WASdown

Morgan Brand and Robert Schlegel

2017-06-25

Contents

Preamble

This book was written using the **bookdown** R package from Yihui Xie (Xie, 2017). It is a combination of work done by Hadley (@hadleywickham), Garrett (@statgarrett) and Chester (@Old_Man_Chester) with some insight from the authors. This a complementory book for a workshop held at the World Aquaculture Conference in Cape Town June 2017. We (the authors) hope to provide exposure to the world of R and some relevant online resources to students which will hopefully put them on track for being self tought coding aquicultureists. An introduction to using R, RStudio, and R Markdown by Chester Ismay is also available in a free book here and more in his DataCamp course at Effective Data Storytelling using the tidyverse. For more insight into the useage we would advice you to look through R for Data Science and ModernDive. For an example of the role within the science workflow read the **Nature** publication Our path to better science in less time using open data science tools and the extensive testing they have done.

Colophon

The source of the book is available here and was built with versions of R packages (and their dependent packages) given below. This may not be of importance for initial readers of this book, but the hope is you can reproduce a duplicate of this book by installing these versions of the packages.

```
devtools::session_info(c("tidyverse"))
```

```
## Session info -------------------------------------------------------------

##  setting  value
##  version  R version 3.3.0 (2016-05-03)
##  system   x86_64, darwin13.4.0
##  ui       X11
##  language (EN)
##  collate  en_US.UTF-8
##  tz       Africa/Johannesburg
##  date     2017-06-25

## Packages -----------------------------------------------------------------

##  package      * version    date       source
##  assertthat     0.2.0      2017-04-11 cran (@0.2.0)
##  BH             1.62.0-1   2016-11-19 cran (@1.62.0-)
##  bindr          0.1        2016-11-13 cran (@0.1)
##  bindrcpp       0.2        2017-06-17 cran (@0.2)
##  broom          0.4.2      2017-02-13 cran (@0.4.2)
##  cellranger     1.1.0      2016-07-27 cran (@1.1.0)
##  colorspace     1.2-6      2015-03-11 CRAN (R 3.3.0)
##  curl           1.2        2016-08-13 CRAN (R 3.3.0)
##  dichromat      2.0-0      2013-01-24 CRAN (R 3.3.0)
##  digest         0.6.12     2017-01-27 cran (@0.6.12)
##  dplyr          0.7.0      2017-06-09 cran (@0.7.0)
##  forcats        0.2.0.9000 2017-06-21 Github (hadley/forcats@714063c)
##  foreign        0.8-66     2015-08-19 CRAN (R 3.3.0)
##  ggplot2        2.2.1.9000 2017-06-22 Github (tidyverse/ggplot2@2907cf5)
##  glue           1.1.0      2017-06-13 cran (@1.1.0)
##  gtable         0.2.0      2016-02-26 CRAN (R 3.3.0)
##  haven          1.0.0      2016-09-23 cran (@1.0.0)
##  hms            0.3        2016-11-22 cran (@0.3)
##  httr           1.2.1      2016-07-03 cran (@1.2.1)
##  jsonlite       1.5        2017-06-01 cran (@1.5)
##  labeling       0.3        2014-08-23 CRAN (R 3.3.0)
##  lattice        0.20-33    2015-07-14 CRAN (R 3.3.0)
##  lazyeval       0.2.0      2016-06-12 CRAN (R 3.3.0)
##  lubridate      1.6.0      2016-09-13 cran (@1.6.0)
```

```
## magrittr      1.5        2014-11-22 CRAN (R 3.3.0)
## MASS          7.3-45     2016-04-21 CRAN (R 3.3.0)
## mime          0.5        2016-07-07 CRAN (R 3.3.0)
## mnormt        1.5-5      2016-10-15 cran (@1.5-5)
## modelr        0.0.0.9000 2017-06-21 Github (hadley/modelr@5ba6af4)
## munsell       0.4.3      2016-02-13 CRAN (R 3.3.0)
## nlme          3.1-127    2016-04-16 CRAN (R 3.3.0)
## openssl       0.9.4      2016-05-25 CRAN (R 3.3.0)
## pkgconfig     2.0.1      2017-03-21 cran (@2.0.1)
## plogr         0.1-1      2016-09-24 cran (@0.1-1)
## plyr          1.8.4      2016-06-08 CRAN (R 3.3.0)
## psych         1.7.3.21   2017-03-22 cran (@1.7.3.2)
## purrr         0.2.2.2    2017-05-11 cran (@0.2.2.2)
## R6            2.2.2      2017-06-17 cran (@2.2.2)
## RColorBrewer  1.1-2      2014-12-07 CRAN (R 3.3.0)
## Rcpp          0.12.11    2017-05-22 cran (@0.12.11)
## readr         1.1.1      2017-05-16 cran (@1.1.1)
## readxl        1.0.0      2017-04-18 cran (@1.0.0)
## rematch       1.0.1      2016-04-21 cran (@1.0.1)
## reshape2      1.4.2      2016-10-22 cran (@1.4.2)
## rlang         0.1.1      2017-05-18 cran (@0.1.1)
## rvest         0.3.2      2016-06-17 CRAN (R 3.3.0)
## scales        0.4.1      2016-11-09 CRAN (R 3.3.2)
## selectr       0.3-1      2016-12-19 CRAN (R 3.3.2)
## stringi       1.1.5      2017-04-07 cran (@1.1.5)
## stringr       1.2.0      2017-02-18 cran (@1.2.0)
## tibble        1.3.3      2017-05-28 cran (@1.3.3)
## tidyr         0.6.3      2017-05-15 cran (@0.6.3)
## tidyverse     1.1.1      2017-01-27 CRAN (R 3.3.2)
## xml2          1.1.1      2017-01-24 cran (@1.1.1)
```

Book was last updated by morganbrand on Sunday, June 25, 2017 15:41:08 SAST.

Chapter 1

Introduction

Before you will be in any shape to get **R**rring you will need to download the basics. There are lots of resources that can guide you through this process and it is not our intention to create duplicates. You can spend your time looking for answers online and there will surley be several versions of the same thing however, you would be wise to follow the likes of @hadleywickham in this process. Below there is an excerpt from his book co-authored with @statgarrett

1.1 Excerpt from R for Data Science

1.1.1 R

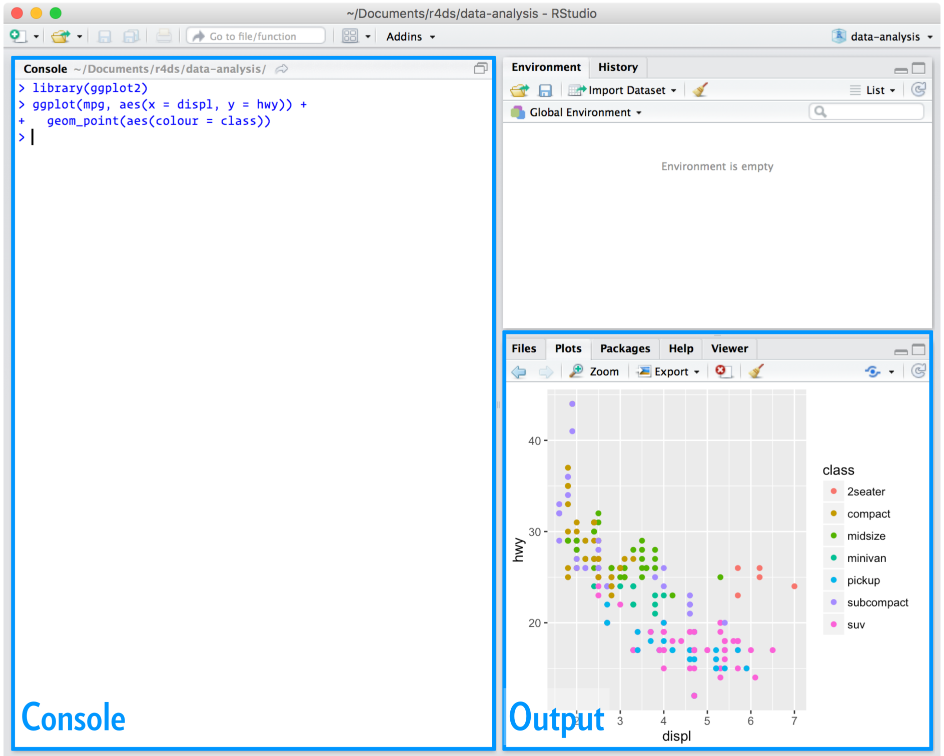
To download R, go to CRAN, the **c**omprehensive **R** **a**rchive **n**etwork. CRAN is composed of a set of mirror servers distributed around the world and is used to distribute R and R packages. Don't try and pick a mirror that's close to you: instead use the cloud mirror, https://cloud.r-project.org, which automatically figures it out for you.

A new major version of R comes out once a year, and there are 2-3 minor releases each year. It's a good idea to update regularly. Upgrading can be a bit of a hassle, especially for major versions, which require you to reinstall all your packages, but putting it off only makes it worse.

1.1.2 RStudio

RStudio is an integrated development environment, or IDE, for R programming. Download and install it from http://www.rstudio.com/download. RStudio is updated a couple of times a year. When a new version is available, RStudio will let you know. It's a good idea to upgrade regularly so you can take advantage of the latest and greatest features. For this book, make sure you have RStudio 1.0.0.

When you start RStudio, you'll see two key regions in the interface:



For now, all you need to know is that you type R code in the console pane, and press enter to run it. You'll learn more as we go along!

1.1.3 The tidyverse

You'll also need to install some R packages. An R **package** is a collection of functions, data, and documentation that extends the capabilities of base R. Using packages is key to the successful use of R. The majority of the packages that you will learn in this book are part of the so-called tidyverse. The packages in the tidyverse share a common philosophy of data and R programming, and are designed to work together naturally.

You can install the complete tidyverse with a single line of code:

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

On your own computer, type that line of code in the console, and then press enter to run it. R will download the packages from CRAN and install them on to your computer. If you have problems installing, make sure that you are connected to the internet, and that https://cloud.r-project.org/ isn't blocked by your firewall or proxy.

You will not be able to use the functions, objects, and help files in a package until you load it with `library()`. Once you have installed a package, you can load it with the `library()` function:

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 3.3.2

## Warning: package 'tibble' was built under R version 3.3.2

## Warning: package 'tidyr' was built under R version 3.3.2

## Warning: package 'readr' was built under R version 3.3.2

## Warning: package 'purrr' was built under R version 3.3.2
```

```
## Warning: package 'dplyr' was built under R version 3.3.2
```

This tells you that tidyverse is loading the ggplot2, tibble, tidyr, readr, purrr, and dplyr packages. These are considered to be the **core** of the tidyverse because you'll use them in almost every analysis.

1.1.4 R Markdown

R Markdown provides an unified authoring framework for data science, combining your code, its results, and your prose commentary. R Markdown documents are fully reproducible and support dozens of output formats, like PDFs, Word files, slideshows, and more.

R Markdown files are designed to be used in three ways:

1. For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.
2. For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them (i.e. the code).
3. As an environment in which to *do* data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.

R Markdown integrates a number of R packages and external tools. This means that help is, by-and-large, not available through `?`. Instead, as you work through this chapter, and use R Markdown in the future, keep these resources close to hand:

- R Markdown Cheat Sheet: *Help > Cheatsheets > R Markdown Cheat Sheet*,
- R Markdown Reference Guide: *Help > Cheatsheets > R Markdown Reference Guide*.

Both cheatsheets are also available at http://rstudio.com/cheatsheets.

1.2 Thesisdown

Thesisdown is built from Bookdown. It is a very useful tool to start working from if your goal is to submit your thesis using the language R Markdown. It allows you to jump into a working template and coustomize the content. The Thesisdown package was created by @Old_Man_Chester and the introduction to the package which can be found here is copied below

---

This project was inspired by the bookdown package and is an updated version of my Senior Thesis template in the `reedtemplates` package here.

Currently, the PDF and gitbook versions are fully-functional. The word and epub versions are developmental, have no templates behind them, and are essentially calls to the appropriate functions in bookdown.

The current output for the four versions is here:

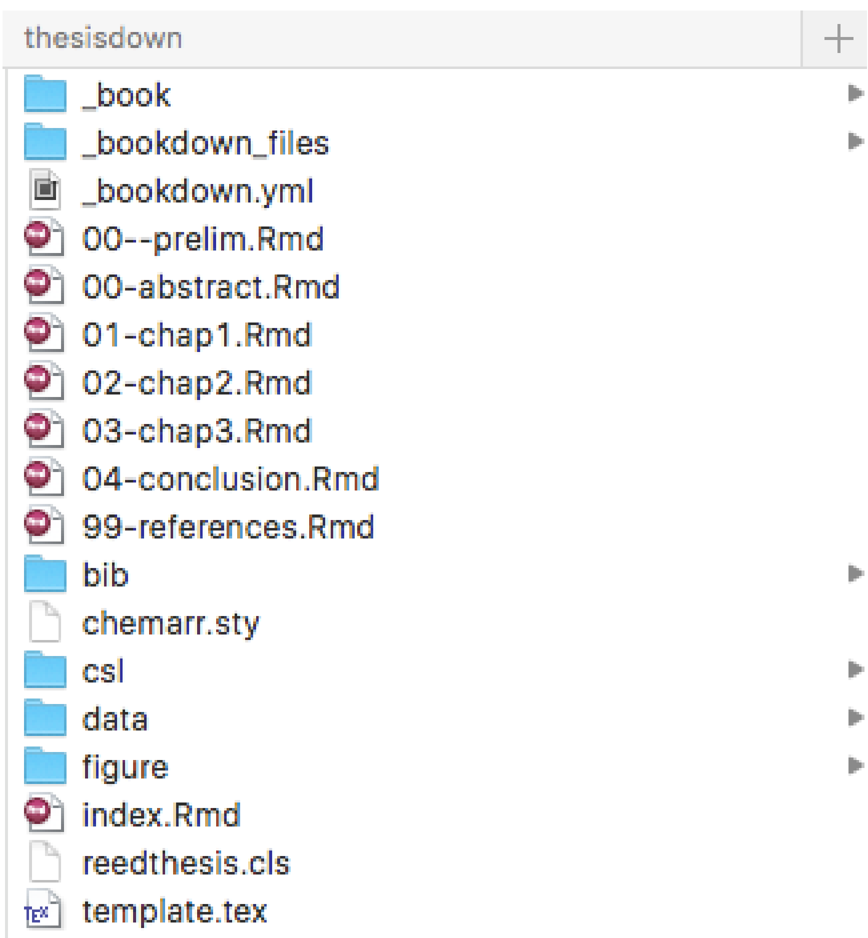
- PDF (Generating LaTeX file is available here with other files at in the book directory.)
- Word
- ePub
- gitbook

Under the hood, the Reed College LaTeX template (and soon the Reed College Word template) is used to ensure that documents conform precisely to submission standards. At the same time, composition and formatting can be done using lightweight markdown syntax, and **R** code and its output can be seamlessly included using rmarkdown.

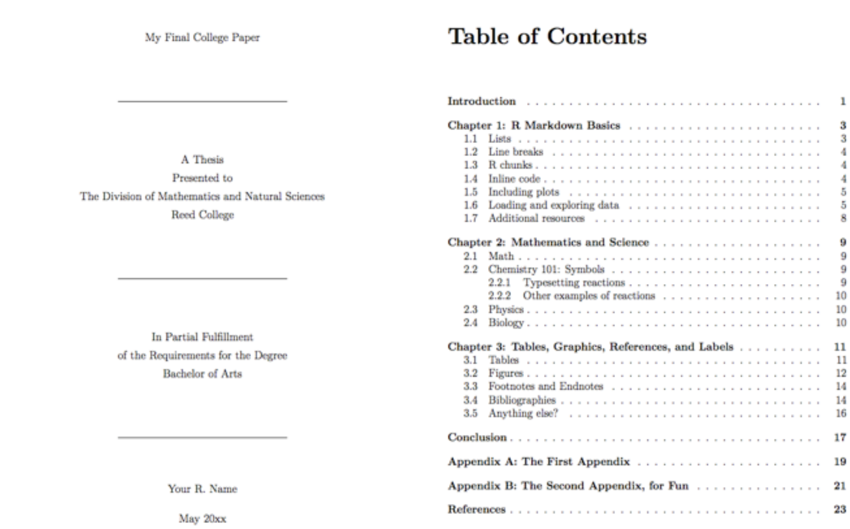
Using **thesisdown** has some prerequisites which are described below. To compile PDF documents using **R**, you are going to need to have LaTeX installed. It can be downloaded for Windows at http://http://miktex.

org/download and for Mac at http://tug.org/mactex/mactex-download.html. Follow the instructions to install the necessary packages after downloading the (somewhat large) installer files. You may need to install a few extra LaTeX packages on your first attempt to knit as well.

1.2.1 The basic filing structure



1.2.2 PDF output



My Final College Paper

A Thesis
Presented to
The Division of Mathematics and Natural Sciences
Reed College

In Partial Fulfillment
of the Requirements for the Degree
Bachelor of Arts

Your R. Name

May 20xx

Table of Contents

1.2.3 YAML

```
---
author: 'Your R. Name'
date: 'May 20xx'
division: 'Mathematics and Natural Sciences'
advisor: 'Advisor F. Name'
#altadvisor: 'Your Other Advisor'
# Delete line 6 if you only have one advisor
department: 'Mathematics'
title: 'My Final College Paper'
knit: "bookdown::render_book"
site: bookdown::bookdown_site
output:
  thesisdown::thesis_pdf: default
#  thesisdown::thesis_gitbook: default
#  thesisdown::thesis_word: default
#  thesisdown::thesis_epub: default
# If you are creating a PDF you'll need to write your preliminary content here or
# use code similar to line 20 for the files.  If you are producing in a different
# format than PDF, you can delete or ignore lines 20-31 in this YAML header.
abstract: >
  `r if(knitr:::is_latex_output()) paste(readLines("00-abstract.Rmd"), collapse = ' ')`
# If you'd rather include the preliminary content in files instead of inline
# like below, use a command like that for the abstract above.  Note that a tab is
# needed on the line after the >.
acknowledgements: >
  I want to thank a few people.
dedication: >
  You can have a dedication here if you wish.
preface: >
  This is an example of a thesis setup to use the reed thesis document class
  (for LaTeX) and the R bookdown package, in general.
bibliography: bib/thesis.bib
# Download your specific bibliography database file and refer to it in the line above.
csl: csl/apa.csl
# Download your specific csl file and refer to it in the line above.
lot: true
lof: true
#space_between_paragraphs: true
# Delete the # at the beginning of the previous line if you'd like
# to have a blank new line between each paragraph
#header-includes:
#- \usepackage{tikz}
---
```

1.3 Blogdown

The beauty of a platform like Blogdown is in its ability to transport your scientific work into the public domain with very little extra effort. A website is generated from R Markdown documents. You can include all your results, analysis, graphics andcan be computed and rendered dynamically from R code to your website!

@xieyihui and @Amber Thomas have put together an open book using bookdown which details the process of setting up a blogdown for your own prive use. A section of their book, Creating Websites with R Markdown is included below and the online version is licensed under the Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

We introduce an R package, **blogdown**, in this short book, to teach you how to create websites using R Markdown and Hugo. If you have experience with creating websites, you may naturally ask what the benefits of using R Markdown are, and how **blogdown** is different with existing popular website platforms, such as WordPress. There are two major highlights of **blogdown**:

1. It produces a static website, meaning the website only consists of static files such as HTML, CSS, JavaScript, and images, etc. You can host the website on any web servers (see Chapter **??** for details).

The website does not require server-side scripts such as PHP or databases like WordPress does. It is just one folder of static files. We will explain more benefits of static websites in Chapter **??**, when we introduce the static website generator Hugo.

2. The website is generated from R Markdown documents (R is optional, i.e., you can use plain Markdown documents without R code chunks). This brings a huge amount of benefits, especially if your website is related to data analysis or (R) programming. Being able to use Markdown implies simplicity and more importantly, *portability* (e.g., you are giving yourself the chance to convert your blog posts to PDF and publish to journals or even books in the future). R Markdown gives you the benefits of dynamic documents — all your results, such as tables, graphics, and inline values, can be computed and rendered dynamically from R code, hence the results you present on your website are more likely to be reproducible. An additional yet important benefit of using R Markdown is that you will be able to write technical documents easily, due to the fact that **blogdown** inherits the HTML output format from **bookdown** (Xie, 2017). For example, it is possible to write LaTeX math equations, BibTeX citations, and even theorems and proofs if you want.

Please do not be misled by the word "blog" in the package name: **blogdown** is for general-purpose websites, and not only for blogs. For example, both authors of this book have their personal websites, where you can find information about their projects, blogs, package documentations, and so on.[1] All their pages are built from **blogdown** and Hugo.

1.4 Git and Github

The initial process of getting gited is sometimes challenging but we encourage you to pursist and get it set up. Again, there are several online resources which provide detailed step by steps and it is not our intention to guide you through this but rather point you towards some of the 'good' ones.

For this section we have taken content from Happy Git and GitHub for the useR which was written by @JennyBryan and licenced under Creative Commons Attribution-NonCommercial 4.0 International License.

1.4.1 Why Git?

Git is a **version control system**. Its original purpose was to help groups of developers work collaboratively on big software projects. Git manages the evolution of a set of files – called a **repository** – in a sane, highly structured way. If you have no idea what I'm talking about, think of it as the "Track Changes" features from Microsoft Word on steroids.

Git has been re-purposed by the data science community. In addition to using it for source code, we use it to manage the motley collection of files that make up typical data analytical projects, which often consist of data, figures, reports, and, yes, source code.

A solo data analyst, working on a single computer, will benefit from adopting version control. But not nearly enough to justify the pain of installation and workflow upheaval. There are much easier ways to get versioned back ups of your files, if that's all you're worried about.

In my opinion, **for new users**, the pros of Git only outweigh the cons when you factor in the overhead of communicating and collaborating with other people. Who among us does not need to do that? Your life is much easier if this is baked into your workflow, as opposed to being a separate process that you dread or neglect.

[1] Yihui's homepage is at https://yihui.name. He writes blog posts in both Chinese (https://yihui.name/cn/) and English (https://yihui.name/en/), and documents his software packages such as **knitr** (https://yihui.name/knitr/) and **animation** (https://yihui.name/animation/). Occasionally he also writes articles like https://yihui.name/rlp/ when he finds interesting topics but does not bother a formal journal submission. Amber's homepage is at https://proquestionasker.github.io. Similarly, you can find her blog and project pages.

1.4.2 Why GitHub?

This is where hosting services like GitHub, Bitbucket, and GitLab come in. They provide a home for your Git-based projects on the internet. If you have no idea what I'm talking about, think of it as DropBox but much, much better. The remote host acts as a distribution channel or clearinghouse for your Git-managed project. It allows other people to see your stuff, sync up with you, and perhaps even make changes. These hosting providers improve upon traditional Unix Git servers with well-designed web-based interfaces.

Even for private solo projects, it's a good idea to push your work to a remote location for peace of mind. Why? Because it's fairly easy to screw up your local Git repository, especially when you're new at this. The good news is that often only the Git infrastructure is borked up. Your files are just fine! Which makes your Git pickle all the more frustrating. There are official Git solutions to these problems, but they might require expertise and patience you can't access at 3a.m. If you've recently pushed your work to GitHub, it's easy to grab a fresh copy, patch things up with the changes that only exist locally, and get on with your life.

Don't get too caught up on public versus private at this point. There are many ways to get private repositories from the major providers for low or no cost. Just get started and figure out if and how Git/GitHub is going to work for you! If you outgrow this arrangement, you can throw some combination of technical savvy and money at the problem. You can either pay for a higher level of service or self-host one of these platforms.

Outside of @JennyBryan book you can find a detailed guide to getting **Git**ed with RStudio by /**?** here.

1.5 Twitterverse

You might also want to follow these guys on Twitter:

- Hadley Wickham @hadleywickham
- Garrett Grolemund @statgarrett
- Chester Ismay @Old_Man_Chester
- Yihui Xie @xieyihui
- Jenny Bryan @JennyBryan
- RStudio Tips @rstudiotips

If you're an active Twitter user, follow the `#rstats` hashtag.

Chapter 2

Petrol

This report shows analyses performed and figures created from 10+ years of petrol usage with a 2003 Volkswagen Polo sedan (1.4).

2.1 Loading the data

```
# Load libraries
library(tidyverse)
library(lubridate)
library(broom)
```

```
## Warning: package 'broom' was built under R version 3.3.2
```

```
# Load data
petrol <- read.csv("data/petrol_info.csv")
#petrol <- read.csv("~/R_WAS/petrol_info.csv")

# Correct date
petrol$date <- as.Date(petrol$date, "%d-%m-%y")

# Correct 'full' categorical label
petrol$full[is.na(petrol$full)] <- 0
petrol$full <- factor(petrol$full, labels = c("no", "yes"))
petrol$full <- factor(petrol$full, levels = c("yes", "no"))

# Remove problem rows
petrol <- petrol[complete.cases(petrol),]

# Add month and year columns
petrol$month <- floor_date(petrol$date, "month")
petrol$year <- floor_date(petrol$date, "year")
```

2.2 Graphical observations

Over the lifespan of a vehicle, one constant will always be the increasing count of the odometre. We may plot this as a time series.

```
ggplot(data = petrol, aes(x = date, y = odom)) +
  geom_line() +
  geom_point() +
  scale_y_continuous(breaks = seq(80000, 200000, 20000)) +
  labs(y = "distance (km)", x = "")
```

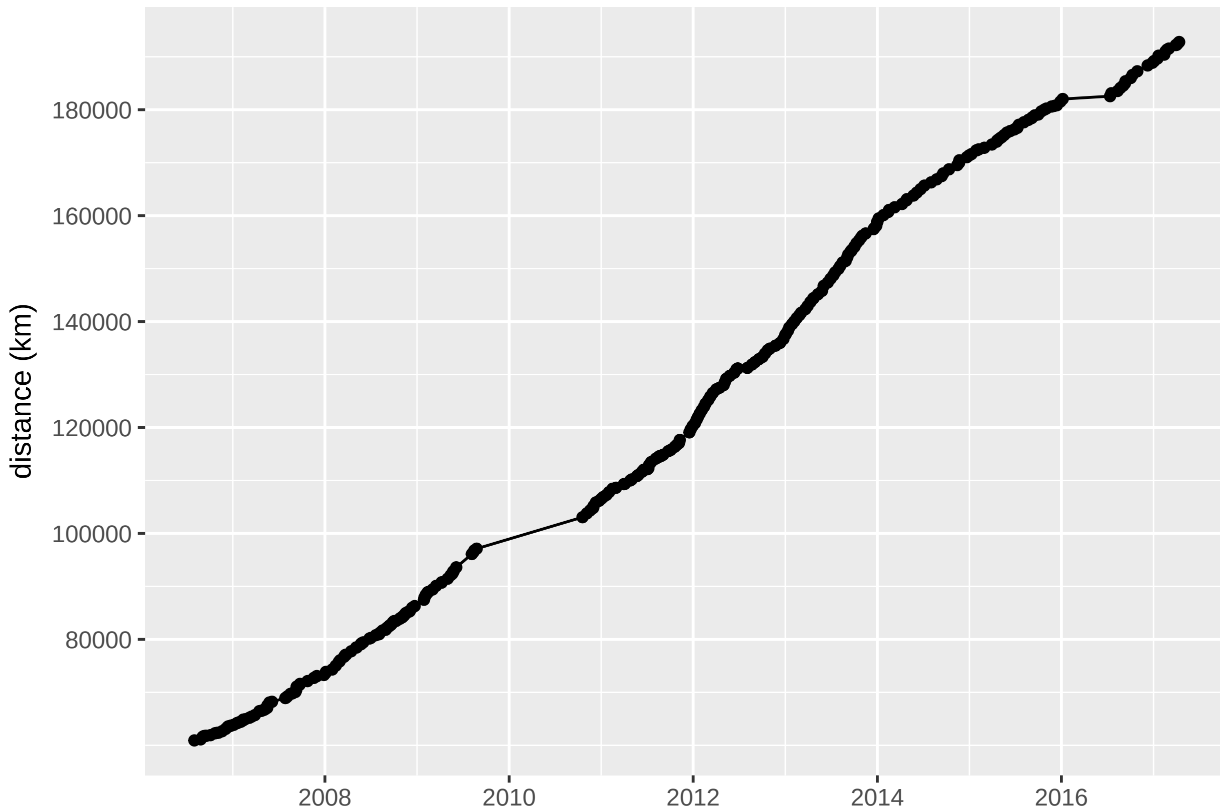


Figure 2.1: Line graph showing total distance traveled (km) over time.

Also of interest is how far the distances between fill ups may be. There are a couple of gaps in this time series, which lend themselves to some dramatic numbers, but overall we are able to get an idea of the true distances.

```
# Create a column showing distance between fill-ups
petrol$dist <- c(0,diff(as.matrix(petrol$odom)))

# Total distance traveled
petrol$dist_total <- petrol$odom-petrol$odom[1]

# Histogram
ggplot(data = petrol, aes(x = date, y = odom)) +
  geom_line() +
  geom_point() +
  geom_line(aes(y = dist), colour = "blue") +
  scale_y_continuous(breaks = seq(0, 200000, 40000)) +
  labs(y = "distance (km)", x = "")
```

This blue line would provide more useful information if visualised as a histogram.

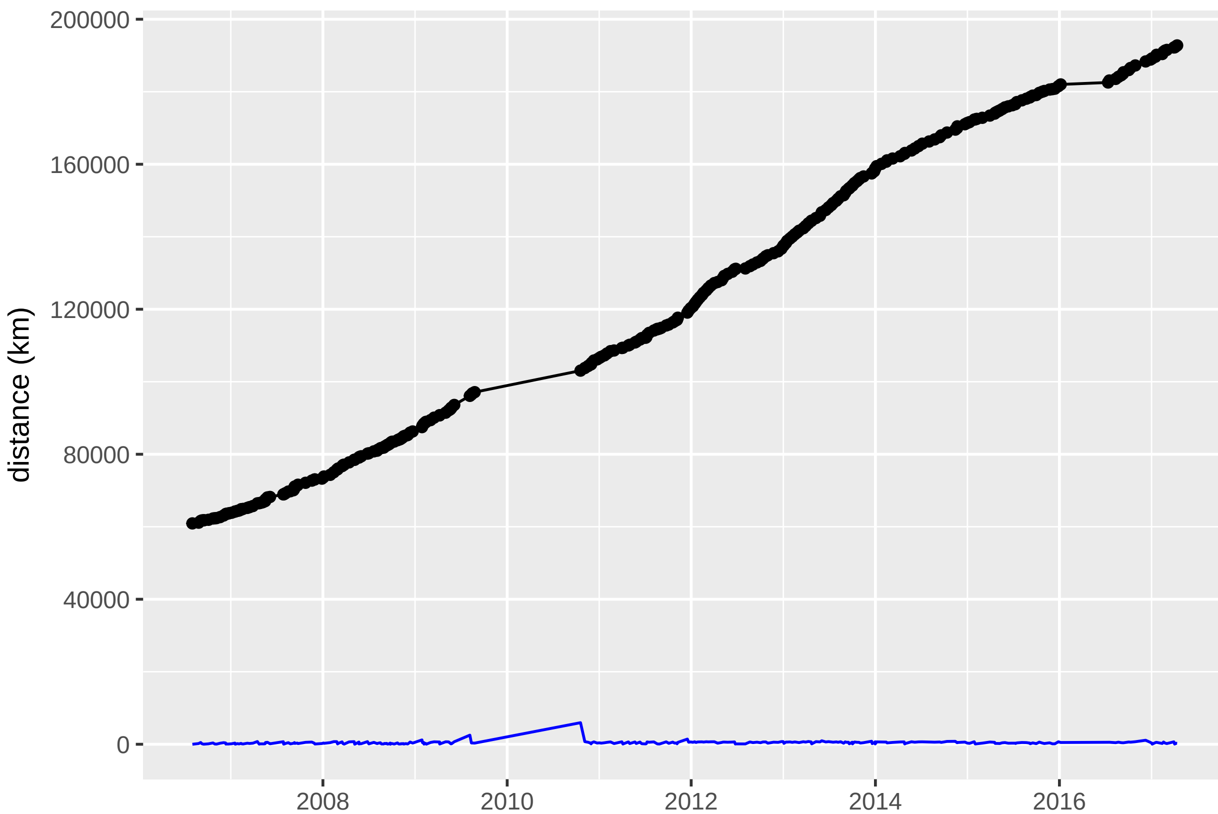


Figure 2.2: Same as previous figure with the total distance between fill ups shown with a blue line.

```
ggplot(data = petrol[petrol$dist <= 1000,], aes(x = dist)) +
  geom_histogram(fill = "violet", colour = "grey40")
```

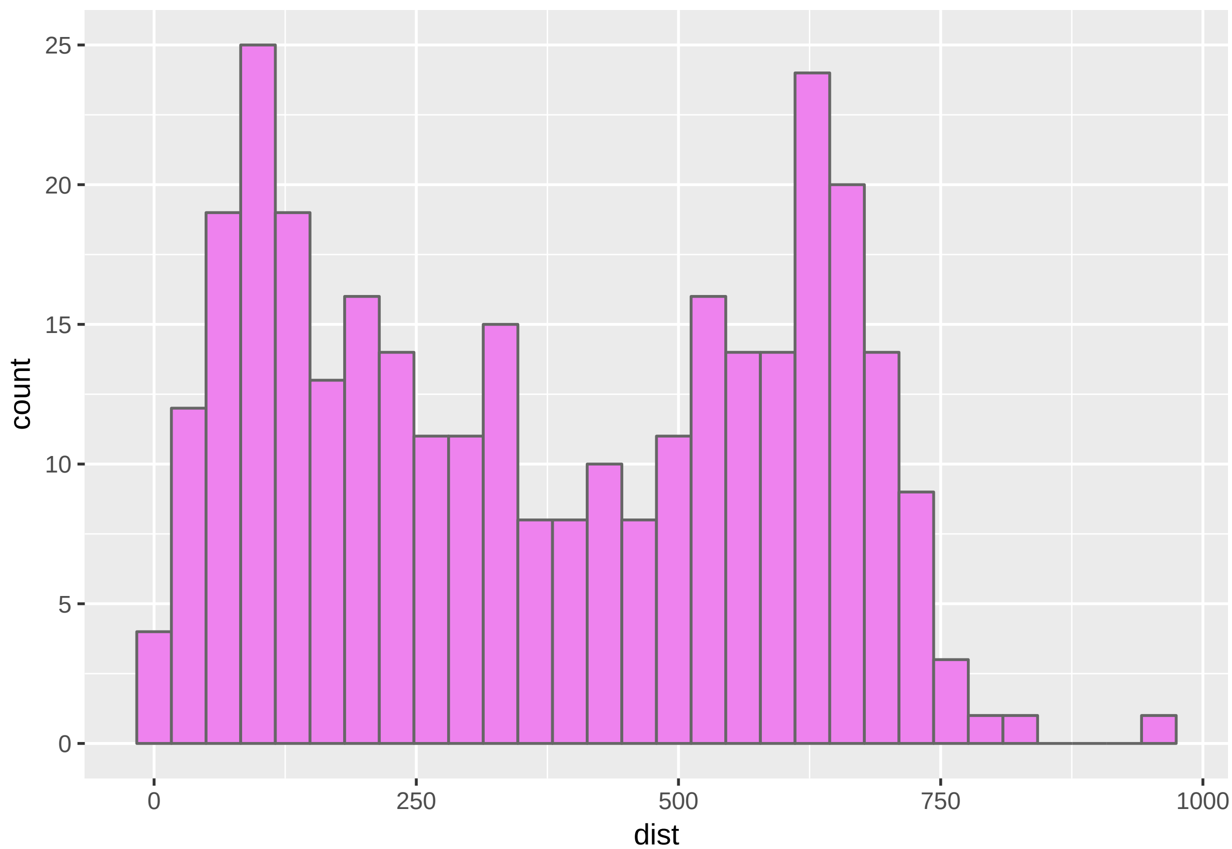


Figure 2.3: Histogram showing the distances traveled between fillings. Any values over 1,000km were subsetted out.

This histogram shows a bimodal distribution with a clustering of distances around 100 kms and 600 kms. This is not so strange if you think about it as it shows that this driver would tend to either go short distance between filling up, or long distances. This is likely linked to spending behaviours, which is the next thing to investigate.

2.3 Spending patterns

We may produce another histogram to plot the amount of money spent per visit to the petrol station.

```
ggplot(data = petrol, aes(x = cost)) +
  geom_histogram(aes(fill = full), colour = "grey40") +
  labs(x = "Price (R)")
```

The distribution shown in this histogram is also not surprising if one thinks about it. This histogram is showing two different spending habits. On the left hand side we see that there are distinct columns rising out from the others. This is when the driver went to the station and spent specifically, R20, R50, R100, R200, R300 or R400. On the right hand side of the histogram we see a more normal distribution of columns. These are the prices spent when filling up the tank to full.

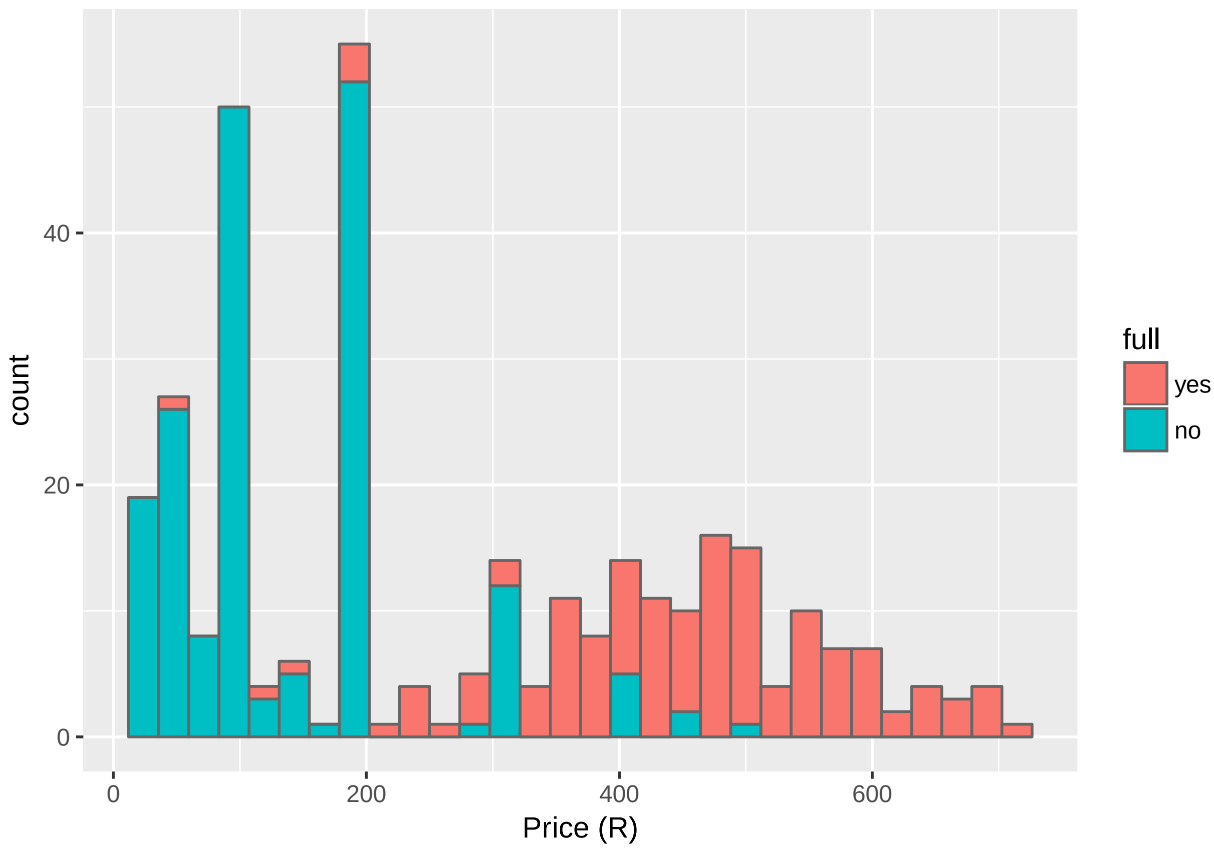
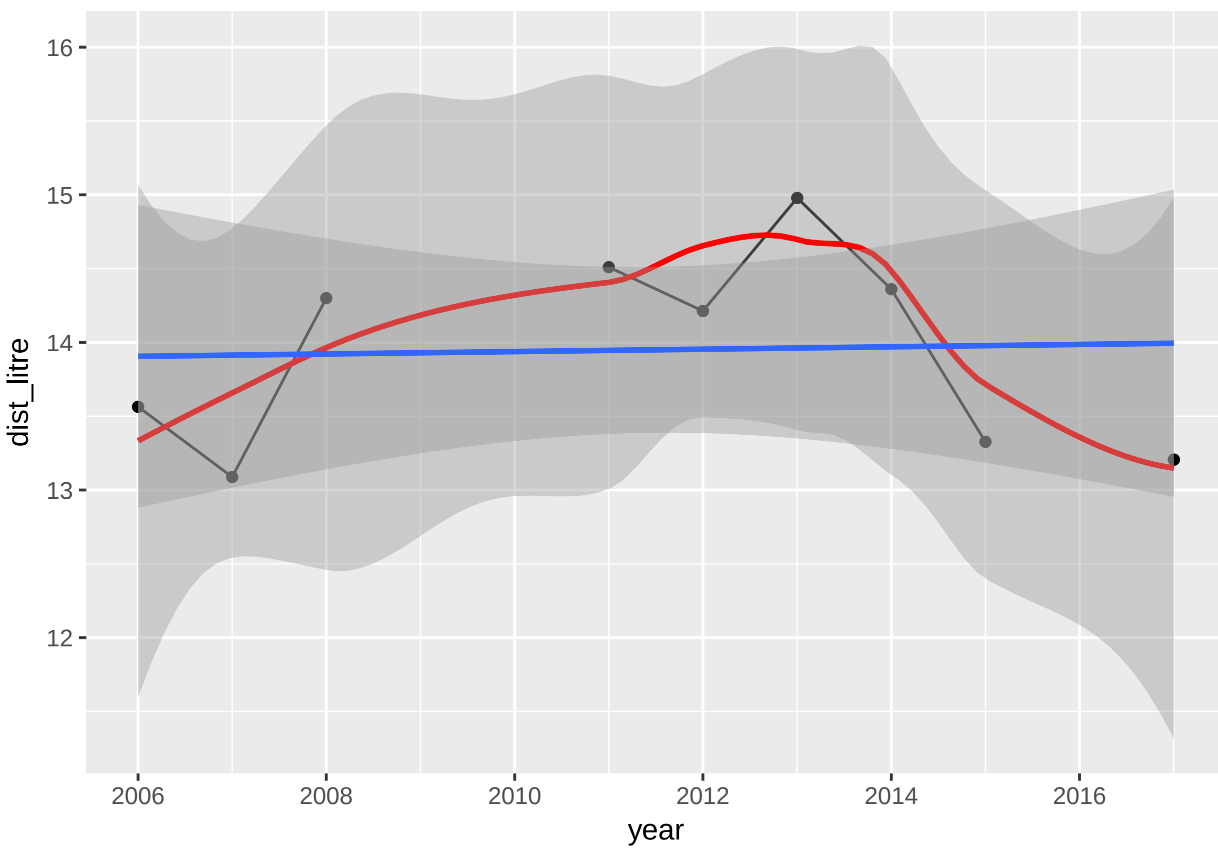


Figure 2.4: Histogram showing the amount of money spent per visit to the petrol station. The colours show if the tank was filled during that visit or not.

2.4 Petrol usage per km

One of the first things any car owner wants to know about their vehicle is the mileage their vehicle is getting. And whether or not this is decreasing with wear. Because we don't know exactly how much petrol is used between each fill up this becomes a bit tricky. We overcome this challenge by creating annual sums of petrol use. With these we may then calculate the distance traveled per litre more broadly. Monthly means are too erratic to be useful.

```
# Create monthly means
petrol_annual <- petrol %>%
  select(-full, -date, -month) %>%
  group_by(year) %>%
  mutate(dist2 = sum(dist)) %>%
  mutate(litre2 = sum (litre)) %>%
  summarise_all(mean) %>%
  mutate(dist_litre = dist2/litre2)

# Remove outliers caused during absences
is.na(petrol_annual$dist_litre) <- petrol_annual$dist_litre > 16

# Plot it
ggplot(data = petrol_annual, aes (x = year, y = dist_litre)) +
  geom_line() +
  geom_point() +
  geom_smooth(colour = "red") +
  geom_smooth(method = "lm")
```

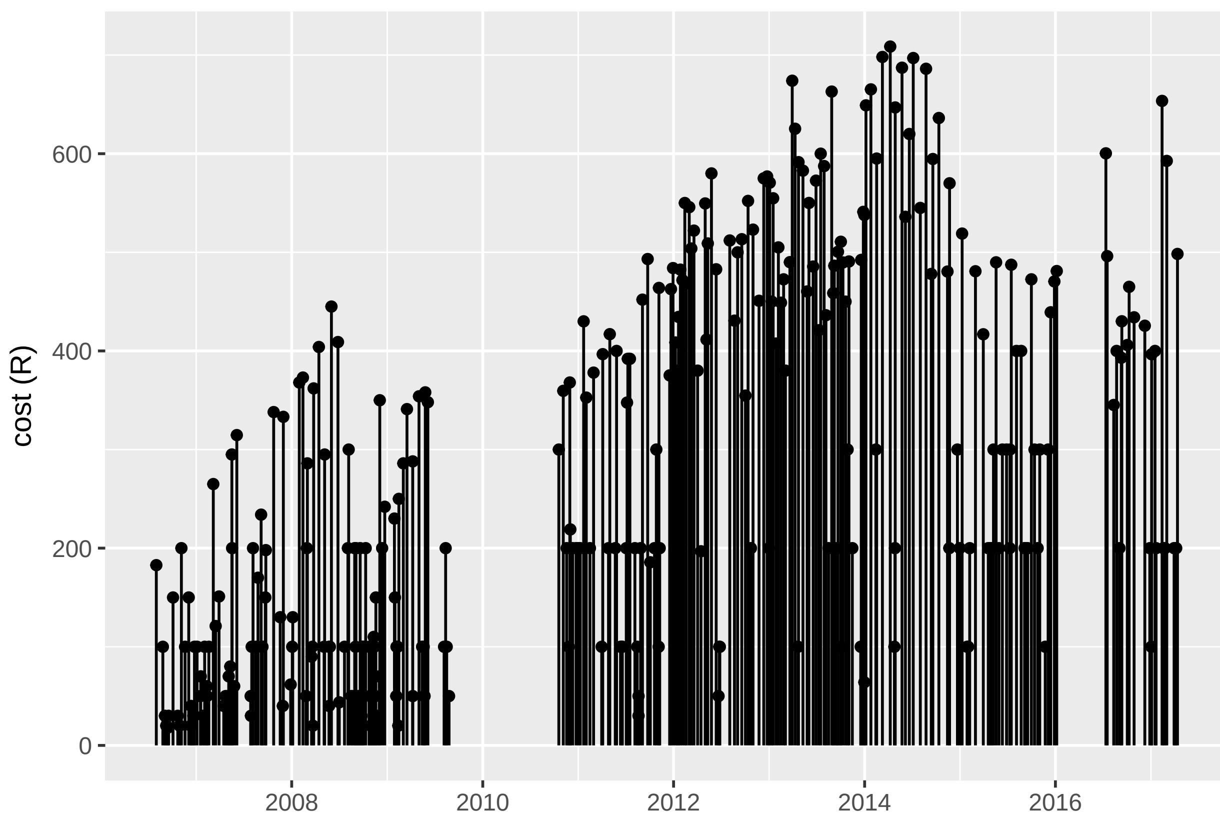


As we may see, the mileage appears to increase until 2013 when it then falls precipitously. The overall change in mileage for this car appears flat when modeled linearly.

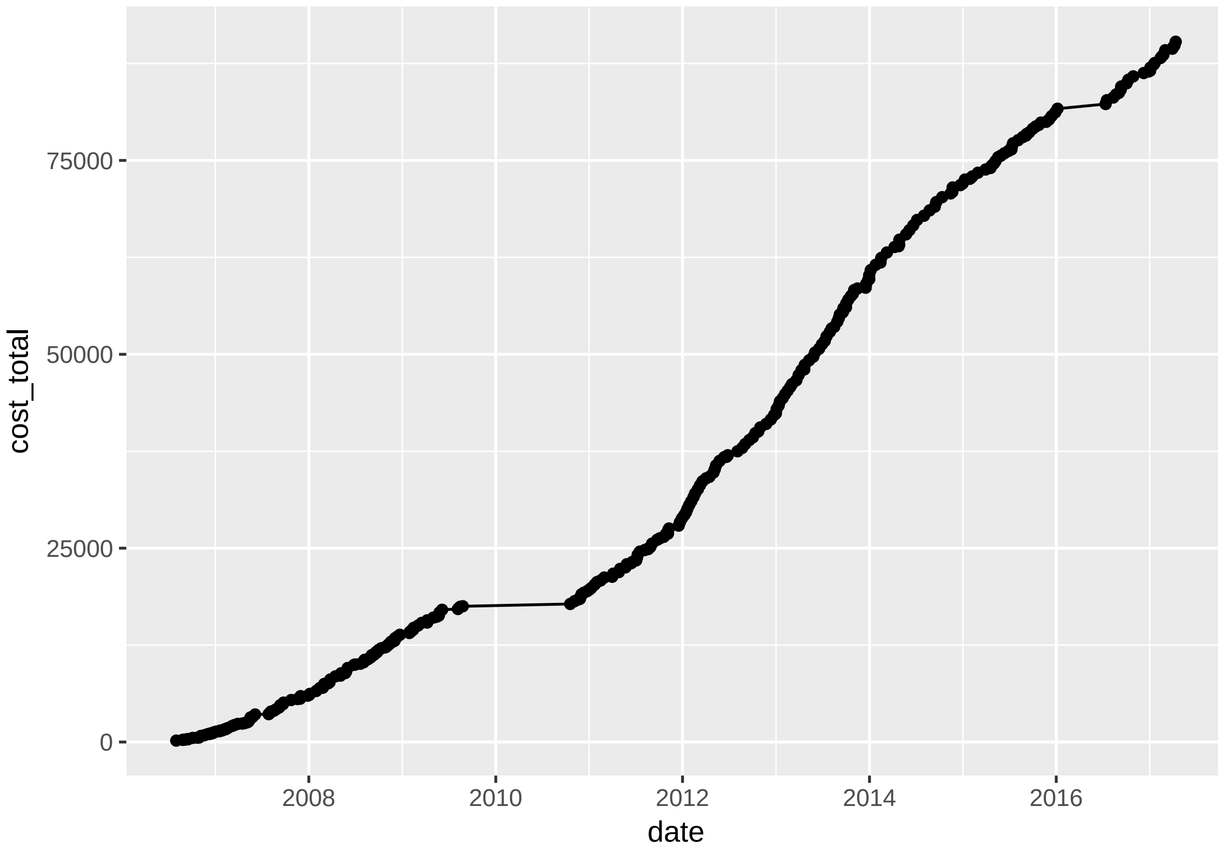
2.5 Petrol prices

Of interest to everyone is the price of petrol. Both how much we have spent and how much we have to spend every time we pull up to the station. First we see a lolliplot of spending behaviour.

```
# Calculate total amount spent
petrol$cost_total <- cumsum(petrol$cost)

# Lolli plot
ggplot(data = petrol, aes(x = date, y = cost)) +
  geom_point() +
  geom_segment(aes(xend = date, y = 0, yend = cost)) +
  labs(y = "cost (R)", x = "")
```




```
ggplot(data = petrol, aes(x = date, y = cost_total)) +
  geom_line() +
  geom_point()
```



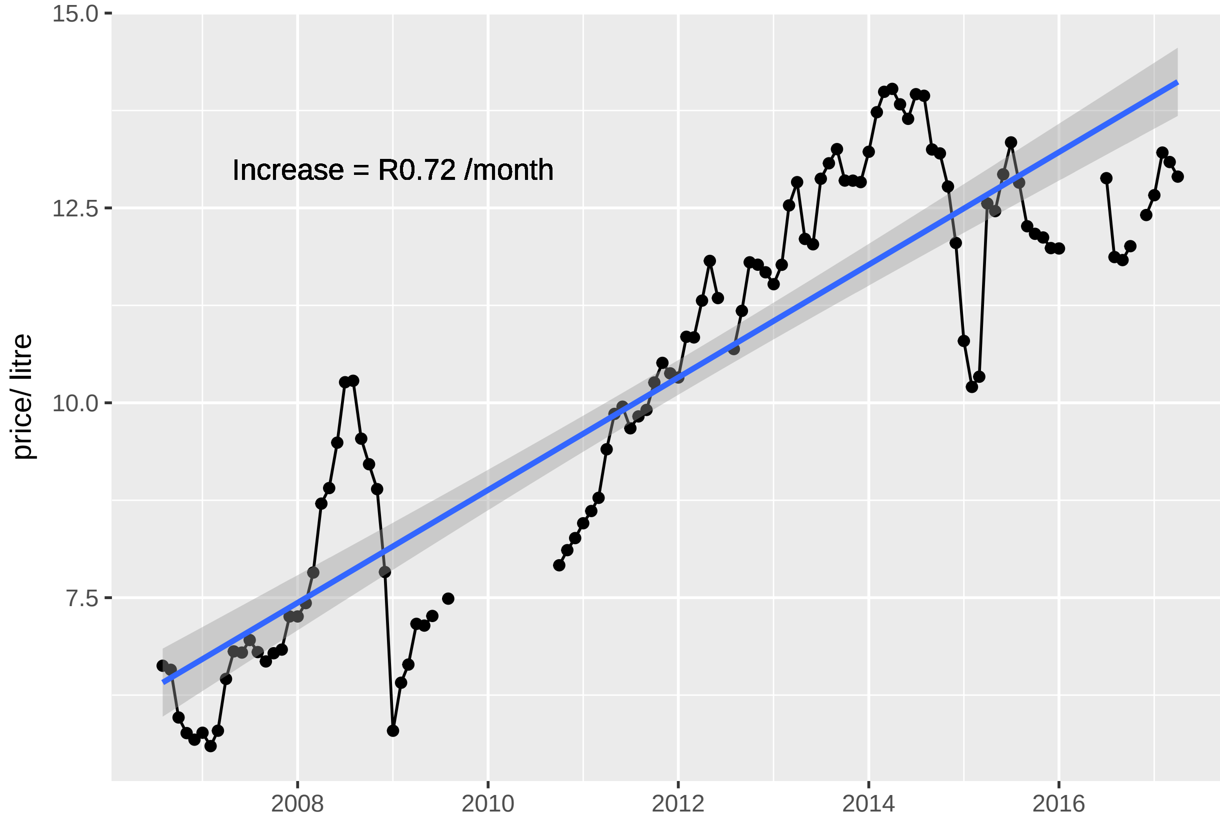
And then the price per litre averaged per month.

```
# Price/ litre/ month
petrol_monthly <- petrol %>%
  select(-full, -date, -year) %>%
  group_by(month) %>%
  summarise_all(mean) %>%
  mutate(price_litre = cost/litre)

# Fill in missing months
month_index <- data.frame(month = seq(petrol_monthly$month[1], petrol_monthly$month[nrow(petrol_monthly)
petrol_monthly <- merge(petrol_monthly, month_index, by = "month", all.y = TRUE)

petrol_trend <- lm(petrol_monthly$price_litre ~ seq(1:nrow(petrol_monthly)))
petrol_augment <- augment(petrol_trend)
petrol_tidy <- tidy(petrol_trend)
petrol_glance <- glance(petrol_trend)
petrol_tidy$estimate[2]*12
```

```
## [1] 0.7226146
```

```
# R 0.72/ month

# Line graph
ggplot(data = petrol_monthly, aes(x = month, y = price_litre)) +
  geom_line() +
  geom_point() +
  geom_smooth(method = "lm") +
```

```
geom_text(aes(x = as.Date("2009-01-01"), y = 13,
              label = paste0("Increase = R", round(petrol_tidy$estimate[2]*12, 2), " /month"))) +
# geom_point(data = petrol[187,], colour = "red")
# scale_y_continuous(breaks = seq(80000, 200000, 20000)) +
labs(y = "price/ litre", x = "")
```



Chapter 3

Bananas

On my farm we have a total of 40 hectares of land. Of this 18 hectares is natural forest with 16 hectares of sugar can and 6 hectares of bananas. Bananas grow throughout the year and from sucker to fruit is approximately 18 months. Under field management practices we are able to maintain three stages of banana trees on each spot thereby decreasing the time to fruit to six month intervals. It is recommended that every 10 years the field is to be replanted to maximize production however, these banana fields have not been replanted recently and it is our intention to investigate what this may mean for the production.

3.1 Loading the data

I transcribe my raw data into Microsoft Excel from the books. For me this is the easiest way to meticulously enter data and when the data sets are not to large finding errors can be done by changing the sort and filter functions within MS Excel. Once I am ready to import data into RStudio the file is saved as a `.csv` file using the MS Excel drop down option in the save menu. Data are easiest to work with when it is in long format, i.e.. each row represents a single observation. This is not crucial because it can be transformed using R.

```
# Load the relevant packages for loading and manipulaiton
library(tidyverse)

# Read in the data using `readr` from the `tidyverse` package
production <- read_csv("data/Banana Production.csv")

# A quick look to make sure the data looks like we expect
head(production)
```

```
## # A tibble: 6 x 12
##       Date Field Bunches    XL     L     M Boxes `Box/Bunch` weight `%XL`
##      <chr> <int>   <int> <int> <int> <int> <int>       <dbl>  <int> <dbl>
## 1  12/3/07   102     162    28    23     4    55        0.34    990  50.9
## 2 12/10/07   101     134    37    51    12   100        0.75   1800  37.0
## 3 12/18/07    93     115    17    47    19    83        0.72   1494  20.5
## 4 12/18/07    96      12     2     7     2    11        0.92    198  18.2
## 5 12/18/07  1011     145    25    46    19    90        0.62   1620  27.8
## 6 12/18/07   102      37     7     7     2    16        0.43    288  43.8
## # ... with 2 more variables: `%L` <chr>, `%M` <dbl>
```

Once the data are loaded and looks like the right stuff I get on to making sure my columns are set as either `date`, `factor`, or `number`. There are multiple ways to do this but I like to use **lubridate** (Grolemund et al.,

2016) when working with dates and the **Tidyverse** group namely **dplyr** (Wickham et al., 2017) for creating factors.

3.1.1 Setting date

```
# Load lubridate to play with the data values
library(lubridate)

# Making the date coloumn actual date values
production$Date <- as.Date(production$Date, "%m/%d/%y")

# I might want to have the month and year as unique values so I have created floor dates for each
# Add month and year columns
production$month <- floor_date(production$Date, "month")
production$year <- floor_date(production$Date, "year")
```

3.1.2 Working with the data in tidyverse

The fields were recorded as numbers but I would rather have the prefixed by 'f' so that I do not confuse them with production.

```
# I have used the pipe function from the package dply alongside the stringr package to do this
dat<-production %>%
  mutate(Field = stringr::str_replace(Field, "93", "f93"),
         Field = stringr::str_replace(Field, "96", "f96"),
         Field = stringr::str_replace(Field, "101", "f101"),
         Field = stringr::str_replace(Field, "1011", "f1011"),
         Field = stringr::str_replace(Field, "102", "f102"))
```

3.2 Plotting bunches per month

3.2.1 Bunches per month

If we were to try an plot the data for number of bunches harvested per month we can start to see that there is some kind of cyclic trend (Figure 3.1). On the plot I added a smooth (a model) in blue but there seems to be a problem with what it is doing. To have a look at what this problem might be I will inspect the data frame and trouble shoot

```
names(dat)
```

```
##  [1] "Date"      "Field"     "Bunches"   "XL"        "L"
##  [6] "M"         "Boxes"     "Box/Bunch" "weight"    "%XL"
## [11] "%L"        "%M"        "month"     "year"
```

```
ggplot(data = dat, aes(x = month, y = Bunches)) +
  geom_bar(stat = "identity") +
  geom_smooth(method = "loess", span = 0.2) +
  scale_x_date(date_breaks = "3 month", date_labels = "%b %y") +
  theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time")
```

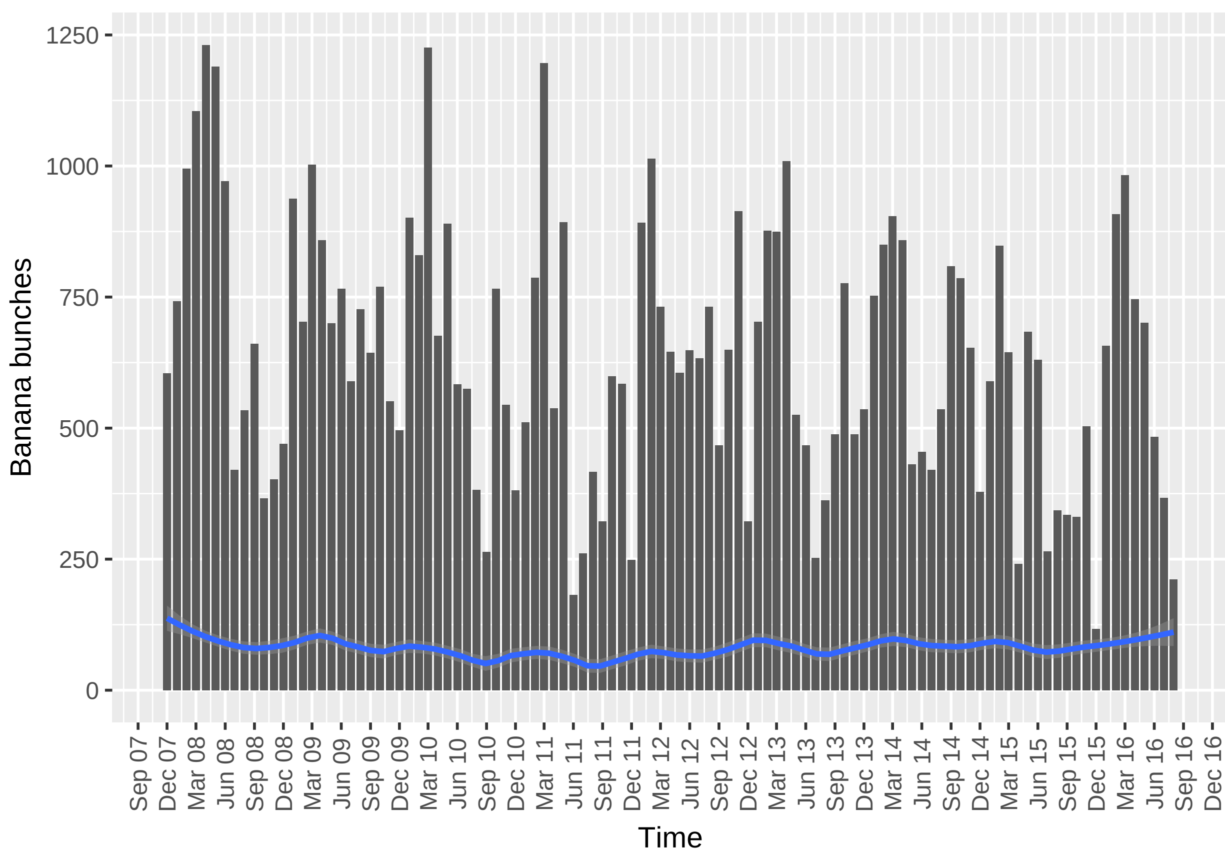


Figure 3.1: Bar graph showing the number of banana bunches harvested per month.

3.2.1.1 Problematic smooth

```
# The str function allows you to see both the variable type and the values.
str(dat)
```

```
## Classes 'tbl_df', 'tbl' and 'data.frame':    836 obs. of  14 variables:
##  $ Date     : Date, format: "2007-12-03" "2007-12-10" ...
##  $ Field    : chr  "f102" "f101" "f93" "f96" ...
##  $ Bunches  : int  162 134 115 12 145 37 95 13 96 71 ...
##  $ XL       : int  28 37 17 2 25 7 27 6 22 11 ...
##  $ L        : int  23 51 47 7 46 7 19 14 29 17 ...
##  $ M        : int  4 12 19 2 19 2 3 1 5 3 ...
##  $ Boxes    : int  55 100 83 11 90 16 49 21 56 31 ...
##  $ Box/Bunch: num  0.34 0.75 0.72 0.92 0.62 0.43 0.52 1.62 0.58 0.44 ...
##  $ weight   : int  990 1800 1494 198 1620 288 882 378 1008 558 ...
##  $ %XL      : num  50.9 37 20.5 18.2 27.8 43.8 55.1 28.6 39.3 35.5 ...
##  $ %L       : chr  "41.8" "51" "56.6" "63.6" ...
##  $ %M       : num  7.27 12 22.89 18.18 21.11 ...
##  $ month    : Date, format: "2007-12-01" "2007-12-01" ...
##  $ year     : Date, format: "2007-01-01" "2007-01-01" ...
```

```
# Open the dat data frame from the environment panel
```

After looking at both the `str` and the data frame the problem is that `Bunches` are not being summed by month. This is an easy fix

3.2.2 Bunches per month

If we want to create a summary data set to only include the sum total of banana bunches per month we can simply use `dplyr` and the pipe function. This creates a sum total for bunches.month^{-1} and the blue line now fits the plot more appropriately (Figure 3.2)

```
names(dat)
```

```
##  [1] "Date"      "Field"     "Bunches"   "XL"        "L"
##  [6] "M"         "Boxes"     "Box/Bunch" "weight"    "%XL"
## [11] "%L"        "%M"        "month"     "year"
```

```
# Using dply to group the data by month and year, create a new column for the sum of bunches picked per
dat.sum_m <-
  dat %>%
  group_by(month, year) %>%
  summarise(bunches.m = sum(Bunches)) %>%
  ungroup()
```

```
# Plotting the data

ggplot(data = dat.sum_m, aes(x = month, y = bunches.m)) +
  geom_bar(stat = "identity") +
  geom_smooth(method = "loess", span = 0.2) +
  scale_x_date(date_breaks = "3 month", date_labels = "%b %y") +
  theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time")
```

From the trend observed in Figure 3.2 it seems apparent that there may be differences in the monthly output which we could look at.

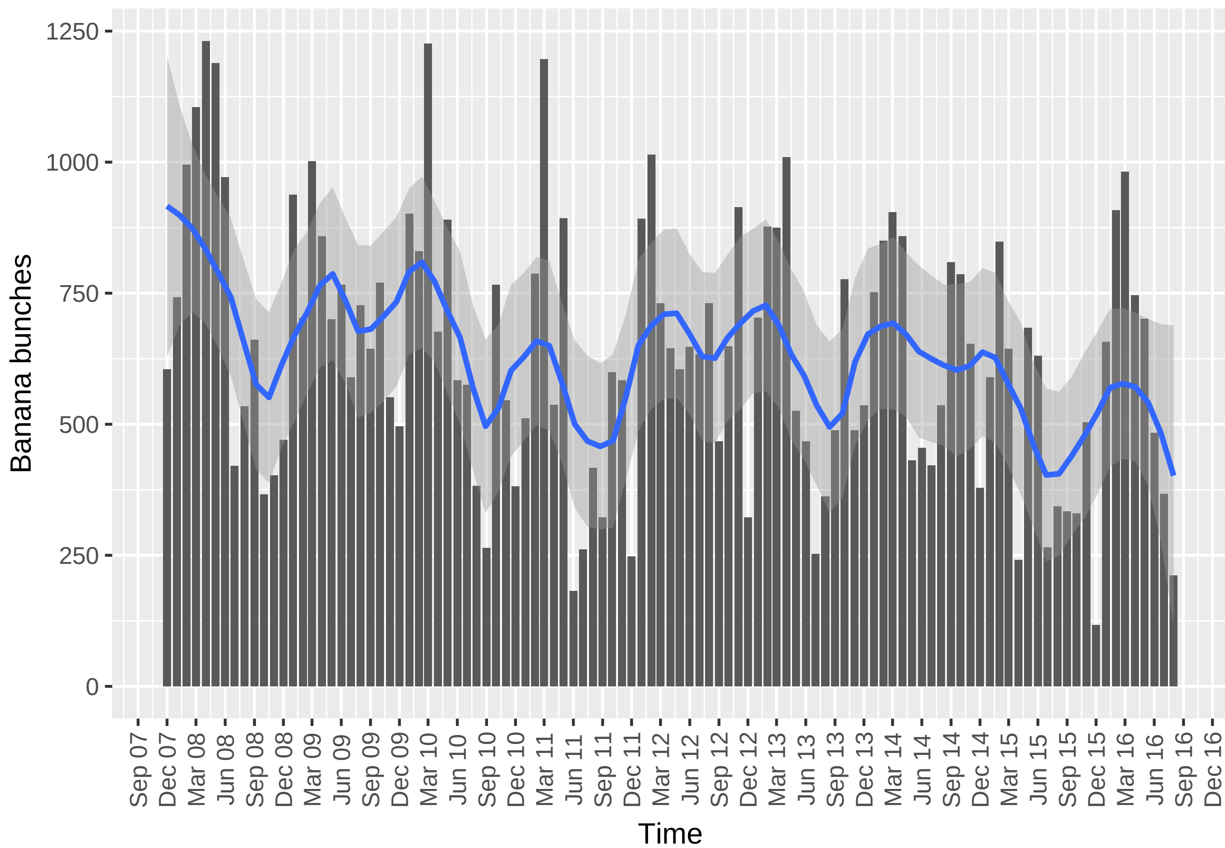


Figure 3.2: Bar graph showing the sum of banana bunches harvested per month with a smooth fitted in blue.

```
# I am going to modify an existing dataframe so I will assign it to 'a' as to not back track
 a <- dat.sum_m

# The month and year are going to be pulled out of the date and given their own coloumn
a$y<-year(a$year)
a$m<-month(a$month)

# If I want to use the month as anything other than a date I should tell R it is a factor
a1 <-
  a %>%
  mutate(m = as.factor(m))
```

Now that the data are ready to be looked at as bunches per month a simple box plot can tell a quick visual story

```
# a quick boxplot for the bunches harvested per month
ggplot(data = a1, aes(x = m, y = bunches.m)) +
  geom_boxplot() +
  geom_jitter() +
  labs(y = "Banana bunches", x = "Month")
```

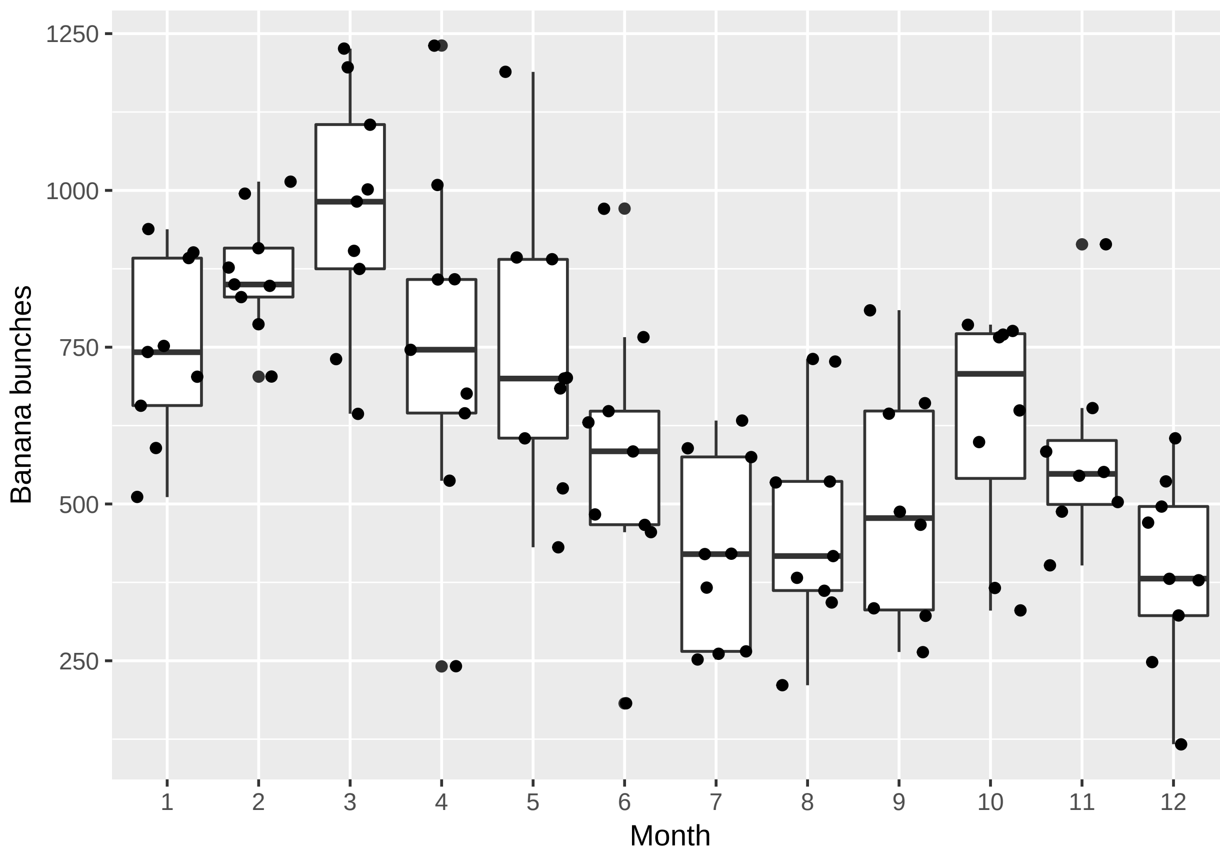


Figure 3.3: Boxplot showing the total banana bunches harvested per month.

3.2.3 Statistics

Are the number of bunches harvest per month statistically different?

To answer this we will run a quick one-way ANOVA

```
library(broom)
aov <- aov(bunches.m ~ m, data = a1)
summary(aov)
```

```
##             Df  Sum Sq Mean Sq F value   Pr(>F)    
## m           11 3159730  287248   8.133 8.66e-10 ***
## Residuals   93 3284786   35320                     
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
TukeyHSD(aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
## 
## Fit: aov(formula = bunches.m ~ m, data = a1)
## 
## $m
##              diff        lwr         upr     p adj
## 2-1    125.222222 -171.81647  422.260918 0.9579975
## 3-1    220.000000  -77.03870  517.038696 0.3634513
## 4-1     12.888889 -284.14981  309.927585 1.0000000
## 5-1     -7.444444 -304.48314  289.594252 1.0000000
## 6-1   -166.555556 -463.59425  130.483141 0.7682453
## 7-1   -322.444444 -619.48314  -25.405748 0.0215313
## 8-1   -271.333333 -568.37203   25.705363 0.1075045
## 9-1   -244.152778 -550.33326   62.027702 0.2560930
## 10-1  -112.527778 -418.70826  193.652702 0.9848828
## 11-1  -162.777778 -468.95826  143.402702 0.8231617
## 12-1  -348.000000 -645.03870  -50.961304 0.0085543
## 3-2     94.777778 -202.26092  391.816474 0.9952604
## 4-2   -112.333333 -409.37203  184.705363 0.9811249
## 5-2   -132.666667 -429.70536  164.372029 0.9376806
## 6-2   -291.777778 -588.81647    5.260918 0.0589203
## 7-2   -447.666667 -744.70536 -150.627971 0.0001326
## 8-2   -396.555556 -693.59425  -99.516859 0.0012407
## 9-2   -369.375000 -675.55548  -63.194520 0.0057736
## 10-2  -237.750000 -543.93048   68.430480 0.2931045
## 11-2  -288.000000 -594.18048   18.180480 0.0854044
## 12-2  -473.222222 -770.26092 -176.183526 0.0000407
## 4-3   -207.111111 -504.14981   89.927585 0.4582263
## 5-3   -227.444444 -524.48314   69.594252 0.3134050
## 6-3   -386.555556 -683.59425  -89.516859 0.0018783
## 7-3   -542.444444 -839.48314 -245.405748 0.0000014
## 8-3   -491.333333 -788.37203 -194.294637 0.0000172
## 9-3   -464.152778 -770.33326 -157.972298 0.0001176
## 10-3  -332.527778 -638.70826  -26.347298 0.0214159
## 11-3  -382.777778 -688.95826  -76.597298 0.0034716
## 12-3  -568.000000 -865.03870 -270.961304 0.0000004
## 5-4    -20.333333 -317.37203  276.705363 1.0000000
## 6-4   -179.444444 -476.48314  117.594252 0.6751512
## 7-4   -335.333333 -632.37203  -38.294637 0.0136375
## 8-4   -284.222222 -581.26092   12.816474 0.0740955
## 9-4   -257.041667 -563.22215   49.138813 0.1911989
```

```
## 10-4  -125.416667 -431.59715  180.763813 0.9657731
## 11-4  -175.666667 -481.84715  130.513813 0.7415294
## 12-4  -360.888889 -657.92759  -63.850193 0.0052335
## 6-5   -159.111111 -456.14981  137.927585 0.8160043
## 7-5   -315.000000 -612.03870  -17.961304 0.0277875
## 8-5   -263.888889 -560.92759   33.149807 0.1317895
## 9-5   -236.708333 -542.88881   69.472146 0.2994160
## 10-5  -105.083333 -411.26381  201.097146 0.9912727
## 11-5  -155.333333 -461.51381  150.847146 0.8632505
## 12-5  -340.555556 -637.59425  -43.516859 0.0112750
## 7-6   -155.888889 -452.92759  141.149807 0.8349898
## 8-6   -104.777778 -401.81647  192.260918 0.9890943
## 9-6    -77.597222 -383.77770  228.583258 0.9994023
## 10-6    54.027778 -252.15270  360.208258 0.9999830
## 11-6     3.777778 -302.40270  309.958258 1.0000000
## 12-6  -181.444444 -478.48314  115.594252 0.6598402
## 8-7     51.111111 -245.92759  348.149807 0.9999868
## 9-7     78.291667 -227.88881  384.472146 0.9993502
## 10-7   209.916667  -96.26381  516.097146 0.4849922
## 11-7   159.666667 -146.51381  465.847146 0.8406007
## 12-7   -25.555556 -322.59425  271.483141 1.0000000
## 9-8     27.180556 -278.99992  333.361035 1.0000000
## 10-8   158.805556 -147.37492  464.986035 0.8452559
## 11-8   108.555556 -197.62492  414.736035 0.9886365
## 12-8   -76.666667 -373.70536  220.372029 0.9992910
## 10-9   131.625000 -183.43211  446.682115 0.9605881
## 11-9    81.375000 -233.68211  396.432115 0.9992863
## 12-9  -103.847222 -410.02770  202.333258 0.9920825
## 11-10  -50.250000 -365.30711  264.807115 0.9999940
## 12-10 -235.472222 -541.65270   70.708258 0.3070082
## 12-11 -185.222222 -491.40270  120.958258 0.6732661
```

```r
# The tidyed
tidy(aov)
```

```
##        term df   sumsq    meansq statistic      p.value
## 1         m 11 3159730 287248.20  8.132671 8.659346e-10
## 2 Residuals 93 3284786  35320.28        NA           NA
```

```r
tidy(TukeyHSD(aov))
```

```
##    term comparison    estimate   conf.low   conf.high  adj.p.value
## 1     m        2-1  125.222222 -171.81647  422.260918 9.579975e-01
## 2     m        3-1  220.000000  -77.03870  517.038696 3.634513e-01
## 3     m        4-1   12.888889 -284.14981  309.927585 1.000000e+00
## 4     m        5-1   -7.444444 -304.48314  289.594252 1.000000e+00
## 5     m        6-1 -166.555556 -463.59425  130.483141 7.682453e-01
## 6     m        7-1 -322.444444 -619.48314  -25.405748 2.153135e-02
## 7     m        8-1 -271.333333 -568.37203   25.705363 1.075045e-01
## 8     m        9-1 -244.152778 -550.33326   62.027702 2.560930e-01
## 9     m       10-1 -112.527778 -418.70826  193.652702 9.848828e-01
## 10    m       11-1 -162.777778 -468.95826  143.402702 8.231617e-01
## 11    m       12-1 -348.000000 -645.03870  -50.961304 8.554275e-03
## 12    m        3-2   94.777778 -202.26092  391.816474 9.952604e-01
## 13    m        4-2 -112.333333 -409.37203  184.705363 9.811249e-01
## 14    m        5-2 -132.666667 -429.70536  164.372029 9.376806e-01
```

```
## 15    m         6-2 -291.777778 -588.81647    5.260918 5.892033e-02
## 16    m         7-2 -447.666667 -744.70536 -150.627971 1.325711e-04
## 17    m         8-2 -396.555556 -693.59425  -99.516859 1.240745e-03
## 18    m         9-2 -369.375000 -675.55548  -63.194520 5.773603e-03
## 19    m        10-2 -237.750000 -543.93048   68.430480 2.931045e-01
## 20    m        11-2 -288.000000 -594.18048   18.180480 8.540437e-02
## 21    m        12-2 -473.222222 -770.26092 -176.183526 4.065441e-05
## 22    m         4-3 -207.111111 -504.14981   89.927585 4.582263e-01
## 23    m         5-3 -227.444444 -524.48314   69.594252 3.134050e-01
## 24    m         6-3 -386.555556 -683.59425  -89.516859 1.878265e-03
## 25    m         7-3 -542.444444 -839.48314 -245.405748 1.398298e-06
## 26    m         8-3 -491.333333 -788.37203 -194.294637 1.720851e-05
## 27    m         9-3 -464.152778 -770.33326 -157.972298 1.176102e-04
## 28    m        10-3 -332.527778 -638.70826  -26.347298 2.141590e-02
## 29    m        11-3 -382.777778 -688.95826  -76.597298 3.471610e-03
## 30    m        12-3 -568.000000 -865.03870 -270.961304 3.833872e-07
## 31    m         5-4  -20.333333 -317.37203  276.705363 1.000000e+00
## 32    m         6-4 -179.444444 -476.48314  117.594252 6.751512e-01
## 33    m         7-4 -335.333333 -632.37203  -38.294637 1.363754e-02
## 34    m         8-4 -284.222222 -581.26092   12.816474 7.409549e-02
## 35    m         9-4 -257.041667 -563.22215   49.138813 1.911989e-01
## 36    m        10-4 -125.416667 -431.59715  180.763813 9.657731e-01
## 37    m        11-4 -175.666667 -481.84715  130.513813 7.415294e-01
## 38    m        12-4 -360.888889 -657.92759  -63.850193 5.233516e-03
## 39    m         6-5 -159.111111 -456.14981  137.927585 8.160043e-01
## 40    m         7-5 -315.000000 -612.03870  -17.961304 2.778753e-02
## 41    m         8-5 -263.888889 -560.92759   33.149807 1.317895e-01
## 42    m         9-5 -236.708333 -542.88881   69.472146 2.994160e-01
## 43    m        10-5 -105.083333 -411.26381  201.097146 9.912727e-01
## 44    m        11-5 -155.333333 -461.51381  150.847146 8.632505e-01
## 45    m        12-5 -340.555556 -637.59425  -43.516859 1.127501e-02
## 46    m         7-6 -155.888889 -452.92759  141.149807 8.349898e-01
## 47    m         8-6 -104.777778 -401.81647  192.260918 9.890943e-01
## 48    m         9-6  -77.597222 -383.77770  228.583258 9.994023e-01
## 49    m        10-6   54.027778 -252.15270  360.208258 9.999830e-01
## 50    m        11-6    3.777778 -302.40270  309.958258 1.000000e+00
## 51    m        12-6 -181.444444 -478.48314  115.594252 6.598402e-01
## 52    m         8-7   51.111111 -245.92759  348.149807 9.999868e-01
## 53    m         9-7   78.291667 -227.88881  384.472146 9.993502e-01
## 54    m        10-7  209.916667  -96.26381  516.097146 4.849922e-01
## 55    m        11-7  159.666667 -146.51381  465.847146 8.406007e-01
## 56    m        12-7  -25.555556 -322.59425  271.483141 1.000000e+00
## 57    m         9-8   27.180556 -278.99992  333.361035 1.000000e+00
## 58    m        10-8  158.805556 -147.37492  464.986035 8.452559e-01
## 59    m        11-8  108.555556 -197.62492  414.736035 9.886365e-01
## 60    m        12-8  -76.666667 -373.70536  220.372029 9.992910e-01
## 61    m        10-9  131.625000 -183.43211  446.682115 9.605881e-01
## 62    m        11-9   81.375000 -233.68211  396.432115 9.992863e-01
## 63    m        12-9 -103.847222 -410.02770  202.333258 9.920825e-01
## 64    m       11-10  -50.250000 -365.30711  264.807115 9.999940e-01
## 65    m       12-10 -235.472222 -541.65270   70.708258 3.070082e-01
## 66    m       12-11 -185.222222 -491.40270  120.958258 6.732661e-01
```

3.3 Plotting bunches per field

```
# Creating a cum for each month grouped by field
names(dat)
```

```
##  [1] "Date"      "Field"     "Bunches"   "XL"        "L"
##  [6] "M"         "Boxes"     "Box/Bunch" "weight"    "%XL"
## [11] "%L"        "%M"        "month"     "year"
```

```
dat.sum_m.f <-
  dat %>%
  group_by(month, year, Field) %>%
  summarise(bunches.m.f = sum(Bunches))
```

As described earlier there are different fields which are picked from. A look at the production of bunches by field in Figure 3.4 highlights the fact the two fields (f102 and f93) were taken out of production.

```
ggplot(data = dat.sum_m.f, aes(x = month, y = bunches.m.f)) +
  geom_bar(stat = "identity") +
  geom_smooth(method = "loess", span = 0.2) +
  scale_x_date(date_breaks = "12 month", date_labels = "%y") +
  theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time") +
  facet_wrap(~Field, ncol = 2)
```

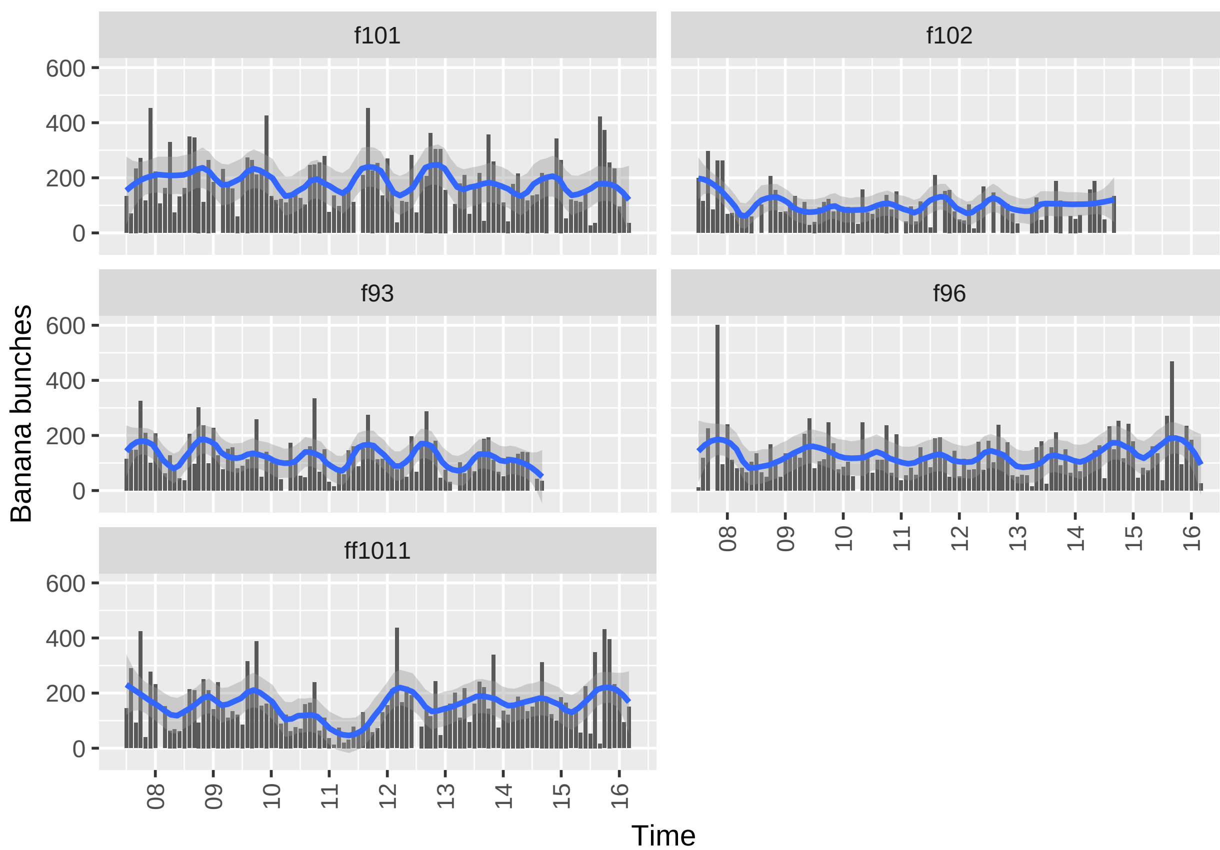


Figure 3.4: Bar graph showing the sum of banana bunches harvested per month per field with a smooth fitted in blue.

3.3.1 Cumulative bunches

It seems interesting to look at a field as a continuous unit and measure the cumulative harvest over time.

```
# Creating a cumulative bunches harvest for each field
names(dat)
```

```
##  [1] "Date"       "Field"      "Bunches"    "XL"         "L"
##  [6] "M"          "Boxes"      "Box/Bunch" "weight"     "%XL"
## [11] "%L"         "%M"         "month"      "year"
```

```
dat.sum_f <-
  dat %>%
  group_by(Field) %>%
  mutate(cumsum = cumsum(Bunches))
```

In Figure 3.5 the cumulative number of bunches harvested for each field highlights that they are not all performing the same. We could quickly add a linear model to this to further visualize the trend.

```
ggplot(data = dat.sum_f, aes(x = Date, y = cumsum, colour = Field)) +
  geom_line() +
  #geom_bar(stat = "identity") +
  #geom_smooth(method = "lm") +
  #scale_x_date(date_breaks = "12 month", date_labels = "%y") +
  #theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time")
```

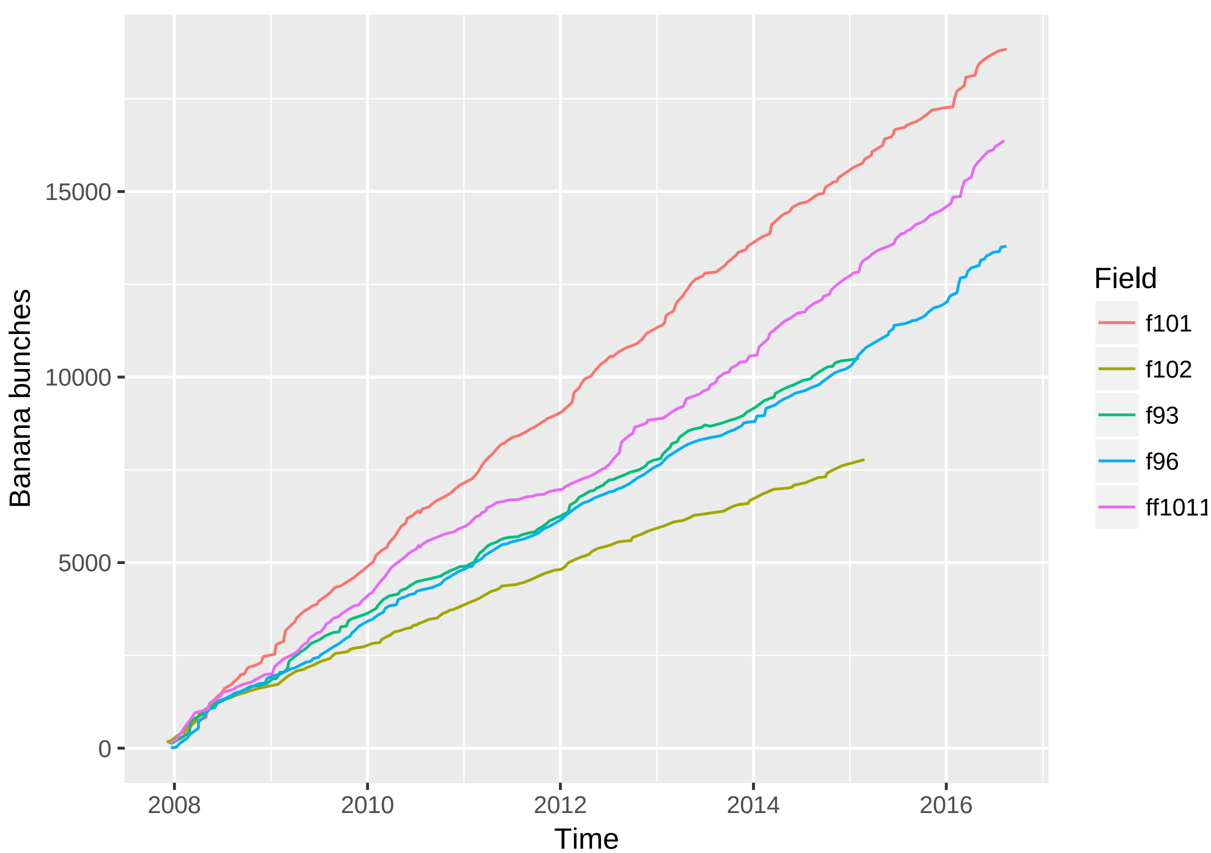


Figure 3.5: Line graph showing the cumulative harvest for banana bunches per field.

```
  #facet_wrap(~Field, ncol = 2)
```

3.3.2 Data normalization

Fields are arbitrary units which are marked out and cultivated. They are divided into smaller manageable units which are in close proximity to each other. For this reason it is necessary to normalize the production by field size so that the values can be compared.

We know that the field sizes are:

- 93 - 1.51 ha
- 96 - 1.31 ha
- 101 - 2.07 ha
- 102 - 1.12 ha
- 1011 - 2.21 ha

```
# Its nice to have the coloumn names handy
names(dat.sum_f)
```

```
##  [1] "Date"      "Field"     "Bunches"   "XL"        "L"
##  [6] "M"         "Boxes"     "Box/Bunch" "weight"    "%XL"
## [11] "%L"        "%M"        "month"     "year"      "cumsum"
```

```
# here we will have to spread the data to wide format, create new coloumns for production/ha, gather th
norm.dat.w <- dat.sum_f %>%
  spread(key = Field, value = cumsum) %>%
  group_by(month) %>%
  mutate(ha.93 = f93 /1.51,
         ha.96 = f96 /1.31,
         ha.101 = f101/2.07,
         ha.1011 = ff1011/2.21,
         ha.102 = f102/1.12)

norm.dat.l <-
  norm.dat.w %>%
  select(month, ha.93, ha.96, ha.101, ha.1011, ha.102) %>%
  gather(field, tonnage, 2:6)%>%
  drop_na()

glimpse(norm.dat.l)
```

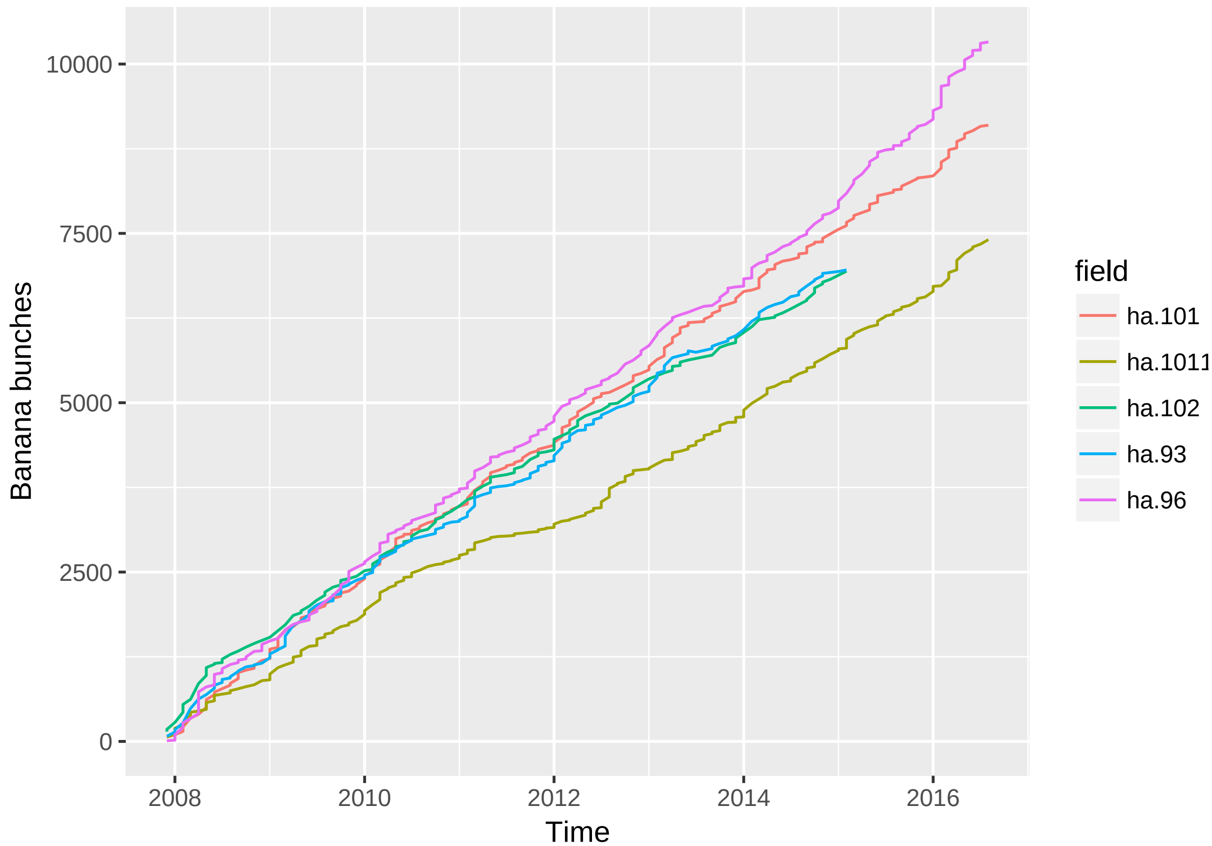
```
## Observations: 836
## Variables: 3
## $ month   <date> 2007-12-01, 2008-01-01, 2008-01-01, 2008-02-01, 2008-...
## $ field   <chr> "ha.93", "ha.93", "ha.93", "ha.93", "ha.93", "ha.93", ...
## $ tonnage <dbl> 76.15894, 139.07285, 173.50993, 270.86093, 486.09272, ...
```

```
names(norm.dat.l)
```

```
## [1] "month"   "field"   "tonnage"
```

```
ggplot(data = norm.dat.l, aes(x = month, y = tonnage, colour = field)) +
  geom_line() +
  #geom_bar(stat = "identity") +
  #geom_smooth(method = "lm") +
  #geom_smooth(method = "loess", span = 0.1) +
  #scale_x_date(date_breaks = "12 month", date_labels = "%y") +
```

```
  #theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time")
```




```
  #facet_wrap(~field, ncol = 2)
```

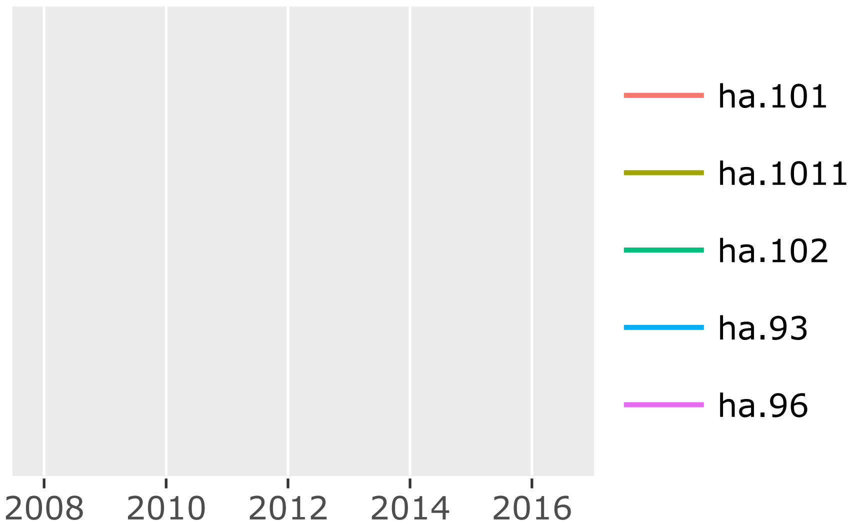
The awesome thing about this is that you can very easily turn a plot into something more than just a plot using `plotly`

```
#devtools::install_github("ropensci/plotly")
#devtools::install_github("hadley/ggplot2")
library(plotly)
```

```
##
## Attaching package: 'plotly'
## The following object is masked from 'package:ggplot2':
##
##     last_plot
## The following object is masked from 'package:stats':
##
##     filter
## The following object is masked from 'package:graphics':
##
##     layout
```

```
plotly <- ggplot(data = norm.dat.l, aes(x = month, y = tonnage, colour = field)) +
  geom_line() +
  #geom_bar(stat = "identity") +
```

```r
  geom_smooth(method = "lm") +
  #geom_smooth(method = "loess", span = 0.1) +
  #scale_x_date(date_breaks = "12 month", date_labels = "%y") +
  #theme(axis.text.x = element_text(angle=90, vjust = 0.5)) +
  labs(y = "Banana bunches", x = "Time")
  #facet_wrap(~field, ncol = 2)

ggplotly(plotly)
```




```r
library(nlme)
```

```
##
## Attaching package: 'nlme'
## The following object is masked from 'package:dplyr':
##
##     collapse
```

```r
mountain.lm <- lm(tonnage ~ field, data = norm.dat.l)
tidy(mountain.lm)
```

```
##           term   estimate std.error statistic       p.value
## 1  (Intercept)  4725.6408  165.7384 28.512656 3.051626e-125
## 2 fieldha.1011 -1111.6938  235.6071 -4.718423  2.787814e-06
## 3  fieldha.102 -1047.6083  275.6327 -3.800740  1.548000e-04
## 4   fieldha.93 -1013.5162  253.6645 -3.995499  7.028677e-05
## 5   fieldha.96   361.3621  235.9184  1.531725  1.259709e-01
```

Chapter 4

Final Words

We have finished a nice book.

Bibliography

Grolemund, G., Spinu, V., and Wickham, H. (2016). *lubridate: Make Dealing with Dates a Little Easier*. R package version 1.6.0.

Wickham, H., Francois, R., Henry, L., and Müller, K. (2017). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.0.

Xie, Y. (2017). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.4.