Data Preparation 15

by **Patrick Boily**, with contributions from **Jen Schellinck**

Once raw data has been collected and stored in a database or a dataset, the focus should shift to data cleaning and processing.

This requires testing for soundness and fixing errors, designing and implementing strategies to deal with missing values and outlying/influential observations, as well as low-level exploratory data analysis and visualization to determine what data transformations and dimension reduction approaches will be needed before embarking on a more sophisticated path.

In this chapter, we establish the essential elements of data cleaning and data processing.

15.1 Introduction

> **Martin K:** Data is messy, Alison.
> **Alison M:** Even after it's been cleaned?
> **Martin K:** Especially after it's been cleaned.
> (P. Boily, J. Schellinck, *The Great Balancing Act* [unpublished]).

Data cleaning and data processing are essential aspects of quantitative analysis projects; analysts and consultants should be prepared to spend up to 80% of their time on data preparation, keeping in mind that:

- processing should **NEVER** be done on the original dataset – make copies along the way;
- **ALL** cleaning steps need to be documented;
- if **too much** of the data requires cleaning up, the data collection procedure might need to be **revisited**, and
- records should only be discarded as a **last resort**.

Another thing to keep in mind is that cleaning and processing may need to take place more than once depending on the type of data collection (one pass, batch, continuously), and that that it is essentially impossible to determine if all data issues have been found and fixed.

Note: in this chapter, we are assuming that the datasets of interest contain only numerical and/or categorical observations. Additional steps must be taken when dealing with unstructured data, such as text or images (we'll have more to say on this topic later).

15.2 General Principles

> **Dilbert:** I didn't have any accurate numbers, so I just made up this one. Studies have shown that accurate numbers aren't any more useful that the ones you make up.
> **Pointy-Haired Boss:** How many studies showed that?
> **Dilbert:** [*beat*] Eighty-seven.
> (S. Adams, Dilbert , 8 May 2008)

15.2.1 Data Cleaning Approaches

We recognize two main **philosophical** approaches to data cleaning and validation:

- **methodical**, and
- **narrative**.

The **methodical** approach consists in running through a **checklist** of potential issues and flagging those that apply to the data.

The **narrative** approach, on the other hand, consists in **exploring** the dataset while searching for unlikely or irregular patterns.

Which approach the consultant/analyst opts to follow depends on a number of factors, not the least of which is the client's needs and views on the matter – it is important to discuss this point with the clients.

15.2.2 Pros and Cons

The methodical approach focuses on **syntax**; the checklist is typically **context-independent**, which means that it (or one of its subsets) can be reused from one project to another – this makes data analysis pipelines **easy to implement** and **automate**. In the same vein, this approach allows for common errors to be **easily identified**.

On the flip side, the checklist may be quite extensive and the entire process may prove **time-consuming**, but the biggest disadvantage of the methodical approach is that it makes it difficult to identify **new types of errors**.

In contrast, the narrative approach focuses on **semantics**; even false starts may simultaneously produce **data understanding** prior to an eventual switch to a more mechanical approach.

It is easy, however, to miss important (and perhaps obvious) sources of errors as well as invalid observations when the datasets have a **large number of features**.

There is an additional downside: **domain expertise**, coupled with the narrative approach, may bias the process by neglecting "uninteresting" areas of the dataset – it takes a special person to spend time on potentially barren lands when they know that greener pastures are available just over yonder.

random missing values	outliers	values outside of expected range - numeric	factors incorrectly/iconsistently coded	date/time values in multiple formats
impossible numeric values	leading or trailing white space	badly formatted date/time values	non-random missing values	logical inconsistencies across fields
characters in numeric field	values outside of expected range - date/time	DCB!	inconsistent or no distinction between null, 0,not available, not applicable,missing	possible factors missing
multiple symbols used for missing values	???	fields incorrectly separated in row	blank fields	logical iconsistencies within field
entire blank rows	character encoding issues	duplicate value in unique field	non-factor values in factor	numeric values in character field

Figure 15.1: Data cleaning bingo card [J. Schellinck].

15.2.3 Tools and Methods

A non-exhaustive list of common data issues can be found in the *Data Cleaning Bingo Card* (see Figure 15.1). Other methods include:

- **visualizations** – which may help easily identify observations that need to be further examined;
- **data summaries** – # of missing observations; 5-pt summary, mean, standard deviation, skew, kurtosis, for numerical variables; distribution tables for categorical variables;
- ***n*-way tables** – counts for joint distributions of categorical variables;
- **small multiples** – tables/visualizations indexed along categorical variables, and
- **preliminary data analyses** – which may provide "huh, that's odd..." realizations.

It is important to note that there is nothing wrong with running a number of analyses to flush out data issues, but remember to label your initial forays as **preliminary** analyses.[1]

Data scientists, dataanalysts, and quantitative consultants alike need to be comfortable with **both** approaches.

As an analogy, the narrative approach is akin to working out a crossword puzzle with a pen and accepting to put down potentially erroneous answers once in a while to try to open up the grid.[2]

The methodical approach, on the other hand, is similar to working out the puzzle with a pencil and a dictionary, only putting down answers when their correctness is guaranteed.[3]

1: From the client or stakeholder's perspective, repeated analyses may create a sense of unease and distrust, even if they form a crucial part of the analytical process.

2: What artificial intelligence researchers call the "exploration" approach.

3: The "exploitation" approach of artificial intelligence.

More puzzles get solved when using the first approach, but missteps tend to be spectacular. Not as many puzzles get solved the second way, but the trade-off is that it leads to fewer mistakes.

15.3 Data Quality

> **Calvin's Dad:** OK Calvin. Let's check over your math homework.
> **Calvin:** Let's not and say we did.
> **Calvin's Dad:** Your teacher says you need to spend more time on it. Have a seat.
> **Calvin:** More time?! I already spent 10 whole minutes on it! 10 minutes shot! Wasted! Down the drain!
> **Calvin's Dad:** You've written here 8 + 4 = 7. Now you know that's not right.
> **Calvin:** So I was off a little bit. Sue me.
> **Calvin's Dad:** You can't **add** things and come with **less** than you started with!
> **Calvin:** I can do that! It's a free country! I've got my rights!
> (B. Watterson, *Calvin and Hobbes*, 15-09-1990.)

The quality of the data has an important effect on the quality of the results: as the saying goes: "garbage in, garbage out."

Data is said to be **sound** when it has as few issues as possible with:

- **validity** – are observations sensible, given data type, range, mandatory response, uniqueness, value, regular expressions, etc. (e.g. a value that is expected to be text value is a number, a value that is expected to be positive is negative, etc.)?;
- **completeness** – are there missing observations (more on this in a subsequent section)?;
- **accuracy and precision** – are there measurement and/or data entry errors (e.g., an individual has 3 children but only 2 are recorded, etc., see Figure 15.2, linking accuracy to bias and precision to the standard error)?;
- **consistency** – are there conflicting observations (e.g., an individual has no children, but the age of one kid is recorded, etc.)?, and
- **uniformity** – are units used uniformly throughout (e.g., an individual is 6ft tall, whereas another one is 145cm tall)?

Finding an issue with data quality after the analyses are completed is a sure-fire way of losing the stakeholder's or client's trust – check early and often!

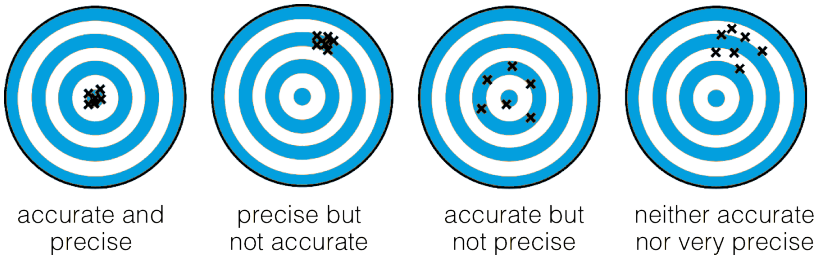


Figure 15.2: Accuracy as bias, precision as standard error [author unknown].

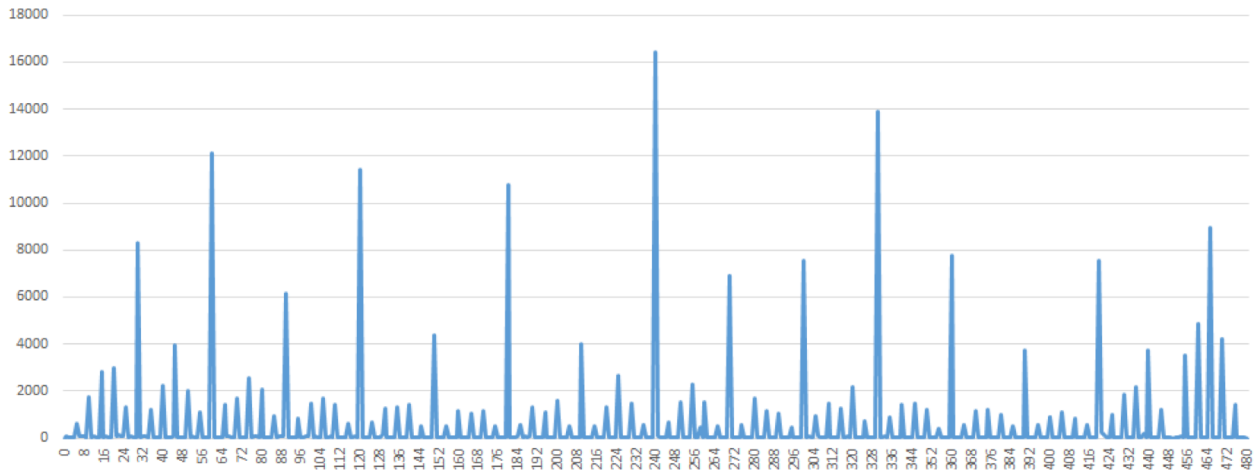


Figure 15.3: An illustration of heaping behaviour: self-reported time spent working in a day [personal file]. The entries for 7, 7.5, and 8 hours are omitted. Note the rounding off at various multiples of 5 minutes.

15.3.1 Common Error Sources

If the analysts have some control over the data collection and initial processing, regular data validation tests are easier to set-up.

When analysts are dealing with **legacy**, **inherited**, or **combined** datasets, however, it can be difficult to recognize errors that arise from:

- missing data being given a code;
- `NA/blank` entries being given a code;
- data entry errors;
- coding errors;
- measurement errors;
- duplicate entries;
- heaping (see Figure 15.3 for an example),
- etc.

15.3.2 Detecting Invalid Entries

Potentially invalid entries can be detected with the help of a number of methods:

- **univariate descriptive statistics** – $z-$score, count, range, mean, median, standard deviation, etc.;
- **multivariate descriptive statistics** – $n-$way tables and logic checks;
- **data visualization** – scatterplot, histogram, joint histogram, etc. (see Chapter 18, *Data Visualization and Data Exploration*, and [2] for more information on the topic),
- and so on.

It is important to point out that univariate tests do not always tell the **whole** story (and may in fact obscure important details).

Example: consider, for instance, an artificial medical dataset consisting of 38 patients' records, containing, among others, fields for the **sex** and the **pregnancy status** of the patients.

A summary of the data of interest is provided in the **frequency counts** (1-way tables) of the table below:

Sex	
Male	19
Female	17
(blank)	2
Total	**38**

Pregnant	
Yes	7
No	27
99	1
(blank)	3
Total	**38**

Analysts can quickly notice that some values are missing (in green) and that an entry has been miscoded as 99 (in yellow). Using only these univariate summaries, however, it is impossible to decide what to do with these invalid entries.

The 2-way frequency counts shed some light on the situation, and uncover other potential issues with the data and/or the data collection protocol.

		Pregnant				Total
		Yes	No	99	(blank)	
Sex	Male	1	17	1	0	**19**
	Female	6	9	0	2	**17**
	(blank)	0	1	0	1	**2**
	Total	**7**	**27**	**1**	**3**	**38**

One of the green entries is actually blank along the two variables; depending on the other information, this entry could be a candidate for **imputation** or outright **deletion** (more on these concepts in the next section).

Three other observations are missing a value along exactly one variable, but the information provided by the other variables may be complete enough to warrant imputation. Of course, if more information is available about the patients, the analyst may be able to determine why the values were missing in the first place (however privacy concerns at the collection stage might muddy the waters).

The mis-coded information on the pregnancy status (99, in yellow) is linked to a male client, and as such re-coding it as 'No' is likely to be a reasonable decision.[4]

A similar reasoning process should make the analyst question the validity of the entry shaded in red – it might very well be correct, but it is important to at least inquire about this data point, as the answer could lead to an eventual re-framing of the definitions and questions used at the collection stage.

In general, there is no universal or one-size-fits-all approach – a lot depends on the **nature of the data**. As always, domain expertise can provide valuable help and suggest fruitful exploration avenues.

4: Although this may **not necessarily be the correct decision**... data measurements are rarely as clear cut as we may think upon only a first reflection.

15.4 Missing Values

> Obviously, the best way to treat missing data is not to have any (T. Orchard, M. Woodbury, [8]).

Why does it matter that some values may be **missing**?

As a start, missing values can potentially introduce **bias** into the analysis, which is rarely (if at all) a good thing, but, more pragmatically, they may interfere with the functioning of most analytical methods, which cannot easily accommodate missing observations without breaking down.[5]

5: As an example, the normal equations $\mathbf{X}^\top\mathbf{X}\beta = \mathbf{X}^\top\mathbf{Y}$ of linear regression could not be solved if some of the data is missing as $\mathbf{X}^\top\mathbf{X}$ is not defined in that case.

Consequently, when faced with missing observations, analysts have two options: they can either **discard** the missing observation (which is not typically recommended, unless the data is missing completely randomly), or they can **create a replacement value** for the missing observation (the **imputation** strategy has drawbacks since we can never be certain that the replacement value is the true value, but is often the best available option; information in this section is taken partly from [5, 9, 12, 10]).

Blank fields come in 4 flavours:

- **nonresponse** – an observation was expected but none was entered;
- **data entry issues** – an observation was recorded but was not entered in the dataset;
- **invalid entries** – an observation was recorded but was considered invalid and has been removed, and
- **expected blanks** – a field has been left blank, but expectedly so.

Too many missing values of the first three types can be indicative of **issues with the data collection process**, while too many missing values of the fourth type can be indicative of **poor questionnaire design** (see Section 10.2 for a brief discussion on these topics).

Either way, missing values cannot simply be **ignored**: either the

- corresponding record is removed from the dataset (not recommended without justification, as doing so may cause a loss of auxiliary information and may bias the analysis results), or
- missing values must be **imputed** (that is to say, a reasonable replacement value must be found).[6]

6: A non-negligible proportion of stakeholders, who would be the first to tell you that they understand nothing about data analysis in the first place, balk at this notion. It helps to remember that we do not generally conduct data analysis to fully understand any particular unit/observation, but rather to get a sense for overall **patterns** in the data (and how we could use these patterns to make predictions for the future, say). We suspect that this is directly linked to the fear that our "personhood" will be erased in favour of numerical summaries, that we are who the data says we are instead of ... well, who we **really** are. Measurement errors happen.

15.4.1 Missing Value Mechanisms

The relevance of an imputation method is dependent on the underlying **missing value mechanism**. Indeed, values may be:

- **missing completely at random** (MCAR) – the item absence is independent of its value or of the unit's auxiliary variables (e.g., an electrical surge randomly deletes an observation in the dataset);
- **missing at random** (MAR) – the item absence is not completely random, and could, in theory, be accounted by the unit's complete auxiliary information, if available (e.g., if women are less likely to tell you their age than men for societal reasons, but not because of the age values themselves), and

- **not missing at random** (NMAR) – the reason for nonresponse is related to the item value itself (e.g., if illicit drug users are less likely to admit to drug use than teetotallers).

The analyst's main challenge in that regard is that the missing mechanism cannot typically be determined with **any degree of certainty**.

15.4.2 Imputation Methods

There are numerous statistical **imputation** methods. They each have their strengths and weaknesses; analysts should take care to select a method which is appropriate for the situation at hand.[7]

7: Imputation methods work best under MCAR or MAR, but keep in mind that they all tend to produce **biased estimates** nonetheless... the hope is the bias is small, and that the benefits of obtaining an estimate in the first place overcomes the presence of bias.

- In **list-wise deletion**, all units with at least one missing value are removed from the dataset. This straightforward imputation strategy assumes MCAR, but it can introduce bias if MCAR does not hold, and it leads to a reduction in the sample size and an increase in standard errors.
- In **mean** or **most frequent imputation**, the missing values are substituted by the average or most frequent value in the unit's subpopulation group (stratum). This commonly-used approach also assumes MCAR, but it can create distortions in the underlying distributions (such as a spike at the mean) and create spurious relationships among variables.
- In **regression** or **correlation imputation**, the missing values are substituted using a regression on the other variables. This model assumes MAR and trains the regression on units with complete information, in order to take full advantage of the auxiliary information when it is available. However, it artificially reduces data variability and produces over-estimates of correlations.
- In **stochastic regression imputation**, the regression estimates are augmented with random error terms added. Just as in regression estimation, the model assumes MAR; an added benefit is that it tends to produce estimates that "look" more realistic than regression imputation, but it comes with an increased risk of type I error (false positives) due to small standard errors.
- **Last observation carried forward** (LOCF) and its cousin **next observation carried backward** (NOCB) are useful for longitudinal data; a missing value can simply be substituted by the previous or next value. LOCF and NOCB can be used when the values do not vary greatly from one observation to the next, and when values are MCAR. Their main drawback is that they may be too "generous" for studies that are trying to determine the effect of a treatment over time, say.
- Finally, in k**-nearest-neighbour imputation**, a missing entry in a MAR scenario is substituted by the average (or median, or mode) value from the subgroup of the k most similar complete respondents. This requires a notion of **similarity** between units (which is not always easy to define reasonably). The choice of k is somewhat arbitrary and can affect the imputation, potentially distorting the data structure when it is too large.

What does imputation look like in practice? Consider the following scenario (which is, somewhat embarrassingly, based on a true story).

Example: after marking the final exams of the 211 students who did not drop her course in *Advanced Retroencabulation* at State University, Dr. Helga Vanderwhede creates a data frame `grades` of final exam grades and mid term-grades.

Setting up the grades data frame

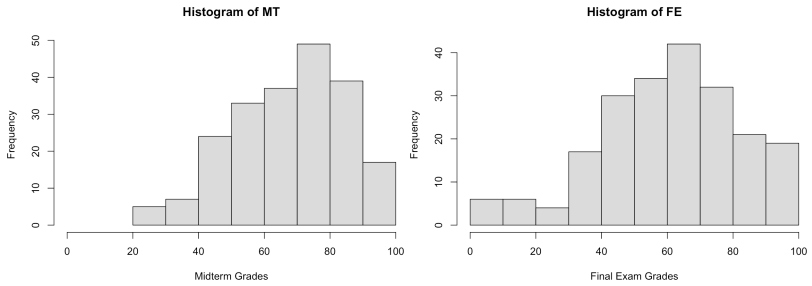
```
MT = c(
80,73,83,60,49,96,87,87,60,53,66,83,32,80,66,90,72,55,76,
46,48,69,45,48,77,52,59,97,76,89,73,73,48,59,55,76,87,55,
80,90,83,66,80,97,80,55,94,73,49,32,76,57,42,94,80,90,90,
62,85,87,97,50,73,77,66,35,66,76,90,73,80,70,73,94,59,52,
81,90,55,73,76,90,46,66,76,69,76,80,42,66,83,80,46,55,80,
76,94,69,57,55,66,46,87,83,49,82,93,47,59,68,65,66,69,76,
38,99,61,46,73,90,66,100,83,48,97,69,62,80,66,55,28,83,59,
48,61,87,72,46,94,48,59,69,97,83,80,66,76,25,55,69,76,38,
21,87,52,90,62,73,73,89,25,94,27,66,66,76,90,83,52,52,83,
66,48,62,80,35,59,72,97,69,62,90,48,83,55,58,66,100,82,78,
62,73,55,84,83,66,49,76,73,54,55,87,50,73,54,52,62,36,87,
80,80
)

FE = c(
41,54,93,49,92,85,37,92,61,42,74,84,61,21,75,49,36,62,92,
85,50,90,52,63,64,85,66,51,41,75,4,46,38,71,42,18,76,42,
94,53,77,65,95,3,74,0,97,62,74,61,80,47,39,92,59,37,59,71,
20,67,69,88,53,52,81,41,81,48,67,65,92,75,68,55,67,51,83,
71,58,37,65,66,51,43,83,34,55,59,20,62,22,70,64,59,73,74,
73,53,44,36,62,45,80,85,41,80,84,44,73,72,60,65,78,60,34,
91,40,41,54,91,49,92,85,37,92,61,42,74,84,61,21,75,49,36,
62,92,85,50,92,52,63,64,85,66,51,41,75,4,46,38,71,42,18,
76,42,92,53,77,65,92,3,74,0,52,62,74,61,80,47,39,92,59,37,
59,71,20,67,69,88,53,52,81,41,81,48,67,65,94,75,68,55,67,
51,83,71,58,37,65,66,51,43,83,34,55,59,20,62,22,70,64,59
)

grades=data.frame(MT,FE)
summary(grades)
```

MT	FE
Min.: 21.0	Min.: 0.00
1st Qu.: 55.00	1st Qu.: 56.50
Median: 70.00	Median: 62.00
Mean: 68.74	Mean: 60.09
3rd Qu.: 82.50	3rd Qu.: 75.00
Max.: 100.00	Max.: 97.00

She plots the final exam grades (y) against the mid-term exam grades (x), as seen below.

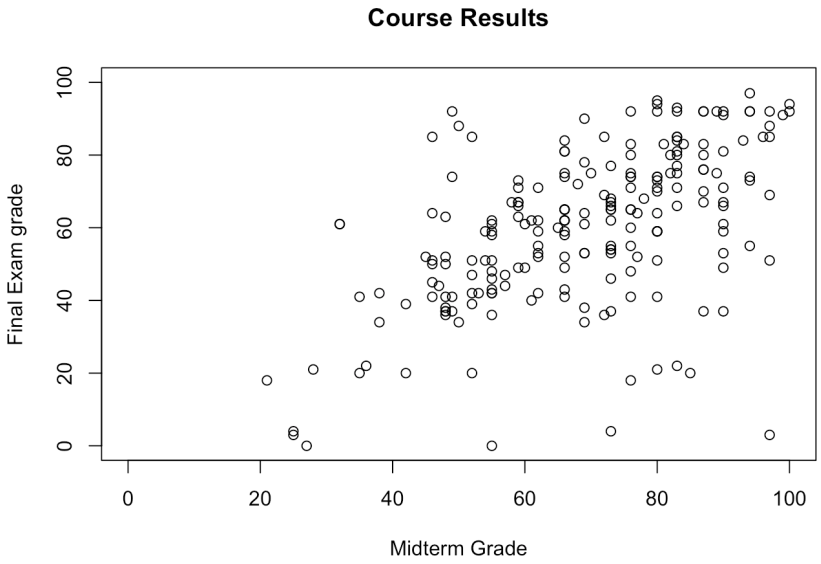
Plotting the grades data frame I

```
hist(MT, xlim=c(0,100), xlab=c("Midterm Grades"))
hist(FE, xlim=c(0,100), xlab=c("Final Exam Grades"))
```



Plotting the grades data frame II

```
plot(grades, xlim=c(0,100), ylim=c(0,100),
     xlab=c("Midterm Grade"), ylab=c("Final Exam grade"),
     main=c("Course Results"))
```



Looking at the data, she sees that final exam grades are **weakly correlated** with mid-term exam grades: students who performed well on the mid-term tended to perform well on the final, and students who performed poorly on the mid-term tended to perform poorly on the final (as is usually the case), but the link is not that strong.

Correlation between mid-term and final grades

```
cor(grades$MT,grades$FE)
```

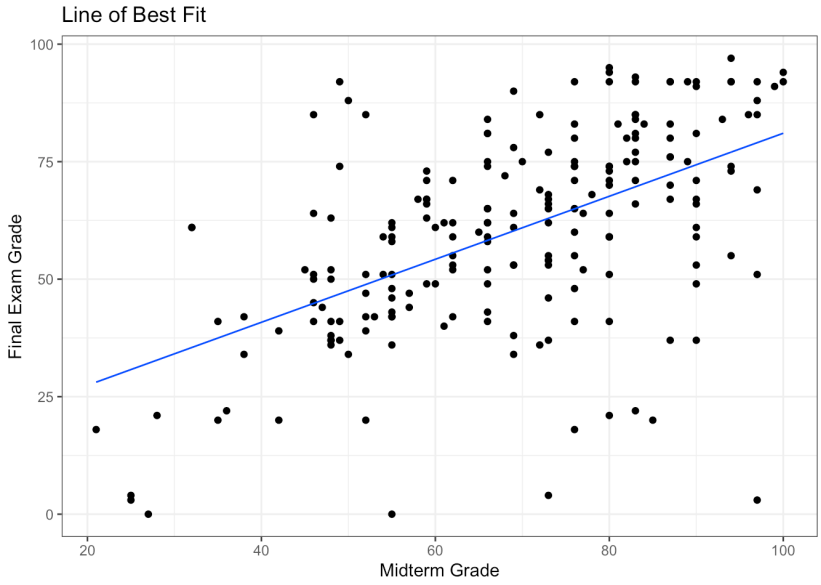
```
[1] 0.5481776
```

She also sees that there is a **fair amount of variability** in the data: the noise is not very tight around the (eye-balled) line of best fit. The linear regression model is:

Line of best fit

```
model <- lm(FE ~ MT, data=grades)
summary(model)

library(ggplot2)
ggplot(model) + geom_point(aes(x=MT, y=FE)) +
    geom_line(aes(x=MT, y=.fitted), color="blue" ) +
    theme_bw() +
    xlab(c("Midterm Grade")) +
    ylab(c("Final Exam Grade")) +
    ggtitle(c("Line of Best Fit"))
```

```
Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 14.00968    5.01523   2.793   0.0057 **
MT           0.67036    0.07075   9.475   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 17.81 on 209 degrees of freedom
Multiple R-squared:  0.3005,    Adjusted R-squared:  0.2972
F-statistic: 89.78 on 1 and 209 DF,  p-value: < 2.2e-16
```



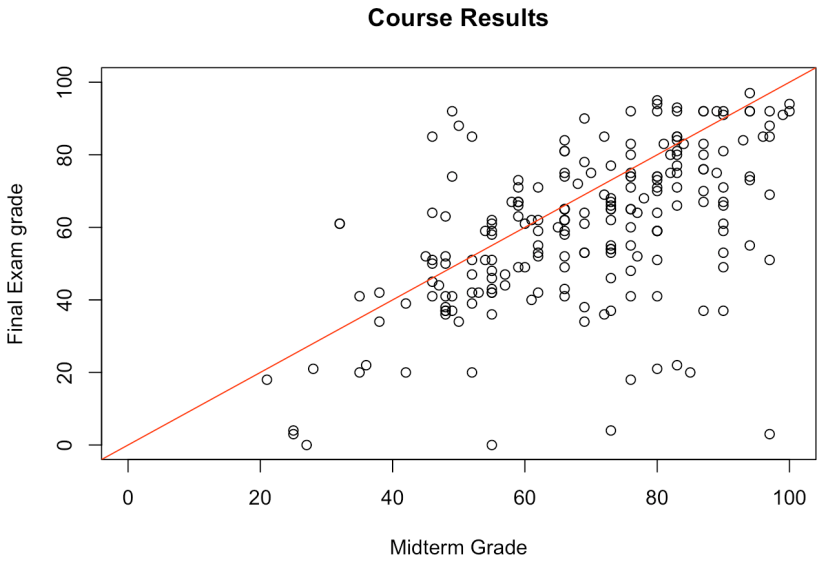
Furthermore, she realizes that the final exam was harder than the students expected (as the slope of the line of best fit is smaller than 1, as only 29% of observations lie above the line MT=FE) – she suspects that they simply did not prepare for the exam seriously,[8] as most of them could not match their mid-term exam performance.

8: And not that she made the exam too difficult, no matter what her ratings on RateMyProfessor.com suggest.

```
sum(grades$MT <= grades$FE)/nrow(grades)
```

```
[1] 0.2890995
```

```
plot(grades, xlim=c(0,100), ylim=c(0,100),
     xlab=c("Midterm Grade"), ylab=c("Final Exam grade"),
     main=c("Course Results"))
abline(a=0, b=1, col="red")
```



As Dr. Vanderwhede comes to terms with her disappointment, she takes a deeper look at the numbers, at some point sorting the dataset according to the mid-term exam grades.

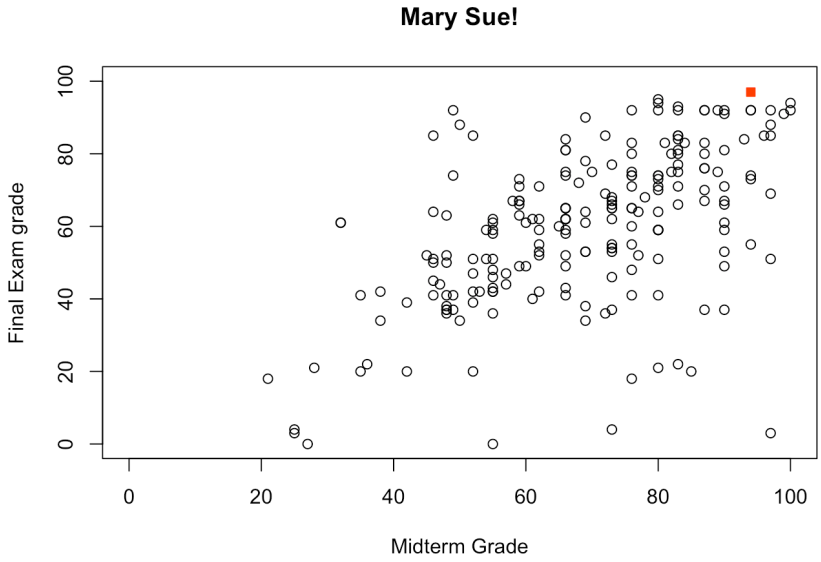
```
s.grades <- grades[order(-grades$MT),]
head(s.grades,16)
```

student	MT	FE
122	100	92
188	100	94
116	99	91
28	97	51
44	97	3
61	97	69
125	97	92
143	97	85
179	97	88
6	96	85
*47	94	97
54	94	92
74	94	55
97	94	73
139	94	92
162	94	74

It looks like good old Mary Sue (row number 47) performed better on the final than on the mid-term (where her performance was already superlative), scoring the highest grade. What a great student she is![9]

9: And such a fantastic person – in spite of her superior intellect, she is adored by all of her classmates, thanks to her sunny disposition and willingness to help at all times. If only all students were like Mary Sue...

```
plot(s.grades[,c("MT","FE")], xlim=c(0,100), ylim=c(0,100),
     col=ifelse(row.names(s.grades)=="47",'red','black'),
     pch=ifelse(row.names(s.grades)=="47",22,1), bg='red',
     xlab=c("Midterm Grade"), ylab=c("Final Exam grade"),
     main=c("Mary Sue!"))
```



She continues to toy with the spreadsheet until the phone rings. After a long and exhausting conversation with Dean Bitterman about teaching loads and State University's reputation, Dr. Vanderwhede returns to the spreadsheet and notices in horror that she has accidentally deleted the final exam grades of all students with a mid-term grade greater than 93.

```
s.grades$FE.NA <- ifelse(s.grades$MT>93,NA,s.grades$FE)
```

What is she to do? Anyone with a modicum of technical savvy would advise her to either undo her changes or to close the file without saving the changes,[10] but in full panic mode, the only solution that comes to her mind is to impute the missing values.

10: Or to simply re-enter the final grades by comparing with the physical papers...

She knows that the missing final grades are MAR (and not MCAR since she remembers sorting the data along the MT values); she produces the imputations shown in Figure 15.4.

List-wise deletion

```
plot(s.grades[,c("MT","FE.NA")], xlim=c(0,100),
     ylim=c(0,100), xlab=c("Midterm Grade"),
     ylab=c("Final Exam grade"),
     main=c("List-wise Deletion"))
```

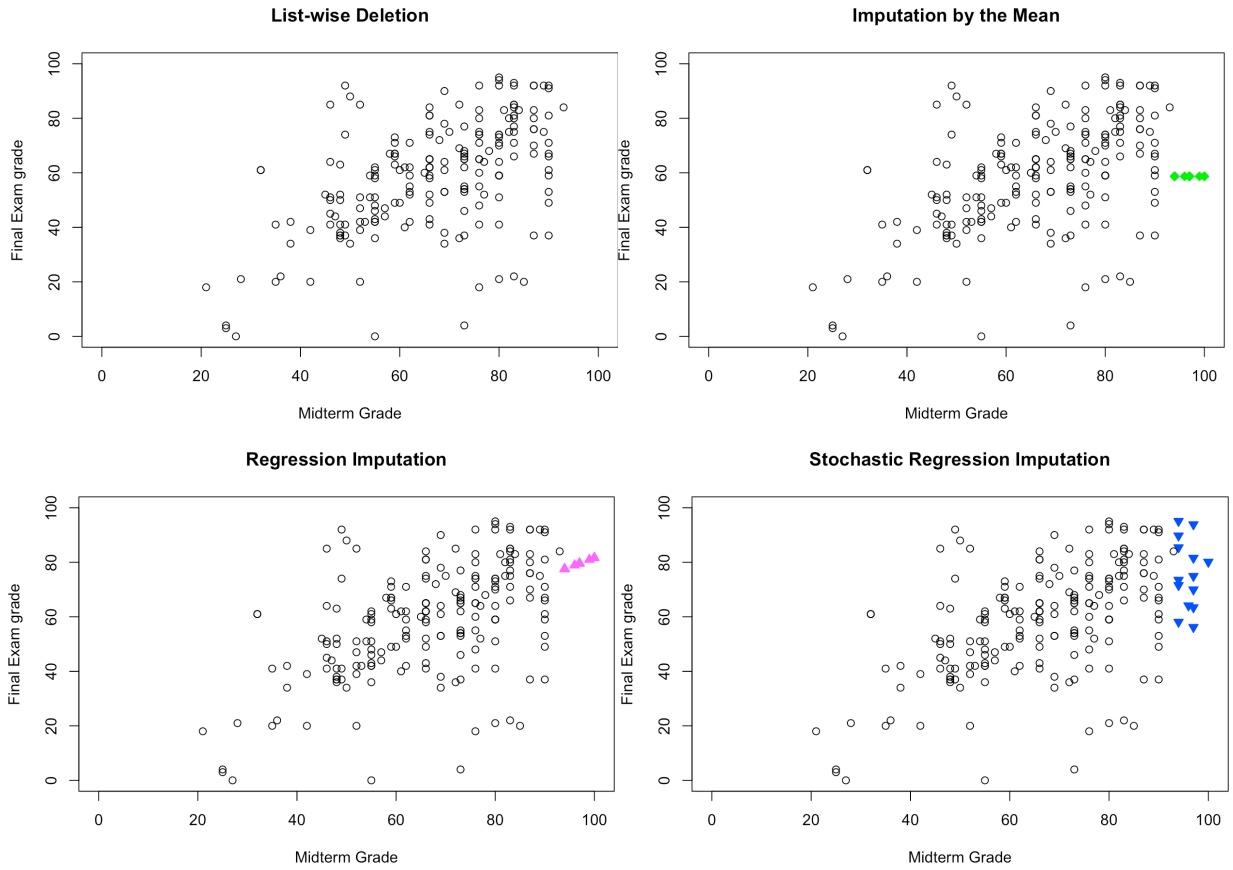


Figure 15.4: Imputation results in the `grades` data frame: listwise deletion (top left), mean imputation (top right), regression imputation (bottom left), stochastic regression imputation (bottom right).

Mean imputation

```
s.mean <- mean(s.grades$FE.NA, na.rm = TRUE)
s.grades$FE.NA.mean <- ifelse(s.grades$MT>93,s.mean,
                              s.grades$FE)
plot(s.grades[,c("MT","FE.NA.mean")], xlim=c(0,100),
     ylim=c(0,100), pch=ifelse(s.grades$MT>93,23,1),
     col=ifelse(s.grades$MT>93,'green','black'),
     bg='green', xlab=c("Midterm Grade"),
     ylab=c("Final Exam grade"),
     main=c("Imputation by the Mean"))
```

Regression imputation

```
model.2 <- lm(FE.NA ~ MT, data=s.grades)
s.grades$FE.NA.reg <- ifelse(s.grades$MT>93,
     model.2[[1]][1]+model.2[[1]][2]*s.grades$MT,
     s.grades$FE)
plot(s.grades[,c("MT","FE.NA.reg")], xlim=c(0,100),
     ylim=c(0,100), pch=ifelse(s.grades$MT>93,24,1),
     col=ifelse(s.grades$MT>93,'magenta','black'),
     bg='magenta', xlab=c("Midterm Grade"),
     ylab=c("Final Exam grade"),
     main=c("Regression Imputation"))
```

Stochastic regression imputation

```
model.3 <- lm(FE.NA ~ MT, data=s.grades)
s.grades$FE.NA.sreg <- ifelse(s.grades$MT>93,
     model.3[[1]][1]+model.3[[1]][2]*s.grades$MT +
     rnorm(nrow(s.grades),0,summary(model.3)$sigma),
     s.grades$FE)
plot(s.grades[,c("MT","FE.NA.sreg")], xlim=c(0,100),
     ylim=c(0,100), pch=ifelse(s.grades$MT>93,25,1),
     col=ifelse(s.grades$MT>93,'blue','black'),
     bg='blue', xlab=c("Midterm Grade"),
     ylab=c("Final Exam grade"),
     main=c("Stochastic Regression Imputation"))
```

She remembers what the data looked like originally, and concludes that the best imputation method is the stochastic regression model.

This conclusion only applies to this specific example, however. In general, that might not be the case due to various *No Free Lunch* results.[11]

11: "There ain't no such thing as a free lunch" – there is no guarantee that a method that works best for a dataset works even reasonably well for another.

The main take-away from this example is that various imputation strategies lead to different outcomes, and perhaps more importantly, that even though the imputed data might "look" like the true data, we have no way to measure its **departure from reality** – any single imputed value is likely to be completely off.

Mathematically, this might not be problematic, as the average departure is likely to be relatively small, but in a business context or a personal one, this might create gigantic problems – how is Mary Sue likely to feel about Dr.Vanderwhede's solution to her conundrum?

```
s.grades[row.names(s.grades) == "47",
         c("MT","FE","FE.NA.reg")]
```

student	MT	FE	FE.NA.reg
*47	94	97	77.54035

And how would Dean Bitterman react were he to find out about the imputation scenario from irate students? The solution has to be compatible with the ultimate data science objective: from Dr. Vanderwhede's perspective, perhaps the only thing that matters is capturing the **essence** of the students' performance, but from the student's perspective, the objective is emphatically different.[12]

12: Analysts cannot simply hide their heads in the sand on this topic: if the data science objectives are incompatible with the units' well-being, it is the objectives that need to change – we cannot ask the entities represented by those units to "get over it".

Even though such questions are not quantitative in nature, their answer will impact any actionable solution.

15.4.3 Multiple Imputation

Another drawback of imputation is that it tends to increase the noise in the data, because the imputed data is treated as the *actual* data.

In **multiple imputation**, the impact of that noise can be reduced by consolidating the analysis outcome from multiple imputed datasets. Once an imputation strategy has been selected on the basis of the (assumed) missing value mechanism,

1. the imputation process is repeated m times to produce m versions of the dataset (assuming a stochastic procedure – if the imputed dataset is always the same, this procedure is worthless);
2. each of these datasets is analyzed, yielding m outcomes, and
3. the m outcomes are pooled into a single result for which the mean, variance, and confidence intervals are known.

On the plus side, multiple imputation is **easy to implement**, **flexible**, as it can be used in a most situations (MCAR, MAR, even NMAR in certain cases), and it accounts for **uncertainty** in the imputed values.

However, m may need to be quite **large** when the values are missing in large quantities from many of the dataset's features, which can substantially slow down the analyses.

There may also be additional technical challenges when the output of the analyses is not a single value but some more complicated object. A generalization of multiple imputation was used by Transport Canada to predict the Blood Alcohol Level (BAC) content level in fatal traffic collisions that involved pedestrians [1].

15.5 Anomalous Observations

> The most exciting phrase to hear [...], the one that heralds the most discoveries, is not "Eureka!" but "That's funny..." [I. Asimov (attributed)].

Outlying observations are data points which are **atypical** in comparison to the unit's remaining features (*within-unit*), or in comparison to the measurements for other units (*between-units*), or as part of a collective subset of observations. Outliers are thus observations which are **dissimilar to other cases** or which contradict **known dependencies**/rules.[13]

13: Outlying observations may be anomalous along any of the individual variables, or in combination.

Note that observations could be anomalous in one context, but not in another. Consider, for instance, an adult male who is 6 feet tall. Such a man would fall in the 86th percentile among Canadian males [6], which, while on the tall side, is not unusual; in Bolivia, however, the same man would land in the 99.9th percentile [6], which would mark him as extremely tall and quite dissimilar to the rest of the population.[14]

14: Anomaly detection points towards interesting questions for analysts and subject matter experts: in this case, why is there such a large discrepancy in the two groups?

A common mistake that analysts make when dealing with outlying observations is to remove them from the dataset without carefully studying whether they are **influential data points**, that is, observations whose absence leads to **markedly different** analysis results.

When influential observations are identified, remedial measures (such as data transformation strategies) may need to be applied to minimize any undue effect. Outliers may be influential, and influential data points may be outliers, but the conditions are neither necessary nor sufficient.

15.5.1 Anomaly Detection

By definition, anomalies are **infrequent** and typically surrounded by **uncertainty** due to their relatively low numbers, which makes it difficult to differentiate them from banal **noise** or **data collection errors**.

Furthermore, the boundary between normal and deviant observations is usually **fuzzy**; with the advent of e-shops, for instance, a purchase which is recorded at 3AM local time does not necessarily raise a red flag anymore.

When anomalies are actually associated to **malicious activities**, they are more than often **disguised** in order to blend in with normal observations, which obviously complicates the detection process.

Numerous methods exist to identify anomalous observations; **none of them are foolproof** and judgement must be used. Methods that employ graphical aids (such as box-plots, scatterplots, scatterplot matrices, and 2D tours) to identify outliers are particularly easy to implement, but a low-dimensional setting is usually required for ease of interpretation.

Analytical detection methods also exist (using Cooke's or Mahalanobis' distances, for instance), but in general some additional level of analysis must be performed, especially when trying to identify influential points (*cf.* **leverage**, Chapter 8, *Classical Regression Analysis*).

With small datasets, anomaly detection can be conducted on a case-by-case basis, but with large datasets, the temptation to use **automated detection/removal** is strong – care must be exercised before the analyst decides to go down that route.[15]

In the early stages of anomaly detection, **simple data analyses** (such as descriptive statistics, 1- and 2-way tables, and traditional visualizations) may be performed to help identify anomalous observations, or to obtain insights about the data, which could eventually lead to modifications of the analysis plan.

15: This stems partly from the fact that once the "anomalous" observations have been removed from the dataset, previously "regular" observations can become anomalous in turn in the smaller dataset; it is not clear when that runaway train will stop.

15.5.2 Outlier Tests

How are outliers *actually* detected? Most methods come in one of two flavours: **supervised** and **unsupervised** (we will discuss those in detail in later sections).

Supervised methods use a historical record of **labeled** (that is to say, previously identified) anomalous observations to build a **predictive classification or regression model** which estimates the probability that a unit is anomalous; domain expertise is required to tag the data.

Since anomalies are typically **infrequent**, these models often also have to accommodate the **rare occurrence problem**.[16]

Unsupervised methods, on the other hand, use no previously labeled information or data, and try to determine if an observation is an outlying one solely by comparing its behaviour to that of the other observations. The following traditional methods and tests of outlier detection fall into this category:[17]

16: Supervised models are built to minimize a cost function; in default settings, it is often the case that the mis-classification cost is assumed to be symmetrical, which can lead to technically correct but useless solutions. For instance, the vast majority (99.999+%) of air passengers emphatically do not bring weapons with them on flights; a model that predicts that no passenger is attempting to smuggle a weapon on board a flight would be 99.999+% accurate, but it would miss the point completely.

17: Note that **normality** of the underlying data is an assumption for most tests; how robust these tests are against departures from this assumption depends on the situation.

- Perhaps the most commonly-used test is **Tukey's boxplot test**; for normally distributed data, regular observations typically lie between the **inner fences**

$$Q_1 - 1.5(Q_3 - Q_1) \quad \text{and} \quad Q_3 + 1.5(Q_3 - Q_1).$$

Suspected outliers lie between the inner fences and their respective **outer fences**

$$Q_1 - 3(Q_3 - Q_1) \quad \text{and} \quad Q_3 + 3(Q_3 - Q_1).$$

Points beyond the outer fences are identified as **outliers** (Q_1 and Q_3 represent the data's 1st and 3rd quartile; see Figure 15.5).

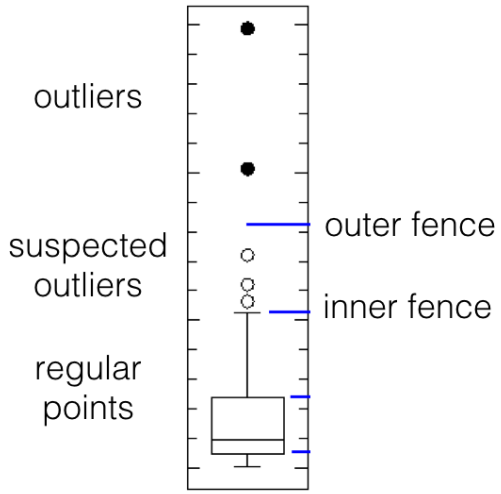
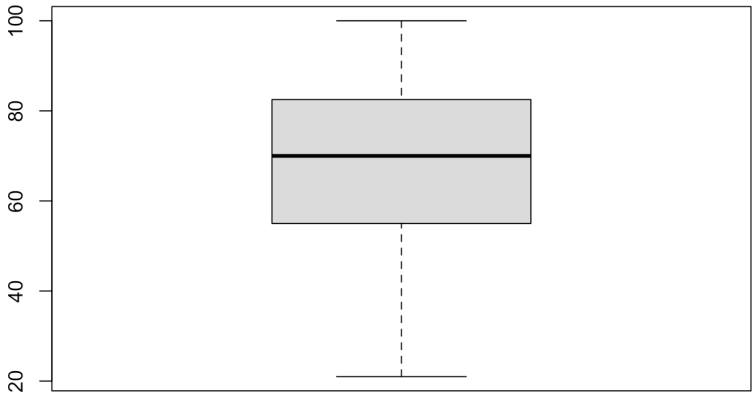


Figure 15.5: Tukey's boxplot test; suspected outliers are marked by white disks, outliers by black disks [author unknown].

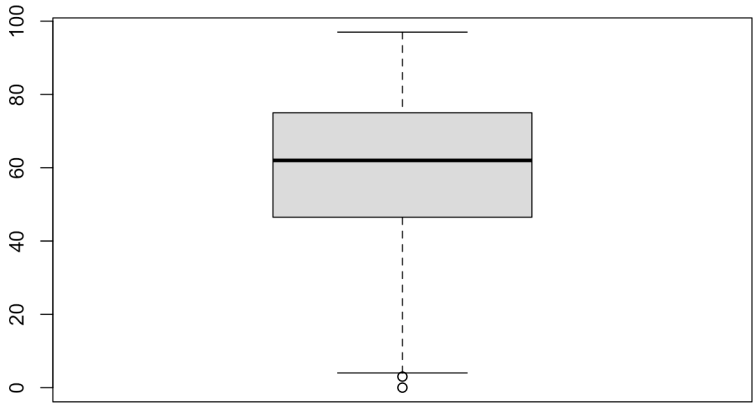
As an example, let's find the outliers for the midterm and final exam grades in Dr. Vanderwhede's *Advanced Retroencabulation* course. There are no boxplot anomalies for midterm grades:

```
boxplot(grades$MT)
```



but there are 4 boxplot anomalies for final exam grades:

```
boxplot(grades$FE)
boxplot.stats(grades$FE)$out
```




```
[1] 3 0 3 0
```

The corresponding observations can be found as follows:

```
out <- boxplot.stats(grades$FE)$out
out_ind <- which(grades$FE %in% c(out))
grades[out_ind,]
```

student	MT	FE	student	MT	FE
44	97	3	161	25	3
46	55	0	163	27	0

- The **Grubbs test** is another univariate test, which takes into consideration the number of observations in the dataset. Let x_i be the value of feature X for the i^{th} unit, $1 \leq i \leq N$, let $(\overline{x}, s_x)$ be the mean and standard deviation of feature X, let α be the desired significance level, and let $T(\alpha, N)$ be the critical value of the Student t-distribution at significance $\alpha/2N$. Then, the i^{th} unit is an **outlier along feature** X if

$$|x_i - \overline{x}| \geq \frac{s_x(N-1)}{\sqrt{N}} \sqrt{\frac{T^2(\alpha, N)}{N - 2 + T^2(\alpha, N)}}.$$

- Other common tests include:
  - the **Mahalanobis distance**, which is linked to the leverage of an observation (a measure of influence), can also be used to find multi-dimensional outliers, when all relationships are linear (or nearly linear);
  - the **Tietjen-Moore** test, which is used to find a specific number of outliers;
  - the **generalized extreme studentized deviate** test, if the number of outliers is unknown;
  - the **chi-square** test, when outliers affect the goodness-of-fit, as well as
  - DBSCAN and other **clustering-based** outlier detection methods.

We will have a lot more to say on the topic in Chapter 26 (*Anomaly Detection and Outlier Analysis*).

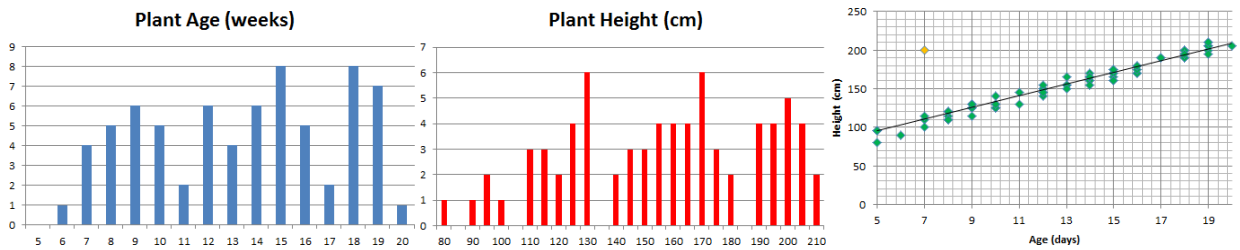


Figure 15.6: Summary visualisations for an artificial plant dataset: age distribution (left), height distribution (middle), height vs. age, with linear trend (right).

15.5.3 Visual Outlier Detection

The following three (simple) examples illustrate the principles underlying visual outlier and anomaly detection.

Example: On a specific day, the height of several plants are measured. The records also show each plant's age (the number of weeks since the seed has been planted).

Histograms of the data are shown in Figure 15.6 (age on the left, height in the middle).

Very little can be said about the data at that stage: the age of the plants (controlled by the nursery staff) seems to be somewhat haphazard, as does the response variable (height). A scatter plot of the data (rightmost chart in Figure 15.6), however, reveals that growth is strongly correlated with age during the early period of a plant's life for the observations in the dataset; points clutter around a linear trend. One point (in yellow) is easily identified as an **outlier**.

There are (at least) two possibilities: either that measurement was botched or mis-entered in the database (representing an invalid entry), or that one specimen has experienced unusual growth (outlier). Either way, the analyst has to investigate further.

Example: a government department has 11 service points in a jurisdiction. Service statistics are recorded: the monthly average arrival rates per teller and average service rates per teller are available for each service point.

A scatter plot of the service rate per teller (y axis) against the arrival rate per teller (x axis), with linear regression trend, is shown in the leftmost chart in Figure 15.7. The trend inches upwards with increasing x values.

A similar chart, but with the left-most point removed from consideration, is shown in the middle chart of Figure 15.7. The trend still slopes upward, but the fit is significantly improved, suggesting that the removed observation is unduly **influential** (or anomalous) – a better understanding of the relationship between arrivals and services is afforded if it is set aside.

Any attempt to fit that data point into the model must take this information into consideration. Note, however, that influential observations depend on the analysis that is ultimately being conducted – a point may be influential for one analysis, but not for another.

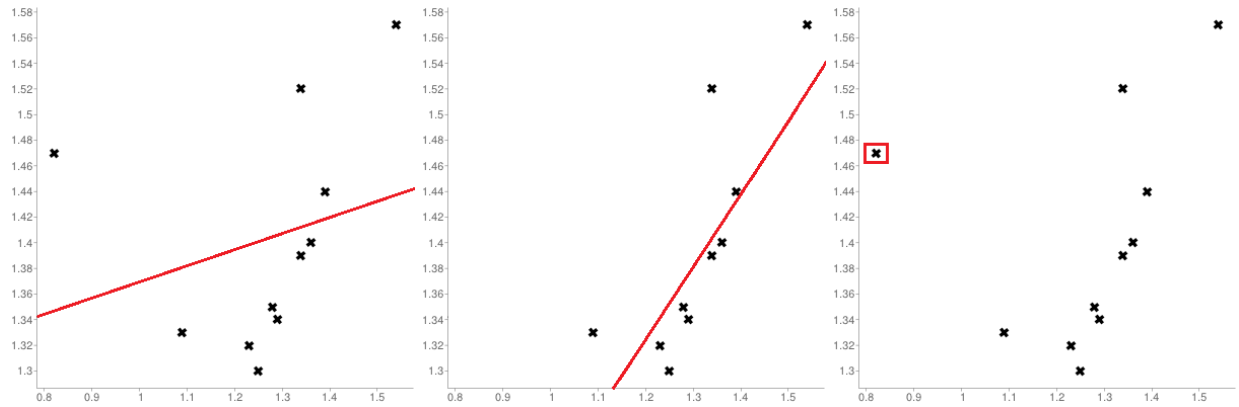


Figure 15.7: Visualisations for an (artificial) service point dataset: trend for 11 service points (left), trend for 10 service points (middle), influential observations (right).

Example: measurements of the length of the appendage of a certain species of insect have been made on 71 individuals. Descriptive statistics have been computed; the results are shown in Table 15.5.

Appendage length (mm)	
Mean	10.35
Standard Deviation	16.98
Kurtosis	16.78
Skewness	4.07
Minimum	0
First Quartile	0
Median	8.77
Third Quartile	10.58
Maximum	88
Range	88
Interquartile Range	10.58
Mode	0
Count	71

Table 15.5: Descriptive statistics for an (artificial) appendage length dataset.

Analysts who are well-versed in statistical methods might recognize the tell-tale signs that the distribution of appendage lengths is likely to be asymmetrical and to have a "fat" tail.[18]

The mode, minimum, and first quartile values belong to individuals without appendages, so there appears to be at least two sub-groups in the population (perhaps split along the lines of juveniles/adults, or males/females).

The maximum value has already been seen to be quite large compared to the rest of the observations, which at first suggests that it might belong to an **outlier**.

The histogram of the measurements, however, shows that there are 3 individuals with very long appendages (see the chart in Figure 15.8): it now becomes plausible for these anomalous entries to belong to individuals from a different species altogether who were **erroneously added** to the dataset. This does not, of course, constitute a proof of such an error, but it raises the possibility, which is often the best that an analyst can do in the absence of subject matter expertise.

18: Since the skewness is non-negligible, and due to the kurtosis being commensurate with the mean and the standard deviation, the range being so much larger than the interquartile range, and the maximum value being so much larger than the third quartile.

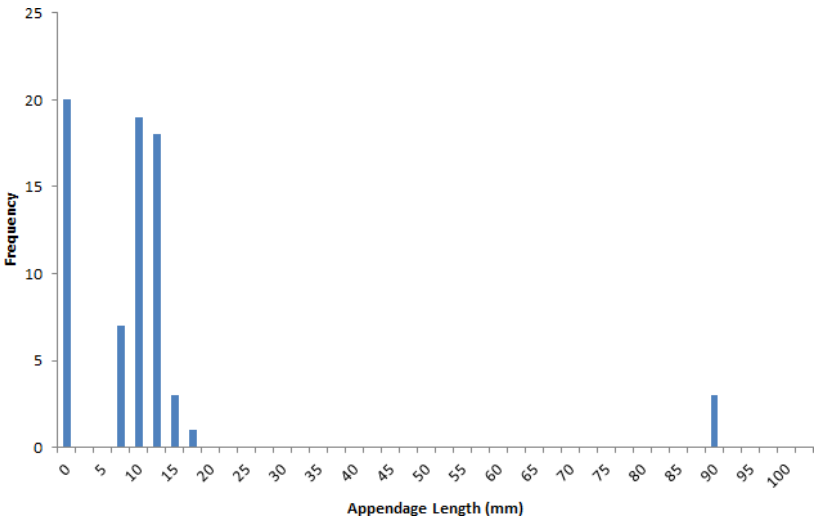


Figure 15.8: Frequency chart of the appendage lengths in the artificial dataset.

15.6 Data Transformations

> History is the transformation of tumultuous conquerors into silent footnotes. [P. Eldridge]

This **crucial** step is often neglected or omitted altogether. Various transformation methods are available, depending on the analysts' needs and data types, including:

- **standardization** and **unit conversion**, which put the dataset's variables on an equal footing – a requirement for basic comparison tasks and more complicated problems of clustering and similarity matching;
- **normalization**, which attempts to force a variable into a normal distribution – an assumption which must be met in order to use number of traditional analysis methods, such as ANOVA or regression analysis, and
- **smoothing methods**, which help remove unwanted noise from the data, but at a price – perhaps removing natural variance in the data.

Another type of data transformation is pre-occupied with the concept of **dimensionality reduction**. There are many advantages to working with low-dimensional data:

- **visualization methods** of all kinds are available to extract and present insights out of such data;
- high-dimensional datasets are subject to the so-called **curse of dimensionality**, which asserts (among other things) that multi-dimensional spaces are vast, and when the number of features in a model increases, the number of observations required to maintain predictive power also increases, but at a **substantially higher rate** (see Figure 15.9),
- another consequence of the curse is that in high-dimension sets, all observations are roughly **dissimilar** to one another – observations tend to be nearer the dataset's boundaries than they are to one another.

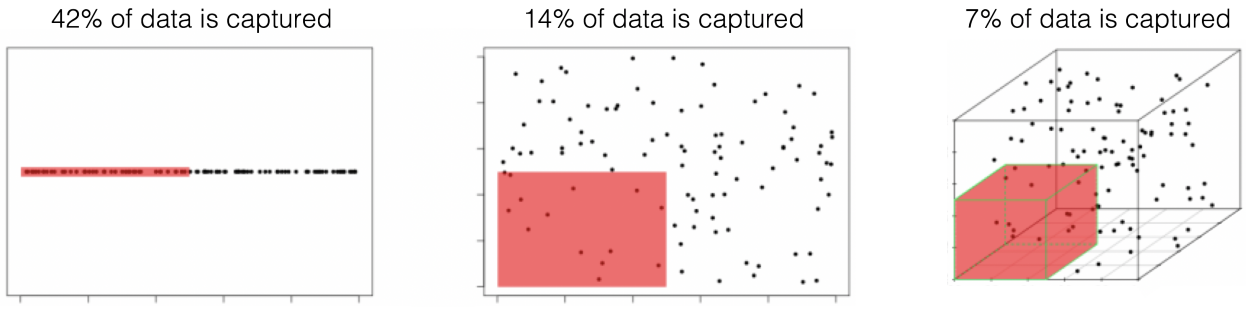


Figure 15.9: Illustration of the curse of dimensionality; $N = 100$ observations are uniformly distributed on the unit hypercube $[0, 1]^d$, $d = 1, 2, 3$. The red regions represent the smaller hypercubes $[0, 0.5]^d$, $d = 1, 2, 3$. The percentage of captured datapoints is seen to decrease with an increase in d [7].

Dimension reduction techniques such as:

- **principal component analysis**, **independent component analysis**, and **factor analysis** for numerical data, or
- **multiple correspondence analysis** for categorical data

project multi-dimensional datasets onto low-dimensional but high information spaces;[19] feature selection techniques, including the popular family of **regularization methods** (see Chapter 20, *Regression and Value Estimation*) select an **optimal subset of variables** with which to accomplish tasks, according to some appropriate, context-dependent criterion.

19: The so-called **Manifold Hypothesis**.

Some information is necessarily lost in the process, but in many instances the drain can be kept under control and the gains made by working with smaller datasets can offset the losses of completeness. We will have more to say on the topic in Chapter 23 (\emph{Feature Selection and Dimension Reduction).

15.6.1 Common Transformations

Models often require that certain data assumptions be met. For instance, ordinary least square regression assumes:

- that the response variable is a **linear combination** of the predictors;
- **constant** error variance;
- **uncorrelated residuals**, which may or may not be statistically independent,
- etc.

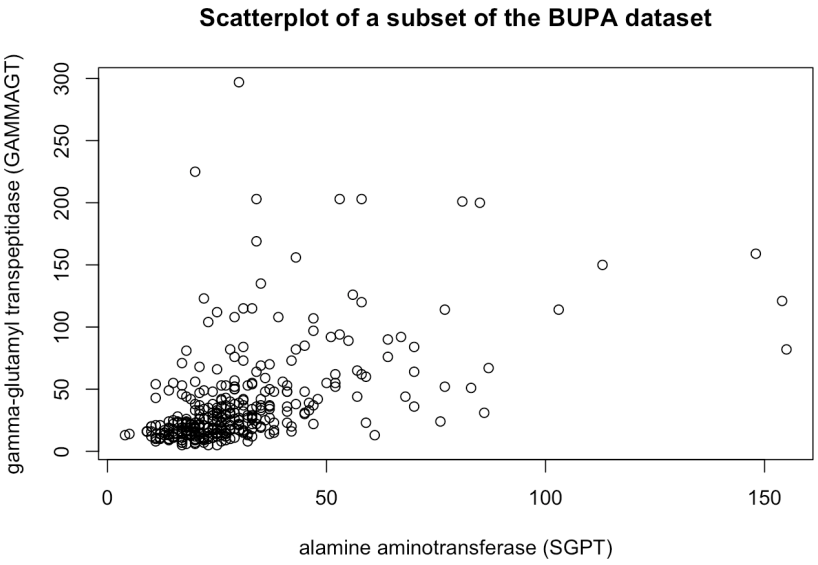
In reality, it is rare that raw data meets all these requirements, but that does not necessarily mean that we need to abandon the model – an **invertible** sequence of data transformations may produce a derived data set which *does* meet the requirements, allowing the consultant to draw conclusions about the original data.

In the regression context, invertibility is guaranteed by **monotonic** transformations: identity, logarithmic, square root, inverse (all members of the power transformations), exponential, etc.

These transformations are illustrated below on a subset of the BUPA **liver disease dataset** [4].

Subset of the BUPA liver disease dataset

```
library(kerndwd)
data(BUPA)
plot(BUPA$X[,3],BUPA$X[,5],
     main="Scatterplot of a subset of the BUPA dataset",
     xlab="alamine aminotransferase (SGPT)",
     ylab="gamma-glutamyl transpeptidase (GAMMAGT)")
```



In Figure 15.10, we show the effect of various transformations on X = SGPT and Y = GAMMAGT.

There are rules of thumb and best practices to transform data, but analysts should not discount the importance of exploring the data visually before making a choice.

Transformations on the **predictors** X may be used to achieve the **linearity assumption**, but they usually come at a price – Pearson correlations are not preserved by such transformations, for instance.[20]

20: Spearman correlations are preserved (in magnitude) by monotonous transformations, however.

Transformations on the target Y can help with **non-normality** of residuals and **non-constant variance** of error terms.

Note that transformations can be applied **both** to the target variable or the predictors: as an example, if the linear relationship between two variables X and Y is expressed as $Y = a + bX$, then a unit increase in X is associated with an average of b units in Y.

But a better fit might be provided by either of

$$\log Y = a + bX, \quad Y = a + b \log X, \quad \text{or} \quad \log Y = a + b \log X,$$

for which:

- a unit increase in X is associated with an average b% increase in Y;
- a 1% increase in X is associated with an average $0.01b$ unit increase in Y, and
- a 1% increase in X is associated with a b% increase in Y, respectively.

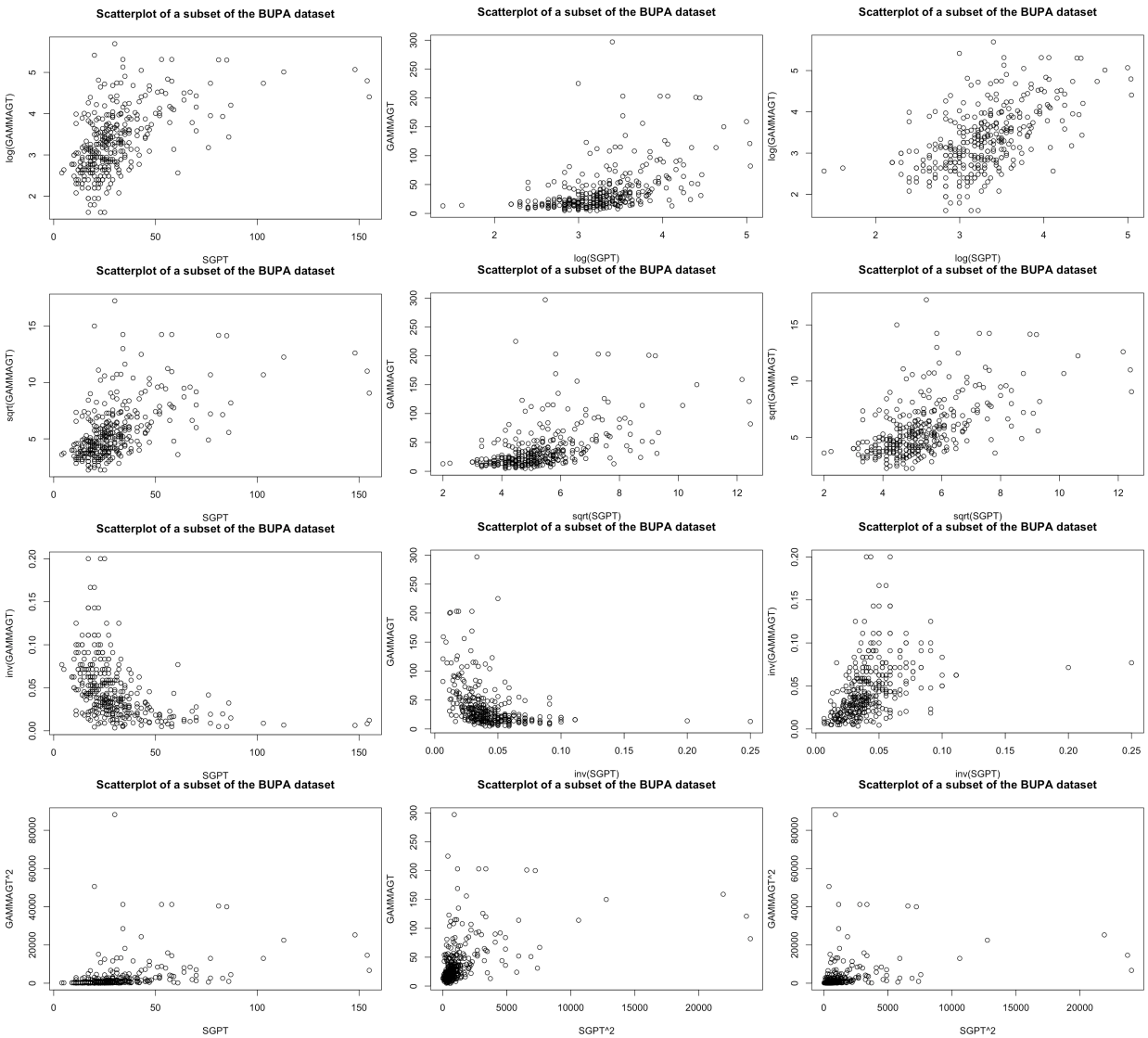


Figure 15.10: Various data transformations for a subset of the BUPA liver disease dataset [4], involving the logarithm, the square root, the inverse, and the square of both variables (see the axes for the specific transformation.

15.6.2 Box-Cox Transformations

There is a useful framework that provides an optimal transformation, in a certain sense. Consider the task of predicting the target Y with the help of the predictors X_j, $j = 1, \ldots, p$. The usual model takes the form

$$y_i = \sum_{j=1}^{p} \beta_j X_{x,i} + \varepsilon_i, \quad i = 1, \ldots, n.$$

If the residuals are skewed, or their variance is not constant, or the trend itself does not appear to be linear, a power transformation on the response might be indicated, but if so, which one? The **Box-Cox transformation** $y_i \mapsto y_i'(\lambda)$, $y_i > 0$ is defined by

$$y_i'(\lambda) = \begin{cases} (y_1 \ldots y_n)^{1/n} \ln y_i, & \text{if } \lambda = 0 \\ \dfrac{y_i^\lambda - 1}{\lambda} (y_1 \ldots y_n)^{\frac{1-\lambda}{n}}, & \text{if } \lambda \neq 0 \end{cases}$$

The **suggested** choice of λ is the value that maximizes the log-likelihood

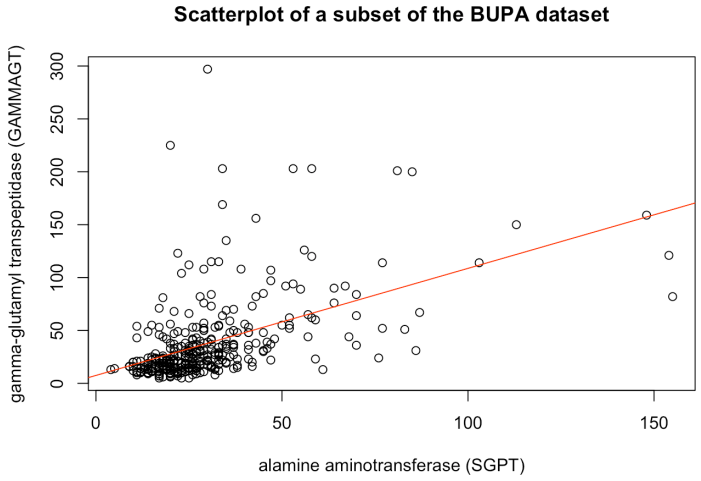
$$\mathscr{L} = -\frac{n}{2}\log\left(\frac{2\pi\hat{\sigma}^2}{(y_1 \dots y_n)^{2(\lambda-1)/n}} + 1\right).$$

The following code shows the effect of the Box-Cox transformation on the linear fit of Y (GAMMAGT) against X (SGPT) in the BUPA dataset.[21]

21: Assume that `library(kerndwd)` and `data(BUPA)` have already been called.

Linear fit in the BUPA liver disease dataset

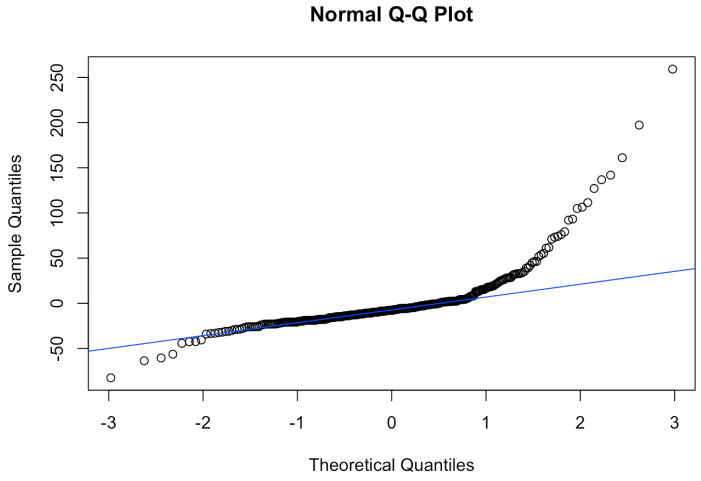
```
model <- lm(BUPA$X[,5] ~ BUPA$X[,3])
plot(BUPA$X[,3],BUPA$X[,5],
     main="Scatterplot of a subset of the BUPA dataset",
     xlab="alamine aminotransferase (SGPT)",
     ylab="gamma-glutamyl transpeptidase (GAMMAGT)")
abline(a=model[[1]][1], b=model[[1]][2], col="red")
```



The fit looks decent, but the $qq-$plot of the residuals makes it clear that the normality assumption of the linear regression model is not met.

QQ plot of the untransformed BUPA model

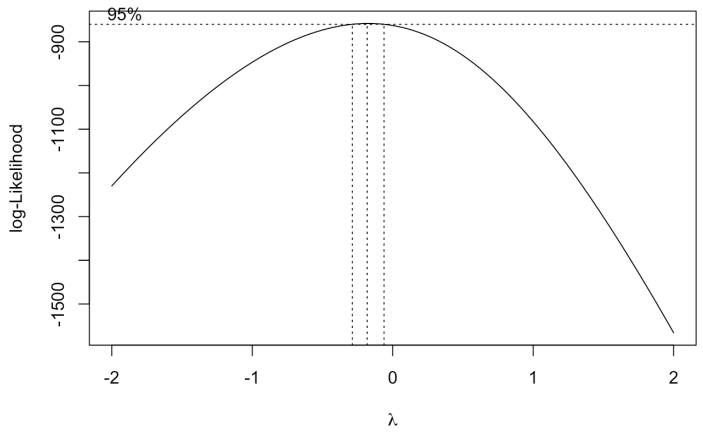
```
qqnorm(model$residuals)
qqline(model$residuals, col="blue")
```



We find the Box-Cox transformation on Y as follows:

Linear fit in the Box-Cos transformed BUPA model

```
library(MASS)
box.cox <- boxcox(BUPA$X[,5] ~ BUPA$X[,3])
(lambda <- box.cox$x[which.max(box.cox$y)])
```

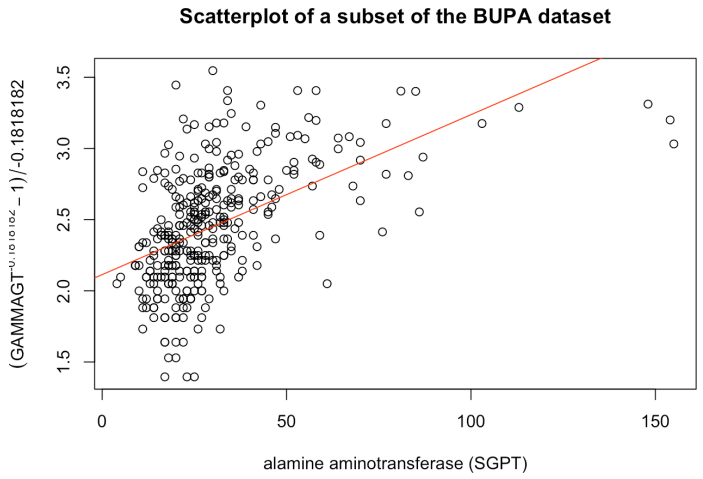



```
[1] -0.1818182
```

The linear model on the Box-Cox transformed response is given as follows.

Linear fit in the Box-Cos transformed BUPA model

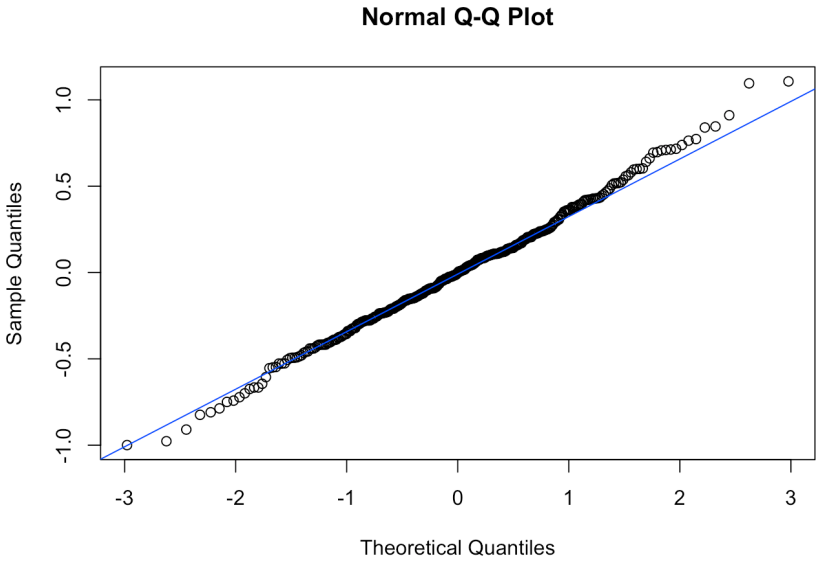
```
box.cox.Y <- (BUPA$X[,5]^lambda-1)/lambda
bc.model <- lm(box.cox.Y ~ BUPA$X[,3])

plot(BUPA$X[,3],box.cox.Y,
     main="Scatterplot of a subset of the BUPA dataset",
     xlab="alamine aminotransferase (SGPT)",
     ylab="Box-Cox response")
abline(a=bc.model[[1]][1], b=bc.model[[1]][2], col="red")
```



That the model on the Box-Cox data is better is evidenced by the $qq-$plot.

QQ plot of the transformed BUPA model

```
qqnorm(bc.model$residuals)
qqline(bc.model$residuals, col="blue")
```



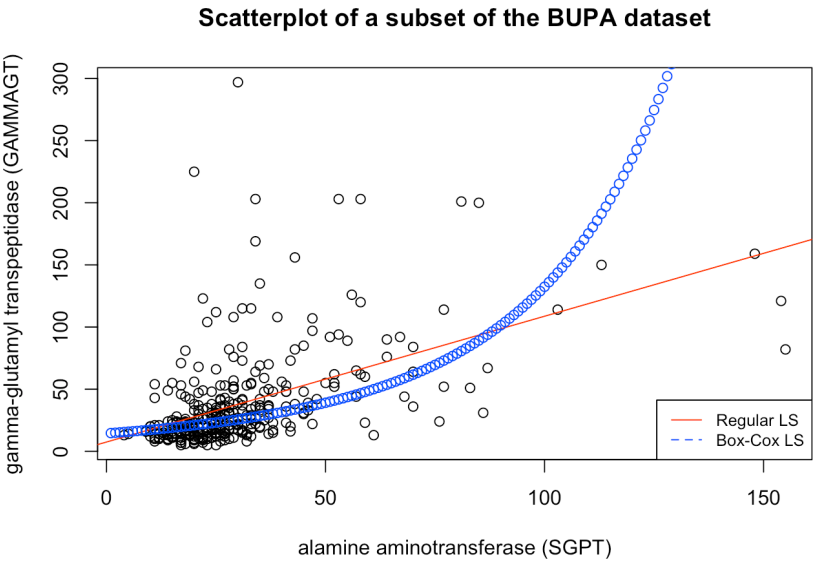
There might be theoretical rationales which favour a particular choice of λ – these are not to be ignored. It is also important to produce a residual analysis, as the best Box-Cox choice does not necessarily meet all the least squares assumptions.

Finally, it is important to remember that the resulting parameters have the least squares property **only with respect to the transformed data points** (in other words, the inverse transformation has to be applied to the results before we can make interpretations about the original data).

In the BUPA example, the corresponding curve in the untransformed space is shown below.

Linear Box-Cox model in the untransformed BUPA data

```
plot(BUPA$X[,3],BUPA$X[,5],
     main="Scatterplot of a subset of the BUPA dataset",
     xlab="alamine aminotransferase (SGPT)",
     ylab="gamma-glutamyl transpeptidase (GAMMAGT)")
df <- data.frame(order(BUPA$X[,3]),
  (lambda * (bc.model[[1]][[1]][1] +
   bc.model[[1]][[2]][1] * order(BUPA$X[,3]))
   + 1)^(1/lambda))
abline(a=model[[1]][1], b=model[[1]][2], col="red")
points(df, col='blue', pch=1)
legend("bottomright", legend=c("Regular LS", "Box-Cox LS"),
       col=c("red", "blue"), lty=1:2, cex=0.8)
```



15.6.3 Scaling

Numeric variables may have different scales (weights and heights, for instance). Since the variance of a large-range variable is typically greater than that of a small-range variable, leaving the data **unscaled** may introduce biases, especially when using unsupervised methods.[22]

22: See Chapter 19, *Machine Learning 101*.

It could also be the case that it is the relative positions (or rankings) which is of importance, in which case it could become important to look at relative distances between levels:

- **standardisation** creates a variable with mean 0 and std deviation 1:

$$Y_i = \frac{X_i - \overline{X}}{s_X},$$

- **normalization** creates a variable in the range $[0, 1]$:

$$Y_i = \frac{X_i - \min\{X_k\}}{\max\{X_k\} - \min\{X_k\}}.$$

There are other options; different schemes can lead to different outputs.

15.6.4 Discretizing

In order to reduce computational complexity, a numeric variable may need to be replaced with an **ordinal** variable (*height* values could be replaced by the qualitative "*short*", "*average*", and "*tall*", for instance.[23]

23: Of course, what these terms represent depend on the context; Canadian short and Bolivian tall may be fairly commensurate, to revisit the example at the start of the preceding section.

It is far from obvious how to determine the bins' limits – **domain expertise** can help, but it could introduce unconscious bias to the analyses. In the absence of such expertise, limits can be set so that either the bins each:

- contain (roughly) the same **number of observations**;
- have the same **width**, or
- the performance of some modeling tool is maximized.

Again, various choices may lead to different outputs.

15.6.5 Creating Variables

Finally, it is possible that new variables may need to be introduced (in contrast with dimensionality reduction). These new variables may arise:

- as **functional relationships** of some subset of available features (introducing powers of a feature, or principal components, say);
- because the modeling tool may require **independence of observations** or **independence of features** (in order to remove multicollinearity, for instance), or
- to simplify the analysis by looking at **aggregated summaries** (often used in text analysis).

There is no limit to the number of new variables that can be added to a dataset – but consultants should strive for **relevant additions**.

15.7 Example: Algae Blooms

This example is based on a Case Study by L. Torgo [11]. It provides a concrete illustration of the data preparation process on a realistic dataset: `algae_blooms.csv`, which is also available at the UCI Machine Learning Repository.

The ultimate problem is to predict the occurrence of harmful algae in water samples. Torgo also uses it to highlight various aspects of **data exploration**, **data cleaning**, and `R` **syntax**.[24]

24: We will continue this work in Section 20.6.

Readers who would prefer to try this example on their own are invited to skip this section and head to the first exercise of Section 15.8.

15.7.1 Problem Description

The ability to monitor and perform early forecasts of various river algae blooms is crucial to control the ecological harm they can cause.

The dataset which is used to train the learning model consists of:

- chemical properties of various water samples of European rivers
- the quantity of seven algae in each of the samples, and
- the characteristics of the collection process for each sample.

What is the data science motivation for such a model? After all, we **can** analyze water samples to determine if various harmful algae are present or absent.

The answer is simple: chemical monitoring is **cheap** and **easy to automate**, whereas biological analysis of samples is **expensive** and **slow**.

Another answer is that analyzing the samples for harmful content does not provide a better understanding of algae **drivers**: it just tells us which samples contain algae.

15.7.2 Loading the Data

Before we can take a look at the data and begin the process in earnest, we need to load it in the in the R workspace. If the dataset was downloaded from the UCI ML repository and stored in the CSV file `algae_blooms.csv`, we can run the following:

```
algae_blooms <- read.csv("algae_blooms.csv", sep=",",
                  stringsAsFactors = TRUE, header=TRUE)
```

It is also available in Torgo's `DMwR` package. As we will use some of its functions in this example, we will show how to install and load it. Unfortunately, it could not be installed directly from CRAN with the `install.packages()` function, as of January 2023.

Instead, we suggest doing the following:

1. Download the package source `DMwR_0.4.1.tar.gz` from the DMwR CRAN archive page (additional information about the package is also available there) and save it locally to some path (in this example, the file was saved to the folder `docs/code/`).
2. Install the following dependencies directly from CRAN:

   ```
   install.packages(c("xts","quantmod","ROCR"))
   ```

   The dependencies list might be different, based on the packages already installed locally; any eventual error message in the next step will inform you of the exact dependencies to install.
3. Install `DMwR` from the package source:

   ```
   install.packages("docs/code/DMwR_0.4.1.tar.gz",
                    repos=NULL, type="source")
   ```

4. Load the package and prepare the data:

   ```
   library(DMwR)
   algae_blooms <- as.data.frame(rbind(DMwR::algae,
                                DMwR::algae.sols))
   ```

Either way, we can get a sense for the data frame's structure by calling the `str` function.

```
str(algae_blooms)
```

```
'data.frame':   340 obs. of  18 variables:
 $ season: Factor w/ 4 levels  "winter" "spring" "autumn" "spring" ...
 $ size  : Factor w/ 3 levels  "small" "small" "small" "small" ...
 $ speed : Factor w/ 3 levels  "medium" "medium" "medium" "medium" ...
 $ mxPH  : num  8 8.35 8.1 8.07 8.06 8.25 8.15 8.05 8.7 7.93 ...
 $ mnO2  : num  9.8 8 11.4 4.8 9 13.1 10.3 10.6 3.4 9.9 ...
 $ Cl    : num  60.8 57.8 40 77.4 55.4 ...
 $ NO3   : num  6.24 1.29 5.33 2.3 10.42 ...
```

```
$ NH4   : num  578 370 346.7 98.2 233.7 ...
$ oPO4  : num  105 428.8 125.7 61.2 58.2 ...
$ PO4   : num  170 558.8 187.1 138.7 97.6 ...
$ Chla  : num  50 1.3 15.6 1.4 10.5 ...
$ a1    : num  0 1.4 3.3 3.1 9.2 15.1 2.4 18.2 25.4 17 ...
$ a2    : num  0 7.6 53.6 41 2.9 14.6 1.2 1.6 5.4 0 ...
$ a3    : num  0 4.8 1.9 18.9 7.5 1.4 3.2 0 2.5 0 ...
$ a4    : num  0 1.9 0 0 0 0 3.9 0 0 2.9 ...
$ a5    : num  34.2 6.7 0 1.4 7.5 22.5 5.8 5.5 0 0 ...
$ a6    : num  8.3 0 0 0 4.1 12.6 6.8 8.7 0 0 ...
$ a7    : num  0 2.1 9.7 1.4 1 2.9 0 0 0 1.7 ...
```

Notes:

- 3 of the fields are categorical (`season`, `size`, `speed`, which refer to the data collection process);
- of the numerical fields, 8 have vaguely "chemical" names;
- presumably, the remaining fields refer to the various algae blooms.

We can get a better feel for the data frame by observing it in its natural habitat, so to speak, using the `head()` or `tail()` functions.

```
head(algae_blooms,4)
```

```
season   size    speed  mxPH   ...   Chla    a1   ...   a7
winter  small   medium  8.00   ...   50.0   0.0  ...   0.0
spring  small   medium  8.35   ...    1.3   1.4  ...   2.1
autumn  small   medium  8.10   ...   15.6   3.3  ...   9.7
spring  small   medium  8.07   ...  138.7   1.4  ...   1.4
```

15.7.3 Summary and Visualization

As it happens, we are not given an awful lot of information about the dataset's **domain**.[25] **Data exploration**, in the form of summaries and visualization, can help provide a handle on the problem at hand.[26]

A call to the `summary` function (on the next page) provides frequency counts for categorical variables, and 6-pt summaries for numerical variables. As a bonus, the number of missing values is also tabulated.[27]

Notes:

- The *chemical* variables all have missing values, ranging from only 2 to 7, 16, and 23.
- The observations seem fairly uniformly distributed in terms of the seasons, but large rivers and low speed rivers are not represented as often as their counterparts.
- All numerical values are non-negative, which makes sense in the context of concentrations
- We do not know what the range of the chemical values *should* take in a real-world context, but some of the maximum values seem . . . unrealistic (NH4!!, oPO4, a7, etc.)
- Does anything else jump at you?

25: We remain woefully ill-prepared to deal with matters of a chemical nature, to our eternal shame.

26: **IMPORTANT NOTE:** we may have given you the impression that exploration is *only really necessary* when domain expertise escapes us. Domain expertise can help analysts frame the problem and the analysis results in the appropriate manner, but it often also gives them a false sense of security. Errors can creep anywhere – data exploration at an early stage may save you a lot of embarrassing back-tracking at a later stage.

27: The default setting only lists a limited number of categorical levels – the `summary` documentation will explain how to increase the number of levels that are displayed.

```
summary(algae_blooms)
```

```
season    autumn spring summer winter
              80     84     86     90

size       large medium  small
              83    136    121

speed     medium   high    low
             140    142     58

       mxPH   mnO2   Cl   NO3     NH4   oPO4    PO4   Chla
Min.:   5.6   1.5   0.2   0.0     5.0    1.0    1.0    0.2
Q1  :   7.8   7.9  10.9   1.1    37.8   13.0   40.0    2.1
Med.:   8.0   9.7  32.4   2.3   107.3   37.2  101.5    5.1
Mean:   7.9   9.1  42.5   3.1   471.7   73.0  136.7   12.7
Q3  :   8.3  10.8  57.7   4.1   244.9   88.1  200.2   17.2
Max.:   9.7  13.4 391.5  45.6 24064.0 1435.0 1690.0  110.4
NA's:   2     2    16     2       2      2      7     23

         a1    a2    a3    a4    a5    a6    a7
Min.:   0.0   0.0   0.0   0.0   0.0   0.0   0.0
Q1  :   1.5   0.0   0.0   0.0   0.0   0.0   0.0
Med.:   7.1   2.8   1.4   0.0   2.2   0.0   0.0
Mean:  16.7   7.2   3.9   1.8   5.5   6.4   2.2
Q3  :  25.1  10.1   4.6   2.3   8.0   7.0   2.2
Max.:  89.8  72.6  42.8  44.6  61.1  77.6  31.6
```

Of course, these summaries each apply to a single variable (1-way tables). Can we find anything else using n-way tables?[28]

28: On categorical variables, by necessity.

2-way tables

```
table(algae_blooms$speed,algae_blooms$size)
table(algae_blooms$speed,algae_blooms$season)
table(algae_blooms$season,algae_blooms$size)
```

```
        large medium small
  high     13     56    73
  low      32     24     2
  medium   38     56    46

        autumn spring summer winter
  high      32     34     38     38
  low       16     13     12     17
  medium    32     37     36     35

        large medium small
  autumn    19     33    28
  spring    21     34    29
  summer    19     36    31
  winter    24     33    33
```

3-way table

```
table(algae_blooms$season,algae_blooms$size,
      algae_blooms$speed)
```

```
, ,  = high
        large medium small
  autumn     3     13    16
  spring     3     14    17
  summer     3     16    19
  winter     4     13    21

, ,  = low
        large medium small
  autumn     9      6     1
  spring     7      6     0
  summer     6      6     0
  winter    10      6     1

, ,  = medium
        large medium small
  autumn     7     14    11
  spring    11     14    12
  summer    10     14    12
  winter    10     14    11
```

The 6-pt summary provides some information about the underlying distribution, but not much on the parametric front. A more traditional summary can be displayed using the `psych` library's `describe()` function.

```
psych::describe(algae_blooms)
```

(the output is shown at the top of the next page)

Notes:

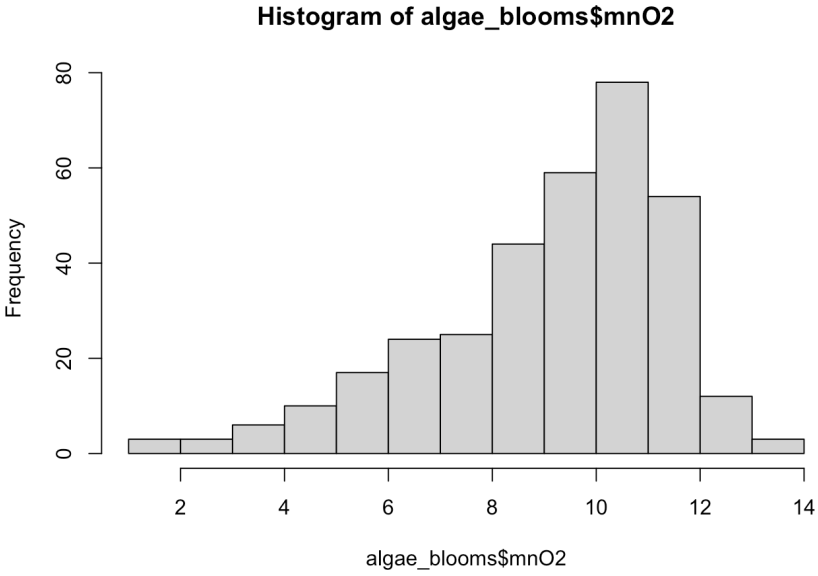
- the categorical variables are marked by an asterisk $*$; the levels are coded with an integer, and treated as numerical variables for the purpose of the analysis, so the results for these fields are meaningless
- the `trimmed` variable refers to the *trimmed mean*, the mean obtained when a certain percentage of the observations are removed from both end of the spectrum (what percentage, exactly?)
- the `mad` variable refers to the *median absolute deviation (from the median)*

We personally find such a table hard to read and really grasp once there are more than a few variables in the dataset. **Visualization** comes in handy in such cases.

Basic histograms can be constructed with the `hist()` function.

	vars	n	mean	sd	median	trimmed	mad	min	max	range	skew	kurtosis	se
season*	1	340	2.547059	1.1186895	3.0000	2.558823	1.482600	1.000	4.000	3.000	-0.0544729	-1.3652641	0.0606695
size*	2	340	2.111765	0.7676208	2.0000	2.139706	1.482600	1.000	3.000	2.000	-0.1915031	-1.2876572	0.0416301
speed*	3	340	1.994118	0.9120437	2.0000	1.992647	1.482600	1.000	3.000	2.000	0.0115387	-1.8012592	0.0494625
mxPH	4	338	7.997293	0.5783188	8.0450	8.035864	0.467019	5.600	9.700	4.100	-0.8402202	1.9865897	0.0314564
mnO2	5	338	9.156716	2.3130799	9.7000	9.388198	1.927380	1.500	13.400	11.900	-0.9392605	0.5497464	0.1258150
Cl	6	324	42.517246	44.4906037	32.4700	34.997591	33.478591	0.222	391.500	391.278	2.8485675	14.0485533	2.4717002
NO3	7	338	3.120784	3.2851622	2.3555	2.699504	2.181646	0.000	45.650	45.650	6.7874378	81.1290663	0.1786893
NH4	8	338	471.734411	1739.0774580	107.3570	156.746838	119.949753	5.000	24064.000	24059.000	9.1184171	105.4980942	94.5933434
oPO4	9	338	73.091882	114.1420517	37.2430	51.792802	43.460936	1.000	1435.000	1434.000	5.9906708	60.0469965	6.2085091
PO4	10	333	136.685699	149.4773125	101.4550	115.867652	114.999352	1.000	1690.000	1689.000	4.2262442	35.1788385	8.1913063
Chla	11	317	12.796196	18.0813363	5.1110	8.782608	6.094969	0.200	110.456	110.256	2.6186753	7.8575379	1.0155490
a1	12	340	16.701765	20.9987076	7.1000	12.593750	10.526460	0.000	89.800	89.800	1.5040883	1.4996118	1.1388148
a2	13	340	7.200882	10.7549412	2.8000	4.854412	4.151280	0.000	72.600	72.600	2.3280170	6.7216331	0.5832686
a3	14	340	3.904412	6.4205247	1.4000	2.358456	2.075640	0.000	42.800	42.800	2.4150758	6.7202844	0.3482018
a4	15	340	1.810000	3.8292948	0.0000	1.036765	0.000000	0.000	44.600	44.600	5.9516431	52.8944003	0.2076727
a5	16	340	5.515588	8.4186630	2.2000	3.692279	3.261720	0.000	61.100	61.100	2.7562479	10.4260075	0.4565661
a6	17	340	6.411471	12.3978237	0.0000	3.279779	0.000000	0.000	77.600	77.600	2.8805737	9.2050442	0.6723664
a7	18	340	2.206471	4.9472217	0.0000	1.040074	0.000000	0.000	31.600	31.600	4.0903606	18.5615144	0.2683008

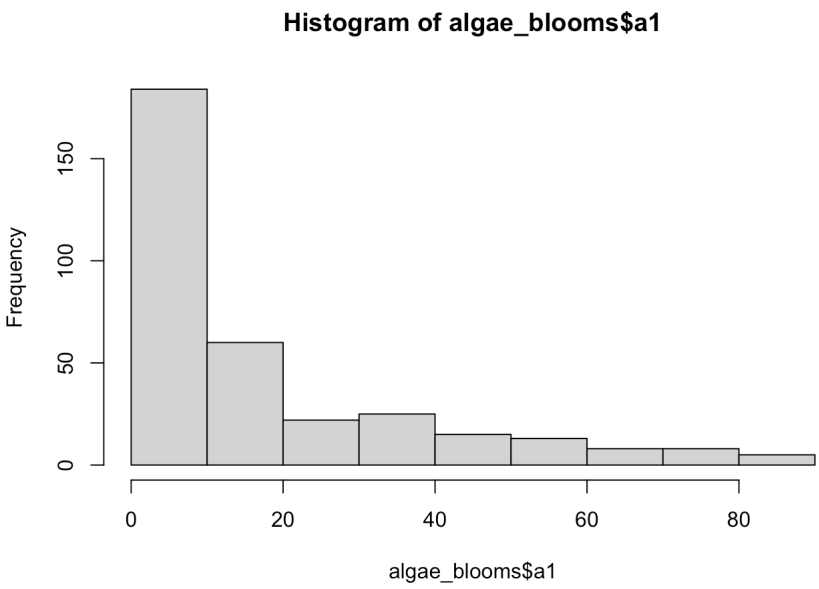
```
hist(algae_blooms$mnO2)
```



Based on this histogram, we can conclude that the underlying distribution of `mnO2` has a **negative skew**, say, which is confirmed by the table above.

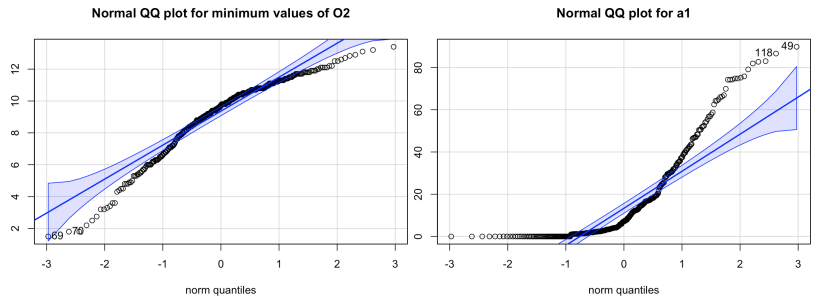
The variable `mnO2` clearly does not follow a normal distribution (it never takes on negative values, and the distribution is skewed negatively, as indications); but we see that viewing it as normal would be a much better approximation than viewing the distribution of `a1` as normal.

```
hist(algae_blooms$a1)
```



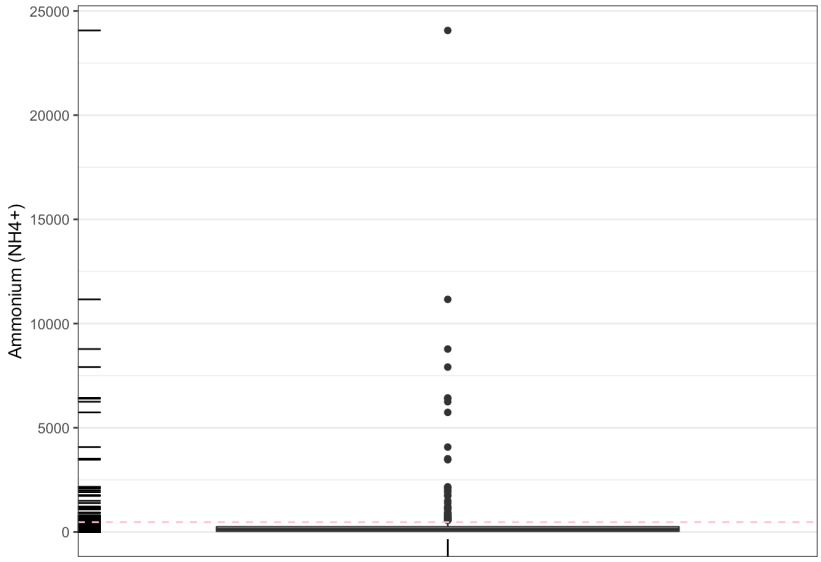
$qq-$plots, another traditional statistical plot, can be produced with the `car` library's `qqPlot()` function. Again, we can see that the normal distribution is not a good fit for `mnO2` (left), but the fit is even worse for `a1` (right).

```
car::qqPlot(algae_blooms$mnO2, ylab="",
   main='Normal QQ plot for minimum values of O2')
car::qqPlot(algae_blooms$a1, ylab="",
   main='Normal QQ plot for a1')
```



We can also take a look at some of the odd values for `NH4` using `ggplot2` [2, 3, 14, 13].

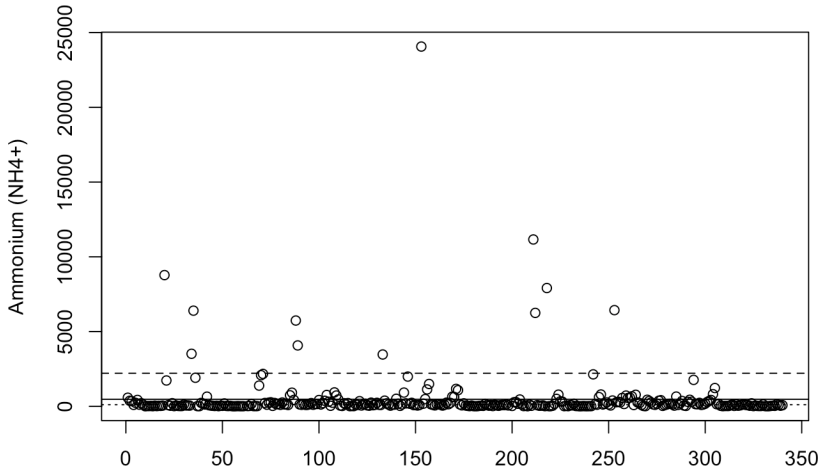
```
library(ggplot2)
ggplot(algae_blooms,aes(x=factor(0),y=NH4)) +
    geom_boxplot() + geom_rug() +
    geom_hline(aes(yintercept=mean(algae_blooms$NH4,
                na.rm=TRUE)), linetype=2, colour="pink") +
    ylab("Ammonium (NH4+)") + xlab("") +
    scale_x_discrete(breaks=NULL)
```



We see that there are a string of values falling way above the boxplot. If the underlying distribution was normal, say, these would definitely be considered outliers.

Let us investigate further.

```
plot(algae_blooms$NH4, xlab="", ylab="Ammonium (NH4+)")
abline(h=mean(algae_blooms$NH4, na.rm=TRUE), lty=1)
abline(h=mean(algae_blooms$NH4, na.rm=TRUE) +
       sd(algae_blooms$NH4, na.rm=TRUE), lty=2)
abline(h=median(algae_blooms$NH4, na.rm=TRUE), lty=3)
```



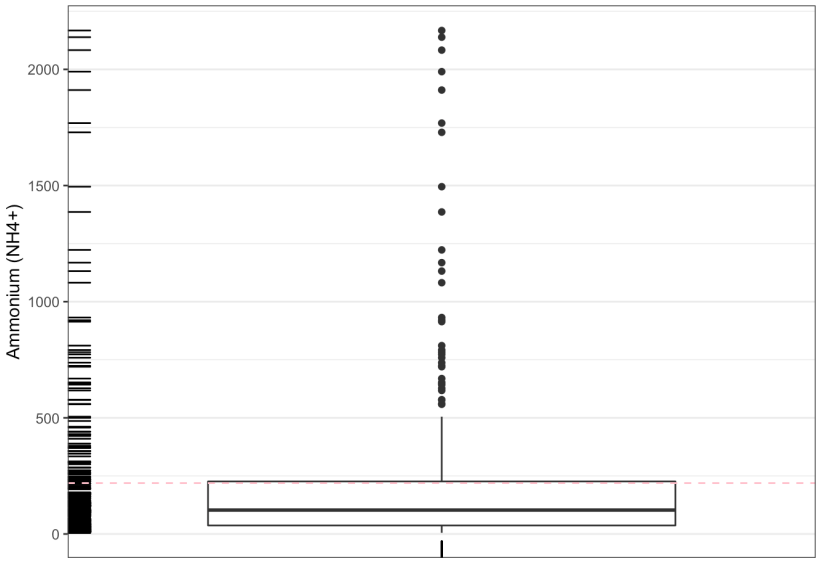
We can also look at the data and see which observations have values of NH4 below 3000 (roughly all values below the long dashed line above).

```
nrow(algae_blooms[-which(algae_blooms$NH4>3000),])
```

```
[1] 329
```

What does the boxplot above looks like without the suspected outliers?

```
ggplot(algae_blooms[-which(algae_blooms$NH4>3000),],
       aes(x=factor(0),y=NH4)) +
    geom_boxplot() + geom_rug() +
    geom_hline(aes(yintercept=mean(algae_blooms[
       -which(algae_blooms$NH4>3000),8], na.rm=TRUE)),
              linetype=2, colour="pink") +
    ylab("Ammonium (NH4+)") + xlab("") +
    scale_x_discrete(breaks=NULL)
```

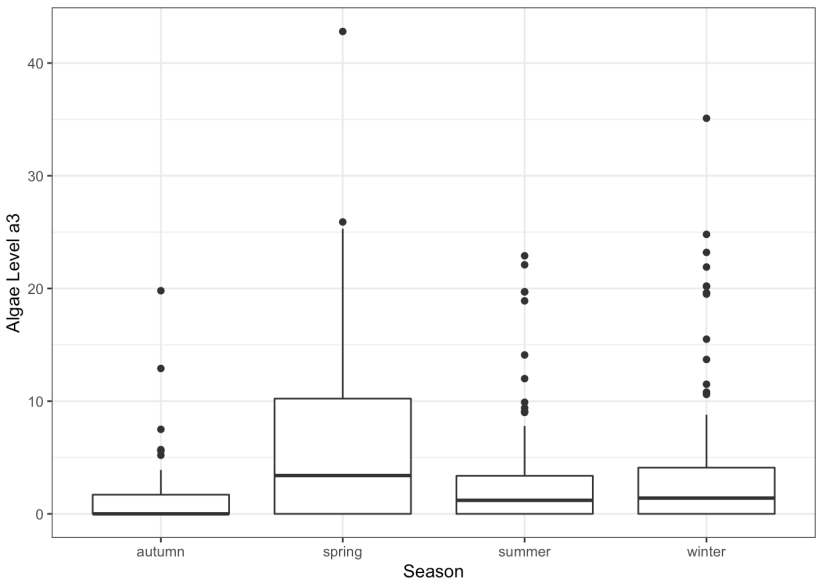


It is a bit better, to be sure.[29]

Now, let us take a look at some of the algae levels.

29: The box structure has expanded, and there still seems to be some very high values. Perhaps that is to be expected? How would we find out?

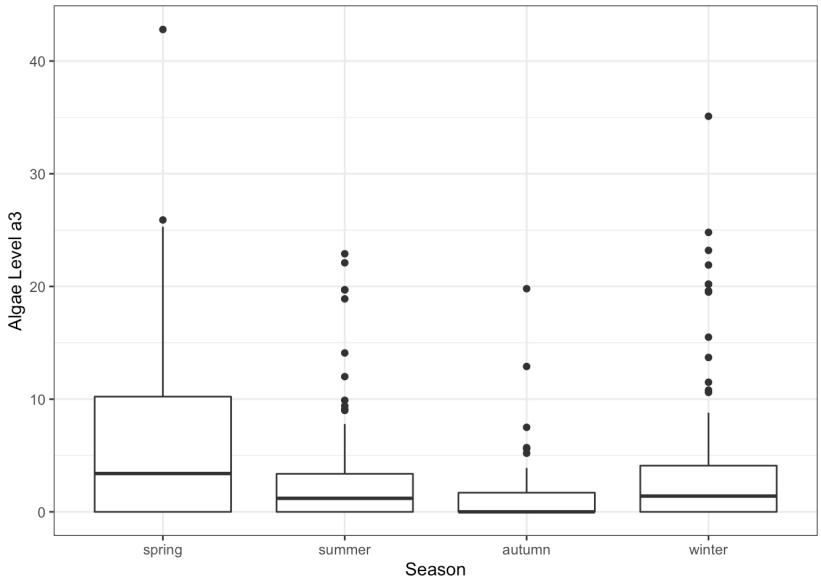
```
ggplot(algae_blooms,aes(x=season,y=a3)) +
  geom_boxplot() + xlab("Season") + ylab("Algae Level a3")
```



What does that tell us? It is hard to get a good handle on the situation because the season are out of sequential order.

We can re-arrange the factors, but it requires a bit of fancy footwork using the `forcats`'library `fct_relevel()` function, and `dplyr`'s `mutate()`.

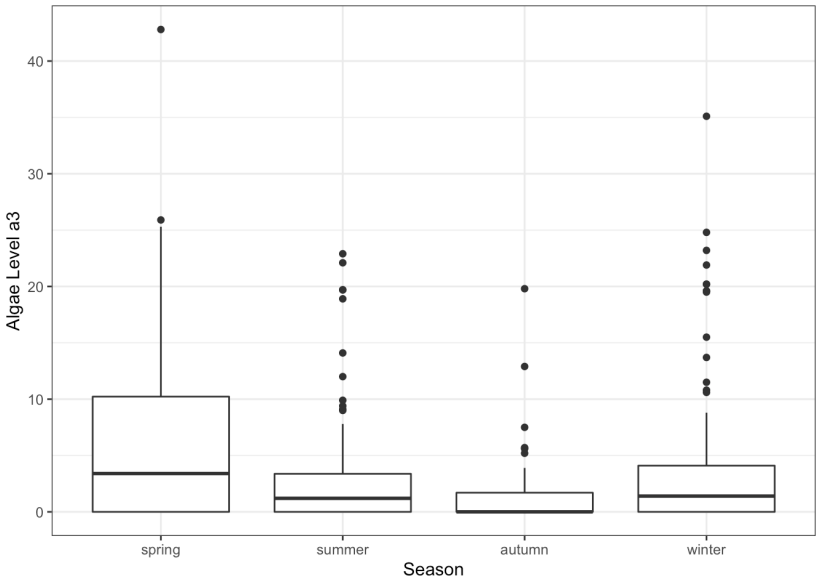
```
library(forcats) # for fct_relevel
library(dplyr)   # for mutate

algae_blooms = mutate(algae_blooms,
  size=fct_relevel(size,c("small","medium","large")),
  speed=fct_relevel(speed,c("low","medium","high")),
  season=fct_relevel(season,c("spring","summer","autumn",
    "winter"))
  )

ggplot(algae_blooms,aes(x=season,y=a3)) +
  geom_boxplot() +
  xlab("Season") +
  ylab("Algae Level a3")
```



We only have 1 year's worth of data, so it might be too early to tell, but it certainly seems as though the `a3` levels decrease from spring to winter.

Violin plots are cousins to the boxplots. Can we get a bit more insight on the `a3` trend?

```
ggplot(algae_blooms,aes(x=season,y=a3)) +
  geom_violin() +
  geom_jitter() +
  xlab("Season") +
  ylab("Algae Level a3")
```



This plot certainly seems to suggest that `a3` levels are cyclic, with a peak in the spring and low levels in the fall.

Let us return to `NH4` for a second to see if we can spot a link with the season (as we did for `a3`). We only keep the observations for which the `NH4` value is greater than 3000, and we bin them with respect to the **quartiles**.

First, filter the `algae_blooms` dataset to remove the 2 observations with missing values.[30]

30: Remember that `library(dplyr)` has been called on the previous page.

```
f.NH4.data <- filter(algae_blooms,!is.na(NH4))
nrow(f.NH4.data)
```

```
[1] 338
```

Next we remove the 11 observations for which `NH4` > 3000 (again, based on the mean + sd "argument")

```
f.NH4.data <- filter(algae_blooms,!is.na(NH4)) |>
     filter(NH4<3000)
nrow(f.NH4.data)
```

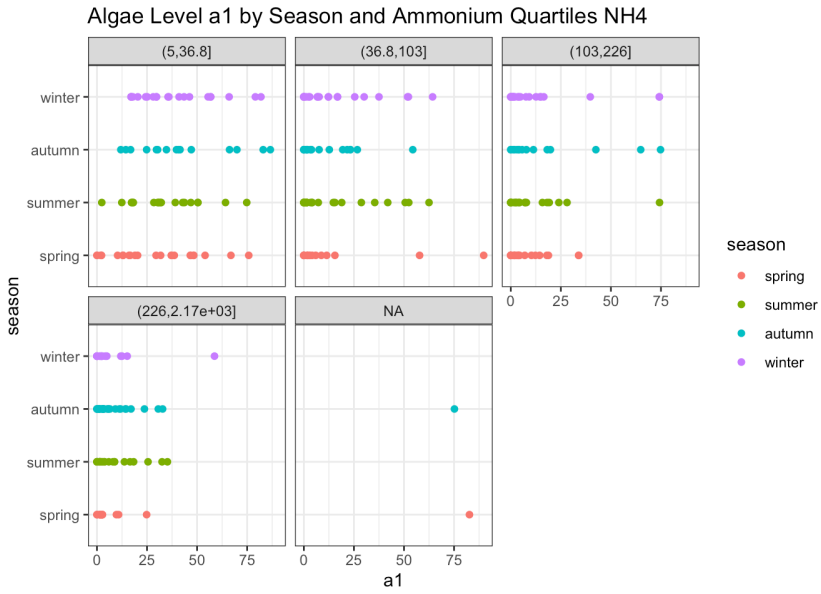
```
[1] 327
```

We create a variable indicating in which quartile the `NH4` value falls.

```
f.NH4.data <- filter(algae_blooms,!is.na(NH4))  |>
     filter(NH4<3000)  |>
     mutate(q.NH4=cut(NH4,
             quantile(NH4,c(0,0.25,0.5,0.75,1))))
```

We can now use the new variable `q.NH4` to make multi-variate comparisons, say between `a1`, `NH4`, and `season`.

```
ggplot(f.NH4.data,aes(x=a1,y=season,color=season)) +
    geom_point() +
    facet_wrap(~q.NH4) +
    ggtitle("Algae Level a1 by Season and
             Ammonium Quartiles NH4")
```



That seems decidedly odd ... why are we seeing missing values here? Have we not just removed the NAs? Let us delve in a bit deeper.

```
f.NH4.data[which(is.na(f.NH4.data$q.NH4)),]
table(f.NH4.data$q.NH4, useNA="ifany")
```

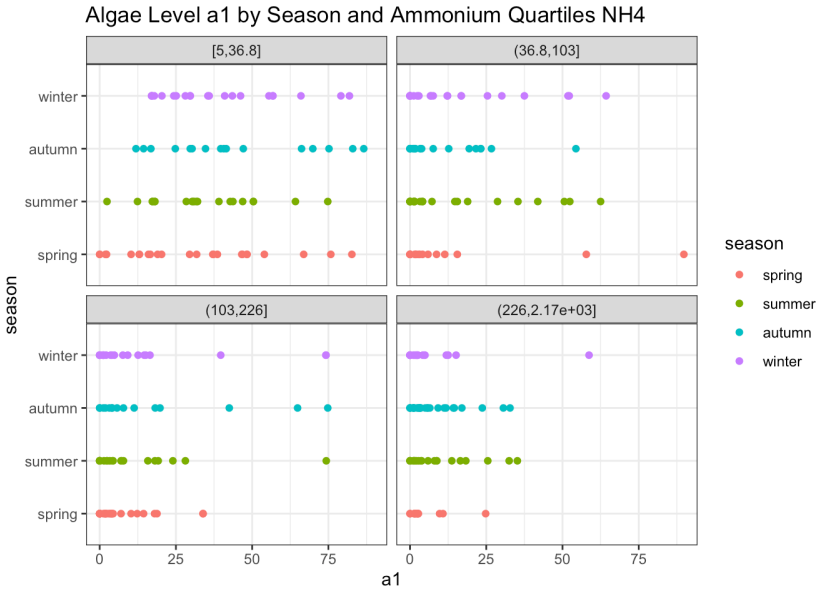
	season	size	speed	mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla	a1	a2	a3	a4	a5	a6	a7	q.NH4
53	spring	small	medium	5.6	11.8	NA	2.22	5	1	1	NA	82.7	0	0	0	0	0	0	NA
223	autumn	small	high	5.9	11.9	NA	1.88	5	1	2	NA	75.2	0	0	0	0	0	0	NA

```
Var1          Freq
(5,36.8]        80
(36.8,103]      82
(103,226]       81
(226,2.17e+03]  82
NA               2
```

The quartiles do not include their lower bound; we can remedy the situation by including an additional parameter in the `mutate()` call.

```
f.NH4.data <- filter(algae_blooms, !is.na(NH4)) |>
    filter(NH4<3000) |> mutate(q.NH4=cut(NH4,
    quantile(NH4,c(0,0.25,0.5,0.75,1)),
        include.lowest=TRUE))
```

```
ggplot(f.NH4.data, aes(x=a1, y=season, color=season)) +
    geom_point() + facet_wrap(~q.NH4) +
    ggtitle("Algae Level a1 by Season and
            Ammonium Quartiles NH4")
```

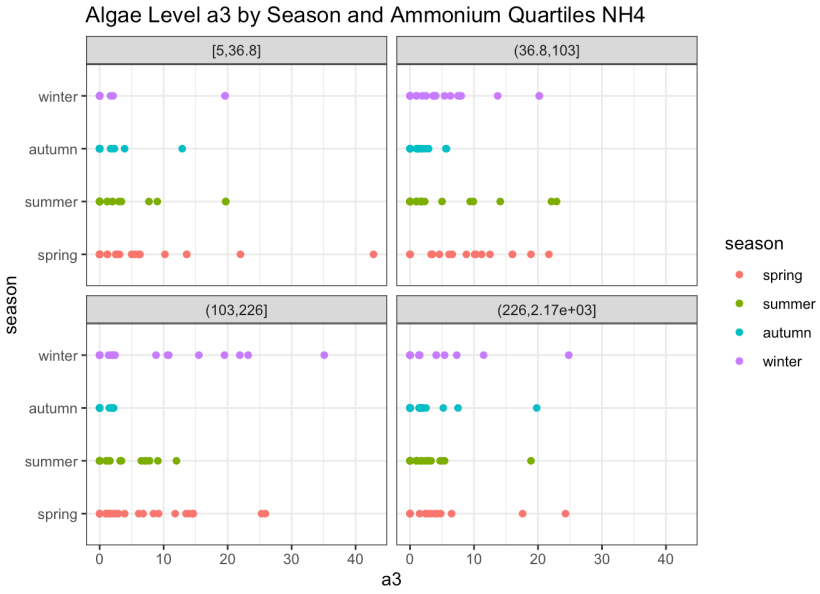


The NAs have disappeared from the graph. In any case, it seems as though the `a1` levels are inversely correlated with the `NH4` levels but that the season does not have much of an effect.[31]

31: Although we would need more work to conclude this with any degree of certainty.

We can create a similar graph for `a3` instead of `a1`.

```
ggplot(f.NH4.data,aes(x=a3,y=season,color=season)) +
    geom_point() + facet_wrap(~q.NH4) +
    ggtitle("Algae Level a3 by Season and
            Ammonium Quartiles NH4")
```



15.7.4 Data Cleaning

We found some potential anomalies in the data when we explored the dataset;[32] now let us take some time to **clean the data** to some extent.

Anomalies come in various flavours; we have already explored some potential **outlying behaviour**, now we handle **missing values**.[33] The function `complete.cases()` lists the observations for which every field is present (note that it says nothing about the **validity** of the case).

32: Although we are electing to keep them in the dataset for the time being as we lack the domain expertise to make a reasonable decision on that front.

33: Again, assume that `library(dplyr)` has already been loaded.

```
table(complete.cases(algae_blooms))
```

```
Var1   Freq
FALSE    34
TRUE    306
```

The vast majority of observations do not have missing cases, but a few still do. Is there anything special about them? Are the values missing completely at random?

```
nrow(filter(algae_blooms, !complete.cases(algae_blooms)))
summary(filter(algae_blooms, !complete.cases(algae_blooms)))
```

```
[1] 34

season    spring summer autumn winter
               7      6      8     13

size       small medium  large
              26      1      7

speed        low medium   high
               4     13     17

       mxPH   mnO2   Cl   NO3    NH4   oPO4    PO4  Chla
Min.:   5.6   5.7   0.2   0.2    5.0    1.0    1.0   0.3
Q1  :   6.6   9.2   4.5   0.8   10.0    1.0    6.0   1.7
Med.:   7.2  10.8   9.0   1.4   11.8    3.6   10.8   4.0
Mean:   7.3  10.1  19.3   2.1   62.0   25.6   34.5  13.9
Q3  :   8.0  11.3  25.2   2.5   46.3   20.2   19.2  12.2
Max.:   9.7  12.6  71.0  11.0  500.0  295.6  380.0  68.0
NA's:   2     2    16     2       2      2      7  23

         a1    a2    a3    a4    a5    a6    a7
Min.:   0.0   0.0   0.0   0.0   0.0   0.0   0.0
Q1  :  16.8   0.0   0.0   0.0   0.0   0.0   0.0
Med.:  30.3   0.0   0.0   0.0   0.0   0.0   0.0
Mean:  36.0   4.5   1.4   2.3   1.5   1.1   1.7
Q3  :  54.4   3.4   1.1   1.9   0.9   0.0   1.6
Max.:  83.0  36.5  14.6  28.8  21.1  14.5  28.0
```

Right off the bat, missing cases seem to be over-represented in `small` rivers and under-represented in `low`-speed rivers. But upon further investigation (that is, comparing with the original dataset), we suspect that the under-representation of `low`-speed rivers is not problematic as it falls in-line with the numbers in the larger dataset.[34]

34: By which we mean that `low`-speed rivers do not seem to have a systematic missing value problem.

Let us assume for now (in the interest of efficiency) that the fact that small rivers have a lot of missing cases (mostly `Cl` and `Chla`) is also not a problem.[35] The bulk of the missing values seem to come from either `Cl`, `Chla`, or `PO4`. There is also a consistent 2 missing values across the board, but we cannot use the summary output to determine if they arise from the same two observations.

35: In a real-life setting, we should *definitely* verify that this assumption is valid.

Which observations have missing `NH4` values, say?

```
algae_blooms[which(is.na(algae_blooms$NH4)),]
```

	season	size	speed	mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla	a1	a2	a3	a4	a5	a6	a7
62	summer	small	medium	6.4	NA	NA	NA	NA	NA	14	NA	19.4	0.0	0.0	2	0	3.9	1.7
199	winter	large	medium	8.0	7.6	NA	NA	NA	NA	NA	NA	0.0	12.5	3.7	1	0	0.0	4.9

While these observations also have missing values in other fields, they do have some non-missing fields as well. But they are both missing 6 of the predictor variables. How useful could they be in training a predictive model?[36] We can easily write a function that will compute how many missing cases there are for each observations.

36: The answer to that question depends on the model, of course.

```
table(apply(algae_blooms[,1:11],1,
      function(x) sum(is.na(x)))) # 1:rows, 2: columns
which(apply(algae_blooms[,1:11],1,
      function(x) sum(is.na(x)))>2)
```

```
Var1 Freq
0     306
1      20
2      12
6       2

[1]  62 199
```

Most observations have no missing cases, which is great news. There are a few with 1 or 2, but observations 62 and 199 are **wild cards**, with 6 missing predictors (out of 11). Based on the small number of such wild cards, we elect to remove them from the analysis.

IMPORTANT NOTES

- If we decide to remove observations for any reason whatsoever, we need to document the process that lead us to eliminate them, and make that process available to other analysts or to the audience.

- Why do we remove the observations with 6 missing cases, but not the ones with 2 missing cases? Had there been observations with 4 missing cases, what should we have done? What factors could influence this decision?

This dataset still contains observations with missing cases, however.

```
algae_blooms.sna = algae_blooms[-which(apply(
    algae_blooms[,1:11],1, function(x) sum(is.na(x)))>2),]
nrow(algae_blooms.sna)
```

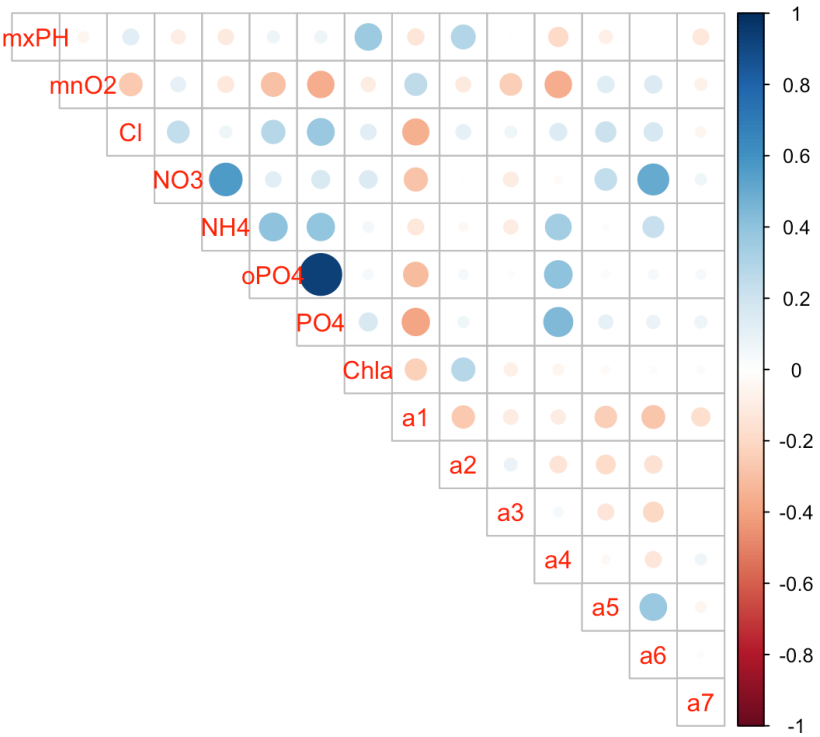
```
[1] 338
```

What can we do with the other observations for which values are missing?

One possibility is to use the set of complete observations to compute a **correlation matrix**, and to see if any numerical field is strongly correlated with another field. That way, if there is a missing value in the first field, the second could be used to impute it.

IMPORTANT NOTE: this approach only works for variables that are linearly correlated to a single other variable. Non-linear correlations and multi-variate associations will not be uncovered.

```
library(corrplot)
corrplot(cor(algae_blooms.sna[,4:18], use="complete.obs"),
         type="upper",tl.pos="d")
```



The correlation between PO4 (which has missing cases) and oPO4 (which does not, anymore) is clear. What is the nature of the relation? We use the set of complete cases to find it.

```
algae_blooms.nona <- algae_blooms.sna[-which(apply(algae_blooms.sna,1,
            function(x) sum(is.na(x)))>0),]
nrow(algae_blooms.nona)
```

```
[1] 306
```

```
PO4.oPO4.model = lm(PO4 ~ oPO4, data=algae_blooms.nona)
PO4.oPO4.model
```

```
Coefficients:
(Intercept)         oPO4
     51.811        1.203
```

The regression function is PO4= 51.811 + 1.203oPO4 (we are not particularly interested in the fit statistics at this point).

```
Intercept = PO4.oPO4.model$coefficients[[1]]
Slope = PO4.oPO4.model$coefficients[[2]]
```

What are the observations for which PO4 is missing?

```
which(is.na(algae_blooms.sna$PO4)==TRUE)
```

```
[1]  28 221 291 326 331 335
```

We can use the regression function to impute the missing PO4 values.

```
algae_blooms.sna2 <- algae_blooms.sna
algae_blooms.sna2$PO4 <- ifelse(is.na(algae_blooms.sna2$PO4),
            max(Intercept + Slope*algae_blooms.sna2$oPO4,0),
            algae_blooms.sna2$PO4)
```

We can clearly see that no values of PO4 are missing anymore.

```
which(is.na(algae_blooms.sna2$PO4)==TRUE)
```

```
integer(0)
```

That takes care of the missing values with strong linear correlation to another field. Where do we stand now?

```
summary(algae_blooms.sna2)
```

We suppress the output in the interest of legibility, but there are still some missing values. And we have exhausted the correlation trick. What else can we do?

There are many ways to tackle the problem, but we will use kNN imputation.[37] The principle is simple:

37: More details on this topic are available in Chapter 21.

1. using some similarity/distance metric (typically based on the Euclidean distance between points), identify the k nearest (complete) neighbours of each observation with a missing case;
2. compute the mean value of the missing case in the $k-$group of complete observations, and use *that* value as the imputed value.

IMPORTANT NOTES

- As we have seen when we were discussing, we often suggest **scaling** the data when dealing with distance metrics. We elected not to scale the data explicitly here. How much of an effect can that have?
- We are going to be using `DMwR`s implementation of `knnImputation()` (below). How would you go about determining if the routine scales the data internally?

```
algae_blooms.sna2 <- DMwR::knnImputation(algae_blooms.sna2,
                                         k=10)
```

Sure enough, there are no further observations with incomplete cases.

```
table(apply(algae_blooms.sna2,1,
      function(x) sum(is.na(x))))
```

```
  0
338
```

15.7.5 Principal Components

Principal components analysis (PCA) is typically used on the (numeric) predictor variables. There are methods that can be used to combine numeric and categorical variables, but for the purposes of this example, we will simply ignore the categorical fields.[38]

38: We revisit this concept in Chapter 23.

```
pca.algae = algae_blooms.sna2[,4:11]
head(pca.algae)
```

mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla
8.00	9.8	60.800	6.238	578.000	105.000	170.000	50.0
8.35	8.0	57.750	1.288	370.000	428.750	558.750	1.3
8.10	11.4	40.020	5.330	346.667	125.667	187.057	15.6
8.07	4.8	77.364	2.302	98.182	61.182	138.700	1.4
8.06	9.0	55.350	10.416	233.700	58.222	97.580	10.5
8.25	13.1	65.750	9.248	430.000	18.250	56.667	28.4

We can scale the data frame using the `scale()` function in R:

```
head(scale(pca.algae))
```

mxPH	mnO2	Cl	NO3	NH4	oPO4	PO4	Chla
-0.0019358	0.2748339	0.4423909	0.9488773	0.0611046	0.2795474	0.0145250	2.1362239
0.6101140	-0.5036258	0.3731037	-0.5578976	-0.0584991	3.1159254	1.4939011	-0.6253276
0.1729355	0.9667981	-0.0296707	0.6724831	-0.0719160	0.4606113	0.0794349	0.1855592
0.1204741	-1.8875541	0.8186771	-0.2492370	-0.2147992	-0.1043426	-0.1045862	-0.6196571
0.1029870	-0.0711482	0.3185826	2.2206563	-0.1368740	-0.1302752	-0.2610671	-0.1036382
0.4352426	1.7020100	0.5548406	1.8651183	-0.0239980	-0.4804704	-0.4167602	0.9113879

Notice the different values in the dataset. The principal components are obtained *via* the `princomp()` function.

```
pca.1 = princomp(scale(pca.algae))
summary(pca.1)
```

```
Importance of components:
                       Comp.1 Comp.2 Comp.3 Comp.4 Comp.5 Comp.6 Comp.7 Comp.8
Standard deviation     1.5334 1.2022 1.0900 0.9329 0.8750 0.8081 0.6549 0.5224
Proportion of Variance 0.2947 0.1812 0.1489 0.1091 0.0959 0.0818 0.0537 0.0342
Cumulative Proportion  0.2947 0.4760 0.6249 0.7341 0.8301 0.9119 0.9657 1.0000
```

If we can live with 75% of the variability in the numerical component of the predictors being explained by principal components, than we can reduce the dataset dimension by 4.

```
reduced.algae = data.frame(algae_blooms.sna2[,1:3],
      pca.1$scores[,1:4], algae_blooms.sna2[12:17])
head(reduced.algae, 3)
```

```
  season  size  speed    Comp.1     Comp.2     Comp.3      Comp.4   a1   a2  a3  a4   a5   a6
1 winter small medium 1.0634865  0.2877982  1.6268674  0.26280655  0.0  0.0 0.0 0.0 34.2  8.3
2 spring small medium 2.1808329  0.5908937 -2.1258588  1.07795406  1.4  7.6 4.8 1.9  6.7  0.0
3 autumn small medium 0.2097857 -0.4229638  0.6216107  0.52245855  3.3 53.6 1.9 0.0  0.0  0.0
```

Whether this reduction proves useful or not will ultimately depend on what we would like to accomplish with the data; we will study this dataset again with a particular objective in mind in Section 20.6.

15.8 Exercises

1. The ability to monitor and perform early forecasts of various river algae blooms is crucial to control the ecological harm they can cause. The `algae_blooms.csv` dataset consists of: chemical properties of various water samples of European rivers; the quantity of seven (biological) algae in each of the samples, and other characteristics of the collection process for each sample.
   a) Identify questions that could be tackled with this dataset.
   b) Determine the structure of the dataset, and provide a summary of its features.
   c) What can you say about the dataset, in terms of missing values and of the ranges of its values?
   d) Do 2-way and 3-way tables (for the categorical variables) provide you with additional insights about the dataset?
   e) Provide some simple (univariate and multivariate) visualizations of `season`, `mn02`, `NH4+`, `a1`, `a3`, and at least one other variable of your choice.
   f) Does your analysis above suggest that there are anomalies in the dataset? Take action as needed.
   g) Identify observations (cases) with only 1 missing value, 2 missing values, and so on. Are there strategies that would allow you to handle some of the cases (hint: what is the relationship between `PO4` and `oPO4`, for instance)? Are there observations that should be removed from the dataset?
   h) Produce a clean dataset to be used in analysis, with justification.
2. Consider the datasets `GlobalCitiesPBI.csv`, `2016collisionsfinal.csv`, `polls_us_election_2016.csv`, `HR_2016_Census_simple.xlsx`, and `UniversalBank.csv`. For each one:
   a) Create a "data dictionary" to explain the different fields and variables. Can you find a source for these datasets online?
   b) Develop a list of questions you would like answered about the datasets.
   c) Investigate individual variables (through simple charts, univariate statistics, etc.).
   d) Repeat the process with bivariate investigations (though simple charts, joint distributions, variable interactions, etc.).
   e) Do you trust the dataset, or not? Support your answer. If you do not trust the dataset, flag potential invalid entries, anomalous observations, missing values, or outliers. How should these entries be treated?
   f) Does any of your analysis suggest that some of the variables should be transformed? Do any of the questions you developed in step 2 support such transformations? If so, transform the data appropriately.
3. Repeat the last question with any dataset of your liking.
4. The remaining exercises use the Gapminder Tools (there is also an offline version).
   a) Explore the dataset with the Gapminder Tools in its default configuration. Do you think that there could be problems with the reported values? For instance, select Sweden and the United States from the checkbox menu on the right and follow their path from 1799 to 2018/2020. From what point onwards are the values sensible? What do you think is happening at the start of the time series?
   b) Follow Eritrea for the same duration. Look up the country's independence date from Ethiopia. What do you think the measurements prior to that date represent?
   c) Follow Austria for the same duration. Look up the historical timeline of the country's boundaries (Austria-Hungary, Anschluss, modern borders, etc.). What does that imply for the measurements?
   d) Follow Finland for the same duration. What happens in 1809? Does that tell you anything about the way data is coded in the dataset?
   e) De-select all countries and let the simulation run from 1799 to 2018/2020. Can you identify instances where a large subset of observations behaves in unexpected manners? If so, do you think that this is due to data cleaning/data processing issues?
   f) Continue exploring the dataset. You may change which variables are displayed or work with some of the other visualization methods. Overall, do you think that the dataset is sound? Would you use it to run analyses? What are some of its strengths and weaknesses?

Chapter References

[1] P. Boily. 'An Imputation Algorithm of Blood Alcohol Content Levels for Drivers and Pedestrians in Fatal Collisions ↗'. In: *Data Science Report Series* (2007).

[2] P. Boily, S. Davies, and J. Schellinck. *The Practice of Data Visualization* ↗. Data Action Lab, 2023.

[3] W. Chang. *R Graphics Cookbook*. O'Reilly, 2013.

[4] Dheeru Dua and Casey Graff. *Liver Disorders dataset at the UCI Machine Learning Repository* ↗. 2017.

[5] S. Hagiwara. *Nonresponse Error in Survey Sampling: Comparison of Different Imputation Methods*. Honours Thesis. 2012.

[6] *Height Percentile Calculator, by Age and Country* ↗.

[7] *Interactive visualization to teach about the curse of dimensionality* ↗.

[8] T. Orchard and M. Woodbury. *A Missing Information Principle: Theory and Applications*. University of California Press, 1972.

[9] T. Raghunathan et al. 'A Multivariate Technique for Multiply Imputing Missing Values Using a Sequence of Regression Models'. In: *Survey Methodology* 27.1 (2001), pp. 85–95.

[10] D.B. Rubin. *Multiple Imputation for Nonresponse in Surveys*. Wiley, 1987.

[11] L. Torgo. *Data Mining with R, 2nd ed.* CRC Press, 2016.

[12] S. van Buuren. *Flexible Imputation of Missing Data*. CRC Press, 2012.

[13] H. Wickham. 'A Layered Grammar of Graphics'. In: *Journal of Computational and Graphical Statistics* 19 (2009), pp. 3–28.

[14] H. Wickham, D. Navarro, and T. Lin Pedersen. *ggplot2: Elegant Graphics for Data Analysis*. Springer, 2021.