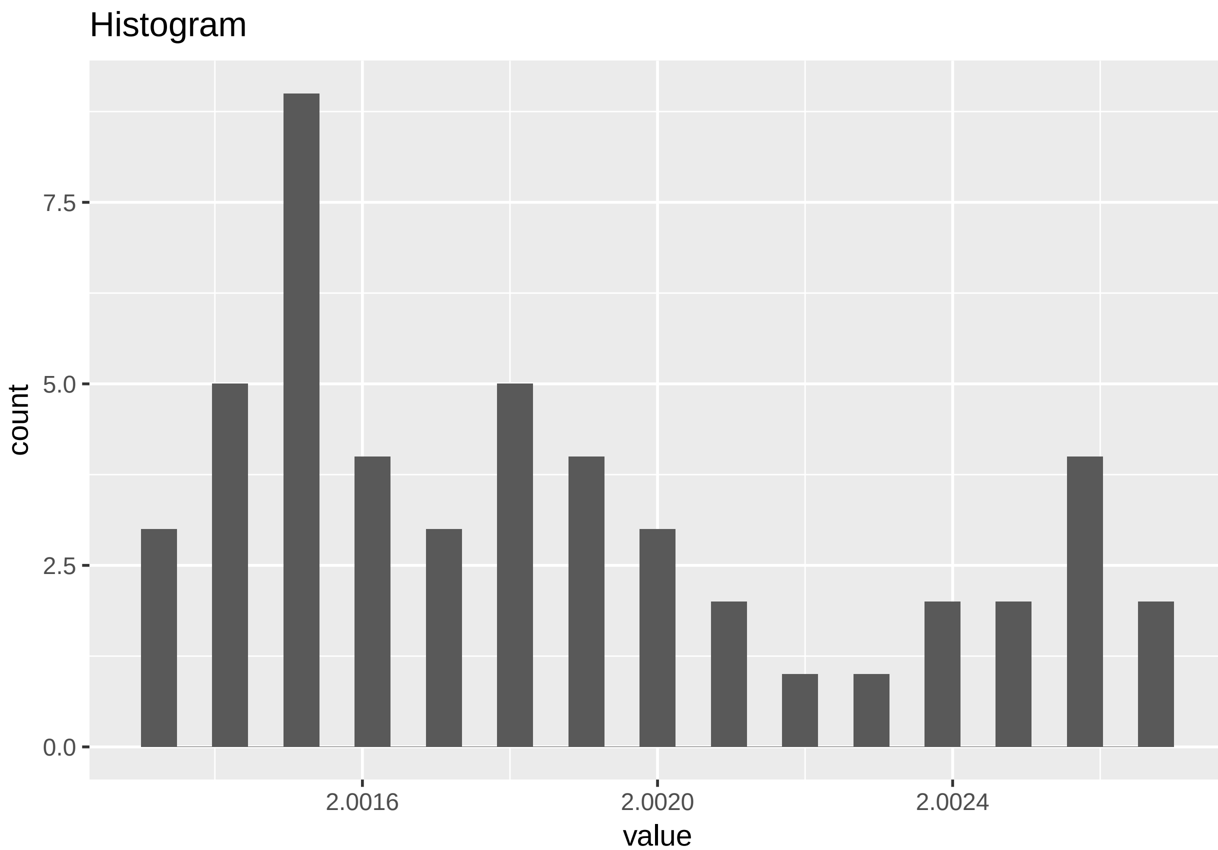

```
ggplot(filter_transmitance, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

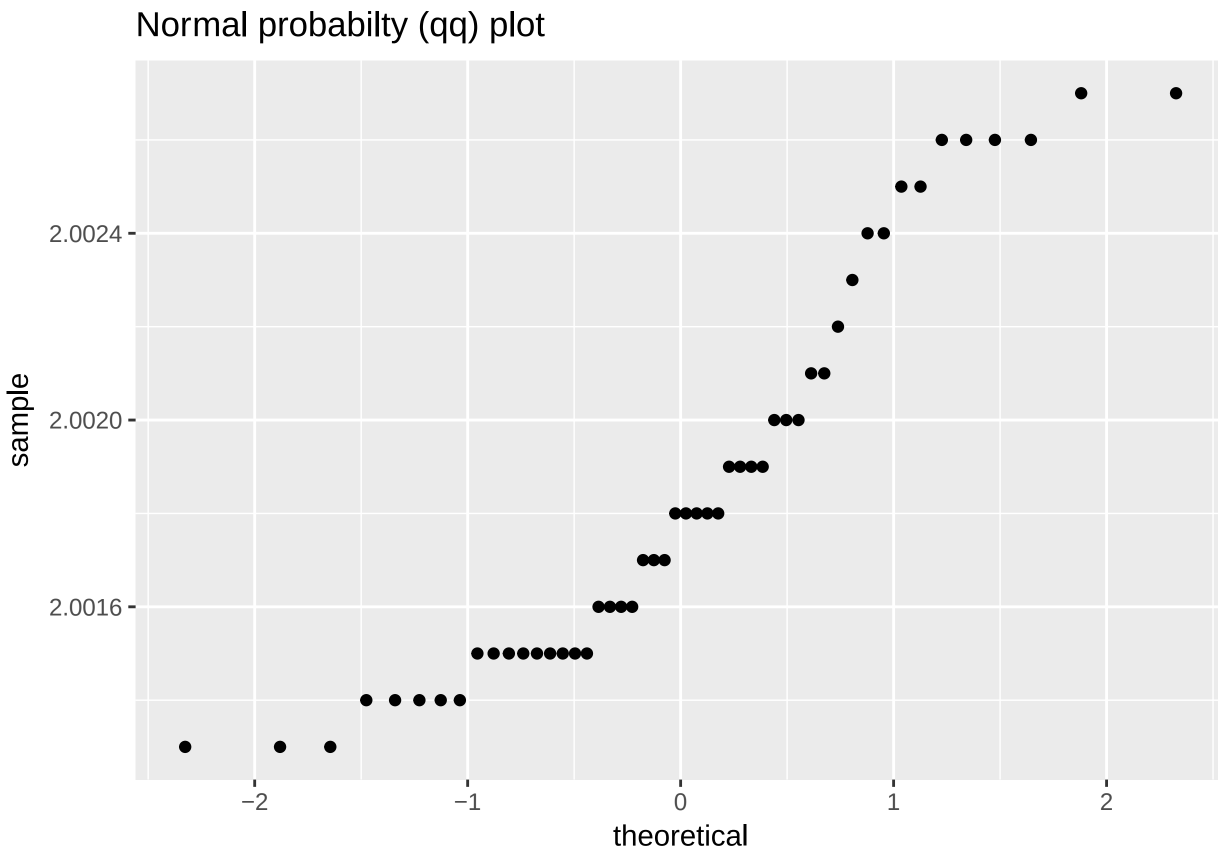



```r
ggplot(filter_transmitance, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(filter_transmitance, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.6 Standard resistor

Standard Resistor

```
standard_resistor <- read_table2("NIST data/DZIUBA1.DAT", skip = 25, col_names = FALSE) %>%
  rowid_to_column() %>%
  rename(month = X1, day = X2, year = X3, resistance = X4)
```

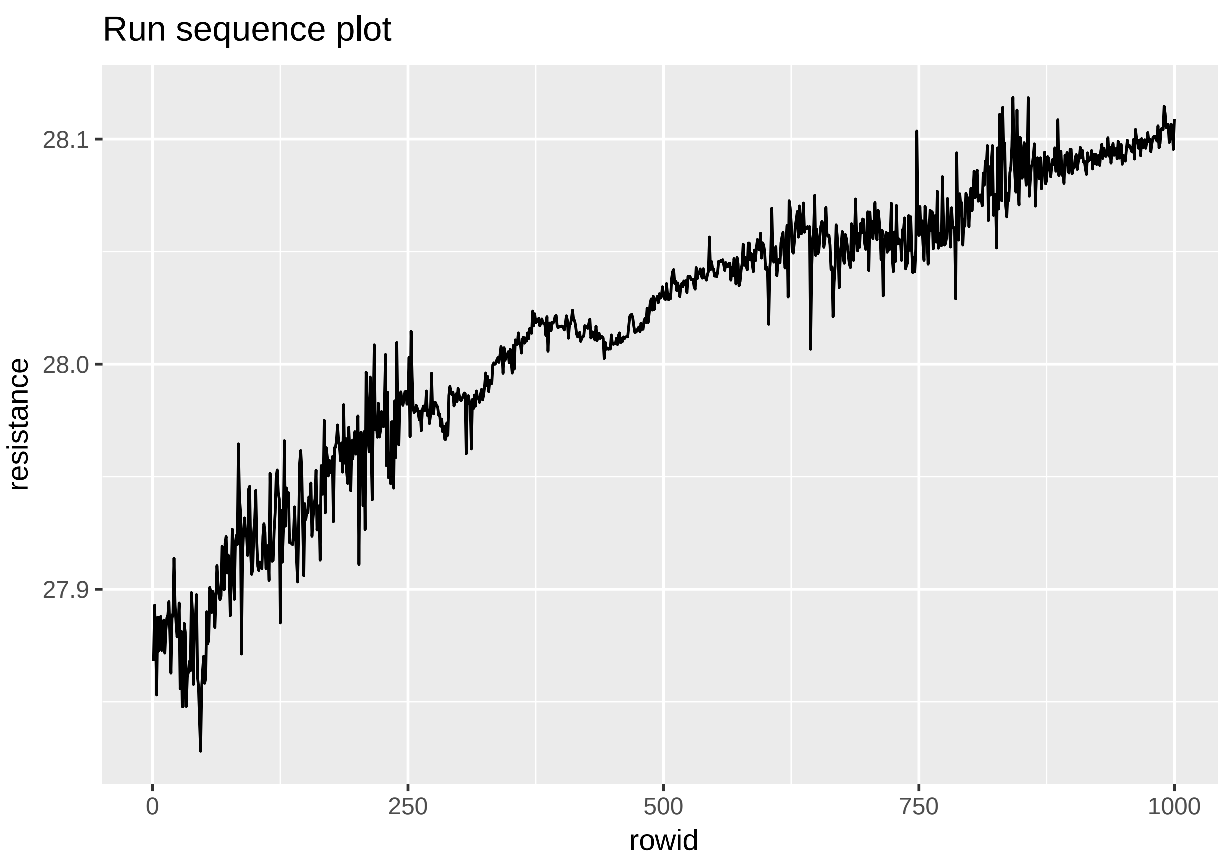
```
## Parsed with column specification:
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_integer(),
##   X4 = col_double()
## )
```

```
standard_resistor
```

```
## # A tibble: 1,000 x 5
##    rowid month day    year resistance
##    <int> <chr> <chr> <int>      <dbl>
##  1     1 2     5        80       27.9
##  2     2 2     12       80       27.9
##  3     3 2     13       80       27.9
##  4     4 2     14       80       27.9
##  5     5 2     28       80       27.9
##  6     6 2     28       80       27.9
```

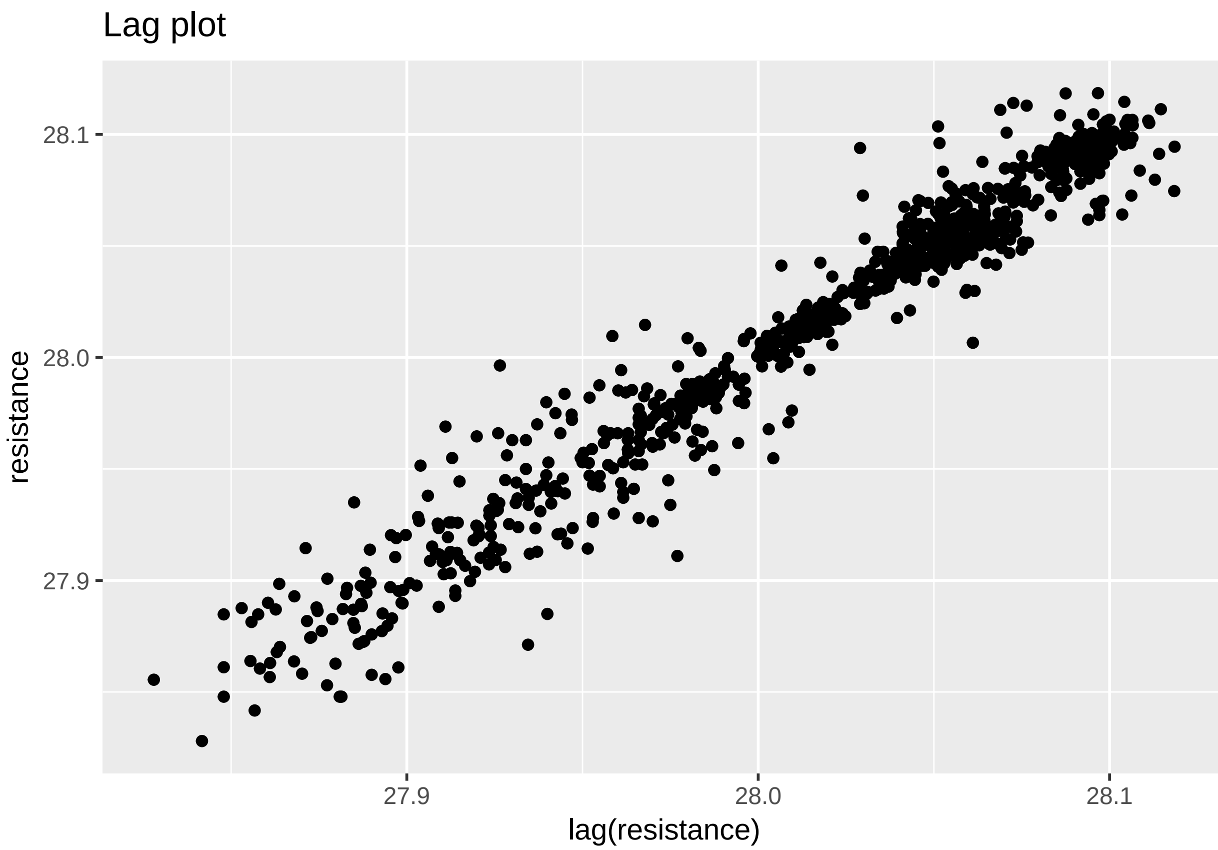
```
##  7      7 3     21       80       27.9
##  8      8 3     24       80       27.9
##  9      9 4     3        80       27.9
## 10     10 4     3        80       27.9
## # ... with 990 more rows
```

```
ggplot(standard_resistor, aes(rowid, resistance)) +
  geom_line() +
  labs(title = "Run sequence plot")
```



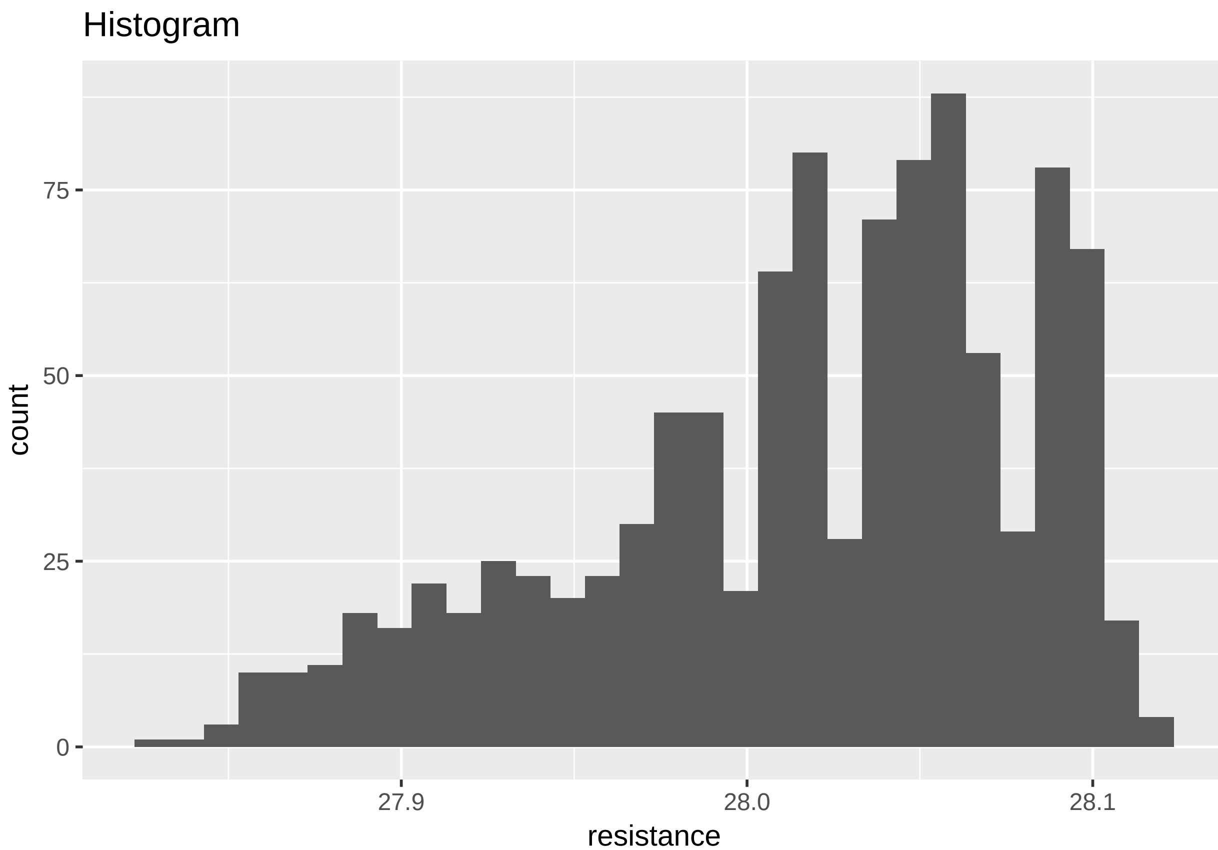

```
ggplot(standard_resistor, aes(lag(resistance), resistance)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

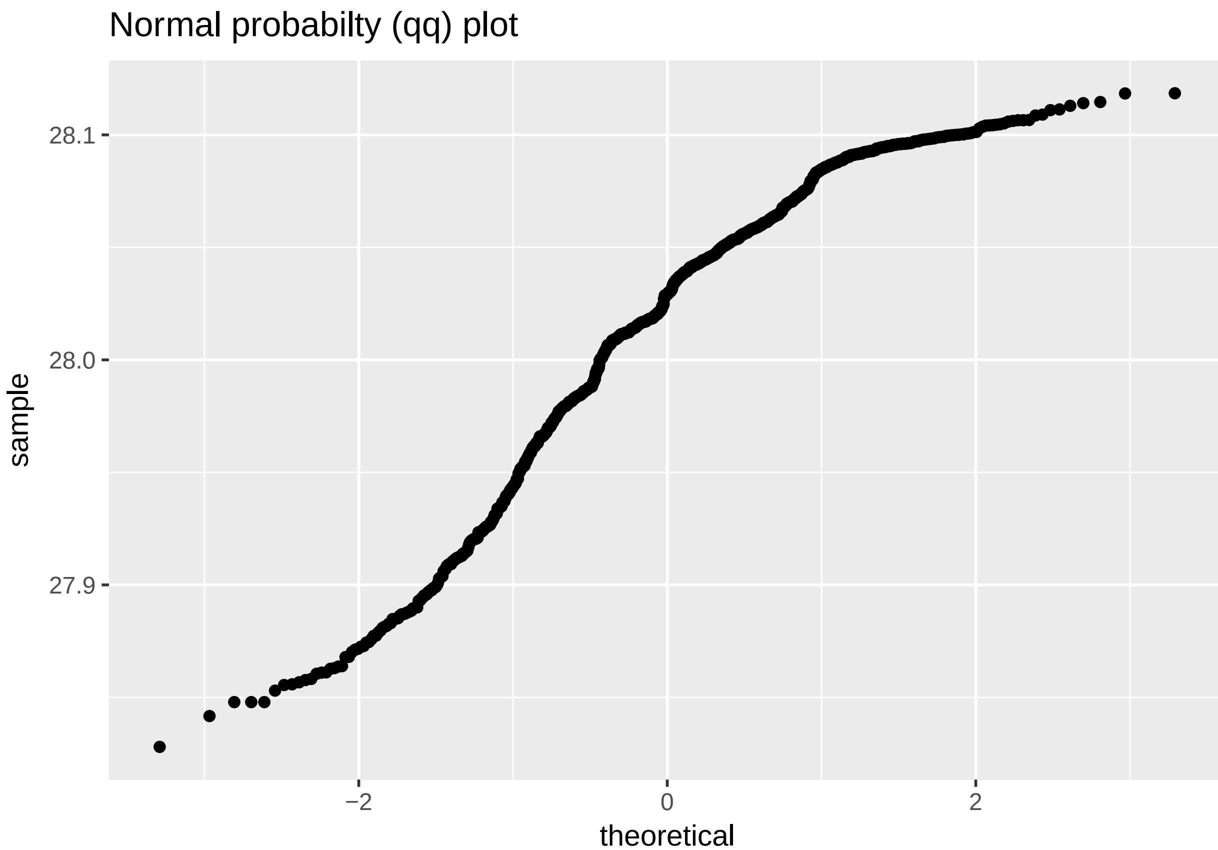



```
ggplot(standard_resistor, aes(resistance)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(standard_resistor, aes(sample = resistance)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.7 Heat flow meter 1

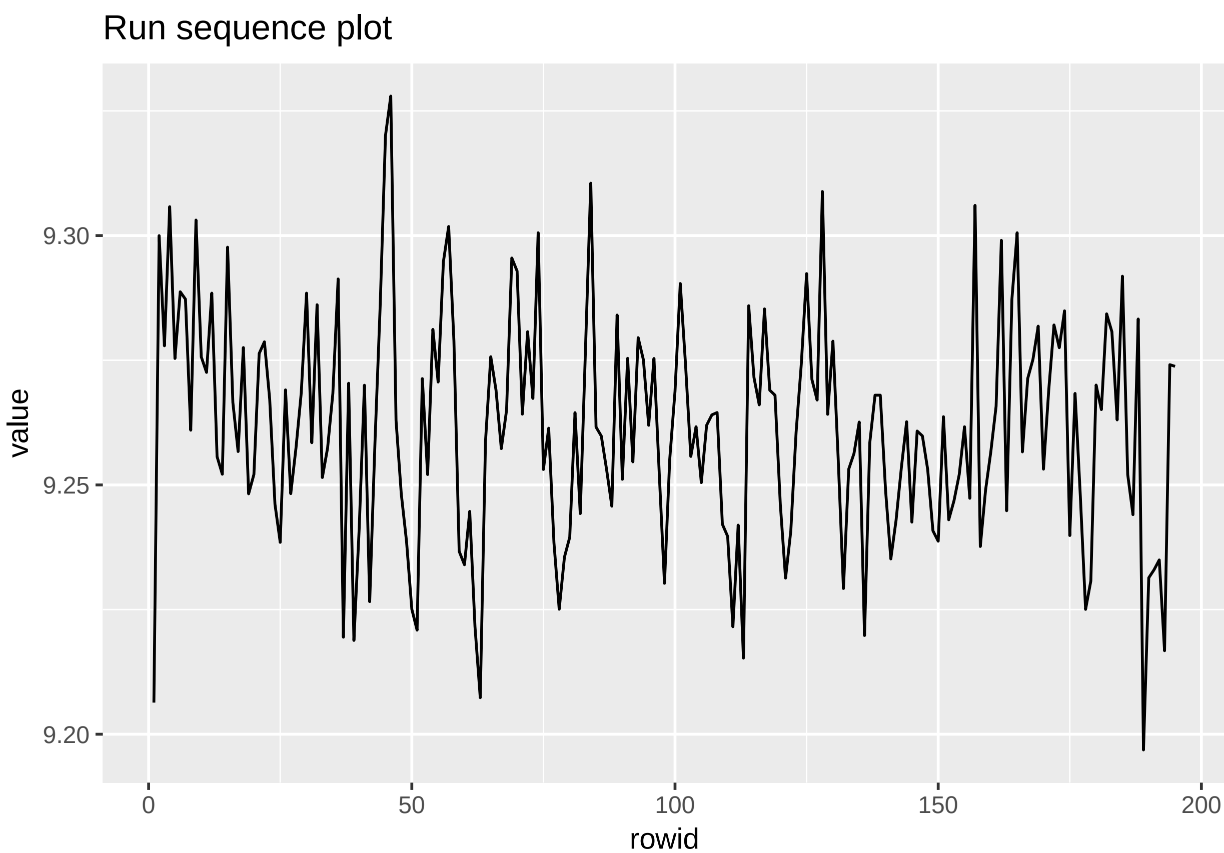
Heat Flow Meter 1

```
heat_flow_meter1 <- scan("NIST data/ZARR13.DAT", skip = 25) %>%
  as.tibble() %>%
  rowid_to_column()

heat_flow_meter1
```

```
## # A tibble: 195 x 2
##    rowid value
##    <int> <dbl>
##  1     1  9.21
##  2     2  9.30
##  3     3  9.28
##  4     4  9.31
##  5     5  9.28
##  6     6  9.29
##  7     7  9.29
##  8     8  9.26
##  9     9  9.30
## 10    10  9.28
## # ... with 185 more rows
```

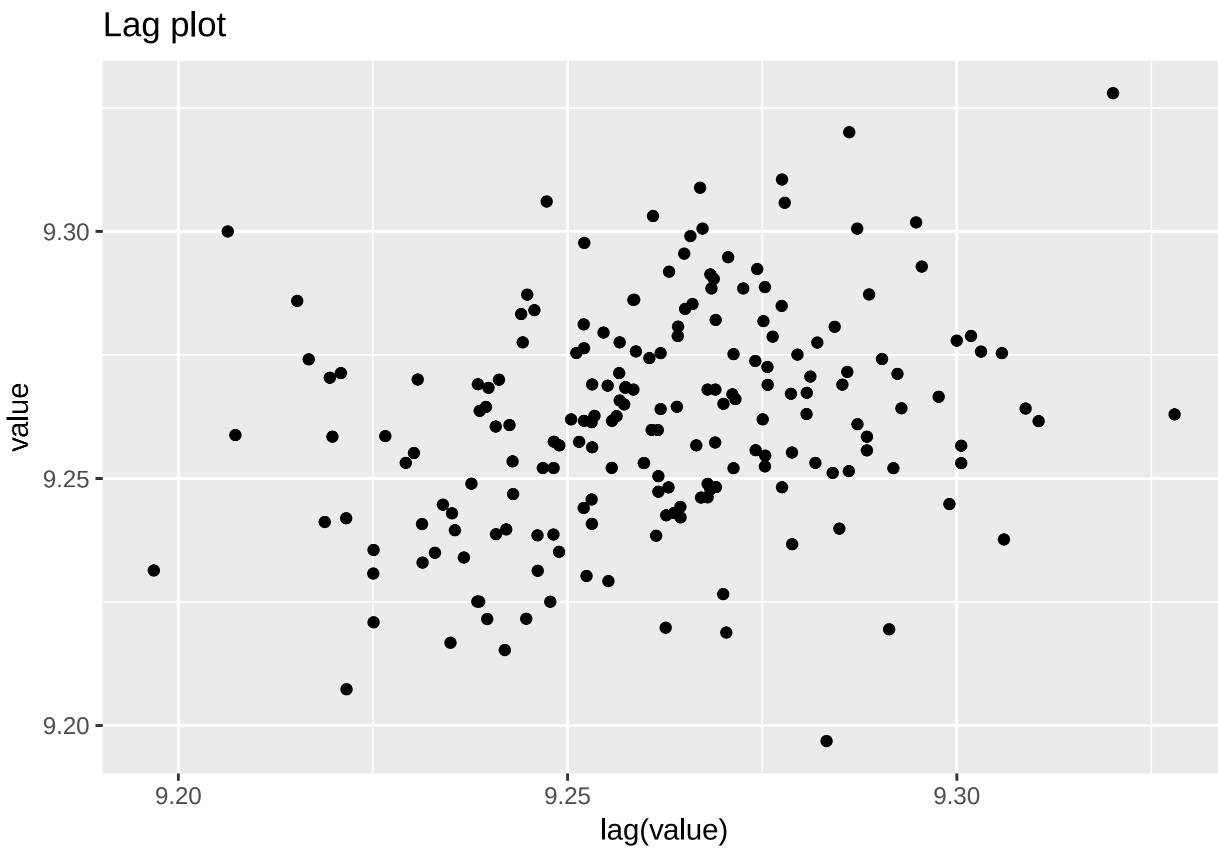
```
ggplot(heat_flow_meter1, aes(rowid, value)) +
  geom_line() +
```

```
  labs(title = "Run sequence plot")
```



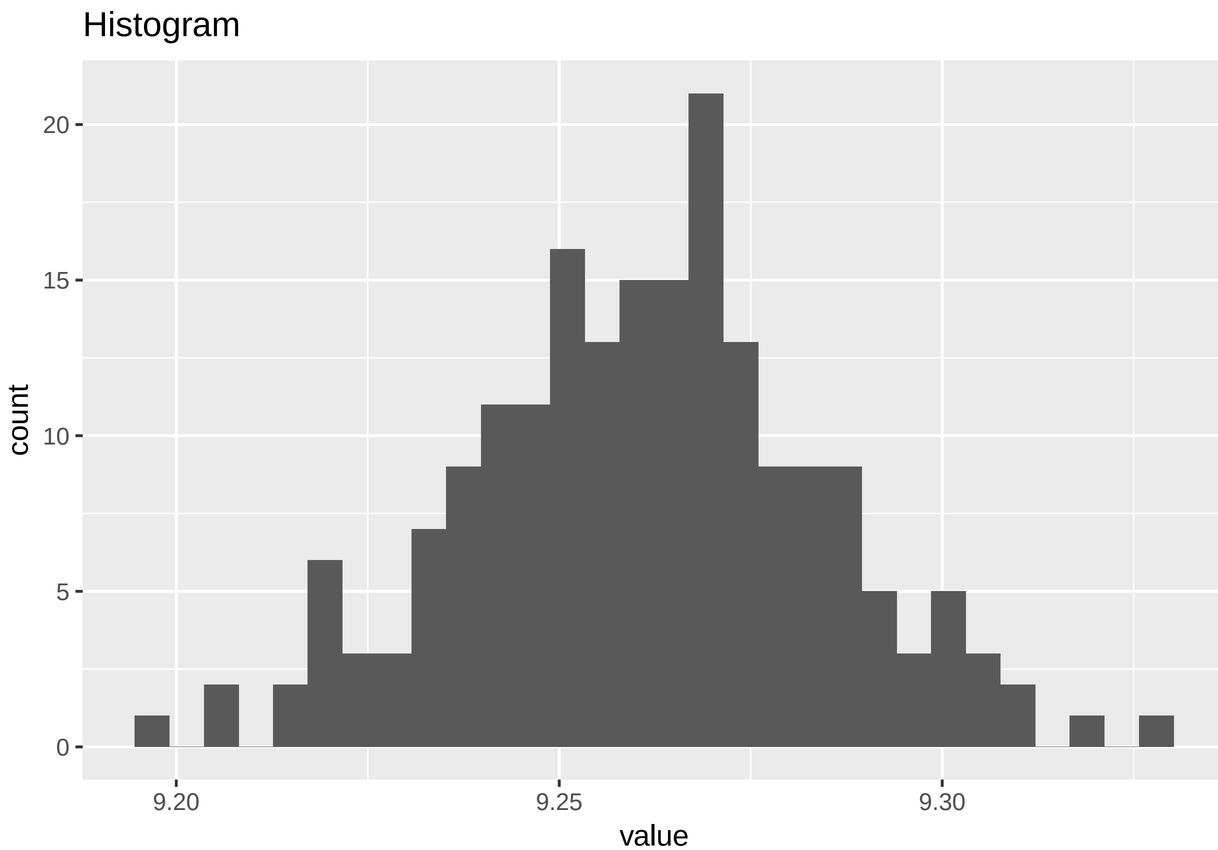

```
ggplot(heat_flow_meter1, aes(lag(value), value)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

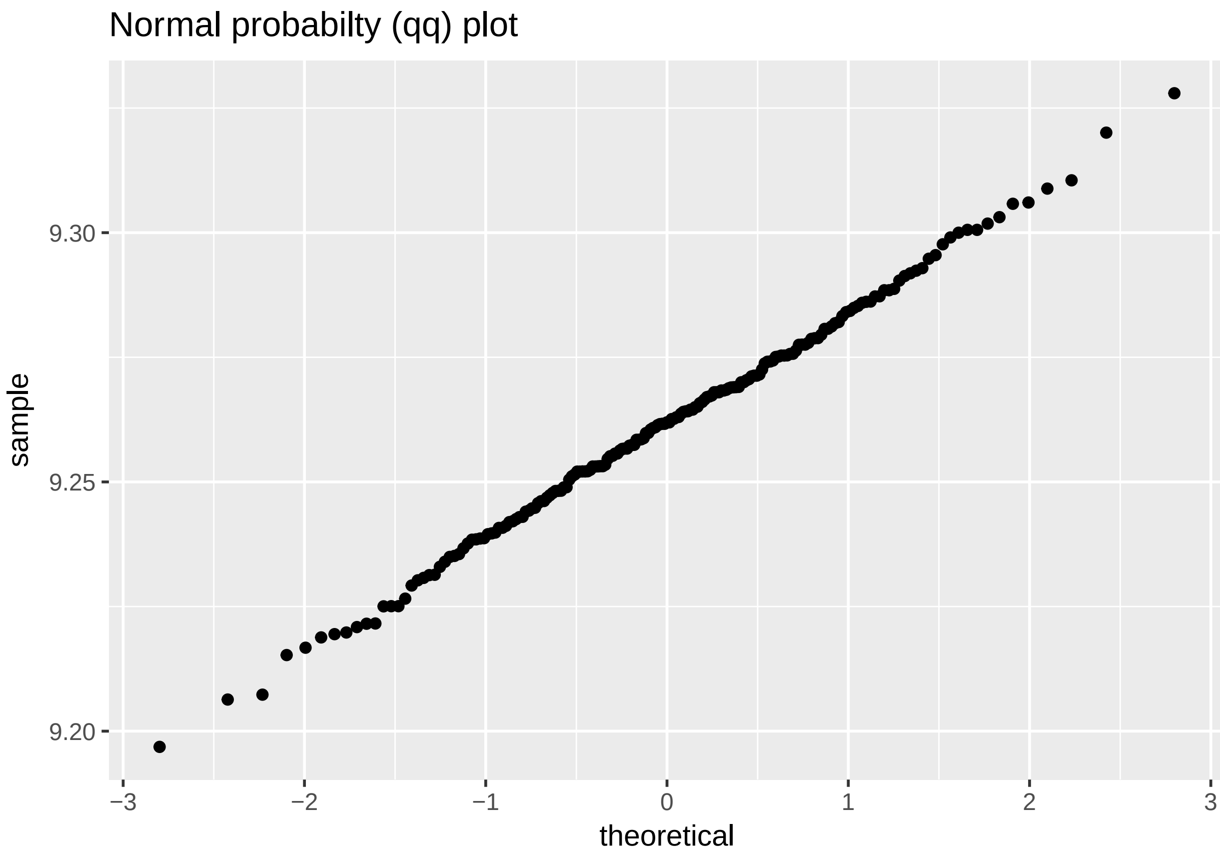



```
ggplot(heat_flow_meter1, aes(value)) +
  geom_histogram() +
  labs(title = "Histogram")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggplot(heat_flow_meter1, aes(sample = value)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



1.4.8 Ceramic strength

Ceramic Strength

```
ceramic_strength <- read_table2("NIST data/JAHANMI2.DAT", skip = 48, col_names = TRUE) %>%
  filter(Lab >= 1) %>%
  rowid_to_column()
```

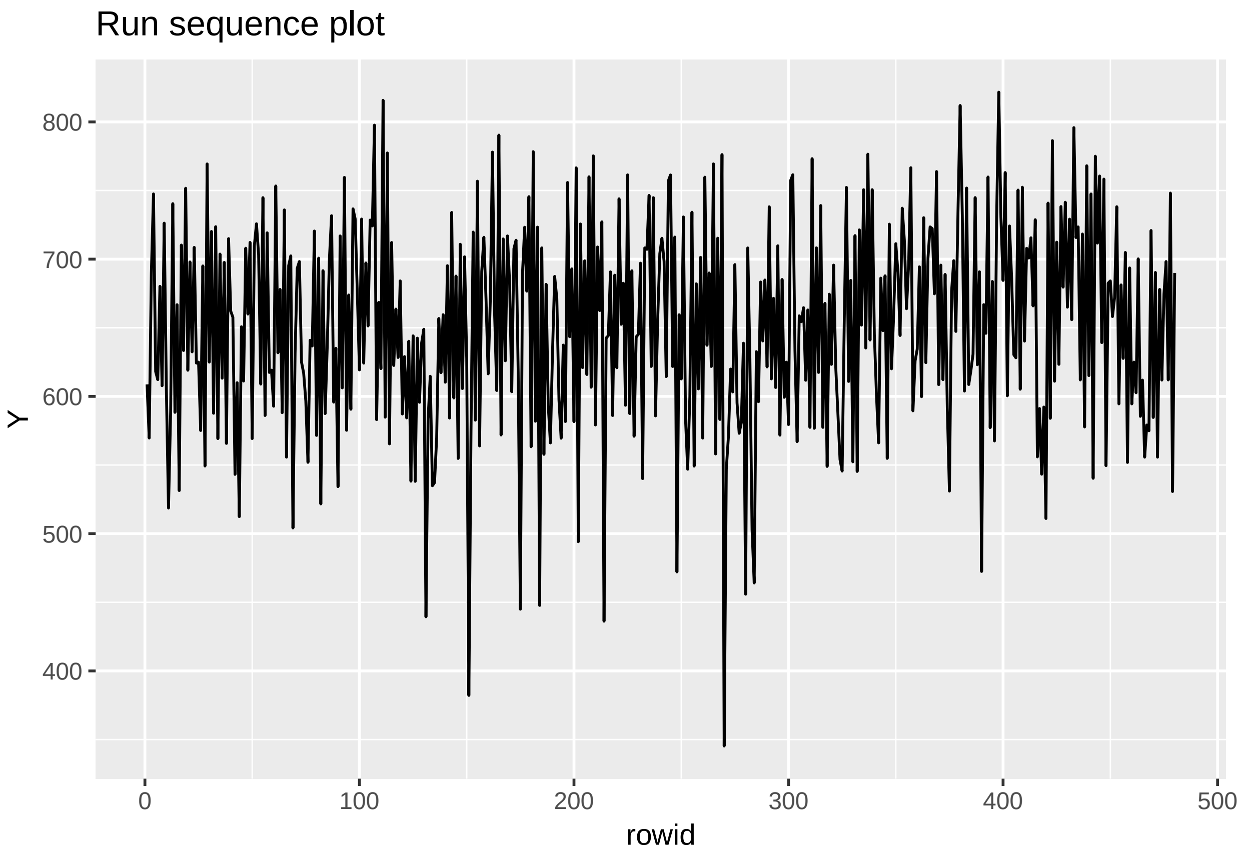
```
## Parsed with column specification:
## cols(
##   Id = col_character(),
##   Lab = col_integer(),
##   Num = col_integer(),
##   Test = col_integer(),
##   Y = col_double(),
##   X1 = col_integer(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   Trt = col_integer(),
##   Set = col_integer(),
##   Llab = col_double(),
##   Rep = col_integer(),
##   Bat = col_integer(),
##   Sblab = col_double(),
##   Set2 = col_integer()
```

```
## )

## Warning in rbind(names(probs), probs_f): number of columns of result is not
## a multiple of vector length (arg 2)

## Warning: 1 parsing failure.
## row # A tibble: 1 x 5 col     row col   expected   actual    file                       expected   <int
```

```
ceramic_strength
```

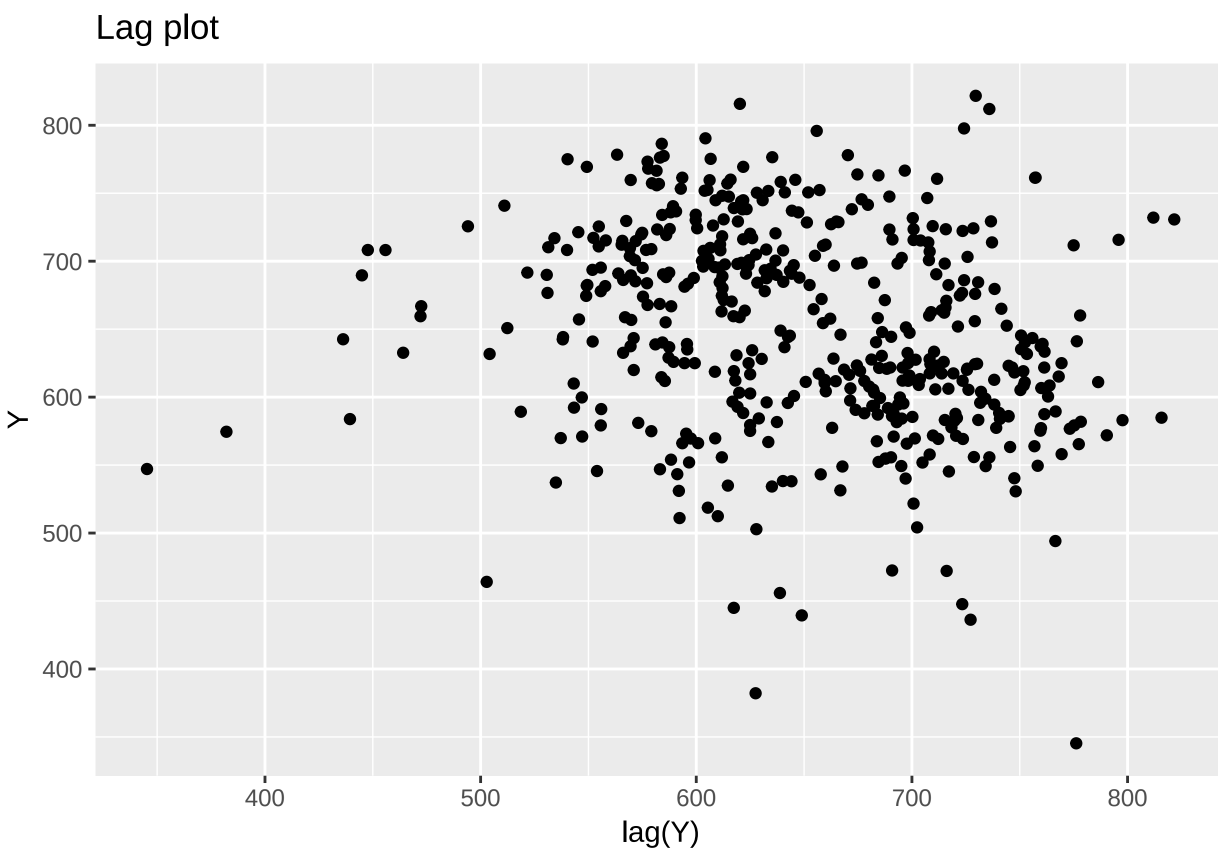
```
## # A tibble: 480 x 17
##    rowid Id      Lab   Num  Test     Y    X1    X2    X3    X4   Trt   Set
##    <int> <chr> <int> <int> <int> <dbl> <int> <int> <int> <int> <int> <int>
##  1     1 1         1     1     1  609.    -1    -1    -1    -1     1     1
##  2     2 2         1     2     1  570.    -1    -1    -1    -1     1     1
##  3     3 3         1     3     1  690.    -1    -1    -1    -1     1     1
##  4     4 4         1     4     1  748.    -1    -1    -1    -1     1     1
##  5     5 5         1     5     1  618.    -1    -1    -1    -1     1     1
##  6     6 6         1     6     1  612.    -1    -1    -1    -1     1     1
##  7     7 7         1     7     1  680.    -1    -1    -1    -1     1     1
##  8     8 8         1     8     1  608.    -1    -1    -1    -1     1     1
##  9     9 9         1     9     1  726.    -1    -1    -1    -1     1     1
## 10    10 10        1    10     1  605.    -1    -1    -1    -1     1     1
## # ... with 470 more rows, and 5 more variables: Llab <dbl>, Rep <int>,
## #   Bat <int>, Sblab <dbl>, Set2 <int>
```

```
ggplot(ceramic_strength, aes(rowid, Y)) +
  geom_line() +
  labs(title = "Run sequence plot")
```



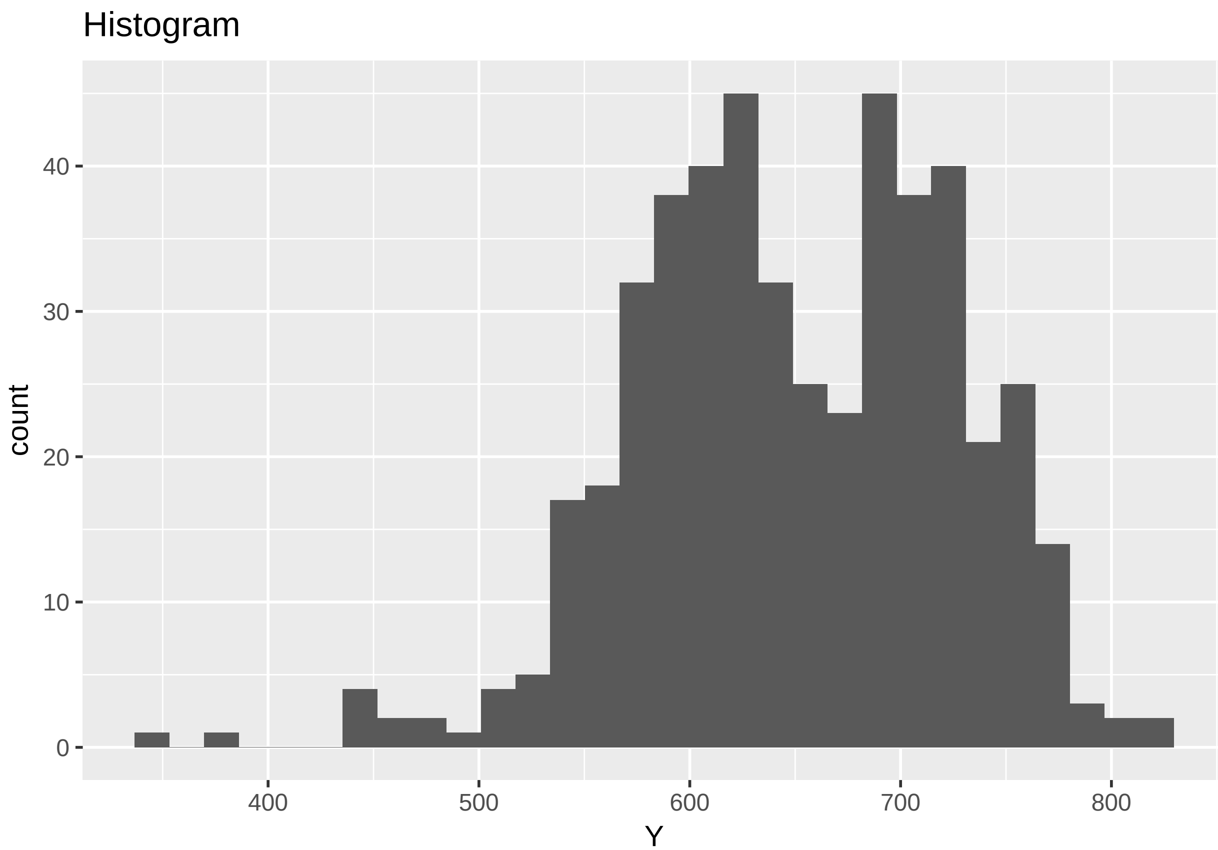
```
ggplot(ceramic_strength, aes(lag(Y), Y)) +
  geom_point() +
  labs(title = "Lag plot")
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

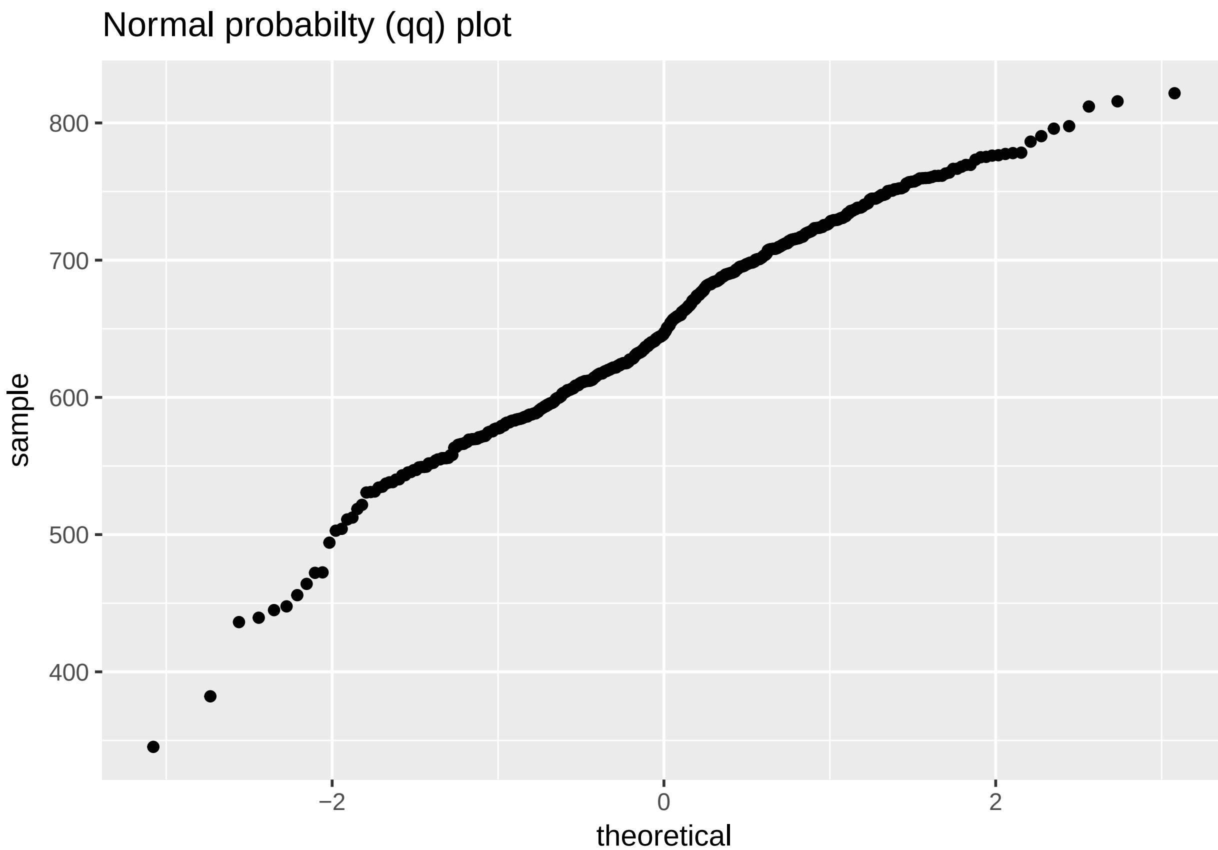



```
ggplot(ceramic_strength, aes(Y)) +
  geom_histogram() +
  labs(title = "Histogram")
```

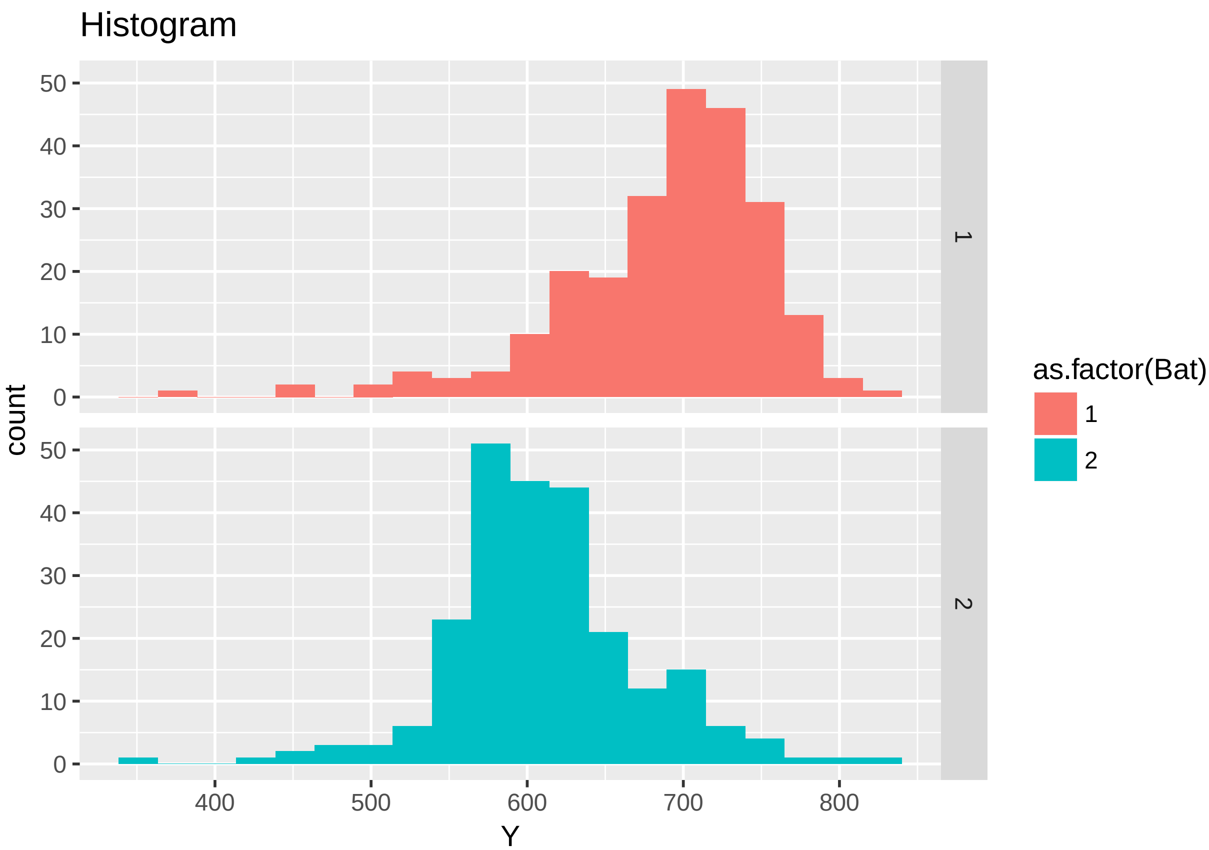
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



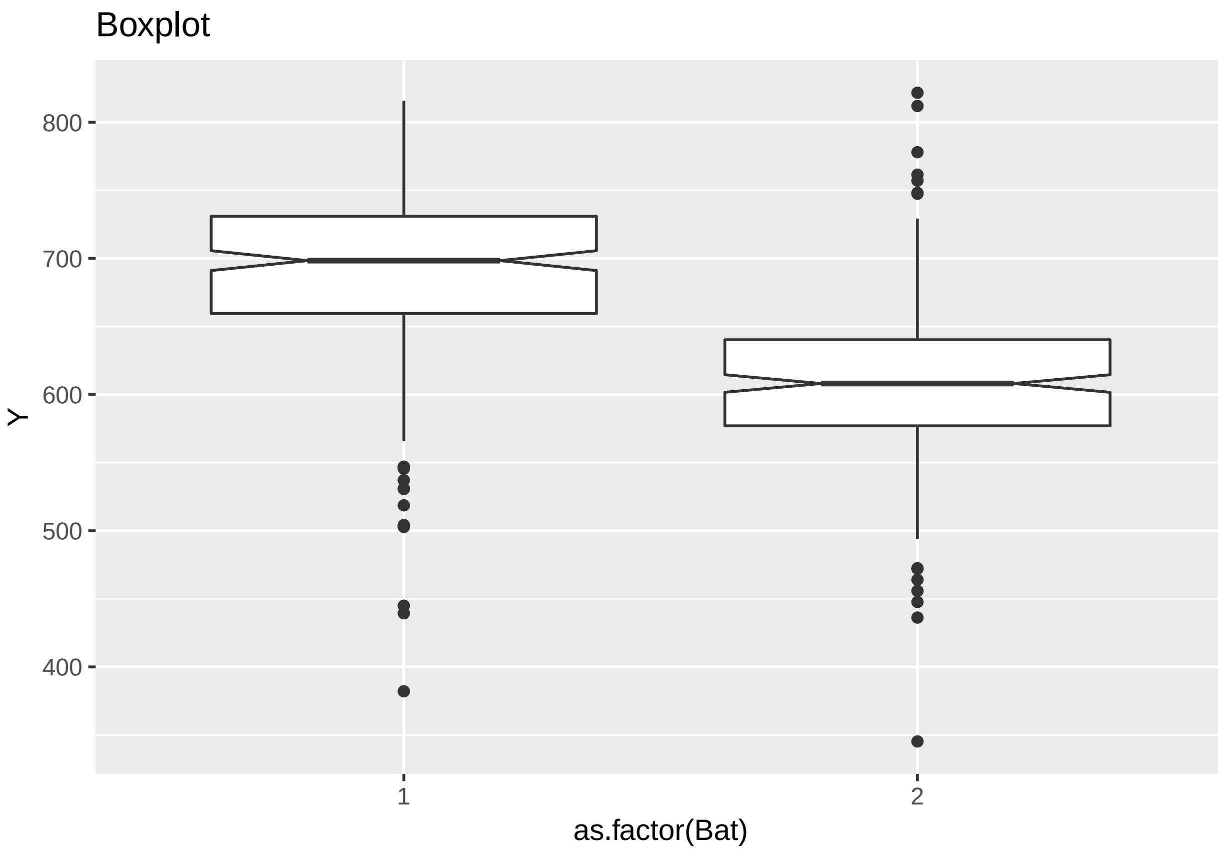

```
ggplot(ceramic_strength, aes(sample = Y)) +
  geom_qq() +
  labs(title = "Normal probabilty (qq) plot")
```



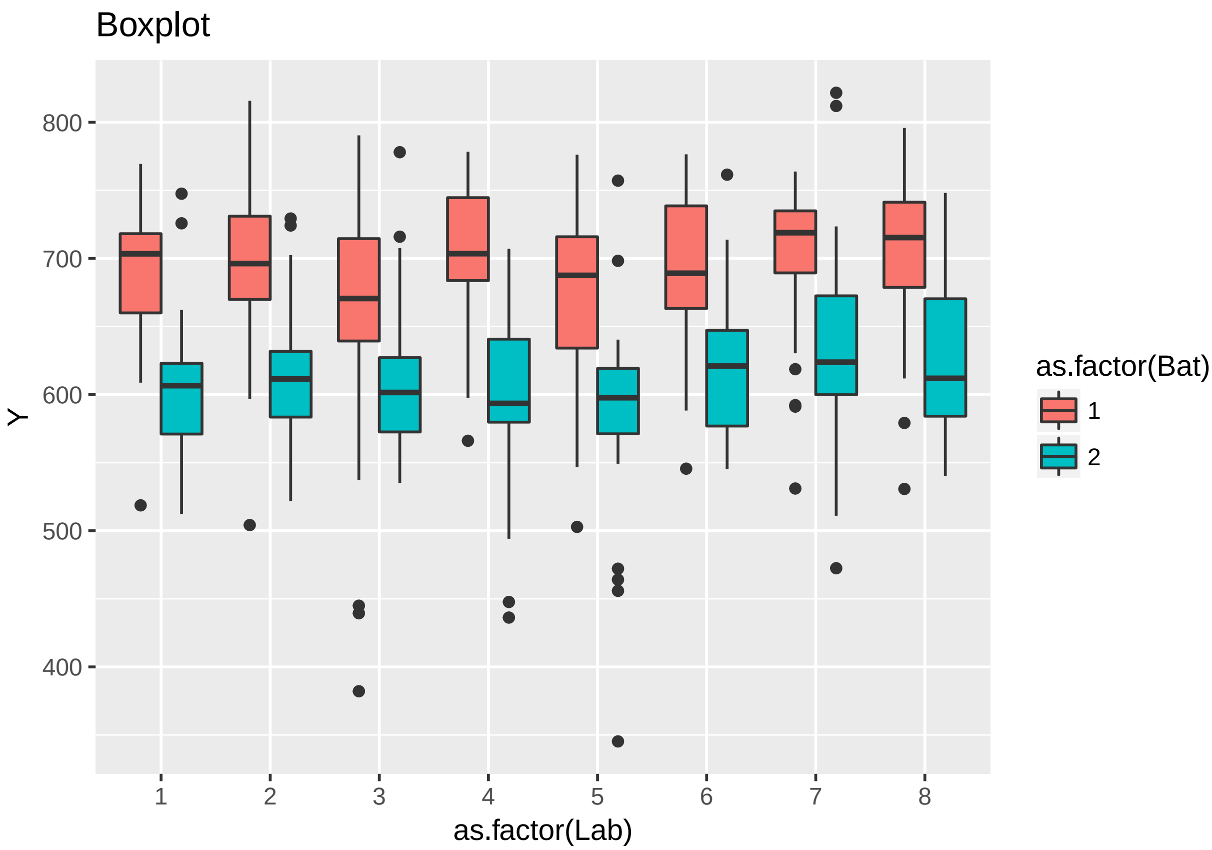

```r
ggplot(ceramic_strength, aes(Y)) +
  geom_histogram(aes(fill = as.factor(Bat)), bins = 20) +
  facet_grid(Bat ~ .) +
  labs(title = "Histogram")
```




```
ggplot(ceramic_strength, aes(as.factor(Bat), Y)) +
  geom_boxplot(notch = TRUE) +
  labs(title = "Boxplot")
```




```
ggplot(ceramic_strength, aes(as.factor(Lab), Y)) +
  geom_boxplot(aes(fill = as.factor(Bat))) +
  labs(title = "Boxplot")
```


Boxplot
800
700
600
500
400
Y
1
2
3
4
5
6
7
8
as.factor(Lab)
as.factor(Bat)
1
2

Chapter 2

Measurement Process Characterization

2.1 Packages used in this chapter

```
library(magrittr)
library(tidyverse)
```

2.2 Characterization

2.3 Gauge R & R studies

2.4 Case Studies

2.4.1 Check standard

2.4.1.1 Background and data

> The measurements on the check standard duplicate certification measurements that were being made, during the same time period, on individual wafers from crystal #51939. For the check standard there were:

- J = 6 repetitions at the center of the wafer on each day
- K = 25 days

Check standard for resistivity measurement

2.4.1.2 Reading the dataset

```
check_standard <- read_table2("NIST data/MPC62_clean.DAT", col_names = TRUE) %>%
  rowid_to_column()
```

```
## Parsed with column specification:
## cols(
##   Crystal_ID = col_integer(),
##   Wafer_ID = col_integer(),
##   Month = col_character(),
##   Day = col_character(),
##   Hour = col_character(),
##   Minute = col_character(),
##   Operator = col_integer(),
##   Humidity = col_integer(),
##   Probe_ID = col_integer(),
##   Temperature = col_double(),
##   Resistance = col_double(),
##   Stdev = col_double(),
##   Df = col_integer()
## )
```

```
check_standard
```

```
## # A tibble: 25 x 14
##    rowid Crystal_ID Wafer_ID Month Day   Hour  Minute Operator Humidity
##    <int>      <int>    <int> <chr> <chr> <chr> <chr>     <int>    <int>
##  1     1      51939      137 03    24    18    01            1       42
##  2     2      51939      137 03    25    12    41            1       35
##  3     3      51939      137 03    25    15    57            1       33
##  4     4      51939      137 03    28    10    10            2       47
##  5     5      51939      137 03    28    13    31            2       44
##  6     6      51939      137 03    28    17    33            1       43
##  7     7      51939      137 03    29    14    40            1       36
##  8     8      51939      137 03    29    16    33            1       35
##  9     9      51939      137 03    30    05    45            2       32
## 10    10      51939      137 03    30    09    26            2       33
## # ... with 15 more rows, and 5 more variables: Probe_ID <int>,
## #   Temperature <dbl>, Resistance <dbl>, Stdev <dbl>, Df <int>
```

2.4.1.3 Level-1 standard deviation

Measurements for J runs over K days for L runs are:

$$Y_{lkj}(l = 1, \ldots, L,\ k = 1, \ldots, K,\ j = 1, \ldots, J)$$

The level-1 repeatability (short term precision) is calcuated from the pooled standard deviation over days and runs

$$s_{1lk} = \sqrt{\frac{1}{J-1}\sum_{j=1}^{J}(Y_{lkj} - \overline{Y}_{lk\bullet})^2}$$

with

$$\overline{Y}_{lk\bullet} = \frac{1}{J}\sum_{j=1}^{J}\overline{Y}_{lkj}$$

As stated in the e-Handbook: >An individual short-term standard deviation will not be a reliable estimate of precision if the degrees of freedom is less than ten, but the individual estimates can be pooled over the K days to obtain a more reliable estimate.

The pooled level-1 standard deviation estimate with v = K(J - 1) degrees of freedom is

$$s_1 = \sqrt{\frac{1}{K}\sum_{k=1}^{K} s_k^2}$$

```
s1_chkstd <- check_standard %>%
  mutate(Stdev_sq = Stdev^2) %$%
  mean(Stdev_sq) %>%
  sqrt()

s1_chkstd
```

```
## [1] 0.06138795
```

Several comments on the code above. I've introduced the `%$%` operator. This allows me to use indivdual columns from my data frame and is useful for preforming mathematical operations on a specific column of data. It is from the **magrittr** package.

I find this type of code easy to read and understand. Describing the operations is simple, I'm just working from inside out of the equation:

- creating a new column of data that is $(Stdev)^2$
- finding the mean of that new column
- taking the square root of that number to give s_1.

2.4.1.4 Level-2 standard deviation (reproducibility)

$$s_{chkstd} = s_2 = \sqrt{\frac{1}{K-1}\sum_{k=1}^{K}\left(\overline{Y}_{k\bullet} - \overline{Y}_{\bullet\bullet}\right)^2}$$

with

$$\overline{Y}_{\bullet\bullet} = \frac{1}{K}\sum_{k=1}^{K}\overline{Y}_{k\bullet}$$

Which is simply the standard deviation of the daily measuremnts

```
s2_chkstd <- check_standard %$%
  sd(Resistance)

s2_chkstd
```

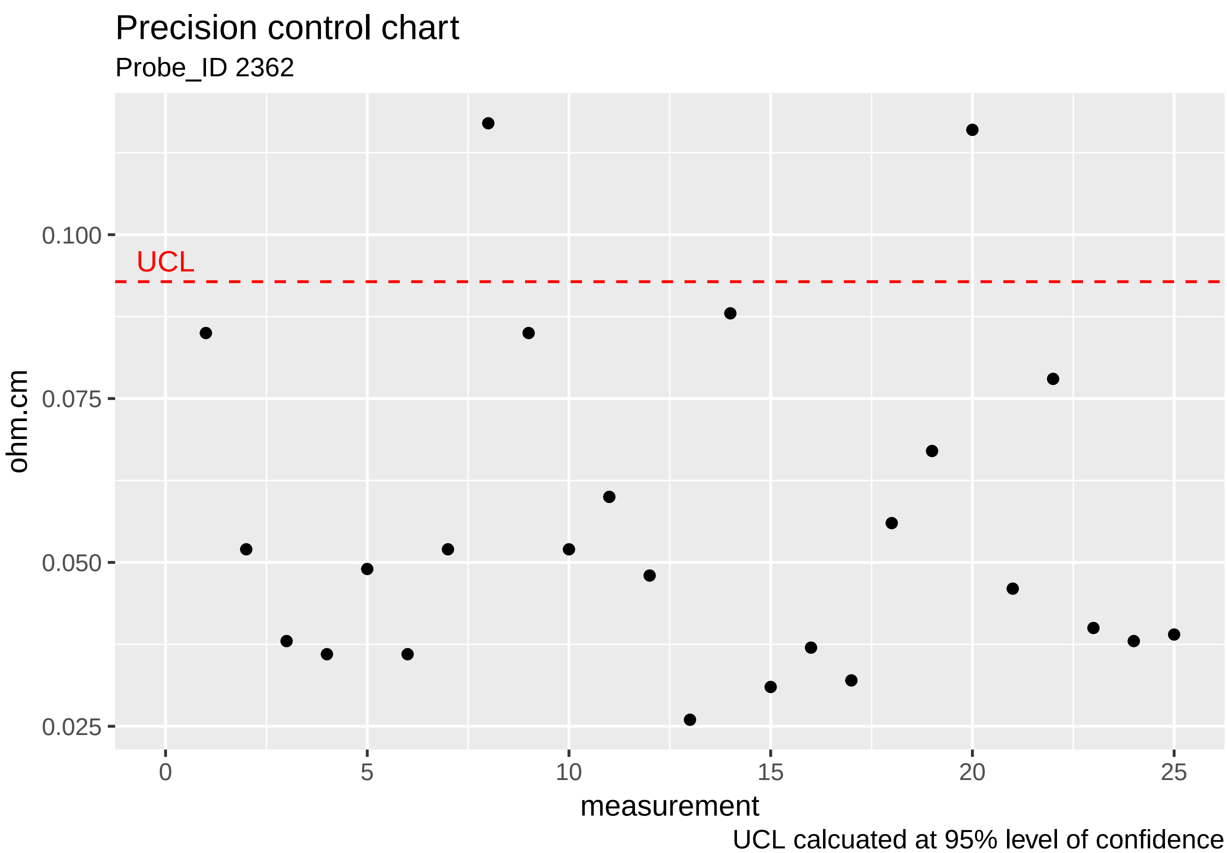
```
## [1] 0.02679813
```

2.4.1.5 Control chart for standard deviation - Precision

```
UCL_precision_ckkstd <- s1_chkstd*sqrt(qf(0.95, 5, 125))
UCL_precision_ckkstd
```

```
## [1] 0.0928313
```

```
ggplot(check_standard) +
  geom_point(aes(rowid, Stdev)) +
  geom_hline(aes(yintercept = UCL_precision_ckkstd), colour = "red", linetype = "dashed") +
  labs(title =  "Precision control chart", subtitle = "Probe_ID 2362", x = "measurement", y = "ohm.cm",
  annotate("text", x = 0, y = 0.096, label = "UCL", colour = "red")
```

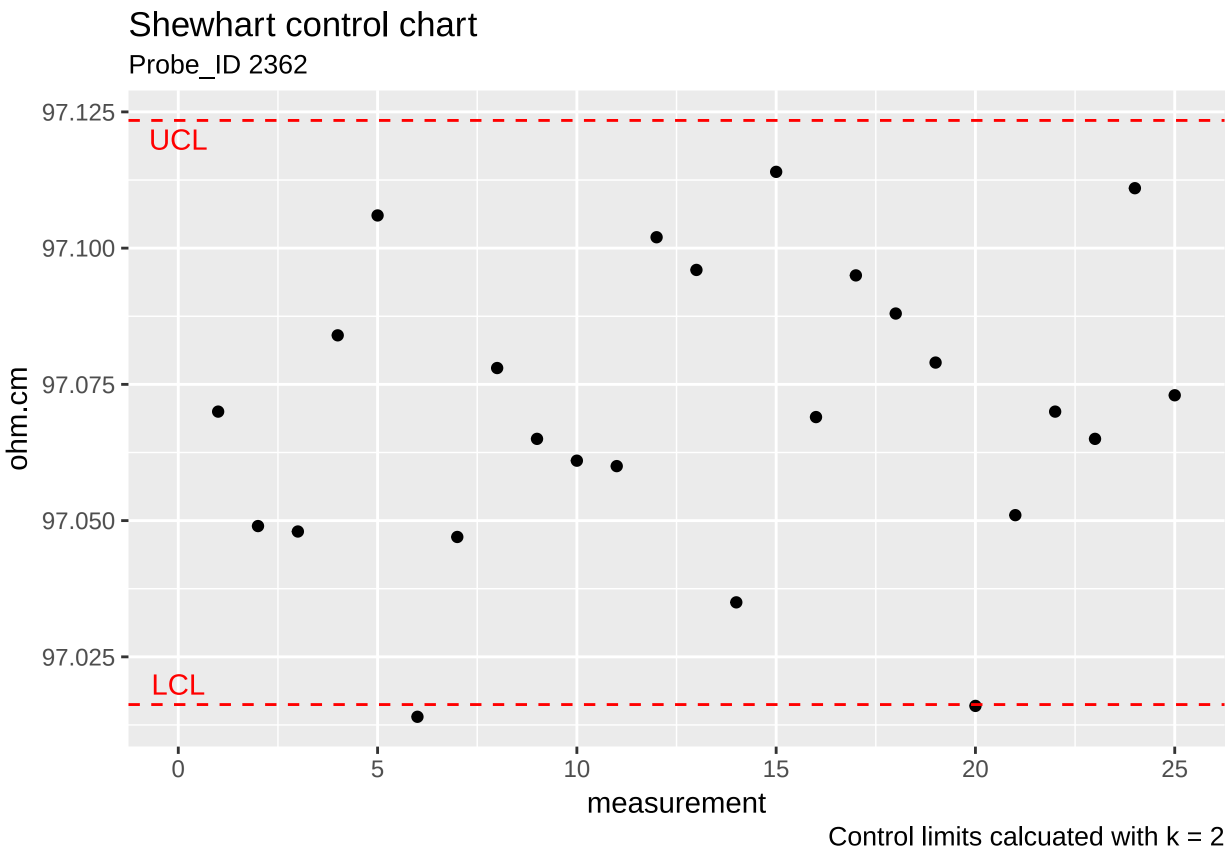


2.4.1.6 Control chart for measurement bias and variability

The control limits for monitoring the bias and long-term variability of resistivity with a Shewhart control chart are given by

$$UCL = \text{Average} + 2 \cdot s_2 Centerline = \text{Average} LCL = \text{Average} 2 \cdot s_2$$

```
ggplot(check_standard) +
  geom_point(aes(rowid, Resistance)) +
  geom_hline(aes(yintercept = (mean(Resistance) + 2*s2_chkstd)), colour = "red", linetype = "dashed") +
  geom_hline(aes(yintercept = (mean(Resistance) - 2*s2_chkstd)), colour = "red", linetype = "dashed") +
  labs(title =  "Shewhart control chart", subtitle = "Probe_ID 2362", x = "measurement", y = "ohm.cm", c
  annotate("text", x = 0, y = 97.12, label = "UCL", colour = "red") +
  annotate("text", x = 0, y = 97.02, label = "LCL", colour = "red")
```



2.4.2 Gauge study

2.4.2.1 Background and data

Measurements on the check standards are used to estimate repeatability, day effect, and run effect The effect of operator was not considered to be significant for this study; therefore, 'day' replaces 'operator' as a factor in the nested design. Averages and standard deviations from J = 6 measurements at the center of each wafer are shown in the table.

- J = 6 measurements at the center of the wafer per day
- K = 6 days (one operator) per repetition
- L = 2 runs (complete)
- Q = 5 wafers (check standards 138, 139, 140, 141, 142)
- R = 5 probes (1, 281, 283, 2062, 2362)

Gauge study of resistivity probes

```
gauge_study <- read_table2("NIST data/MPC61_clean.DAT", col_names = TRUE) %>%
  rowid_to_column()
```

```
## Parsed with column specification:
## cols(
##   RUN = col_integer(),
##   WAFER = col_double(),
##   PROBE = col_double(),
##   MONTH = col_double(),
##   DAY = col_double(),
```

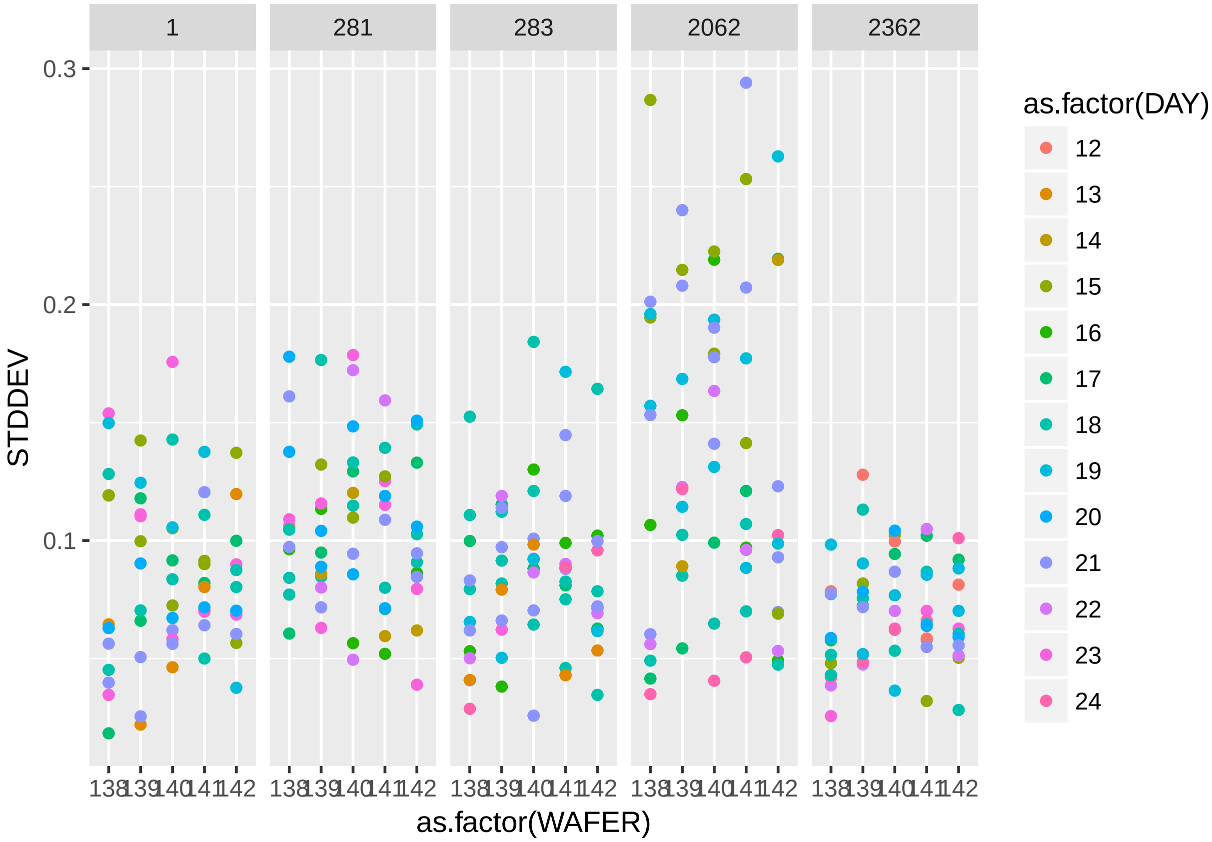
```
##   OP = col_double(),
##   TEMP = col_double(),
##   AVERAGE = col_double(),
##   STDDEV = col_double()
## )
```

```
gauge_study
```

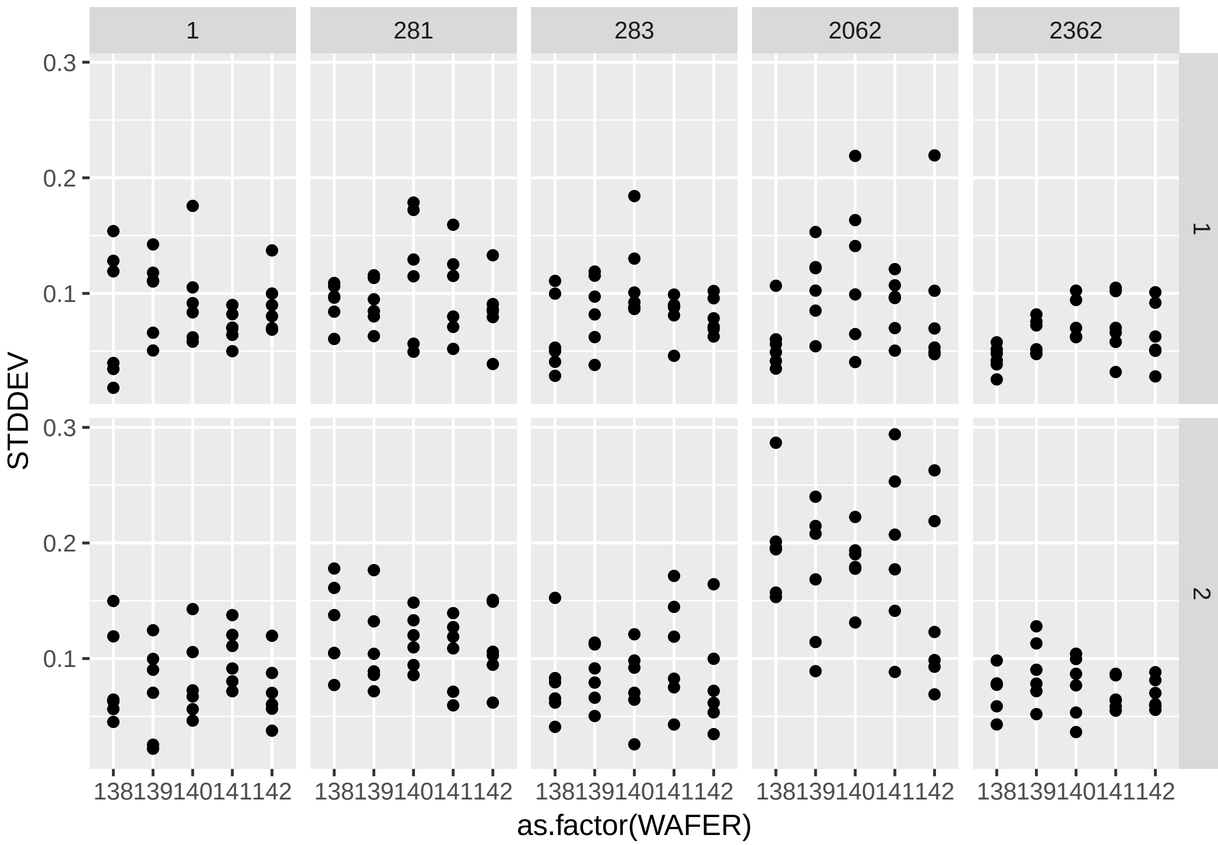
```
## # A tibble: 300 x 10
##    rowid   RUN WAFER PROBE MONTH   DAY    OP  TEMP AVERAGE STDDEV
##    <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>  <dbl>
##  1     1     1  138.    1.    3.   15.    1.  23.0    95.2 0.119
##  2     2     1  138.    1.    3.   17.    1.  23.0    95.2 0.0183
##  3     3     1  138.    1.    3.   18.    1.  22.8    95.2 0.128
##  4     4     1  138.    1.    3.   21.    1.  23.2    95.2 0.0398
##  5     5     1  138.    1.    3.   23.    2.  23.2    95.1 0.0346
##  6     6     1  138.    1.    3.   23.    1.  23.2    95.1 0.154
##  7     7     1  138.  281.    3.   16.    1.  23.0    95.2 0.0963
##  8     8     1  138.  281.    3.   17.    1.  23.0    95.1 0.0606
##  9     9     1  138.  281.    3.   18.    1.  22.8    95.1 0.0842
## 10    10     1  138.  281.    3.   21.    1.  23.3    95.1 0.0973
## # ... with 290 more rows
```

2.4.2.2 Repeatability standard deviations

```
ggplot(gauge_study) +
  geom_point(aes(as.factor(WAFER), STDDEV, colour = as.factor(DAY))) +
  facet_wrap(~ as.factor(PROBE), nrow = 1)
```

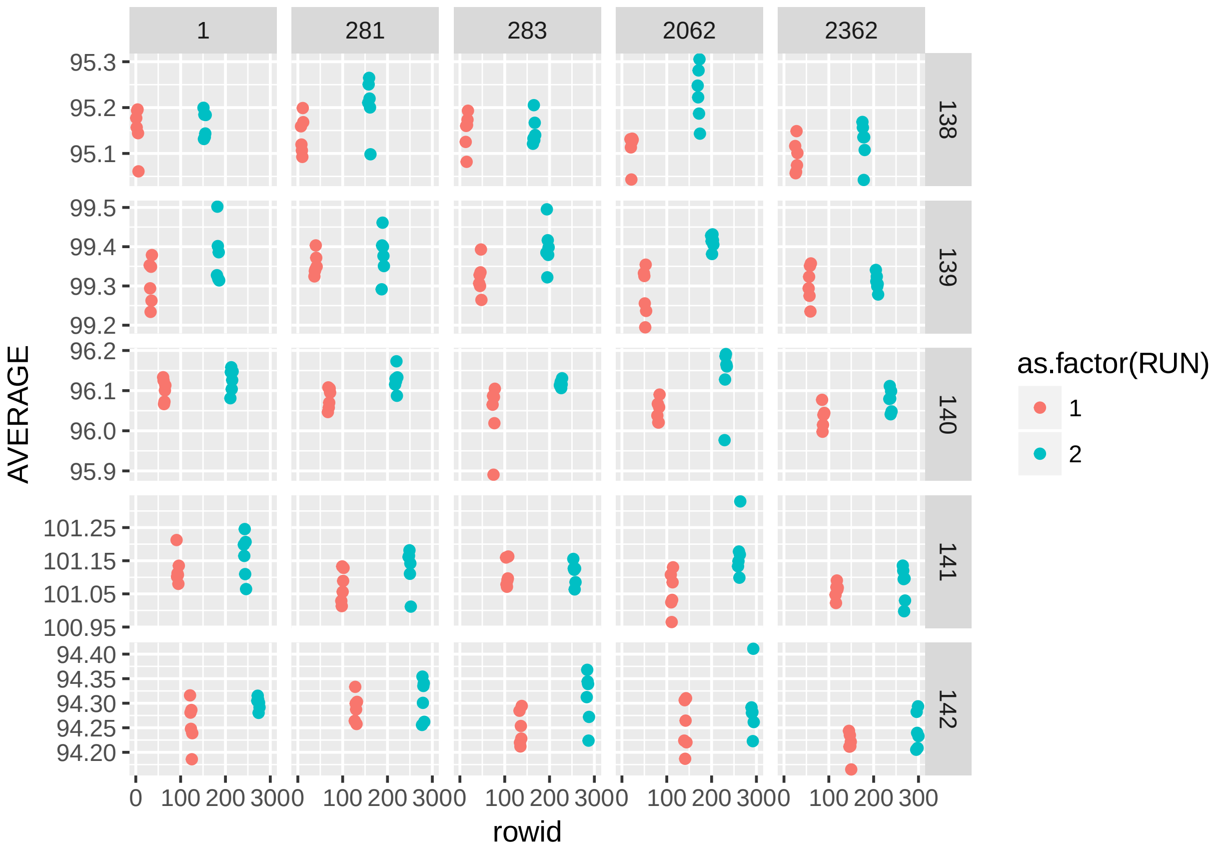



```r
ggplot(gauge_study) +
  geom_point(aes(as.factor(WAFER), STDDEV)) +
  facet_grid(as.factor(RUN) ~ as.factor(PROBE))
```



2.4.2.3 Effects of days and long-term stability

```
ggplot(gauge_study) +
  geom_point(aes(rowid, AVERAGE, colour = as.factor(RUN))) +
  facet_grid(as.factor(WAFER) ~ as.factor(PROBE), scales = "free_y")
```



2.4.2.4 Differences among 5 probes

```
probe_means_run <- gauge_study %>%
  group_by(PROBE, WAFER, RUN) %>%
  summarise(n = n(), probe_mean = mean(AVERAGE)) %>%
  unite(join_id, WAFER, RUN, sep = "_", remove = FALSE) %>%
  ungroup()

probe_means_run
```

```
## # A tibble: 50 x 6
##    PROBE join_id WAFER   RUN     n probe_mean
##    <dbl> <chr>   <dbl> <int> <int>      <dbl>
##  1    1. 138_1    138.     1     6       95.2
##  2    1. 138_2    138.     2     6       95.2
##  3    1. 139_1    139.     1     6       99.3
##  4    1. 139_2    139.     2     6       99.4
##  5    1. 140_1    140.     1     6       96.1
##  6    1. 140_2    140.     2     6       96.1
##  7    1. 141_1    141.     1     6      101.
##  8    1. 141_2    141.     2     6      101.
##  9    1. 142_1    142.     1     6       94.3
## 10    1. 142_2    142.     2     6       94.3
## # ... with 40 more rows
```

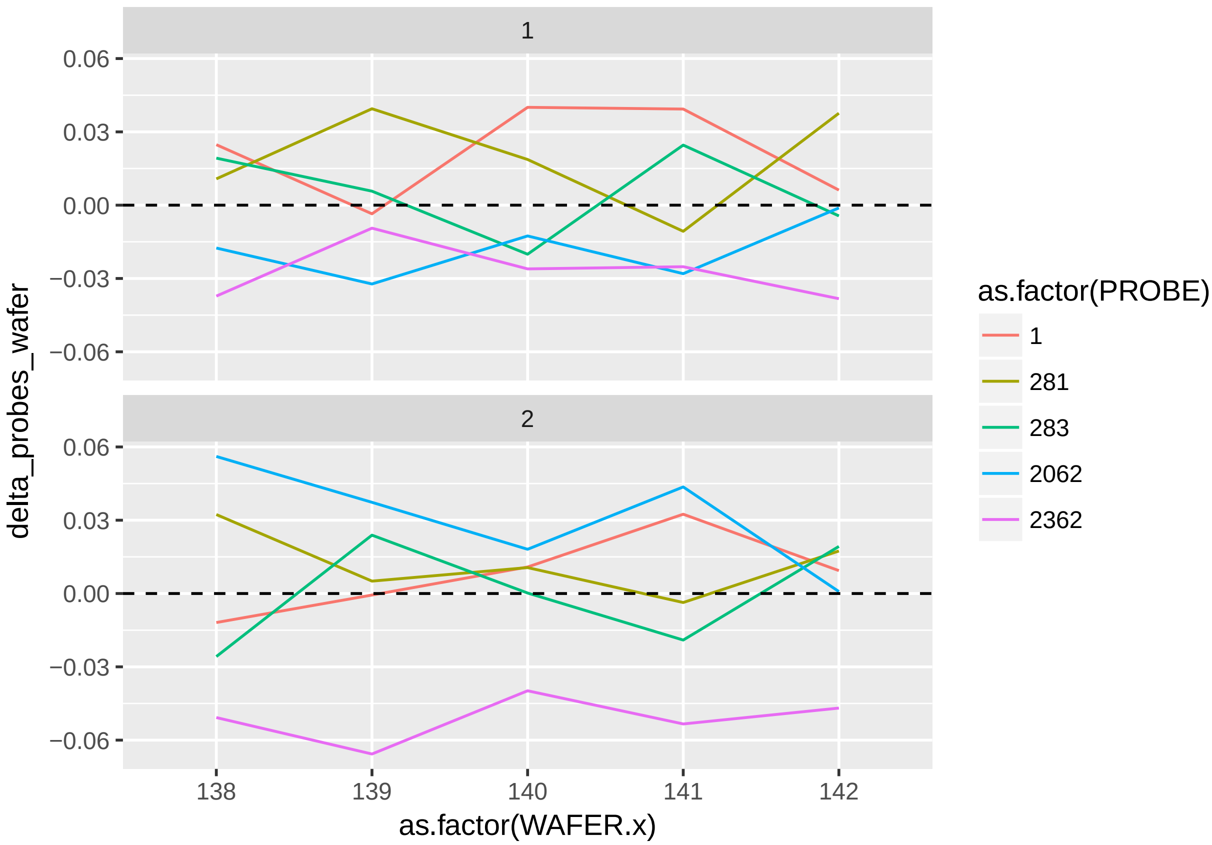
```
wafer_means_run <- gauge_study %>%
  group_by(WAFER, RUN) %>%
  summarise(n = n(), wafer_means = mean(AVERAGE)) %>%
  unite(join_id, WAFER, RUN, sep = "_", remove = FALSE) %>%
  ungroup()

wafer_means_run
```

```
## # A tibble: 10 x 5
##    join_id WAFER   RUN     n wafer_means
##    <chr>   <dbl> <int> <int>       <dbl>
##  1 138_1    138.     1    30        95.1
##  2 138_2    138.     2    30        95.2
##  3 139_1    139.     1    30        99.3
##  4 139_2    139.     2    30        99.4
##  5 140_1    140.     1    30        96.1
##  6 140_2    140.     2    30        96.1
##  7 141_1    141.     1    30       101.
##  8 141_2    141.     2    30       101.
##  9 142_1    142.     1    30        94.3
## 10 142_2    142.     2    30        94.3
```

```
delta_probes <- left_join(probe_means_run, wafer_means_run, by = "join_id") %>%
  mutate(delta_probes_wafer = probe_mean - wafer_means)

delta_probes
```

```
## # A tibble: 50 x 11
##    PROBE join_id WAFER.x RUN.x   n.x probe_mean WAFER.y RUN.y   n.y
##    <dbl> <chr>     <dbl> <int> <int>      <dbl>   <dbl> <int> <int>
##  1    1. 138_1      138.     1     6       95.2    138.     1    30
##  2    1. 138_2      138.     2     6       95.2    138.     2    30
##  3    1. 139_1      139.     1     6       99.3    139.     1    30
##  4    1. 139_2      139.     2     6       99.4    139.     2    30
##  5    1. 140_1      140.     1     6       96.1    140.     1    30
##  6    1. 140_2      140.     2     6       96.1    140.     2    30
##  7    1. 141_1      141.     1     6      101.     141.     1    30
##  8    1. 141_2      141.     2     6      101.     141.     2    30
##  9    1. 142_1      142.     1     6       94.3    142.     1    30
## 10    1. 142_2      142.     2     6       94.3    142.     2    30
## # ... with 40 more rows, and 2 more variables: wafer_means <dbl>,
## #   delta_probes_wafer <dbl>
```

```
ggplot(delta_probes) +
  geom_line(aes(as.factor(WAFER.x), delta_probes_wafer, group = as.factor(PROBE), colour = as.factor(PR
  geom_hline(aes(yintercept = 0), linetype = "dashed") +
  facet_wrap(~ as.factor(RUN.x), ncol = 1)
```



2.4.2.5 Analysis and interpretation

Table of estimates for probe #2362

> A graphical analysis shows repeatability standard deviations plotted by wafer and probe… The plots show that for both runs the precision of this probe is better than for the other probes.
>
> Probe #2362, because of its superior precision, was chosen as the tool for measuring all 100 ohm.cm resistivity wafers at NIST. Therefore, the remainder of the analysis focuses on this probe.

2.4.2.6 probe #2362

```
probe_2362 <- gauge_study %>%
  filter(PROBE == 2362)

probe_2362
```

```
## # A tibble: 60 x 10
##    rowid   RUN WAFER PROBE MONTH   DAY    OP  TEMP AVERAGE STDDEV
##    <int> <int> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>   <dbl>  <dbl>
##  1    25     1  138. 2362.    3.   15.    1.  23.1    95.1 0.0480
##  2    26     1  138. 2362.    3.   17.    1.  23.0    95.1 0.0577
##  3    27     1  138. 2362.    3.   18.    1.  23.0    95.1 0.0516
##  4    28     1  138. 2362.    3.   22.    1.  23.2    95.1 0.0386
##  5    29     1  138. 2362.    3.   23.    2.  23.3    95.1 0.0256
```

```
## 6    30     1  138. 2362.    3.    24.    2.  23.1    95.1 0.0420
## 7    55     1  139. 2362.    3.    15.    1.  23.1    99.3 0.0818
## 8    56     1  139. 2362.    3.    17.    1.  23.0    99.3 0.0723
## 9    57     1  139. 2362.    3.    18.    1.  22.9    99.3 0.0756
## 10   58     1  139. 2362.    3.    22.    1.  23.3    99.4 0.0475
## # ... with 50 more rows
```

Pooled level-1 standard deviations (ohm.cm)

```
s1_2362_1 <- probe_2362 %>%
  filter(RUN == 1) %>%
  mutate(Stdev_sq = STDDEV^2) %$%
  mean(Stdev_sq) %>%
  sqrt()

s1_2362_1
```

```
## [1] 0.06750898
```

```
s1_2362_2 <- probe_2362 %>%
  filter(RUN == 2) %>%
  mutate(Stdev_sq = STDDEV^2) %$%
  mean(Stdev_sq) %>%
  sqrt()

s1_2362_2
```

```
## [1] 0.07785664
```

```
s1_2362 <- probe_2362 %>%
  mutate(Stdev_sq = STDDEV^2) %$%
  mean(Stdev_sq) %>%
  sqrt()

s1_2362
```

```
## [1] 0.07286673
```

Level-2 standard deviations (ohm.cm) for 5 wafers

```
s2_2362 <- gauge_study %>%
  group_by(PROBE, WAFER, RUN) %>%
  filter(PROBE == 2362) %>%
  summarise(df = n()-1, probe_mean = mean(AVERAGE), probe_stdev = sd(AVERAGE), probe_stdev_sq = probe_s
  group_by(RUN) %>%
  summarise(s2_run = sqrt(mean(probe_stdev_sq)))

s2_2362
```

```
## # A tibble: 2 x 2
##     RUN s2_run
##   <int>  <dbl>
## 1     1 0.0333
## 2     2 0.0388
```

Over both runs

```
s2_2352_all <- s2_2362 %>%
  mutate(s2_run_sq = s2_run^2) %$%
```

```
  mean(s2_run_sq) %>%
  sqrt()

s2_2352_all
```

```
## [1] 0.03616824
```

```
sd_2362_wafer <- gauge_study %>%
  group_by(PROBE, WAFER, RUN) %>%
  filter(PROBE == 2362) %>%
  summarise(probe_mean = mean(AVERAGE)) %>%
  mutate(
    run_number = case_when(
      RUN == 1 ~ "Run1",
      RUN == 2 ~ "Run2"
    )
  ) %>%
  dplyr::select(PROBE, WAFER, probe_mean, run_number) %>%
  group_by(WAFER) %>%
  summarise(sd_wafer = sd(probe_mean))


sd_2362_wafer
```

```
## # A tibble: 5 x 2
##   WAFER sd_wafer
##   <dbl>    <dbl>
## 1  138.  0.0222
## 2  139.  0.00271
## 3  140.  0.0288
## 4  141.  0.0133
## 5  142.  0.0205
```

```
s3_2362 <- sd_2362_wafer %>%
  mutate(sd_wafer_sq = sd_wafer^2) %$%
  mean(sd_wafer_sq) %>%
  sqrt()

s3_2362
```

```
## [1] 0.01964524
```

Chapter 3

Production Process Characterization

3.1 Pacakges used in this chapter

3.2 Case Studies

3.2.1 Furnace Case Study

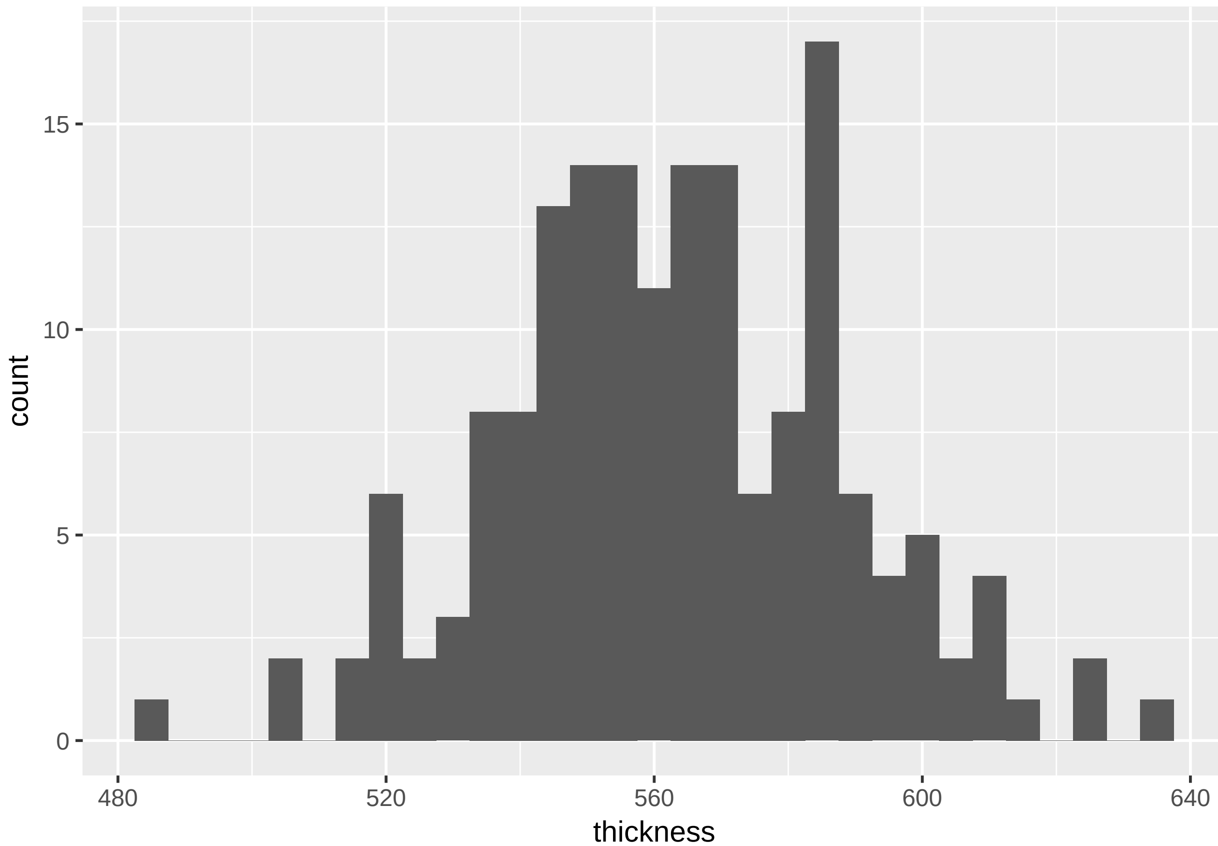
3.2.1.1 Background and Data

> Introduction In a semiconductor manufacturing process flow, we have a step whereby we grow an oxide film on the silicon wafer using a furnace. In this step, a cassette of wafers is placed in a quartz "boat" and the boats are placed in the furnace. The furnace can hold four boats. A gas flow is created in the furnace and it is brought up to temperature and held there for a specified period of time (which corresponds to the desired oxide thickness). This study was conducted to determine if the process was stable and to characterize sources of variation so that a process control strategy could be developed.
>
> The goal of this study is to determine if this process is capable of consistently growing oxide films with a thickness of 560 Angstroms +/- 100 Angstroms. An additional goal is to determine important sources of variation for use in the development of a process control strategy.

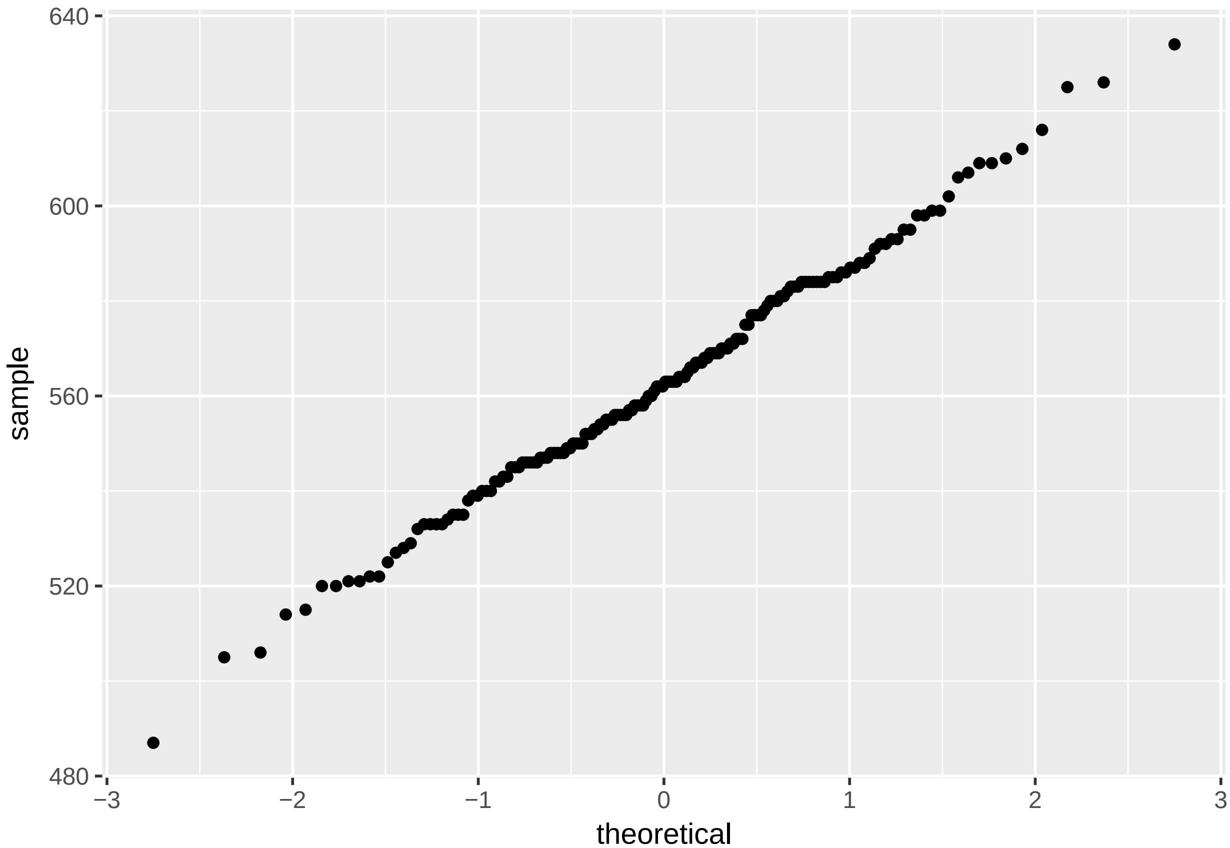
```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer()
## )
```

3.2.1.2 Histogram and normal probability plots of all data

```
ggplot(data = furnace, mapping = aes(x = thickness)) +
  geom_histogram(binwidth = 5)
```




```
ggplot(data = furnace) +
  geom_qq(aes(sample =thickness))
```



3.2.1.3 Summary statistics and standard deviation of film thickness

```
summary(furnace$thickness)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
##   487.0   546.8   562.5   563.0   582.2   634.0
```

```
sd(furnace$thickness)
```

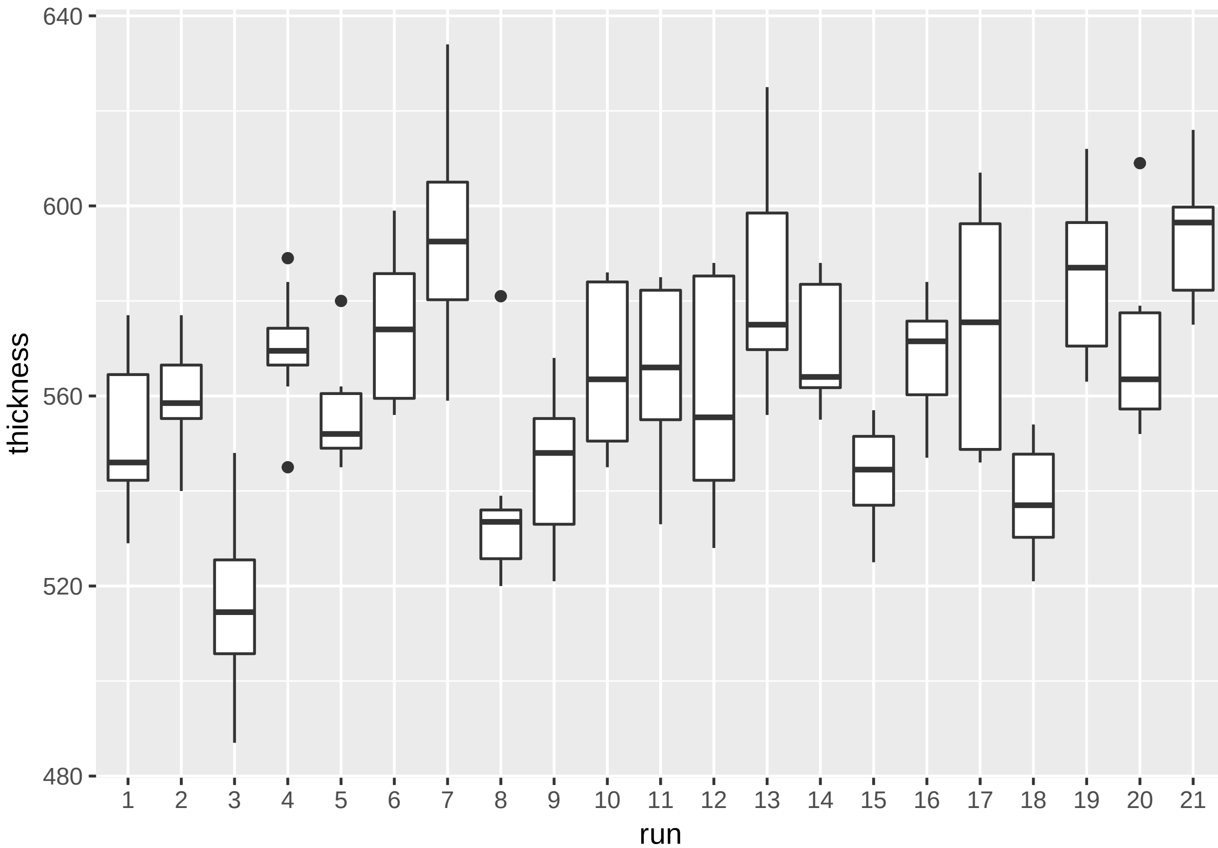
```
## [1] 25.38468
```

The NIST/SEMATECH e-Handbook ask for a capability analysis; however, this is covered in Chapter 6

3.2.1.4 Sources of variation

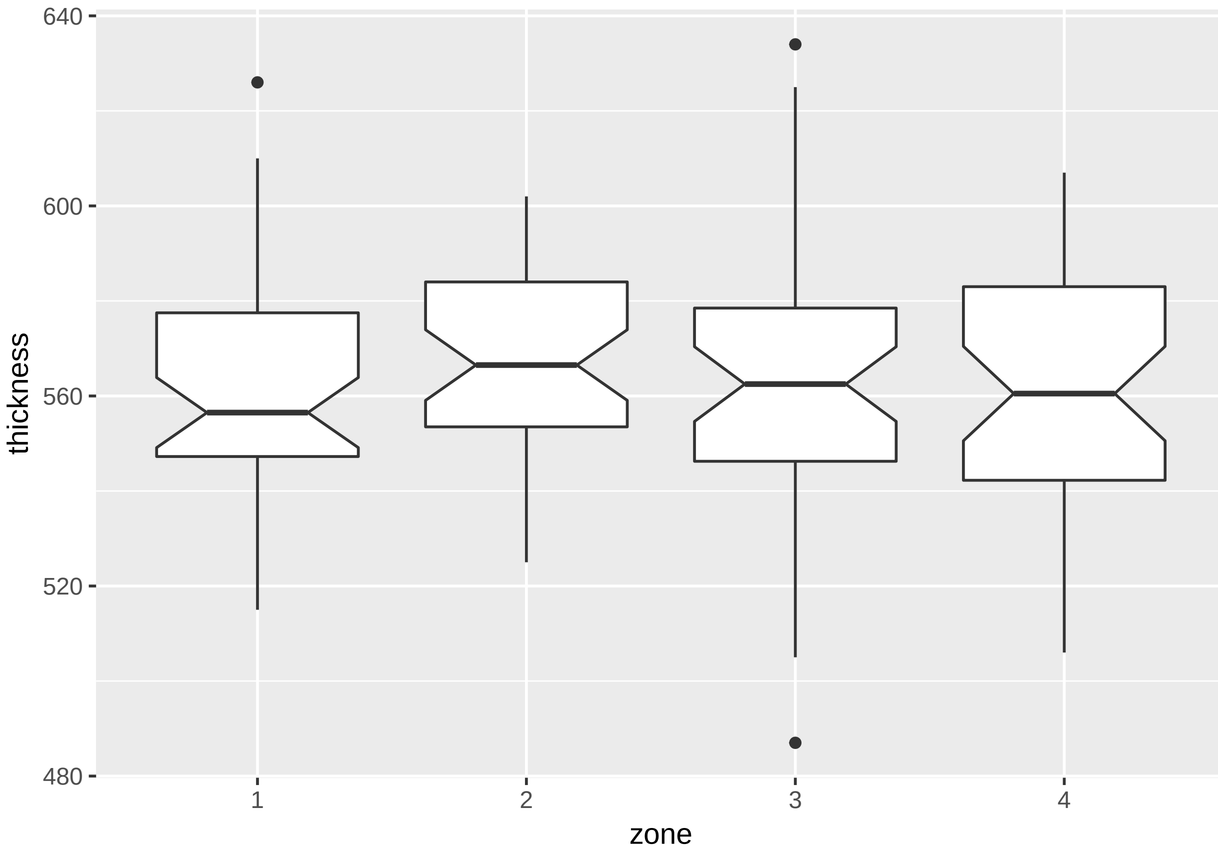
3.2.1.4.1 Boxplot by run

```
ggplot(data = furnace, mapping = aes(x = run, y = thickness)) +
         geom_boxplot()
```



3.2.1.4.2 Boxplot by zone

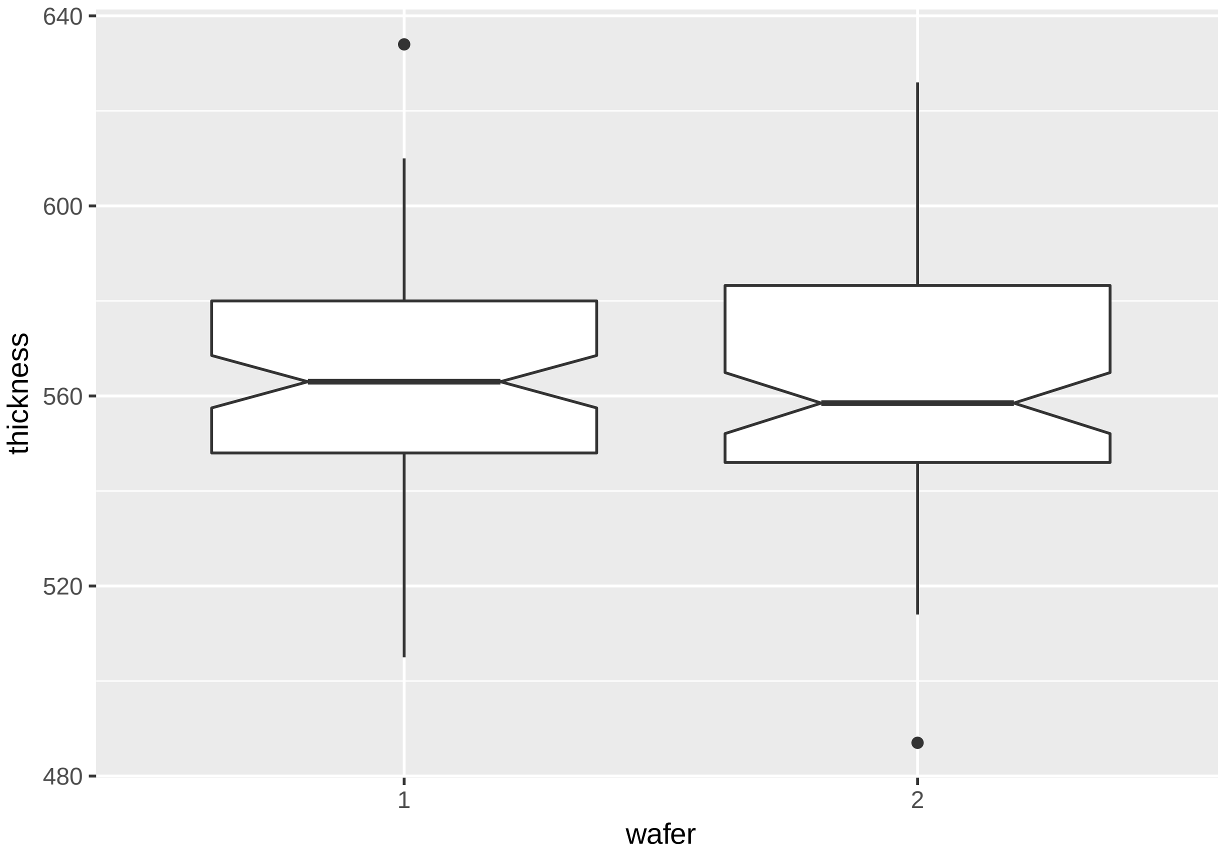
```
ggplot(data = furnace, mapping = aes(x = zone, y = thickness)) +
  geom_boxplot(notch = TRUE)
```



Notch if FALSE (default) make a standard box plot. If TRUE, make a notched box plot. Notches are used to compare groups; if the notches of two boxes do not overlap, this suggests that the medians are significantly different.

3.2.1.4.3 Boxplots by wafer

```
ggplot(data = furnace, mapping = aes(x = wafer, y = thickness)) +
  geom_boxplot(notch = TRUE)
```



3.2.1.4.4 One-way ANOVA to confirm thickness is different by run

```
aov.thickness <- aov(thickness ~ run, data = furnace)
summary(aov.thickness)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## run          20  61442  3072.1   9.781 <2e-16 ***
## Residuals   147  46170   314.1
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2.1.4.5 One-way ANOVA to confirm thickness is not different by zone

```
aov.zone <- aov(thickness ~ zone, data = furnace)
summary(aov.zone)
```

```
##              Df Sum Sq Mean Sq F value Pr(>F)
## zone          3    913   304.2   0.468  0.705
## Residuals   164 106699   650.6
```

3.2.1.4.6 Nested ANOVA

```
aov.thickness.nested <- aov(thickness ~ run + run:zone, data = furnace)
summary(aov.thickness.nested)
```

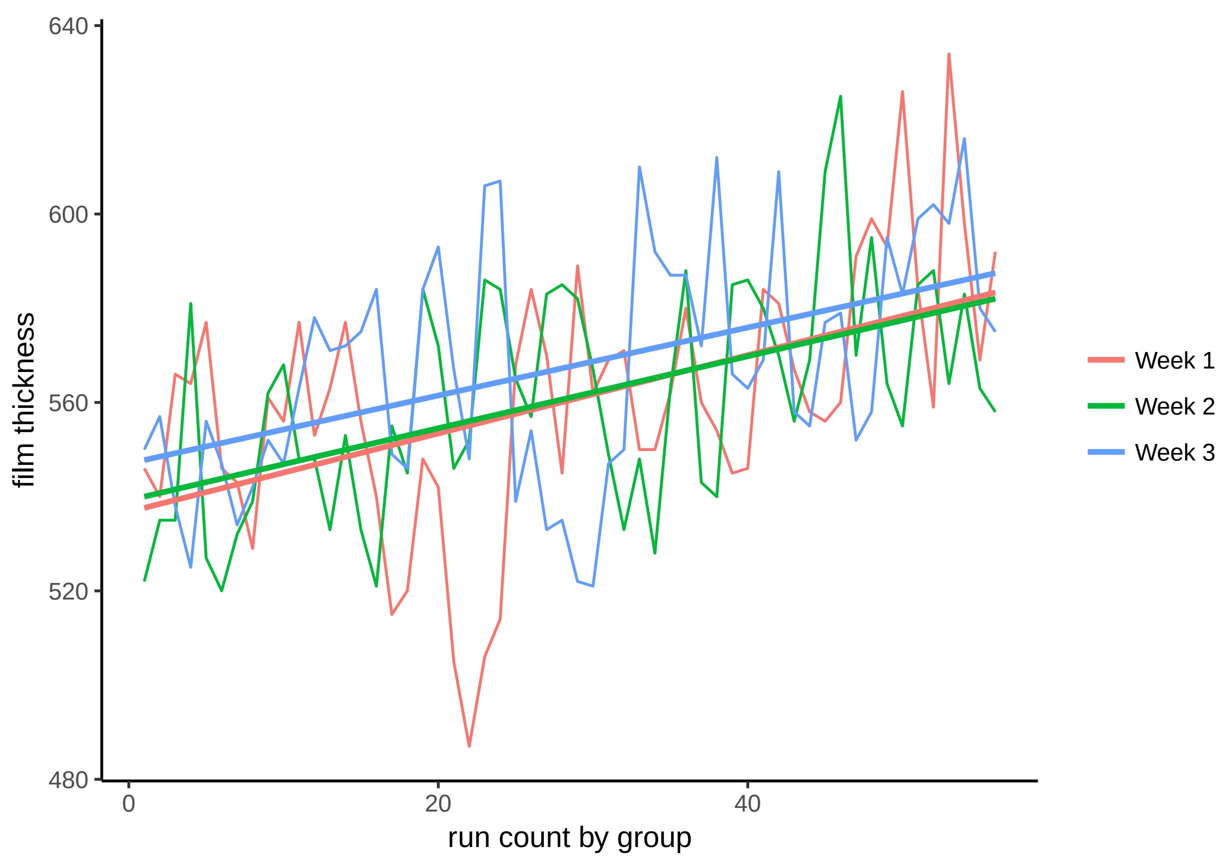
```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## run         20  61442  3072.1  25.412  < 2e-16 ***
```

```
## run:zone     63  36014   571.7   4.729 3.85e-11 ***
## Residuals    84  10155   120.9
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

3.2.1.4.7 Observed trend by week

```
furnace_group <- furnace %>%
  mutate(run = as.integer(run)) %>%
  mutate(grouping = case_when(run <= 7 ~ "Week 1",
                              run > 7 & run <= 14 ~ "Week 2",
                              run > 14 ~ "Week 3")) %>%
  mutate(counting = 1:n()) %>%
  # mutate(counting = as.double(counting)) %>%
  mutate(count_by_group = case_when(counting <= 56 ~ counting,
                              counting > 56 & counting <= 112 ~ counting - 56L,
                              counting > 112 ~ counting - 112L))
```

```
ggplot(furnace_group) +
  geom_line(aes(x = count_by_group, y = thickness, group = grouping, colour = grouping)) +
  geom_smooth(aes(x = count_by_group, y = thickness, group = grouping, colour = grouping),
            method = "lm", se = FALSE) +
  theme_classic() +
  theme(legend.title=element_blank()) +
  labs(x = "run count by group", y = "film thickness")
```



3.2.2 Machine Case Study

3.2.2.1 Background and Data

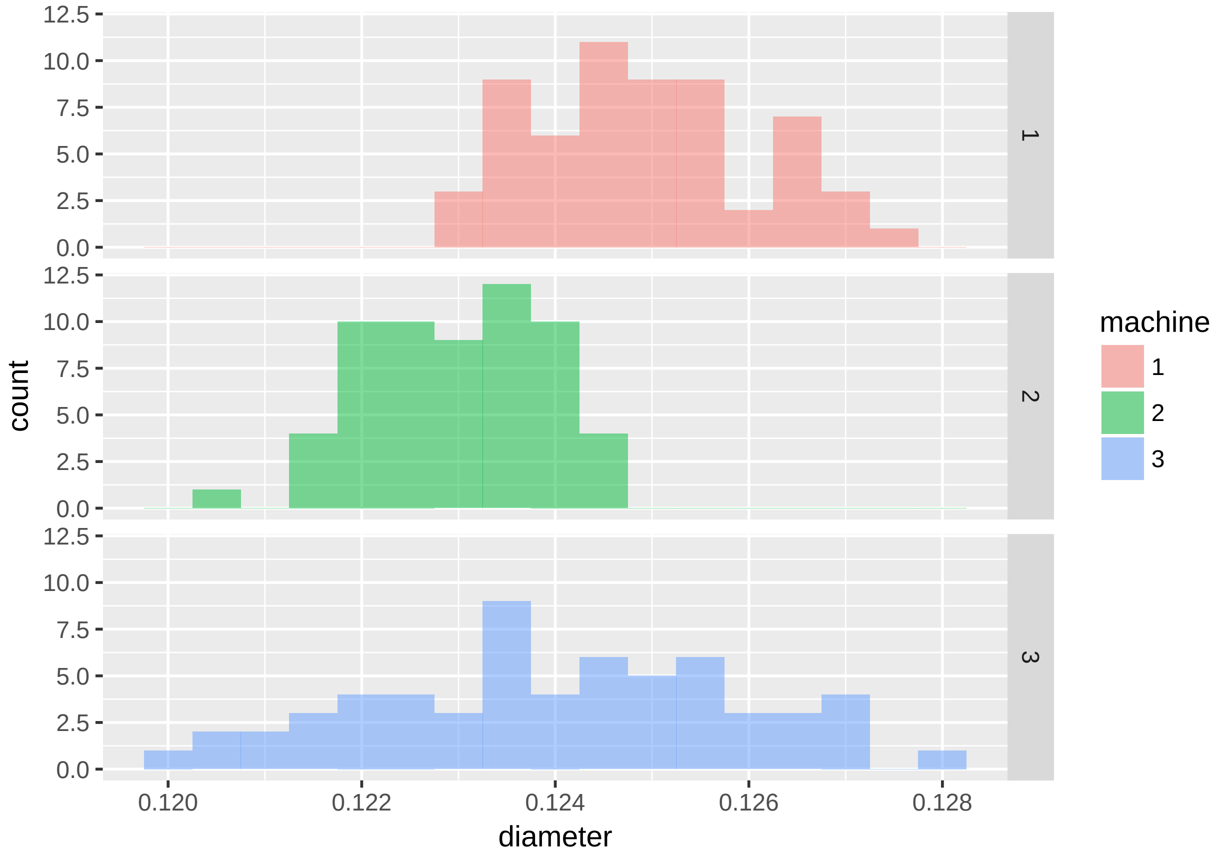
Background and Data Introduction A machine shop has three automatic screw machines that produce various parts. The shop has enough capital to replace one of the machines. The quality control department has been asked to conduct a study and make a recommendation as to which machine should be replaced. It was decided to monitor one of the most commonly produced parts (an 1/8th inch diameter pin) on each of the machines and see which machine is the least stable.

Goal The goal of this study is to determine which machine is least stable in manufacturing a steel pin with a diameter of .125 +/- .003 inches. Stability will be measured in terms of a constant variance about a constant mean. If all machines are stable, the decision will be based on process variability and throughput. Namely, the machine with the highest variability and lowest throughput will be selected for replacement.

```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   X2 = col_integer(),
##   X3 = col_integer(),
##   X4 = col_integer(),
##   X5 = col_double()
## )
```

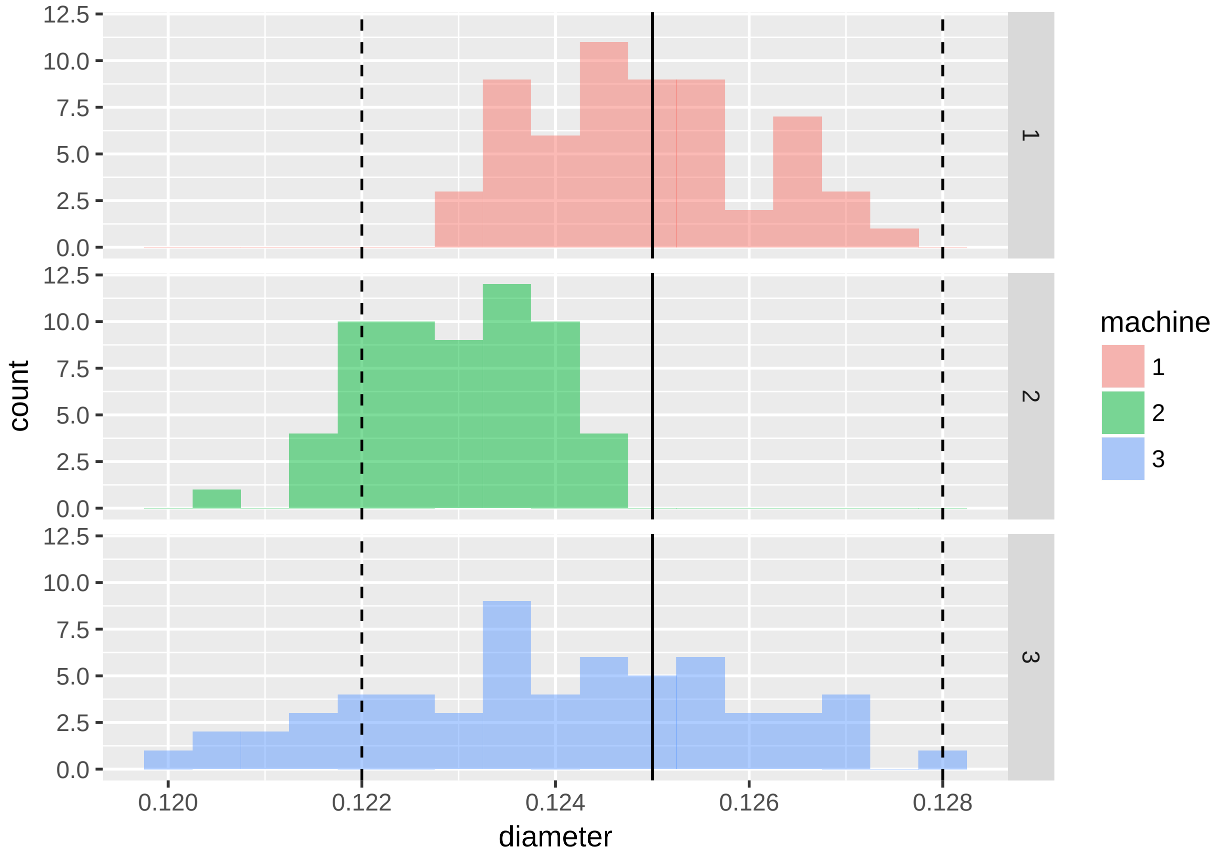
3.2.2.2 Histogram and normal probability plots of all data

```
ggplot(machine, mapping = aes(x = diameter, fill = machine)) +
  geom_histogram(binwidth = 0.0005, alpha = 0.5) +
  facet_grid(machine ~ .)
```

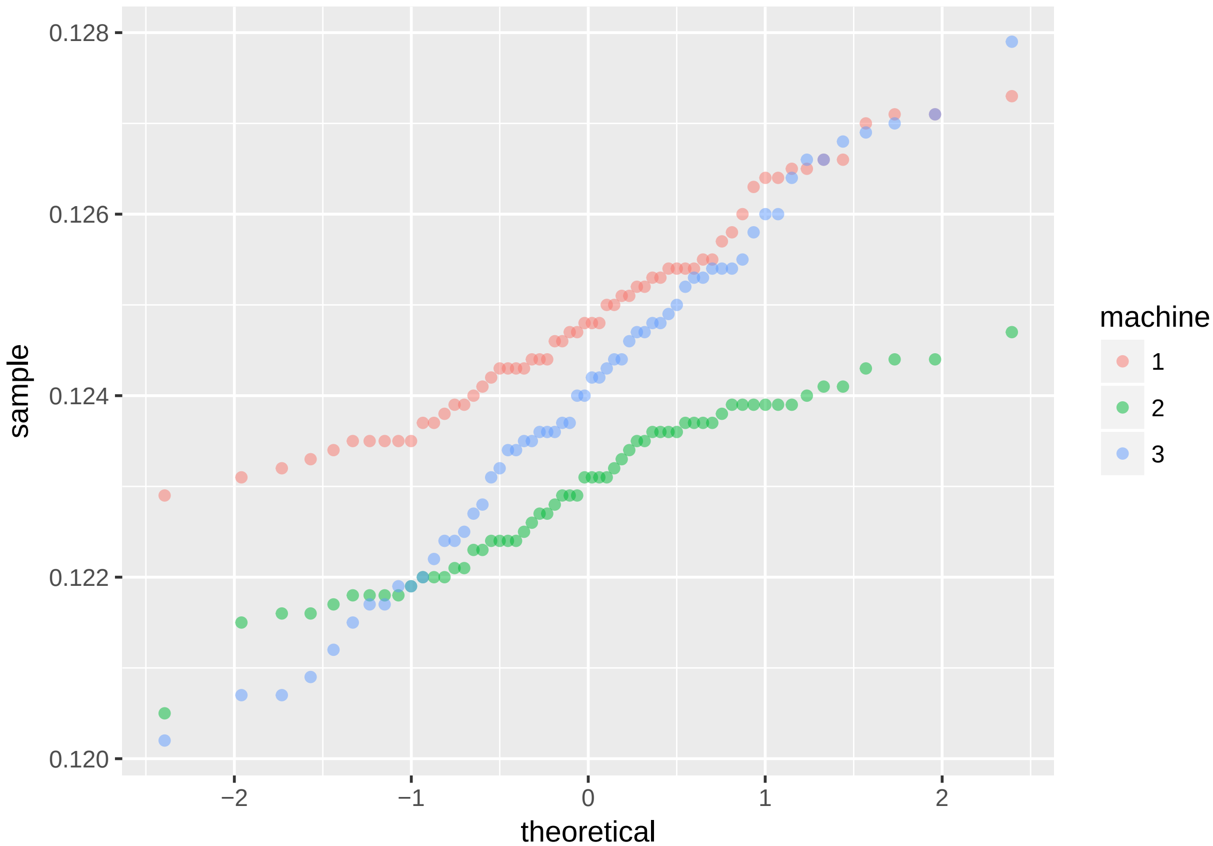


Since we are given the target diamter and tolerance, we can include these on the plot.

```
ggplot(machine, mapping = aes(x = diameter, fill = machine)) +
  geom_histogram(binwidth = 0.0005, alpha = 0.5) +
  geom_vline(aes(xintercept = 0.125)) +
  geom_vline(aes(xintercept = 0.128), linetype = 2) +
  geom_vline(aes(xintercept = 0.122), linetype = 2) +
  facet_grid(machine ~ .)
```



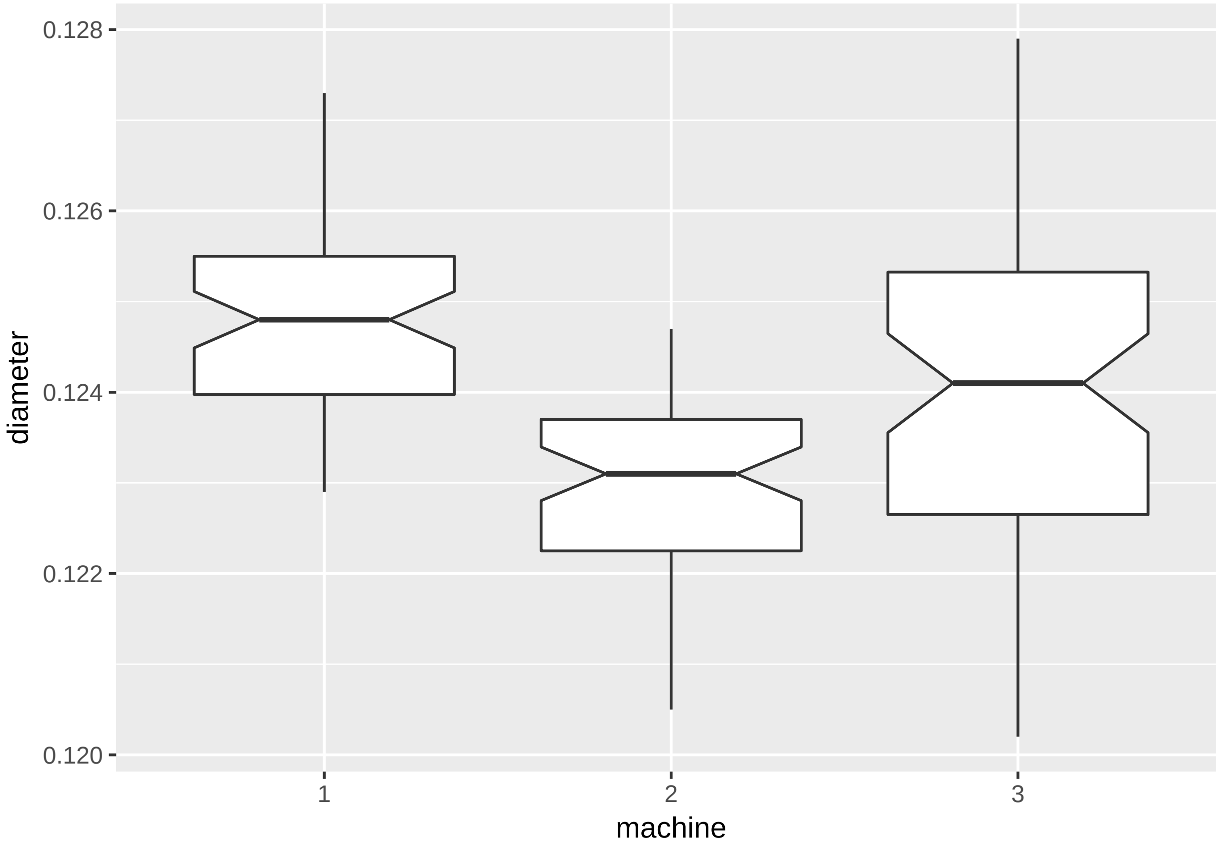

```
ggplot(machine, mapping = aes(colour = machine)) +
  geom_qq(aes(sample = diameter), alpha = 0.5)
```



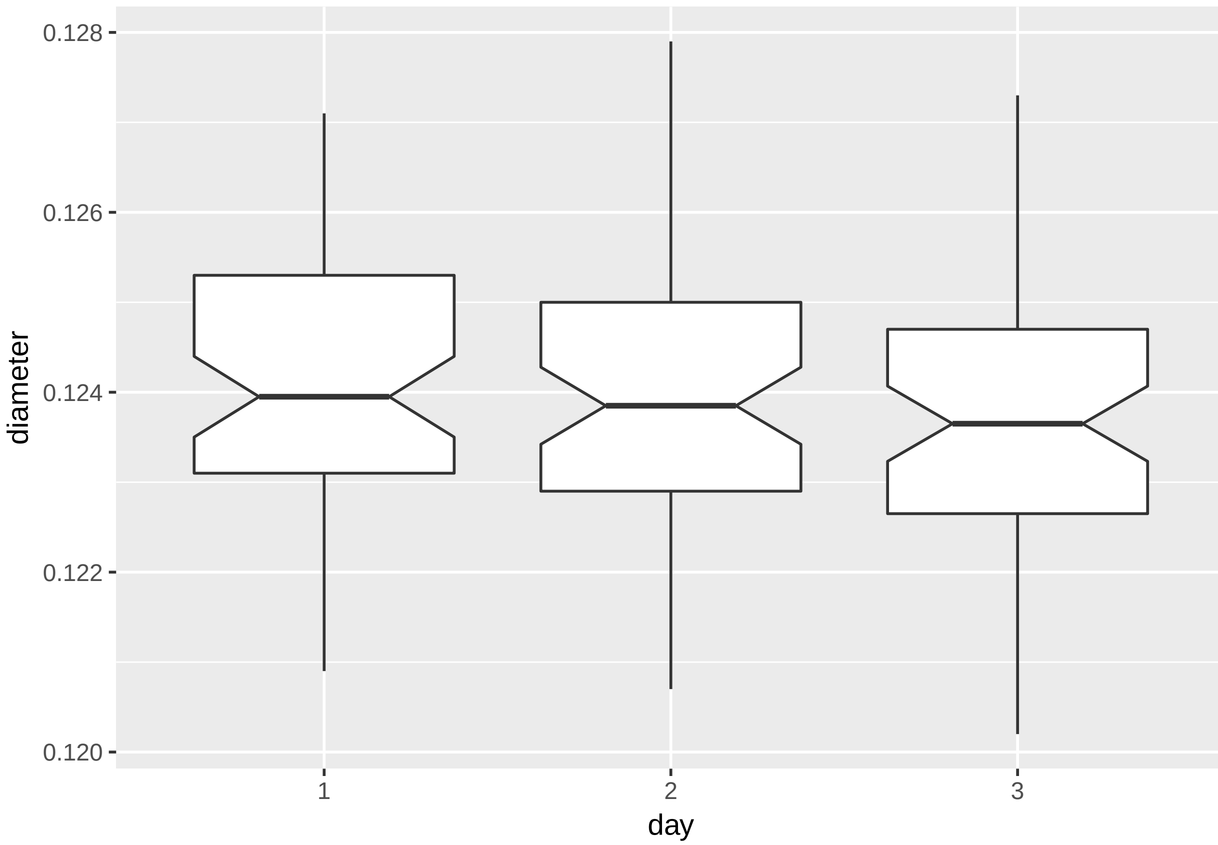
3.2.2.3 Sources of variation

3.2.2.3.1 Boxplots by factors

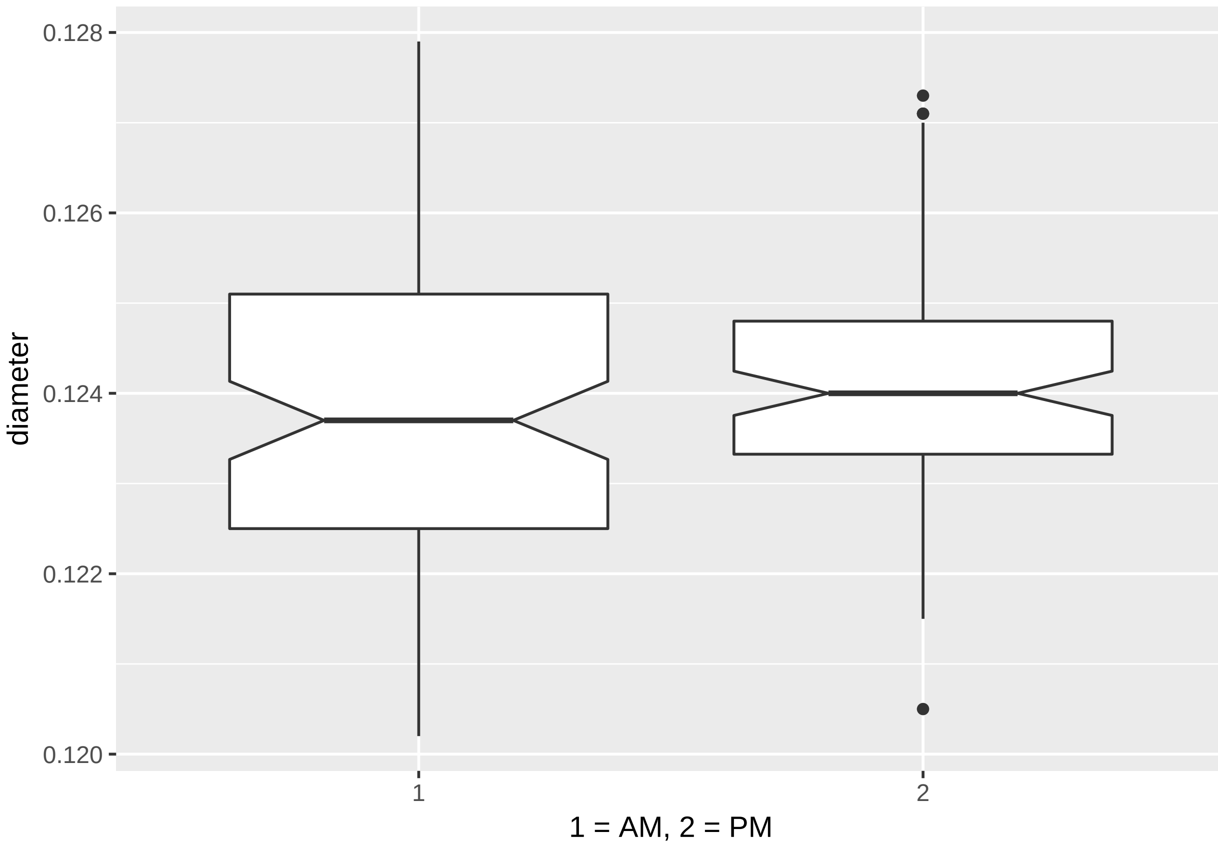
```
ggplot(data = machine, mapping = aes(x = machine, y = diameter)) +
  geom_boxplot(notch = TRUE)
```



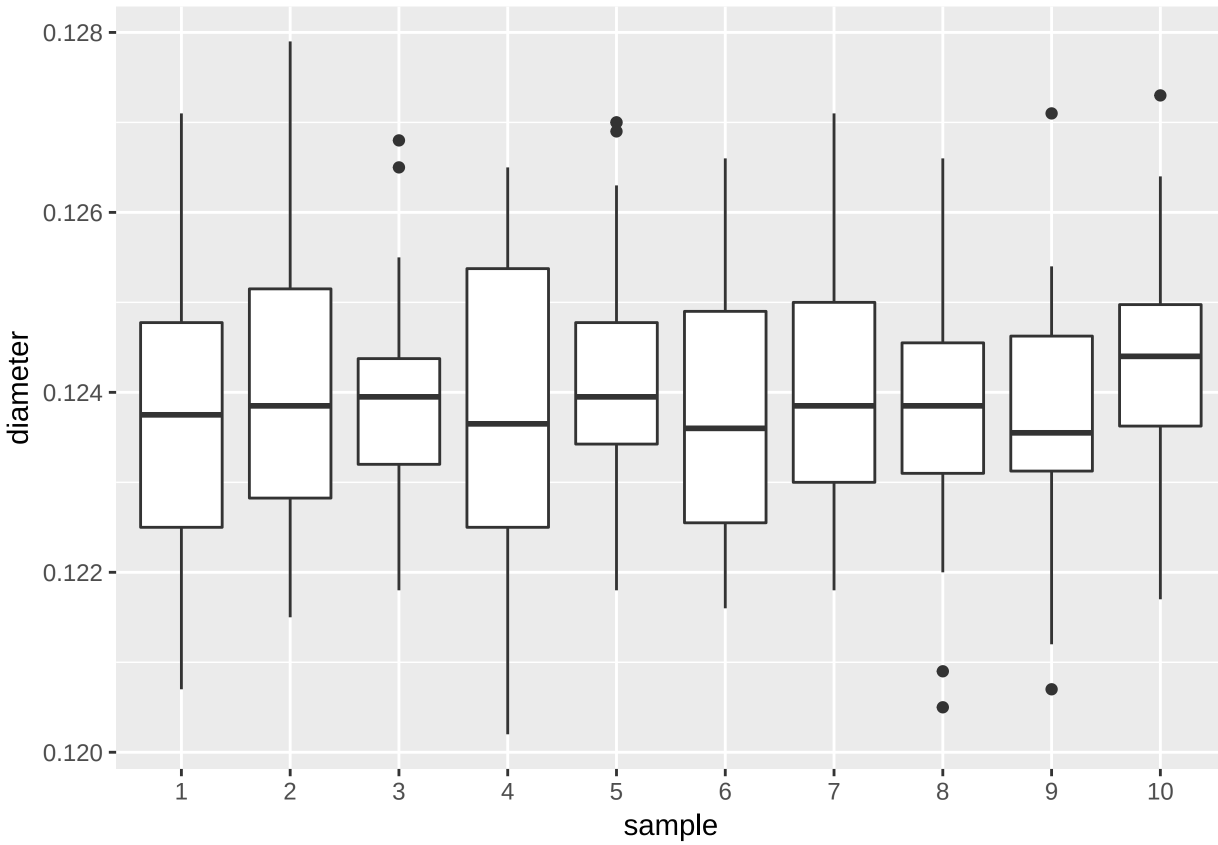

```
ggplot(data = machine, mapping = aes(x = day, y = diameter)) +
  geom_boxplot(notch = TRUE)
```




```
ggplot(data = machine, mapping = aes(x = time, y = diameter)) +
  geom_boxplot(notch = TRUE) +
  labs(x = "1 = AM, 2 = PM")
```




```
ggplot(data = machine, mapping = aes(x = sample, y = diameter)) +
  geom_boxplot(notch = FALSE)
```



3.2.2.3.2 ANOVA to confirm diameter by machine is different

```
aov.diameter <- aov(diameter ~ machine + day + time + sample, data = machine)
summary(aov.diameter)
```

```
##              Df    Sum Sq   Mean Sq F value   Pr(>F)    
## machine       2 1.107e-04 5.538e-05  29.316 1.28e-11 ***
## day           2 3.730e-06 1.870e-06   0.988    0.374    
## time          1 2.360e-06 2.360e-06   1.248    0.266    
## sample        9 8.850e-06 9.800e-07   0.521    0.858    
## Residuals   165 3.117e-04 1.890e-06                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aov.diameter.machine <- aov(diameter ~ machine, data = machine)
summary(aov.diameter.machine)
```

```
##              Df    Sum Sq   Mean Sq F value   Pr(>F)    
## machine       2 0.0001108 5.538e-05   30.01 5.99e-12 ***
## Residuals   177 0.0003266 1.850e-06                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 4

Modeling

4.1 Packages used in this chapter

```
library(tidyverse)
library(ggplot2)
library(broom)
```

4.2 Introduction

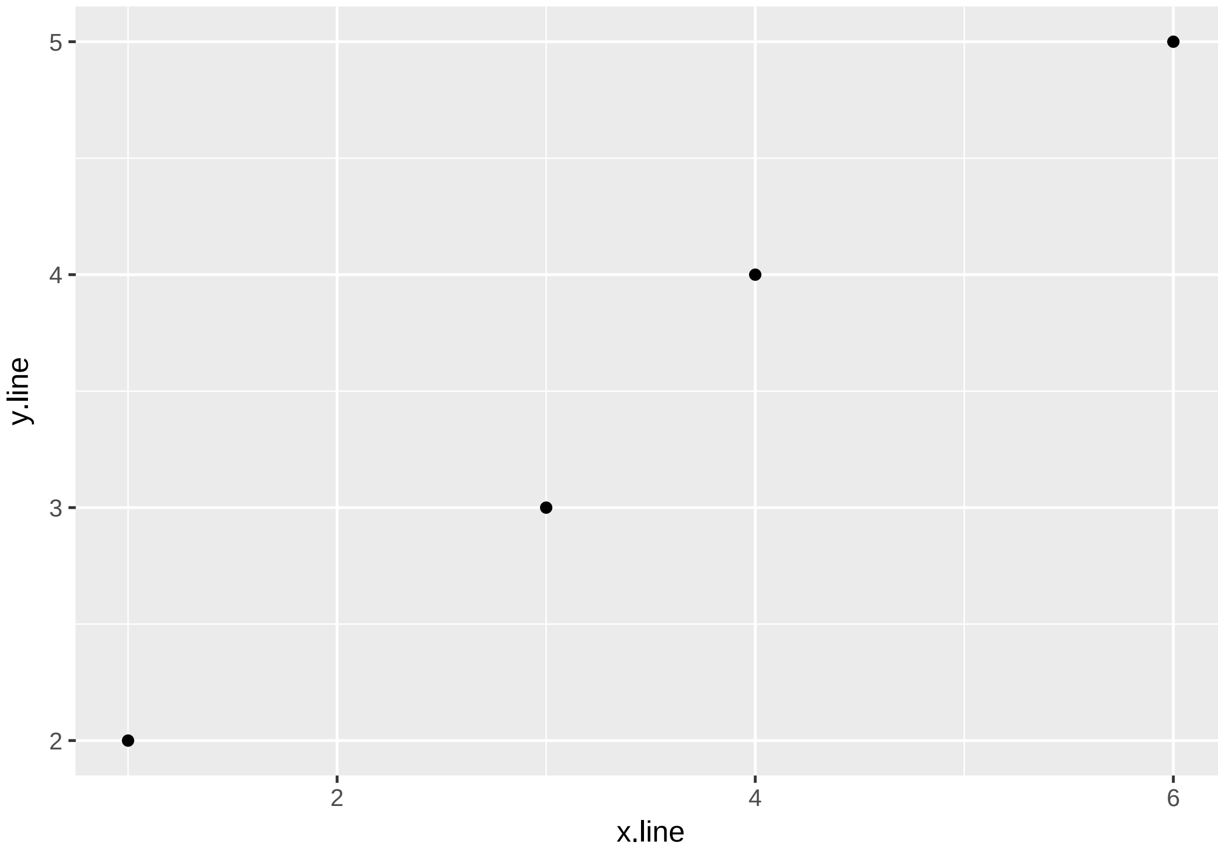
4.2.1 A simple linear regression model

```
simple_line <- tribble(
  ~x.line, ~y.line,
  1., 2.,
  3., 3.,
  4., 4.,
  6., 5.
)
simple_line
```

```
## # A tibble: 4 x 2
##   x.line y.line
##    <dbl>  <dbl>
## 1     1.     2.
## 2     3.     3.
## 3     4.     4.
## 4     6.     5.
```

4.2.1.1 Plot of the data

```
ggplot(simple_line, aes(x.line, y.line)) +
  geom_point()
```



4.2.1.2 Linear regression

Below, the data is fit to the line

$$y = mx + b$$

the intercept is assumed unless explicity removed using either y ~ x -1 or y ~ 0 + x.

```
m_sl <- lm(y.line ~ x.line,
          data = simple_line)
m_sl
```

```
##
## Call:
## lm(formula = y.line ~ x.line, data = simple_line)
##
## Coefficients:
## (Intercept)       x.line
##      1.3462       0.6154
```

```
summary(m_sl)
```

```
##
## Call:
## lm(formula = y.line ~ x.line, data = simple_line)
##
## Residuals:
```

```
##        1        2        3        4
##  0.03846 -0.19231  0.19231 -0.03846
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.34615    0.21414   6.286  0.02438 *
## x.line       0.61538    0.05439  11.314  0.00772 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1961 on 2 degrees of freedom
## Multiple R-squared:  0.9846, Adjusted R-squared:  0.9769
## F-statistic:   128 on 1 and 2 DF,  p-value: 0.007722
```

```r
tidy(m_sl)
```

```
##          term  estimate  std.error statistic     p.value
## 1 (Intercept) 1.3461538 0.21414478  6.286186 0.024384322
## 2      x.line 0.6153846 0.05439283 11.313708 0.007722123
```

```r
sl_slope <- tidy(m_sl) %>%
  filter(term == "x.line") %>%
  dplyr::select(estimate)

sl_intercept <- tidy(m_sl) %>%
  filter(term == "(Intercept)") %>%
  dplyr::select(estimate)

sl_slope
```

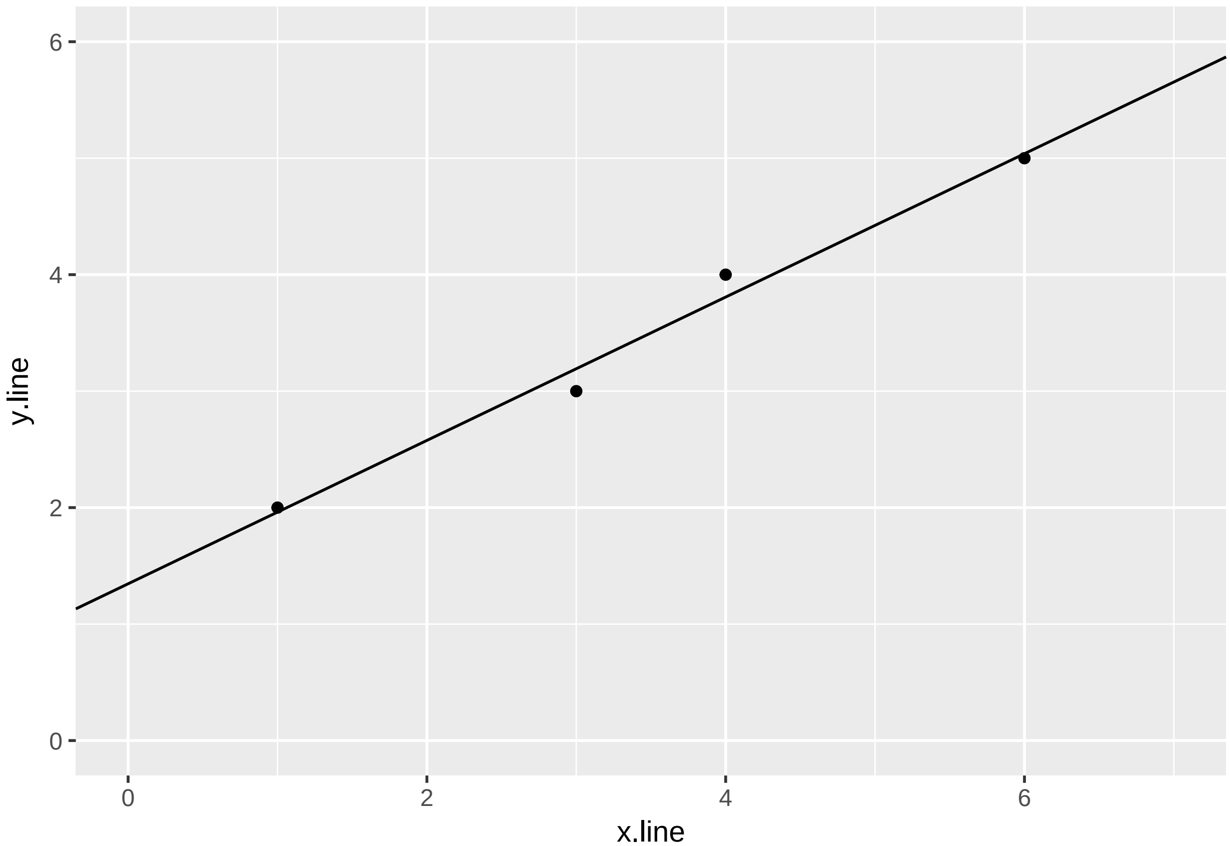
```
##    estimate
## 1 0.6153846
```

```r
sl_intercept
```

```
##   estimate
## 1 1.346154
```

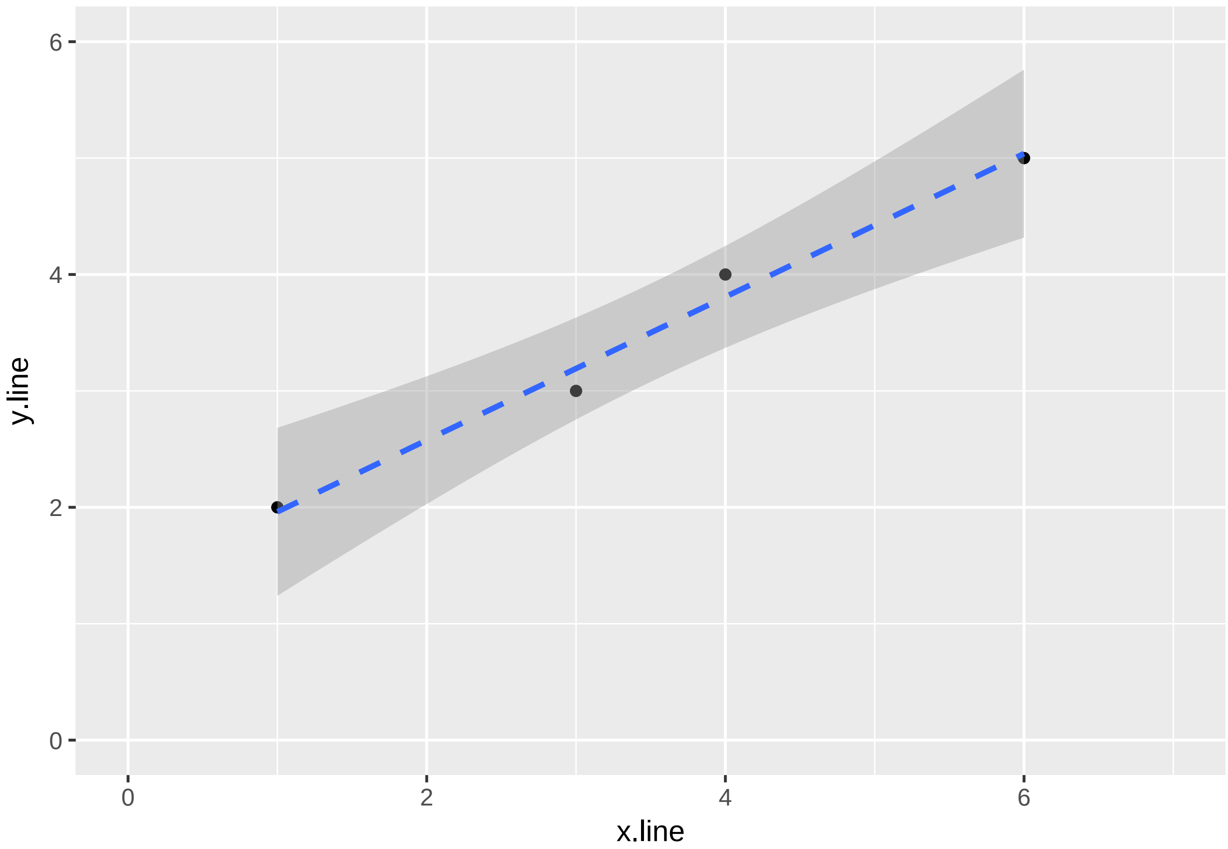
4.2.1.3 Plot of the data and linear regression

```r
ggplot(simple_line, aes(x.line, y.line)) +
  geom_point() +
  geom_abline(slope = sl_slope$estimate, intercept = sl_intercept$estimate) +
  ylim(0,6) +
  xlim(0,7)
```



If we don't need the coeficients, we can plot the data and linear regression using ggplot2

```
ggplot(simple_line, aes(x.line, y.line)) +
  geom_point() +
  stat_smooth(method = lm, linetype = "dashed") +
  ylim(0,6) +
  xlim(0,7)
```



4.2.1.4 Finally, let's add prediction intervals to the graph

```
temp_var <- m_sl %>%
  predict(interval="predict") %>%
  as_tibble()
```

```
## Warning in predict.lm(., interval = "predict"): predictions on current data refer to _future_ respons
```

```
temp_var
```

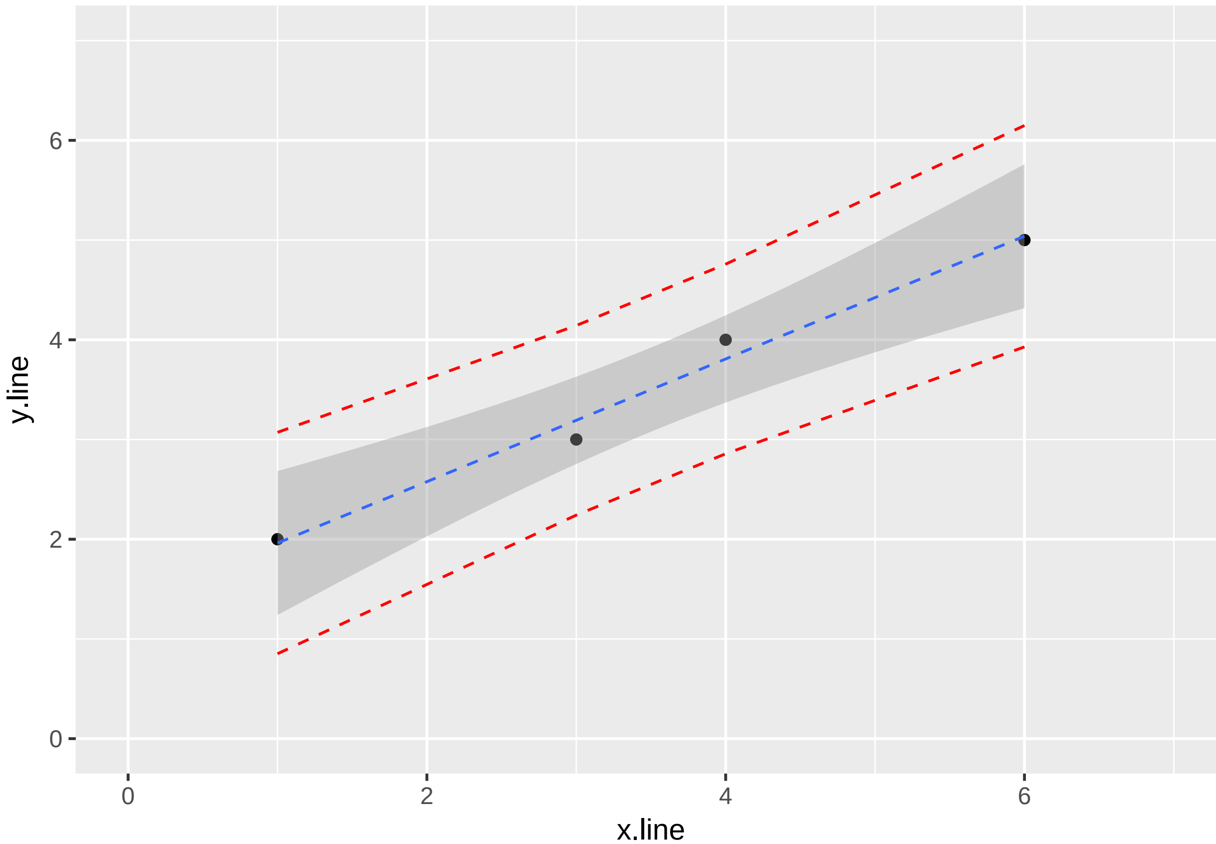
```
## # A tibble: 4 x 3
##     fit   lwr   upr
##   <dbl> <dbl> <dbl>
## 1  1.96 0.851  3.07
## 2  3.19 2.24   4.14
## 3  3.81 2.86   4.76
## 4  5.04 3.93   6.15
```

```
simple_line_predict <- bind_cols(simple_line, temp_var)
simple_line_predict
```

```
## # A tibble: 4 x 5
##   x.line y.line   fit   lwr   upr
##    <dbl>  <dbl> <dbl> <dbl> <dbl>
## 1     1.     2.  1.96 0.851  3.07
## 2     3.     3.  3.19 2.24   4.14
## 3     4.     4.  3.81 2.86   4.76
```

```
## 4     6.     5.  5.04 3.93   6.15
```

```
ggplot(simple_line_predict, aes(x.line, y.line)) +
  geom_point() +
  stat_smooth(method = lm, linetype = "dashed", size = 0.5) +
  geom_line(aes(y=lwr), color = "red", linetype = "dashed") +
  geom_line(aes(y=upr), color = "red", linetype = "dashed") +
  ylim(0,7) +
  xlim(0,7)
```



4.2.2 Beyond the linear regression

4.2.2.1 A simple data set for non-linear regression modeling—exponential decay

Example is from Brown, LeMay.

```
kinetics1 <- tribble(
  ~time, ~conc,
  0., 0.100,
  50., 0.0905,
  100., 0.0820,
  150., 0.0741,
  200., 0.0671,
  300., 0.0549,
  400., 0.0448,
  500., 0.0368,
  800., 0.0200
```

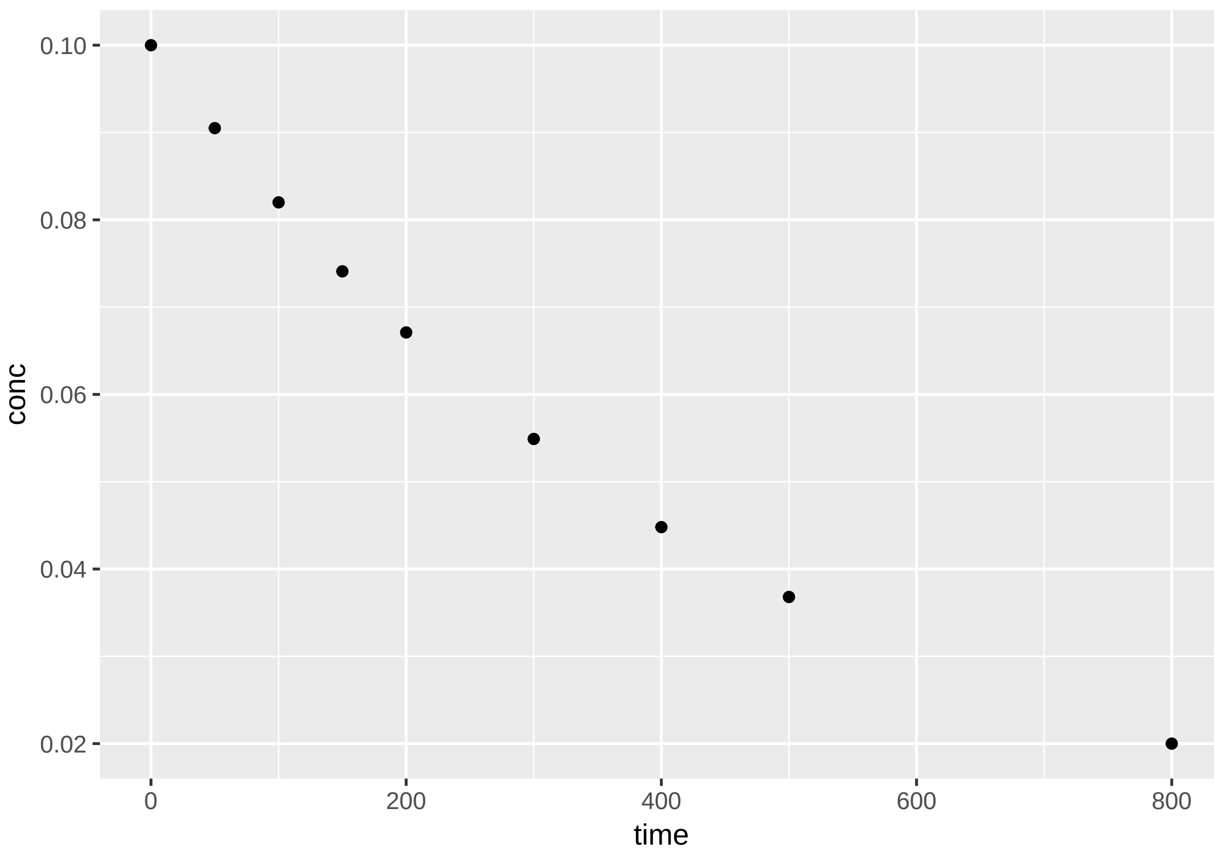
```
)
kinetics1
```

```
## # A tibble: 9 x 2
##    time   conc
##   <dbl>  <dbl>
## 1    0. 0.100
## 2   50. 0.0905
## 3  100. 0.0820
## 4  150. 0.0741
## 5  200. 0.0671
## 6  300. 0.0549
## 7  400. 0.0448
## 8  500. 0.0368
## 9  800. 0.0200
```

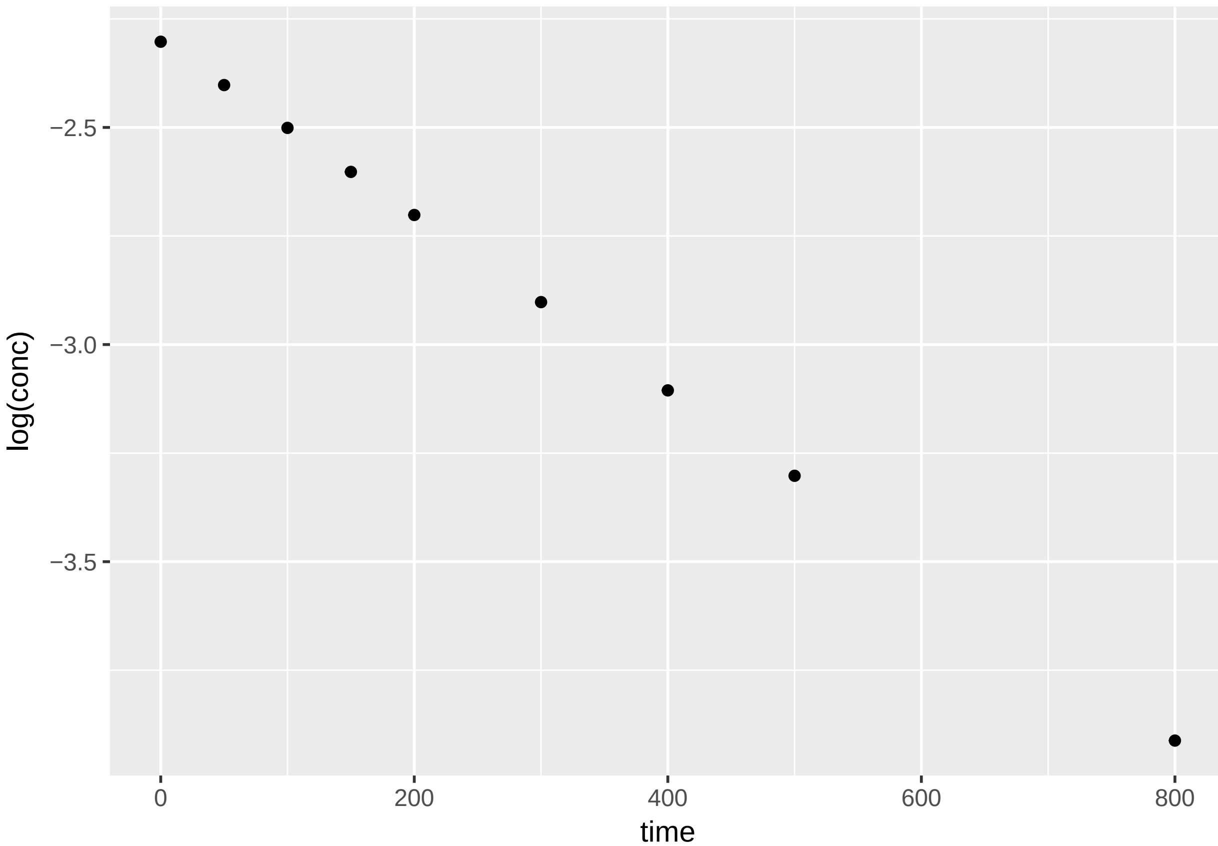
4.2.2.2 Simple plots of the data

We can plot the orginal data set, conc vs. time to view the trend. A simple test to confirm the data follows a first-order decay, we can plot log(conc) vs. time.

```
ggplot(kinetics1, aes(time, conc)) +
  geom_point()
```




```
ggplot(kinetics1, aes(time, log(conc))) +
  geom_point()
```



4.2.2.3 Using the nls function

```
k1 <- nls(conc ~ 0.1*exp(-a1*time),
          data = kinetics1, start = list(a1 = 0.002), trace = T)
```

```
## 7.545743e-08 :  0.002
## 7.088224e-08 :  0.0020017
## 7.088224e-08 :  0.002001699
```

```
summary(k1)
```

```
##
## Formula: conc ~ 0.1 * exp(-a1 * time)
##
## Parameters:
##     Estimate Std. Error t value Pr(>|t|)
## a1 2.002e-03  2.367e-06   845.8   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.413e-05 on 8 degrees of freedom
##
## Number of iterations to convergence: 2
## Achieved convergence tolerance: 2.138e-07
```

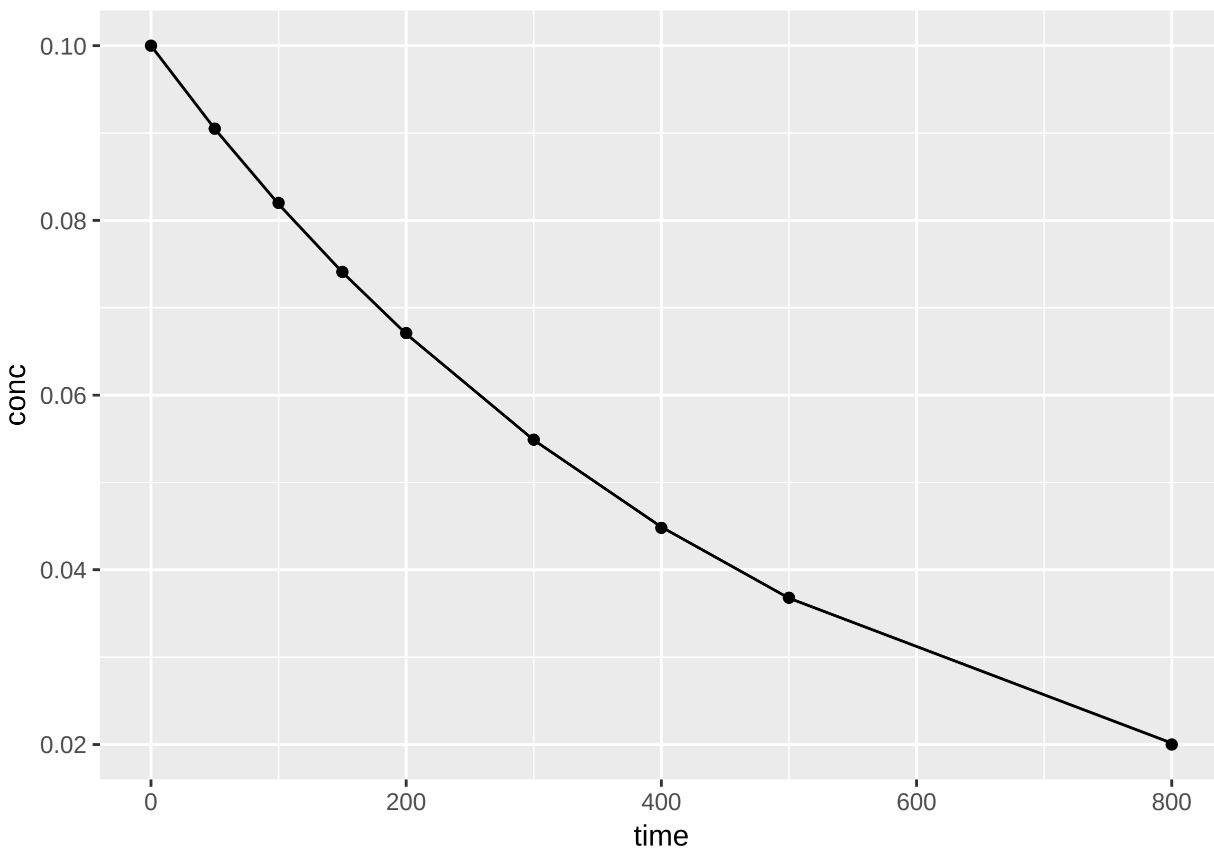
4.2.2.4 Ploting the model results

Using the `augment()` function from the **broom** package, we can plot both the data and predicted values from th `nls()` model.

```
augment(k1)
```

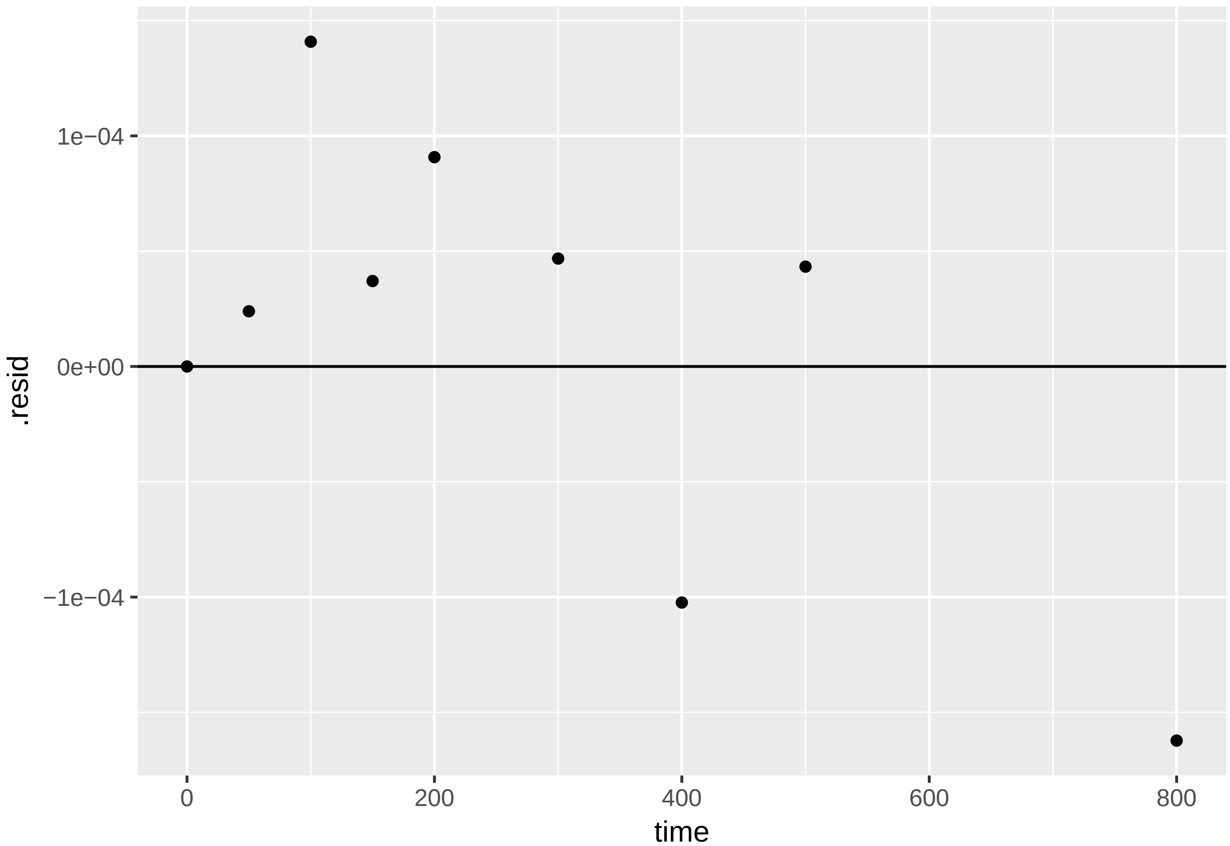
```
##   time   conc    .fitted         .resid
## 1    0 0.1000 0.10000000  0.000000e+00
## 2   50 0.0905 0.09047605  2.394510e-05
## 3  100 0.0820 0.08185917  1.408349e-04
## 4  150 0.0741 0.07406294  3.705685e-05
## 5  200 0.0671 0.06700923  9.077090e-05
## 6  300 0.0549 0.05485320  4.680452e-05
## 7  400 0.0448 0.04490237 -1.023678e-04
## 8  500 0.0368 0.03675670  4.329657e-05
## 9  800 0.0200 0.02016223 -1.622264e-04
```

```
ggplot()+
  geom_point(aes(time, conc), kinetics1) +
  geom_line(aes(time, .fitted), augment(k1))
```



We can also use the output of `augment()` to plot the residuals

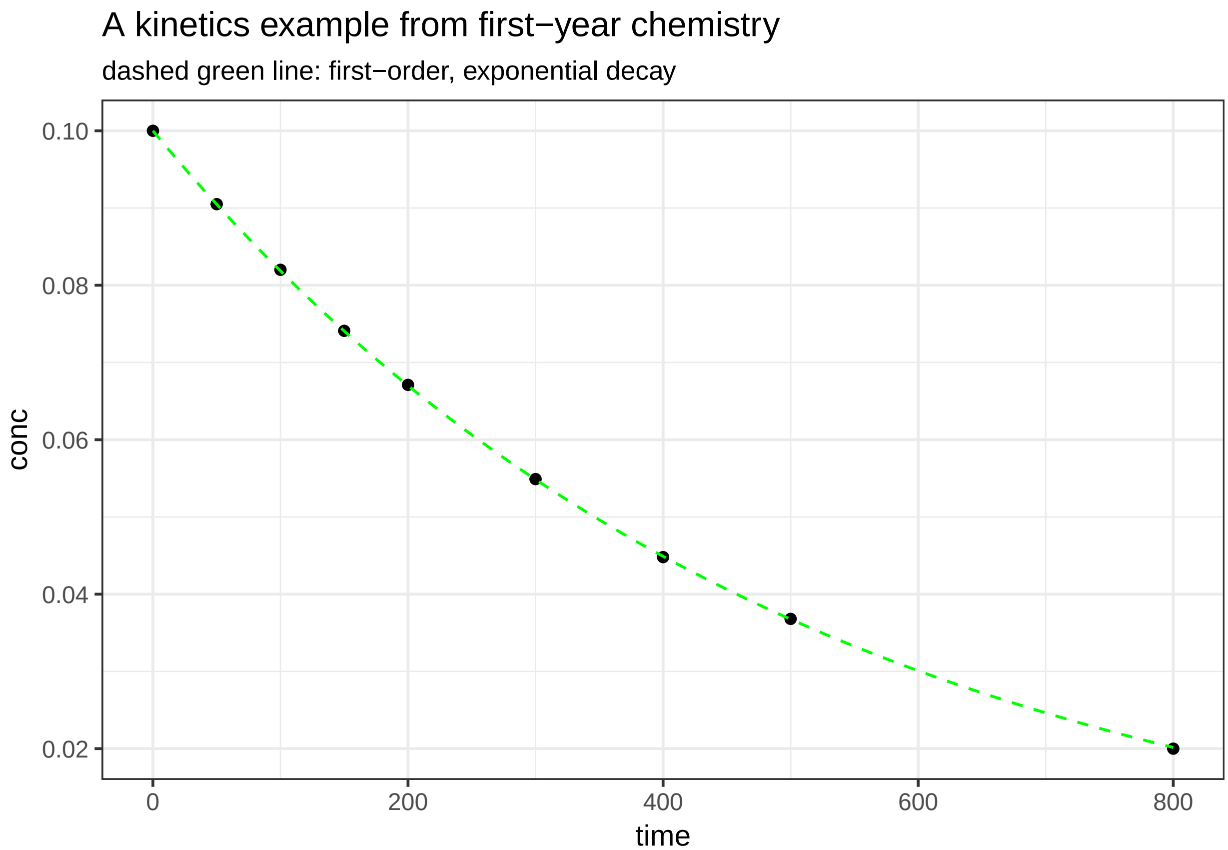
```
ggplot()+
  geom_point(aes(time, .resid), augment(k1)) +
  geom_hline(yintercept = 0)
```



4.2.2.4.1 create a function for the fit

If we want to create a smooth curve of the fit, we need to create a function and use the calculated coefficients from the `nls()` model. We can then use the `stat_function()` geom to superimpose the function on the base plot.

```
conc.fit <- function(t) {
  0.1*exp(-t*summary(k1)$coefficients[1])
  }

ggplot(kinetics1, mapping = aes(time, conc)) +
  geom_point() +
  stat_function(fun = conc.fit, linetype = "dashed", colour = "green") +
  ggtitle("A kinetics example from first-year chemistry", subtitle = "dashed green line: first-order, e
  theme_bw()
```



4.3 Case Stuidies

4.3.1 Load cell output

Load cell calibration

> The data collected in the calibration experiment consisted of a known load, applied to the load cell, and the corresponding deflection of the cell from its nominal position. Forty measurements were made over a range of loads from 150,000 to 3,000,000 units. The data were collected in two sets in order of increasing load. The systematic run order makes it difficult to determine whether or not there was any drift in the load cell or measuring equipment over time. Assuming there is no drift, however, the experiment should provide a good description of the relationship between the load applied to the cell and its response.

```
library(tidyverse)

load_cell <- read_table2(
  "NIST data/PONTIUS.dat", skip = 25, col_names = FALSE, col_types = "dd") %>%
  rename(Deflection = X1, Load = X2)
load_cell
```

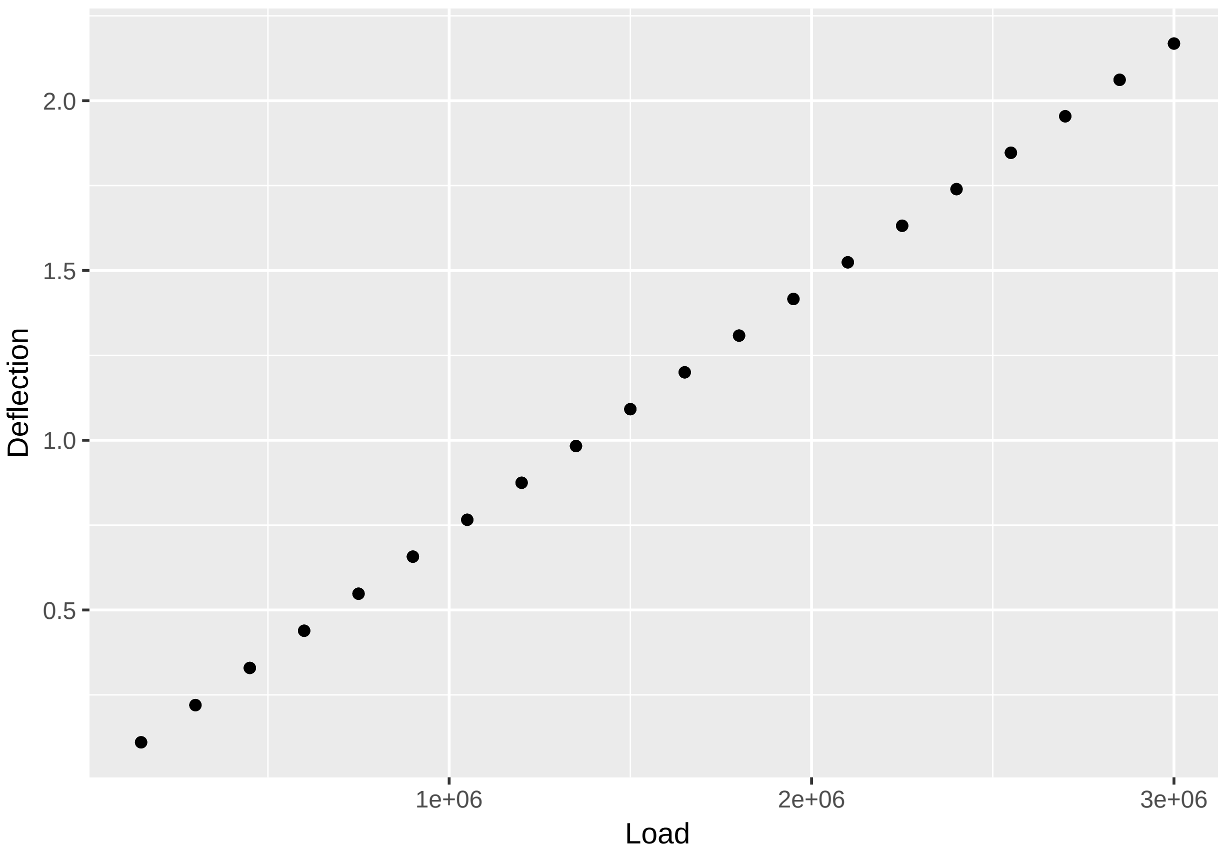
```
## # A tibble: 40 x 2
##    Deflection     Load
##         <dbl>    <dbl>
##  1      0.110  150000.
##  2      0.220  300000.
```

```
## 3      0.329  450000.
## 4      0.439  600000.
## 5      0.548  750000.
## 6      0.657  900000.
## 7      0.766 1050000.
## 8      0.875 1200000.
## 9      0.983 1350000.
## 10     1.09  1500000.
## # ... with 30 more rows
```

4.3.1.1 Selection of Inital Model

First, let's view the data.

```
ggplot(load_cell) +
  geom_point(aes(Load, Deflection))
```



The data looks linear. We can use a simple linear model to view the data

$$y = mx + b$$

```
load_cell_model <- lm(Deflection ~ Load, load_cell)
summary(load_cell_model)
```

```
##
```

```
## Call:
## lm(formula = Deflection ~ Load, data = load_cell)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -0.0042751 -0.0016308  0.0005818  0.0018932  0.0024211
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 6.150e-03  7.132e-04    8.623 1.77e-10 ***
## Load        7.221e-07  3.969e-10 1819.289  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.002171 on 38 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 3.31e+06 on 1 and 38 DF,  p-value: < 2.2e-16
```

Wow! an R-squared value of 1! it must be perfect.

4.3.1.1.1 A new package to work with summary information: broom()

broom package is part of the tidyverse and inccludes `glance()`, `tidy`, and `augment()`. These functions create tidy data frames based on the model.

```r
load_cell_glance <- glance(load_cell_model)
load_cell_glance
```

```
##   r.squared adj.r.squared       sigma statistic      p.value df  logLik
## 1 0.9999885     0.9999882 0.002171273   3309811 1.773069e-95  2 189.566
##         AIC       BIC     deviance df.residual
## 1 -373.132 -368.0654 0.0001791481          38
```

```r
load_cell_tidy <- tidy(load_cell_model)
load_cell_tidy
```

```
##          term     estimate    std.error   statistic      p.value
## 1 (Intercept) 6.149684e-03 7.132052e-04    8.622602 1.772153e-10
## 2        Load 7.221026e-07 3.969148e-10 1819.288717 1.773069e-95
```

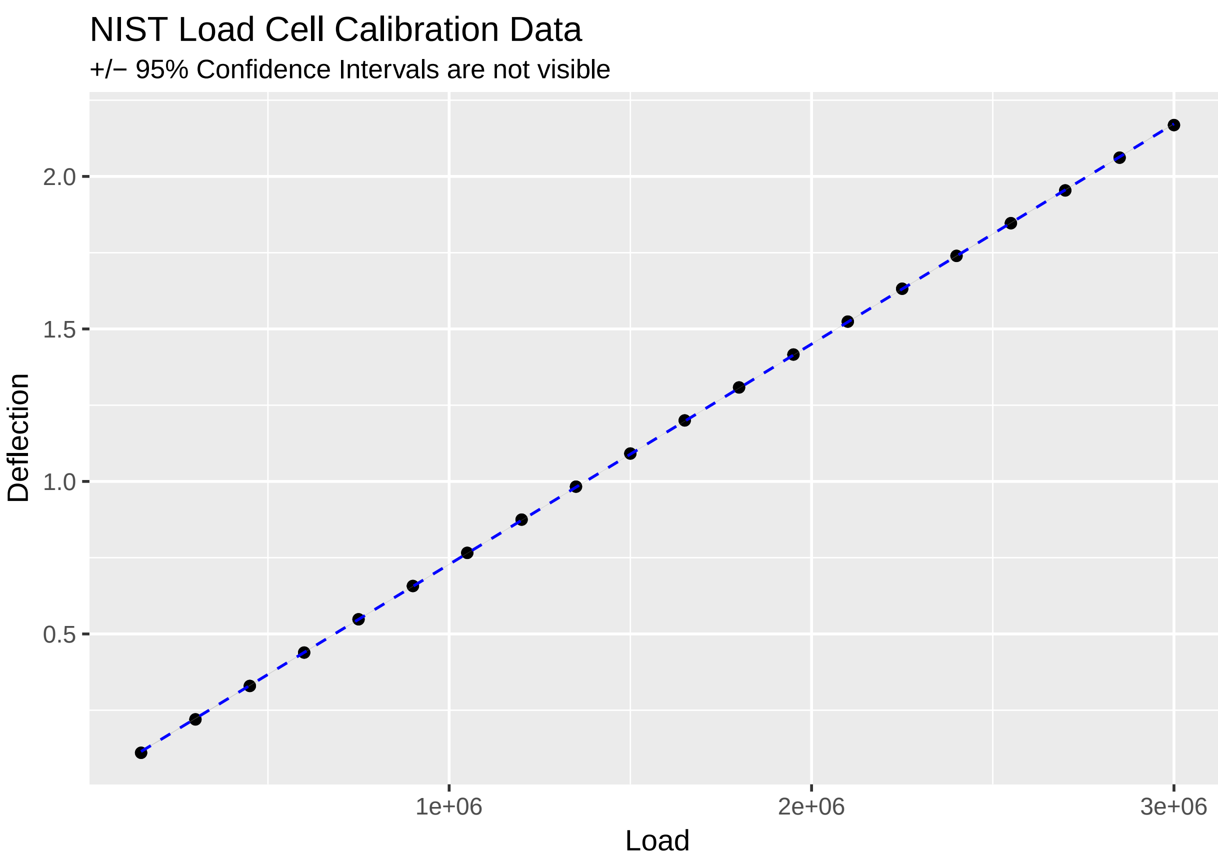
```r
augment(load_cell_model)
```

```
##    Deflection    Load   .fitted      .se.fit        .resid       .hat
## 1     0.11019  150000 0.1144651 0.0006616404 -0.0042750714 0.09285714
## 2     0.21956  300000 0.2227805 0.0006115258 -0.0032204586 0.07932331
## 3     0.32949  450000 0.3310958 0.0005632485 -0.0016058459 0.06729323
## 4     0.43899  600000 0.4394112 0.0005173233 -0.0004212331 0.05676692
## 5     0.54803  750000 0.5477266 0.0004744336  0.0003033797 0.04774436
## 6     0.65694  900000 0.6560420 0.0004354772  0.0008979925 0.04022556
## 7     0.76562 1050000 0.7643574 0.0004016005  0.0012626053 0.03421053
## 8     0.87487 1200000 0.8726728 0.0003741856  0.0021972180 0.02969925
## 9     0.98292 1350000 0.9809882 0.0003547339  0.0019318308 0.02669173
## 10    1.09146 1500000 1.0893036 0.0003445966  0.0021564436 0.02518797
## 11    1.20001 1650000 1.1976189 0.0003445966  0.0023910564 0.02518797
## 12    1.30822 1800000 1.3059343 0.0003547339  0.0022856692 0.02669173
## 13    1.41599 1950000 1.4142497 0.0003741856  0.0017402820 0.02969925
```

```
## 14    1.52399 2100000 1.5225651 0.0004016005  0.0014248947 0.03421053
## 15    1.63194 2250000 1.6308805 0.0004354772  0.0010595075 0.04022556
## 16    1.73947 2400000 1.7391959 0.0004744336  0.0002741203 0.04774436
## 17    1.84646 2550000 1.8475113 0.0005173233 -0.0010512669 0.05676692
## 18    1.95392 2700000 1.9558267 0.0005632485 -0.0019066541 0.06729323
## 19    2.06128 2850000 2.0641420 0.0006115258 -0.0028620414 0.07932331
## 20    2.16844 3000000 2.1724574 0.0006616404 -0.0040174286 0.09285714
## 21    0.11052  150000 0.1144651 0.0006616404 -0.0039450714 0.09285714
## 22    0.22018  300000 0.2227805 0.0006115258 -0.0026004586 0.07932331
## 23    0.32939  450000 0.3310958 0.0005632485 -0.0017058459 0.06729323
## 24    0.43886  600000 0.4394112 0.0005173233 -0.0005512331 0.05676692
## 25    0.54798  750000 0.5477266 0.0004744336  0.0002533797 0.04774436
## 26    0.65739  900000 0.6560420 0.0004354772  0.0013479925 0.04022556
## 27    0.76596 1050000 0.7643574 0.0004016005  0.0016026053 0.03421053
## 28    0.87474 1200000 0.8726728 0.0003741856  0.0020672180 0.02969925
## 29    0.98300 1350000 0.9809882 0.0003547339  0.0020118308 0.02669173
## 30    1.09150 1500000 1.0893036 0.0003445966  0.0021964436 0.02518797
## 31    1.20004 1650000 1.1976189 0.0003445966  0.0024210564 0.02518797
## 32    1.30818 1800000 1.3059343 0.0003547339  0.0022456692 0.02669173
## 33    1.41613 1950000 1.4142497 0.0003741856  0.0018802820 0.02969925
## 34    1.52408 2100000 1.5225651 0.0004016005  0.0015148947 0.03421053
## 35    1.63159 2250000 1.6308805 0.0004354772  0.0007095075 0.04022556
## 36    1.73965 2400000 1.7391959 0.0004744336  0.0004541203 0.04774436
## 37    1.84696 2550000 1.8475113 0.0005173233 -0.0005512669 0.05676692
## 38    1.95445 2700000 1.9558267 0.0005632485 -0.0013766541 0.06729323
## 39    2.06177 2850000 2.0641420 0.0006115258 -0.0023720414 0.07932331
## 40    2.16829 3000000 2.1724574 0.0006616404 -0.0041674286 0.09285714
##         .sigma      .cooksd .std.resid
## 1  0.002073000 0.2187217636 -2.0672413
## 2  0.002130114 0.1029350494 -1.5457875
## 3  0.002183373 0.0211558343 -0.7658028
## 4  0.002199263 0.0012007249 -0.1997554
## 5  0.002199825 0.0005139600  0.1431843
## 6  0.002195253 0.0037346550  0.4221568
## 7  0.002190258 0.0062011368  0.5917142
## 8  0.002169647 0.0161517779  1.0273197
## 9  0.002176743 0.0111520670  0.9018401
## 10 0.002170924 0.0130728091  1.0059197
## 11 0.002164101 0.0160720896  1.1153599
## 12 0.002167204 0.0156114747  1.0670231
## 13 0.002181165 0.0101324217  0.8136771
## 14 0.002187470 0.0078977192  0.6677705
## 15 0.002193224 0.0051989205  0.4980869
## 16 0.002199934 0.0004196031  0.1293749
## 17 0.002193211 0.0074786732 -0.4985275
## 18 0.002176350 0.0298240268 -0.9092535
## 19 0.002145083 0.0812979572 -1.3737509
## 20 0.002088296 0.1931530525 -1.9426563
## 21 0.002092402 0.1862580234 -1.9076675
## 22 0.002154838 0.0671162569 -1.2481938
## 23 0.002181174 0.0238727261 -0.8134912
## 24 0.002198439 0.0020562178 -0.2614035
## 25 0.002200004 0.0003585089  0.1195861
## 26 0.002188761 0.0084155020  0.6337070
```

```
## 27 0.002184026 0.0099905466  0.7510537
## 28 0.002173203 0.0142970553  0.9665376
## 29 0.002174730 0.0120948393  0.9391866
## 30 0.002169812 0.0135622837  1.0245786
## 31 0.002163176 0.0164779249  1.1293541
## 32 0.002168365 0.0150698435  1.0483498
## 33 0.002177927 0.0118282359  0.8791347
## 34 0.002185777 0.0089269075  0.7099485
## 35 0.002197195 0.0023314123  0.3335478
## 36 0.002199088 0.0011515906  0.2143284
## 37 0.002198439 0.0020564702 -0.2614196
## 38 0.002187904 0.0155479140 -0.6565048
## 39 0.002162561 0.0558434622 -1.1385558
## 40 0.002079520 0.2078459555 -2.0151899
```

```
ggplot(load_cell, aes(Load, Deflection)) +
  geom_point() +
  stat_smooth(method = lm, linetype = "dashed", colour = "blue", size = 0.5) +
  ggtitle("NIST Load Cell Calibration Data", subtitle = "+/- 95% Confidence Intervals are not visible")
```



4.3.1.2 But wait! What about the residuals?

```
load_cell_resid = resid(load_cell_model)
load_cell_resid
```

```
##            1            2            3            4            5
```

```
## -0.0042750714 -0.0032204586 -0.0016058459 -0.0004212331  0.0003033797
##             6             7             8             9            10
##  0.0008979925  0.0012626053  0.0021972180  0.0019318308  0.0021564436
##            11            12            13            14            15
##  0.0023910564  0.0022856692  0.0017402820  0.0014248947  0.0010595075
##            16            17            18            19            20
##  0.0002741203 -0.0010512669 -0.0019066541 -0.0028620414 -0.0040174286
##            21            22            23            24            25
## -0.0039450714 -0.0026004586 -0.0017058459 -0.0005512331  0.0002533797
##            26            27            28            29            30
##  0.0013479925  0.0016026053  0.0020672180  0.0020118308  0.0021964436
##            31            32            33            34            35
##  0.0024210564  0.0022456692  0.0018802820  0.0015148947  0.0007095075
##            36            37            38            39            40
##  0.0004541203 -0.0005512669 -0.0013766541 -0.0023720414 -0.0041674286
```

```
# ggplot() +
#   geom_point(aes(LoadCell$Load, LC.resid)) +
#   geom_hline(aes(yintercept=0)) +
#   geom_hline(aes(yintercept=+2*(summary(m.LC)$sigma)), linetype = "dashed") +
#   geom_hline(aes(yintercept=-2*(summary(m.LC)$sigma)), linetype = "dashed") +
#   ggtitle("Deflection Load Residuals", subtitle = "+/- 2(Residual Statndard Deviation)") +
#   theme(plot.title = element_text(hjust = 0.5), plot.subtitle = element_text(hjust = 0.5))
```

Using the **augment()** function we can plot the residuals very easily

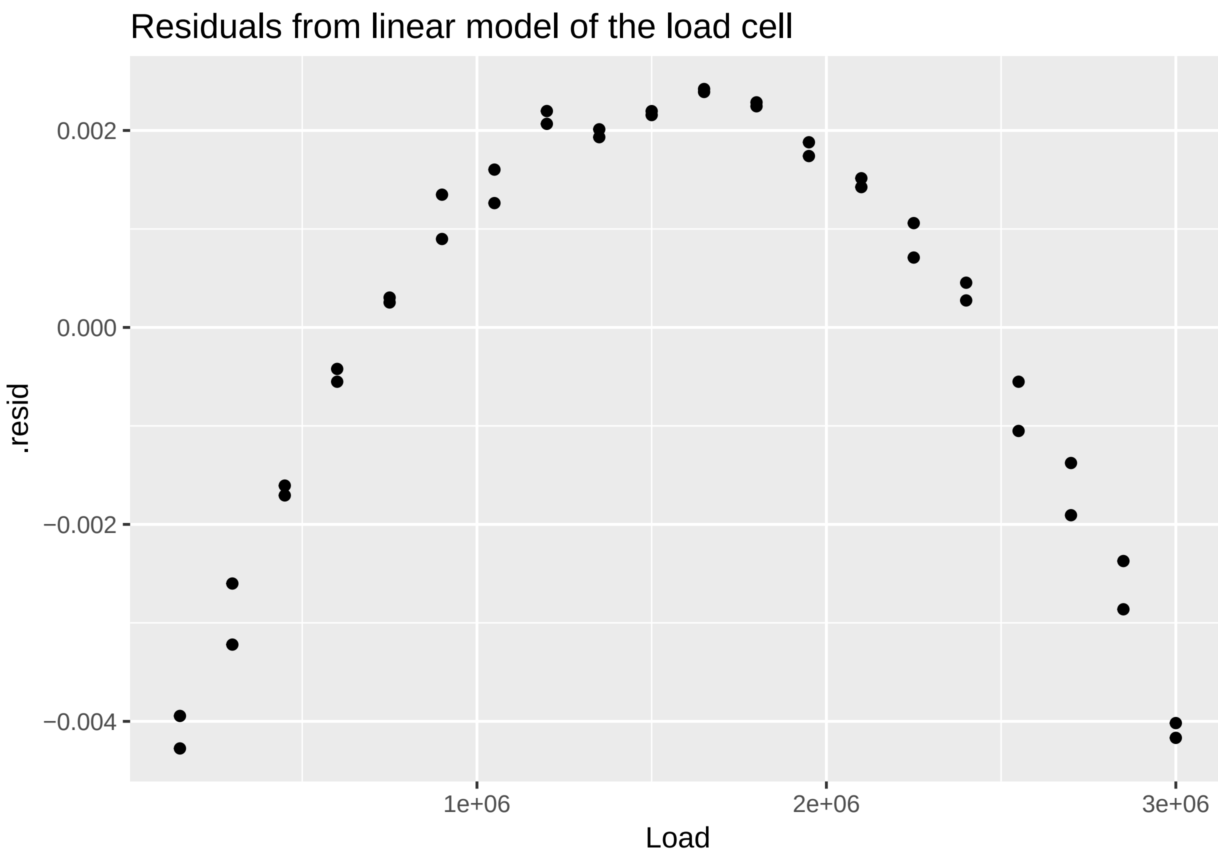
```
load_cell_fit <- augment(load_cell_model)
load_cell_fit
```

```
##    Deflection    Load   .fitted      .se.fit        .resid       .hat
## 1     0.11019  150000 0.1144651 0.0006616404 -0.0042750714 0.09285714
## 2     0.21956  300000 0.2227805 0.0006115258 -0.0032204586 0.07932331
## 3     0.32949  450000 0.3310958 0.0005632485 -0.0016058459 0.06729323
## 4     0.43899  600000 0.4394112 0.0005173233 -0.0004212331 0.05676692
## 5     0.54803  750000 0.5477266 0.0004744336  0.0003033797 0.04774436
## 6     0.65694  900000 0.6560420 0.0004354772  0.0008979925 0.04022556
## 7     0.76562 1050000 0.7643574 0.0004016005  0.0012626053 0.03421053
## 8     0.87487 1200000 0.8726728 0.0003741856  0.0021972180 0.02969925
## 9     0.98292 1350000 0.9809882 0.0003547339  0.0019318308 0.02669173
## 10    1.09146 1500000 1.0893036 0.0003445966  0.0021564436 0.02518797
## 11    1.20001 1650000 1.1976189 0.0003445966  0.0023910564 0.02518797
## 12    1.30822 1800000 1.3059343 0.0003547339  0.0022856692 0.02669173
## 13    1.41599 1950000 1.4142497 0.0003741856  0.0017402820 0.02969925
## 14    1.52399 2100000 1.5225651 0.0004016005  0.0014248947 0.03421053
## 15    1.63194 2250000 1.6308805 0.0004354772  0.0010595075 0.04022556
## 16    1.73947 2400000 1.7391959 0.0004744336  0.0002741203 0.04774436
## 17    1.84646 2550000 1.8475113 0.0005173233 -0.0010512669 0.05676692
## 18    1.95392 2700000 1.9558267 0.0005632485 -0.0019066541 0.06729323
## 19    2.06128 2850000 2.0641420 0.0006115258 -0.0028620414 0.07932331
## 20    2.16844 3000000 2.1724574 0.0006616404 -0.0040174286 0.09285714
## 21    0.11052  150000 0.1144651 0.0006616404 -0.0039450714 0.09285714
## 22    0.22018  300000 0.2227805 0.0006115258 -0.0026004586 0.07932331
## 23    0.32939  450000 0.3310958 0.0005632485 -0.0017058459 0.06729323
## 24    0.43886  600000 0.4394112 0.0005173233 -0.0005512331 0.05676692
## 25    0.54798  750000 0.5477266 0.0004744336  0.0002533797 0.04774436
```

```
## 26    0.65739  900000 0.6560420 0.0004354772  0.0013479925 0.04022556
## 27    0.76596 1050000 0.7643574 0.0004016005  0.0016026053 0.03421053
## 28    0.87474 1200000 0.8726728 0.0003741856  0.0020672180 0.02969925
## 29    0.98300 1350000 0.9809882 0.0003547339  0.0020118308 0.02669173
## 30    1.09150 1500000 1.0893036 0.0003445966  0.0021964436 0.02518797
## 31    1.20004 1650000 1.1976189 0.0003445966  0.0024210564 0.02518797
## 32    1.30818 1800000 1.3059343 0.0003547339  0.0022456692 0.02669173
## 33    1.41613 1950000 1.4142497 0.0003741856  0.0018802820 0.02969925
## 34    1.52408 2100000 1.5225651 0.0004016005  0.0015148947 0.03421053
## 35    1.63159 2250000 1.6308805 0.0004354772  0.0007095075 0.04022556
## 36    1.73965 2400000 1.7391959 0.0004744336  0.0004541203 0.04774436
## 37    1.84696 2550000 1.8475113 0.0005173233 -0.0005512669 0.05676692
## 38    1.95445 2700000 1.9558267 0.0005632485 -0.0013766541 0.06729323
## 39    2.06177 2850000 2.0641420 0.0006115258 -0.0023720414 0.07932331
## 40    2.16829 3000000 2.1724574 0.0006616404 -0.0041674286 0.09285714
##         .sigma      .cooksd .std.resid
## 1  0.002073000 0.2187217636 -2.0672413
## 2  0.002130114 0.1029350494 -1.5457875
## 3  0.002183373 0.0211558343 -0.7658028
## 4  0.002199263 0.0012007249 -0.1997554
## 5  0.002199825 0.0005139600  0.1431843
## 6  0.002195253 0.0037346550  0.4221568
## 7  0.002190258 0.0062011368  0.5917142
## 8  0.002169647 0.0161517779  1.0273197
## 9  0.002176743 0.0111520670  0.9018401
## 10 0.002170924 0.0130728091  1.0059197
## 11 0.002164101 0.0160720896  1.1153599
## 12 0.002167204 0.0156114747  1.0670231
## 13 0.002181165 0.0101324217  0.8136771
## 14 0.002187470 0.0078977192  0.6677705
## 15 0.002193224 0.0051989205  0.4980869
## 16 0.002199934 0.0004196031  0.1293749
## 17 0.002193211 0.0074786732 -0.4985275
## 18 0.002176350 0.0298240268 -0.9092535
## 19 0.002145083 0.0812979572 -1.3737509
## 20 0.002088296 0.1931530525 -1.9426563
## 21 0.002092402 0.1862580234 -1.9076675
## 22 0.002154838 0.0671162569 -1.2481938
## 23 0.002181174 0.0238727261 -0.8134912
## 24 0.002198439 0.0020562178 -0.2614035
## 25 0.002200004 0.0003585089  0.1195861
## 26 0.002188761 0.0084155020  0.6337070
## 27 0.002184026 0.0099905466  0.7510537
## 28 0.002173203 0.0142970553  0.9665376
## 29 0.002174730 0.0120948393  0.9391866
## 30 0.002169812 0.0135622837  1.0245786
## 31 0.002163176 0.0164779249  1.1293541
## 32 0.002168365 0.0150698435  1.0483498
## 33 0.002177927 0.0118282359  0.8791347
## 34 0.002185777 0.0089269075  0.7099485
## 35 0.002197195 0.0023314123  0.3335478
## 36 0.002199088 0.0011515906  0.2143284
## 37 0.002198439 0.0020564702 -0.2614196
## 38 0.002187904 0.0155479140 -0.6565048
```

```
## 39 0.002162561 0.0558434622 -1.1385558
## 40 0.002079520 0.2078459555 -2.0151899
```

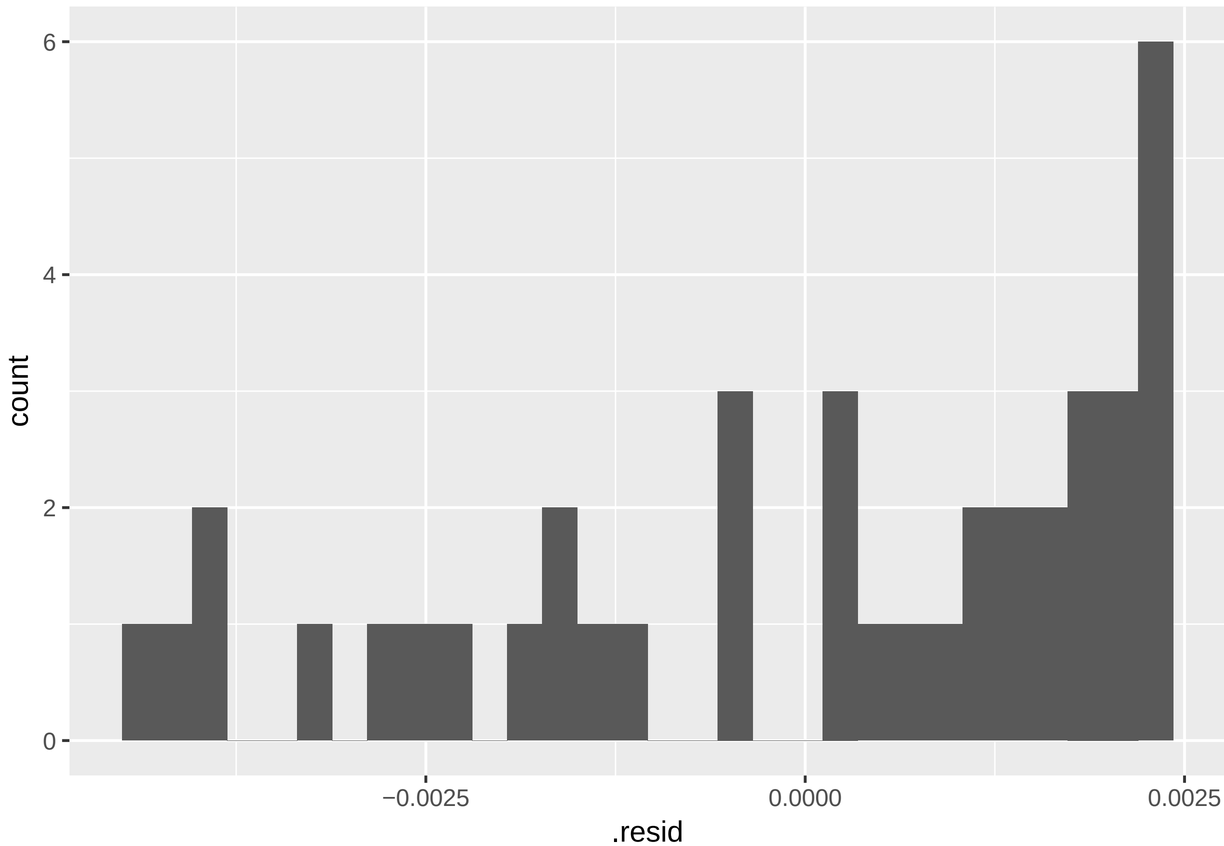
```
ggplot(load_cell_fit) +
  geom_point(aes(Load, .resid)) +
  ggtitle("Residuals from linear model of the load cell")
```



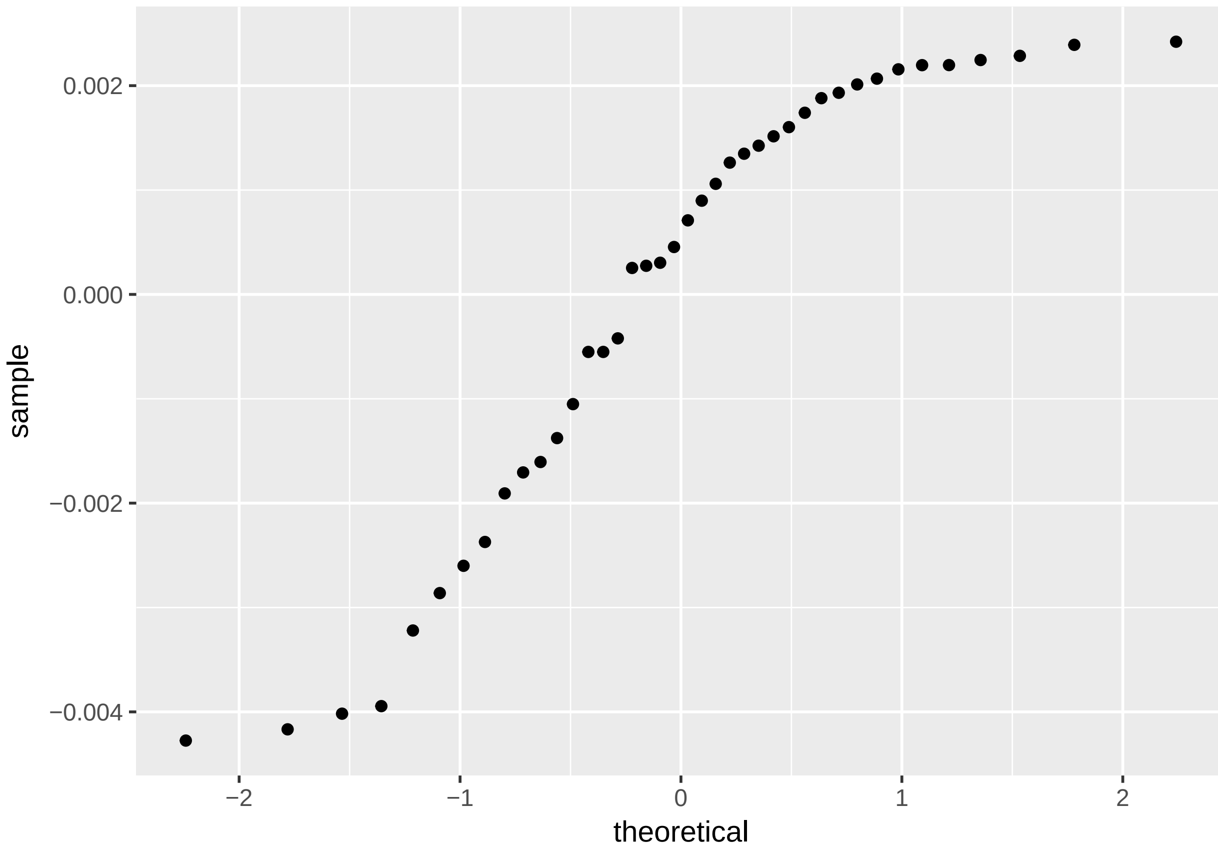
The residuals from a good model would be random. Although not necessary, we can plot a histogram or qqplot to demonstrate the residuals are not following a normal distribution.

```
ggplot(load_cell_fit) +
  geom_histogram(aes(.resid))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(load_cell_fit) +
  geom_qq(aes(sample = .resid))
```



4.3.1.3 Model Refinement

$$D = \beta_0 + \beta_1 L + \beta_2 L^2 + \varepsilon$$

We can use the linear model function, `lm()`, by creating a new variable L^2.

```
load_cell_2 <- mutate(load_cell, Load_squared = Load^2)
load_cell_2
```

```
## # A tibble: 40 x 3
##    Deflection     Load Load_squared
##         <dbl>    <dbl>        <dbl>
##  1      0.110  150000.      2.25e10
##  2      0.220  300000.      9.00e10
##  3      0.329  450000.      2.02e11
##  4      0.439  600000.      3.60e11
##  5      0.548  750000.      5.62e11
##  6      0.657  900000.      8.10e11
##  7      0.766 1050000.      1.10e12
##  8      0.875 1200000.      1.44e12
##  9      0.983 1350000.      1.82e12
## 10      1.09  1500000.      2.25e12
## # ... with 30 more rows
```

```
load_cell_model_2 <- lm(Deflection ~ Load + Load_squared, load_cell_2)
summary(load_cell_model_2)
```

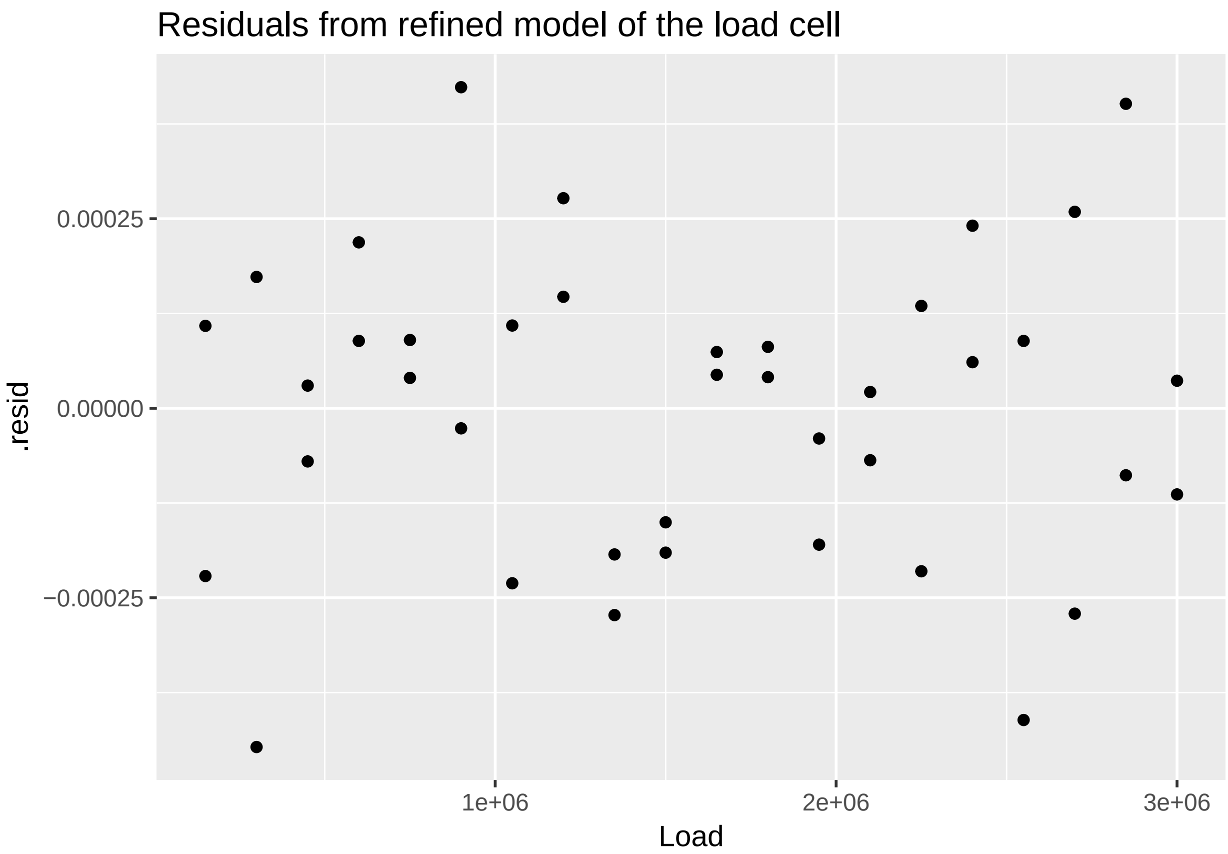
```
##
## Call:
## lm(formula = Deflection ~ Load + Load_squared, data = load_cell_2)
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -4.468e-04 -1.578e-04  3.817e-05  1.088e-04  4.235e-04
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   6.736e-04  1.079e-04    6.24 2.97e-07 ***
## Load          7.321e-07  1.578e-10 4638.65  < 2e-16 ***
## Load_squared -3.161e-15  4.867e-17  -64.95  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0002052 on 37 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.853e+08 on 2 and 37 DF,  p-value: < 2.2e-16
```

```
load_cell_fit_2 <- augment(load_cell_model_2)
load_cell_fit_2
```

```
##    Deflection    Load Load_squared   .fitted      .se.fit         .resid
## 1     0.11019  150000   2.2500e+10 0.1104113 8.834303e-05 -2.213214e-04
## 2     0.21956  300000   9.0000e+10 0.2200068 7.185366e-05 -4.468402e-04
## 3     0.32949  450000   2.0250e+11 0.3294601 5.888246e-05  2.987782e-05
## 4     0.43899  600000   3.6000e+11 0.4387712 4.986859e-05  2.188327e-04
## 5     0.54803  750000   5.6250e+11 0.5479400 4.495244e-05  9.002444e-05
## 6     0.65694  900000   8.1000e+11 0.6569665 4.354343e-05 -2.654699e-05
## 7     0.76562 1050000   1.1025e+12 0.7658509 4.437259e-05 -2.308816e-04
## 8     0.87487 1200000   1.4400e+12 0.8745930 4.609026e-05  2.770207e-04
## 9     0.98292 1350000   1.8225e+12 0.9831928 4.770597e-05 -2.728402e-04
## 10    1.09146 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.904643e-04
## 11    1.20001 1650000   2.7225e+12 1.1999659 4.864177e-05  4.414850e-05
## 12    1.30822 1800000   3.2400e+12 1.3081390 4.770597e-05  8.099812e-05
## 13    1.41599 1950000   3.8025e+12 1.4161699 4.609026e-05 -1.799154e-04
## 14    1.52399 2100000   4.4100e+12 1.5240586 4.437259e-05 -6.859211e-05
## 15    1.63194 2250000   5.0625e+12 1.6318050 4.354343e-05  1.349680e-04
## 16    1.73947 2400000   5.7600e+12 1.7394092 4.495244e-05  6.076504e-05
## 17    1.84646 2550000   6.5025e+12 1.8468712 4.986859e-05 -4.112011e-04
## 18    1.95392 2700000   7.2900e+12 1.9541909 5.888246e-05 -2.709305e-04
## 19    2.06128 2850000   8.1225e+12 2.0613684 7.185366e-05 -8.842293e-05
## 20    2.16844 3000000   9.0000e+12 2.1684037 8.834303e-05  3.632143e-05
## 21    0.11052  150000   2.2500e+10 0.1104113 8.834303e-05  1.086786e-04
## 22    0.22018  300000   9.0000e+10 0.2200068 7.185366e-05  1.731598e-04
## 23    0.32939  450000   2.0250e+11 0.3294601 5.888246e-05 -7.012218e-05
## 24    0.43886  600000   3.6000e+11 0.4387712 4.986859e-05  8.883271e-05
## 25    0.54798  750000   5.6250e+11 0.5479400 4.495244e-05  4.002444e-05
## 26    0.65739  900000   8.1000e+11 0.6569665 4.354343e-05  4.234530e-04
## 27    0.76596 1050000   1.1025e+12 0.7658509 4.437259e-05  1.091184e-04
## 28    0.87474 1200000   1.4400e+12 0.8745930 4.609026e-05  1.470207e-04
## 29    0.98300 1350000   1.8225e+12 0.9831928 4.770597e-05 -1.928402e-04
## 30    1.09150 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.504643e-04
## 31    1.20004 1650000   2.7225e+12 1.1999659 4.864177e-05  7.414850e-05
```

```
## 32    1.30818 1800000   3.2400e+12 1.3081390 4.770597e-05  4.099812e-05
## 33    1.41613 1950000   3.8025e+12 1.4161699 4.609026e-05 -3.991541e-05
## 34    1.52408 2100000   4.4100e+12 1.5240586 4.437259e-05  2.140789e-05
## 35    1.63159 2250000   5.0625e+12 1.6318050 4.354343e-05 -2.150320e-04
## 36    1.73965 2400000   5.7600e+12 1.7394092 4.495244e-05  2.407650e-04
## 37    1.84696 2550000   6.5025e+12 1.8468712 4.986859e-05  8.879887e-05
## 38    1.95445 2700000   7.2900e+12 1.9541909 5.888246e-05  2.590695e-04
## 39    2.06177 2850000   8.1225e+12 2.0613684 7.185366e-05  4.015771e-04
## 40    2.16829 3000000   9.0000e+12 2.1684037 8.834303e-05 -1.136786e-04
##          .hat       .sigma      .cooksd .std.resid
## 1  0.18538961 0.0002039531 0.1083557670 -1.1951404
## 2  0.12264183 0.0001922123 0.2518888277 -2.3250603
## 3  0.08235931 0.0002079426 0.0006913290  0.1520138
## 4  0.05907382 0.0002045811 0.0253004370  1.0995244
## 5  0.04800068 0.0002074384 0.0033987136  0.4496893
## 6  0.04503873 0.0002079583 0.0002755909 -0.1324015
## 7  0.04677033 0.0002042395 0.0217256885 -1.1525531
## 8  0.05046138 0.0002025394 0.0340077449  1.3855631
## 9  0.05406129 0.0002026849 0.0356120360 -1.3672481
## 10 0.05620301 0.0002054251 0.0181237925 -0.9555308
## 11 0.05620301 0.0002078697 0.0009737642  0.2214864
## 12 0.05406129 0.0002075440 0.0031385560  0.4058952
## 13 0.05046138 0.0002057188 0.0143446582 -0.8998756
## 14 0.04677033 0.0002076778 0.0019175348 -0.3424095
## 15 0.04503873 0.0002067300 0.0071235449  0.6731448
## 16 0.04800068 0.0002077485 0.0015484646  0.3035330
## 17 0.05907382 0.0001956411 0.0893330479 -2.0660791
## 18 0.08235931 0.0002025961 0.0568463516 -1.3784528
## 19 0.12264183 0.0002074117 0.0098635717 -0.4600943
## 20 0.18538961 0.0002078994 0.0029183068  0.1961365
## 21 0.18538961 0.0002070372 0.0261272044  0.5868666
## 22 0.12264183 0.0002057130 0.0378266955  0.9010087
## 23 0.08235931 0.0002076495 0.0038080074 -0.3567710
## 24 0.05907382 0.0002074468 0.0041691675  0.4463397
## 25 0.04800068 0.0002078952 0.0006718065  0.1999297
## 26 0.04503873 0.0001950675 0.0701204501  2.1119458
## 27 0.04677033 0.0002071719 0.0048527858  0.5447155
## 28 0.05046138 0.0002064820 0.0095787822  0.7353473
## 29 0.05406129 0.0002053659 0.0177899720 -0.9663547
## 30 0.05620301 0.0002063997 0.0113106830 -0.7548568
## 31 0.05620301 0.0002076183 0.0027467977  0.3719919
## 32 0.05406129 0.0002078889 0.0008040960  0.2054485
## 33 0.05046138 0.0002078955 0.0007060488 -0.1996433
## 34 0.04677033 0.0002079755 0.0001867854  0.1068675
## 35 0.04503873 0.0002047490 0.0180817416 -1.0724586
## 36 0.04800068 0.0002039013 0.0243097555  1.2026674
## 37 0.05907382 0.0002074473 0.0041659922  0.4461697
## 38 0.08235931 0.0002030652 0.0519780155  1.3181064
## 39 0.12264183 0.0001953495 0.2034427365  2.0895409
## 40 0.18538961 0.0002069456 0.0285865876 -0.6138667
```

```r
ggplot(load_cell_fit_2) +
  geom_point(aes(Load, .resid)) +
  ggtitle("Residuals from refined model of the load cell")
```



4.3.1.3.1 Could we have used a non-linear least squares fit model?

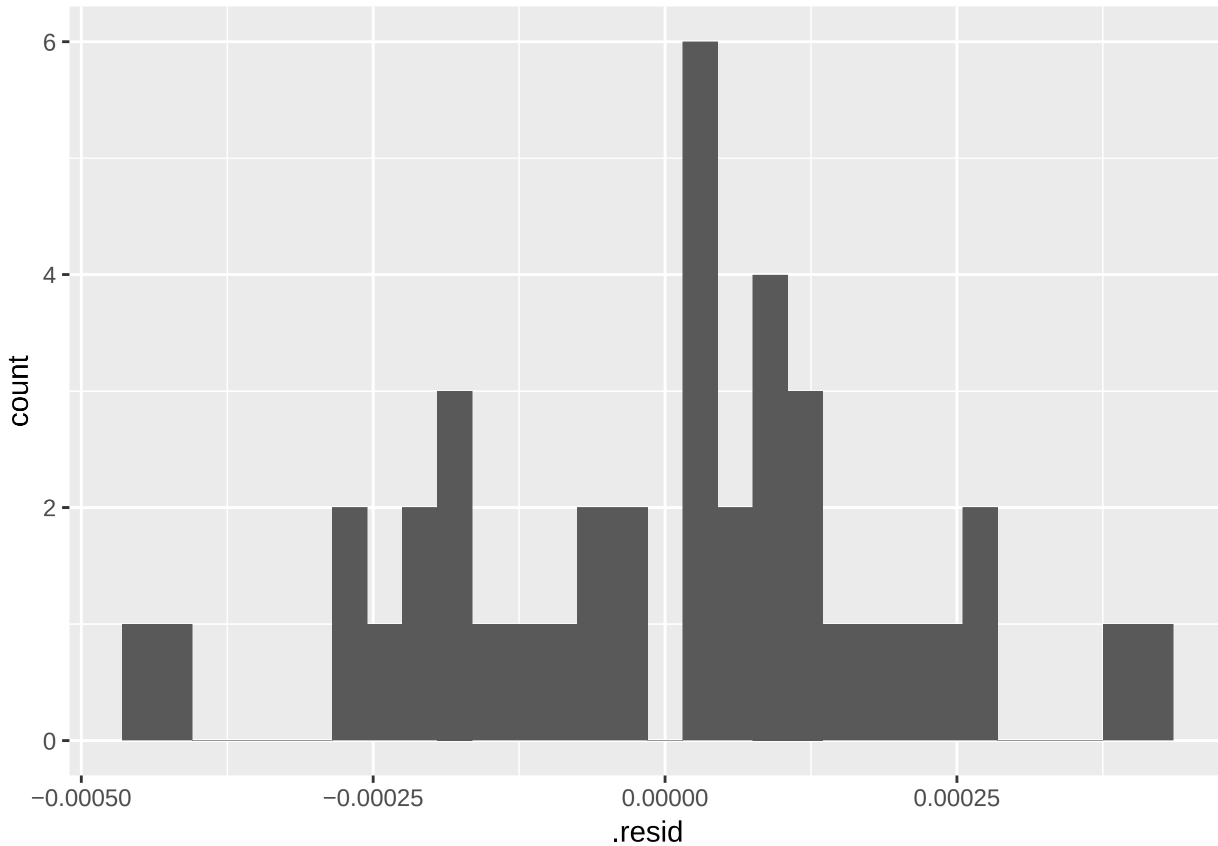
```
load_cell_model_3 <- nls(Deflection ~ b0 + b1*Load + b2*Load^2, load_cell_2, start = c(b0 = 0, b1 = 0,b2

summary(load_cell_model_3)
```

```
##
## Formula: Deflection ~ b0 + b1 * Load + b2 * Load^2
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## b0  6.736e-04  1.079e-04    6.24 2.97e-07 ***
## b1  7.321e-07  1.578e-10 4638.65  < 2e-16 ***
## b2 -3.161e-15  4.867e-17  -64.95  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0002052 on 37 degrees of freedom
##
## Number of iterations to convergence: 1
## Achieved convergence tolerance: 1.328e-06
```

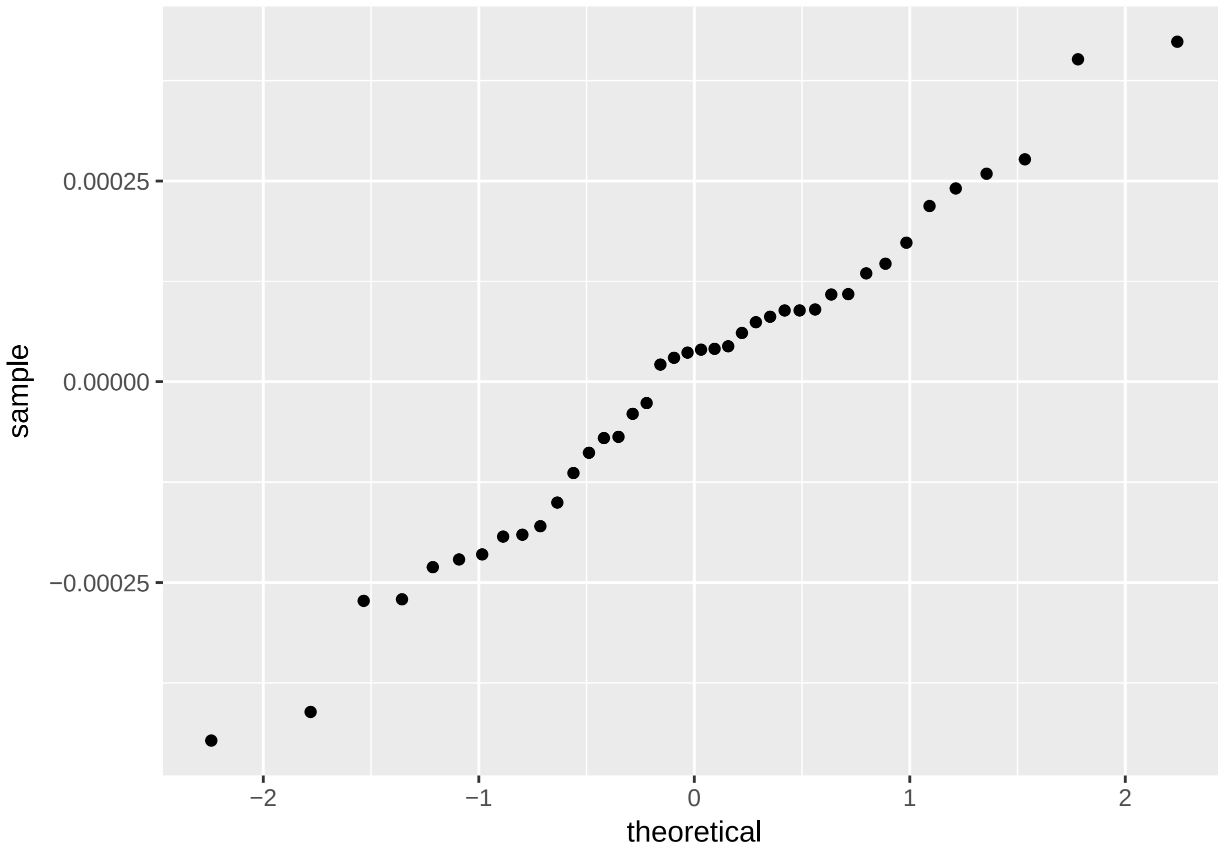
The results are identical.

```
ggplot(load_cell_fit_2) +
  geom_histogram(aes(.resid))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```r
ggplot(load_cell_fit_2) +
  geom_qq(aes(sample = .resid))
```



4.3.2 Thermal expansion of copper

from section 4.6.4. Thermal Expansion of Copper Case Study

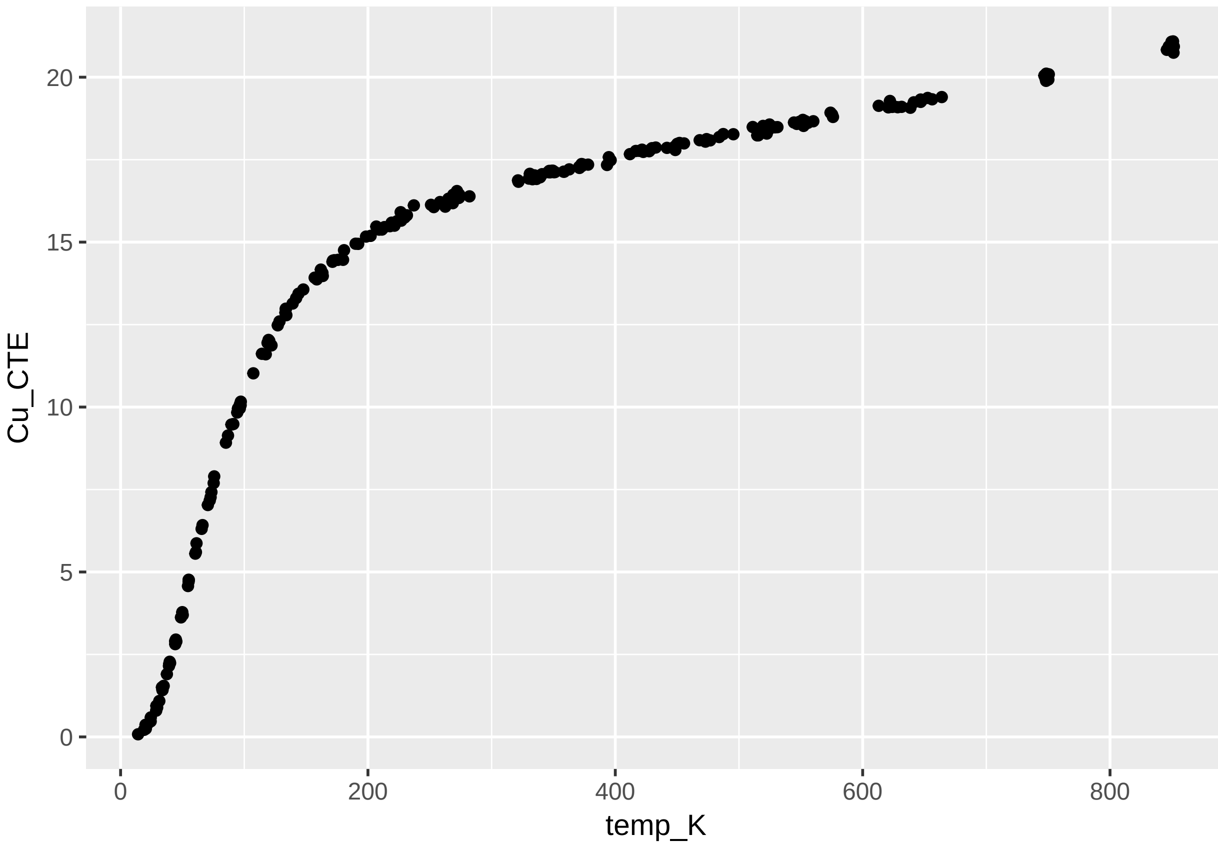
> This case study illustrates the use of a class of nonlinear models called rational function models. The data set used is the thermal expansion of copper related to temperature.

```
CTECu <- read_table2(
  "NIST data/HAHN1.dat", skip = 25, col_names = FALSE)
```

```
## Parsed with column specification:
## cols(
##   X1 = col_double(),
##   X2 = col_double()
## )
```

```
CTECu <- CTECu %>%
  rename(temp_K = X1, Cu_CTE = X2)
View(CTECu)
```

```
ggplot(CTECu, aes(temp_K, Cu_CTE)) +
  geom_point()
```



4.3.3 Quadratic/Quadratic (Q/Q) model

The NIST handbook has a procedure for calculing estimates for the model, below, I just used guess values for the equation

$$y = \frac{(A0 + A1 \cdot x + A2 \cdot x^2)}{(1 + B1 \cdot x + B2 \cdot X^2)}$$

```
model_Cu <- nls(Cu_CTE ~ ((a0 + a1*temp_K + a2*temp_K^2)/(1 + b1*temp_K + b2*temp_K^2)),
              CTECu, start = list(a0 = 0, a1 = -1, a2 = -1, b1 = 0, b2 = 0), trace = T)
```

```
## 1.344556e+13 :   0 -1 -1  0  0
## 1697329 :  -3.351400e+00  1.797879e-01 -5.745645e-04  7.915158e-07 -3.877688e-10
## 177589.3 :  -3.319661e+00  1.750319e-01 -3.592189e-04  1.144561e-03 -6.445222e-07
## 32411.82 :  -4.730736e+00  2.146213e-01 -3.235814e-04  3.851759e-03 -2.232626e-06
## 5374.946 :  -6.413603e+00  2.683958e-01 -2.844465e-04  7.799104e-03 -4.266107e-06
## 690.5713 :  -6.901674e+00  2.867489e-01 -1.804611e-04  1.009256e-02 -3.131612e-06
## 217.1164 :  -5.970973e+00  2.481232e-01  1.250216e-04  8.918691e-03  1.150181e-05
## 114.0929 :  -3.145220e+00  1.061122e-01  1.610442e-03  4.783451e-03  8.707152e-05
## 64.59622 :   3.1477349163 -0.2965983768  0.0078946767  0.0013004690  0.0003974542
## 42.78158 :   9.3230025114 -0.8210512798  0.0188009501  0.0103976361  0.0009147904
## 34.34313 :  11.848742450 -1.098105431  0.025888747  0.024139632  0.001231568
## 33.5614 :  11.838847499 -1.129134608  0.027249383  0.029538247  0.001286009
## 33.55298 :  12.220405792 -1.169396827  0.028259353  0.031296606  0.001332281
## 33.55278 :  11.988150281 -1.149285630  0.027831681  0.030880324  0.001312172
## 33.55273 :  12.165006923 -1.165285413  0.028186412  0.031296535  0.001328754
## 33.55271 :  12.034452089 -1.153533233  0.027926998  0.030997296  0.001316621
```

```
## 33.55269 :  12.131480095 -1.162274660  0.028120094  0.031220657  0.001325652
## 33.55268 :  12.059618930 -1.155801947  0.027977138  0.031055397  0.001318966
## 33.55268 :  12.112946981 -1.160605895  0.028083247  0.031178098  0.001323928
## 33.55268 :  12.073406215 -1.157044238  0.028004583  0.031087151  0.001320249
## 33.55267 :  12.102793316 -1.159691514  0.028063056  0.031154770  0.001322984
## 33.55267 :  12.081046435 -1.157732650  0.028019791  0.031104752  0.001320961
## 33.55267 :  12.097167203 -1.159184809  0.028051866  0.031141839  0.001322461
## 33.55267 :  12.085177311 -1.158104731  0.028028009  0.031114250  0.001321345
## 33.55267 :  12.094068505 -1.158905649  0.028045699  0.031134704  0.001322172
## 33.55267 :  12.087436601 -1.158308224  0.028032503  0.031119442  0.001321555
## 33.55267 :  12.092386536 -1.158754113  0.028042351  0.031130832  0.001322016
## 33.55267 :  12.088732963 -1.158425030  0.028035083  0.031122431  0.001321676
## 33.55267 :  12.091445289 -1.158669369  0.028040481  0.031128672  0.001321928
## 33.55267 :  12.08942043 -1.15848695  0.02803645  0.03112401  0.00132174
## 33.55267 :  12.090900623 -1.158620261  0.028039395  0.031127413  0.001321877
## 33.55267 :  12.089833503 -1.158524186  0.028037274  0.031124964  0.001321778
## 33.55267 :  12.090620445 -1.158595059  0.028038839  0.031126773  0.001321851
```

```r
summary(model_Cu)
```

```
##
## Formula: Cu_CTE ~ ((a0 + a1 * temp_K + a2 * temp_K^2)/(1 + b1 * temp_K +
##     b2 * temp_K^2))
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## a0 12.0906204  4.9457268   2.445  0.01525 *
## a1 -1.1585951  0.4459281  -2.598  0.00997 **
## a2  0.0280388  0.0099339   2.823  0.00518 **
## b1  0.0311268  0.0123461   2.521  0.01237 *
## b2  0.0013219  0.0004639   2.849  0.00478 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3811 on 231 degrees of freedom
##
## Number of iterations to convergence: 32
## Achieved convergence tolerance: 8.018e-06
```

```r
glance(model_Cu)
```

```
##       sigma isConv       finTol    logLik      AIC      BIC deviance
## 1 0.3811163   TRUE 8.017872e-06 -104.6851 221.3702 242.1532 33.55267
##   df.residual
## 1         231
```

```r
augment(model_Cu)
```

```
##     temp_K Cu_CTE     .fitted       .resid
## 1    24.41  0.591  0.20263740  0.388362604
## 2    34.82  1.547  1.55799629 -0.010996290
## 3    44.09  2.902  3.13916746 -0.237167465
## 4    45.07  2.894  3.30744524 -0.413445244
## 5    54.98  4.703  4.94210005 -0.239100048
## 6    65.51  6.307  6.48782610 -0.180826097
## 7    70.53  7.030  7.14914778 -0.119147778
```

```
## 8    75.70  7.898  7.78154503  0.116454973
## 9    89.57  9.470  9.25903236  0.210967637
## 10   91.14  9.484  9.40826472  0.075735280
## 11   96.40 10.072  9.88458561  0.187414386
## 12   97.19 10.163  9.95311700  0.209882997
## 13  114.26 11.615 11.26656602  0.348433983
## 14  120.25 12.005 11.66166565  0.343334345
## 15  127.08 12.478 12.07731227  0.400687732
## 16  133.55 12.982 12.44039495  0.541605047
## 17  133.61 12.970 12.44363272  0.526367277
## 18  158.67 13.926 13.62043804  0.305561962
## 19  172.74 14.452 14.15567440  0.296325603
## 20  171.31 14.404 14.10466048  0.299339518
## 21  202.14 15.190 15.06560294  0.124397055
## 22  220.55 15.550 15.52648312  0.023516883
## 23  221.05 15.528 15.53805198 -0.010051982
## 24  221.39 15.499 15.54589234 -0.046892338
## 25  250.99 16.131 16.15515873 -0.024158729
## 26  268.99 16.438 16.46610887 -0.028108874
## 27  271.80 16.387 16.51126548 -0.124265483
## 28  271.97 16.549 16.51397008  0.035029917
## 29  321.31 16.872 17.18698756 -0.314987562
## 30  321.69 16.830 17.19142843 -0.361428433
## 31  330.14 16.926 17.28772364 -0.361723642
## 32  333.03 16.907 17.31961419 -0.412614189
## 33  333.47 16.966 17.32442437 -0.358424366
## 34  340.77 17.060 17.40253947 -0.342539470
## 35  345.65 17.122 17.45304116 -0.331041159
## 36  373.11 17.311 17.71409685 -0.403096846
## 37  373.79 17.355 17.72010437 -0.365104372
## 38  411.82 17.668 18.02623331 -0.358233309
## 39  419.51 17.767 18.08174912 -0.314749116
## 40  421.59 17.803 18.09643544 -0.293435444
## 41  422.02 17.765 18.09945445 -0.334454452
## 42  422.47 17.768 18.10260764 -0.334607642
## 43  422.61 17.736 18.10358734 -0.367587337
## 44  441.75 17.858 18.23197297 -0.373972966
## 45  447.41 17.877 18.26793801 -0.390938005
## 46  448.70 17.912 18.27601424 -0.364014242
## 47  472.89 18.046 18.41967798 -0.373677978
## 48  476.69 18.085 18.44098172 -0.355981723
## 49  522.47 18.291 18.67429455 -0.383294553
## 50  522.62 18.357 18.67499461 -0.317994608
## 51  524.43 18.426 18.68341166 -0.257411661
## 52  546.75 18.584 18.78280566 -0.198805658
## 53  549.53 18.610 18.79464198 -0.184641978
## 54  575.29 18.870 18.89908047 -0.029080472
## 55  576.00 18.795 18.90183163 -0.106831626
## 56  625.55 19.111 19.07892819  0.032071807
## 57   20.15  0.367  0.05976671  0.307233295
## 58   28.78  0.796  0.65887516  0.137124838
## 59   29.57  0.892  0.76317614  0.128823858
## 60   37.41  1.903  1.98987963 -0.086879626
## 61   39.12  2.150  2.28182032 -0.131820319
```

```
## 62   50.24  3.697  4.17855148 -0.481551482
## 63   61.38  5.870  5.90731429 -0.037314289
## 64   66.25  6.421  6.58833329 -0.167333292
## 65   73.42  7.422  7.50855925 -0.086559249
## 66   95.52  9.944  9.80734632  0.136653680
## 67  107.32 11.023 10.76867928  0.254320724
## 68  122.04 11.870 11.77402819  0.095971811
## 69  134.03 12.786 12.46623314  0.319766858
## 70  163.19 14.067 13.80079176  0.266208244
## 71  163.48 13.974 13.81207896  0.161921040
## 72  175.70 14.462 14.25903034  0.202969659
## 73  179.86 14.464 14.39938426  0.064615736
## 74  211.27 15.381 15.30300305  0.077996951
## 75  217.78 15.483 15.46153988  0.021460123
## 76  219.14 15.590 15.49360705  0.096392951
## 77  262.52 16.075 16.35879653 -0.283796530
## 78  268.01 16.347 16.45015753 -0.103157525
## 79  268.62 16.181 16.46009888 -0.279098880
## 80  336.25 16.915 17.35454484 -0.439544844
## 81  337.23 17.003 17.36505255 -0.362052548
## 82  339.33 16.978 17.38737853 -0.409378534
## 83  427.38 17.756 18.13660333 -0.380603334
## 84  428.58 17.808 18.14479952 -0.336799522
## 85  432.68 17.868 18.17247767 -0.304477673
## 86  528.99 18.481 18.70437208 -0.223372077
## 87  531.08 18.486 18.71386351 -0.227863507
## 88  628.34 19.090 19.08809722  0.001902777
## 89  253.24 16.062 16.19621410 -0.134214101
## 90  273.13 16.337 16.53234301 -0.195343011
## 91  273.66 16.345 16.54069023 -0.195690232
## 92  282.10 16.388 16.66974746 -0.281747456
## 93  346.62 17.159 17.46292119 -0.303921190
## 94  347.19 17.116 17.46870293 -0.352702926
## 95  348.78 17.164 17.48473760 -0.320737599
## 96  351.18 17.123 17.50868379 -0.385683795
## 97  450.10 17.979 18.28472932 -0.305729323
## 98  450.35 17.974 18.28628016 -0.312280163
## 99  451.92 18.007 18.29598212 -0.288982125
## 100 455.56 17.993 18.31823099 -0.325230985
## 101 552.22 18.523 18.80598605 -0.282986051
## 102 553.56 18.669 18.81159745 -0.142597449
## 103 555.74 18.617 18.82067084 -0.203670839
## 104 652.59 19.371 19.16459642  0.206403582
## 105 656.20 19.330 19.17551650  0.154483498
## 106  14.13  0.080  0.77348741 -0.693487408
## 107  20.41  0.248  0.05662611  0.191373893
## 108  31.30  1.089  1.00816471  0.080835292
## 109  33.84  1.418  1.39956756  0.018432442
## 110  39.70  2.278  2.38155051 -0.103550513
## 111  48.83  3.624  3.94438216 -0.320382160
## 112  54.50  4.574  4.86657271 -0.292572712
## 113  60.41  5.556  5.76612933 -0.210129333
## 114  72.77  7.267  7.42904396 -0.162043958
## 115  75.25  7.695  7.72838906 -0.033389062
```

```
## 116  86.84  9.136  8.99136369  0.144636308
## 117  94.88  9.959  9.75056787  0.208432127
## 118  96.40  9.957  9.88458561  0.072414386
## 119 117.37 11.600 11.47547190  0.124528102
## 120 139.08 13.138 12.72947330  0.408526704
## 121 147.73 13.564 13.14657103  0.417428974
## 122 158.63 13.871 13.61880392  0.252196080
## 123 161.84 13.994 13.74780396  0.246196039
## 124 192.11 14.947 14.78222963  0.164770366
## 125 206.76 15.473 15.18803137  0.284968629
## 126 209.07 15.379 15.24746384  0.131536163
## 127 213.32 15.455 15.35384680  0.101153203
## 128 226.44 15.908 15.65987906  0.248120943
## 129 237.12 16.114 15.88663535  0.227364654
## 130 330.90 17.071 17.29616030 -0.225160304
## 131 358.72 17.135 17.58196286 -0.446962856
## 132 370.77 17.282 17.69326577 -0.411265769
## 133 372.72 17.368 17.71064206 -0.342642062
## 134 396.24 17.483 17.90752311 -0.424523105
## 135 416.59 17.764 18.06089749 -0.296897490
## 136 484.02 18.185 18.48117310 -0.296173098
## 137 495.47 18.271 18.54167975 -0.270679752
## 138 514.78 18.236 18.63788133 -0.401881325
## 139 515.65 18.237 18.64205311 -0.405053108
## 140 519.47 18.523 18.66021207 -0.137212071
## 141 544.47 18.627 18.77301150 -0.146011498
## 142 560.11 18.665 18.83865465 -0.173654648
## 143 620.77 19.086 19.06303418  0.022965825
## 144  18.97  0.214  0.09784057  0.116159433
## 145  28.93  0.943  0.67826891  0.264731089
## 146  33.91  1.429  1.41076863  0.018231366
## 147  40.03  2.241  2.43840011 -0.197400110
## 148  44.66  2.951  3.23712804 -0.286128037
## 149  49.87  3.782  4.11738756 -0.335387561
## 150  55.16  4.757  4.97031350 -0.213313499
## 151  60.90  5.602  5.83768000 -0.235680001
## 152  72.08  7.169  7.34380175 -0.174801749
## 153  85.15  8.920  8.82027800  0.099721997
## 154  97.06 10.055  9.94189178  0.113108223
## 155 119.63 12.035 11.62215376  0.412846236
## 156 133.27 12.861 12.42525502  0.435744978
## 157 143.84 13.436 12.96400254  0.471997455
## 158 161.91 14.167 13.75056958  0.416430418
## 159 180.67 14.755 14.42607069  0.328929310
## 160 198.44 15.168 14.96397105  0.204028954
## 161 226.86 15.651 15.66915570 -0.018155704
## 162 229.65 15.746 15.73001114  0.015988863
## 163 258.27 16.216 16.28565218 -0.069652183
## 164 273.77 16.445 16.54241898 -0.097418978
## 165 339.15 16.965 17.38547498 -0.420474978
## 166 350.13 17.121 17.49824509 -0.377245093
## 167 362.75 17.206 17.61995849 -0.413958492
## 168 371.03 17.250 17.69559252 -0.445592517
## 169 393.32 17.339 17.88428738 -0.545287384
```

```
## 170 448.53 17.793 18.27495246 -0.481952459
## 171 473.78 18.123 18.42469678 -0.301696776
## 172 511.12 18.490 18.62018208 -0.130182079
## 173 524.70 18.566 18.68466247 -0.118662470
## 174 548.75 18.645 18.79133263 -0.146332628
## 175 551.64 18.706 18.80354911 -0.097549110
## 176 574.02 18.924 18.89414305  0.029856952
## 177 623.86 19.100 19.07333565  0.026664346
## 178  21.46  0.375  0.06146604  0.313533960
## 179  24.33  0.471  0.19669499  0.274305007
## 180  33.43  1.504  1.33435088  0.169649123
## 181  39.22  2.204  2.29899563 -0.094995630
## 182  44.18  2.813  3.15464924 -0.341649241
## 183  55.02  4.765  4.94837488 -0.183374875
## 184  94.33  9.835  9.70136218  0.133637823
## 185  96.44 10.040  9.88807384  0.151926161
## 186 118.82 11.946 11.57006498  0.375935018
## 187 128.48 12.596 12.15828826  0.437711736
## 188 141.94 13.303 12.87190265  0.431097345
## 189 156.92 13.922 13.54830088  0.373699124
## 190 171.65 14.440 14.11685477  0.323145234
## 191 190.00 14.951 14.71932283  0.231677171
## 192 223.26 15.627 15.58863528  0.038364719
## 193 223.88 15.639 15.60266672  0.036333280
## 194 231.50 15.814 15.76964133  0.044358666
## 195 265.05 16.315 16.40132663 -0.086326629
## 196 269.44 16.334 16.47339806 -0.139398063
## 197 271.78 16.430 16.51094709 -0.080947092
## 198 273.46 16.423 16.53754381 -0.114543806
## 199 334.61 17.024 17.33683230 -0.312832301
## 200 339.79 17.009 17.39223463 -0.383234635
## 201 349.52 17.165 17.49215378 -0.327153784
## 202 358.18 17.134 17.57681079 -0.442810793
## 203 377.98 17.349 17.75667290 -0.407672903
## 204 394.77 17.576 17.89586620 -0.319866198
## 205 429.66 17.848 18.15213896 -0.304138963
## 206 468.22 18.090 18.39304523 -0.303045226
## 207 487.27 18.276 18.49862363 -0.222623626
## 208 519.54 18.404 18.66054244 -0.256542438
## 209 523.03 18.519 18.67690613 -0.157906127
## 210 612.99 19.133 19.03665298  0.096347018
## 211 638.59 19.074 19.12111752 -0.047117521
## 212 641.36 19.239 19.12986584  0.109134164
## 213 622.05 19.280 19.06731345  0.212686549
## 214 631.50 19.101 19.09838763  0.002612371
## 215 663.97 19.398 19.19863025  0.199369748
## 216 646.90 19.252 19.14714500  0.104855005
## 217 748.29 19.890 19.41945281  0.470547190
## 218 749.21 20.007 19.42159564  0.585404361
## 219 750.14 19.929 19.42375656  0.505243437
## 220 647.04 19.268 19.14757794  0.120422056
## 221 646.89 19.324 19.14711406  0.176885936
## 222 746.90 20.049 19.41620554  0.632794460
## 223 748.43 20.107 19.41977922  0.687220777
```

```
## 224 747.35 20.062 19.41725810  0.644741900
## 225 749.27 20.065 19.42173521  0.643264790
## 226 647.61 19.286 19.14933879  0.136661207
## 227 747.78 19.972 19.41826273  0.553737271
## 228 750.51 20.088 19.42461484  0.663385162
## 229 851.37 20.743 19.63143059  1.111569415
## 230 845.97 20.830 19.62157695  1.208423055
## 231 847.54 20.935 19.62445443  1.310545566
## 232 849.93 21.035 19.62881490  1.406185103
## 233 851.61 20.930 19.63186569  1.298134306
## 234 849.75 21.074 19.62848733  1.445512673
## 235 850.98 21.085 19.63072302  1.454276977
## 236 848.23 20.935 19.62571577  1.309284225
```

```
tidy(model_Cu)
```

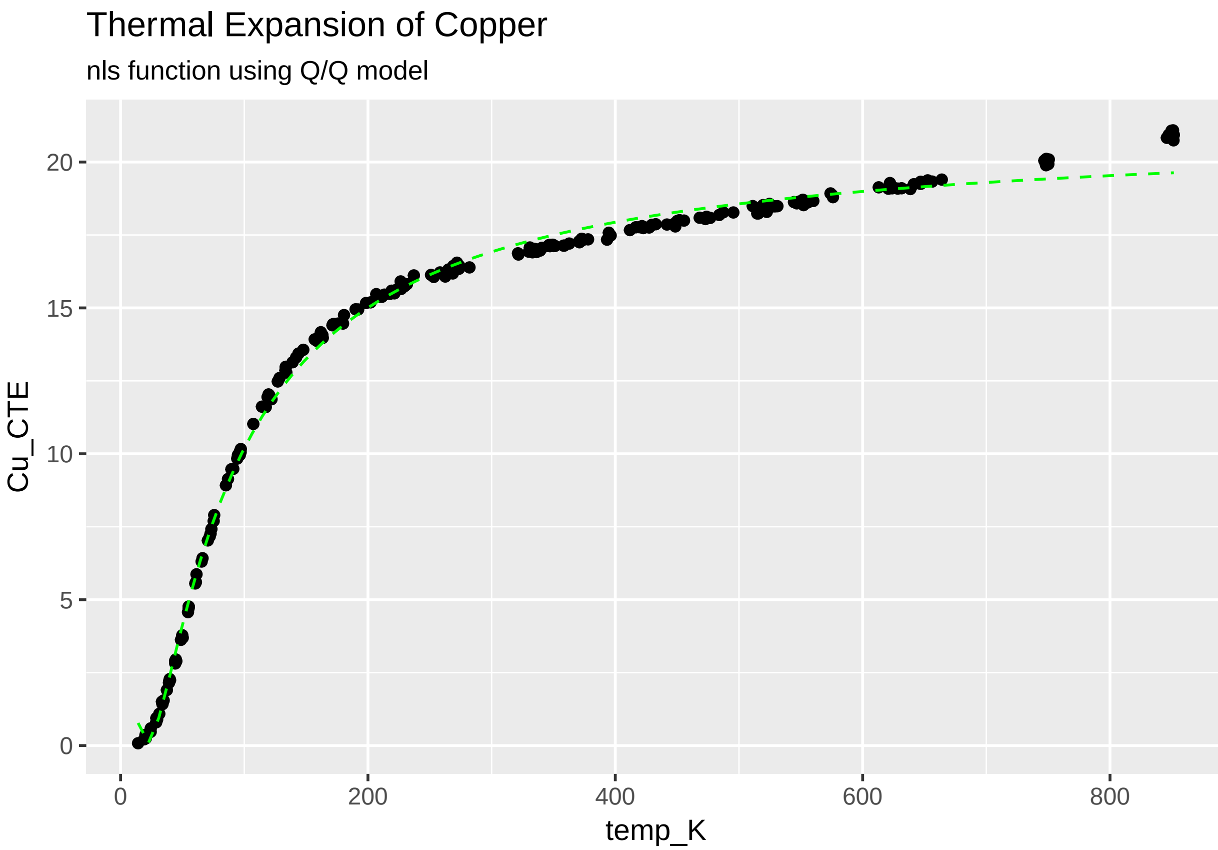
```
##   term     estimate    std.error statistic     p.value
## 1   a0 12.090620445 4.9457267991  2.444660 0.015247600
## 2   a1 -1.158595059 0.4459281342 -2.598165 0.009974985
## 3   a2  0.028038839 0.0099338856  2.822545 0.005179792
## 4   b1  0.031126773 0.0123461159  2.521179 0.012370785
## 5   b2  0.001321851 0.0004639428  2.849169 0.004779119
```

4.3.3.1 Create a function using the fit paramters

```
Cu_fit <- function(x) {
  ((summary(model_Cu)$coefficients[1] + summary(model_Cu)$coefficients[2]*x + summary(model_Cu)$coeffic
  }
```

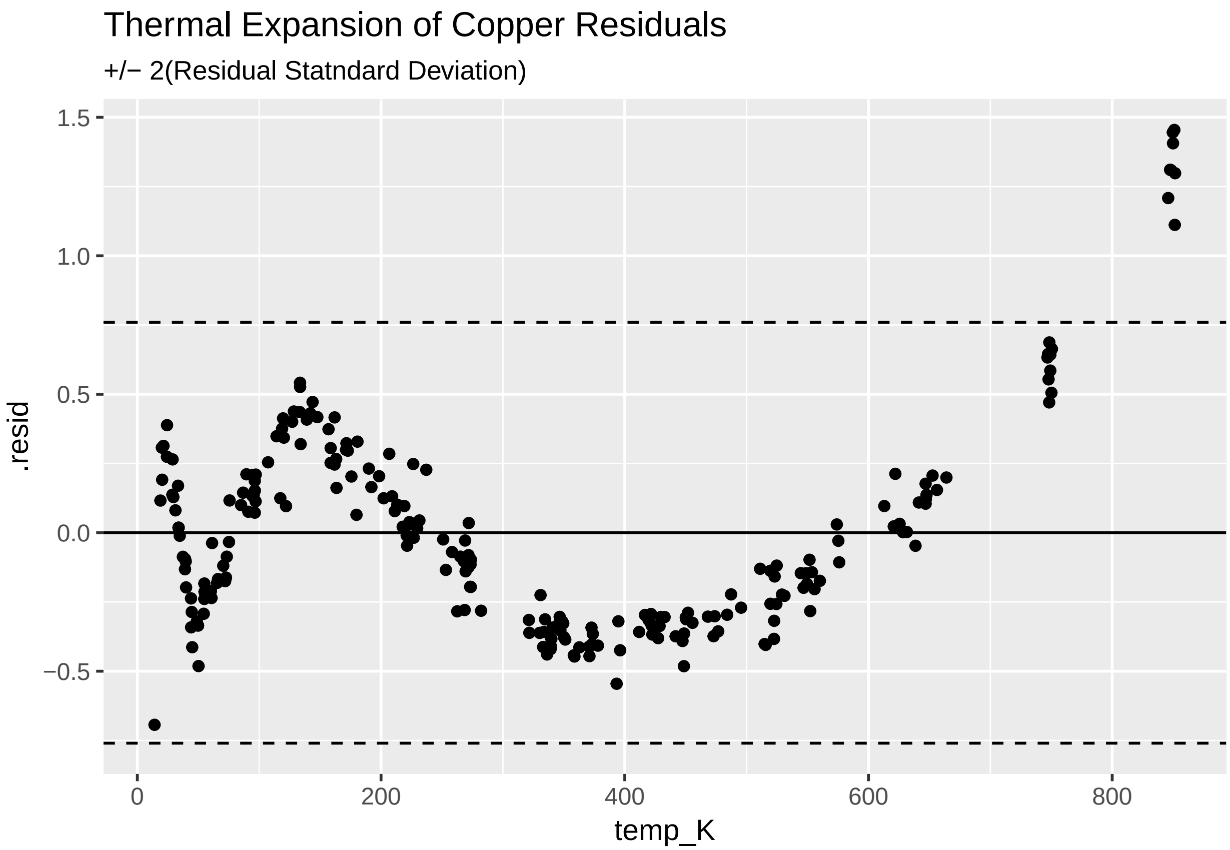
4.3.3.2 Add the fitted curve to the graph

```
ggplot(CTECu, aes(temp_K, Cu_CTE)) +
  geom_point() +
  stat_function(fun = Cu_fit, colour = "green", linetype = "dashed") +
  ggtitle("Thermal Expansion of Copper", subtitle = "nls function using Q/Q model")
```



4.3.3.3 Plot the residulas

```
ggplot(augment(model_Cu)) +
  geom_point(aes(temp_K, .resid)) +
  geom_hline(aes(yintercept=0)) +
  geom_hline(aes(yintercept=+2*(0.38)), linetype = "dashed") +
  geom_hline(aes(yintercept=-2*(0.38)), linetype = "dashed") +
  ggtitle("Thermal Expansion of Copper Residuals", subtitle = "+/- 2(Residual Statndard Deviation)")
```



The fit is not very good, shows a clear structure, and indicates the Q/Q model is insufficient.

4.3.4 Cubic/Cubic Rational Function

$$y = \frac{(A0 + A1 * x + A2x^2 + A3X^3)}{(1 + B1x + B2X^2 + B3X^3)}$$

```
mcc_Cu <- nls(Cu_CTE ~ ((a0 + a1*temp_K + a2*temp_K^2 + a3*temp_K^3)/(1 + b1*temp_K + b2*temp_K^2 + b3*
            CTECu, start = list(a0 = 0, a1 = -1, a2 = -1, a3 = 0, b1 = 0, b2 = 0, b3 = 0),
            trace = T)
```

```
## 1.344556e+13 :   0 -1 -1  0  0  0  0
## 1.339595e+13 :  -4.089856e-03 -9.988271e-01 -9.983564e-01  6.676610e-04 -6.676602e-04 -1.055201e-11
## 1.334366e+13 :  -4.807459e-03 -9.966232e-01 -9.964072e-01  6.664584e-04 -6.677607e-04 -1.094787e-11
## 1.32396e+13 :  -6.863478e-03 -9.922082e-01 -9.925166e-01  6.639777e-04 -6.678813e-04 -1.223912e-11
## 1.30334e+13 :  -1.594000e-02 -9.833127e-01 -9.847661e-01  6.593020e-04 -6.683904e-04 -1.602231e-11
## 1.262831e+13 :  -6.540076e-02 -9.650849e-01 -9.693885e-01  6.507887e-04 -6.701861e-04 -3.033631e-11
## 1.184743e+13 :  -1.788916e-01 -9.289031e-01 -9.391153e-01  6.341455e-04 -6.739522e-04 -6.184747e-11
## 1.040295e+13 :  -4.142861e-01 -8.583930e-01 -8.804668e-01  5.994049e-04 -6.790616e-04 -1.428677e-10
## 7.989692e+12 :  -9.341122e-01 -7.234575e-01 -7.705432e-01  5.175015e-04 -6.708211e-04 -4.851866e-10
## 6.371193e+12 :  -1.922326e+00 -4.855256e-01 -5.789681e-01 -4.405781e-04  4.048477e-04 -1.132038e-09
## 1.590227e+12 :  -3.385858e+00 -1.294959e-01 -2.899297e-01 -2.187966e-04  4.037949e-04 -2.100189e-09
## 19889209 :  -4.850624e+00  2.266128e-01 -8.927502e-04  3.128700e-06  3.991144e-04 -5.877643e-09  2.69
## 825292.5 :  -4.854099e+00  2.244501e-01 -7.872443e-04  1.327192e-06  8.773391e-04  4.504172e-08 -1.18
## 72834.96 :  -4.698904e+00  2.158071e-01 -6.255161e-04  8.206990e-07  1.206312e-03  1.711327e-06 -1.28
## 11897.69 :  -4.260919e+00  1.962288e-01 -4.518142e-04  6.174559e-07  5.394675e-04  9.144746e-06 -6.25
## 3068.102 :  -3.363680e+00  1.545785e-01 -1.051538e-04  4.250646e-07 -1.433242e-03  2.984461e-05 -2.07
```

```
## 909.5887 :  -1.947093e+00  8.097392e-02  6.980495e-04 -2.856405e-08 -4.239912e-03  7.556907e-05 -5.39
## 220.8618 :  -4.057550e-01 -1.376339e-02  2.096694e-03 -9.279485e-07 -6.080614e-03  1.470438e-04 -1.05
## 31.88262 :   7.090768e-01 -9.381732e-02  3.536101e-03 -1.838264e-06 -6.162496e-03  2.145881e-04 -1.48
## 3.200182 :   1.102048e+00 -1.249213e-01  4.149347e-03 -1.951524e-06 -5.756966e-03  2.422411e-04 -1.49
## 1.55733 :   1.100520e+00 -1.246480e-01  4.131519e-03 -1.571143e-06 -5.729023e-03  2.422069e-04 -1.30
## 1.532465 :   1.080053e+00 -1.228904e-01  4.090638e-03 -1.435504e-06 -5.758189e-03  2.407076e-04 -1.23
## 1.532438 :   1.077745e+00 -1.227015e-01  4.086549e-03 -1.426532e-06 -5.760914e-03  2.405448e-04 -1.23
## 1.532438 :   1.077639e+00 -1.226932e-01  4.086381e-03 -1.426274e-06 -5.760992e-03  2.405376e-04 -1.23
```

```r
summary(mcc_Cu)
```

```
##
## Formula: Cu_CTE ~ ((a0 + a1 * temp_K + a2 * temp_K^2 + a3 * temp_K^3)/(1 +
##     b1 * temp_K + b2 * temp_K^2 + b3 * temp_K^3))
##
## Parameters:
##      Estimate Std. Error t value Pr(>|t|)
## a0  1.078e+00  1.707e-01   6.313 1.40e-09 ***
## a1 -1.227e-01  1.200e-02 -10.224  < 2e-16 ***
## a2  4.086e-03  2.251e-04  18.155  < 2e-16 ***
## a3 -1.426e-06  2.758e-07  -5.172 5.06e-07 ***
## b1 -5.761e-03  2.471e-04 -23.312  < 2e-16 ***
## b2  2.405e-04  1.045e-05  23.019  < 2e-16 ***
## b3 -1.231e-07  1.303e-08  -9.453  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.0818 on 229 degrees of freedom
##
## Number of iterations to convergence: 23
## Achieved convergence tolerance: 1.664e-06
```

```r
glance(mcc_Cu)
```

```
##        sigma isConv       finTol   logLik       AIC       BIC deviance
## 1 0.08180385   TRUE 1.664197e-06 259.4932 -502.9863 -475.2757 1.532438
##   df.residual
## 1         229
```

```r
augment(mcc_Cu)
```

```
##     temp_K Cu_CTE    .fitted       .resid
## 1    24.41  0.591  0.4963639  0.094636113
## 2    34.82  1.547  1.5653400 -0.018340010
## 3    44.09  2.902  2.9005739  0.001426132
## 4    45.07  2.894  3.0533083 -0.159308281
## 5    54.98  4.703  4.6386624  0.064337570
## 6    65.51  6.307  6.2804623  0.026537706
## 7    70.53  7.030  7.0126772  0.017322841
## 8    75.70  7.898  7.7231356  0.174864443
## 9    89.57  9.470  9.3955397  0.074460287
## 10   91.14  9.484  9.5636985 -0.079698527
## 11   96.40 10.072 10.0975170 -0.025516971
## 12   97.19 10.163 10.1738762 -0.010876165
## 13  114.26 11.615 11.6058598  0.009140235
## 14  120.25 12.005 12.0216811 -0.016681059
```

```
## 15   127.08 12.478 12.4499677  0.028032300
## 16   133.55 12.982 12.8157611  0.166238889
## 17   133.61 12.970 12.8189865  0.151013467
## 18   158.67 13.926 13.9459607 -0.019960705
## 19   172.74 14.452 14.4277145  0.024285466
## 20   171.31 14.404 14.3826277  0.021372299
## 21   202.14 15.190 15.2041280 -0.014127993
## 22   220.55 15.550 15.5795355 -0.029535518
## 23   221.05 15.528 15.5888324 -0.060832403
## 24   221.39 15.499 15.5951299 -0.096129866
## 25   250.99 16.131 16.0783145  0.052685472
## 26   268.99 16.438 16.3224628  0.115537197
## 27   271.80 16.387 16.3579471  0.029052870
## 28   271.97 16.549 16.3600733  0.188926721
## 29   321.31 16.872 16.8979959 -0.025995876
## 30   321.69 16.830 16.9016483 -0.071648338
## 31   330.14 16.926 16.9813826 -0.055382599
## 32   333.03 16.907 17.0080323 -0.101032295
## 33   333.47 16.966 17.0120633 -0.046063327
## 34   340.77 17.060 17.0779701 -0.017970082
## 35   345.65 17.122 17.1210593  0.000940730
## 36   373.11 17.311 17.3512466 -0.040246649
## 37   373.79 17.355 17.3567171 -0.001717084
## 38   411.82 17.668 17.6493340  0.018665965
## 39   419.51 17.767 17.7059105  0.061089474
## 40   421.59 17.803 17.7210942  0.081905760
## 41   422.02 17.765 17.7242272  0.040772842
## 42   422.47 17.768 17.7275036  0.040496385
## 43   422.61 17.736 17.7285225  0.007477495
## 44   441.75 17.858 17.8659854 -0.007985381
## 45   447.41 17.877 17.9060218 -0.029021763
## 46   448.70 17.912 17.9151127 -0.003112748
## 47   472.89 18.046 18.0836406 -0.037640604
## 48   476.69 18.085 18.1098368 -0.024836835
## 49   522.47 18.291 18.4221886 -0.131188615
## 50   522.62 18.357 18.4232070 -0.066206966
## 51   524.43 18.426 18.4354946 -0.009494602
## 52   546.75 18.584 18.5870962 -0.003096202
## 53   549.53 18.610 18.6060088  0.003991184
## 54   575.29 18.870 18.7819540  0.088045977
## 55   576.00 18.795 18.7868265  0.008173532
## 56   625.55 19.111 19.1315632 -0.020563155
## 57    20.15  0.367  0.2578734  0.109126633
## 58    28.78  0.796  0.8706755 -0.074675533
## 59    29.57  0.892  0.9508152 -0.058815215
## 60    37.41  1.903  1.9126172 -0.009617204
## 61    39.12  2.150  2.1545245 -0.004524486
## 62    50.24  3.697  3.8760413 -0.179041267
## 63    61.38  5.870  5.6506367  0.219363250
## 64    66.25  6.421  6.3908398  0.030160222
## 65    73.42  7.422  7.4155213  0.006478695
## 66    95.52  9.944 10.0113074 -0.067307437
## 67   107.32 11.023 11.0710890 -0.048089021
## 68   122.04 11.870 12.1384228 -0.268422792
```

```
## 69   134.03 12.786 12.8414824 -0.055482433
## 70   163.19 14.067 14.1104537 -0.043453696
## 71   163.48 13.974 14.1206749 -0.146674921
## 72   175.70 14.462 14.5185343 -0.056534286
## 73   179.86 14.464 14.6407452 -0.176745156
## 74   211.27 15.381 15.3987890 -0.017789027
## 75   217.78 15.483 15.5272412 -0.044241243
## 76   219.14 15.590 15.5530853  0.036914739
## 77   262.52 16.075 16.2382086 -0.163208614
## 78   268.01 16.347 16.3099337  0.037066300
## 79   268.62 16.181 16.3177419 -0.136741912
## 80   336.25 16.915 17.0373752 -0.122375209
## 81   337.23 17.003 17.0462347 -0.043234658
## 82   339.33 16.978 17.0651106 -0.087110573
## 83   427.38 17.756 17.7631127 -0.007112672
## 84   428.58 17.808 17.7717775  0.036222457
## 85   432.68 17.868 17.8012757  0.066724340
## 86   528.99 18.481 18.4664497  0.014550340
## 87   531.08 18.486 18.4806380  0.005362040
## 88   628.34 19.090 19.1513119 -0.061311930
## 89   253.24 16.062 16.1105736 -0.048573567
## 90   273.13 16.337 16.3745196 -0.037519609
## 91   273.66 16.345 16.3810847 -0.036084688
## 92   282.10 16.388 16.4827743 -0.094774326
## 93   346.62 17.159 17.1295367  0.029463321
## 94   347.19 17.116 17.1345051 -0.018505086
## 95   348.78 17.164 17.1483135  0.015686490
## 96   351.18 17.123 17.1690174 -0.046017428
## 97   450.10 17.979 17.9249654  0.054034648
## 98   450.35 17.974 17.9267233  0.047276717
## 99   451.92 18.007 17.9377531  0.069246851
## 100 455.56 17.993 17.9632617  0.029738317
## 101 552.22 18.523 18.6243198 -0.101319806
## 102 553.56 18.669 18.6334455  0.035554457
## 103 555.74 18.617 18.6482984 -0.031298442
## 104 652.59 19.371 19.3248625  0.046137474
## 105 656.20 19.330 19.3510119 -0.021011943
## 106  14.13  0.080  0.1612725 -0.081272481
## 107  20.41  0.248  0.2685479 -0.020547945
## 108  31.30  1.089  1.1383031 -0.049303084
## 109  33.84  1.418  1.4408211 -0.022821106
## 110  39.70  2.278  2.2385532  0.039446784
## 111  48.83  3.624  3.6497122 -0.025712192
## 112  54.50  4.574  4.5616267  0.012373331
## 113  60.41  5.556  5.4996610  0.056338998
## 114  72.77  7.267  7.3261617 -0.059161721
## 115  75.25  7.695  7.6631465  0.031853502
## 116  86.84  9.136  9.0931449  0.042855052
## 117  94.88  9.959  9.9478398  0.011160239
## 118  96.40  9.957 10.0975170 -0.140516971
## 119 117.37 11.600 11.8267293 -0.226729304
## 120 139.08 13.138 13.1011146  0.036885374
## 121 147.73 13.564 13.5032486  0.060751378
## 122 158.63 13.871 13.9444602 -0.073460197
```

```
## 123 161.84 13.994 14.0623548 -0.068354756
## 124 192.11 14.947 14.9677463 -0.020746271
## 125 206.76 15.473 15.3048785  0.168121477
## 126 209.07 15.379 15.3535064  0.025493579
## 127 213.32 15.455 15.4401119  0.014888101
## 128 226.44 15.908 15.6864092  0.221590783
## 129 237.12 16.114 15.8666313  0.247368679
## 130 330.90 17.071 16.9884203  0.082579731
## 131 358.72 17.135 17.2330261 -0.098026092
## 132 370.77 17.282 17.3323460 -0.050345962
## 133 372.72 17.368 17.3481047  0.019895254
## 134 396.24 17.483 17.5323108 -0.049310842
## 135 416.59 17.764 17.6845113  0.079488723
## 136 484.02 18.185 18.1602046  0.024795393
## 137 495.47 18.271 18.2385282  0.032471776
## 138 514.78 18.236 18.3699663 -0.133966278
## 139 515.65 18.237 18.3758764 -0.138876358
## 140 519.47 18.523 18.4018198  0.121180208
## 141 544.47 18.627 18.5715926  0.055407414
## 142 560.11 18.665 18.6780990 -0.013099005
## 143 620.77 19.086 19.0978246 -0.011824627
## 144  18.97  0.214  0.2160310 -0.002031040
## 145  28.93  0.943  0.8856173  0.057382657
## 146  33.91  1.429  1.4495793 -0.020579292
## 147  40.03  2.241  2.2867741 -0.045774104
## 148  44.66  2.951  2.9892364 -0.038236434
## 149  49.87  3.782  3.8165695 -0.034569455
## 150  55.16  4.757  4.6675252  0.089474794
## 151  60.90  5.602  5.5760576  0.025942404
## 152  72.08  7.169  7.2305064 -0.061506448
## 153  85.15  8.920  8.8994792  0.020520834
## 154  97.06 10.055 10.1613775 -0.106377474
## 155 119.63 12.035 11.9804653  0.054534701
## 156 133.27 12.861 12.8006702  0.060329793
## 157 143.84 13.436 13.3286469  0.107353146
## 158 161.91 14.167 14.0648699  0.102130081
## 159 180.67 14.755 14.6638377  0.091162293
## 160 198.44 15.168 15.1198750  0.048125043
## 161 226.86 15.651 15.6938160 -0.042816044
## 162 229.65 15.746 15.7423289  0.003671052
## 163 258.27 16.216 16.1807994  0.035200620
## 164 273.77 16.445 16.3824445  0.062555507
## 165 339.15 16.965 17.0634984 -0.098498370
## 166 350.13 17.121 17.1599798 -0.038979782
## 167 362.75 17.206 17.2666301 -0.060630075
## 168 371.03 17.250 17.3344519 -0.084451920
## 169 393.32 17.339 17.5099757 -0.170975691
## 170 448.53 17.793 17.9139154 -0.120915404
## 171 473.78 18.123 18.0897817  0.033218343
## 172 511.12 18.490 18.3450954  0.144904609
## 173 524.70 18.566 18.4373275  0.128672496
## 174 548.75 18.645 18.6007013  0.044298675
## 175 551.64 18.706 18.6203708  0.085629235
## 176 574.02 18.924 18.7732422  0.150757834
```

```
## 177 623.86 19.100 19.1196210 -0.019620959
## 178  21.46  0.375  0.3169130  0.058086952
## 179  24.33  0.471  0.4906796 -0.019679635
## 180  33.43  1.504  1.3899566  0.114043426
## 181  39.22  2.204  2.1689447  0.035055321
## 182  44.18  2.813  2.9145398 -0.101539787
## 183  55.02  4.765  4.6450777  0.119922345
## 184  94.33  9.835  9.8927746 -0.057774647
## 185  96.44 10.040 10.1014067 -0.061406663
## 186 118.82 11.946 11.9260020  0.019998000
## 187 128.48 12.596 12.5322413  0.063758682
## 188 141.94 13.303 13.2397256  0.063274444
## 189 156.92 13.922 13.8795489  0.042451100
## 190 171.65 14.440 14.3934209  0.046579088
## 191 190.00 14.951 14.9146360  0.036363980
## 192 223.26 15.627 15.6294176 -0.002417573
## 193 223.88 15.639 15.6406575 -0.001657497
## 194 231.50 15.814 15.7738532  0.040146846
## 195 265.05 16.315 16.2715923  0.043407743
## 196 269.44 16.334 16.3281890  0.005810976
## 197 271.78 16.430 16.3576968  0.072303157
## 198 273.46 16.423 16.3786099  0.044390110
## 199 334.61 17.024 17.0224756  0.001524429
## 200 339.79 17.009 17.0692258 -0.060225818
## 201 349.52 17.165 17.1547149  0.010285129
## 202 358.18 17.134 17.2284921 -0.094492125
## 203 377.98 17.349 17.3902120 -0.041211980
## 204 394.77 17.576 17.5210839  0.054916093
## 205 429.66 17.848 17.7795636  0.068436368
## 206 468.22 18.090 18.0513573  0.038642735
## 207 487.27 18.276 18.1824758  0.093524152
## 208 519.54 18.404 18.4022951  0.001704889
## 209 523.03 18.519 18.4259904  0.093009575
## 210 612.99 19.133 19.0431600  0.089840025
## 211 638.59 19.074 19.2242391 -0.150239081
## 212 641.36 19.239 19.2440522 -0.005052151
## 213 622.05 19.280 19.1068475  0.173152498
## 214 631.50 19.101 19.1737313 -0.072731259
## 215 663.97 19.398 19.4075930 -0.009592967
## 216 646.90 19.252 19.2838185 -0.031818479
## 217 748.29 19.890 20.0529391 -0.162939136
## 218 749.21 20.007 20.0603469 -0.053346895
## 219 750.14 19.929 20.0678442 -0.138844239
## 220 647.04 19.268 19.2848259 -0.016825875
## 221 646.89 19.324 19.2837465  0.040253473
## 222 746.90 20.049 20.0417638  0.007236178
## 223 748.43 20.107 20.0540658  0.052934170
## 224 747.35 20.062 20.0453795  0.016620484
## 225 749.27 20.065 20.0608303  0.004169680
## 226 647.61 19.286 19.2889287 -0.002928687
## 227 747.78 19.972 20.0488365 -0.076836491
## 228 750.51 20.088 20.0708296  0.017170406
## 229 851.37 20.743 20.9461745 -0.203174508
## 230 845.97 20.830 20.8958249 -0.065824945
```

```
## 231 847.54 20.935 20.9104183  0.024581707
## 232 849.93 21.035 20.9327049  0.102295138
## 233 851.61 20.930 20.9484225 -0.018422516
## 234 849.75 21.074 20.9310234  0.142976630
## 235 850.98 21.085 20.9425234  0.142476633
## 236 848.23 20.935 20.9168436  0.018156359
```

```
tidy(mcc_Cu)
```

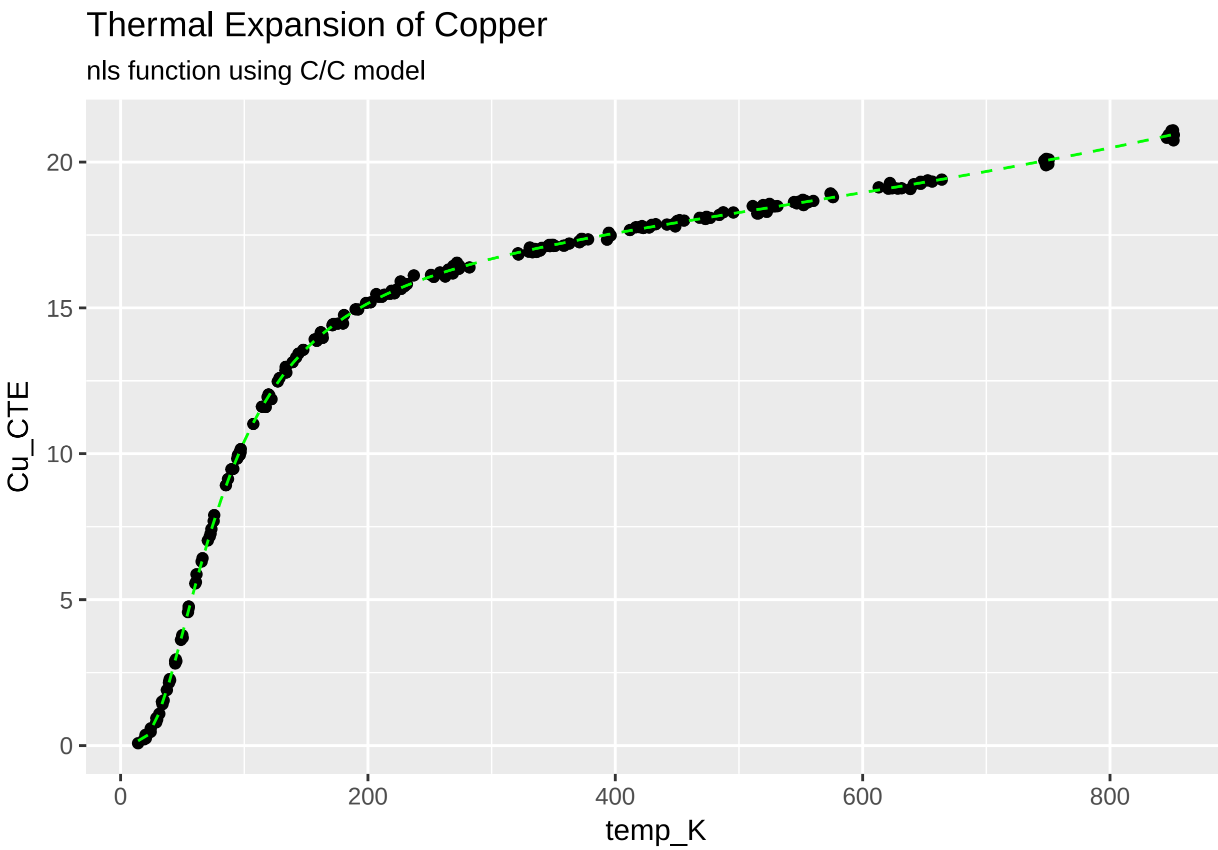
```
##   term      estimate    std.error  statistic      p.value
## 1   a0  1.077639e+00 1.707017e-01   6.312994 1.404129e-09
## 2   a1 -1.226932e-01 1.200030e-02 -10.224180 1.866827e-20
## 3   a2  4.086381e-03 2.250834e-04  18.154961 3.109432e-46
## 4   a3 -1.426274e-06 2.757806e-07  -5.171771 5.055842e-07
## 5   b1 -5.760992e-03 2.471290e-04 -23.311682 2.178299e-62
## 6   b2  2.405376e-04 1.044939e-05  23.019305 1.657786e-61
## 7   b3 -1.231449e-07 1.302735e-08  -9.452798 4.078186e-18
```

4.3.4.1 Create a function using the fit parameters

```
cc.Cu.fit <- function(x) {
  ((summary(mcc_Cu)$coefficients[1] + summary(mcc_Cu)$coefficients[2]*x +
      summary(mcc_Cu)$coefficients[3]*x^2 + summary(mcc_Cu)$coefficients[4]*x^3)/(1 + summary(mcc_Cu)$co
  }
```

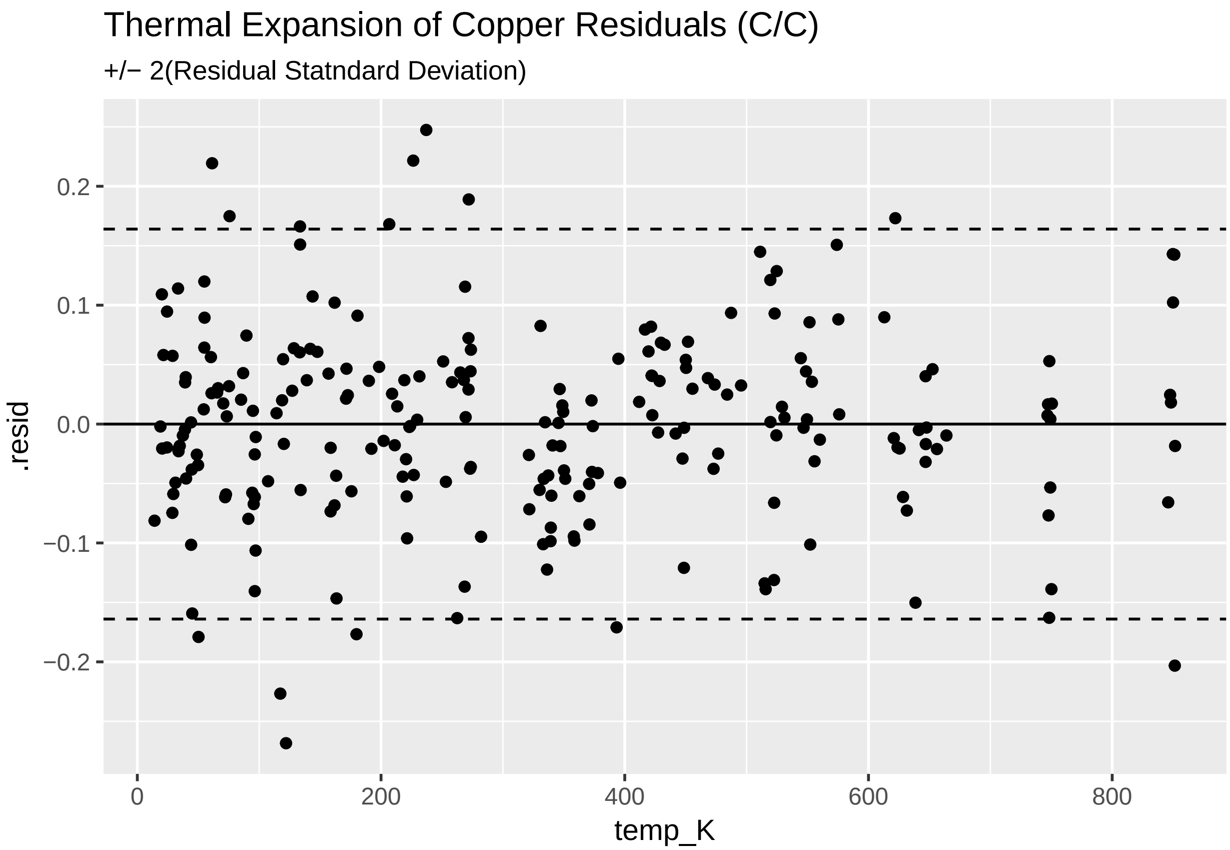
4.3.4.2 Add the fitted curve to the graph

```
ggplot(CTECu, aes(temp_K, Cu_CTE)) +
  geom_point() +
  stat_function(fun = cc.Cu.fit, colour = "green", linetype = "dashed") +
  ggtitle("Thermal Expansion of Copper", subtitle = "nls function using C/C model")
```



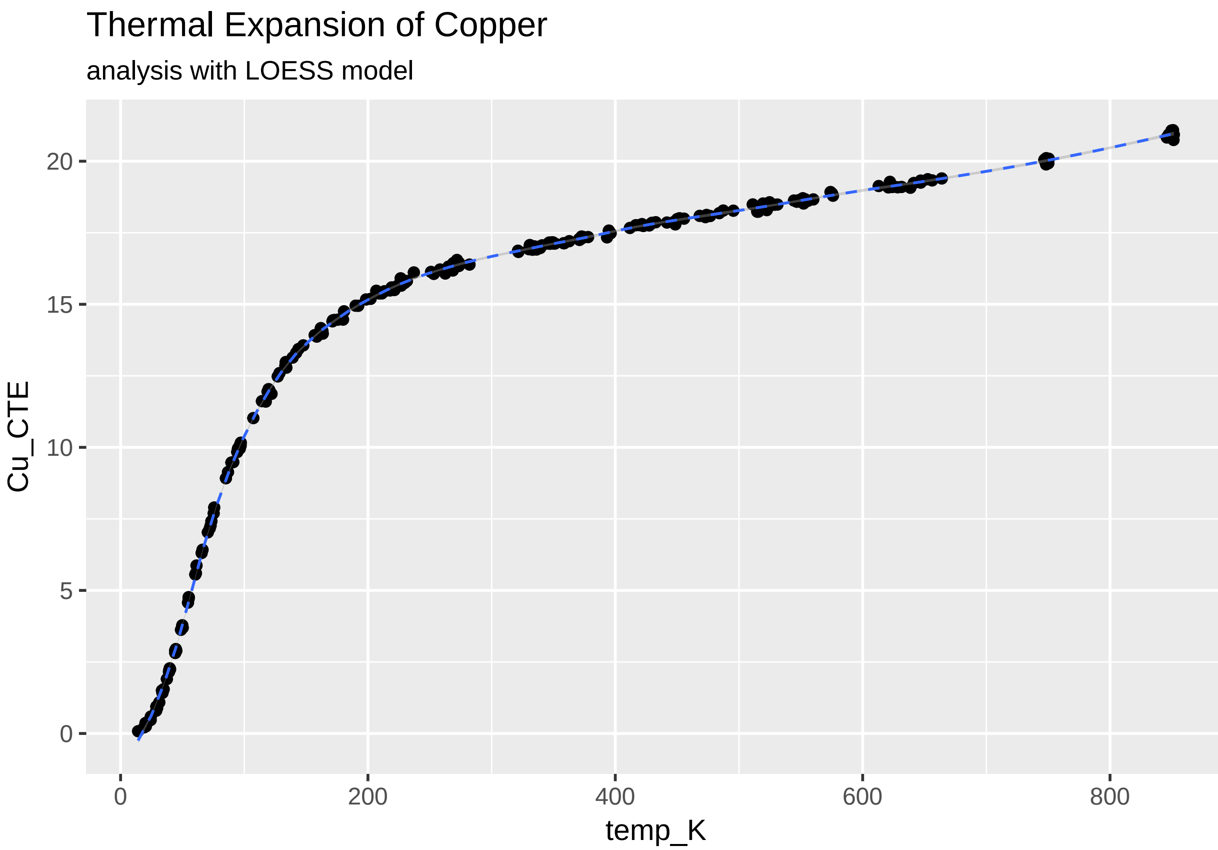
4.3.4.3 Plot the residuals from the C/C model

```
ggplot(augment(mcc_Cu)) +
  geom_point(aes(temp_K, .resid)) +
  geom_hline(aes(yintercept=0)) +
  geom_hline(aes(yintercept=+2*0.082), linetype = "dashed") +
  geom_hline(aes(yintercept=-2*0.082), linetype = "dashed") +
  ggtitle("Thermal Expansion of Copper Residuals (C/C)", subtitle = "+/- 2(Residual Statndard Deviation)
```



4.3.4.4 Finally, lets fit the data with th LOESS method and compare to the C/C model

```r
ggplot(CTECu, aes(temp_K, Cu_CTE)) +
  geom_point() +
  stat_smooth(method = "loess", span = 0.2, linetype = "dashed", size = 0.5) +
  ggtitle("Thermal Expansion of Copper", subtitle = "analysis with LOESS model")
```



We can look at the quality of the fit using the LOESS model direclty

```
mloess_Cu <- loess(Cu_CTE ~ temp_K, CTECu, span = 0.2)

summary(mloess_Cu)
```

```
## Call:
## loess(formula = Cu_CTE ~ temp_K, data = CTECu, span = 0.2)
##
## Number of Observations: 236
## Equivalent Number of Parameters: 15.02
## Residual Standard Error: 0.09039
## Trace of smoother matrix: 16.62  (exact)
##
## Control settings:
##   span     :  0.2
##   degree   :  2
##   family   :  gaussian
##   surface  :  interpolate      cell = 0.2
##   normalize:  TRUE
##  parametric:  FALSE
## drop.square:  FALSE
```

```
augment(mloess_Cu)
```

```
##     Cu_CTE temp_K    .fitted    .se.fit        .resid
## 1    0.591  24.41  0.5893886 0.02340924  1.611396e-03
## 2    1.547  34.82  1.7035504 0.01790375 -1.565504e-01
```

```
## 3    2.902  44.09  2.9075529 0.01956124 -5.552921e-03
## 4    2.894  45.07  3.0426764 0.01971197 -1.486764e-01
## 5    4.703  54.98  4.6006680 0.02177693  1.023320e-01
## 6    6.307  65.51  6.2727258 0.02223313  3.427422e-02
## 7    7.030  70.53  7.0291230 0.02424350  8.770125e-04
## 8    7.898  75.70  7.7223713 0.02316620  1.756287e-01
## 9    9.470  89.57  9.3678543 0.02312016  1.021457e-01
## 10   9.484  91.14  9.5326204 0.02293994 -4.862038e-02
## 11  10.072  96.40 10.0532384 0.02243139  1.876162e-02
## 12  10.163  97.19 10.1306239 0.02241359  3.237612e-02
## 13  11.615 114.26 11.5706405 0.02163989  4.435953e-02
## 14  12.005 120.25 12.0049191 0.02295519  8.089826e-05
## 15  12.478 127.08 12.4554822 0.02242131  2.251778e-02
## 16  12.982 133.55 12.8408206 0.02259261  1.411794e-01
## 17  12.970 133.61 12.8442640 0.02259171  1.257360e-01
## 18  13.926 158.67 13.9557957 0.02266600 -2.979572e-02
## 19  14.452 172.74 14.4192091 0.02364575  3.279085e-02
## 20  14.404 171.31 14.3758966 0.02330013  2.810341e-02
## 21  15.190 202.14 15.2087537 0.02229812 -1.875374e-02
## 22  15.550 220.55 15.6042108 0.02178006 -5.421077e-02
## 23  15.528 221.05 15.6140038 0.02182985 -8.600384e-02
## 24  15.499 221.39 15.6206473 0.02185661 -1.216473e-01
## 25  16.131 250.99 16.1091714 0.01914307  2.182861e-02
## 26  16.438 268.99 16.3398608 0.02202205  9.813921e-02
## 27  16.387 271.80 16.3734167 0.02243304  1.358326e-02
## 28  16.549 271.97 16.3754236 0.02245998  1.735764e-01
## 29  16.872 321.31 16.8639118 0.02213125  8.088198e-03
## 30  16.830 321.69 16.8672482 0.02215705 -3.724821e-02
## 31  16.926 330.14 16.9420265 0.02190354 -1.602647e-02
## 32  16.907 333.03 16.9680613 0.02136006 -6.106125e-02
## 33  16.966 333.47 16.9720533 0.02124855 -6.053335e-03
## 34  17.060 340.77 17.0380252 0.01915004  2.197485e-02
## 35  17.122 345.65 17.0800345 0.01826731  4.196548e-02
## 36  17.311 373.11 17.3095335 0.02446425  1.466498e-03
## 37  17.355 373.79 17.3154949 0.02456639  3.950514e-02
## 38  17.668 411.82 17.6648664 0.02216767  3.133620e-03
## 39  17.767 419.51 17.7287099 0.02243940  3.829015e-02
## 40  17.803 421.59 17.7448828 0.02218004  5.811720e-02
## 41  17.765 422.02 17.7481608 0.02211155  1.683925e-02
## 42  17.768 422.47 17.7515723 0.02203853  1.642772e-02
## 43  17.736 422.61 17.7526301 0.02201581 -1.663008e-02
## 44  17.858 441.75 17.8917493 0.02046968 -3.374927e-02
## 45  17.877 447.41 17.9304245 0.02154521 -5.342452e-02
## 46  17.912 448.70 17.9393108 0.02179951 -2.731083e-02
## 47  18.046 472.89 18.0999205 0.02478693 -5.392046e-02
## 48  18.085 476.69 18.1246418 0.02573444 -3.964181e-02
## 49  18.291 522.47 18.4283764 0.02104353 -1.373764e-01
## 50  18.357 522.62 18.4294420 0.02102223 -7.244200e-02
## 51  18.426 524.43 18.4423209 0.02071864 -1.632094e-02
## 52  18.584 546.75 18.6034919 0.02059169 -1.949187e-02
## 53  18.610 549.53 18.6244942 0.02087707 -1.449418e-02
## 54  18.870 575.29 18.8101404 0.02332466  5.985959e-02
## 55  18.795 576.00 18.8151025 0.02335059 -2.010248e-02
## 56  19.111 625.55 19.1441090 0.02323769 -3.310901e-02
```

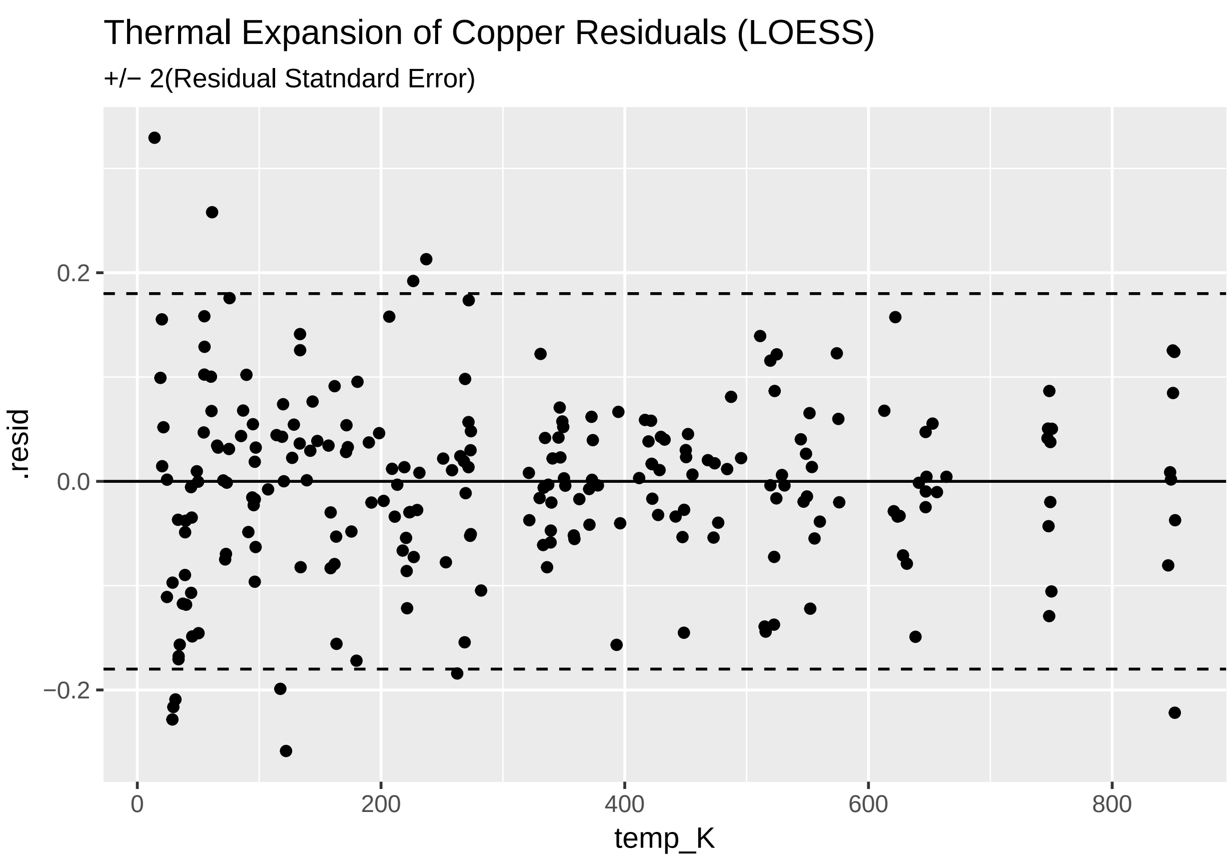
```
## 57    0.367  20.15  0.2116990 0.03085448  1.553010e-01
## 58    0.796  28.78  1.0242942 0.01913119 -2.282942e-01
## 59    0.892  29.57  1.1083269 0.01870637 -2.163269e-01
## 60    1.903  37.41  2.0202956 0.01828370 -1.172956e-01
## 61    2.150  39.12  2.2397801 0.01860451 -8.978013e-02
## 62    3.697  50.24  3.8425643 0.02049747 -1.455643e-01
## 63    5.870  61.38  5.6119898 0.02116325  2.580102e-01
## 64    6.421  66.25  6.3886426 0.02256694  3.235736e-02
## 65    7.422  73.42  7.4232996 0.02409276 -1.299626e-03
## 66    9.944  95.52  9.9668213 0.02241709 -2.282127e-02
## 67   11.023 107.32 11.0307166 0.02035314 -7.716646e-03
## 68   11.870 122.04 12.1285181 0.02287899 -2.585181e-01
## 69   12.786 134.03 12.8683172 0.02258018 -8.231716e-02
## 70   14.067 163.19 14.1199802 0.02247130 -5.298022e-02
## 71   13.974 163.48 14.1297883 0.02245355 -1.557883e-01
## 72   14.462 175.70 14.5100999 0.02412487 -4.809988e-02
## 73   14.464 179.86 14.6359457 0.02384732 -1.719457e-01
## 74   15.381 211.27 15.4148497 0.02158281 -3.384974e-02
## 75   15.483 217.78 15.5493643 0.02140571 -6.636426e-02
## 76   15.590 219.14 15.5764283 0.02159609  1.357166e-02
## 77   16.075 262.52 16.2591976 0.02071960 -1.841976e-01
## 78   16.347 268.01 16.3276907 0.02191144  1.930926e-02
## 79   16.181 268.62 16.3352840 0.02198209 -1.542840e-01
## 80   16.915 336.25 16.9973389 0.02032597 -8.233893e-02
## 81   17.003 337.23 17.0062403 0.02000511 -3.240328e-03
## 82   16.978 339.33 17.0252000 0.01945679 -4.719998e-02
## 83   17.756 427.38 17.7882079 0.02146238 -3.220789e-02
## 84   17.808 428.58 17.7972085 0.02139069  1.079149e-02
## 85   17.868 432.68 17.8279861 0.02096926  4.001391e-02
## 86   18.481 528.99 18.4749238 0.01983701  6.076167e-03
## 87   18.486 531.08 18.4899317 0.01953248 -3.931729e-03
## 88   19.090 628.34 19.1609875 0.02259820 -7.098748e-02
## 89   16.062 253.24 16.1395756 0.01902599 -7.757559e-02
## 90   16.337 273.13 16.3892160 0.02262455 -5.221599e-02
## 91   16.345 273.66 16.3956097 0.02268052 -5.060973e-02
## 92   16.388 282.10 16.4926643 0.02260402 -1.046643e-01
## 93   17.159 346.62 17.0882502 0.01830996  7.074984e-02
## 94   17.116 347.19 17.0930759 0.01837768  2.292413e-02
## 95   17.164 348.78 17.1065650 0.01870251  5.743503e-02
## 96   17.123 351.18 17.1271492 0.01934207 -4.149195e-03
## 97   17.979 450.10 17.9490269 0.02201781  2.997309e-02
## 98   17.974 450.35 17.9507714 0.02204810  2.322857e-02
## 99   18.007 451.92 17.9616731 0.02219872  4.532692e-02
## 100  17.993 455.56 17.9865023 0.02238314  6.497679e-03
## 101  18.523 552.22 18.6450517 0.02125465 -1.220517e-01
## 102  18.669 553.56 18.6552772 0.02147989  1.372284e-02
## 103  18.617 555.74 18.6717598 0.02187462 -5.475980e-02
## 104  19.371 652.59 19.3156329 0.02348120  5.536710e-02
## 105  19.330 656.20 19.3403259 0.02347526 -1.032591e-02
## 106   0.080  14.13 -0.2493998 0.04620251  3.293998e-01
## 107   0.248  20.41  0.2334920 0.03031239  1.450797e-02
## 108   1.089  31.30  1.2980967 0.01809055 -2.090967e-01
## 109   1.418  33.84  1.5885439 0.01783812 -1.705439e-01
## 110   2.278  39.70  2.3157401 0.01871910 -3.774011e-02
```

```
## 111  3.624  48.83  3.6143394 0.02014915  9.660635e-03
## 112  4.574  54.50  4.5271268 0.02174540  4.687324e-02
## 113  5.556  60.41  5.4556488 0.02116604  1.003512e-01
## 114  7.267  72.77  7.3366019 0.02425223 -6.960186e-02
## 115  7.695  75.25  7.6639677 0.02337864  3.103232e-02
## 116  9.136  86.84  9.0681217 0.02259302  6.787825e-02
## 117  9.959  94.88  9.9042264 0.02241509  5.477360e-02
## 118  9.957  96.40 10.0532384 0.02243139 -9.623838e-02
## 119 11.600 117.37 11.7989267 0.02256946 -1.989267e-01
## 120 13.138 139.08 13.1369635 0.02161945  1.036507e-03
## 121 13.564 147.73 13.5252987 0.02034951  3.870126e-02
## 122 13.871 158.63 13.9542333 0.02266416 -8.323330e-02
## 123 13.994 161.84 14.0732954 0.02256812 -7.929537e-02
## 124 14.947 192.11 14.9673282 0.02138477 -2.032820e-02
## 125 15.473 206.76 15.3151149 0.02240447  1.578851e-01
## 126 15.379 209.07 15.3669923 0.02203888  1.200768e-02
## 127 15.455 213.32 15.4582440 0.02127550 -3.244012e-03
## 128 15.908 226.44 15.7159178 0.02202286  1.920822e-01
## 129 16.114 237.12 15.9010123 0.02219862  2.129877e-01
## 130 17.071 330.90 16.9488424 0.02178562  1.221576e-01
## 131 17.135 358.72 17.1902627 0.02080437 -5.526271e-02
## 132 17.282 370.77 17.2893556 0.02397387 -7.355555e-03
## 133 17.368 372.72 17.3061323 0.02439585  6.186773e-02
## 134 17.483 396.24 17.5232072 0.02129821 -4.020717e-02
## 135 17.764 416.59 17.7051251 0.02251976  5.887493e-02
## 136 18.185 484.02 18.1732409 0.02658515  1.175908e-02
## 137 18.271 495.47 18.2487424 0.02413964  2.225757e-02
## 138 18.236 514.78 18.3751895 0.02146154 -1.391895e-01
## 139 18.237 515.65 18.3810788 0.02145767 -1.440788e-01
## 140 18.523 519.47 18.4073027 0.02134191  1.156973e-01
## 141 18.627 544.47 18.5866978 0.02039557  4.030219e-02
## 142 18.665 560.11 18.7036081 0.02255376 -3.860805e-02
## 143 19.086 620.77 19.1146279 0.02451487 -2.862791e-02
## 144  0.214  18.97  0.1147946 0.03344855  9.920537e-02
## 145  0.943  28.93  1.0401129 0.01904309 -9.711289e-02
## 146  1.429  33.91  1.5967118 0.01784020 -1.677118e-01
## 147  2.241  40.03  2.3592580 0.01878496 -1.182580e-01
## 148  2.951  44.66  2.9857198 0.01965356 -3.471981e-02
## 149  3.782  49.87  3.7823568 0.02039192 -3.567609e-04
## 150  4.757  55.16  4.6280117 0.02177796  1.289883e-01
## 151  5.602  60.90  5.5345962 0.02115332  6.740384e-02
## 152  7.169  72.08  7.2438760 0.02434648 -7.487605e-02
## 153  8.920  85.15  8.8765497 0.02201842  4.345026e-02
## 154 10.055  97.06 10.1179699 0.02242059 -6.296988e-02
## 155 12.035 119.63 11.9610696 0.02293092  7.393042e-02
## 156 12.861 133.27 12.8247261 0.02259458  3.627390e-02
## 157 13.436 143.84 13.3595004 0.02046845  7.649958e-02
## 158 14.167 161.91 14.0757601 0.02256296  9.123987e-02
## 159 14.755 180.67 14.6595578 0.02369360  9.544219e-02
## 160 15.168 198.44 15.1218205 0.02179637  4.617947e-02
## 161 15.651 226.86 15.7235686 0.02203213 -7.256862e-02
## 162 15.746 229.65 15.7734814 0.02211877 -2.748143e-02
## 163 16.216 258.27 16.2053310 0.01963096  1.066899e-02
## 164 16.445 273.77 16.3969366 0.02269081  4.806339e-02
```

```
## 165 16.965 339.15 17.0235835 0.01949557 -5.858347e-02
## 166 17.121 350.13 17.1180966 0.01906644  2.903356e-03
## 167 17.206 362.75 17.2230444 0.02170905 -1.704439e-02
## 168 17.250 371.03 17.2915718 0.02403694 -4.157182e-02
## 169 17.339 393.32 17.4957437 0.02172636 -1.567437e-01
## 170 17.793 448.53 17.9381365 0.02176832 -1.451365e-01
## 171 18.123 473.78 18.1056936 0.02501800  1.730638e-02
## 172 18.490 511.12 18.3506976 0.02148107  1.393024e-01
## 173 18.566 524.70 18.4442453 0.02066771  1.217547e-01
## 174 18.645 548.75 18.6185616 0.02078771  2.643841e-02
## 175 18.706 551.64 18.6406162 0.02116414  6.538383e-02
## 176 18.924 574.02 18.8012573 0.02328181  1.227427e-01
## 177 19.100 623.86 19.1337700 0.02369936 -3.377002e-02
## 178  0.375  21.46  0.3231433 0.02823514  5.185672e-02
## 179  0.471  24.33  0.5818836 0.02351969 -1.108836e-01
## 180  1.504  33.43  1.5408606 0.01783466 -3.686060e-02
## 181  2.204  39.22  2.2528267 0.01862413 -4.882670e-02
## 182  2.813  44.18  2.9198854 0.01957625 -1.068854e-01
## 183  4.765  55.02  4.6067413 0.02177754  1.582587e-01
## 184  9.835  94.33  9.8504461 0.02243031 -1.544608e-02
## 185 10.040  96.44 10.0571828 0.02243189 -1.718276e-02
## 186 11.946 118.82 11.9032635 0.02284777  4.273650e-02
## 187 12.596 128.48 12.5416275 0.02240950  5.437255e-02
## 188 13.303 141.94 13.2736796 0.02084791  2.932044e-02
## 189 13.922 156.92 13.8877209 0.02247490  3.427913e-02
## 190 14.440 171.65 14.3861863 0.02338380  5.381374e-02
## 191 14.951 190.00 14.9137132 0.02157548  3.728681e-02
## 192 15.627 223.26 15.6566755 0.02194939 -2.967552e-02
## 193 15.639 223.88 15.6684179 0.02196806 -2.941790e-02
## 194 15.814 231.50 15.8057944 0.02219606  8.205561e-03
## 195 16.315 265.05 16.2908371 0.02136428  2.416293e-02
## 196 16.334 269.44 16.3453544 0.02207639 -1.135442e-02
## 197 16.430 271.78 16.3731807 0.02242985  5.681925e-02
## 198 16.423 273.46 16.3931927 0.02266094  2.980726e-02
## 199 17.024 334.61 16.9824164 0.02089562  4.158361e-02
## 200 17.009 339.79 17.0293219 0.01936440 -2.032188e-02
## 201 17.165 349.52 17.1128736 0.01889826  5.212642e-02
## 202 17.134 358.18 17.1858476 0.02070164 -5.184756e-02
## 203 17.349 377.98 17.3528982 0.02471271 -3.898163e-03
## 204 17.576 394.77 17.5093938 0.02148845  6.660620e-02
## 205 17.848 429.66 17.8054024 0.02132226  4.259762e-02
## 206 18.090 468.22 18.0696956 0.02364603  2.030442e-02
## 207 18.276 487.27 18.1949994 0.02624330  8.100057e-02
## 208 18.404 519.54 18.4077893 0.02133746 -3.789293e-03
## 209 18.519 523.03 18.4323559 0.02096056  8.664405e-02
## 210 19.133 612.99 19.0653124 0.02580219  6.768757e-02
## 211 19.074 638.59 19.2230166 0.02234591 -1.490166e-01
## 212 19.239 641.36 19.2403795 0.02270688 -1.379523e-03
## 213 19.280 622.05 19.1225896 0.02419340  1.574104e-01
## 214 19.101 631.50 19.1799682 0.02215217 -7.896818e-02
## 215 19.398 663.97 19.3936246 0.02323551  4.375380e-03
## 216 19.252 646.90 19.2767622 0.02330845 -2.476219e-02
## 217 19.890 748.29 20.0192002 0.02557545 -1.292002e-01
## 218 20.007 749.21 20.0268252 0.02554427 -1.982522e-02
```

```
## 219 19.929 750.14 20.0345476 0.02551122 -1.055476e-01
## 220 19.268 647.04 19.2777172 0.02331586 -9.717234e-03
## 221 19.324 646.89 19.2766941 0.02330790  4.730595e-02
## 222 20.049 746.90 20.0077110 0.02561713  4.128896e-02
## 223 20.107 748.43 20.0203596 0.02557081  8.664040e-02
## 224 20.062 747.35 20.0114251 0.02560513  5.057491e-02
## 225 20.065 749.27 20.0273230 0.02554218  3.767700e-02
## 226 19.286 647.61 19.2816098 0.02334364  4.390250e-03
## 227 19.972 747.78 20.0149794 0.02559209 -4.297938e-02
## 228 20.088 750.51 20.0376239 0.02549766  5.037605e-02
## 229 20.743 851.37 20.9648418 0.03249677 -2.218418e-01
## 230 20.830 845.97 20.9105761 0.03024912 -8.057613e-02
## 231 20.935 847.54 20.9262958 0.03086868  8.704217e-03
## 232 21.035 849.93 20.9503176 0.03186570  8.468244e-02
## 233 20.930 851.61 20.9672661 0.03260409 -3.726611e-02
## 234 21.074 849.75 20.9485047 0.03178839  1.254953e-01
## 235 21.085 850.98 20.9609045 0.03232366  1.240955e-01
## 236 20.935 848.23 20.9332198 0.03114992  1.780210e-03
```

```
ggplot(augment(mloess_Cu)) +
  geom_point(aes(temp_K, .resid)) +
  geom_hline(aes(yintercept=0)) +
  geom_hline(aes(yintercept=+2*(0.09)), linetype = "dashed") +
  geom_hline(aes(yintercept=-2*(0.09)), linetype = "dashed") +
  ggtitle("Thermal Expansion of Copper Residuals (LOESS)", subtitle = "+/- 2(Residual Statndard Error)")
```



The C/C is slightly bette, but rquired signifiantly more work.

4.3.4.5 Progaming with `case_when()`

Looking at the orginal data set, we can see that there are two sets of data. We might want to label these as "run1" and "run2." we could do this in Excel using an `IF()` function. In the **tidyverse**, we can use the `case_when()` function.

```
load_cell_fit_2
```

```
##    Deflection    Load Load_squared   .fitted      .se.fit        .resid
## 1     0.11019  150000   2.2500e+10 0.1104113 8.834303e-05 -2.213214e-04
## 2     0.21956  300000   9.0000e+10 0.2200068 7.185366e-05 -4.468402e-04
## 3     0.32949  450000   2.0250e+11 0.3294601 5.888246e-05  2.987782e-05
## 4     0.43899  600000   3.6000e+11 0.4387712 4.986859e-05  2.188327e-04
## 5     0.54803  750000   5.6250e+11 0.5479400 4.495244e-05  9.002444e-05
## 6     0.65694  900000   8.1000e+11 0.6569665 4.354343e-05 -2.654699e-05
## 7     0.76562 1050000   1.1025e+12 0.7658509 4.437259e-05 -2.308816e-04
## 8     0.87487 1200000   1.4400e+12 0.8745930 4.609026e-05  2.770207e-04
## 9     0.98292 1350000   1.8225e+12 0.9831928 4.770597e-05 -2.728402e-04
## 10    1.09146 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.904643e-04
## 11    1.20001 1650000   2.7225e+12 1.1999659 4.864177e-05  4.414850e-05
## 12    1.30822 1800000   3.2400e+12 1.3081390 4.770597e-05  8.099812e-05
## 13    1.41599 1950000   3.8025e+12 1.4161699 4.609026e-05 -1.799154e-04
## 14    1.52399 2100000   4.4100e+12 1.5240586 4.437259e-05 -6.859211e-05
## 15    1.63194 2250000   5.0625e+12 1.6318050 4.354343e-05  1.349680e-04
## 16    1.73947 2400000   5.7600e+12 1.7394092 4.495244e-05  6.076504e-05
## 17    1.84646 2550000   6.5025e+12 1.8468712 4.986859e-05 -4.112011e-04
## 18    1.95392 2700000   7.2900e+12 1.9541909 5.888246e-05 -2.709305e-04
## 19    2.06128 2850000   8.1225e+12 2.0613684 7.185366e-05 -8.842293e-05
## 20    2.16844 3000000   9.0000e+12 2.1684037 8.834303e-05  3.632143e-05
## 21    0.11052  150000   2.2500e+10 0.1104113 8.834303e-05  1.086786e-04
## 22    0.22018  300000   9.0000e+10 0.2200068 7.185366e-05  1.731598e-04
## 23    0.32939  450000   2.0250e+11 0.3294601 5.888246e-05 -7.012218e-05
## 24    0.43886  600000   3.6000e+11 0.4387712 4.986859e-05  8.883271e-05
## 25    0.54798  750000   5.6250e+11 0.5479400 4.495244e-05  4.002444e-05
## 26    0.65739  900000   8.1000e+11 0.6569665 4.354343e-05  4.234530e-04
## 27    0.76596 1050000   1.1025e+12 0.7658509 4.437259e-05  1.091184e-04
## 28    0.87474 1200000   1.4400e+12 0.8745930 4.609026e-05  1.470207e-04
## 29    0.98300 1350000   1.8225e+12 0.9831928 4.770597e-05 -1.928402e-04
## 30    1.09150 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.504643e-04
## 31    1.20004 1650000   2.7225e+12 1.1999659 4.864177e-05  7.414850e-05
## 32    1.30818 1800000   3.2400e+12 1.3081390 4.770597e-05  4.099812e-05
## 33    1.41613 1950000   3.8025e+12 1.4161699 4.609026e-05 -3.991541e-05
## 34    1.52408 2100000   4.4100e+12 1.5240586 4.437259e-05  2.140789e-05
## 35    1.63159 2250000   5.0625e+12 1.6318050 4.354343e-05 -2.150320e-04
## 36    1.73965 2400000   5.7600e+12 1.7394092 4.495244e-05  2.407650e-04
## 37    1.84696 2550000   6.5025e+12 1.8468712 4.986859e-05  8.879887e-05
## 38    1.95445 2700000   7.2900e+12 1.9541909 5.888246e-05  2.590695e-04
## 39    2.06177 2850000   8.1225e+12 2.0613684 7.185366e-05  4.015771e-04
## 40    2.16829 3000000   9.0000e+12 2.1684037 8.834303e-05 -1.136786e-04
##          .hat       .sigma      .cooksd .std.resid
## 1  0.18538961 0.0002039531 0.1083557670 -1.1951404
```

```
## 2  0.12264183 0.0001922123 0.2518888277 -2.3250603
## 3  0.08235931 0.0002079426 0.0006913290  0.1520138
## 4  0.05907382 0.0002045811 0.0253004370  1.0995244
## 5  0.04800068 0.0002074384 0.0033987136  0.4496893
## 6  0.04503873 0.0002079583 0.0002755909 -0.1324015
## 7  0.04677033 0.0002042395 0.0217256885 -1.1525531
## 8  0.05046138 0.0002025394 0.0340077449  1.3855631
## 9  0.05406129 0.0002026849 0.0356120360 -1.3672481
## 10 0.05620301 0.0002054251 0.0181237925 -0.9555308
## 11 0.05620301 0.0002078697 0.0009737642  0.2214864
## 12 0.05406129 0.0002075440 0.0031385560  0.4058952
## 13 0.05046138 0.0002057188 0.0143446582 -0.8998756
## 14 0.04677033 0.0002076778 0.0019175348 -0.3424095
## 15 0.04503873 0.0002067300 0.0071235449  0.6731448
## 16 0.04800068 0.0002077485 0.0015484646  0.3035330
## 17 0.05907382 0.0001956411 0.0893330479 -2.0660791
## 18 0.08235931 0.0002025961 0.0568463516 -1.3784528
## 19 0.12264183 0.0002074117 0.0098635717 -0.4600943
## 20 0.18538961 0.0002078994 0.0029183068  0.1961365
## 21 0.18538961 0.0002070372 0.0261272044  0.5868666
## 22 0.12264183 0.0002057130 0.0378266955  0.9010087
## 23 0.08235931 0.0002076495 0.0038080074 -0.3567710
## 24 0.05907382 0.0002074468 0.0041691675  0.4463397
## 25 0.04800068 0.0002078952 0.0006718065  0.1999297
## 26 0.04503873 0.0001950675 0.0701204501  2.1119458
## 27 0.04677033 0.0002071719 0.0048527858  0.5447155
## 28 0.05046138 0.0002064820 0.0095787822  0.7353473
## 29 0.05406129 0.0002053659 0.0177899720 -0.9663547
## 30 0.05620301 0.0002063997 0.0113106830 -0.7548568
## 31 0.05620301 0.0002076183 0.0027467977  0.3719919
## 32 0.05406129 0.0002078889 0.0008040960  0.2054485
## 33 0.05046138 0.0002078955 0.0007060488 -0.1996433
## 34 0.04677033 0.0002079755 0.0001867854  0.1068675
## 35 0.04503873 0.0002047490 0.0180817416 -1.0724586
## 36 0.04800068 0.0002039013 0.0243097555  1.2026674
## 37 0.05907382 0.0002074473 0.0041659922  0.4461697
## 38 0.08235931 0.0002030652 0.0519780155  1.3181064
## 39 0.12264183 0.0001953495 0.2034427365  2.0895409
## 40 0.18538961 0.0002069456 0.0285865876 -0.6138667
```

```r
load_cell_fit_2_runs <- load_cell_fit_2 %>%
  mutate(
    run = case_when(seq_along(Load) <= 20 ~ "run1",
                    seq_along(Load) > 20 ~ "run2"))

load_cell_fit_2_runs
```

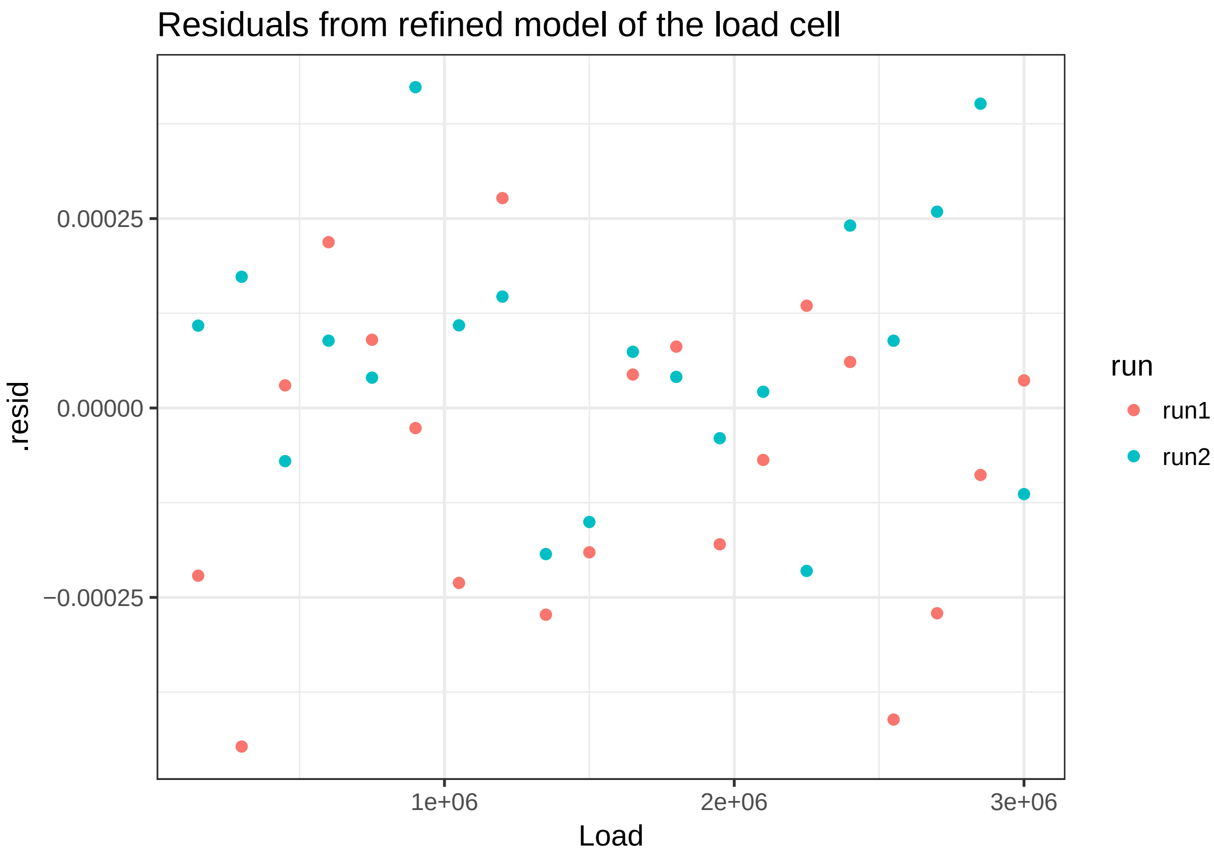
```
##    Deflection    Load Load_squared   .fitted      .se.fit        .resid
## 1     0.11019  150000   2.2500e+10 0.1104113 8.834303e-05 -2.213214e-04
## 2     0.21956  300000   9.0000e+10 0.2200068 7.185366e-05 -4.468402e-04
## 3     0.32949  450000   2.0250e+11 0.3294601 5.888246e-05  2.987782e-05
## 4     0.43899  600000   3.6000e+11 0.4387712 4.986859e-05  2.188327e-04
## 5     0.54803  750000   5.6250e+11 0.5479400 4.495244e-05  9.002444e-05
## 6     0.65694  900000   8.1000e+11 0.6569665 4.354343e-05 -2.654699e-05
## 7     0.76562 1050000   1.1025e+12 0.7658509 4.437259e-05 -2.308816e-04
```

```
## 8     0.87487 1200000   1.4400e+12 0.8745930 4.609026e-05  2.770207e-04
## 9     0.98292 1350000   1.8225e+12 0.9831928 4.770597e-05 -2.728402e-04
## 10    1.09146 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.904643e-04
## 11    1.20001 1650000   2.7225e+12 1.1999659 4.864177e-05  4.414850e-05
## 12    1.30822 1800000   3.2400e+12 1.3081390 4.770597e-05  8.099812e-05
## 13    1.41599 1950000   3.8025e+12 1.4161699 4.609026e-05 -1.799154e-04
## 14    1.52399 2100000   4.4100e+12 1.5240586 4.437259e-05 -6.859211e-05
## 15    1.63194 2250000   5.0625e+12 1.6318050 4.354343e-05  1.349680e-04
## 16    1.73947 2400000   5.7600e+12 1.7394092 4.495244e-05  6.076504e-05
## 17    1.84646 2550000   6.5025e+12 1.8468712 4.986859e-05 -4.112011e-04
## 18    1.95392 2700000   7.2900e+12 1.9541909 5.888246e-05 -2.709305e-04
## 19    2.06128 2850000   8.1225e+12 2.0613684 7.185366e-05 -8.842293e-05
## 20    2.16844 3000000   9.0000e+12 2.1684037 8.834303e-05  3.632143e-05
## 21    0.11052  150000   2.2500e+10 0.1104113 8.834303e-05  1.086786e-04
## 22    0.22018  300000   9.0000e+10 0.2200068 7.185366e-05  1.731598e-04
## 23    0.32939  450000   2.0250e+11 0.3294601 5.888246e-05 -7.012218e-05
## 24    0.43886  600000   3.6000e+11 0.4387712 4.986859e-05  8.883271e-05
## 25    0.54798  750000   5.6250e+11 0.5479400 4.495244e-05  4.002444e-05
## 26    0.65739  900000   8.1000e+11 0.6569665 4.354343e-05  4.234530e-04
## 27    0.76596 1050000   1.1025e+12 0.7658509 4.437259e-05  1.091184e-04
## 28    0.87474 1200000   1.4400e+12 0.8745930 4.609026e-05  1.470207e-04
## 29    0.98300 1350000   1.8225e+12 0.9831928 4.770597e-05 -1.928402e-04
## 30    1.09150 1500000   2.2500e+12 1.0916505 4.864177e-05 -1.504643e-04
## 31    1.20004 1650000   2.7225e+12 1.1999659 4.864177e-05  7.414850e-05
## 32    1.30818 1800000   3.2400e+12 1.3081390 4.770597e-05  4.099812e-05
## 33    1.41613 1950000   3.8025e+12 1.4161699 4.609026e-05 -3.991541e-05
## 34    1.52408 2100000   4.4100e+12 1.5240586 4.437259e-05  2.140789e-05
## 35    1.63159 2250000   5.0625e+12 1.6318050 4.354343e-05 -2.150320e-04
## 36    1.73965 2400000   5.7600e+12 1.7394092 4.495244e-05  2.407650e-04
## 37    1.84696 2550000   6.5025e+12 1.8468712 4.986859e-05  8.879887e-05
## 38    1.95445 2700000   7.2900e+12 1.9541909 5.888246e-05  2.590695e-04
## 39    2.06177 2850000   8.1225e+12 2.0613684 7.185366e-05  4.015771e-04
## 40    2.16829 3000000   9.0000e+12 2.1684037 8.834303e-05 -1.136786e-04
##          .hat       .sigma      .cooksd .std.resid  run
## 1  0.18538961 0.0002039531 0.1083557670 -1.1951404 run1
## 2  0.12264183 0.0001922123 0.2518888277 -2.3250603 run1
## 3  0.08235931 0.0002079426 0.0006913290  0.1520138 run1
## 4  0.05907382 0.0002045811 0.0253004370  1.0995244 run1
## 5  0.04800068 0.0002074384 0.0033987136  0.4496893 run1
## 6  0.04503873 0.0002079583 0.0002755909 -0.1324015 run1
## 7  0.04677033 0.0002042395 0.0217256885 -1.1525531 run1
## 8  0.05046138 0.0002025394 0.0340077449  1.3855631 run1
## 9  0.05406129 0.0002026849 0.0356120360 -1.3672481 run1
## 10 0.05620301 0.0002054251 0.0181237925 -0.9555308 run1
## 11 0.05620301 0.0002078697 0.0009737642  0.2214864 run1
## 12 0.05406129 0.0002075440 0.0031385560  0.4058952 run1
## 13 0.05046138 0.0002057188 0.0143446582 -0.8998756 run1
## 14 0.04677033 0.0002076778 0.0019175348 -0.3424095 run1
## 15 0.04503873 0.0002067300 0.0071235449  0.6731448 run1
## 16 0.04800068 0.0002077485 0.0015484646  0.3035330 run1
## 17 0.05907382 0.0001956411 0.0893330479 -2.0660791 run1
## 18 0.08235931 0.0002025961 0.0568463516 -1.3784528 run1
## 19 0.12264183 0.0002074117 0.0098635717 -0.4600943 run1
## 20 0.18538961 0.0002078994 0.0029183068  0.1961365 run1
```

```
## 21 0.18538961 0.0002070372 0.0261272044  0.5868666 run2
## 22 0.12264183 0.0002057130 0.0378266955  0.9010087 run2
## 23 0.08235931 0.0002076495 0.0038080074 -0.3567710 run2
## 24 0.05907382 0.0002074468 0.0041691675  0.4463397 run2
## 25 0.04800068 0.0002078952 0.0006718065  0.1999297 run2
## 26 0.04503873 0.0001950675 0.0701204501  2.1119458 run2
## 27 0.04677033 0.0002071719 0.0048527858  0.5447155 run2
## 28 0.05046138 0.0002064820 0.0095787822  0.7353473 run2
## 29 0.05406129 0.0002053659 0.0177899720 -0.9663547 run2
## 30 0.05620301 0.0002063997 0.0113106830 -0.7548568 run2
## 31 0.05620301 0.0002076183 0.0027467977  0.3719919 run2
## 32 0.05406129 0.0002078889 0.0008040960  0.2054485 run2
## 33 0.05046138 0.0002078955 0.0007060488 -0.1996433 run2
## 34 0.04677033 0.0002079755 0.0001867854  0.1068675 run2
## 35 0.04503873 0.0002047490 0.0180817416 -1.0724586 run2
## 36 0.04800068 0.0002039013 0.0243097555  1.2026674 run2
## 37 0.05907382 0.0002074473 0.0041659922  0.4461697 run2
## 38 0.08235931 0.0002030652 0.0519780155  1.3181064 run2
## 39 0.12264183 0.0001953495 0.2034427365  2.0895409 run2
## 40 0.18538961 0.0002069456 0.0285865876 -0.6138667 run2
```

4.3.4.6 EDA of load cell data by run

```r
ggplot(load_cell_fit_2_runs) +
  geom_point(aes(Load, .resid, colour = run)) +
  ggtitle("Residuals from refined model of the load cell") +
  theme_bw()
```



4.4 Applying models to multiple datasets

4.4.1 Revisting the Ascombe dataset

```
x_anscombe <- anscombe %>%  # results will be storred into a new object x_anscombe; we start with the o
  dplyr::select(x1, x2, x3, x4) %>%  # select the columns we want to work with
  rename(group1 = x1, group2 = x2, group3 = x3, group4 = x4) %>% # rename the values using a generic he
  gather(key = group, value = x_values, group1, group2, group3, group4) # gather the columns into rows

x_anscombe
```

```
##      group x_values
## 1   group1       10
## 2   group1        8
## 3   group1       13
## 4   group1        9
## 5   group1       11
## 6   group1       14
## 7   group1        6
## 8   group1        4
## 9   group1       12
## 10  group1        7
## 11  group1        5
## 12  group2       10
## 13  group2        8
```

```
## 14 group2       13
## 15 group2        9
## 16 group2       11
## 17 group2       14
## 18 group2        6
## 19 group2        4
## 20 group2       12
## 21 group2        7
## 22 group2        5
## 23 group3       10
## 24 group3        8
## 25 group3       13
## 26 group3        9
## 27 group3       11
## 28 group3       14
## 29 group3        6
## 30 group3        4
## 31 group3       12
## 32 group3        7
## 33 group3        5
## 34 group4        8
## 35 group4        8
## 36 group4        8
## 37 group4        8
## 38 group4        8
## 39 group4        8
## 40 group4        8
## 41 group4       19
## 42 group4        8
## 43 group4        8
## 44 group4        8
```

```
y_anscombe <- anscombe %>%
  dplyr::select(y1, y2, y3, y4) %>%
  gather(key = group, value = y_values, y1, y2, y3, y4) %>% # I don't need to rename the columns as I w
  dplyr::select(y_values)

y_anscombe
```

```
##    y_values
## 1      8.04
## 2      6.95
## 3      7.58
## 4      8.81
## 5      8.33
## 6      9.96
## 7      7.24
## 8      4.26
## 9     10.84
## 10     4.82
## 11     5.68
## 12     9.14
## 13     8.14
## 14     8.74
## 15     8.77
```

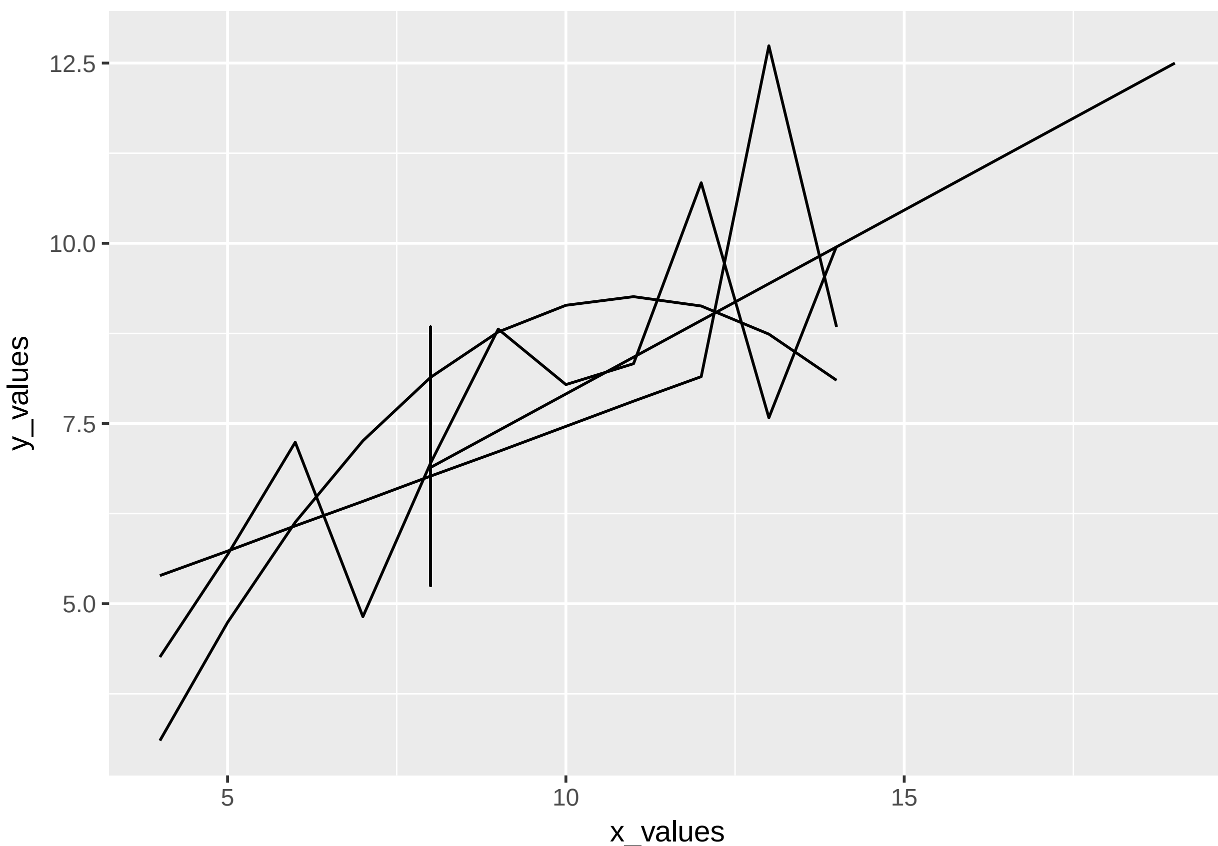
```
## 16     9.26
## 17     8.10
## 18     6.13
## 19     3.10
## 20     9.13
## 21     7.26
## 22     4.74
## 23     7.46
## 24     6.77
## 25    12.74
## 26     7.11
## 27     7.81
## 28     8.84
## 29     6.08
## 30     5.39
## 31     8.15
## 32     6.42
## 33     5.73
## 34     6.58
## 35     5.76
## 36     7.71
## 37     8.84
## 38     8.47
## 39     7.04
## 40     5.25
## 41    12.50
## 42     5.56
## 43     7.91
## 44     6.89
```

```
anscombe_tidy <- bind_cols(x_anscombe, y_anscombe)
anscombe_tidy
```

```
##     group x_values y_values
## 1  group1       10     8.04
## 2  group1        8     6.95
## 3  group1       13     7.58
## 4  group1        9     8.81
## 5  group1       11     8.33
## 6  group1       14     9.96
## 7  group1        6     7.24
## 8  group1        4     4.26
## 9  group1       12    10.84
## 10 group1        7     4.82
## 11 group1        5     5.68
## 12 group2       10     9.14
## 13 group2        8     8.14
## 14 group2       13     8.74
## 15 group2        9     8.77
## 16 group2       11     9.26
## 17 group2       14     8.10
## 18 group2        6     6.13
## 19 group2        4     3.10
## 20 group2       12     9.13
## 21 group2        7     7.26
```

```
## 22 group2        5     4.74
## 23 group3       10     7.46
## 24 group3        8     6.77
## 25 group3       13    12.74
## 26 group3        9     7.11
## 27 group3       11     7.81
## 28 group3       14     8.84
## 29 group3        6     6.08
## 30 group3        4     5.39
## 31 group3       12     8.15
## 32 group3        7     6.42
## 33 group3        5     5.73
## 34 group4        8     6.58
## 35 group4        8     5.76
## 36 group4        8     7.71
## 37 group4        8     8.84
## 38 group4        8     8.47
## 39 group4        8     7.04
## 40 group4        8     5.25
## 41 group4       19    12.50
## 42 group4        8     5.56
## 43 group4        8     7.91
## 44 group4        8     6.89
```

```r
ggplot(anscombe_tidy, aes(x_values, y_values)) +
  geom_line(aes(group = group))
```



In chapter 1, we constructed models for each group; however, for large datasets, we'd like to be able to streamline this process.

Following the example from chapter 11 of Hadley Wickham:

```
anscombe_models <- anscombe_tidy %>%
  group_by(group) %>%
  do(mod = lm(y_values ~ x_values, data = ., na.action = na.exclude
     )) %>%
  ungroup()

anscombe_models
```

```
## # A tibble: 4 x 2
##   group  mod
## * <chr>  <list>
## 1 group1 <S3: lm>
## 2 group2 <S3: lm>
## 3 group3 <S3: lm>
## 4 group4 <S3: lm>
```

4.4.2 Model-level summaries

```
model_summary <- anscombe_models %>%
  rowwise() %>%
  glance(mod)

model_summary
```

```
## # A tibble: 4 x 12
## # Groups:   group [4]
##   group r.squared adj.r.squared sigma statistic p.value    df logLik   AIC
##   <chr>     <dbl>         <dbl> <dbl>     <dbl>   <dbl> <int>  <dbl> <dbl>
## 1 grou~     0.667         0.629  1.24      18.0 0.00217     2  -16.8  39.7
## 2 grou~     0.666         0.629  1.24      18.0 0.00218     2  -16.8  39.7
## 3 grou~     0.666         0.629  1.24      18.0 0.00218     2  -16.8  39.7
## 4 grou~     0.667         0.630  1.24      18.0 0.00216     2  -16.8  39.7
## # ... with 3 more variables: BIC <dbl>, deviance <dbl>, df.residual <int>
```

4.4.3 Coefficient-level summaries

```
coefficient_summary <- anscombe_models %>%
  rowwise() %>%
  tidy(mod)

coefficient_summary
```

```
## # A tibble: 8 x 6
## # Groups:   group [4]
##   group  term        estimate std.error statistic p.value
##   <chr>  <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 group1 (Intercept)    3.00      1.12       2.67 0.0257
## 2 group1 x_values       0.500     0.118      4.24 0.00217
## 3 group2 (Intercept)    3.00      1.13       2.67 0.0258
```

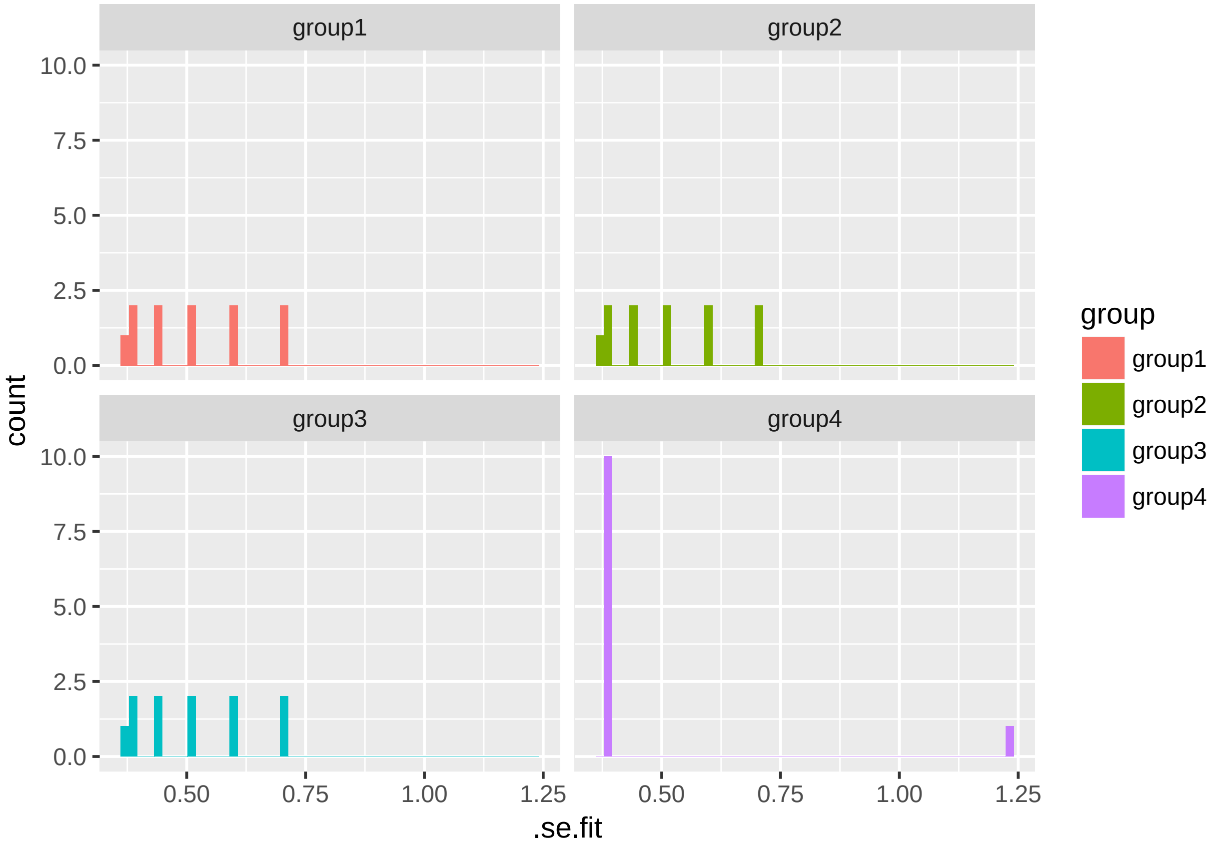
```
## 4 group2 x_values       0.500     0.118      4.24 0.00218
## 5 group3 (Intercept)    3.00      1.12       2.67 0.0256
## 6 group3 x_values       0.500     0.118      4.24 0.00218
## 7 group4 (Intercept)    3.00      1.12       2.67 0.0256
## 8 group4 x_values       0.500     0.118      4.24 0.00216
```

4.4.4 Observation Data

```
observation_summary <- anscombe_models %>%
  rowwise() %>%
  augment(mod)

observation_summary
```

```
## # A tibble: 44 x 10
## # Groups:   group [4]
##    group  y_values x_values .fitted .se.fit  .resid   .hat .sigma  .cooksd
##    <chr>     <dbl>    <dbl>   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>    <dbl>
##  1 group1     8.04      10.    8.00   0.391  0.0390 0.100    1.31  6.14e-5
##  2 group1     6.95       8.    7.00   0.391 -0.0508 0.100    1.31  1.04e-4
##  3 group1     7.58      13.    9.50   0.601 -1.92   0.236    1.06  4.89e-1
##  4 group1     8.81       9.    7.50   0.373  1.31   0.0909   1.22  6.16e-2
##  5 group1     8.33      11.    8.50   0.441 -0.171  0.127    1.31  1.60e-3
##  6 group1     9.96      14.   10.0    0.698 -0.0414 0.318    1.31  3.83e-4
##  7 group1     7.24       6.    6.00   0.514  1.24   0.173    1.22  1.27e-1
##  8 group1     4.26       4.    5.00   0.698 -0.740  0.318    1.27  1.23e-1
##  9 group1    10.8       12.    9.00   0.514  1.84   0.173    1.10  2.79e-1
## 10 group1     4.82       7.    6.50   0.441 -1.68   0.127    1.15  1.54e-1
## # ... with 34 more rows, and 1 more variable: .std.resid <dbl>
```

```
ggplot(observation_summary, aes(.se.fit, fill = group)) +
  geom_histogram(bins = 50) +
  facet_wrap(~ group)
```



Having model information for each dataset can make the overall analysis much more efficient.

Chapter 5

Process Improvment

5.1 Packages used in this chapter

```r
library(tidyverse)
library(ggplot2)
library(broom)
```

5.2 Case Stuidies

5.2.1 Eddy current probe sensitivity

Eddy current probe sensitivity

5.2.1.1 Background

> The data for this case study is a subset of a study performed by Capobianco, Splett, and Iyer. Capobianco was a member of the NIST Electromagnetics Division and Splett and Iyer were members of the NIST Statistical Engineering Division at the time of this study.
>
> The goal of this project is to develop a nondestructive portable device for detecting cracks and fractures in metals. A primary application would be the detection of defects in airplane wings. The internal mechanism of the detector would be for sensing crack-induced changes in the detector's electromagnetic field, which would in turn result in changes in the impedance level of the detector. This change of impedance is termed "sensitivity" and it is a sub-goal of this experiment to maximize such sensitivity as the detector is moved from an unflawed region to a flawed region on the metal.

5.2.1.2 Statistical goals

> The case study illustrates the analysis of a 23 full factorial experimental design. The specific statistical goals of the experiment are: (1) Determine the important factors that affect sensitivity. (2) Determine the settings that maximize sensitivity. (3) Determine a predicition equation that functionally relates sensitivity to various factors.

5.2.1.3 Data

```
eddy_probe <- read_table2("NIST data/SPLETT3.DAT",
                          skip = 25, col_names = FALSE, col_types = "diiii") %>%
  rename(probe_impedance = X1, number_turns = X2, winding_distance = X3, wire_gauge = X4, run_sequence =
eddy_probe
```

```
## # A tibble: 8 x 5
##   probe_impedance number_turns winding_distance wire_gauge run_sequence
##             <dbl>        <int>            <int>      <int>        <int>
## 1           1.70            -1               -1         -1            2
## 2           4.57             1               -1         -1            8
## 3           0.550           -1                1         -1            3
## 4           3.39             1                1         -1            6
## 5           1.51            -1               -1          1            7
## 6           4.59             1               -1          1            1
## 7           0.670           -1                1          1            4
## 8           4.29             1                1          1            5
```

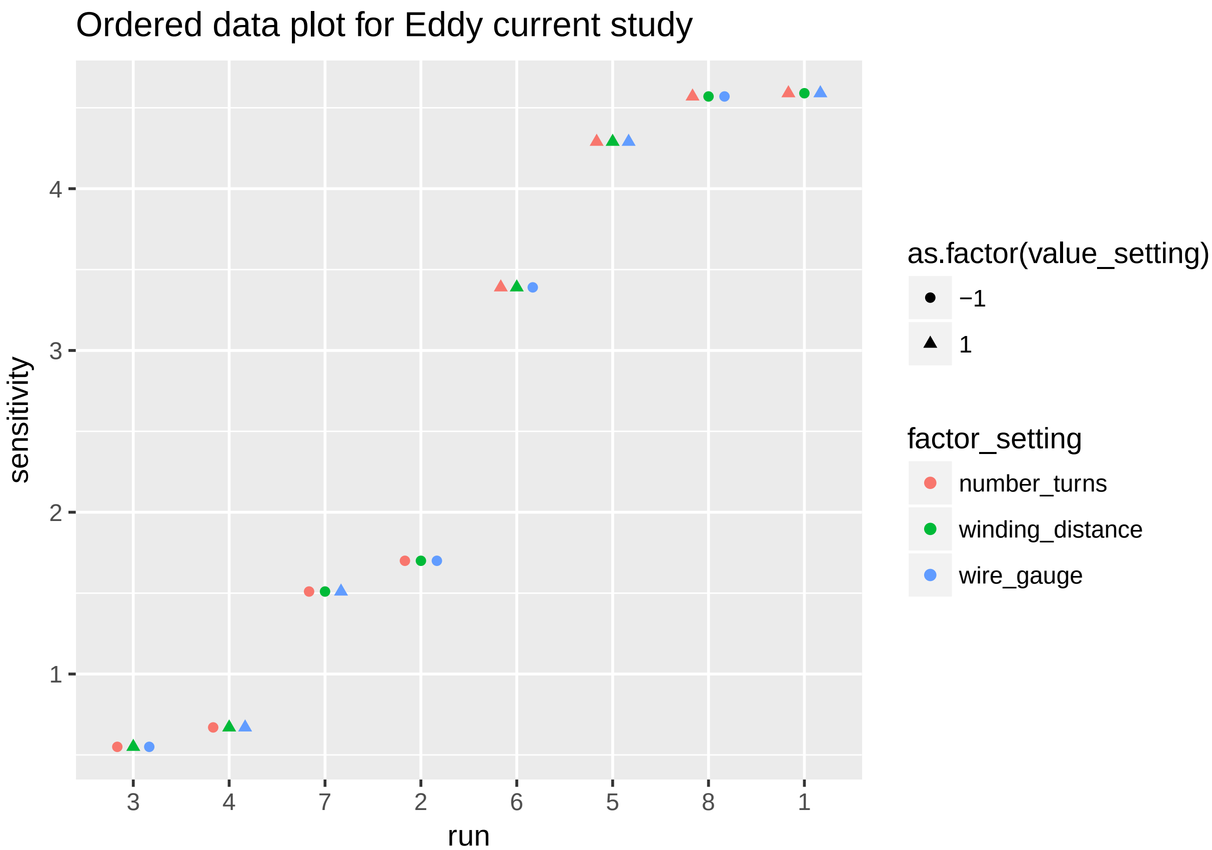
5.2.1.4 Ordered data plot

There are several differnet ways we could structure an ordered data plot; below is an example.

```
eddy_probe_tidy <- eddy_probe %>%
  gather(key = factor_setting, value = value_setting, number_turns, winding_distance, wire_gauge)
eddy_probe_tidy
```

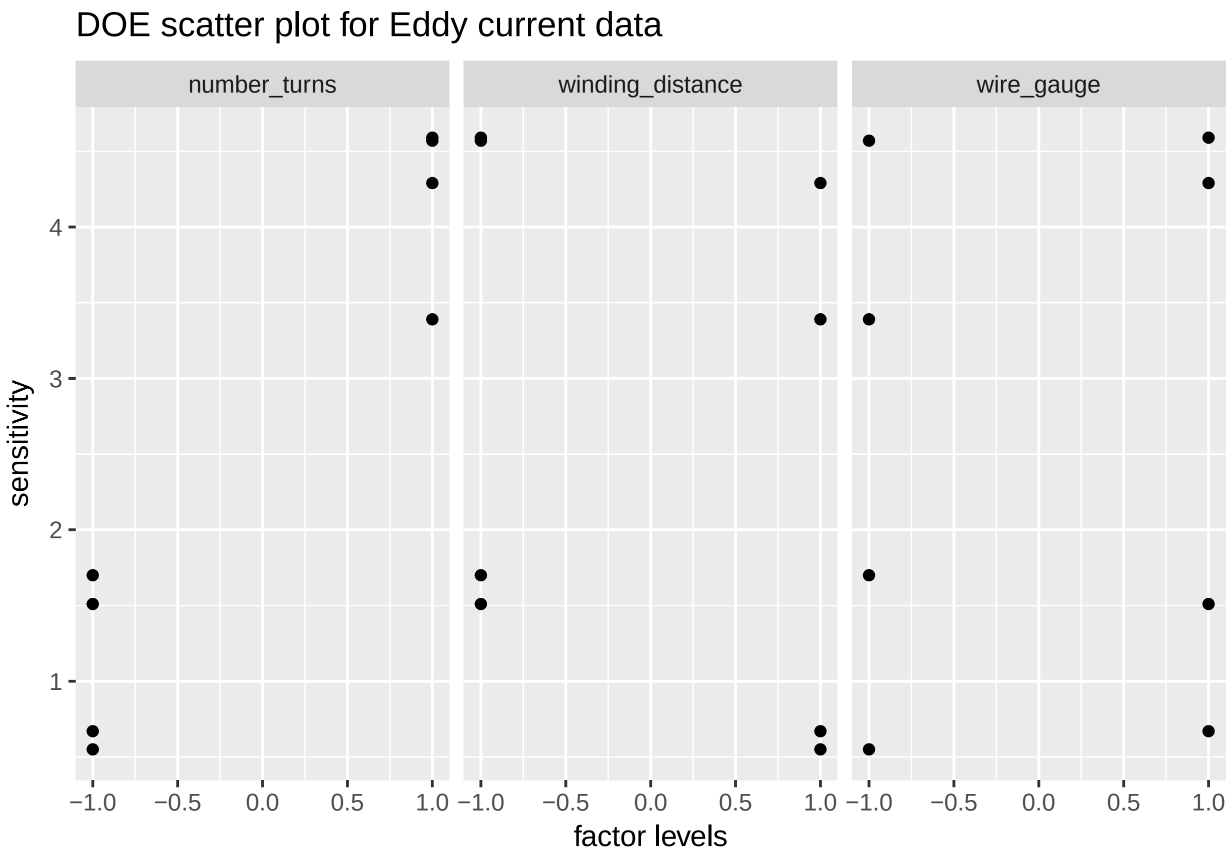
```
## # A tibble: 24 x 4
##    probe_impedance run_sequence factor_setting   value_setting
##              <dbl>        <int> <chr>                    <int>
##  1           1.70             2 number_turns                -1
##  2           4.57             8 number_turns                 1
##  3           0.550            3 number_turns                -1
##  4           3.39             6 number_turns                 1
##  5           1.51             7 number_turns                -1
##  6           4.59             1 number_turns                 1
##  7           0.670            4 number_turns                -1
##  8           4.29             5 number_turns                 1
##  9           1.70             2 winding_distance            -1
## 10           4.57             8 winding_distance            -1
## # ... with 14 more rows
```

```
ggplot(eddy_probe_tidy) +
  geom_point(aes(reorder(run_sequence, probe_impedance), probe_impedance, colour = factor_setting, shape
  labs(title = "Ordered data plot for Eddy current study", y = "sensitivity", x = "run")
```



5.2.1.5 DOE scatter plot

```
ggplot(eddy_probe_tidy) +
  geom_point(aes(value_setting, probe_impedance)) +
  facet_wrap(~factor_setting) +
  labs(title = "DOE scatter plot for Eddy current data", y = "sensitivity", x = "factor levels")
```

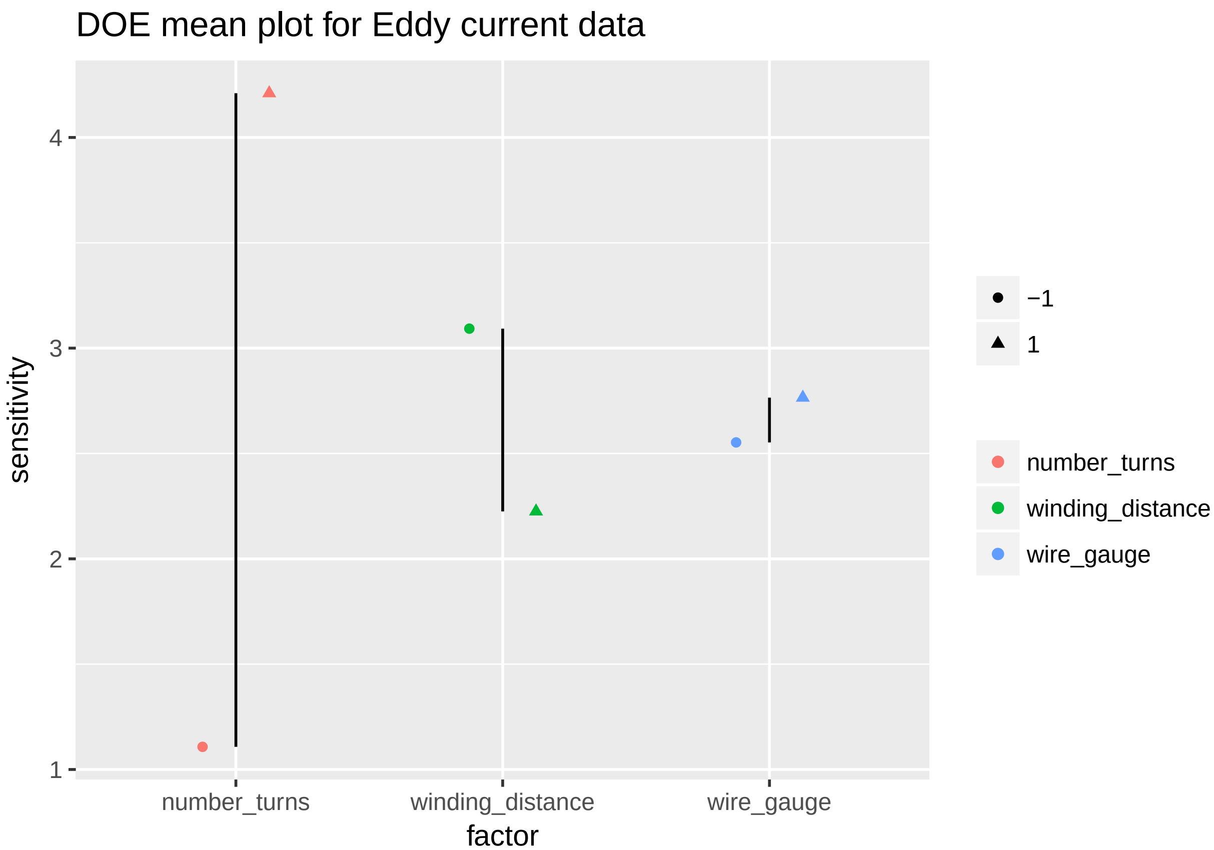


5.2.1.6 DOE mean plot

```
eddy_probe_means <- eddy_probe_tidy %>%
  group_by(factor_setting, value_setting) %>%
  summarise(n = n(), average_factor = mean(probe_impedance))
eddy_probe_means
```

```
## # A tibble: 6 x 4
## # Groups:   factor_setting [?]
##   factor_setting   value_setting     n average_factor
##   <chr>                    <int> <int>          <dbl>
## 1 number_turns                -1     4           1.11
## 2 number_turns                 1     4           4.21
## 3 winding_distance            -1     4           3.09
## 4 winding_distance             1     4           2.22
## 5 wire_gauge                  -1     4           2.55
## 6 wire_gauge                   1     4           2.76
```

```
ggplot(eddy_probe_means, aes(factor_setting, average_factor)) +
  geom_point(aes(colour = factor_setting, shape = as.factor(value_setting)), position = position_dodge(
  geom_line(aes(group = factor_setting)) +
  labs(title = "DOE mean plot for Eddy current data", y = "sensitivity", x = "factor") +
  theme(legend.title = element_blank())
```



5.2.1.7 DOE interaction plot for eddy current data

```
eddy_probe_interaction <- lm(average_factor ~ factor_setting, data = eddy_probe_means)
summary(eddy_probe_interaction)
```

```
##
## Call:
## lm(formula = average_factor ~ factor_setting, data = eddy_probe_means)
##
## Residuals:
##       1       2       3       4       5       6
## -1.5512  1.5513  0.4337 -0.4338 -0.1062  0.1062
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     2.659e+00  9.320e-01   2.853    0.065 .
## factor_settingwinding_distance -8.802e-16  1.318e+00   0.000    1.000
## factor_settingwire_gauge       -2.220e-16  1.318e+00   0.000    1.000
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.318 on 3 degrees of freedom
## Multiple R-squared:  1.892e-31,  Adjusted R-squared:  -0.6667
## F-statistic: 2.838e-31 on 2 and 3 DF,  p-value: 1
```

Chapter 6

Process Monitoring

6.1 Packages used in this chapter

```r
library(magrittr) # used for %$% pipe
library(tidyverse)
library(ggplot2)
library(broom)
```

6.2 Case Stuidies

6.2.1 Lithography Process Example

Lithography process example

6.2.1.1 Background

> One of the assumptions in using classical Shewhart SPC charts is that the only source of variation is from part to part (or within subgroup variation). This is the case for most continuous processing situations. However, many of today's processing situations have different sources of variation. The semiconductor industry is one of the areas where the processing creates multiple sources of variation.
>
> In semiconductor processing, the basic experimental unit is a silicon wafer. Operations are performed on the wafer, but individual wafers can be grouped multiple ways. In the diffusion area, up to 150 wafers are processed in one time in a diffusion tube. In the etch area, single wafers are processed individually. In the lithography area, the light exposure is done on sub-areas of the wafer. There are many times during the production of a computer chip where the experimental unit varies and thus there are different sources of variation in this batch processing environment.
>
> The following is a case study of a lithography process. Five sites are measured on each wafer, three wafers are measured in a cassette (typically a grouping of 24 - 25 wafers) and thirty cassettes of wafers are used in the study. The width of a line is the measurement under study. There are two line width variables. The first is the original data and the second has been cleaned up somewhat. This case study uses the raw data. The entire data table is 450 rows long with six columns.

6.2.1.2 Data

```
litho <- read_table2("NIST data/monitor-6.6.1.1.dat",
    skip = 4, col_names = FALSE) %>%
  rename(cassette = X1, wafer = X2, site = X3, raw_linewidth = X4, run_number = X5, cleaned_linewidth =
```

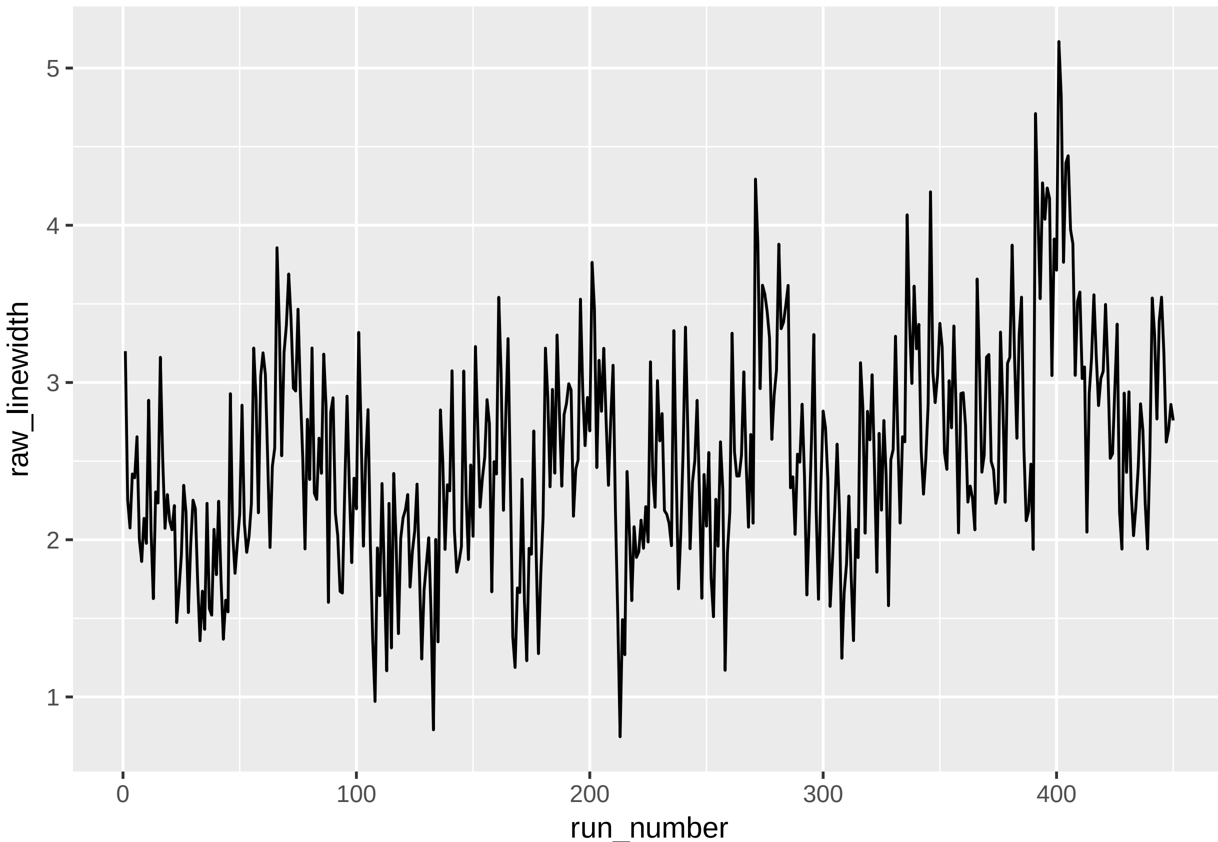
```
## Parsed with column specification:
## cols(
##   X1 = col_integer(),
##   X2 = col_integer(),
##   X3 = col_character(),
##   X4 = col_double(),
##   X5 = col_integer(),
##   X6 = col_double()
## )
```

```
litho
```

```
## # A tibble: 450 x 6
##    cassette wafer site  raw_linewidth run_number cleaned_linewidth
##       <int> <int> <chr>         <dbl>      <int>             <dbl>
##  1        1     1 Top            3.20          1              3.20
##  2        1     1 Lef            2.25          2              2.25
##  3        1     1 Cen            2.07          3              2.07
##  4        1     1 Rgt            2.42          4              2.41
##  5        1     1 Bot            2.39          5              2.38
##  6        1     2 Top            2.65          6              2.64
##  7        1     2 Lef            2.00          7              1.99
##  8        1     2 Cen            1.86          8              1.85
##  9        1     2 Rgt            2.14          9              2.12
## 10        1     2 Bot            1.98         10              1.96
## # ... with 440 more rows
```

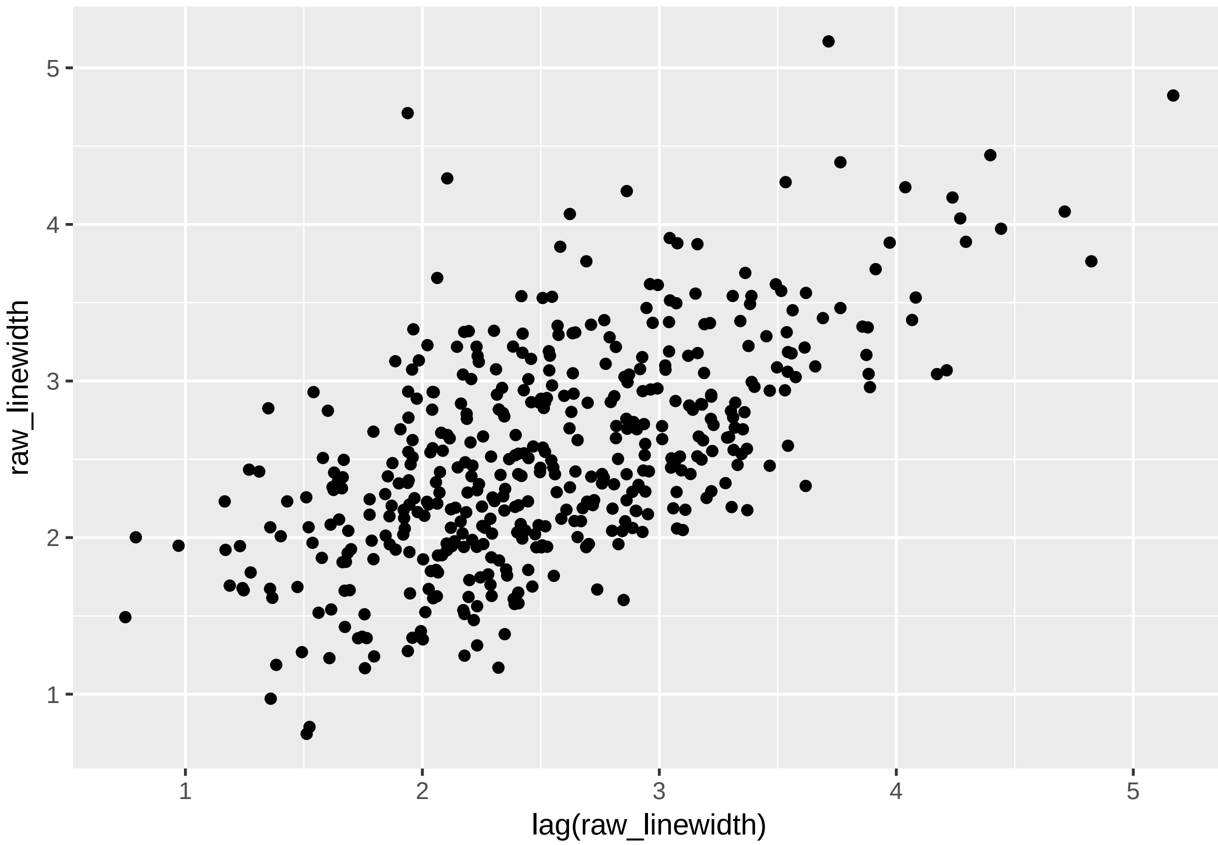
6.2.1.3 Generate some simple plots

```
ggplot(litho) +
  geom_line(aes(run_number, raw_linewidth))
```



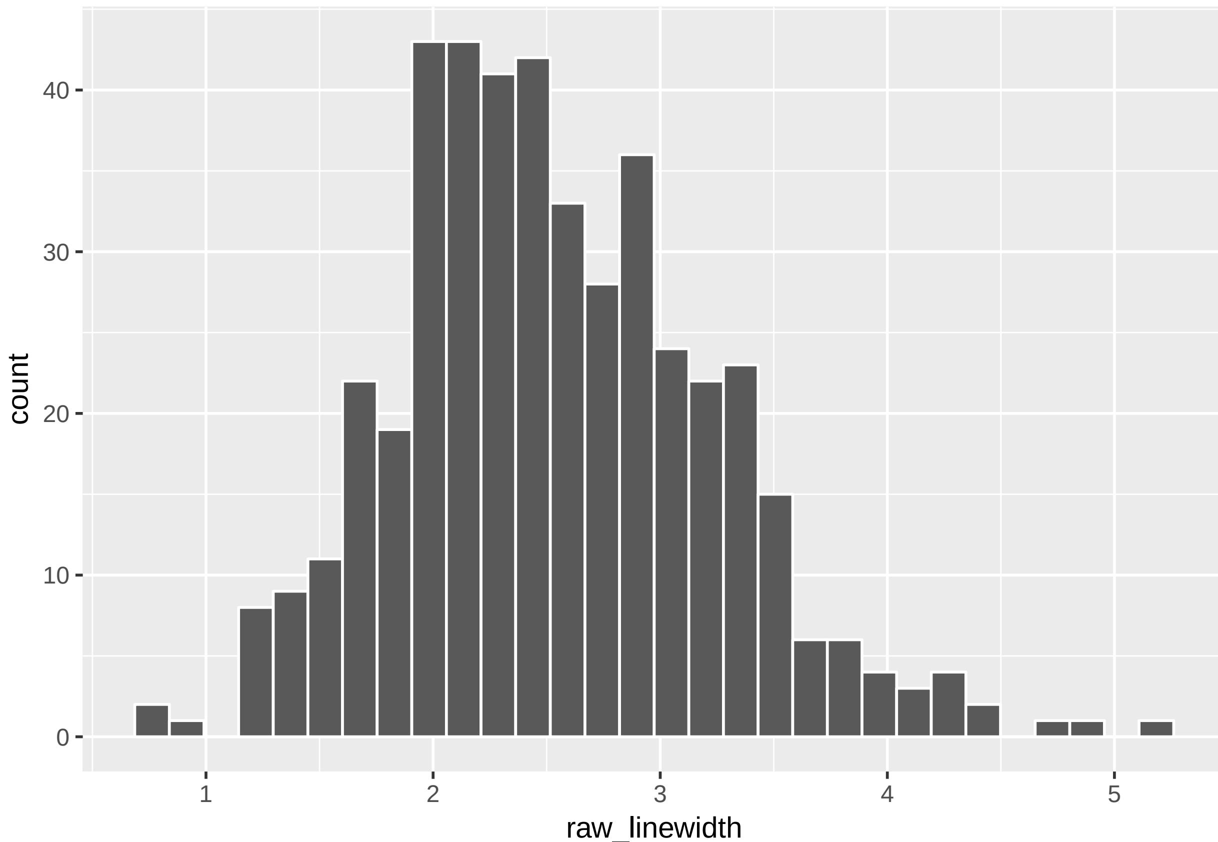

```
ggplot(litho) +
  geom_point(aes(lag(raw_linewidth), raw_linewidth))
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

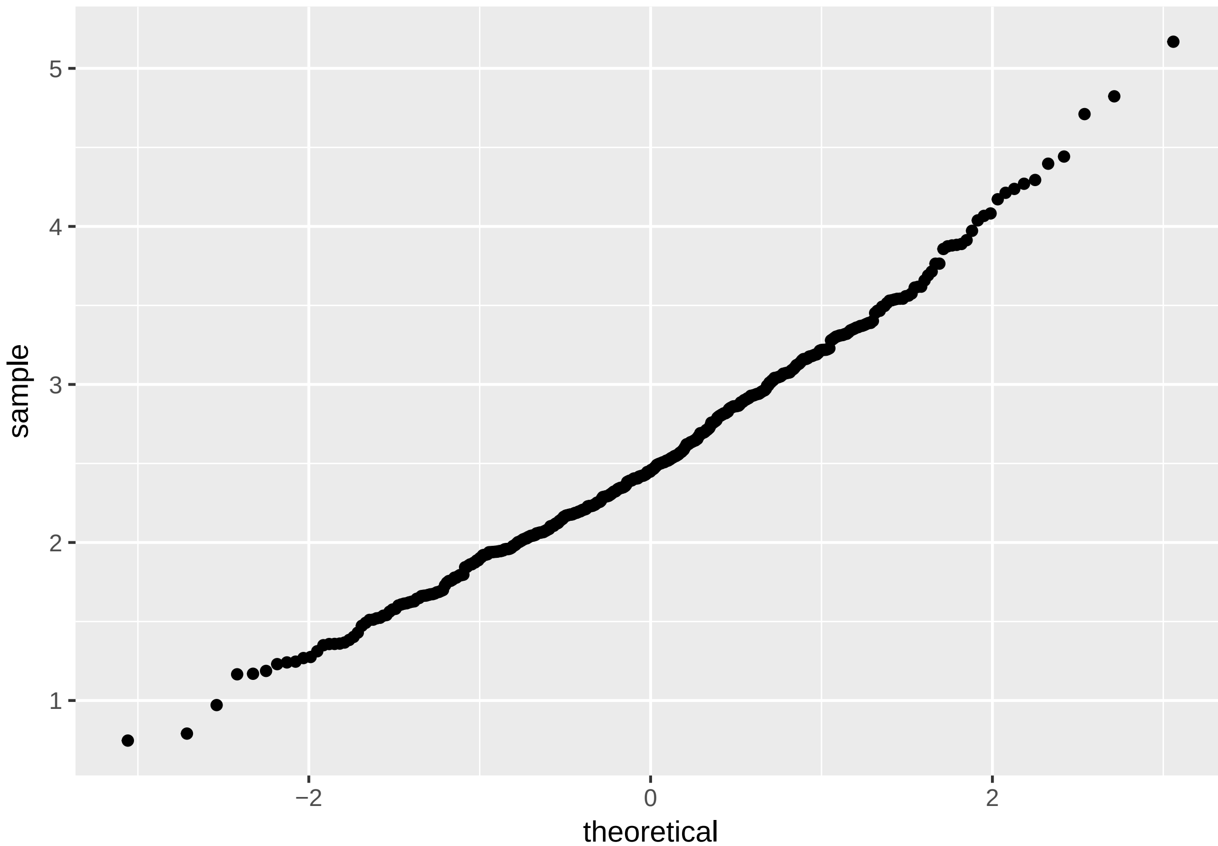



```
ggplot(litho) +
  geom_histogram(aes(raw_linewidth), colour = "white")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(litho) +
  geom_qq(aes(sample = raw_linewidth))
```



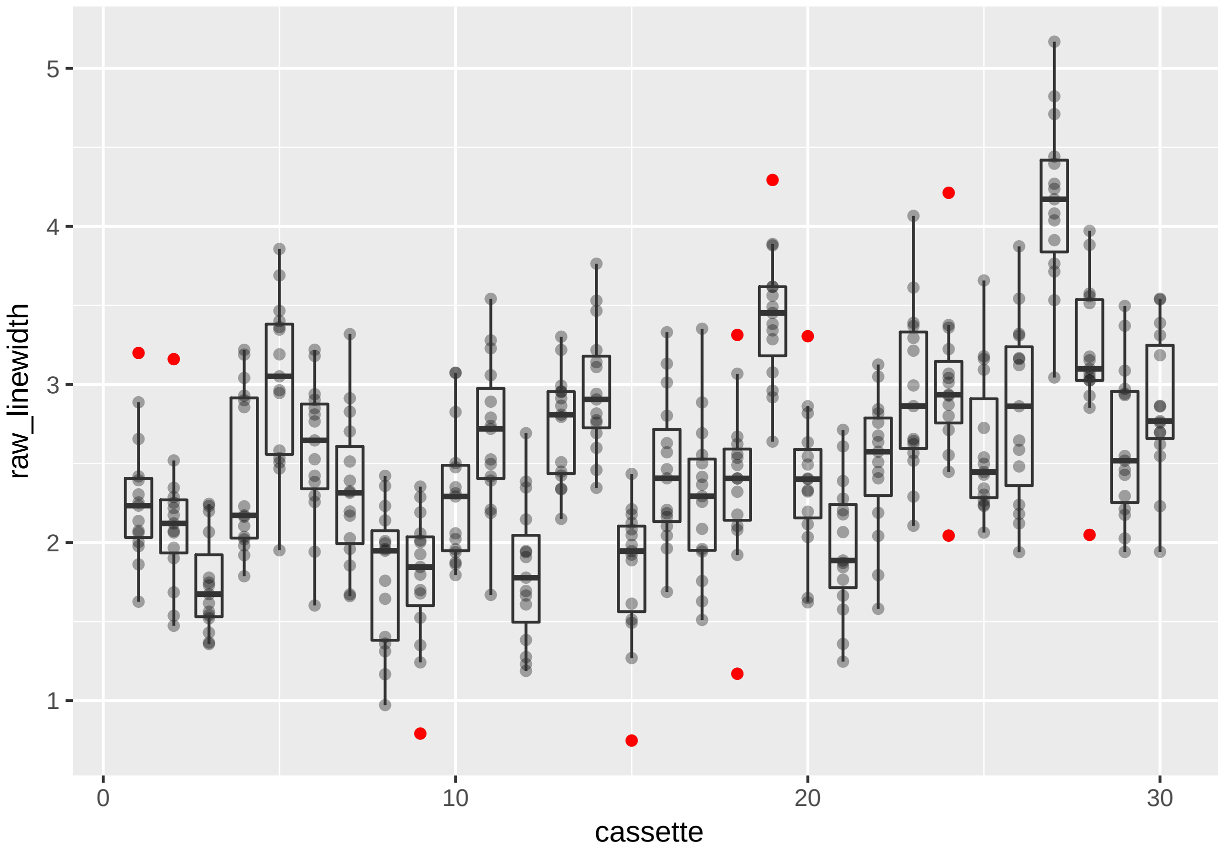
6.2.1.4 Summarise the raw linewidth and cleaned linesidth data

```
litho %>%
  dplyr::select(raw_linewidth, cleaned_linewidth) %>%
  summary()
```

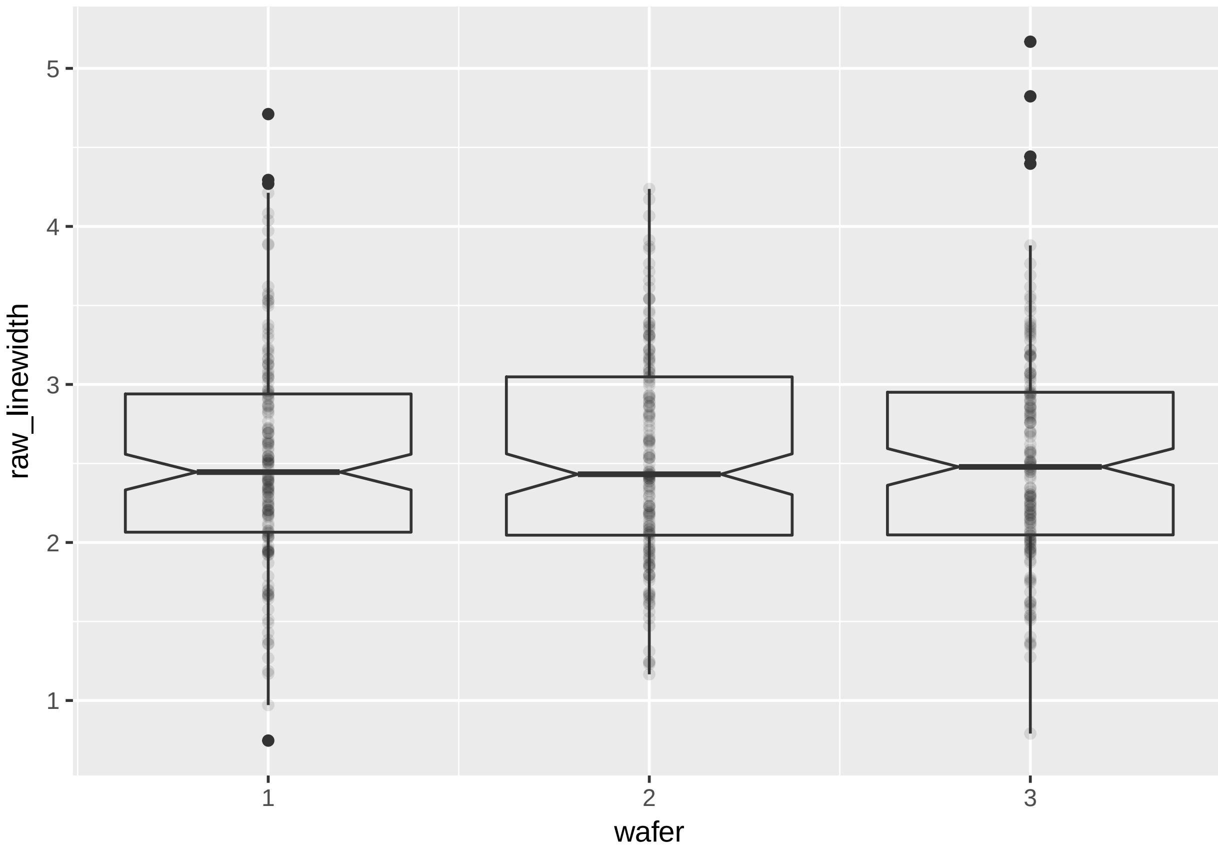
```
##  raw_linewidth    cleaned_linewidth
##  Min.   :0.7465   Min.   :0.3205
##  1st Qu.:2.0505   1st Qu.:1.6476
##  Median :2.4533   Median :2.0367
##  Mean   :2.5323   Mean   :2.0813
##  3rd Qu.:2.9697   3rd Qu.:2.4856
##  Max.   :5.1687   Max.   :4.3667
```

6.2.1.5 Plot the response against individual factors

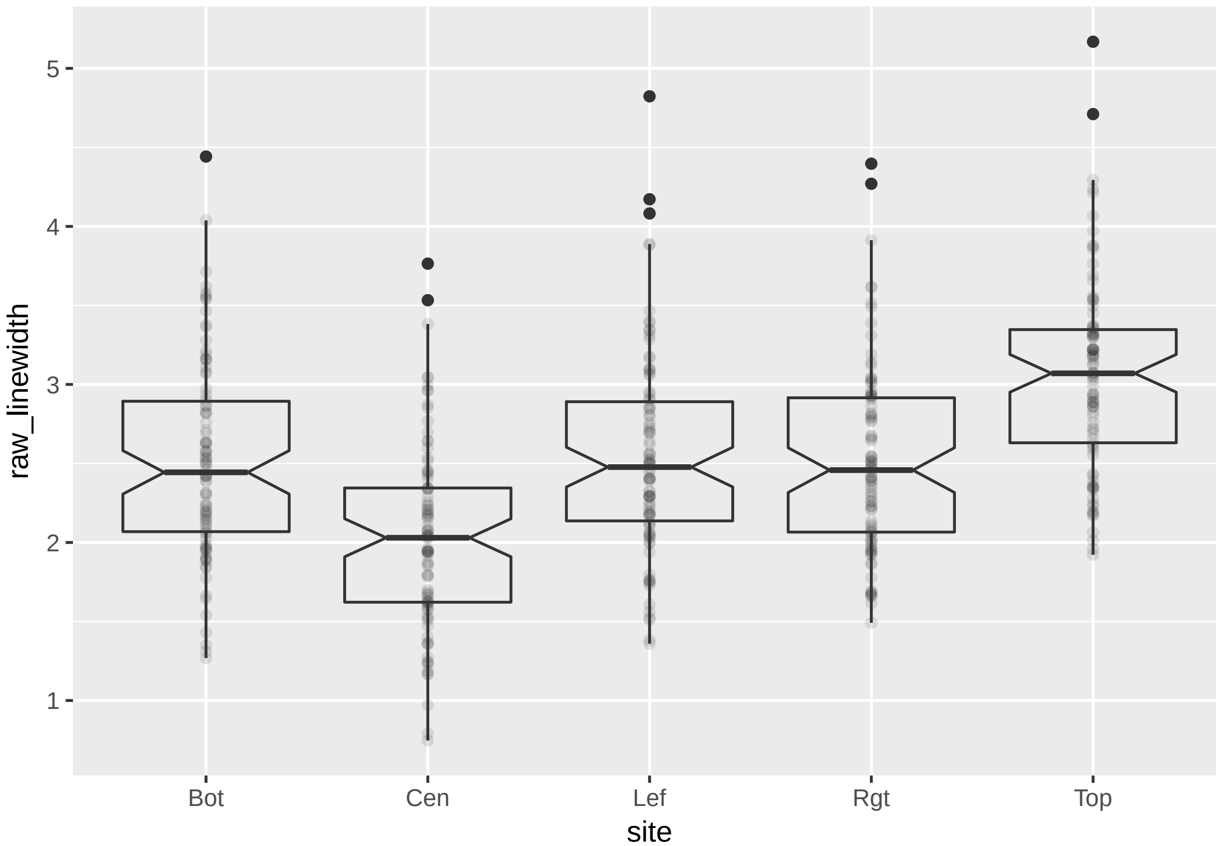
```
ggplot(litho) +
  geom_point(aes(cassette, raw_linewidth), alpha = 1/3) +
  geom_boxplot(aes(cassette, raw_linewidth, group = cassette), alpha = 0, outlier.alpha = 1, outlier.co
```




```
ggplot(litho) +
  geom_point(aes(wafer, raw_linewidth), alpha = 1/10) +
  geom_boxplot(aes(wafer, raw_linewidth, group = wafer), alpha = 0, notch = TRUE, outlier.alpha = 1)
```




```
ggplot(litho) +
  geom_point(aes(site, raw_linewidth), alpha = 1/10) +
  geom_boxplot(aes(site, raw_linewidth), alpha = 0, notch = TRUE, outlier.alpha = 1)
```



6.2.1.6 DOE plots

We need to gather the factors in to a single column

```
litho_DOE <- litho %>%
  gather(`cassette`, `wafer`, `site`, key = DOE_factors, value = "value")
litho_DOE
```

```
## # A tibble: 1,350 x 5
##    raw_linewidth run_number cleaned_linewidth DOE_factors value
##            <dbl>      <int>             <dbl> <chr>       <chr>
##  1          3.20          1              3.20 cassette    1
##  2          2.25          2              2.25 cassette    1
##  3          2.07          3              2.07 cassette    1
##  4          2.42          4              2.41 cassette    1
##  5          2.39          5              2.38 cassette    1
##  6          2.65          6              2.64 cassette    1
##  7          2.00          7              1.99 cassette    1
##  8          1.86          8              1.85 cassette    1
##  9          2.14          9              2.12 cassette    1
## 10          1.98         10              1.96 cassette    1
## # ... with 1,340 more rows
```

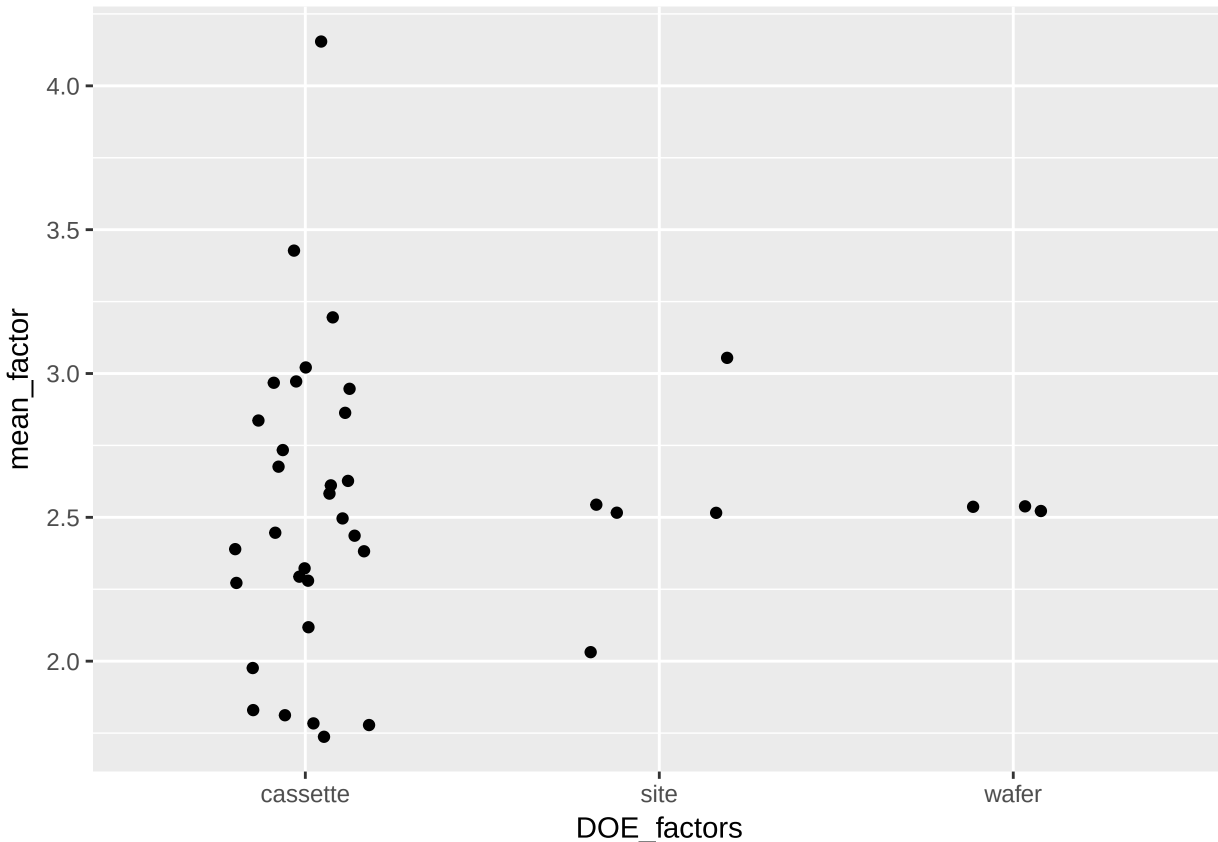
```
litho_group <- litho_DOE %>%
  group_by(DOE_factors, value)
litho_group
```

```
## # A tibble: 1,350 x 5
## # Groups:   DOE_factors, value [38]
##    raw_linewidth run_number cleaned_linewidth DOE_factors value
##            <dbl>      <int>             <dbl> <chr>       <chr>
##  1          3.20          1              3.20 cassette    1
##  2          2.25          2              2.25 cassette    1
##  3          2.07          3              2.07 cassette    1
##  4          2.42          4              2.41 cassette    1
##  5          2.39          5              2.38 cassette    1
##  6          2.65          6              2.64 cassette    1
##  7          2.00          7              1.99 cassette    1
##  8          1.86          8              1.85 cassette    1
##  9          2.14          9              2.12 cassette    1
## 10          1.98         10              1.96 cassette    1
## # ... with 1,340 more rows
```

```
litho_summary <- litho_group %>%
  summarise(mean_factor = mean(raw_linewidth), count = n())
litho_summary
```

```
## # A tibble: 38 x 4
## # Groups:   DOE_factors [?]
##    DOE_factors value mean_factor count
##    <chr>       <chr>       <dbl> <int>
##  1 cassette    1            2.27    15
##  2 cassette    10           2.29    15
##  3 cassette    11           2.68    15
##  4 cassette    12           1.81    15
##  5 cassette    13           2.73    15
##  6 cassette    14           2.97    15
##  7 cassette    15           1.83    15
##  8 cassette    16           2.45    15
##  9 cassette    17           2.28    15
## 10 cassette    18           2.39    15
## # ... with 28 more rows
```

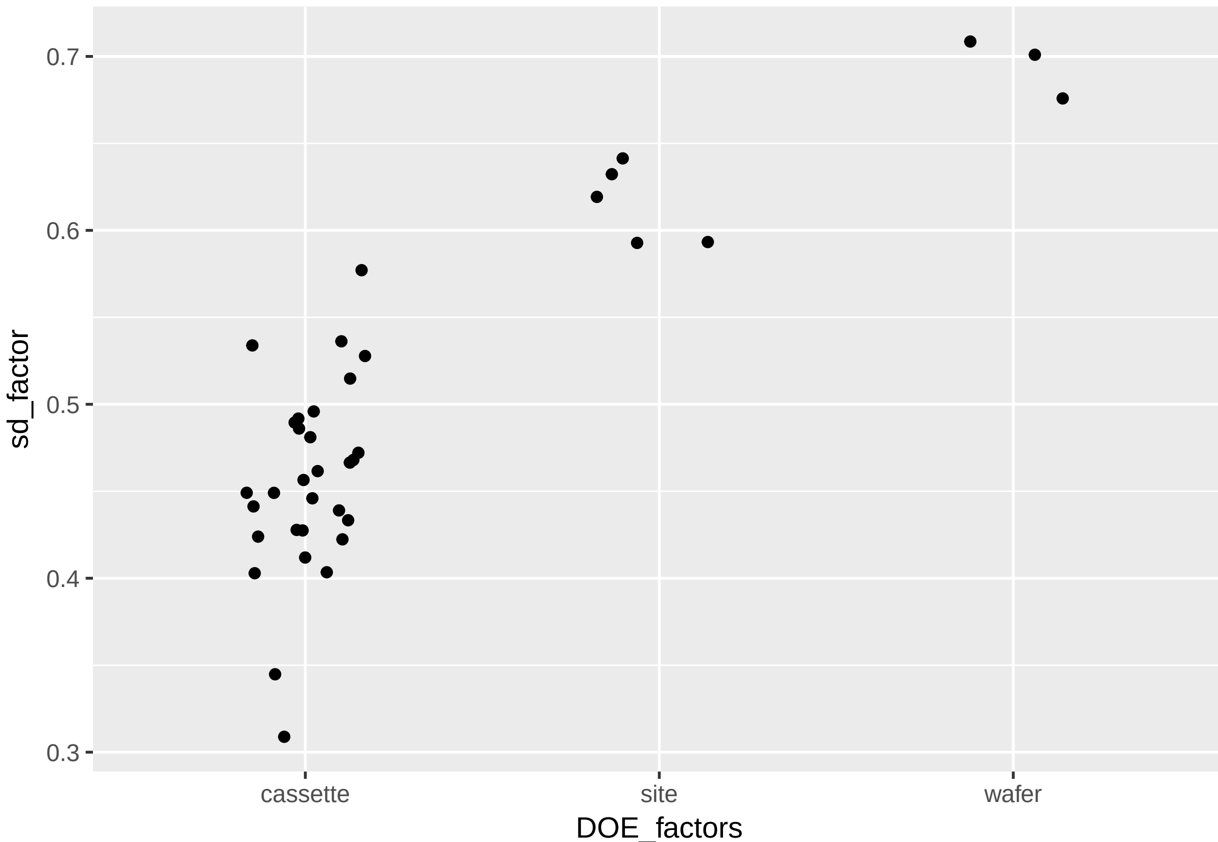
```
ggplot(litho_summary) +
  geom_jitter(aes(DOE_factors, mean_factor), width = 0.2)
```




```
litho_summary_sd <- litho_group %>%
  summarise(sd_factor = sd(raw_linewidth), count = n())
litho_summary_sd
```

```
## # A tibble: 38 x 4
## # Groups:   DOE_factors [?]
##    DOE_factors value sd_factor count
##    <chr>       <chr>     <dbl> <int>
##  1 cassette    1         0.403    15
##  2 cassette    10        0.428    15
##  3 cassette    11        0.486    15
##  4 cassette    12        0.449    15
##  5 cassette    13        0.345    15
##  6 cassette    14        0.403    15
##  7 cassette    15        0.433    15
##  8 cassette    16        0.466    15
##  9 cassette    17        0.492    15
## 10 cassette    18        0.496    15
## # ... with 28 more rows
```

```
ggplot(litho_summary_sd) +
  geom_jitter(aes(DOE_factors, sd_factor), width = 0.2)
```

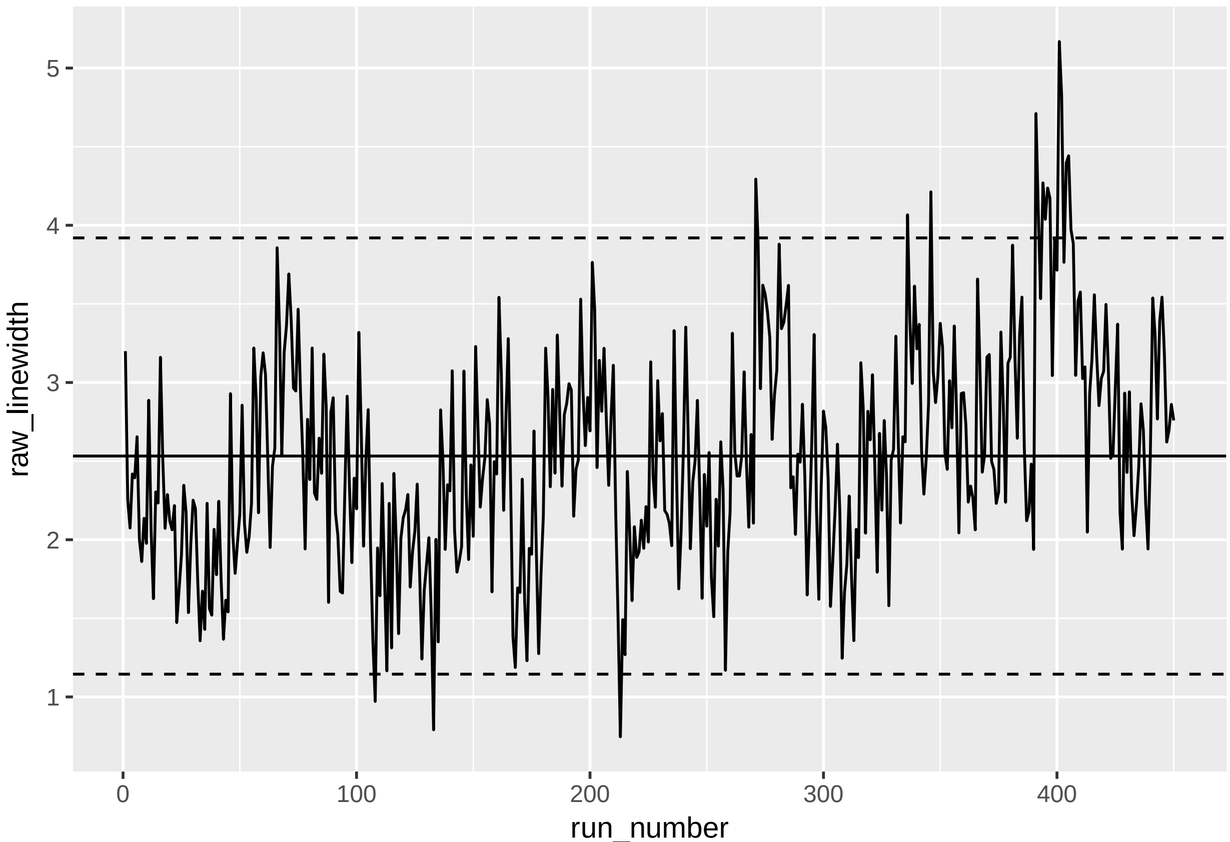


6.2.1.7 Subgroup analysis

6.2.1.7.1 Run chart

The chart below adds the mean and control limits based on the standard deviation of the data.

```
sd_lw <- litho %$%
  sd(raw_linewidth)

mean_lw <- litho %$%
  mean(raw_linewidth)

ggplot(litho) +
  geom_line(aes(run_number, raw_linewidth)) +
  geom_hline(yintercept = mean_lw + 2*sd_lw, linetype = "dashed") +
  geom_hline(yintercept = mean_lw - 2*sd_lw, linetype = "dashed") +
  geom_hline(yintercept = mean_lw)
```



6.2.1.7.2 Summarise by wafer

```
litho_wafer <- litho %>%
  group_by(cassette, wafer) %>%
  summarise(wafer_mean = mean(raw_linewidth), wafer_sd = sd(raw_linewidth)) %>%
  rowid_to_column(var = "wafer_number") %>%
  ungroup()

litho_wafer
```

```
## # A tibble: 90 x 5
##    wafer_number cassette wafer wafer_mean wafer_sd
##           <int>    <int> <int>      <dbl>    <dbl>
##  1            1        1     1       2.47    0.431
##  2            2        1     2       2.13    0.311
##  3            3        1     3       2.22    0.456
##  4            4        2     1       2.43    0.443
##  5            5        2     2       1.87    0.296
##  6            6        2     3       2.05    0.322
##  7            7        3     1       1.68    0.331
##  8            8        3     2       1.83    0.311
##  9            9        3     3       1.70    0.333
## 10           10        4     1       2.18    0.441
## # ... with 80 more rows
```

6.2.1.7.3 Wafer stats

```
sd_wafer <- litho_wafer %$%
  sd(wafer_mean)

rms_sd_wafer <- litho_wafer %>%
  dplyr::select(wafer_sd) %>%
  mutate(sd_squared = wafer_sd^2) %$%
  sqrt(mean(sd_squared))

mean_wafer_sd <- litho_wafer %$%
  mean(wafer_sd)

mean_wafer_sd
```

```
## [1] 0.407502
```

```
sd_wafer
```

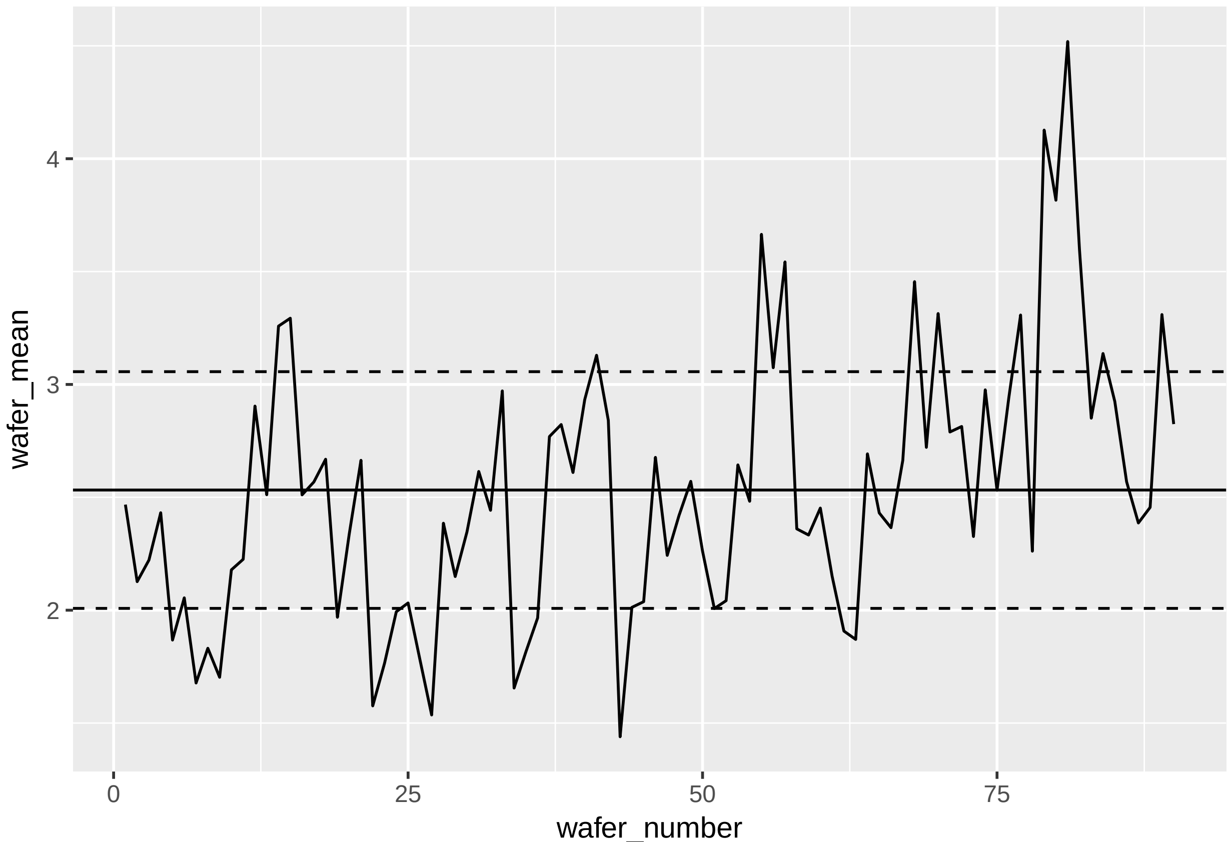
```
## [1] 0.5862159
```

```
rms_sd_wafer
```

```
## [1] 0.4189227
```

6.2.1.7.4 Wafer mean control chart

```
ggplot(litho_wafer) +
  geom_line(aes(wafer_number, wafer_mean)) +
  geom_hline(yintercept = mean_lw + 2*sd_wafer/sqrt(5), linetype = "dashed") +
  geom_hline(yintercept = mean_lw - 2*sd_wafer/sqrt(5), linetype = "dashed") +
  geom_hline(yintercept = mean_lw)
```

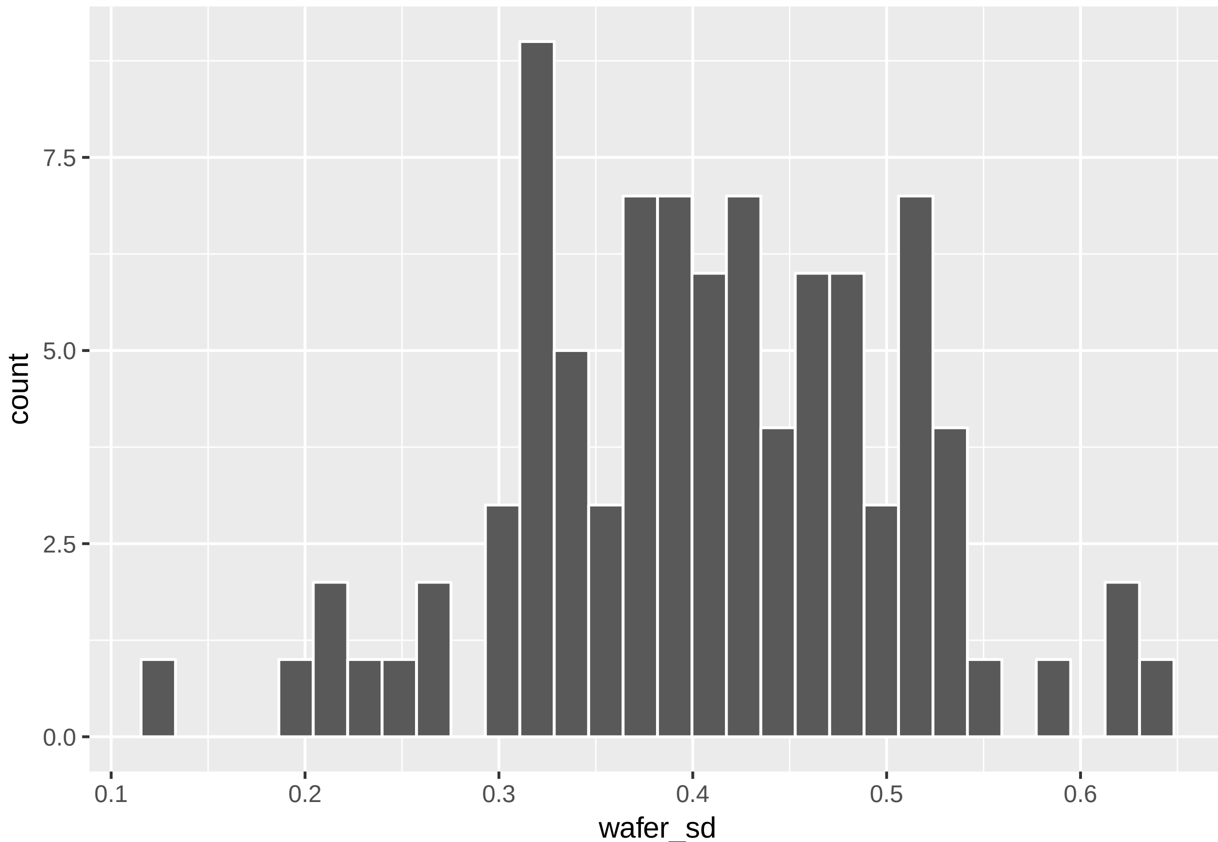


6.2.1.7.5 SD control chart by wafer

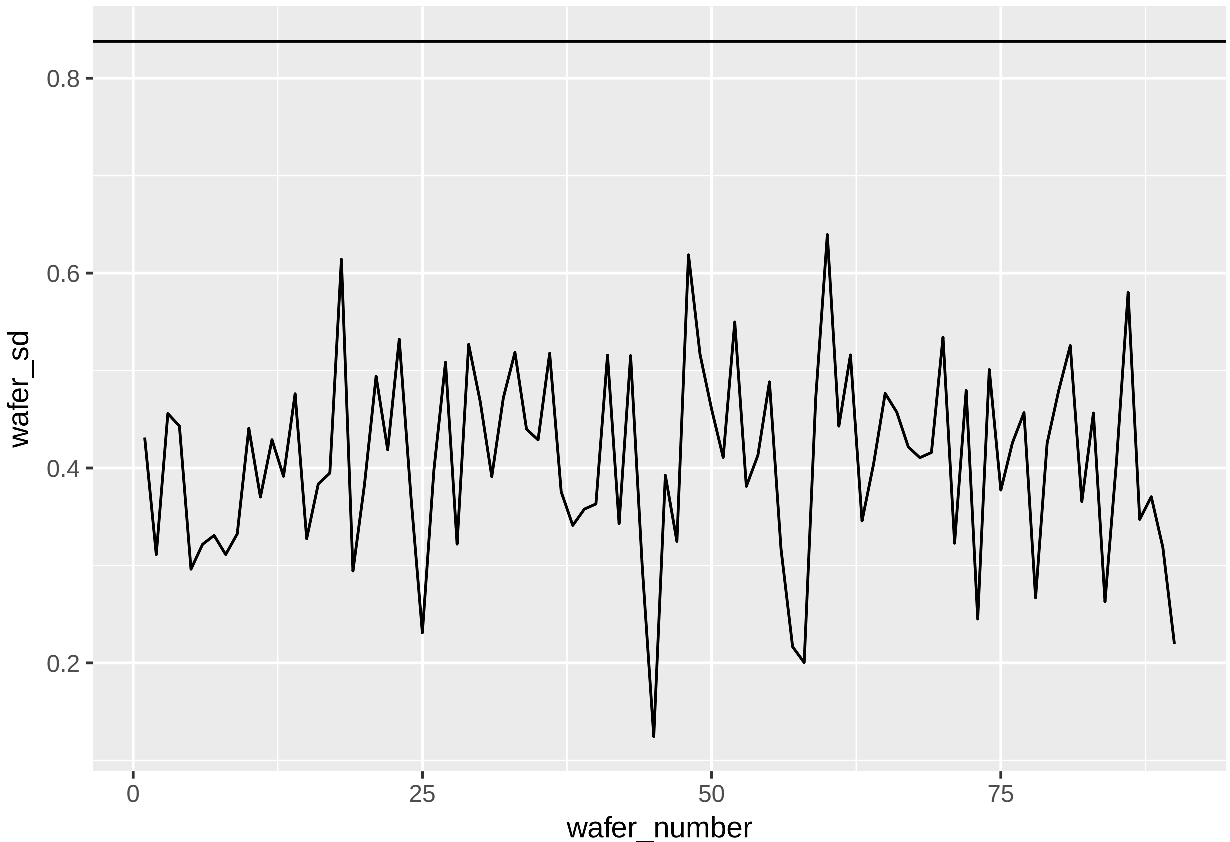
Using the methods from (2.2.3.1. Control chart for standard)[https://www.itl.nist.gov/div898/handbook/mpc/section2/mpc231.htm] we can construct an UCL for the standard deviations

```
ggplot(litho_wafer) +
 geom_histogram(aes(wafer_sd), colour = "white")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
ggplot(litho_wafer) +
  geom_line(aes(wafer_number, wafer_sd)) +
  geom_hline(yintercept = 2*rms_sd_wafer)
```



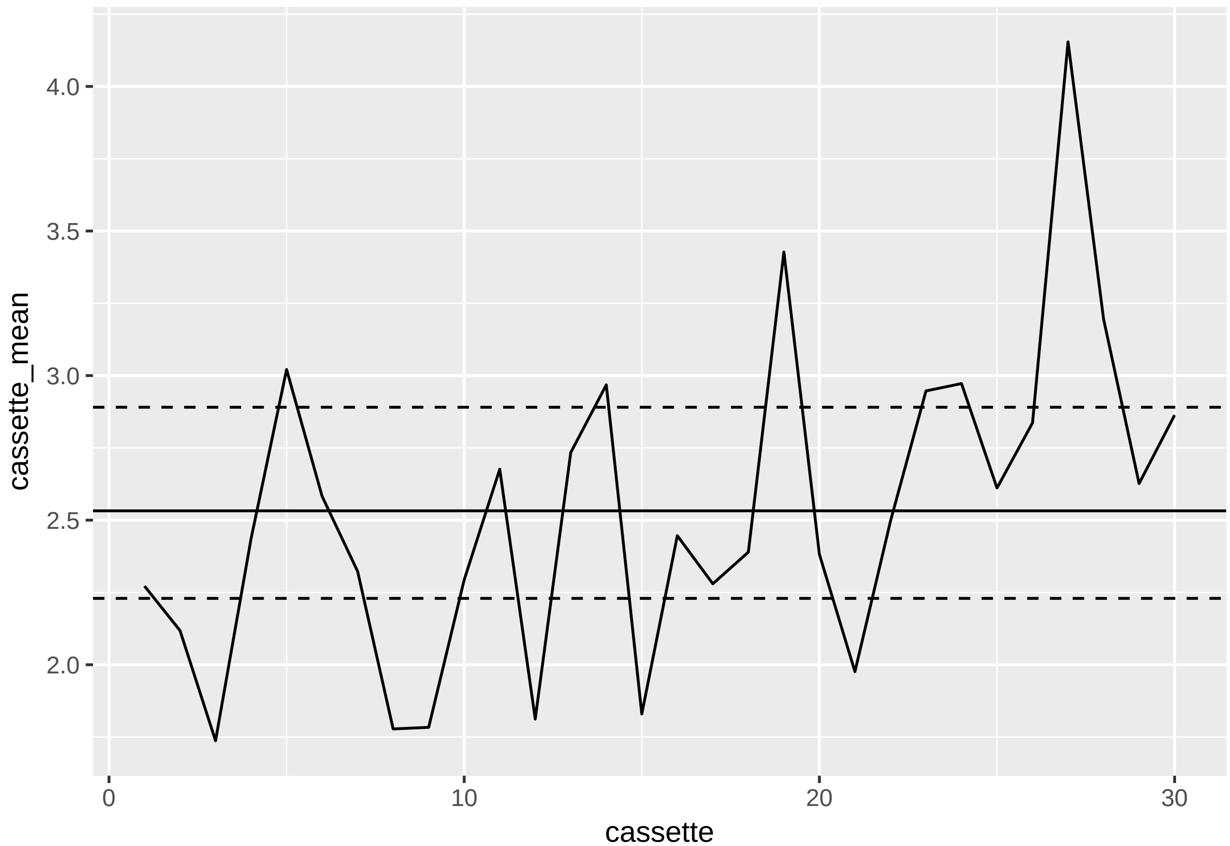
6.2.1.7.6 Summarise by cassette

```
litho_cassette <- litho %>%
  group_by(cassette) %>%
  summarise(cassette_mean = mean(raw_linewidth), cassette_sd = sd(raw_linewidth)) %>%
  ungroup()

litho_cassette
```

```
## # A tibble: 30 x 3
##    cassette cassette_mean cassette_sd
##       <int>         <dbl>       <dbl>
##  1        1          2.27       0.403
##  2        2          2.12       0.412
##  3        3          1.74       0.309
##  4        4          2.44       0.515
##  5        5          3.02       0.528
##  6        6          2.58       0.446
##  7        7          2.32       0.472
##  8        8          1.78       0.449
##  9        9          1.78       0.422
## 10       10          2.29       0.428
## # ... with 20 more rows
```

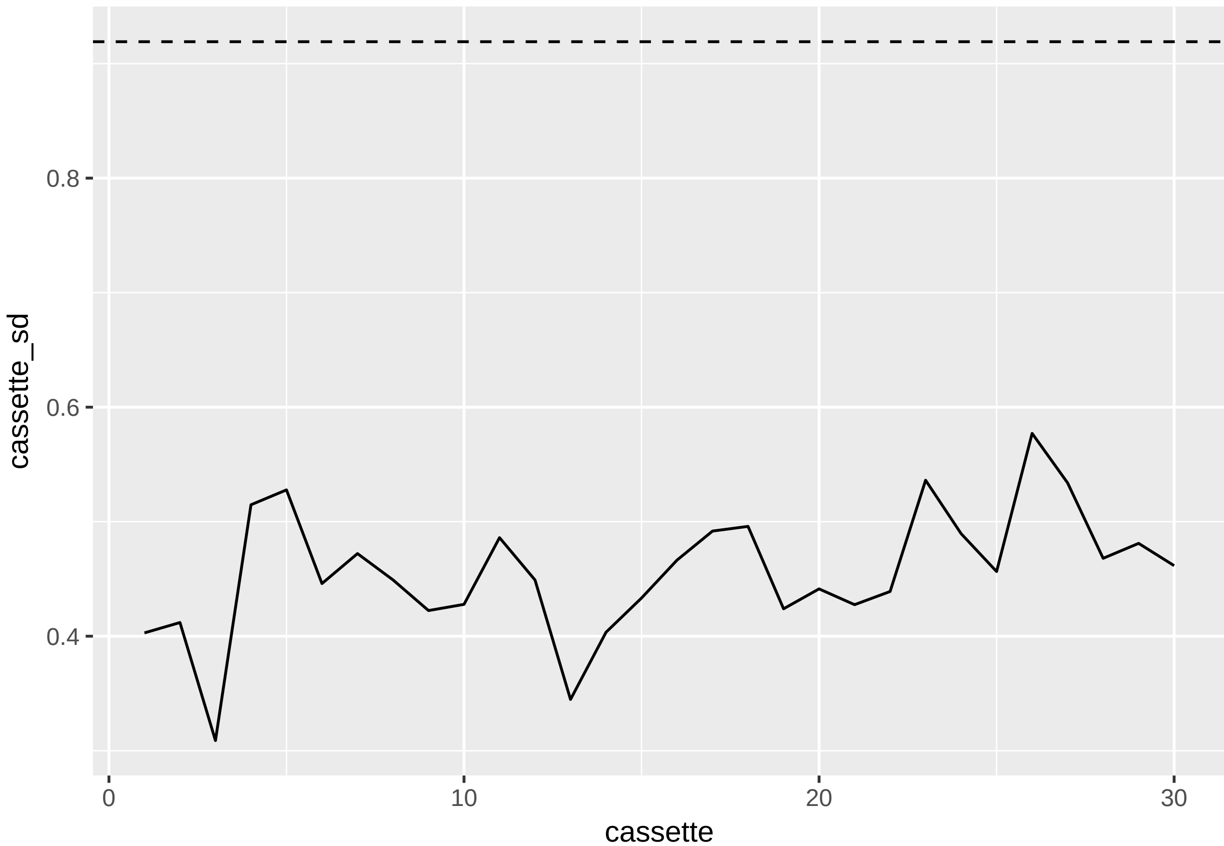
6.2.1.7.7 Mean control chart by cassette

```
ggplot(litho_cassette) +
  geom_line(aes(cassette, cassette_mean)) +
  geom_hline(yintercept = mean_lw + 2*sd_lw/sqrt(15), linetype = "dashed") +
  geom_hline(yintercept = mean_lw - 2*sd_wafer/sqrt(15), linetype = "dashed") +
  geom_hline(yintercept = mean_lw)
```



6.2.1.7.8 SD control chart by cassette

```
rms_sd_cassette <- litho_cassette %>%
  dplyr::select(cassette_sd) %>%
  mutate(sd_squared = cassette_sd^2) %$%
  sqrt(mean(sd_squared))


ggplot(litho_cassette) +
  geom_line(aes(cassette, cassette_sd)) +
  geom_hline(yintercept = 2*rms_sd_cassette, linetype = "dashed")
```



6.2.1.7.9 Variance compoent estimation

Attach the nessary libraries

```
library(lme4)
library(broom)
```

Fit the random effects model and print the variance compoents

```
random_effects_model <- lmer(raw_linewidth ~ 1|cassette/wafer, data = litho)
summary(random_effects_model)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: raw_linewidth ~ 1 | cassette/wafer
##    Data: litho
##
## REML criterion at convergence: 645.5
##
## Scaled residuals:
##      Min       1Q   Median       3Q      Max
## -2.45850 -0.62363 -0.03559  0.57612  2.52347
##
## Random effects:
##  Groups         Name        Variance Std.Dev.
##  wafer:cassette (Intercept) 0.04997  0.2235
##  cassette       (Intercept) 0.26452  0.5143
##  Residual                   0.17550  0.4189
## Number of obs: 450, groups:  wafer:cassette, 90; cassette, 30
```

```
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept)  2.53228    0.09881   25.63
```

```r
augment(random_effects_model)
```

```
##     raw_linewidth cassette wafer  .fitted       .resid   .fixed      .mu
## 1        3.199275        1     1 2.397405  0.801869573 2.532284 2.397405
## 2        2.253081        1     1 2.397405 -0.144324427 2.532284 2.397405
## 3        2.074308        1     1 2.397405 -0.323097427 2.532284 2.397405
## 4        2.418206        1     1 2.397405  0.020800573 2.532284 2.397405
## 5        2.393732        1     1 2.397405 -0.003673427 2.532284 2.397405
## 6        2.654947        1     2 2.196917  0.458029715 2.532284 2.196917
## 7        2.003234        1     2 2.196917 -0.193683285 2.532284 2.196917
## 8        1.861268        1     2 2.196917 -0.335649285 2.532284 2.196917
## 9        2.136102        1     2 2.196917 -0.060815285 2.532284 2.196917
## 10       1.976495        1     2 2.196917 -0.220422285 2.532284 2.196917
## 11       2.887053        1     3 2.253183  0.633869650 2.532284 2.253183
## 12       2.061239        1     3 2.253183 -0.191944350 2.532284 2.253183
## 13       1.625191        1     3 2.253183 -0.627992350 2.532284 2.253183
## 14       2.304313        1     3 2.253183  0.051129650 2.532284 2.253183
## 15       2.233187        1     3 2.253183 -0.019996350 2.532284 2.253183
## 16       3.160233        2     1 2.318913  0.841320221 2.532284 2.318913
## 17       2.518913        2     1 2.318913  0.200000221 2.532284 2.318913
## 18       2.072211        2     1 2.318913 -0.246701779 2.532284 2.318913
## 19       2.287210        2     1 2.318913 -0.031702779 2.532284 2.318913
## 20       2.120452        2     1 2.318913 -0.198460779 2.532284 2.318913
## 21       2.063058        2     2 1.987554  0.075503944 2.532284 1.987554
## 22       2.217220        2     2 1.987554  0.229665944 2.532284 1.987554
## 23       1.472945        2     2 1.987554 -0.514609056 2.532284 1.987554
## 24       1.684581        2     2 1.987554 -0.302973056 2.532284 1.987554
## 25       1.900688        2     2 1.987554 -0.086866056 2.532284 1.987554
## 26       2.346254        2     3 2.097438  0.248816193 2.532284 2.097438
## 27       2.172825        2     3 2.097438  0.075387193 2.532284 2.097438
## 28       1.536538        2     3 2.097438 -0.560899807 2.532284 2.097438
## 29       1.966630        2     3 2.097438 -0.130807807 2.532284 2.097438
## 30       2.251576        2     3 2.097438  0.154138193 2.532284 2.097438
## 31       2.198141        3     1 1.733829  0.464312376 2.532284 1.733829
## 32       1.728784        3     1 1.733829 -0.005044624 2.532284 1.733829
## 33       1.357348        3     1 1.733829 -0.376480624 2.532284 1.733829
## 34       1.673159        3     1 1.733829 -0.060669624 2.532284 1.733829
## 35       1.429586        3     1 1.733829 -0.304242624 2.532284 1.733829
## 36       2.231291        3     2 1.824294  0.406997082 2.532284 1.824294
## 37       1.561993        3     2 1.824294 -0.262300918 2.532284 1.824294
## 38       1.520104        3     2 1.824294 -0.304189918 2.532284 1.824294
## 39       2.066068        3     2 1.824294  0.241774082 2.532284 1.824294
## 40       1.777603        3     2 1.824294 -0.046690918 2.532284 1.824294
## 41       2.244736        3     3 1.748700  0.496036378 2.532284 1.748700
## 42       1.745877        3     3 1.748700 -0.002822622 2.532284 1.748700
## 43       1.366895        3     3 1.748700 -0.381804622 2.532284 1.748700
## 44       1.615229        3     3 1.748700 -0.133470622 2.532284 1.748700
## 45       1.540863        3     3 1.748700 -0.207836622 2.532284 1.748700
## 46       2.929037        4     1 2.288878  0.640159073 2.532284 2.288878
## 47       2.035900        4     1 2.288878 -0.252977927 2.532284 2.288878
```

```
## 48       1.786147        4     1 2.288878 -0.502730927 2.532284 2.288878
## 49       1.980323        4     1 2.288878 -0.308554927 2.532284 2.288878
## 50       2.162919        4     1 2.288878 -0.125958927 2.532284 2.288878
## 51       2.855798        4     2 2.316285  0.539512663 2.532284 2.316285
## 52       2.104193        4     2 2.316285 -0.212092337 2.532284 2.316285
## 53       1.919507        4     2 2.316285 -0.396778337 2.532284 2.316285
## 54       2.019415        4     2 2.316285 -0.296870337 2.532284 2.316285
## 55       2.228705        4     2 2.316285 -0.087580337 2.532284 2.316285
## 56       3.219292        4     3 2.714948  0.504343818 2.532284 2.714948
## 57       2.900430        4     3 2.714948  0.185481818 2.532284 2.714948
## 58       2.171262        4     3 2.714948 -0.543686182 2.532284 2.714948
## 59       3.041250        4     3 2.714948  0.326301818 2.532284 2.714948
## 60       3.188804        4     3 2.714948  0.473855818 2.532284 2.714948
## 61       3.051234        5     1 2.702226  0.349007940 2.532284 2.702226
## 62       2.506230        5     1 2.702226 -0.195996060 2.532284 2.702226
## 63       1.950486        5     1 2.702226 -0.751740060 2.532284 2.702226
## 64       2.467719        5     1 2.702226 -0.234507060 2.532284 2.702226
## 65       2.581881        5     1 2.702226 -0.120345060 2.532284 2.702226
## 66       3.857221        5     2 3.140901  0.716319781 2.532284 3.140901
## 67       3.347343        5     2 3.140901  0.206441781 2.532284 3.140901
## 68       2.533870        5     2 3.140901 -0.607031219 2.532284 3.140901
## 69       3.190375        5     2 3.140901  0.049473781 2.532284 3.140901
## 70       3.362746        5     2 3.140901  0.221844781 2.532284 3.140901
## 71       3.690306        5     3 3.161507  0.528799176 2.532284 3.161507
## 72       3.401584        5     3 3.161507  0.240077176 2.532284 3.161507
## 73       2.963117        5     3 3.161507 -0.198389824 2.532284 3.161507
## 74       2.945828        5     3 3.161507 -0.215678824 2.532284 3.161507
## 75       3.466115        5     3 3.161507  0.304608176 2.532284 3.161507
## 76       2.938241        6     1 2.538503  0.399738314 2.532284 2.538503
## 77       2.526568        6     1 2.538503 -0.011934686 2.532284 2.538503
## 78       1.941370        6     1 2.538503 -0.597132686 2.532284 2.538503
## 79       2.765849        6     1 2.538503  0.227346314 2.532284 2.538503
## 80       2.382781        6     1 2.538503 -0.155721686 2.532284 2.538503
## 81       3.219665        6     2 2.572006  0.647658691 2.532284 2.572006
## 82       2.296011        6     2 2.572006 -0.275995309 2.532284 2.572006
## 83       2.256196        6     2 2.572006 -0.315810309 2.532284 2.572006
## 84       2.645933        6     2 2.572006  0.073926691 2.532284 2.572006
## 85       2.422187        6     2 2.572006 -0.149819309 2.532284 2.572006
## 86       3.180348        6     3 2.631210  0.549138363 2.532284 2.631210
## 87       2.849264        6     3 2.631210  0.218054363 2.532284 2.631210
## 88       1.601288        6     3 2.631210 -1.029921637 2.532284 2.631210
## 89       2.810051        6     3 2.631210  0.178841363 2.532284 2.631210
## 90       2.902980        6     3 2.631210  0.271770363 2.532284 2.631210
## 91       2.169679        7     1 2.123160  0.046518853 2.532284 2.123160
## 92       2.026506        7     1 2.123160 -0.096654147 2.532284 2.123160
## 93       1.671804        7     1 2.123160 -0.451356147 2.532284 2.123160
## 94       1.660760        7     1 2.123160 -0.462400147 2.532284 2.123160
## 95       2.314734        7     1 2.123160  0.191573853 2.532284 2.123160
## 96       2.912838        7     2 2.338686  0.574152360 2.532284 2.338686
## 97       2.323665        7     2 2.338686 -0.015020640 2.532284 2.338686
## 98       1.854223        7     2 2.338686 -0.484462640 2.532284 2.338686
## 99       2.391240        7     2 2.338686  0.052554360 2.532284 2.338686
## 100      2.196071        7     2 2.338686 -0.142614640 2.532284 2.338686
## 101      3.318517        7     3 2.531614  0.786902777 2.532284 2.531614
```

```
## 102       2.702735          7       3 2.531614  0.171120777 2.532284 2.531614
## 103       1.959008          7       3 2.531614 -0.572606223 2.532284 2.531614
## 104       2.512517          7       3 2.531614 -0.019097223 2.532284 2.531614
## 105       2.827469          7       3 2.531614  0.295854777 2.532284 2.531614
## 106       1.958022          8       1 1.689584  0.268438271 2.532284 1.689584
## 107       1.360106          8       1 1.689584 -0.329477729 2.532284 1.689584
## 108       0.971193          8       1 1.689584 -0.718390729 2.532284 1.689584
## 109       1.947857          8       1 1.689584  0.258273271 2.532284 1.689584
## 110       1.643580          8       1 1.689584 -0.046003729 2.532284 1.689584
## 111       2.357633          8       2 1.800398  0.557234600 2.532284 1.800398
## 112       1.757725          8       2 1.800398 -0.042673400 2.532284 1.800398
## 113       1.165886          8       2 1.800398 -0.634512400 2.532284 1.800398
## 114       2.231143          8       2 1.800398  0.430744600 2.532284 1.800398
## 115       1.311626          8       2 1.800398 -0.488772400 2.532284 1.800398
## 116       2.421686          8       3 1.934560  0.487126019 2.532284 1.934560
## 117       1.993855          8       3 1.934560  0.059295019 2.532284 1.934560
## 118       1.402543          8       3 1.934560 -0.532016981 2.532284 1.934560
## 119       2.008543          8       3 1.934560  0.073983019 2.532284 1.934560
## 120       2.139370          8       3 1.934560  0.204810019 2.532284 1.934560
## 121       2.190676          9       1 1.959423  0.231253427 2.532284 1.959423
## 122       2.287483          9       1 1.959423  0.328060427 2.532284 1.959423
## 123       1.698943          9       1 1.959423 -0.260479573 2.532284 1.959423
## 124       1.925731          9       1 1.959423 -0.033691573 2.532284 1.959423
## 125       2.057440          9       1 1.959423  0.098017427 2.532284 1.959423
## 126       2.353597          9       2 1.812932  0.540665183 2.532284 1.812932
## 127       1.796236          9       2 1.812932 -0.016695817 2.532284 1.812932
## 128       1.241040          9       2 1.812932 -0.571891817 2.532284 1.812932
## 129       1.677429          9       2 1.812932 -0.135502817 2.532284 1.812932
## 130       1.845041          9       2 1.812932  0.032109183 2.532284 1.812932
## 131       2.012669          9       3 1.667946  0.344723358 2.532284 1.667946
## 132       1.523769          9       3 1.667946 -0.144176642 2.532284 1.667946
## 133       0.790789          9       3 1.667946 -0.877156642 2.532284 1.667946
## 134       2.001942          9       3 1.667946  0.333996358 2.532284 1.667946
## 135       1.350051          9       3 1.667946 -0.317894642 2.532284 1.667946
## 136       2.825749         10       1 2.357094  0.468654771 2.532284 2.357094
## 137       2.502445         10       1 2.357094  0.145350771 2.532284 2.357094
## 138       1.938239         10       1 2.357094 -0.418855229 2.532284 2.357094
## 139       2.349497         10       1 2.357094 -0.007597229 2.532284 2.357094
## 140       2.310817         10       1 2.357094 -0.046277229 2.532284 2.357094
## 141       3.074576         10       2 2.218263  0.856312640 2.532284 2.218263
## 142       2.057821         10       2 2.218263 -0.160442360 2.532284 2.218263
## 143       1.793617         10       2 2.218263 -0.424646360 2.532284 2.218263
## 144       1.862251         10       2 2.218263 -0.356012360 2.532284 2.218263
## 145       1.956753         10       2 2.218263 -0.261510360 2.532284 2.218263
## 146       3.072840         10       3 2.334552  0.738288287 2.532284 2.334552
## 147       2.291035         10       3 2.334552 -0.043516713 2.532284 2.334552
## 148       1.873878         10       3 2.334552 -0.460673713 2.532284 2.334552
## 149       2.475640         10       3 2.334552  0.141088287 2.532284 2.334552
## 150       2.021472         10       3 2.334552 -0.313079713 2.532284 2.334552
## 151       3.228835         11       1 2.634201  0.594634324 2.532284 2.634201
## 152       2.719495         11       1 2.634201  0.085294324 2.532284 2.634201
## 153       2.207198         11       1 2.634201 -0.427002676 2.532284 2.634201
## 154       2.391608         11       1 2.634201 -0.242592676 2.532284 2.634201
## 155       2.525587         11       1 2.634201 -0.108613676 2.532284 2.634201
```

```
## 156       2.891103        11      2 2.533058  0.358044656 2.532284 2.533058
## 157       2.738007        11      2 2.533058  0.204948656 2.532284 2.533058
## 158       1.668337        11      2 2.533058 -0.864721344 2.532284 2.533058
## 159       2.496426        11      2 2.533058 -0.036632344 2.532284 2.533058
## 160       2.417926        11      2 2.533058 -0.115132344 2.532284 2.533058
## 161       3.541799        11      3 2.843834  0.697964510 2.532284 2.843834
## 162       3.058768        11      3 2.843834  0.214933510 2.532284 2.843834
## 163       2.187061        11      3 2.843834 -0.656773490 2.532284 2.843834
## 164       2.790261        11      3 2.843834 -0.053573490 2.532284 2.843834
## 165       3.279238        11      3 2.843834  0.435403510 2.532284 2.843834
## 166       2.347662        12      1 1.748642  0.599020267 2.532284 1.748642
## 167       1.383336        12      1 1.748642 -0.365305733 2.532284 1.748642
## 168       1.187168        12      1 1.748642 -0.561473733 2.532284 1.748642
## 169       1.693292        12      1 1.748642 -0.055349733 2.532284 1.748642
## 170       1.664072        12      1 1.748642 -0.084569733 2.532284 1.748642
## 171       2.385320        12      2 1.842730  0.542589503 2.532284 1.842730
## 172       1.607784        12      2 1.842730 -0.234946497 2.532284 1.842730
## 173       1.230307        12      2 1.842730 -0.612423497 2.532284 1.842730
## 174       1.945423        12      2 1.842730  0.102692503 2.532284 1.842730
## 175       1.907580        12      2 1.842730  0.064849503 2.532284 1.842730
## 176       2.691576        12      3 1.931171  0.760405467 2.532284 1.931171
## 177       1.938755        12      3 1.931171  0.007584467 2.532284 1.931171
## 178       1.275409        12      3 1.931171 -0.655761533 2.532284 1.931171
## 179       1.777315        12      3 1.931171 -0.153855533 2.532284 1.931171
## 180       2.146161        12      3 1.931171  0.214990467 2.532284 1.931171
## 181       3.218655        13      1 2.746640  0.472015127 2.532284 2.746640
## 182       2.912180        13      1 2.746640  0.165540127 2.532284 2.746640
## 183       2.336436        13      1 2.746640 -0.410203873 2.532284 2.746640
## 184       2.956036        13      1 2.746640  0.209396127 2.532284 2.746640
## 185       2.423235        13      1 2.746640 -0.323404873 2.532284 2.746640
## 186       3.302224        13      2 2.777867  0.524357171 2.532284 2.777867
## 187       2.808816        13      2 2.777867  0.030949171 2.532284 2.777867
## 188       2.340386        13      2 2.777867 -0.437480829 2.532284 2.777867
## 189       2.795120        13      2 2.777867  0.017253171 2.532284 2.777867
## 190       2.865800        13      2 2.777867  0.087933171 2.532284 2.777867
## 191       2.992217        13      3 2.652991  0.339226160 2.532284 2.652991
## 192       2.952106        13      3 2.652991  0.299115160 2.532284 2.652991
## 193       2.149299        13      3 2.652991 -0.503691840 2.532284 2.652991
## 194       2.448046        13      3 2.652991 -0.204944840 2.532284 2.652991
## 195       2.507733        13      3 2.652991 -0.145257840 2.532284 2.652991
## 196       3.530112        14      1 2.930121  0.599990877 2.532284 2.930121
## 197       2.940489        14      1 2.930121  0.010367877 2.532284 2.930121
## 198       2.598357        14      1 2.930121 -0.331764123 2.532284 2.930121
## 199       2.905165        14      1 2.930121 -0.024956123 2.532284 2.930121
## 200       2.692078        14      1 2.930121 -0.238043123 2.532284 2.930121
## 201       3.764270        14      2 3.045290  0.718980355 2.532284 3.045290
## 202       3.465960        14      2 3.045290  0.420670355 2.532284 3.045290
## 203       2.458628        14      2 3.045290 -0.586661645 2.532284 3.045290
## 204       3.141132        14      2 3.045290  0.095842355 2.532284 3.045290
## 205       2.816526        14      2 3.045290 -0.228763645 2.532284 3.045290
## 206       3.217614        14      3 2.875946  0.341667894 2.532284 2.875946
## 207       2.758171        14      3 2.875946 -0.117775106 2.532284 2.875946
## 208       2.345921        14      3 2.875946 -0.530025106 2.532284 2.875946
## 209       2.773653        14      3 2.875946 -0.102293106 2.532284 2.875946
```

```
## 210       3.109704       14      3 2.875946  0.233757894 2.532284 2.875946
## 211       2.177593       15      1 1.628454  0.549138745 2.532284 1.628454
## 212       1.511781       15      1 1.628454 -0.116673255 2.532284 1.628454
## 213       0.746546       15      1 1.628454 -0.881908255 2.532284 1.628454
## 214       1.491730       15      1 1.628454 -0.136724255 2.532284 1.628454
## 215       1.268580       15      1 1.628454 -0.359874255 2.532284 1.628454
## 216       2.433994       15      2 1.965194  0.468799916 2.532284 1.965194
## 217       2.045667       15      2 1.965194  0.080472916 2.532284 1.965194
## 218       1.612699       15      2 1.965194 -0.352495084 2.532284 1.965194
## 219       2.082860       15      2 1.965194  0.117665916 2.532284 1.965194
## 220       1.887341       15      2 1.965194 -0.077853084 2.532284 1.965194
## 221       1.923003       15      3 1.979982 -0.056978905 2.532284 1.979982
## 222       2.124461       15      3 1.979982  0.144479095 2.532284 1.979982
## 223       1.945048       15      3 1.979982 -0.034933905 2.532284 1.979982
## 224       2.210698       15      3 1.979982  0.230716095 2.532284 1.979982
## 225       1.985225       15      3 1.979982  0.005243095 2.532284 1.979982
## 226       3.131536       16      1 2.585214  0.546321560 2.532284 2.585214
## 227       2.405975       16      1 2.585214 -0.179239440 2.532284 2.585214
## 228       2.206320       16      1 2.585214 -0.378894440 2.532284 2.585214
## 229       3.012211       16      1 2.585214  0.426996560 2.532284 2.585214
## 230       2.628723       16      1 2.585214  0.043508560 2.532284 2.585214
## 231       2.802486       16      2 2.330170  0.472315945 2.532284 2.330170
## 232       2.185010       16      2 2.330170 -0.145160055 2.532284 2.330170
## 233       2.161802       16      2 2.330170 -0.168368055 2.532284 2.330170
## 234       2.102560       16      2 2.330170 -0.227610055 2.532284 2.330170
## 235       1.961968       16      2 2.330170 -0.368202055 2.532284 2.330170
## 236       3.330183       16      3 2.433764  0.896419252 2.532284 2.433764
## 237       2.464046       16      3 2.433764  0.030282252 2.532284 2.433764
## 238       1.687408       16      3 2.433764 -0.746355748 2.532284 2.433764
## 239       2.043322       16      3 2.433764 -0.390441748 2.532284 2.433764
## 240       2.570657       16      3 2.433764  0.136893252 2.532284 2.433764
## 241       3.352633       17      1 2.460815  0.891817709 2.532284 2.460815
## 242       2.691645       17      1 2.460815  0.230829709 2.532284 2.460815
## 243       1.942410       17      1 2.460815 -0.518405291 2.532284 2.460815
## 244       2.366055       17      1 2.460815 -0.094760291 2.532284 2.460815
## 245       2.500987       17      1 2.460815  0.040171709 2.532284 2.460815
## 246       2.886284       17      2 2.279169  0.607114639 2.532284 2.279169
## 247       2.292503       17      2 2.279169  0.013333639 2.532284 2.279169
## 248       1.627562       17      2 2.279169 -0.651607361 2.532284 2.279169
## 249       2.415076       17      2 2.279169  0.135906639 2.532284 2.279169
## 250       2.086134       17      2 2.279169 -0.193035361 2.532284 2.279169
## 251       2.554848       17      3 2.129874  0.424974257 2.532284 2.129874
## 252       1.755843       17      3 2.129874 -0.374030743 2.532284 2.129874
## 253       1.510124       17      3 2.129874 -0.619749743 2.532284 2.129874
## 254       2.257347       17      3 2.129874  0.127473257 2.532284 2.129874
## 255       1.958592       17      3 2.129874 -0.171281743 2.532284 2.129874
## 256       2.622733       18      1 2.191171  0.431561594 2.532284 2.191171
## 257       2.321079       18      1 2.191171  0.129907594 2.532284 2.191171
## 258       1.169269       18      1 2.191171 -1.021902406 2.532284 2.191171
## 259       1.921457       18      1 2.191171 -0.269714406 2.532284 2.191171
## 260       2.176377       18      1 2.191171 -0.014794406 2.532284 2.191171
## 261       3.313367       18      2 2.544520  0.768846869 2.532284 2.544520
## 262       2.559725       18      2 2.544520  0.015204869 2.532284 2.544520
## 263       2.404662       18      2 2.544520 -0.139858131 2.532284 2.544520
```

```
## 264      2.405249        18      2 2.544520 -0.139271131 2.532284 2.544520
## 265      2.535618        18      2 2.544520 -0.008902131 2.532284 2.544520
## 266      3.067851        18      3 2.449793  0.618058143 2.532284 2.449793
## 267      2.490359        18      3 2.449793  0.040566143 2.532284 2.449793
## 268      2.079477        18      3 2.449793 -0.370315857 2.532284 2.449793
## 269      2.669512        18      3 2.449793  0.219719143 2.532284 2.449793
## 270      2.105103        18      3 2.449793 -0.344689857 2.532284 2.449793
## 271      4.293889        19      1 3.531144  0.762745066 2.532284 3.531144
## 272      3.888826        19      1 3.531144  0.357682066 2.532284 3.531144
## 273      2.960655        19      1 3.531144 -0.570488934 2.532284 3.531144
## 274      3.618864        19      1 3.531144  0.087720066 2.532284 3.531144
## 275      3.562480        19      1 3.531144  0.031336066 2.532284 3.531144
## 276      3.451872        19      2 3.184155  0.267717185 2.532284 3.184155
## 277      3.285934        19      2 3.184155  0.101779185 2.532284 3.184155
## 278      2.638294        19      2 3.184155 -0.545860815 2.532284 3.184155
## 279      2.918810        19      2 3.184155 -0.265344815 2.532284 3.184155
## 280      3.076231        19      2 3.184155 -0.107923815 2.532284 3.184155
## 281      3.879683        19      3 3.459376  0.420306531 2.532284 3.459376
## 282      3.342026        19      3 3.459376 -0.117350469 2.532284 3.459376
## 283      3.382833        19      3 3.459376 -0.076543469 2.532284 3.459376
## 284      3.491666        19      3 3.459376  0.032289531 2.532284 3.459376
## 285      3.617621        19      3 3.459376  0.158244531 2.532284 3.459376
## 286      2.329987        20      1 2.375244 -0.045257340 2.532284 2.375244
## 287      2.400277        20      1 2.375244  0.025032660 2.532284 2.375244
## 288      2.033941        20      1 2.375244 -0.341303340 2.532284 2.375244
## 289      2.544367        20      1 2.375244  0.169122660 2.532284 2.375244
## 290      2.493079        20      1 2.375244  0.117834660 2.532284 2.375244
## 291      2.862084        20      2 2.359172  0.502911902 2.532284 2.359172
## 292      2.404703        20      2 2.359172  0.045530902 2.532284 2.359172
## 293      1.648662        20      2 2.359172 -0.710510098 2.532284 2.359172
## 294      2.115465        20      2 2.359172 -0.243707098 2.532284 2.359172
## 295      2.633930        20      2 2.359172  0.274757902 2.532284 2.359172
## 296      3.305211        20      3 2.429361  0.875850030 2.532284 2.429361
## 297      2.194991        20      3 2.429361 -0.234369970 2.532284 2.429361
## 298      1.620963        20      3 2.429361 -0.808397970 2.532284 2.429361
## 299      2.322678        20      3 2.429361 -0.106682970 2.532284 2.429361
## 300      2.818449        20      3 2.429361  0.389088030 2.532284 2.429361
## 301      2.712915        21      1 2.100673  0.612241684 2.532284 2.100673
## 302      2.389121        21      1 2.100673  0.288447684 2.532284 2.100673
## 303      1.575833        21      1 2.100673 -0.524840316 2.532284 2.100673
## 304      1.870484        21      1 2.100673 -0.230189316 2.532284 2.100673
## 305      2.203262        21      1 2.100673  0.102588684 2.532284 2.100673
## 306      2.607972        21      2 1.958099  0.649873442 2.532284 1.958099
## 307      2.177747        21      2 1.958099  0.219648442 2.532284 1.958099
## 308      1.246016        21      2 1.958099 -0.712082558 2.532284 1.958099
## 309      1.663096        21      2 1.958099 -0.295002558 2.532284 1.958099
## 310      1.843187        21      2 1.958099 -0.114911558 2.532284 1.958099
## 311      2.277813        21      3 1.936304  0.341509308 2.532284 1.936304
## 312      1.764940        21      3 1.936304 -0.171363692 2.532284 1.936304
## 313      1.358137        21      3 1.936304 -0.578166692 2.532284 1.936304
## 314      2.065713        21      3 1.936304  0.129409308 2.532284 1.936304
## 315      1.885897        21      3 1.936304 -0.050406692 2.532284 1.936304
## 316      3.126184        22      1 2.613089  0.513095276 2.532284 2.613089
## 317      2.843505        22      1 2.613089  0.230416276 2.532284 2.613089
```

```
## 318        2.041466         22       1 2.613089 -0.571622724 2.532284 2.613089
## 319        2.816967         22       1 2.613089  0.203878276 2.532284 2.613089
## 320        2.635127         22       1 2.613089  0.022038276 2.532284 2.613089
## 321        3.049442         22       2 2.459297  0.590145132 2.532284 2.459297
## 322        2.446904         22       2 2.459297 -0.012392868 2.532284 2.459297
## 323        1.793442         22       2 2.459297 -0.665854868 2.532284 2.459297
## 324        2.676519         22       2 2.459297  0.217222132 2.532284 2.459297
## 325        2.187865         22       2 2.459297 -0.271431868 2.532284 2.459297
## 326        2.758416         22       3 2.420937  0.337478956 2.532284 2.420937
## 327        2.405744         22       3 2.420937 -0.015193044 2.532284 2.420937
## 328        1.580387         22       3 2.420937 -0.840550044 2.532284 2.420937
## 329        2.508542         22       3 2.420937  0.087604956 2.532284 2.420937
## 330        2.574564         22       3 2.420937  0.153626956 2.532284 2.420937
## 331        3.294288         23       1 2.764077  0.530211208 2.532284 2.764077
## 332        2.641762         23       1 2.764077 -0.122314792 2.532284 2.764077
## 333        2.105774         23       1 2.764077 -0.658302792 2.532284 2.764077
## 334        2.655097         23       1 2.764077 -0.108979792 2.532284 2.764077
## 335        2.622482         23       1 2.764077 -0.141594792 2.532284 2.764077
## 336        4.066631         23       2 3.229016  0.837615083 2.532284 3.229016
## 337        3.389733         23       2 3.229016  0.160717083 2.532284 3.229016
## 338        2.993666         23       2 3.229016 -0.235349917 2.532284 3.229016
## 339        3.613128         23       2 3.229016  0.384112083 2.532284 3.229016
## 340        3.213809         23       2 3.229016 -0.015206917 2.532284 3.229016
## 341        3.369665         23       3 2.797817  0.571848447 2.532284 2.797817
## 342        2.566891         23       3 2.797817 -0.230925553 2.532284 2.797817
## 343        2.289899         23       3 2.797817 -0.507917553 2.532284 2.797817
## 344        2.517418         23       3 2.797817 -0.280398553 2.532284 2.797817
## 345        2.862723         23       3 2.797817  0.064906447 2.532284 2.797817
## 346        4.212664         24       1 3.155523  1.057140501 2.532284 3.155523
## 347        3.068342         24       1 3.155523 -0.087181499 2.532284 3.155523
## 348        2.872188         24       1 3.155523 -0.283335499 2.532284 3.155523
## 349        3.040890         24       1 3.155523 -0.114633499 2.532284 3.155523
## 350        3.376318         24       1 3.155523  0.220794501 2.532284 3.155523
## 351        3.223384         24       2 2.847300  0.376084016 2.532284 2.847300
## 352        2.552726         24       2 2.847300 -0.294573984 2.532284 2.847300
## 353        2.447344         24       2 2.847300 -0.399955984 2.532284 2.847300
## 354        3.011574         24       2 2.847300  0.164274016 2.532284 2.847300
## 355        2.711774         24       2 2.847300 -0.135525984 2.532284 2.847300
## 356        3.359505         24       3 2.861631  0.497873609 2.532284 2.861631
## 357        2.800742         24       3 2.861631 -0.060889391 2.532284 2.861631
## 358        2.043396         24       3 2.861631 -0.818235391 2.532284 2.861631
## 359        2.929792         24       3 2.861631  0.068160609 2.532284 2.861631
## 360        2.935356         24       3 2.861631  0.073724609 2.532284 2.861631
## 361        2.724871         25       1 2.440734  0.284136829 2.532284 2.440734
## 362        2.239013         25       1 2.440734 -0.201721171 2.532284 2.440734
## 363        2.341512         25       1 2.440734 -0.099222171 2.532284 2.440734
## 364        2.263617         25       1 2.440734 -0.177117171 2.532284 2.440734
## 365        2.062748         25       1 2.440734 -0.377986171 2.532284 2.440734
## 366        3.658082         25       2 2.822441  0.835641036 2.532284 2.822441
## 367        3.093268         25       2 2.822441  0.270827036 2.532284 2.822441
## 368        2.429341         25       2 2.822441 -0.393099964 2.532284 2.822441
## 369        2.538365         25       2 2.822441 -0.284075964 2.532284 2.822441
## 370        3.161795         25       2 2.822441  0.339354036 2.532284 2.822441
## 371        3.178246         25       3 2.561028  0.617217839 2.532284 2.561028
```

```
## 372      2.498102       25      3 2.561028 -0.062926161 2.532284 2.561028
## 373      2.445810       25      3 2.561028 -0.115218161 2.532284 2.561028
## 374      2.231248       25      3 2.561028 -0.329780161 2.532284 2.561028
## 375      2.302298       25      3 2.561028 -0.258730161 2.532284 2.561028
## 376      3.320688       26      1 2.885606  0.435081630 2.532284 2.885606
## 377      2.861800       26      1 2.885606 -0.023806370 2.532284 2.885606
## 378      2.238258       26      1 2.885606 -0.647348370 2.532284 2.885606
## 379      3.122050       26      1 2.885606  0.236443630 2.532284 2.885606
## 380      3.160876       26      1 2.885606  0.275269630 2.532284 2.885606
## 381      3.873888       26      2 3.101135  0.772753435 2.532284 3.101135
## 382      3.166345       26      2 3.101135  0.065210435 2.532284 3.101135
## 383      2.645267       26      2 3.101135 -0.455867565 2.532284 3.101135
## 384      3.309867       26      2 3.101135  0.208732435 2.532284 3.101135
## 385      3.542882       26      2 3.101135  0.441747435 2.532284 3.101135
## 386      2.586453       26      3 2.486541  0.099911614 2.532284 2.486541
## 387      2.120604       26      3 2.486541 -0.365937386 2.532284 2.486541
## 388      2.180847       26      3 2.486541 -0.305694386 2.532284 2.486541
## 389      2.480888       26      3 2.486541 -0.005653386 2.532284 2.486541
## 390      1.938037       26      3 2.486541 -0.548504386 2.532284 2.486541
## 391      4.710718       27      1 4.073210  0.637507673 2.532284 4.073210
## 392      4.082083       27      1 4.073210  0.008872673 2.532284 4.073210
## 393      3.533026       27      1 4.073210 -0.540184327 2.532284 4.073210
## 394      4.269929       27      1 4.073210  0.196718673 2.532284 4.073210
## 395      4.038166       27      1 4.073210 -0.035044327 2.532284 4.073210
## 396      4.237233       27      2 3.890661  0.346571680 2.532284 3.890661
## 397      4.171702       27      2 3.890661  0.281040680 2.532284 3.890661
## 398      3.043940       27      2 3.890661 -0.846721320 2.532284 3.890661
## 399      3.912960       27      2 3.890661  0.022298680 2.532284 3.890661
## 400      3.714229       27      2 3.890661 -0.176432320 2.532284 3.890661
## 401      5.168668       27      3 4.303624  0.865043796 2.532284 4.303624
## 402      4.823275       27      3 4.303624  0.519650796 2.532284 4.303624
## 403      3.764272       27      3 4.303624 -0.539352204 2.532284 4.303624
## 404      4.396897       27      3 4.303624  0.093272796 2.532284 4.303624
## 405      4.442094       27      3 4.303624  0.138469796 2.532284 4.303624
## 406      3.972279       28      1 3.405434  0.566845014 2.532284 3.405434
## 407      3.883295       28      1 3.405434  0.477861014 2.532284 3.405434
## 408      3.045145       28      1 3.405434 -0.360288986 2.532284 3.405434
## 409      3.514590       28      1 3.405434  0.109156014 2.532284 3.405434
## 410      3.575446       28      1 3.405434  0.170012014 2.532284 3.405434
## 411      3.024903       28      2 2.966293  0.058610338 2.532284 2.966293
## 412      3.099192       28      2 2.966293  0.132899338 2.532284 2.966293
## 413      2.048139       28      2 2.966293 -0.918153662 2.532284 2.966293
## 414      2.927978       28      2 2.966293 -0.038314662 2.532284 2.966293
## 415      3.152570       28      2 2.966293  0.186277338 2.532284 2.966293
## 416      3.558060       28      3 3.134582  0.423477722 2.532284 3.134582
## 417      3.176292       28      3 3.134582  0.041709722 2.532284 3.134582
## 418      2.852873       28      3 3.134582 -0.281709278 2.532284 3.134582
## 419      3.026064       28      3 3.134582 -0.108518278 2.532284 3.134582
## 420      3.071975       28      3 3.134582 -0.062607278 2.532284 3.134582
## 421      3.496634       29      1 2.797622  0.699011772 2.532284 2.797622
## 422      3.087091       29      1 2.797622  0.289468772 2.532284 2.797622
## 423      2.517673       29      1 2.797622 -0.279949228 2.532284 2.797622
## 424      2.547344       29      1 2.797622 -0.250278228 2.532284 2.797622
## 425      2.971948       29      1 2.797622  0.174325772 2.532284 2.797622
```

```
## 426      3.371306       29      2 2.589240  0.782065942 2.532284 2.589240
## 427      2.175046       29      2 2.589240 -0.414194058 2.532284 2.589240
## 428      1.940111       29      2 2.589240 -0.649129058 2.532284 2.589240
## 429      2.932408       29      2 2.589240  0.343167942 2.532284 2.589240
## 430      2.428069       29      2 2.589240 -0.161171058 2.532284 2.589240
## 431      2.941041       29      3 2.481746  0.459295244 2.532284 2.481746
## 432      2.294009       29      3 2.481746 -0.187736756 2.532284 2.481746
## 433      2.025674       29      3 2.481746 -0.456071756 2.532284 2.481746
## 434      2.211540       29      3 2.481746 -0.270205756 2.532284 2.481746
## 435      2.459684       29      3 2.481746 -0.022061756 2.532284 2.481746
## 436      2.864670       30      1 2.610508  0.254162464 2.532284 2.610508
## 437      2.695163       30      1 2.610508  0.084655464 2.532284 2.610508
## 438      2.229518       30      1 2.610508 -0.380989536 2.532284 2.610508
## 439      1.940917       30      1 2.610508 -0.669590536 2.532284 2.610508
## 440      2.547318       30      1 2.610508 -0.063189536 2.532284 2.610508
## 441      3.537562       30      2 3.112203  0.425359367 2.532284 3.112203
## 442      3.311361       30      2 3.112203  0.199158367 2.532284 3.112203
## 443      2.767771       30      2 3.112203 -0.344431633 2.532284 3.112203
## 444      3.388622       30      2 3.112203  0.276419367 2.532284 3.112203
## 445      3.542701       30      2 3.112203  0.430498367 2.532284 3.112203
## 446      3.184652       30      3 2.827247  0.357405038 2.532284 2.827247
## 447      2.620947       30      3 2.827247 -0.206299962 2.532284 2.827247
## 448      2.697619       30      3 2.827247 -0.129627962 2.532284 2.827247
## 449      2.860684       30      3 2.827247  0.033437038 2.532284 2.827247
## 450      2.758571       30      3 2.827247 -0.068675962 2.532284 2.827247
##     .offset .sqrtXwt .sqrtrwt .weights       .wtres
## 1         0        1        1        1  0.801869573
## 2         0        1        1        1 -0.144324427
## 3         0        1        1        1 -0.323097427
## 4         0        1        1        1  0.020800573
## 5         0        1        1        1 -0.003673427
## 6         0        1        1        1  0.458029715
## 7         0        1        1        1 -0.193683285
## 8         0        1        1        1 -0.335649285
## 9         0        1        1        1 -0.060815285
## 10        0        1        1        1 -0.220422285
## 11        0        1        1        1  0.633869650
## 12        0        1        1        1 -0.191944350
## 13        0        1        1        1 -0.627992350
## 14        0        1        1        1  0.051129650
## 15        0        1        1        1 -0.019996350
## 16        0        1        1        1  0.841320221
## 17        0        1        1        1  0.200000221
## 18        0        1        1        1 -0.246701779
## 19        0        1        1        1 -0.031702779
## 20        0        1        1        1 -0.198460779
## 21        0        1        1        1  0.075503944
## 22        0        1        1        1  0.229665944
## 23        0        1        1        1 -0.514609056
## 24        0        1        1        1 -0.302973056
## 25        0        1        1        1 -0.086866056
## 26        0        1        1        1  0.248816193
## 27        0        1        1        1  0.075387193
## 28        0        1        1        1 -0.560899807
```

```
## 29          0          1          1          1 -0.130807807
## 30          0          1          1          1  0.154138193
## 31          0          1          1          1  0.464312376
## 32          0          1          1          1 -0.005044624
## 33          0          1          1          1 -0.376480624
## 34          0          1          1          1 -0.060669624
## 35          0          1          1          1 -0.304242624
## 36          0          1          1          1  0.406997082
## 37          0          1          1          1 -0.262300918
## 38          0          1          1          1 -0.304189918
## 39          0          1          1          1  0.241774082
## 40          0          1          1          1 -0.046690918
## 41          0          1          1          1  0.496036378
## 42          0          1          1          1 -0.002822622
## 43          0          1          1          1 -0.381804622
## 44          0          1          1          1 -0.133470622
## 45          0          1          1          1 -0.207836622
## 46          0          1          1          1  0.640159073
## 47          0          1          1          1 -0.252977927
## 48          0          1          1          1 -0.502730927
## 49          0          1          1          1 -0.308554927
## 50          0          1          1          1 -0.125958927
## 51          0          1          1          1  0.539512663
## 52          0          1          1          1 -0.212092337
## 53          0          1          1          1 -0.396778337
## 54          0          1          1          1 -0.296870337
## 55          0          1          1          1 -0.087580337
## 56          0          1          1          1  0.504343818
## 57          0          1          1          1  0.185481818
## 58          0          1          1          1 -0.543686182
## 59          0          1          1          1  0.326301818
## 60          0          1          1          1  0.473855818
## 61          0          1          1          1  0.349007940
## 62          0          1          1          1 -0.195996060
## 63          0          1          1          1 -0.751740060
## 64          0          1          1          1 -0.234507060
## 65          0          1          1          1 -0.120345060
## 66          0          1          1          1  0.716319781
## 67          0          1          1          1  0.206441781
## 68          0          1          1          1 -0.607031219
## 69          0          1          1          1  0.049473781
## 70          0          1          1          1  0.221844781
## 71          0          1          1          1  0.528799176
## 72          0          1          1          1  0.240077176
## 73          0          1          1          1 -0.198389824
## 74          0          1          1          1 -0.215678824
## 75          0          1          1          1  0.304608176
## 76          0          1          1          1  0.399738314
## 77          0          1          1          1 -0.011934686
## 78          0          1          1          1 -0.597132686
## 79          0          1          1          1  0.227346314
## 80          0          1          1          1 -0.155721686
## 81          0          1          1          1  0.647658691
## 82          0          1          1          1 -0.275995309
```

```
## 83          0         1         1         1 -0.315810309
## 84          0         1         1         1  0.073926691
## 85          0         1         1         1 -0.149819309
## 86          0         1         1         1  0.549138363
## 87          0         1         1         1  0.218054363
## 88          0         1         1         1 -1.029921637
## 89          0         1         1         1  0.178841363
## 90          0         1         1         1  0.271770363
## 91          0         1         1         1  0.046518853
## 92          0         1         1         1 -0.096654147
## 93          0         1         1         1 -0.451356147
## 94          0         1         1         1 -0.462400147
## 95          0         1         1         1  0.191573853
## 96          0         1         1         1  0.574152360
## 97          0         1         1         1 -0.015020640
## 98          0         1         1         1 -0.484462640
## 99          0         1         1         1  0.052554360
## 100         0         1         1         1 -0.142614640
## 101         0         1         1         1  0.786902777
## 102         0         1         1         1  0.171120777
## 103         0         1         1         1 -0.572606223
## 104         0         1         1         1 -0.019097223
## 105         0         1         1         1  0.295854777
## 106         0         1         1         1  0.268438271
## 107         0         1         1         1 -0.329477729
## 108         0         1         1         1 -0.718390729
## 109         0         1         1         1  0.258273271
## 110         0         1         1         1 -0.046003729
## 111         0         1         1         1  0.557234600
## 112         0         1         1         1 -0.042673400
## 113         0         1         1         1 -0.634512400
## 114         0         1         1         1  0.430744600
## 115         0         1         1         1 -0.488772400
## 116         0         1         1         1  0.487126019
## 117         0         1         1         1  0.059295019
## 118         0         1         1         1 -0.532016981
## 119         0         1         1         1  0.073983019
## 120         0         1         1         1  0.204810019
## 121         0         1         1         1  0.231253427
## 122         0         1         1         1  0.328060427
## 123         0         1         1         1 -0.260479573
## 124         0         1         1         1 -0.033691573
## 125         0         1         1         1  0.098017427
## 126         0         1         1         1  0.540665183
## 127         0         1         1         1 -0.016695817
## 128         0         1         1         1 -0.571891817
## 129         0         1         1         1 -0.135502817
## 130         0         1         1         1  0.032109183
## 131         0         1         1         1  0.344723358
## 132         0         1         1         1 -0.144176642
## 133         0         1         1         1 -0.877156642
## 134         0         1         1         1  0.333996358
## 135         0         1         1         1 -0.317894642
## 136         0         1         1         1  0.468654771
```

```
## 137        0        1        1        1  0.145350771
## 138        0        1        1        1 -0.418855229
## 139        0        1        1        1 -0.007597229
## 140        0        1        1        1 -0.046277229
## 141        0        1        1        1  0.856312640
## 142        0        1        1        1 -0.160442360
## 143        0        1        1        1 -0.424646360
## 144        0        1        1        1 -0.356012360
## 145        0        1        1        1 -0.261510360
## 146        0        1        1        1  0.738288287
## 147        0        1        1        1 -0.043516713
## 148        0        1        1        1 -0.460673713
## 149        0        1        1        1  0.141088287
## 150        0        1        1        1 -0.313079713
## 151        0        1        1        1  0.594634324
## 152        0        1        1        1  0.085294324
## 153        0        1        1        1 -0.427002676
## 154        0        1        1        1 -0.242592676
## 155        0        1        1        1 -0.108613676
## 156        0        1        1        1  0.358044656
## 157        0        1        1        1  0.204948656
## 158        0        1        1        1 -0.864721344
## 159        0        1        1        1 -0.036632344
## 160        0        1        1        1 -0.115132344
## 161        0        1        1        1  0.697964510
## 162        0        1        1        1  0.214933510
## 163        0        1        1        1 -0.656773490
## 164        0        1        1        1 -0.053573490
## 165        0        1        1        1  0.435403510
## 166        0        1        1        1  0.599020267
## 167        0        1        1        1 -0.365305733
## 168        0        1        1        1 -0.561473733
## 169        0        1        1        1 -0.055349733
## 170        0        1        1        1 -0.084569733
## 171        0        1        1        1  0.542589503
## 172        0        1        1        1 -0.234946497
## 173        0        1        1        1 -0.612423497
## 174        0        1        1        1  0.102692503
## 175        0        1        1        1  0.064849503
## 176        0        1        1        1  0.760405467
## 177        0        1        1        1  0.007584467
## 178        0        1        1        1 -0.655761533
## 179        0        1        1        1 -0.153855533
## 180        0        1        1        1  0.214990467
## 181        0        1        1        1  0.472015127
## 182        0        1        1        1  0.165540127
## 183        0        1        1        1 -0.410203873
## 184        0        1        1        1  0.209396127
## 185        0        1        1        1 -0.323404873
## 186        0        1        1        1  0.524357171
## 187        0        1        1        1  0.030949171
## 188        0        1        1        1 -0.437480829
## 189        0        1        1        1  0.017253171
## 190        0        1        1        1  0.087933171
```

```
## 191         0         1         1         1  0.339226160
## 192         0         1         1         1  0.299115160
## 193         0         1         1         1 -0.503691840
## 194         0         1         1         1 -0.204944840
## 195         0         1         1         1 -0.145257840
## 196         0         1         1         1  0.599990877
## 197         0         1         1         1  0.010367877
## 198         0         1         1         1 -0.331764123
## 199         0         1         1         1 -0.024956123
## 200         0         1         1         1 -0.238043123
## 201         0         1         1         1  0.718980355
## 202         0         1         1         1  0.420670355
## 203         0         1         1         1 -0.586661645
## 204         0         1         1         1  0.095842355
## 205         0         1         1         1 -0.228763645
## 206         0         1         1         1  0.341667894
## 207         0         1         1         1 -0.117775106
## 208         0         1         1         1 -0.530025106
## 209         0         1         1         1 -0.102293106
## 210         0         1         1         1  0.233757894
## 211         0         1         1         1  0.549138745
## 212         0         1         1         1 -0.116673255
## 213         0         1         1         1 -0.881908255
## 214         0         1         1         1 -0.136724255
## 215         0         1         1         1 -0.359874255
## 216         0         1         1         1  0.468799916
## 217         0         1         1         1  0.080472916
## 218         0         1         1         1 -0.352495084
## 219         0         1         1         1  0.117665916
## 220         0         1         1         1 -0.077853084
## 221         0         1         1         1 -0.056978905
## 222         0         1         1         1  0.144479095
## 223         0         1         1         1 -0.034933905
## 224         0         1         1         1  0.230716095
## 225         0         1         1         1  0.005243095
## 226         0         1         1         1  0.546321560
## 227         0         1         1         1 -0.179239440
## 228         0         1         1         1 -0.378894440
## 229         0         1         1         1  0.426996560
## 230         0         1         1         1  0.043508560
## 231         0         1         1         1  0.472315945
## 232         0         1         1         1 -0.145160055
## 233         0         1         1         1 -0.168368055
## 234         0         1         1         1 -0.227610055
## 235         0         1         1         1 -0.368202055
## 236         0         1         1         1  0.896419252
## 237         0         1         1         1  0.030282252
## 238         0         1         1         1 -0.746355748
## 239         0         1         1         1 -0.390441748
## 240         0         1         1         1  0.136893252
## 241         0         1         1         1  0.891817709
## 242         0         1         1         1  0.230829709
## 243         0         1         1         1 -0.518405291
## 244         0         1         1         1 -0.094760291
```

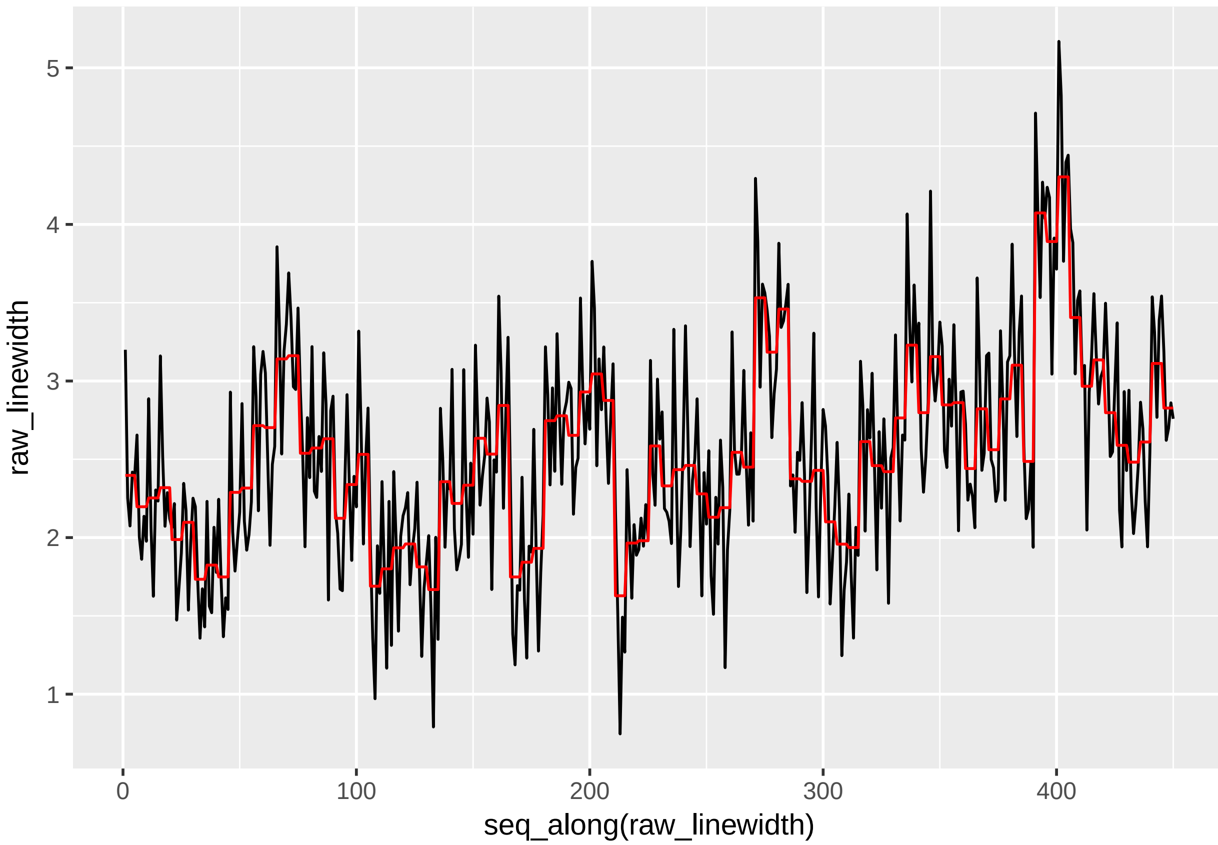
```
## 245        0        1        1        1  0.040171709
## 246        0        1        1        1  0.607114639
## 247        0        1        1        1  0.013333639
## 248        0        1        1        1 -0.651607361
## 249        0        1        1        1  0.135906639
## 250        0        1        1        1 -0.193035361
## 251        0        1        1        1  0.424974257
## 252        0        1        1        1 -0.374030743
## 253        0        1        1        1 -0.619749743
## 254        0        1        1        1  0.127473257
## 255        0        1        1        1 -0.171281743
## 256        0        1        1        1  0.431561594
## 257        0        1        1        1  0.129907594
## 258        0        1        1        1 -1.021902406
## 259        0        1        1        1 -0.269714406
## 260        0        1        1        1 -0.014794406
## 261        0        1        1        1  0.768846869
## 262        0        1        1        1  0.015204869
## 263        0        1        1        1 -0.139858131
## 264        0        1        1        1 -0.139271131
## 265        0        1        1        1 -0.008902131
## 266        0        1        1        1  0.618058143
## 267        0        1        1        1  0.040566143
## 268        0        1        1        1 -0.370315857
## 269        0        1        1        1  0.219719143
## 270        0        1        1        1 -0.344689857
## 271        0        1        1        1  0.762745066
## 272        0        1        1        1  0.357682066
## 273        0        1        1        1 -0.570488934
## 274        0        1        1        1  0.087720066
## 275        0        1        1        1  0.031336066
## 276        0        1        1        1  0.267717185
## 277        0        1        1        1  0.101779185
## 278        0        1        1        1 -0.545860815
## 279        0        1        1        1 -0.265344815
## 280        0        1        1        1 -0.107923815
## 281        0        1        1        1  0.420306531
## 282        0        1        1        1 -0.117350469
## 283        0        1        1        1 -0.076543469
## 284        0        1        1        1  0.032289531
## 285        0        1        1        1  0.158244531
## 286        0        1        1        1 -0.045257340
## 287        0        1        1        1  0.025032660
## 288        0        1        1        1 -0.341303340
## 289        0        1        1        1  0.169122660
## 290        0        1        1        1  0.117834660
## 291        0        1        1        1  0.502911902
## 292        0        1        1        1  0.045530902
## 293        0        1        1        1 -0.710510098
## 294        0        1        1        1 -0.243707098
## 295        0        1        1        1  0.274757902
## 296        0        1        1        1  0.875850030
## 297        0        1        1        1 -0.234369970
## 298        0        1        1        1 -0.808397970
```

```
## 299        0        1        1        1 -0.106682970
## 300        0        1        1        1  0.389088030
## 301        0        1        1        1  0.612241684
## 302        0        1        1        1  0.288447684
## 303        0        1        1        1 -0.524840316
## 304        0        1        1        1 -0.230189316
## 305        0        1        1        1  0.102588684
## 306        0        1        1        1  0.649873442
## 307        0        1        1        1  0.219648442
## 308        0        1        1        1 -0.712082558
## 309        0        1        1        1 -0.295002558
## 310        0        1        1        1 -0.114911558
## 311        0        1        1        1  0.341509308
## 312        0        1        1        1 -0.171363692
## 313        0        1        1        1 -0.578166692
## 314        0        1        1        1  0.129409308
## 315        0        1        1        1 -0.050406692
## 316        0        1        1        1  0.513095276
## 317        0        1        1        1  0.230416276
## 318        0        1        1        1 -0.571622724
## 319        0        1        1        1  0.203878276
## 320        0        1        1        1  0.022038276
## 321        0        1        1        1  0.590145132
## 322        0        1        1        1 -0.012392868
## 323        0        1        1        1 -0.665854868
## 324        0        1        1        1  0.217222132
## 325        0        1        1        1 -0.271431868
## 326        0        1        1        1  0.337478956
## 327        0        1        1        1 -0.015193044
## 328        0        1        1        1 -0.840550044
## 329        0        1        1        1  0.087604956
## 330        0        1        1        1  0.153626956
## 331        0        1        1        1  0.530211208
## 332        0        1        1        1 -0.122314792
## 333        0        1        1        1 -0.658302792
## 334        0        1        1        1 -0.108979792
## 335        0        1        1        1 -0.141594792
## 336        0        1        1        1  0.837615083
## 337        0        1        1        1  0.160717083
## 338        0        1        1        1 -0.235349917
## 339        0        1        1        1  0.384112083
## 340        0        1        1        1 -0.015206917
## 341        0        1        1        1  0.571848447
## 342        0        1        1        1 -0.230925553
## 343        0        1        1        1 -0.507917553
## 344        0        1        1        1 -0.280398553
## 345        0        1        1        1  0.064906447
## 346        0        1        1        1  1.057140501
## 347        0        1        1        1 -0.087181499
## 348        0        1        1        1 -0.283335499
## 349        0        1        1        1 -0.114633499
## 350        0        1        1        1  0.220794501
## 351        0        1        1        1  0.376084016
## 352        0        1        1        1 -0.294573984
```

```
## 353         0         1         1         1 -0.399955984
## 354         0         1         1         1  0.164274016
## 355         0         1         1         1 -0.135525984
## 356         0         1         1         1  0.497873609
## 357         0         1         1         1 -0.060889391
## 358         0         1         1         1 -0.818235391
## 359         0         1         1         1  0.068160609
## 360         0         1         1         1  0.073724609
## 361         0         1         1         1  0.284136829
## 362         0         1         1         1 -0.201721171
## 363         0         1         1         1 -0.099222171
## 364         0         1         1         1 -0.177117171
## 365         0         1         1         1 -0.377986171
## 366         0         1         1         1  0.835641036
## 367         0         1         1         1  0.270827036
## 368         0         1         1         1 -0.393099964
## 369         0         1         1         1 -0.284075964
## 370         0         1         1         1  0.339354036
## 371         0         1         1         1  0.617217839
## 372         0         1         1         1 -0.062926161
## 373         0         1         1         1 -0.115218161
## 374         0         1         1         1 -0.329780161
## 375         0         1         1         1 -0.258730161
## 376         0         1         1         1  0.435081630
## 377         0         1         1         1 -0.023806370
## 378         0         1         1         1 -0.647348370
## 379         0         1         1         1  0.236443630
## 380         0         1         1         1  0.275269630
## 381         0         1         1         1  0.772753435
## 382         0         1         1         1  0.065210435
## 383         0         1         1         1 -0.455867565
## 384         0         1         1         1  0.208732435
## 385         0         1         1         1  0.441747435
## 386         0         1         1         1  0.099911614
## 387         0         1         1         1 -0.365937386
## 388         0         1         1         1 -0.305694386
## 389         0         1         1         1 -0.005653386
## 390         0         1         1         1 -0.548504386
## 391         0         1         1         1  0.637507673
## 392         0         1         1         1  0.008872673
## 393         0         1         1         1 -0.540184327
## 394         0         1         1         1  0.196718673
## 395         0         1         1         1 -0.035044327
## 396         0         1         1         1  0.346571680
## 397         0         1         1         1  0.281040680
## 398         0         1         1         1 -0.846721320
## 399         0         1         1         1  0.022298680
## 400         0         1         1         1 -0.176432320
## 401         0         1         1         1  0.865043796
## 402         0         1         1         1  0.519650796
## 403         0         1         1         1 -0.539352204
## 404         0         1         1         1  0.093272796
## 405         0         1         1         1  0.138469796
## 406         0         1         1         1  0.566845014
```

```
## 407       0       1       1       1  0.477861014
## 408       0       1       1       1 -0.360288986
## 409       0       1       1       1  0.109156014
## 410       0       1       1       1  0.170012014
## 411       0       1       1       1  0.058610338
## 412       0       1       1       1  0.132899338
## 413       0       1       1       1 -0.918153662
## 414       0       1       1       1 -0.038314662
## 415       0       1       1       1  0.186277338
## 416       0       1       1       1  0.423477722
## 417       0       1       1       1  0.041709722
## 418       0       1       1       1 -0.281709278
## 419       0       1       1       1 -0.108518278
## 420       0       1       1       1 -0.062607278
## 421       0       1       1       1  0.699011772
## 422       0       1       1       1  0.289468772
## 423       0       1       1       1 -0.279949228
## 424       0       1       1       1 -0.250278228
## 425       0       1       1       1  0.174325772
## 426       0       1       1       1  0.782065942
## 427       0       1       1       1 -0.414194058
## 428       0       1       1       1 -0.649129058
## 429       0       1       1       1  0.343167942
## 430       0       1       1       1 -0.161171058
## 431       0       1       1       1  0.459295244
## 432       0       1       1       1 -0.187736756
## 433       0       1       1       1 -0.456071756
## 434       0       1       1       1 -0.270205756
## 435       0       1       1       1 -0.022061756
## 436       0       1       1       1  0.254162464
## 437       0       1       1       1  0.084655464
## 438       0       1       1       1 -0.380989536
## 439       0       1       1       1 -0.669590536
## 440       0       1       1       1 -0.063189536
## 441       0       1       1       1  0.425359367
## 442       0       1       1       1  0.199158367
## 443       0       1       1       1 -0.344431633
## 444       0       1       1       1  0.276419367
## 445       0       1       1       1  0.430498367
## 446       0       1       1       1  0.357405038
## 447       0       1       1       1 -0.206299962
## 448       0       1       1       1 -0.129627962
## 449       0       1       1       1  0.033437038
## 450       0       1       1       1 -0.068675962
```

```r
ggplot(augment(random_effects_model)) +
  geom_line(aes(seq_along(raw_linewidth), raw_linewidth)) +
  geom_line(aes(seq_along(raw_linewidth), .fitted), colour = "red")
```


5
4
3
2
1
raw_linewidth
0
100
200
300
400
seq_along(raw_linewidth)

Chapter 7

Product and Process Comparisons

7.1 Packages used in this chapter

```
library(magrittr) # used for %$% pipe
library(tidyverse)
library(ggplot2)
```

7.2 Exercises

7.2.1 7.2.2. Are the data consistent with the assumed process mean?

process comparision

7.3 Student's t-test

7.3.1 "illustrative example of the t-test" in section 7.2.2 - particle (contamination) counts

> Over a long run the process average for wafer particle counts has been 50 counts per wafer. We want to test whether a change has occurred based on new data. The null hypothesis that the process mean is 50 counts is tested against the alternative hypothesis that the process mean is not equal to 50 counts.

$$H_0 : \mu = \mu_0$$
$$H_a : \mu \neq \mu_0$$

The purpose of the two-sided alternative is to rule out a possible process change in either direction.

```
particles <- tribble( ~test1, 50, 48, 44, 56, 61, 52, 53, 55, 67, 51)
particles
```

```
## # A tibble: 10 x 1
##    test1
##    <dbl>
```

```
## 1   50.
## 2   48.
## 3   44.
## 4   56.
## 5   61.
## 6   52.
## 7   53.
## 8   55.
## 9   67.
## 10  51.
```

We can generate the needed summary statistics:

```
particle_summary <- particles %>%
summarise(particle_mean = mean(test1), particle_sd = sd(test1), particle_count = n())

particle_summary
```

```
## # A tibble: 1 x 3
##   particle_mean particle_sd particle_count
##           <dbl>       <dbl>          <int>
## 1          53.7        6.57             10
```

Let do this simple example by hand and then compare the result to the `t.test()` function from the `stats` package

$$t = \frac{\overline{Y} - \mu_0}{s\ \sqrt{n}}$$

```
t_particle <- (particle_summary$particle_mean - 50)/(particle_summary$particle_sd/sqrt(particle_summary$
t_critical <- qt(1-0.05/2, df = particle_summary$particle_count - 1)
t_critical
```

```
## [1] 2.262157
```

```
t_particle
```

```
## [1] 1.781768
```

Becaues the value of t_paticle is inside the interval (-2.26, 2.26), we can not reject the null hypothysis and, therefore, we may continue to assume the process mean is 50 counts.

```
particle_summary_t <- particles %>%
summarise(particle_mean = mean(test1), particle_sd = sd(test1), particle_count = n(), t_particle = (par

particle_summary_t
```

```
## # A tibble: 1 x 5
##   particle_mean particle_sd particle_count t_particle t_critical
##           <dbl>       <dbl>          <int>      <dbl>      <dbl>
## 1          53.7        6.57             10       1.78       2.26
```

An alternative method would be to use the `t.test()` function

```
particle_t_test <- t.test(particles$test1, alternative = "two.sided", mu = 50, conf.level = 0.95)
particle_t_test
```

```
##
##  One Sample t-test
```

```
##
## data:  particles$test1
## t = 1.7818, df = 9, p-value = 0.1085
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  49.00243 58.39757
## sample estimates:
## mean of x
##      53.7
```

NEW function alert!
Load the `library(magrittr)` to use the `%$%` function. This allows calling column names within the piped function which is useful for working with base R functions

```
# library(magrittr) # to use the %$% function; allows calling column names within the piped function; u

particle_t_test2 <- particles %$%
  t.test(test1, alternative = "two.sided", mu = 50, conf.level = 0.95)
particle_t_test2
```

```
##
##  One Sample t-test
##
## data:  test1
## t = 1.7818, df = 9, p-value = 0.1085
## alternative hypothesis: true mean is not equal to 50
## 95 percent confidence interval:
##  49.00243 58.39757
## sample estimates:
## mean of x
##      53.7
```

7.3.2 Do two processes have the same mean? in section 7.3.1 - Example of unequal number of data points

A new procedure (process 2) to assemble a device is introduced and tested for possible improvement in time of assembly. The question being addressed is whether the mean, 2, of the new assembly process is smaller than the mean, 1, for the old assembly process (process 1).

$$H_0 : \mu_{process\,2} = \mu_{process\,1}$$
$$H_a : \mu_{process\,2} < \mu_{process\,1}$$

```
device_test <- tribble(
~device, ~process_old, ~process_new,
1, 32, 36,
2, 37, 31,
3, 35, 30,
4, 28, 31,
5, 41, 34,
6, 44, 36,
7, 35, 29,
8, 31, 32,
9, 34, 31,
10, 38, NA,
```

```
11, 42, NA)

device_test
```

```
## # A tibble: 11 x 3
##    device process_old process_new
##     <dbl>       <dbl>       <dbl>
##  1     1.         32.         36.
##  2     2.         37.         31.
##  3     3.         35.         30.
##  4     4.         28.         31.
##  5     5.         41.         34.
##  6     6.         44.         36.
##  7     7.         35.         29.
##  8     8.         31.         32.
##  9     9.         34.         31.
## 10    10.         38.         NA
## 11    11.         42.         NA
```

```
device_summary <- device_test %>%
  dplyr::select(process_old, process_new) %>%
  summary()

device_summary
```

```
##   process_old     process_new
##  Min.   :28.00   Min.   :29.00
##  1st Qu.:33.00   1st Qu.:31.00
##  Median :35.00   Median :31.00
##  Mean   :36.09   Mean   :32.22
##  3rd Qu.:39.50   3rd Qu.:34.00
##  Max.   :44.00   Max.   :36.00
##                  NA's   :2
```

```
device_t_test <- device_test %$%
  t.test(process_new, process_old, alternative = "less", var.equal = FALSE, conf.level = 0.95)
device_t_test
```

```
##
##  Welch Two Sample t-test
##
## data:  process_new and process_old
## t = -2.2694, df = 15.533, p-value = 0.01894
## alternative hypothesis: true difference in means is less than 0
## 95 percent confidence interval:
##        -Inf -0.8869087
## sample estimates:
## mean of x mean of y
##  32.22222  36.09091
```

```
(-qt(1-0.05, df = 15.533))
```

```
## [1] -1.749109
```

7.4 One more classic example! (from Student himself)

From the article

> I will conclude with an example which comes beyond the range of the tables, there being eleven experiments.
>
> To test whether it is of advantage to kiln-dry barley seed before sowing, seven varieties of barley wore sown (both kiln-dried and not kiln-dried in 1899 and four in 1900; the results are given in the table.

```
corn <- read_table2("sample reg kiln
1  1903 2009
2  1935 1915
3  1910 2011
4  2496 2463
5  2108 2180
6  1961 1925
7  2060 2122
8  1444 1482
9  1612 1542
10 1316 1443
11 1511 1535", col_names = TRUE, col_types = cols("i", "d", "d"))

corn %<>% mutate(year = case_when(
  sample <= 7 ~ "cy1889",
  sample > 7 ~ "cy1900"
))

corn
```

```
## # A tibble: 11 x 4
##    sample   reg  kiln year
##     <int> <dbl> <dbl> <chr>
##  1      1 1903. 2009. cy1889
##  2      2 1935. 1915. cy1889
##  3      3 1910. 2011. cy1889
##  4      4 2496. 2463. cy1889
##  5      5 2108. 2180. cy1889
##  6      6 1961. 1925. cy1889
##  7      7 2060. 2122. cy1889
##  8      8 1444. 1482. cy1900
##  9      9 1612. 1542. cy1900
## 10     10 1316. 1443. cy1900
## 11     11 1511. 1535. cy1900
```

```
corn_t_test_wrong <- corn %$%
  t.test(reg, kiln, alternative = "two.sided", var.equal = TRUE, conf.level = 0.95)

corn_t_test_wrong
```

```
##
##  Two Sample t-test
##
## data:  reg and kiln
## t = -0.23413, df = 20, p-value = 0.8173
```

```
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -334.2127  266.7581
## sample estimates:
## mean of x mean of y
##  1841.455  1875.182
```

```r
corn_t_test_correct <- corn %$%
  t.test(reg, kiln, paired = TRUE, alternative = "two.sided", var.equal = TRUE, conf.level = 0.95)

corn_t_test_correct
```

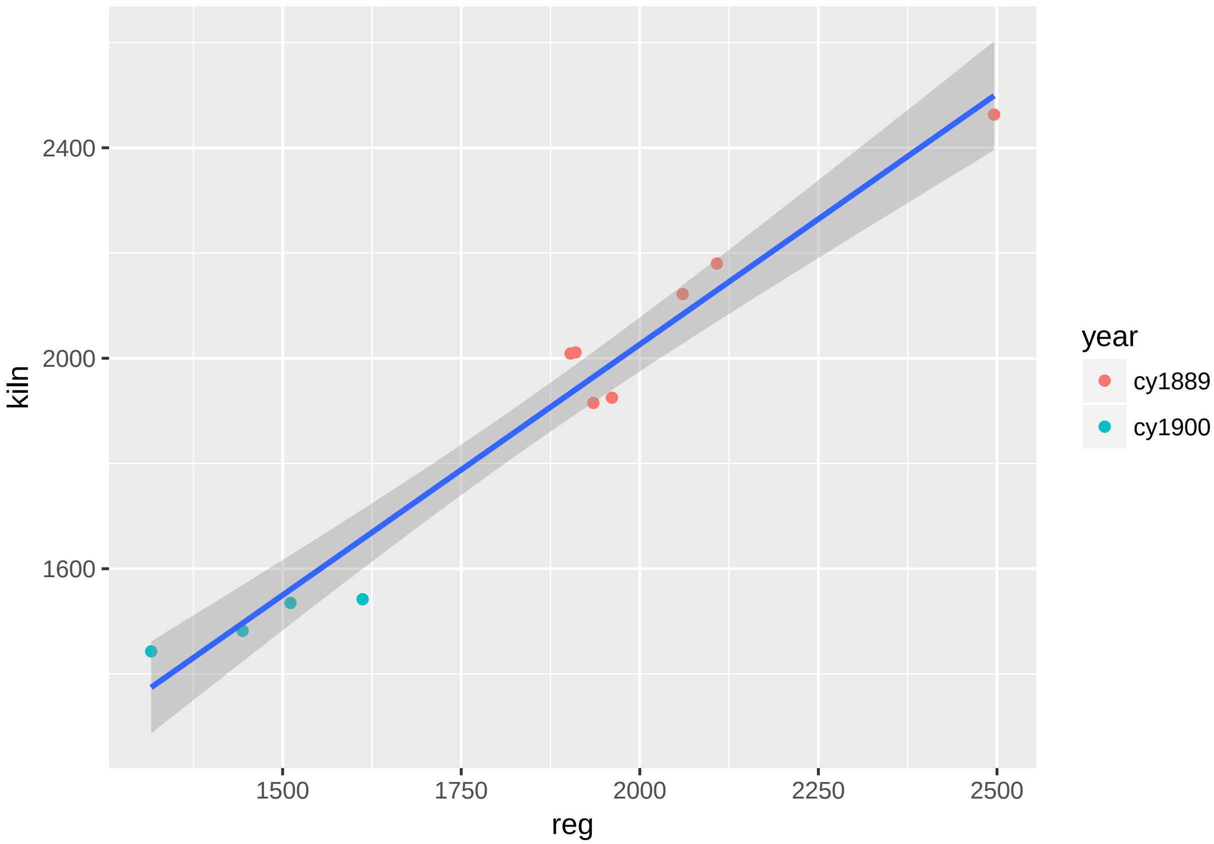
```
##
##  Paired t-test
##
## data:  reg and kiln
## t = -1.6905, df = 10, p-value = 0.1218
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -78.18164  10.72710
## sample estimates:
## mean of the differences
##               -33.72727
```

7.4.1 plot of Student's (W.S. Gossett) data

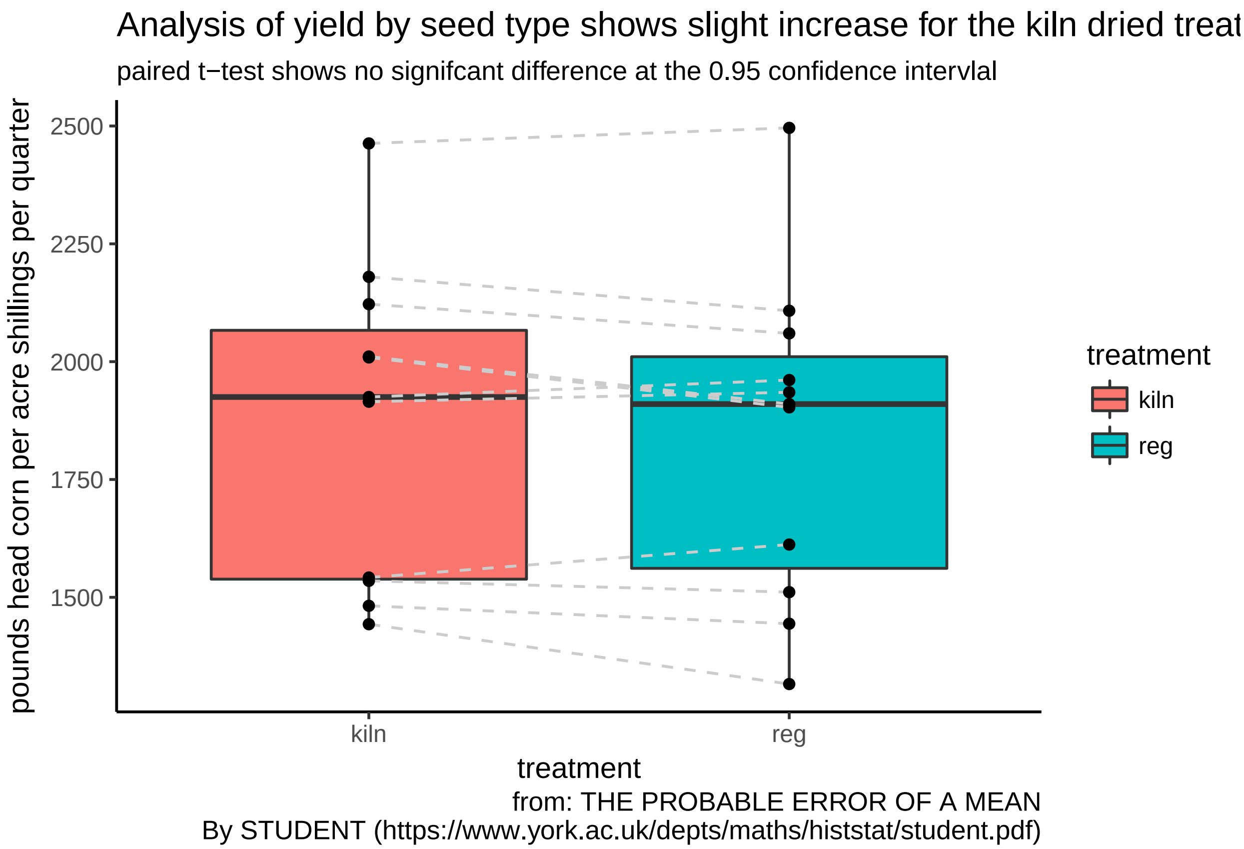
```r
corn_tidy <- corn %>%
  gather(reg, kiln, key = treatment, value = "yield")

corn_tidy
```

```
## # A tibble: 22 x 4
##    sample year   treatment yield
##     <int> <chr>  <chr>     <dbl>
##  1      1 cy1889 reg       1903.
##  2      2 cy1889 reg       1935.
##  3      3 cy1889 reg       1910.
##  4      4 cy1889 reg       2496.
##  5      5 cy1889 reg       2108.
##  6      6 cy1889 reg       1961.
##  7      7 cy1889 reg       2060.
##  8      8 cy1900 reg       1444.
##  9      9 cy1900 reg       1612.
## 10     10 cy1900 reg       1316.
## # ... with 12 more rows
```

```r
ggplot(corn) +
  geom_point(aes(reg, kiln, colour = year)) +
  geom_smooth(aes(reg, kiln), method = "lm")
```




```r
ggplot(corn_tidy, aes(treatment, yield)) +
  geom_boxplot(aes(fill = treatment)) +
  geom_line(aes(group = sample), linetype = "dashed", colour = "grey80") +
  geom_point() +
  theme_classic() +
  labs(title = "Analysis of yield by seed type shows slight increase for the kiln dried treatement",
       subtitle = "paired t-test shows no signifcant difference at the 0.95 confidence intervlal",
       y = "pounds head corn per acre shillings per quarter", caption = "from: THE PROBABLE ERROR OF A M
By STUDENT (https://www.york.ac.uk/depts/maths/histstat/student.pdf)")
```



7.5 Anova

From the *NIST Engineering and Statistics Handbook*

> ANOVA is a general technique that can be used to test the hypothesis that the means among two or more groups are equal, under the assumption that the sampled populations are normally distributed.
>
> The ANOVA procedure is one of the most powerful statistical techniques

The following example is adapted from https://onlinecourses.science.psu.edu/stat502/node/150

> a plant biologist thinks that plant height may be affected by applying different fertilizers. They tested three kinds of fertilizer and also one group of plants that are untreated (the control). They kept all the plants under controlled conditions in the greenhouse. (In addition, we need to have some information about replication and randomization.) They randomly assigned the fertilizer treatment levels to individual containerized plants to produce 6 replications of each of the fertilizer applications.

Image available

```
lesson1_data <- read_table2("Control    F1  F2  F3
21  32  22.5    28
19.5    30.5    26  27.5
22.5    25  28  31
21.5    27.5    27  29.5
20.5    28  26.5    30
```

```
21  28.6    25.2    29.2", col_names = TRUE)
lesson1_data
```

```
## # A tibble: 6 x 4
##   Control    F1    F2    F3
##     <dbl> <dbl> <dbl> <dbl>
## 1    21.0  32.0  22.5  28.0
## 2    19.5  30.5  26.0  27.5
## 3    22.5  25.0  28.0  31.0
## 4    21.5  27.5  27.0  29.5
## 5    20.5  28.0  26.5  30.0
## 6    21.0  28.6  25.2  29.2
```

One-way ANOVA table: the basic format

Source of Variation	Sum of Squares (SS)	Degrees of Feedom (df)	Mean Squares (MS)	F-Ratio
Between samples	SSB	k - 1	MSB	MSB/MSW
Within samples	SSW	n(total) - k	MSW	
Total	SST	n(total) - 1		

One-way ANOVA table: NIST Handbook

Source	SS	DF	MS	F
Treatments	SST	k−1	SST/(k−1)	MST/MSE
Error	SSE	N−k	SSE/(N−k)	
Total (corrected)	SS	N−1		

$$\textbf{Total Sum of Squares } SST = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(x_{ij} - \overline{\overline{x}}\right)^2$$

7.5.1 Tidy the data and compute the sum of squares

```
lesson1_gather <- lesson1_data %>%
  gather(key = treatment, value = value, Control, F1, F2, F3)
lesson1_gather
```

```
## # A tibble: 24 x 2
##    treatment value
##    <chr>     <dbl>
##  1 Control    21.0
##  2 Control    19.5
##  3 Control    22.5
##  4 Control    21.5
##  5 Control    20.5
##  6 Control    21.0
##  7 F1         32.0
##  8 F1         30.5
##  9 F1         25.0
```

```
## 10 F1          27.5
## # ... with 14 more rows
```

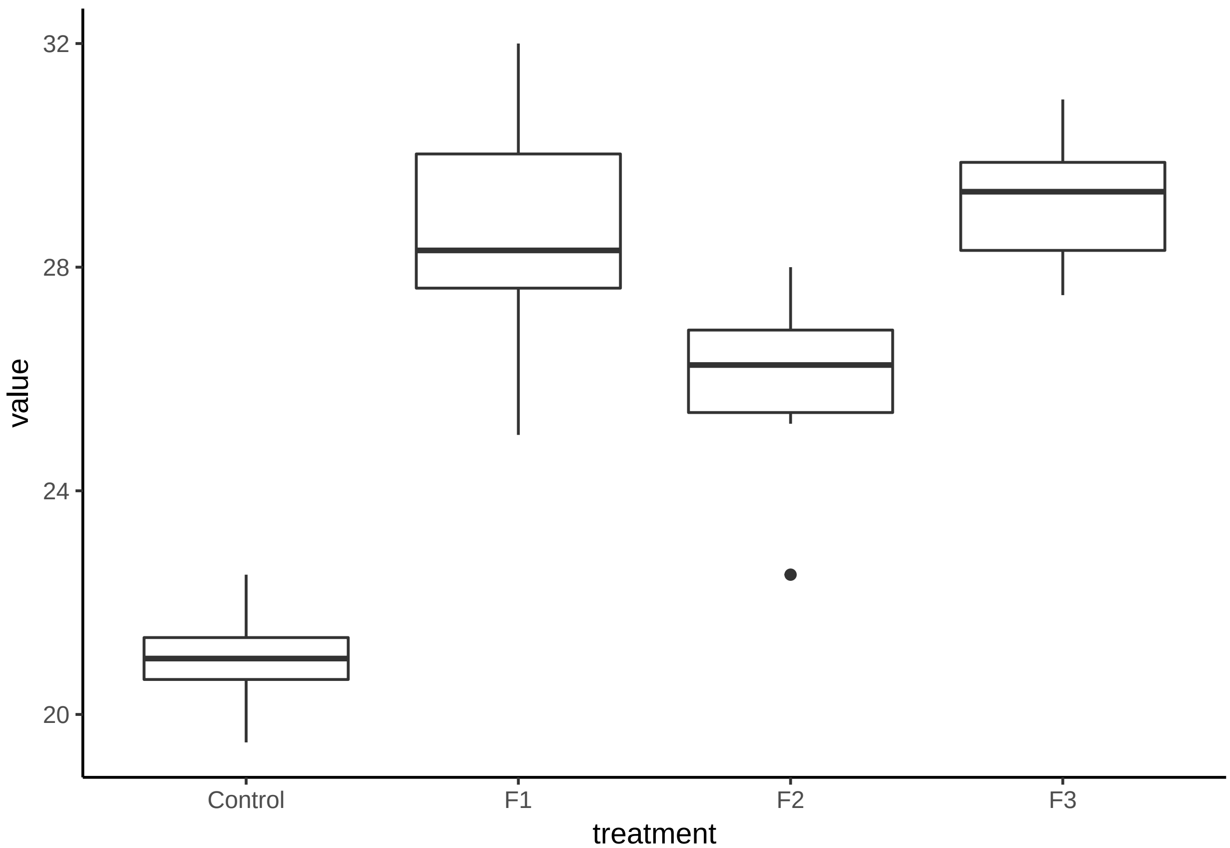
```
lesson1_grand_mean <- lesson1_gather %$%
  mean(value)

lesson1_grand_mean
```

```
## [1] 26.16667
```

```
lesson1_SST <- lesson1_gather %$%
  sum((value - mean(value))^2)

lesson1_SST
```

```
## [1] 312.4733
```

```
ggplot(lesson1_gather) +
  geom_boxplot(aes(treatment, value)) +
  theme_classic()
```



One-way ANOVA table: Lesson1 Example

Source of Variation	Sum of Squares (SS)	Degrees of Feedom (df)	Mean Squares (MS)	F-Ratio
Between samples	SSB	k - 1	MSB	MSB/MSW
Within samples	SSW	n(total) - k	MSW	
Total	SST = 312.43	n(total) - 1 = 23		

$$\textbf{Total Sum of Squares } SST = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(x_{ij} - \overline{\overline{x}}\right)^2$$

$$\textbf{Sum of Squares Between } SSB = \sum_{i=1}^{k} n_i\left(\overline{x}_i - \overline{\overline{x}}\right)^2$$

$$\textbf{Sum of Squares Within } SSW = SST - SSBorSSW = \sum_{i=1}^{k}\sum_{j=1}^{n_i}\left(x_{ij} - \overline{x_i}\right)^2$$

```
summary_within_groups <- lesson1_gather %>%
  group_by(treatment) %>%
  summarise(N = n(), mean = mean(value)) # SS = (sum( (value - mean(value))^2

summary_within_groups
```

```
## # A tibble: 4 x 3
##   treatment     N  mean
##   <chr>     <int> <dbl>
## 1 Control       6  21.0
## 2 F1            6  28.6
## 3 F2            6  25.9
## 4 F3            6  29.2
```

```
lesson1_SSB <- summary_within_groups %$%
  sum((mean - lesson1_grand_mean)^2)*6
message(cat("SSB ", lesson1_SSB))
```

```
## SSB  251.44
```

```
##
```

```
lesson1_SSW = lesson1_SST - lesson1_SSB
message(cat("SSW ", lesson1_SSW))
```

```
## SSW  61.03333
```

```
##
```

One-way ANOVA table: Lesson1 Example

Source of Variation	Sum of Squares (SS)	Degrees of Feedom (df)	Mean Squares (MS)	F-Ratio
Between samples	SSB = 251.44	k - 1 = 3	83.81	MSB/MSW
Within samples	SSW = 61.03	n(total) - k = 20	3.05	
Total	SST = 312.43	n(total) - 1 = 23		

$$\frac{MSB}{BSW} = \frac{83.81}{3.05} = 27.47$$

Calculate the critical F-statistic (or look it up in a table)

```
qf(0.95, df1=3, df2=20)
```

```
## [1] 3.098391
```

Wtih 27.47 > 3.1 we can regect the null hypothesis.

7.6 Let's let R do the work:

```
lesson1_aov <- aov(value ~ treatment, lesson1_gather)
summary(lesson1_aov)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)    
## treatment    3 251.44   83.81   27.46 2.71e-07 ***
## Residuals   20  61.03    3.05                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.7 Which populations have different means?

7.7.1 Tukey (or Tukey-Kramer) test

For the example above, we would have constructed the following hypothesis:

$$H_0 : \mu_{control} = \mu_{F1} = \mu_{F2} = \mu_{F3}$$
$$H_a : \text{At least two population means are different.}$$

The ANOVA analysis above only tells us that there is a difference between two or more of the population means.

We could do a pairwise comparison using confidence intervals for each mean; however, this method does not use the entire population variance.

The *Tukey-Kramer proceedure for multiple comparisons* is one method to compare two or more groups

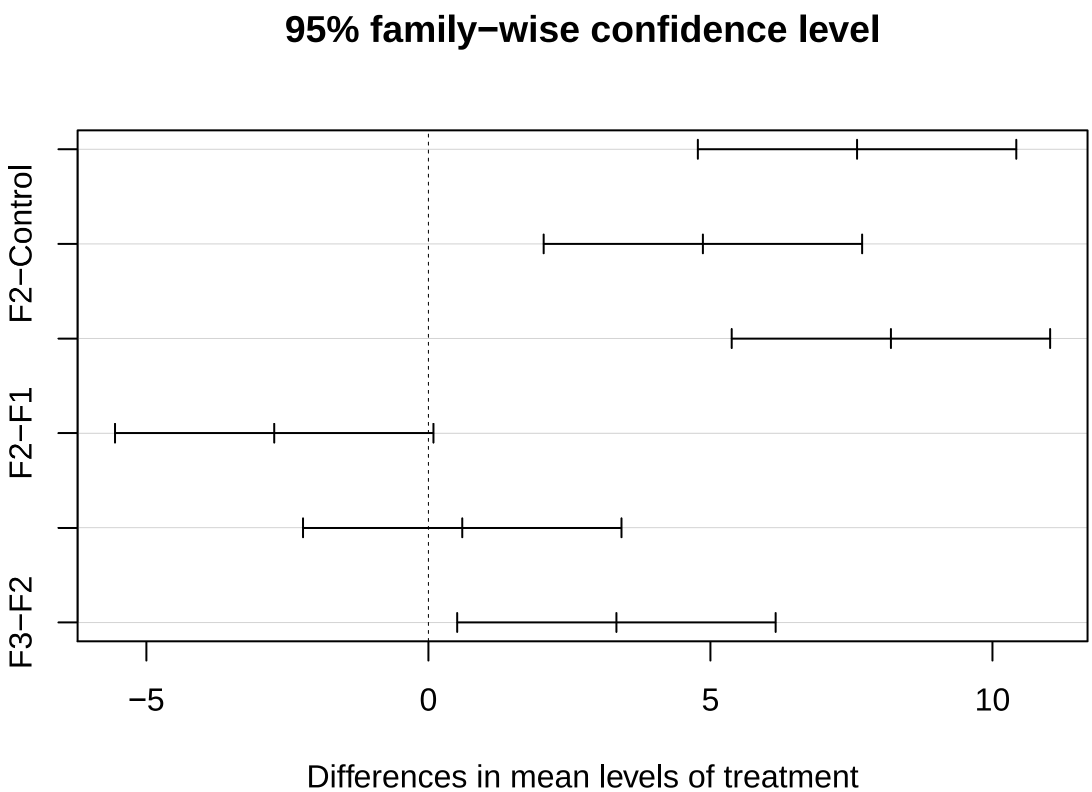
```
lesson1_Tukey <- TukeyHSD(lesson1_aov)
lesson1_Tukey
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = value ~ treatment, data = lesson1_gather)
## 
## $treatment
##                 diff        lwr         upr     p adj
## F1-Control  7.600000  4.7770648 10.42293521 0.0000016
## F2-Control  4.866667  2.0437315  7.68960188 0.0005509
## F3-Control  8.200000  5.3770648 11.02293521 0.0000005
## F2-F1      -2.733333 -5.5562685  0.08960188 0.0598655
## F3-F1       0.600000 -2.2229352  3.42293521 0.9324380
## F3-F2       3.333333  0.5103981  6.15626854 0.0171033
```

```
library(broom)
tidy(lesson1_Tukey)
```

```
##        term comparison  estimate   conf.low   conf.high  adj.p.value
## 1 treatment F1-Control  7.600000  4.7770648 10.42293521 1.637988e-06
## 2 treatment F2-Control  4.866667  2.0437315  7.68960188 5.509424e-04
## 3 treatment F3-Control  8.200000  5.3770648 11.02293521 5.148374e-07
## 4 treatment      F2-F1 -2.733333 -5.5562685  0.08960188 5.986551e-02
## 5 treatment      F3-F1  0.600000 -2.2229352  3.42293521 9.324380e-01
## 6 treatment      F3-F2  3.333333  0.5103981  6.15626854 1.710330e-02
```

```
plot(lesson1_Tukey)
```



F3-F2 : a and b
F3-F1 : a
F2-F1 : b
F3-control : c
F2-control : c
F1-control : c

```
summary_lesson1 <- lesson1_gather %>%
  group_by(treatment) %>%
  summarise(N = n(), mean = mean(value), sd = sd(value),
            se = sd/sqrt(N), ci = se*qt(0.975,N-1)) %>%
  mutate(labels = c("c", "ab", "b", "a"))

summary_lesson1
```

```
## # A tibble: 4 x 7
##   treatment     N  mean    sd    se    ci labels
##   <chr>     <int> <dbl> <dbl> <dbl> <dbl> <chr>
## 1 Control       6  21.0  1.00 0.408  1.05 c
## 2 F1            6  28.6  2.44 0.995  2.56 ab
```

```
## 3 F2            6  25.9  1.90 0.775  1.99 b
## 4 F3            6  29.2  1.29 0.526  1.35 a
```

Shouldn't R be able to do this work for us?

```
library(multcomp)
library(multcompView)

greenhouse_letters <- multcompLetters4(lesson1_aov, lesson1_Tukey)
greenhouse_letters
```

```
## $treatment
##      F3      F1      F2 Control
##     "a"    "ab"     "b"     "c"
```

```
str(greenhouse_letters)
```

```
## List of 1
##  $ treatment:List of 3
##   ..$ Letters          : Named chr [1:4] "a" "ab" "b" "c"
##   .. ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##   ..$ monospacedLetters: Named chr [1:4] "a  " "ab " " b " "  c"
##   .. ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##   ..$ LetterMatrix     : logi [1:4, 1:3] TRUE TRUE FALSE FALSE FALSE TRUE ...
##   .. ..- attr(*, "dimnames")=List of 2
##   .. .. ..$ : chr [1:4] "F3" "F1" "F2" "Control"
##   .. .. ..$ : chr [1:3] "a" "b" "c"
##   ..- attr(*, "class")= chr "multcompLetters"
```

```
library(purrr)
gh_letters_flatten <- greenhouse_letters %>%
  flatten()
str(gh_letters_flatten)
```

```
## List of 3
##  $ Letters          : Named chr [1:4] "a" "ab" "b" "c"
##   ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##  $ monospacedLetters: Named chr [1:4] "a  " "ab " " b " "  c"
##   ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##  $ LetterMatrix     : logi [1:4, 1:3] TRUE TRUE FALSE FALSE FALSE TRUE ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "F3" "F1" "F2" "Control"
##   .. ..$ : chr [1:3] "a" "b" "c"
```

```
View(as_tibble(gh_letters_flatten$Letters))

gh_letters_pluck <- greenhouse_letters %>%
  pluck(1)
str(gh_letters_pluck)
```

```
## List of 3
##  $ Letters          : Named chr [1:4] "a" "ab" "b" "c"
##   ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##  $ monospacedLetters: Named chr [1:4] "a  " "ab " " b " "  c"
##   ..- attr(*, "names")= chr [1:4] "F3" "F1" "F2" "Control"
##  $ LetterMatrix     : logi [1:4, 1:3] TRUE TRUE FALSE FALSE FALSE TRUE ...
##   ..- attr(*, "dimnames")=List of 2
```

```
##   .. ..$ : chr [1:4] "F3" "F1" "F2" "Control"
##   .. ..$ : chr [1:3] "a" "b" "c"
##  - attr(*, "class")= chr "multcompLetters"
```

```
gh_letters_unlist <- greenhouse_letters %>%
  unlist %>%
  as_tibble()
gh_letters_unlist
```

```
## # A tibble: 20 x 1
##     value
##   * <chr>
##   1 a
##   2 ab
##   3 b
##   4 c
##   5 "a  "
##   6 "ab "
##   7 " b "
##   8 "  c"
##   9 TRUE
##  10 TRUE
##  11 FALSE
##  12 FALSE
##  13 FALSE
##  14 TRUE
##  15 TRUE
##  16 FALSE
##  17 FALSE
##  18 FALSE
##  19 FALSE
##  20 TRUE
```

```
str(gh_letters_unlist)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    20 obs. of  1 variable:
##  $ value: chr  "a" "ab" "b" "c" ...
```

```
gh_letters_row <- as_tibble(gh_letters_flatten$Letters)
gh_letters_row
```

```
## # A tibble: 4 x 1
##   value
## * <chr>
## 1 a
## 2 ab
## 3 b
## 4 c
```

```
letters_final <- as_tibble(names(greenhouse_letters$treatment[["Letters"]]))
letters_final %<>%  rename(treatement = value)
letters_final
```

```
## # A tibble: 4 x 1
##   treatement
##   <chr>
## 1 F3
```

```
## 2 F1
## 3 F2
## 4 Control
```

```
final_final <- bind_cols(letters_final, gh_letters_row)
final_final
```

```
## # A tibble: 4 x 2
##   treatement value
##   <chr>      <chr>
## 1 F3         a
## 2 F1         ab
## 3 F2         b
## 4 Control    c
```

```
# greenhouse1_lm <- lm(value ~ treatment, data = lesson1_gather)
# greenhouse1_lsm <- lsmeans(greenhouse1_lm, ~ treatment)
# greenhouse1_cld <- cld(greenhouse1_lsm,by = NULL, Letters = letters, alpha = .05, reversed = TRUE, me
# greenhouse1_cld
```

7.8 ANOVA Block analysis

```
ecoli <- read_table2("Month  WR01  WR02  WR03  WR04
March   3.  57.6    12  21.3
April   121.    14.6    6.3 39.9
May 307.6   290.9   290.9   435.2
June    44.1    30.1    34.1    81.3
July    108.1   88  14.8    178.2
August  106.70  146.70  98.70   275.50
September   148.30  517.20  185.00  387.30
October 43.2    81.6    53  198.9", col_names = TRUE)

ecoli
```

```
## # A tibble: 8 x 5
##   Month       WR01  WR02   WR03  WR04
##   <chr>      <dbl> <dbl>  <dbl> <dbl>
## 1 March       3.00  57.6  12.0   21.3
## 2 April     121.    14.6   6.30  39.9
## 3 May       308.   291.  291.   435.
## 4 June       44.1   30.1  34.1   81.3
## 5 July      108.    88.0  14.8  178.
## 6 August    107.   147.   98.7  276.
## 7 September 148.   517.  185.   387.
## 8 October    43.2   81.6  53.0  199.
```

7.8.1 Tidy up the data

```
ecoli_tidy <- ecoli %>%
  gather(key = "site", value = "counts", WR01, WR02, WR03, WR04)

ecoli_tidy
```

```
## # A tibble: 32 x 3
##    Month     site  counts
##    <chr>     <chr>  <dbl>
##  1 March     WR01    3.00
##  2 April     WR01  121.
##  3 May       WR01  308.
##  4 June      WR01   44.1
##  5 July      WR01  108.
##  6 August    WR01  107.
##  7 September WR01  148.
##  8 October   WR01   43.2
##  9 March     WR02   57.6
## 10 April     WR02   14.6
## # ... with 22 more rows
```

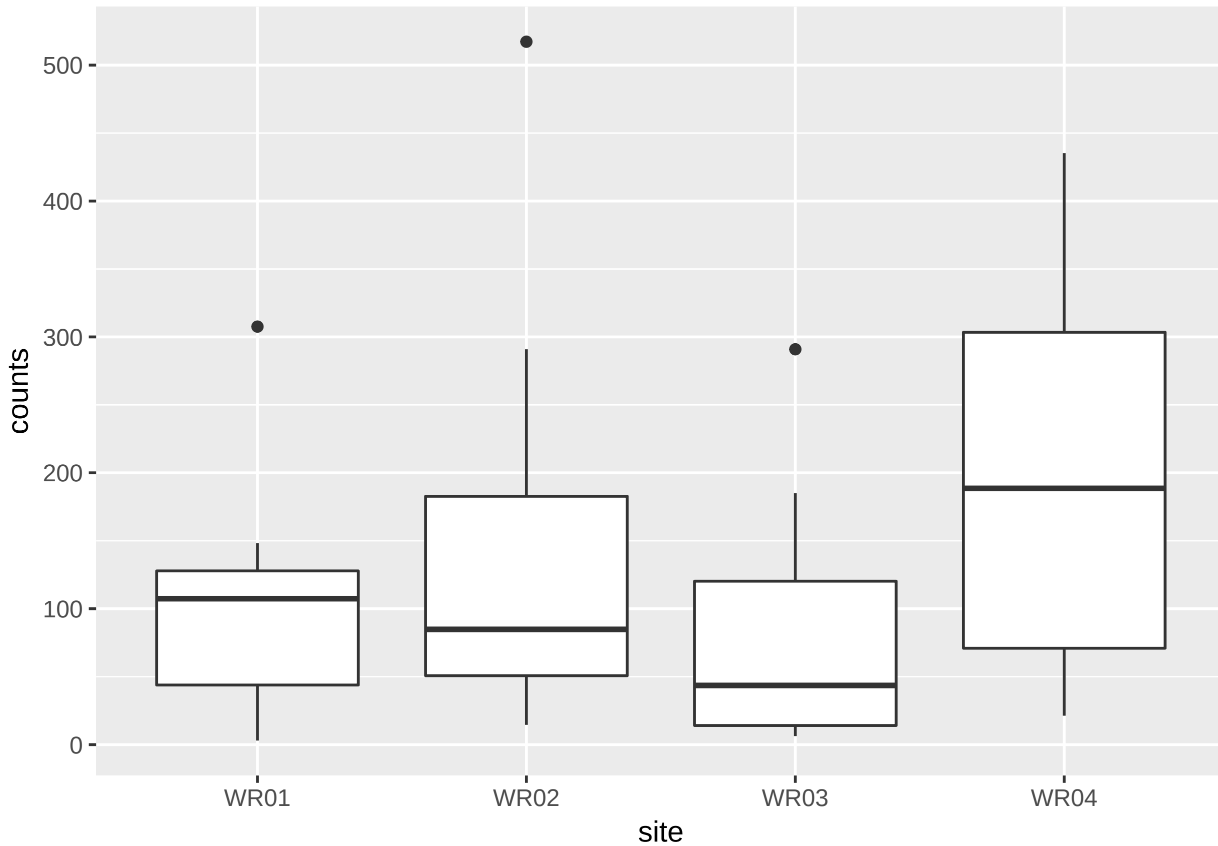
7.8.2 One-way ANOVA

```
ecoli_aov <- aov(counts ~ site, data = ecoli_tidy)

summary(ecoli_aov)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## site         3  61945   20648   1.142  0.349
## Residuals   28 506096   18075
```

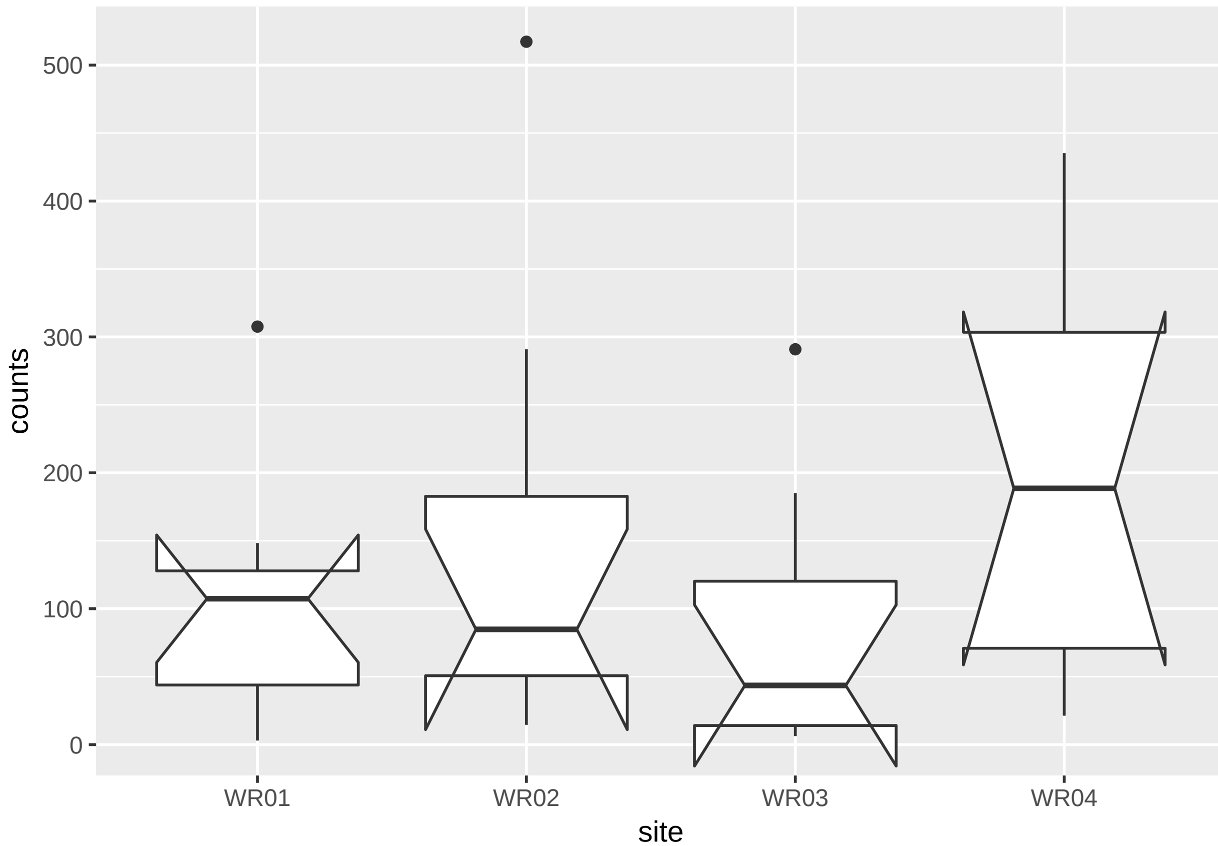
7.8.3 Plot of the data

```
ggplot(ecoli_tidy) +
  geom_boxplot(aes(site, counts))
```

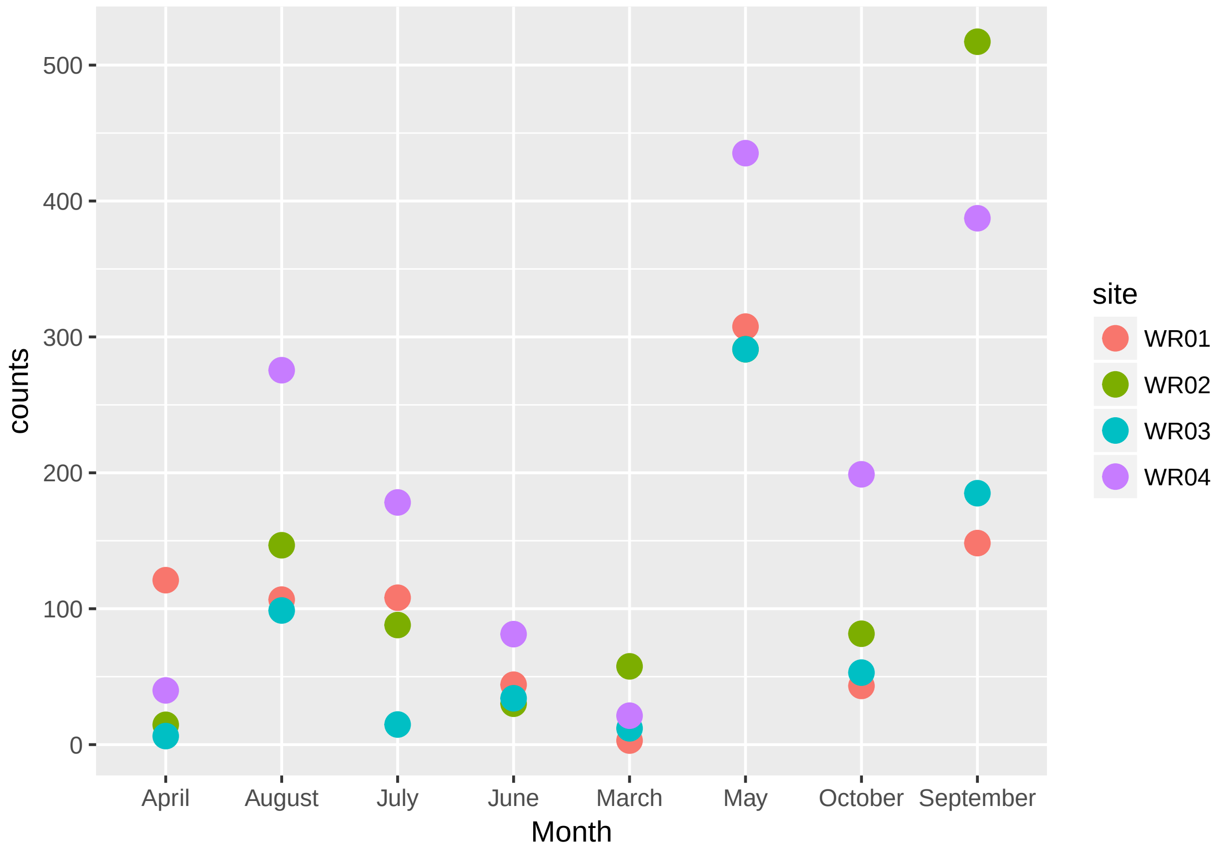



```
ggplot(ecoli_tidy) +
  geom_boxplot(aes(site, counts), notch = TRUE)
```

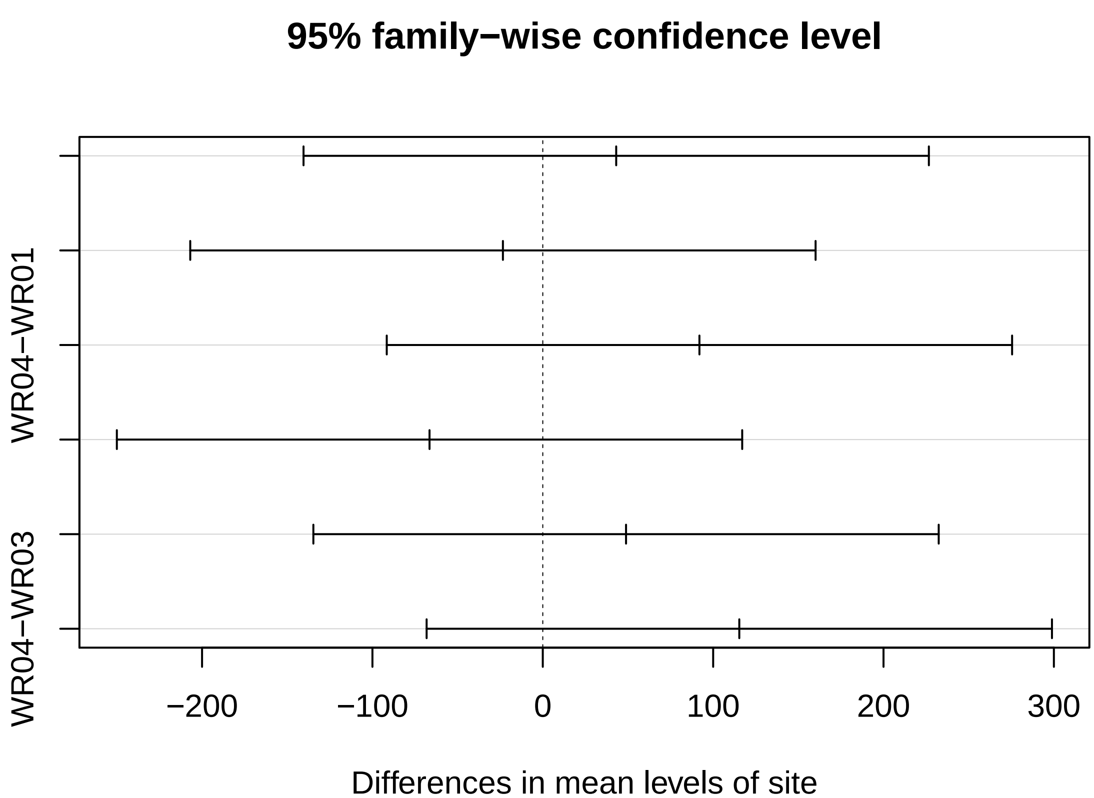
```
## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
## notch went outside hinges. Try setting notch=FALSE.
```




```r
ggplot(ecoli_tidy) +
  geom_point(aes(Month, counts, colour = site), size = 4)
```




```
plot(TukeyHSD(ecoli_aov, conf.level = 0.95))
```



7.8.4 ANOVA with blocking factor

```
ecoli_aov_block <- aov(counts ~ Month + site, data = ecoli_tidy)

summary(ecoli_aov_block)
```

```
##             Df Sum Sq Mean Sq F value  Pr(>F)
## Month        7 402112   57445   11.60 5.7e-06 ***
## site         3  61945   20648    4.17  0.0183 *
## Residuals   21 103984    4952
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

7.8.5 But what is different?

```
ecoli_block_tukey <- TukeyHSD(ecoli_aov_block, conf.level = 0.95)
ecoli_block_tukey
```

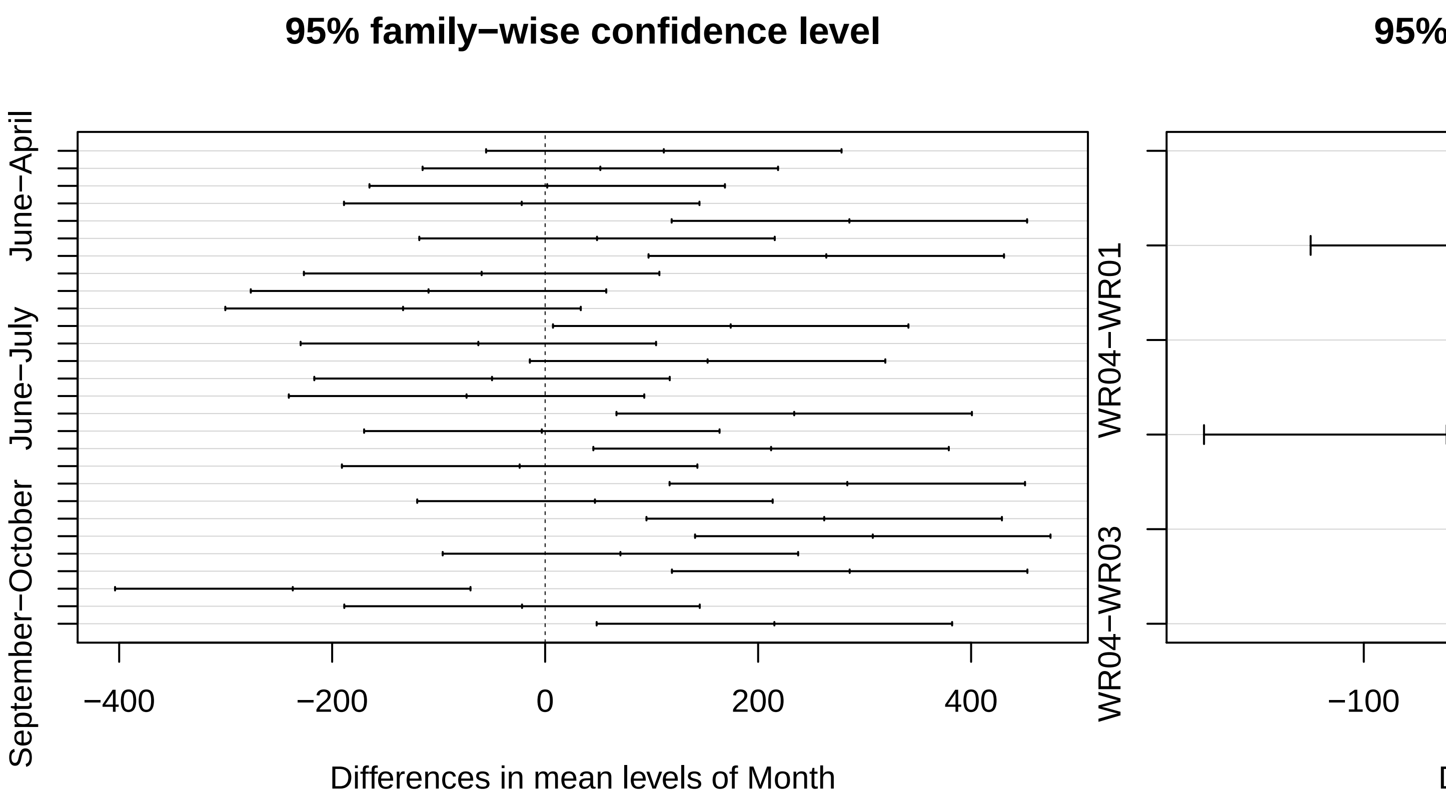
```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = counts ~ Month + site, data = ecoli_tidy)
##
## $Month
##                       diff         lwr       upr     p adj
## August-April       111.450  -55.443795 278.34379 0.3691183
## July-April          51.825 -115.068795 218.71879 0.9620137
## June-April           1.950 -164.943795 168.84379 1.0000000
## March-April        -21.975 -188.868795 144.91879 0.9997950
## May-April          285.700  118.806205 452.59379 0.0002417
## October-April       48.725 -118.168795 215.61879 0.9726099
## September-April    264.000   97.106205 430.89379 0.0006458
## July-August        -59.625 -226.518795 107.26879 0.9235806
## June-August       -109.500 -276.393795  57.39379 0.3899013
## March-August      -133.425 -300.318795  33.46879 0.1827645
## May-August         174.250    7.356205 341.14379 0.0366874
## October-August     -62.725 -229.618795 104.16879 0.9032585
## September-August   152.550  -14.343795 319.44379 0.0894265
## June-July          -49.875 -216.768795 117.01879 0.9689635
## March-July         -73.800 -240.693795  93.09379 0.8076556
## May-July           233.875   66.981205 400.76879 0.0025586
## October-July        -3.100 -169.993795 163.79379 1.0000000
## September-July     212.175   45.281205 379.06879 0.0068760
## March-June         -23.925 -190.818795 142.96879 0.9996407
## May-June           283.750  116.856205 450.64379 0.0002639
## October-June        46.775 -120.118795 213.66879 0.9780722
## September-June     262.050   95.156205 428.94379 0.0007058
## May-March          307.675  140.781205 474.56879 0.0000907
## October-March       70.700  -96.193795 237.59379 0.8378113
## September-March    285.975  119.081205 452.86879 0.0002388
## October-May       -236.975 -403.868795 -70.08121 0.0022204
## September-May      -21.700 -188.593795 145.19379 0.9998114
```

```
## September-October  215.275   48.381205 382.16879 0.0059746
##
## $site
##                diff         lwr       upr     p adj
## WR02-WR01  43.0875  -54.981447 141.15645 0.6186609
## WR03-WR01 -23.4000 -121.468947  74.66895 0.9090016
## WR04-WR01  91.9500   -6.118947 190.01895 0.0712136
## WR03-WR02 -66.4875 -164.556447  31.58145 0.2623702
## WR04-WR02  48.8625  -49.206447 146.93145 0.5197897
## WR04-WR03 115.3500   17.281053 213.41895 0.0174029
```

```
multcompLetters4(ecoli_aov_block, ecoli_block_tukey)
```

```
## $Month
##       May September    August      July   October      June     April
##       "a"      "ab"      "bc"       "c"       "c"       "c"       "c"
##     March
##       "c"
##
## $site
## WR04 WR02 WR01 WR03
##  "a" "ab" "ab"  "b"
```

```
plot(TukeyHSD(ecoli_aov_block, conf.level = 0.95))
```



7.9 Two-way ANOVA with interaction

```
lab_data_2way_anova <- read_table2("46.5    138.4   180.9   39.8    132.4   176.8
                                   47.3 144.4   180.5   40.3    132.4   173.6
                                   46.9 142.7   183 41.2    130.3   174.9", col_names = FALSE)
lab_data_2way_anova
```

```
## # A tibble: 3 x 6
##      X1    X2    X3    X4    X5    X6
##   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1  46.5  138.  181.  39.8  132.  177.
## 2  47.3  144.  180.  40.3  132.  174.
## 3  46.9  143.  183.  41.2  130.  175.
```

```
# I want to stack X1-X3 on top of X4-X6
library(dplyr)

lab_data_method1 <- lab_data_2way_anova %>%
  dplyr::select(X1:X3) %>%
  rename(dose1 = X1, dose2 = X2, dose3 = X3) %>%
  mutate(method = "method1")

lab_data_method1
```

```
## # A tibble: 3 x 4
##   dose1 dose2 dose3 method
##   <dbl> <dbl> <dbl> <chr>
## 1  46.5  138.  181. method1
## 2  47.3  144.  180. method1
## 3  46.9  143.  183. method1
```

```
lab_data_method2 <- lab_data_2way_anova %>%
  dplyr::select(X4:X6) %>%
  rename(dose1 = X4, dose2 = X5, dose3 = X6) %>%
  mutate(method = "method2")

lab_data_method2
```

```
## # A tibble: 3 x 4
##   dose1 dose2 dose3 method
##   <dbl> <dbl> <dbl> <chr>
## 1  39.8  132.  177. method2
## 2  40.3  132.  174. method2
## 3  41.2  130.  175. method2
```

```
lab_data_stack <- bind_rows(lab_data_method1, lab_data_method2)
lab_data_stack
```

```
## # A tibble: 6 x 4
##   dose1 dose2 dose3 method
##   <dbl> <dbl> <dbl> <chr>
## 1  46.5  138.  181. method1
## 2  47.3  144.  180. method1
## 3  46.9  143.  183. method1
## 4  39.8  132.  177. method2
## 5  40.3  132.  174. method2
```

```
## 6  41.2  130.  175. method2
```

```r
lab_data_tidy <- lab_data_stack %>%
  gather(key = doping_level, value = conc, dose1, dose2, dose3)

lab_data_tidy
```

```
## # A tibble: 18 x 3
##    method  doping_level  conc
##    <chr>   <chr>        <dbl>
##  1 method1 dose1         46.5
##  2 method1 dose1         47.3
##  3 method1 dose1         46.9
##  4 method2 dose1         39.8
##  5 method2 dose1         40.3
##  6 method2 dose1         41.2
##  7 method1 dose2        138.
##  8 method1 dose2        144.
##  9 method1 dose2        143.
## 10 method2 dose2        132.
## 11 method2 dose2        132.
## 12 method2 dose2        130.
## 13 method1 dose3        181.
## 14 method1 dose3        180.
## 15 method1 dose3        183.
## 16 method2 dose3        177.
## 17 method2 dose3        174.
## 18 method2 dose3        175.
```

```r
View(lab_data_tidy)
# run the anova

lab_data_aov <- aov(conc ~ method + doping_level, lab_data_tidy)
summary(lab_data_aov)
```

```
##              Df Sum Sq Mean Sq F value   Pr(>F)
## method        1    264     264   80.27 3.58e-07 ***
## doping_level  2  57026   28513 8677.63  < 2e-16 ***
## Residuals    14     46       3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
lab_data_aov_cross <- aov(conc ~ method*doping_level, lab_data_tidy)

summary(lab_data_aov_cross)
```

```
##                     Df Sum Sq Mean Sq   F value   Pr(>F)
## method               1    264     264    98.347 3.92e-07 ***
## doping_level         2  57026   28513 10632.526  < 2e-16 ***
## method:doping_level  2     14       7     2.577    0.117
## Residuals           12     32       3
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Chapter 8

Assessing Product Reliability

This chapter was not covered in the course and may be added at a later date.

Bibliography

Xie, Y. (2015). *Dynamic Documents with R and knitr.* Chapman and Hall/CRC, Boca Raton, Florida, 2nd edition. ISBN 978-1498716963.

Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown.* R package version 0.7.