We R Under Way: A Data Science Portfolio

Kaydee S. Barker

Contents

Writing and code by Kaydee Barker, assignments by Dr. Ross and Dr. Mueller (SOCR 580A7), Dr. Lefsky (ESS 330) of Colorado State University. Data cited within chapters.

Chapter 1

Introduction

> "There are two kinds of data scientists: 1) Those who can extrapolate from incomplete data."

I began my foray into R in the spring of 2020, first teaching myself some basic syntax and then using it for statistical analysis on my research projects as an undergraduate researcher at Colorado State University (CSU). With the help of my research mentors and many amazing people of the internet, I was able to fumble my way forward and learn a number of techniques to analyze and visualize data in R. I have since been building on my R and data science skills, including with the help of two key courses at CSU: "Quantitative Reasoning for Ecosystem Science" (ESS 330) and "Introduction to Environmental Data Science" (SOCR 580A7). Since I can't yet publish data from my research projects, this portfolio is constructed of public data examples, primarily from my coursework in those two courses. Its purpose is a) to serve as a reference for myself and others learning to use R for environmental analyses, and b) to demonstrate my current R knowledge to advisors and colleagues.

Chapter 2

Interactive Graphing: Discharge of the Poudre River

> "Someone asked me to name two structures that hold water. I was like, 'well… damn!' "

This assignment used a unique package of R Markdown (dygraphs) in order to create an interactive chart.

Data and assignment provided by Dr. Matthew Ross and Dr. Nathan Mueller of Colorado State University.

2.1 Background on the Poudre River

Cache La Poudre River is an important watershed that supports **agriculture, industry, recreation, and residential needs** on the Front Range of Colorado. It also provides for cottonwood forest, shrub, and grassland ecosystems that support wildlife from the mountains down to the prairies. The unique **biodiversity** and **history** of the Cache La Poudre watershed are valued widely; 45 miles along the Poudre are encompassed in a National Heritage Area. The history of Cache La Poudre is linked to the *history of the West*, because its banks supported the first major irrigation-based agricultural settlement of its kind in 1870, which would soon spread through the Arid West.

2.2 Interactive Discharge Chart

```
q <- readNWISdv(siteNumbers = '06752260',
                parameterCd = '00060',
                startDate = '2017-01-01',
                endDate = '2022-01-01') %>%
  rename(q = 'X_00060_00003')

q_xts <- xts(q$q, order.by = q$Date)

dygraph(q_xts) %>%
  dyAxis("y", label = "Discharge (cfs)") %>%
  dyOptions(drawPoints = TRUE, pointSize = 2)
```

```
## PhantomJS not found. You can install it with webshot::install_phantomjs(). If it is
```

Discharge of the Poudre River in cubic feet per second from January 2017 to December 2021.

Chapter 3

Looking at Effects of Fire on Vegetation

"What happens when a wildfire tells you a joke? You get burned!"

This assignment demonstrates the benefit of visualizing data to see potential correlations.

Data and assignment provided by Dr. Matthew Ross and Dr. Nathan Mueller of Colorado State University.

3.1 Introduction

The Hayman Fire, started by arsen in summer of 2002, was the largest wildfire in Colorado history until the 2020 wildfire season. It burned a large area of over 138 thousand acres between the Kenosha Mountains and Pikes Peak, affecting wildlife and causing water quality concerns for the Front Range populations through damage to watersheds that contribute to the South Platte River.

3.2 What is the correlation between NDVI and NDMI?

The Normalized Difference Vegetation Index (NDVI) is positively correlated with the Normalized Difference Moisture Index (NDMI). In everyday terms, NDVI indicates plant health as shown by how well leaves reflect near infrared and red light, while NDMI represents plant water content and is calculated from near infrared and short-wave infrared reflectance values (Agricolus, 2018).

These values can also tell us about how much vegetative cover there is at a given site, with the lowest NDVI (<0.1) and NDMI (<-0.8) values indicating bare soil.

Not surprisingly, the plot below shows that canopy cover is greatly decreased for the burned site compared to the unburned site.

```
#ggplot of wide set in summer
full_wide %>%
  filter(month %in% c(6,7,8,9,10)) %>%
  filter(year >= 2002) %>%
ggplot(., aes(x=ndmi,y=ndvi, color=treatment)) +
  geom_point(shape=1) +
  xlab("NDMI") + ylab("NDVI") +
  ggtitle("Burned vs. Unburned Vegetation") +
  theme_few(base_size = 16) +
  scale_color_brewer(palette = "Set2") +
  theme(panel.grid.major=element_blank(), panel.grid.minor=element_blank(), legend.pos
```

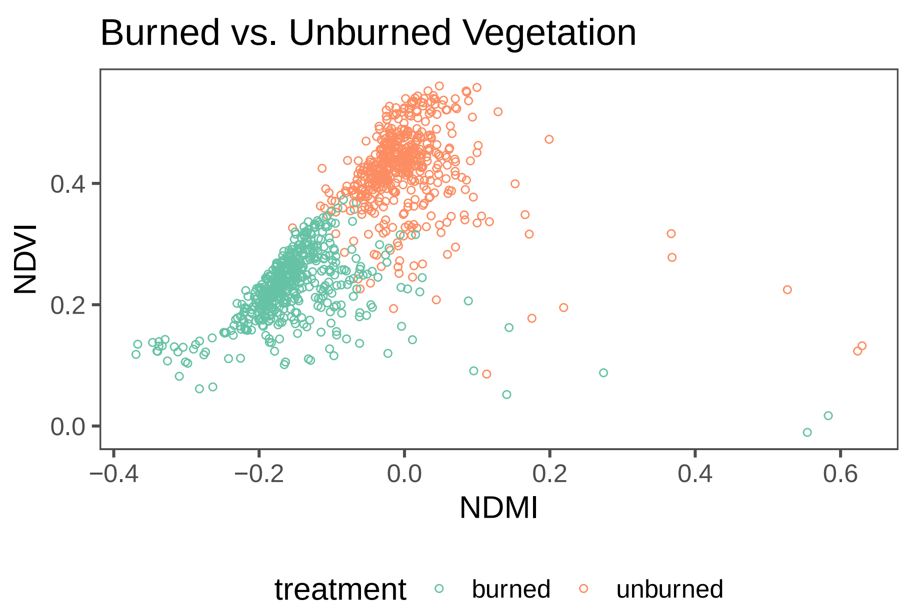


Figure 3.1: NDVI and NDMI values from 2002 to 2019 in Colorado sites that were burned (teal) or left unburned (orange) during the Hayman Fire.

As may be expected, vegetative growth (NDVI) is positively associated with the previous winter's snowfall, as shown in the plot below.

```
#ggplot winter NDSI to summer NDVI
ggplot(ndvi_ndsi, aes(x = mean_NDVI, y = mean_NDSI)) +
  geom_point(fill = "blue",
             shape = 21,
             size = 2) +
  geom_smooth(method = "lm",
              se = TRUE,
              lty = 1,
              color = "black",
              fill = "lightgrey",
              size = 1) +
  xlab("Mean NDSI") + ylab("Mean NDVI") +
  ggtitle("Winter NDSI vs. Summer NDVI") +
  theme_few(base_size = 16) +
  scale_y_continuous(breaks = pretty(c(-0.4,0.5), n = 4)) +
  scale_x_continuous(breaks = pretty(c(0.2,0.5), n = 6)) +
  theme(panel.grid.major=element_blank(), panel.grid.minor=element_blank(), legend.position="bott
```

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

3.3 What month is the greenest month on average?

If we plot monthly means of NDVI, we can see that the greenest month in Colorado is August.

```
#ggplot of monthly means
monthly_sum %>%
  filter(data == "ndvi") %>%
  mutate_at(vars(month), funs(factor)) %>%
ggplot(., aes(x=month, y=value_mean, fill=month)) +
  geom_bar(stat = "identity", width = 0.7, position = "dodge") +
  geom_errorbar(aes(ymin=value_mean-value_std.error, ymax=value_mean+value_std.error),
                colour = "black", width = 0.7, position = "dodge") +
  scale_x_discrete(labels=c("5"="May", "6"="June", "7"="Jul.", "8"="Aug.", "9"="Sept.")) +
  xlab("Month") +  ylab("NDVI") +
  ggtitle("Average NDVI per Month") +
  theme_few() +
  scale_fill_brewer(palette = "Greens") +
  theme(panel.grid.major=element_blank(),
        panel.grid.minor=element_blank(), legend.position="none")
```

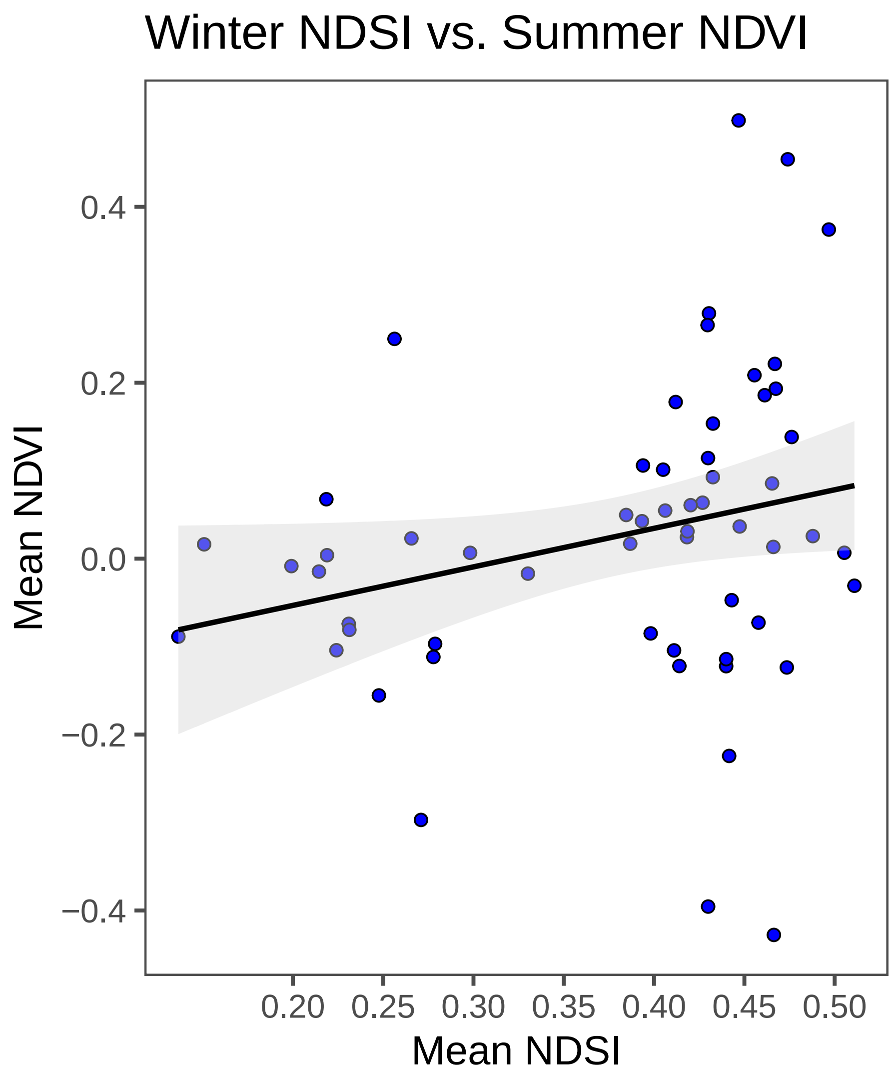


Figure 3.2: Linear models for mean summer NDVI and mean winter NDSI for pre- and post-burn and burned and unburned sites.

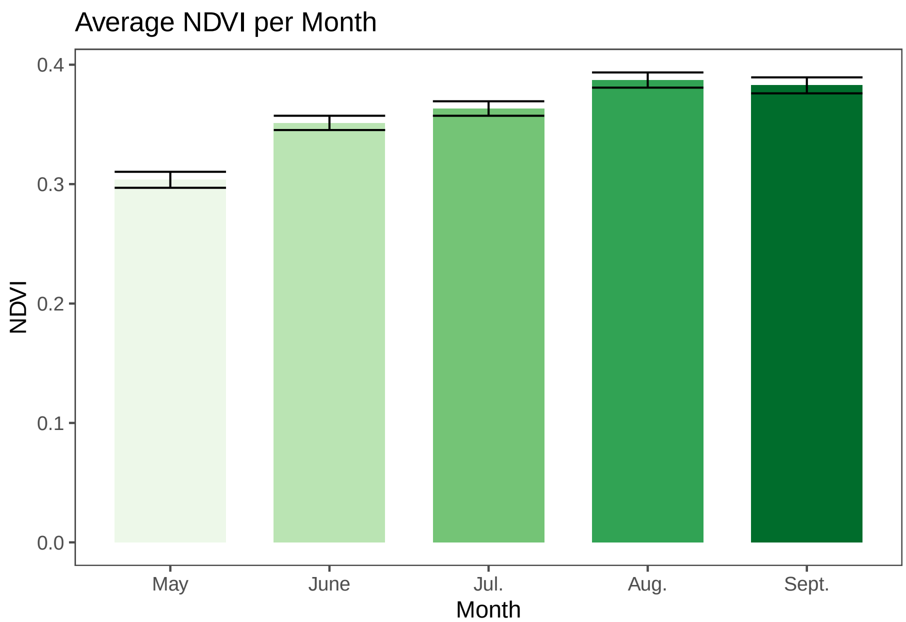


Figure 3.3: Mean NDVI and standard error per summer month across sites from 1984 to 2019.

3.4 What month is the snowiest on average?

If we plot the NDSI means for the winter months, we can see that the highest snowfall is January.

```
# Change ordering manually and make month into factor
monthly_win$month <- factor(monthly_win$month,
                  levels = c("11","12", "1", "2", "3"))

monthly_win %>%
  filter(data == "ndsi") %>%
ggplot(., aes(x=month,y=value_mean, fill=month)) +
  geom_bar(stat = "identity", width = 0.7, position = "dodge") +
  geom_errorbar(aes(ymin=value_mean-value_std.error, ymax=value_mean+value_std.error),
                colour = "black", width = 0.7, position = "dodge") +
  scale_x_discrete(labels=c("11"="Nov.", "12"="Dec", "1"="Jan.", "2"="Feb.",
                            "3"="Mar.")) +
  xlab("Month") +  ylab("NDSI") +
  ggtitle("Average NDSI per Month") +
  theme_few() +
  scale_fill_brewer(palette = "Purples") +
```

```
  theme(panel.grid.major=element_blank(), panel.grid.minor=element_blank(),
        legend.position="none")
```

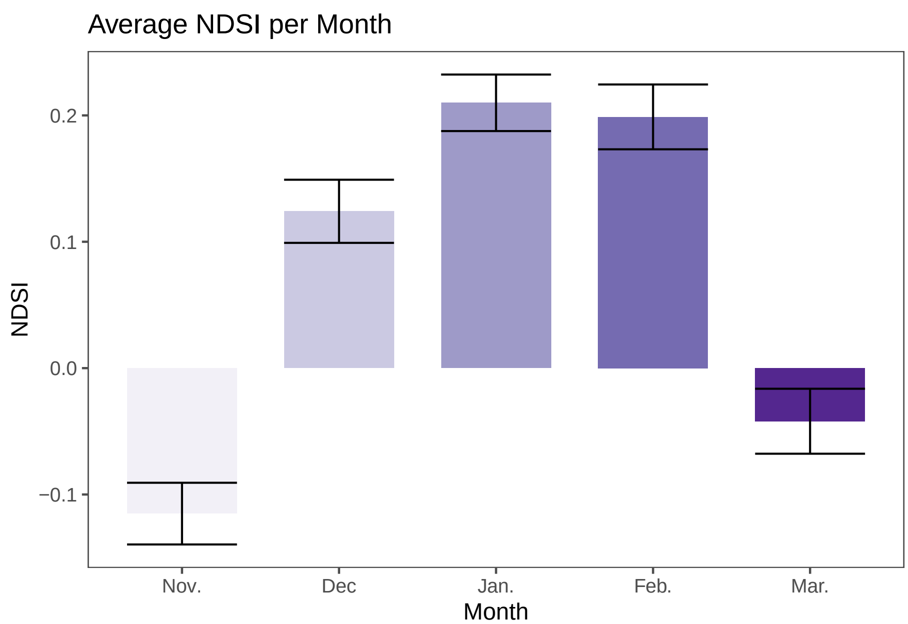


Figure 3.4: Mean NDSI and standard error per winter month across sites from 1984 to 2019.

Chapter 4

Fire Effects on Fish Populations

Wildfires don't only impact vegetation, but a wide variety of abiotic and biotic elements of the ecosystem. In this assignment, I looked at how fish in the Cache La Poudre Watershed were impacted by the High Park Fire in 2012.

Data and assignment provided by Dr. Michael Lefsky of Colorado State University.

4.1 Pre versus post fire fish length and mass

```
#summarize fishdata_R by time (another way to do this without subsets)
summary(fishdata_4R[fishdata_4R$time=="pre-fire",])
```

```
##      time             capture_id       length_cm         mass_g
##  Length:100         Min.   :  1.00   Min.   : 5.00   Min.   : 66
##  Class :character   1st Qu.: 25.75   1st Qu.:15.00   1st Qu.:132
##  Mode  :character   Median : 50.50   Median :18.00   Median :151
##                     Mean   : 50.50   Mean   :19.16   Mean   :154
##                     3rd Qu.: 75.25   3rd Qu.:23.00   3rd Qu.:182
##                     Max.   :100.00   Max.   :32.00   Max.   :252
```

```
summary(fishdata_4R[fishdata_4R$time=="post-fire",])
```

```
##      time             capture_id   length_cm         mass_g
##  Length:97          Min.   : 1   Min.   : 5.00   Min.   : 45.0
```

```
##  Class :character   1st Qu.:25   1st Qu.:15.00   1st Qu.: 89.0
##  Mode  :character   Median :49   Median :20.00   Median :113.0
##                     Mean   :49   Mean   :19.76   Mean   :107.9
##                     3rd Qu.:73   3rd Qu.:25.00   3rd Qu.:126.0
##                     Max.   :97   Max.   :38.00   Max.   :157.0
```

```
# create function to run statistics
lab_stats <- function(x) c(sd(x),sd(x)^2,sd(x)/sqrt(length(x))) #calculate standard de

#Pre-fire statistics
lab_stats(fishdata_4R[fishdata_4R$time=="pre-fire",]$length_cm) #fish length
```

```
## [1]  6.2145479 38.6206061  0.6214548
```

```
lab_stats(fishdata_4R[fishdata_4R$time=="pre-fire",]$mass_g) #fish mass
```

```
## [1]   36.277409 1316.050404    3.627741
```

```
#Post-fire statistics
lab_stats(fishdata_4R[fishdata_4R$time=="post-fire",]$length_cm) #fish length
```

```
## [1]  7.0574624 49.8077749  0.7165767
```

```
lab_stats(fishdata_4R[fishdata_4R$time=="post-fire",]$mass_g) #fish mass
```

```
## [1]  26.894853 723.333119   2.730759
```

```
# 1-way ANOVA on pre- vs. post-fire mass and length
summary(aov(fishdata_4R$length_cm~fishdata_4R$time)) #ANOVA for fish length pre vs. po
```

```
##                   Df Sum Sq Mean Sq F value Pr(>F)
## fishdata_4R$time   1     18   17.90   0.406  0.525
## Residuals        195   8605   44.13
```

```
summary(aov(fishdata_4R$mass_g~fishdata_4R$time)) #ANOVA for fish mass pre vs. post fi
```

```
##                   Df Sum Sq Mean Sq F value Pr(>F)
## fishdata_4R$time   1 104798  104798   102.3 <2e-16 ***
## Residuals        195 199729    1024
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# Make a 2 x 2 matrix of histograms for pre- and post-fire mass and length
par(mfrow=c(2,2)) #tell R how I want figures arranged

#Pre-fire histograms
hist(fishdata_4R[fishdata_4R$time == "pre-fire",]$length_cm,main="Pre-fire length (cm)",xlab="Len
hist(fishdata_4R[fishdata_4R$time == "pre-fire",]$mass_g,main="Pre-fire mass (g)",xlab="Mass (g)"

#Post-fire histograms
hist(fishdata_4R[fishdata_4R$time == "post-fire",]$length_cm,main="Post-fire length (cm)",xlab="L
hist(fishdata_4R[fishdata_4R$time == "post-fire",]$mass_g,main="Post-fire mass (g)",xlab="Mass (g
```

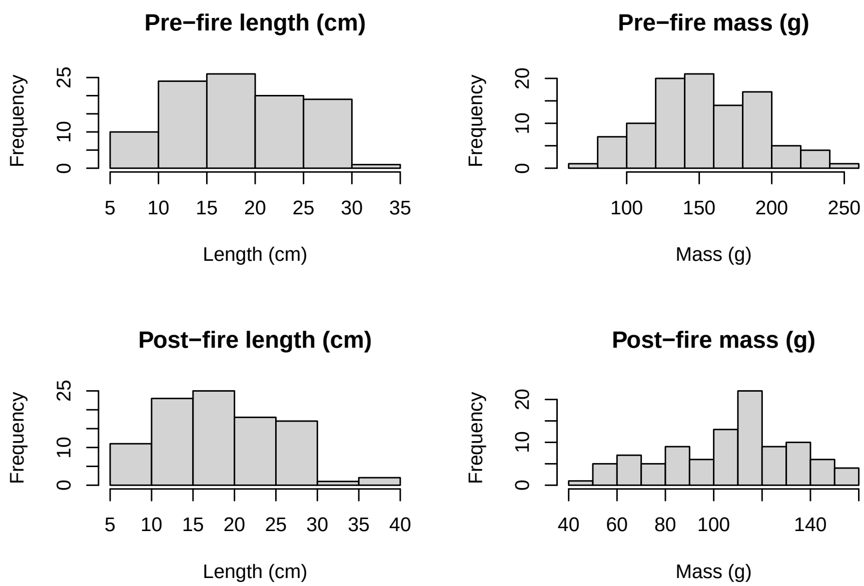


Figure 4.1: Histograms showing frequency of various lengths in centimeters and masses in grams of fish in in Cache La Poudre Watershed in 2012 before the High Park Fire (Pre-fire) and in 2013 after the High Park Fire (Post-fire).

```
# Make a 2 x 2 matrix of histograms for pre- and post-fire mass and length
par(mfrow=c(2,2)) #tell R how I want figures arranged

# Make two boxplots side by side
par(mfrow=c(1,2)) #tell R I want two plots
boxplot(fishdata_4R$length_cm~fishdata_4R$time, main="Length (cm)",ylab = "Frequency",xlab="Time"
boxplot(fishdata_4R$mass_g~fishdata_4R$time, main="Mass (g)",ylab = "Frequency",xlab="Time") #len
```

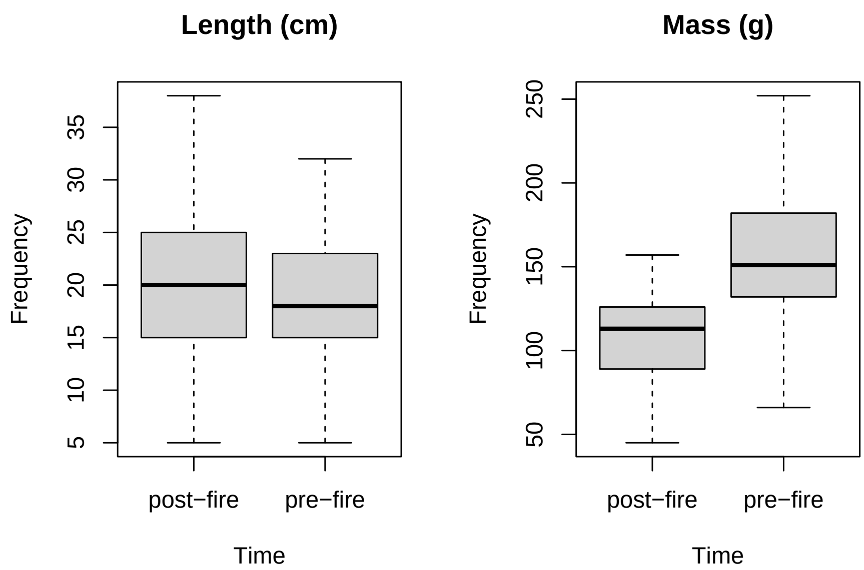


Figure 4.2: Boxplots for fish length in centimeters and mass in grams pre and post fire.

```
# Reset setting for plots
par(mfrow=c(1,1)) #return to single plot
```

4.2 Linear regression of fish mass vs. length for before and after the fire

```
# Pre-fire
# Scatterplot of length and mass where length is the independent variable and mass is
plot(mass_g ~ length_cm, data=fishdata_4R[fishdata_4R$time=="pre-fire",], xlab="Length
title("Pre-fire Fish Mass vs. Length")

# Linear regression on mass vs.length
lm_pre <- lm(mass_g ~ length_cm,data=fishdata_4R[fishdata_4R$time=="pre-fire",])
abline(lm_pre)  #Adds the trendline to the regression scatterplot
```

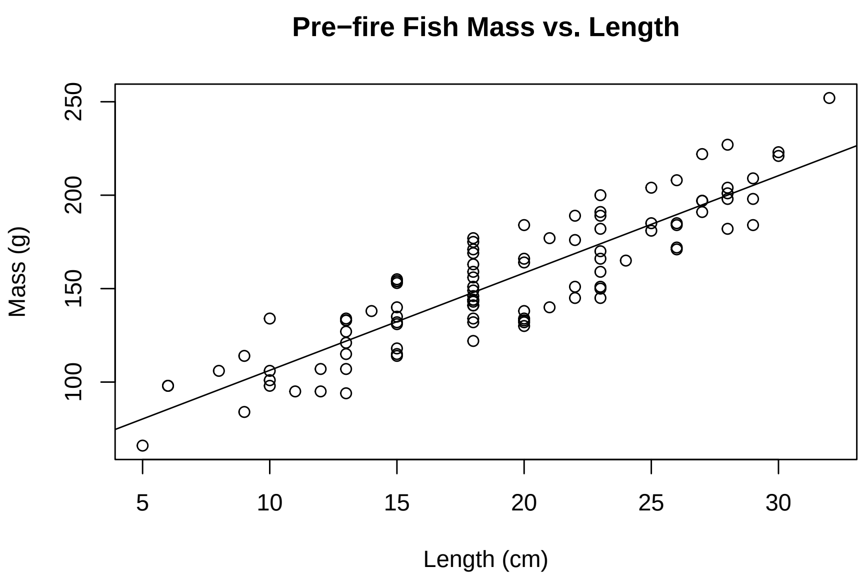


Figure 4.3: Scatterplot and linear regression line of fish length in centimeters versus fish mass in grams in Cache La Poudre in 2012 before the High Park Fire.

```
summary(aov(lm_pre)) #shows the results of the pre-fire linear regression ANOVA
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## length_cm    1 103690  103690     382 <2e-16 ***
## Residuals   98  26599     271
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(lm_pre) #shows equation of the line, multiple R-squared value
```

```
##
## Call:
## lm(formula = mass_g ~ length_cm, data = fishdata_4R[fishdata_4R$time ==
##     "pre-fire", ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -28.987 -14.472  -0.307  12.543  31.144
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  54.2113     5.3641   10.11   <2e-16 ***
## length_cm     5.2077     0.2664   19.55   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.47 on 98 degrees of freedom
## Multiple R-squared:  0.7958, Adjusted R-squared:  0.7938
## F-statistic:   382 on 1 and 98 DF,  p-value: < 2.2e-16
```

```
# Post-fire
# Scatterplot of length and mass where length is the independent variable and mass is
plot(mass_g ~ length_cm, data=fishdata_4R[fishdata_4R$time=="post-fire",], xlab="Length
title("Post-fire Fish Mass vs. Length")

# Linear regression on mass vs.length
lm_post <- lm(mass_g ~ length_cm,data=fishdata_4R[fishdata_4R$time=="post-fire",])
abline(lm_post)  #Adds the trendline to the regression scatterplot
```

```
summary(aov(lm_post)) #shows the results of the pre-fire linear regression ANOVA
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
```

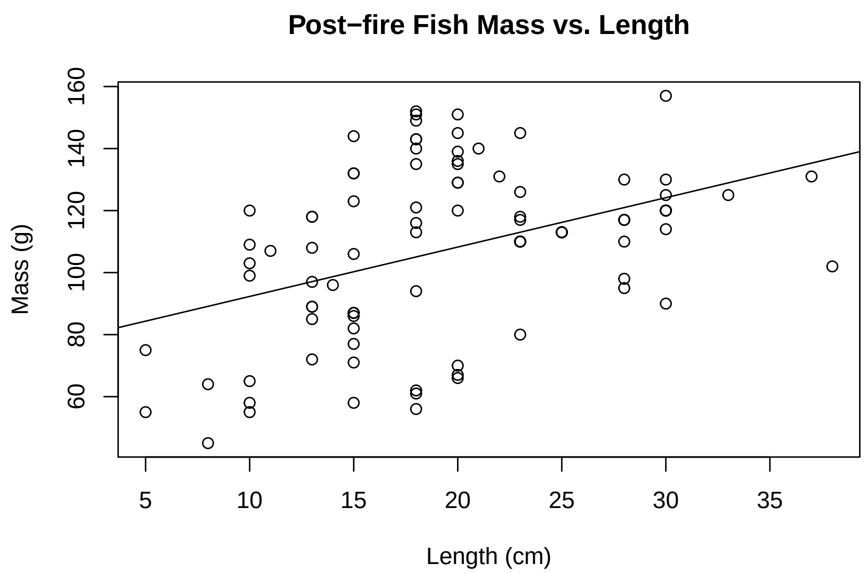


Figure 4.4: Scatterplot and linear regression line of fish length in centimeters versus fish mass in grams in Cache La Poudre in 2013 after the High Park Fire.

```
## length_cm    1  12126   12126    20.1 2.05e-05 ***
## Residuals   95  57313     603
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(lm_post) #shows equation of the line, multiple R-squared value
```

```
##
## Call:
## lm(formula = mass_g ~ length_cm, data = fishdata_4R[fishdata_4R$time ==
##     "post-fire", ])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.048 -13.271  -3.011  19.582  46.952
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  76.3830     7.4498  10.253  < 2e-16 ***
## length_cm     1.5925     0.3552   4.483 2.05e-05 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 24.56 on 95 degrees of freedom
## Multiple R-squared:  0.1746, Adjusted R-squared:  0.1659
## F-statistic:  20.1 on 1 and 95 DF,  p-value: 2.054e-05
```

```
#Pre- and Post-Fire on same graph

# First plot the pre-fire linear regression
# ylim sets the range of the y-axis; pch="+" makes points appear as plus signs; col="b
plot(mass_g ~length_cm,data=fishdata_4R[fishdata_4R$time == "pre-fire",],xlab="Length
title("Pre-Fire (+) and Post-Fire (o) Mass vs. Length")

# Run linear regression of pre-fire mass and length to obtain the trend line.
lm_pre=lm(mass_g ~ length_cm,data=fishdata_4R[fishdata_4R$time == "pre-fire",])
abline(lm_pre,col="blue")   #adds a trendline to the plot and makes the line blue

# Overlay the post-fire linear regression onto the plot of the pre-fire linear regress
# Plots post-fire data as o's and colors them red
points(mass_g ~length_cm,data=fishdata_4R[fishdata_4R$time == "post-fire",],xlab="Leng

# Run linear regression of post-fire mass and length to obtain the trend line.
lm_post=lm(mass_g ~ length_cm,data=fishdata_4R[fishdata_4R$time == "post-fire",])
abline(lm_post,col="red")   #adds a trendline to the post-fire linear regression and m
```

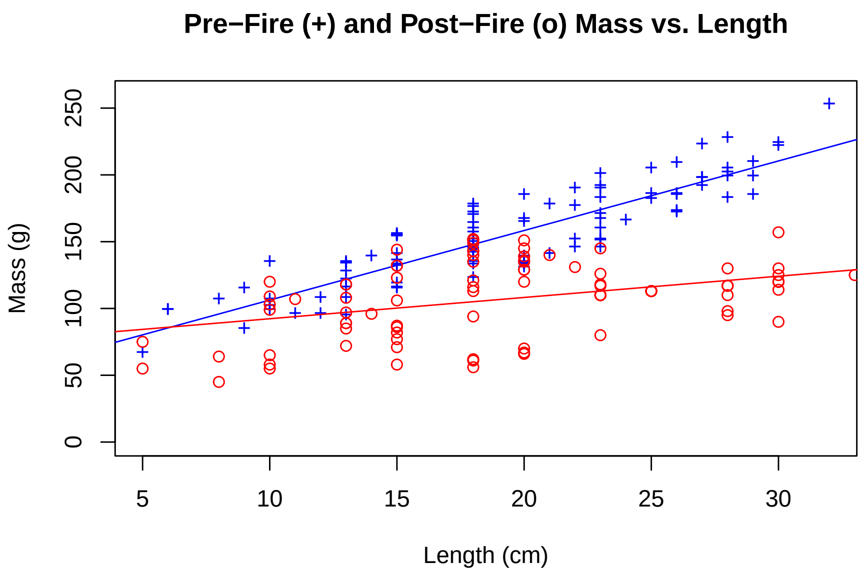


Figure 4.5: Scatterplot and linear regression line of fish length in centimeters versus fish mass in grams in Cache La Poudre in 2012 before the High Park Fire (blue, +) and in 2013 after the High Park Fire (red, o).

Chapter 5

Extracting and Visualizing Meteorological Data

> "What do you call dangerous precipitation? A rain of terror."

For this assignment, we used custom functions to read in and look at average meteorological data scraped from a public data archive.

Data is from Snowstudies.org. Assignment by Dr. Matthew Ross and Dr. Nathan Mueller of Colorado State University.

5.1 Extract the meteorological data URLs.

```r
# Read HTML page
snowarchive <- read_html("https://snowstudies.org/archived-data/")

# Read link with specific pattern
links <- snowarchive %>%
  html_nodes('a') %>% #look for links
  .[grepl('forcing',.)] %>% #filter to only links with "forcing" term
  html_attr('href') #tell it these are urls

links # view
```

```
## [1] "https://snowstudies.org/wp-content/uploads/2022/02/SBB_SASP_Forcing_Data.txt"
## [2] "https://snowstudies.org/wp-content/uploads/2022/02/SBB_SBSP_Forcing_Data.txt"
```

5.2 Download the meteorological data from the URL

```
# Grab only the name of the file by splitting out on forward slashes
splits <- str_split_fixed(links,'/',8)

#Keep only the 8th column
files <- splits[,8]

files
```

```
## [1] "SBB_SASP_Forcing_Data.txt" "SBB_SBSP_Forcing_Data.txt"
```

```
# Generate a file list for where the data goes
file_names <- paste0('Data_sci_bookdown/data/snow/', files)

# For loop that downloads each - i for every instance, length function tells how many
for(i in 1:length(file_names)){
  download.file(links[i],destfile=file_names[i])
}

# Download via map function
#map2(links, file_names, download.file)

# Map version of the for loop (downloading files)
downloaded <- file.exists(file_names)
evaluate <- !all(downloaded) # sees if files are downloaded (T/F)
if(evaluate == T){
  map2(links[1:2],file_names[1:2],download.file)
}else{print('data downloaded')}
```

```
## [1] "data downloaded"
```

5.3 Write a custom function to read in the data and append a site column to the data

```
# Traditional read in

SASP <- read.csv("Data_sci_bookdown/data/snow/SBB_SASP_Forcing_Data.csv") %>%
```

```
  select(1,2,3,7,10)

colnames(SASP) <- c("year","month","day","precip","temp")

SBSP <- read.csv("Data_sci_bookdown/data/snow/SBB_SBSP_Forcing_Data.csv") %>%
  select(1,2,3,7,10)

colnames(SBSP) <- c("year","month","day","precip","temp")

# Combine csvs
alldata <- rbind(SASP,SBSP)

# Read in via new function

# Grab headers from metadata pdf
library(pdftools)
```

```
## Using poppler version 20.12.1
```

```
headers <- pdf_text('https://snowstudies.org/wp-content/uploads/2022/02/Serially-Complete-Metadat
  readr::read_lines(.) %>%
  trimws(.) %>%
  str_split_fixed(.,'\\.',2) %>%
  .[,2] %>%
  .[1:26] %>%
  str_trim(side = "left")
```

5.4 Use the map function to read in both meteorological files

```
# Pull site name out of the file name and read in the .txt files
read_data <- function(file){
  name = str_split_fixed(file,'_',2)[,2] %>%
    gsub('_Forcing_Data.txt','',.)
  df <- read_fwf(file) %>%
    select(year=1, month=2, day=3, hour=4, precip=7, air_temp=10) %>% #choose and name columns
    mutate(site = name) #add column
}

alldata2 <- map_dfr(file_names,read_data)
```

```
## Rows: 69168 Columns: 19
## -- Column specification --------------------------------------------------------
##
## chr  (2): X12, X14
## dbl (17): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X13, X15, X16, X17, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
## Rows: 69168 Columns: 19
## -- Column specification --------------------------------------------------------
##
## chr  (2): X12, X14
## dbl (17): X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X13, X15, X16, X17, ...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
summary(alldata2)
```

```
##       year          month             day             hour
##  Min.   :2003   Min.   : 1.000   Min.   : 1.00   Min.   : 0.00
##  1st Qu.:2005   1st Qu.: 3.000   1st Qu.: 8.00   1st Qu.: 5.75
##  Median :2007   Median : 6.000   Median :16.00   Median :11.50
##  Mean   :2007   Mean   : 6.472   Mean   :15.76   Mean   :11.50
##  3rd Qu.:2009   3rd Qu.: 9.000   3rd Qu.:23.00   3rd Qu.:17.25
##  Max.   :2011   Max.   :12.000   Max.   :31.00   Max.   :23.00
##      precip              air_temp         site
##  Min.   :0.000e+00   Min.   :242.1   Length:138336
##  1st Qu.:0.000e+00   1st Qu.:265.8   Class :character
##  Median :0.000e+00   Median :272.6   Mode  :character
##  Mean   :3.838e-05   Mean   :272.6
##  3rd Qu.:0.000e+00   3rd Qu.:279.7
##  Max.   :6.111e-03   Max.   :295.8
```

5.5 Make a line plot of mean temp by year by site

```r
temp_yearly <- alldata2 %>%
group_by(year, site) %>%
summarise(mean_temp = mean(`air_temp`, na.rm=T))
```

```
## `summarise()` has grouped output by 'year'. You can override using the `.groups`
```

```
## argument.
```

```
ggplot(temp_yearly,aes(x=year, y=mean_temp, color=site)) +
  geom_point() + geom_line() +
  xlab("Year") + ylab("Mean Temperature (Degrees Kelvin)") +
  ggthemes::theme_few() +
  scale_color_brewer(palette = "Set2") +
  scale_x_continuous(breaks = pretty(c(2003,2012), n = 6)) +
  theme(legend.position="bottom")
```

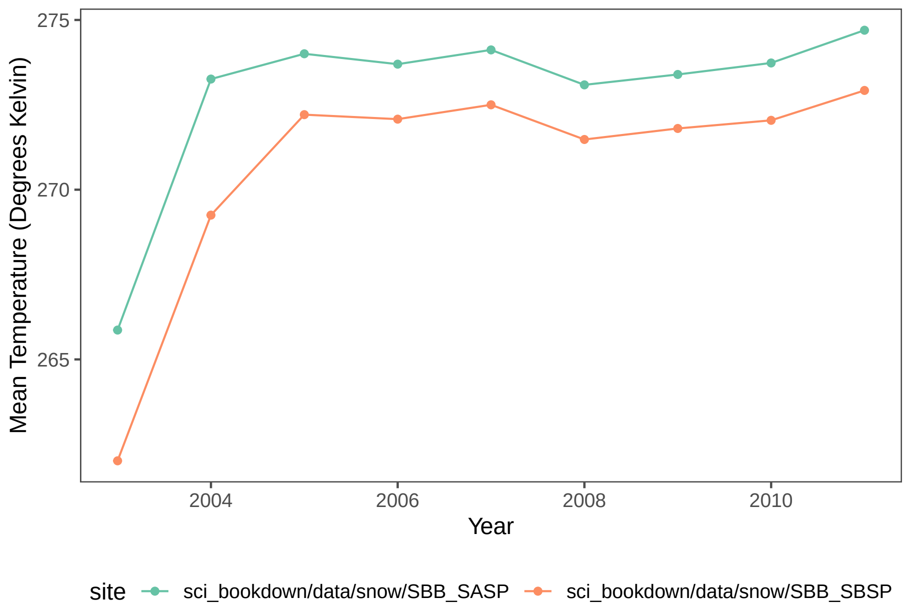


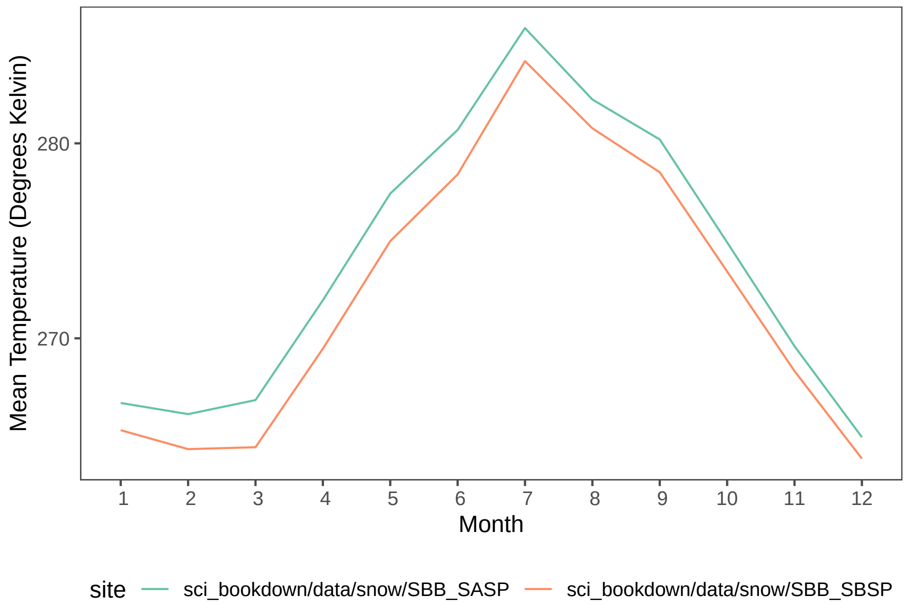
Figure 5.1: Mean temperature of the SASP (teal) and SBSP (orange) sites from 2003 to 2012, in degrees Kelvin.

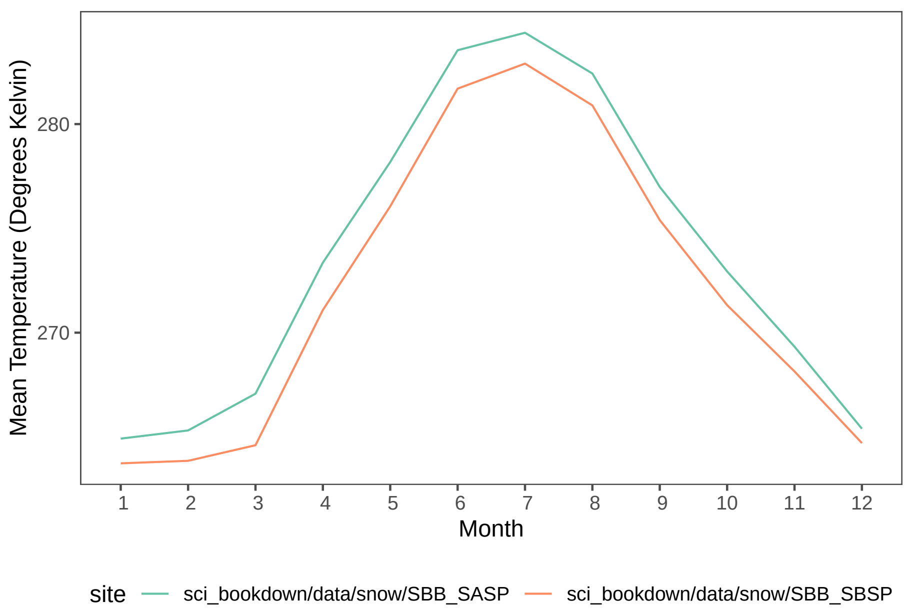
5.6 Write a function that makes line plots of monthly average temperature at each site for a given year. Use a for loop to make these plots for 2005 to 2010.

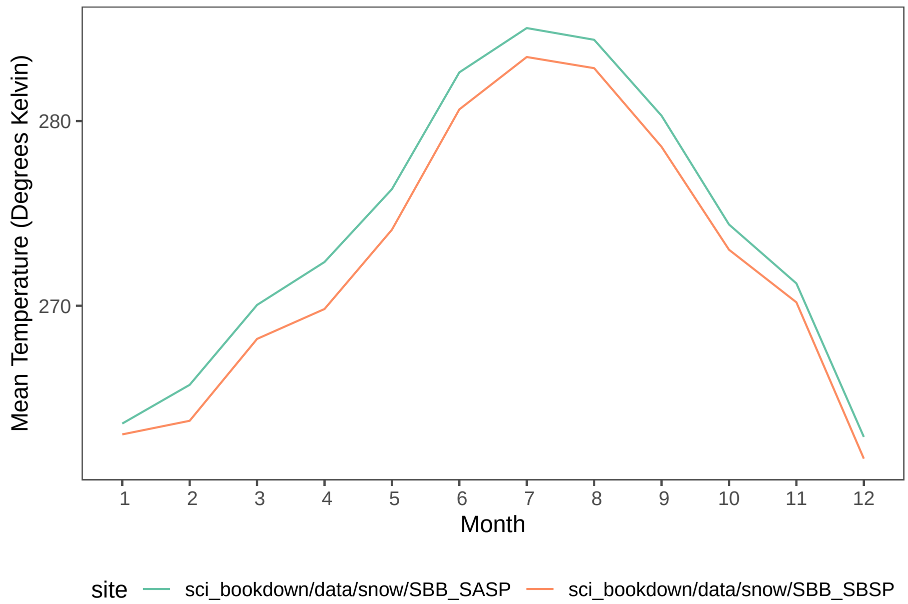
```
temp_monthly <- alldata2 %>%
    group_by(year, month, site) %>%
    summarize(mean_temp = mean(`air_temp`, na.rm=T))
```

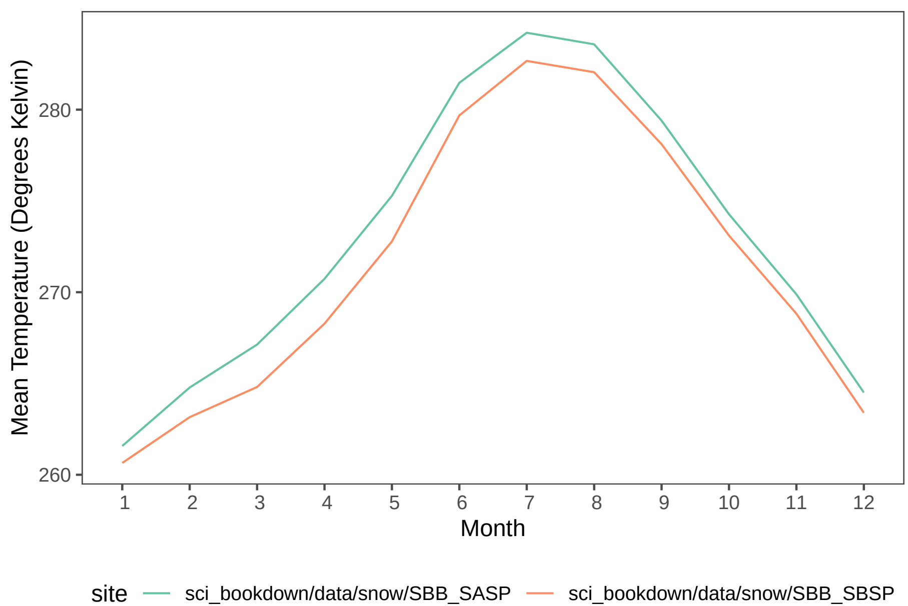
```
## `summarise()` has grouped output by 'year', 'month'. You can override using the
## `.groups` argument.
```

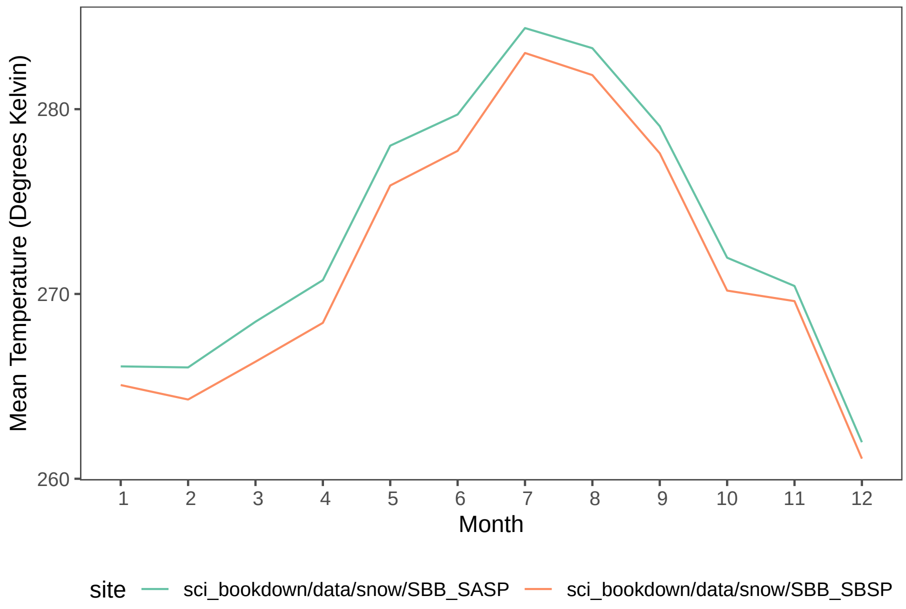
```
par(mfrow=c(5,1))

plot_monthly <- function(year.no) {
  plot <- temp_monthly %>%
    filter(year == year.no) %>%
    ggplot(aes(x=month, y=mean_temp, color=site)) +
      geom_line() +
      xlab("Month") + ylab("Mean Temperature (Degrees Kelvin)") +
      ggthemes::theme_few() +
      scale_color_brewer(palette = "Set2") +
      scale_x_discrete(limits = c(1,2,3,4,5,6,7,8,9,10,11,12)) +
      scale_y_continuous(breaks = pretty(c(255,290), n = 4)) +
      theme(legend.position="bottom")
  print(plot)
  }

for(i in 2005:2010){
  plot_monthly(i)
}
```

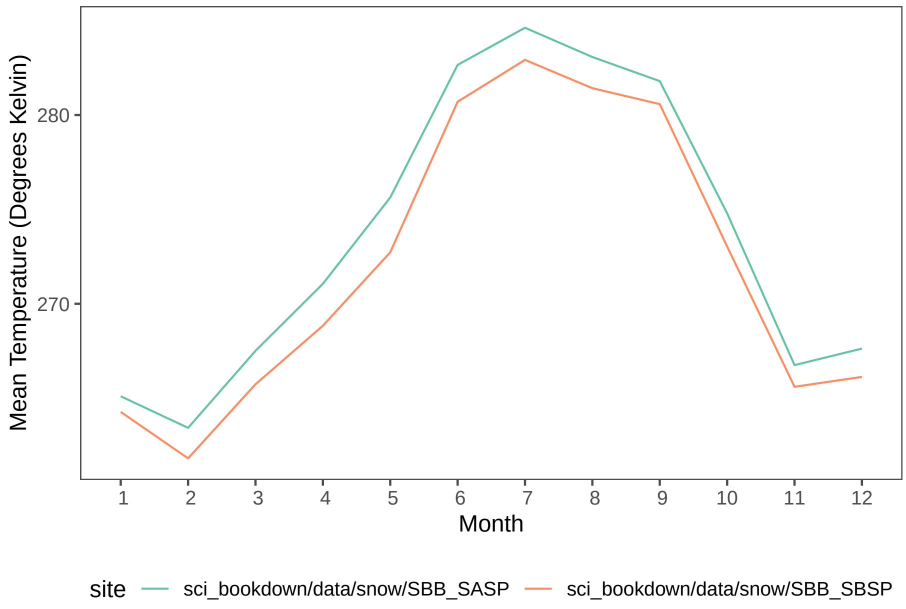

Mean Temperature (Degrees Kelvin)
280
270
1 2 3 4 5 6 7 8 9 10 11 12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP


Mean Temperature (Degrees Kelvin)
280
270
1 2 3 4 5 6 7 8 9 10 11 12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP


Mean Temperature (Degrees Kelvin)
280
270
1 2 3 4 5 6 7 8 9 10 11 12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP


Mean Temperature (Degrees Kelvin)
280
270
260
1 2 3 4 5 6 7 8 9 10 11 12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP


Mean Temperature (Degrees Kelvin)
260
270
280
1
2
3
4
5
6
7
8
9
10
11
12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP


Mean Temperature (Degrees Kelvin)
270
280
1
2
3
4
5
6
7
8
9
10
11
12
Month
site
sci_bookdown/data/snow/SBB_SASP
sci_bookdown/data/snow/SBB_SBSP

5.7 Make a plot of average daily precipitation by day of year (averaged across all available years)

```
precip_daily <- alldata2 %>%
  mutate(date = make_date(year, month, day),
                day_no = yday(date)) %>%
  group_by(day_no) %>%
  summarize(mean_precip = mean(`precip`*86400, na.rm=T))

ggplot(precip_daily, aes(x=day_no, y=mean_precip)) +
      geom_line() +
      xlab("Day of Year") + ylab("Mean Precipitation (mm/day)") +
      ggthemes::theme_few() +
      scale_color_brewer(palette = "Set2") +
      scale_y_continuous(breaks = pretty(c(0,14), n = 7)) +
      scale_x_continuous(breaks = pretty(c(1,365), n = 8))
```

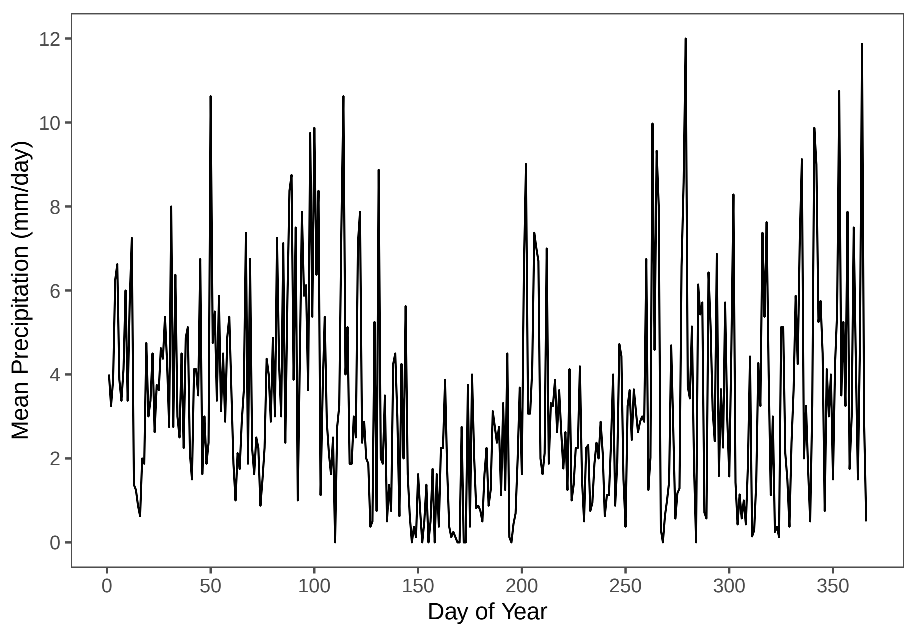


Figure 5.2: Mean daily precipitation by day of year, averaged from 2003 to 2012.

Chapter 6

Spatial Analysis in R

> "Why are latitude and longitude so smart? Because they have so many degrees!"

In this assignment, I learned to use R for spatial analyses.

Data is from the LAGOS dataset. Assignment by Dr. Matthew Ross and Dr. Nathan Mueller of Colorado State University.

6.1 Loading in data

6.1.1 First download and then specifically grab the locus (or site lat longs)

```
# #Lagos download script
#LAGOSNE::lagosne_get(dest_folder = LAGOSNE:::lagos_path(), overwrite = TRUE)


#Load in lagos
lagos <- lagosne_load()
```

```
## Warning in (function (version = NULL, fpath = NA) : LAGOSNE version unspecified,
## loading version: 1.087.3
```

```
#Grab the lake centroid info
lake_centers <- lagos$locus
```

```
# Make an sf object
spatial_lakes <- st_as_sf(lake_centers,coords=c('nhd_long','nhd_lat'),
                          crs=4326)

#Grab the water quality data
nutr <- lagos$epi_nutr

#Look at column names
#names(nutr)
```

6.1.2 Convert to spatial data

```
#Look at the column names
#names(lake_centers)

#Look at the structure
#str(lake_centers)

#View the full dataset
#View(lake_centers %>% slice(1:100))

spatial_lakes <- st_as_sf(x = lake_centers, coords = c("nhd_long","nhd_lat"), crs = 43
  st_transform(2163)

#mapview(spatial_lakes)

#Subset for plotting
subset_spatial <- spatial_lakes %>%
  slice(1:100)

subset_baser <- spatial_lakes[1:100,]

#Dynamic mapviewer
#mapview(subset_spatial)
```

6.1.3 Subset to only Minnesota

```
states <- us_states()

#Plot all the states to check if they loaded
#mapview(states)
```

```r
minnesota <- states %>%
  filter(name == 'Minnesota') %>%
  st_transform(2163)
#mapview(minnesota)

#Subset lakes based on spatial position
minnesota_lakes <- spatial_lakes[minnesota,]

#Plotting the first 1000 lakes
minnesota_lakes %>%
  arrange(-lake_area_ha) %>%
    slice(1:1000)
```

```
## Simple feature collection with 1000 features and 16 fields
## Geometry type: POINT
## Dimension:     XY
## Bounding box:  xmin: 254441 ymin: -154522.4 xmax: 755222.3 ymax: 464949.4
## Projected CRS: NAD27 / US National Atlas Equal Area
## First 10 features:
##    lagoslakeid     nhdid           gnis_name lake_area_ha lake_perim_meters
## 1        15162 123319728   Lake of the Woods   123779.817         401005.02
## 2        34986 105567868      Lower Red Lake    66650.332         115825.47
## 3         2498 120019294     Mille Lacs Lake    51867.225         151701.94
## 4        39213 105567402      Upper Red Lake    48288.325          99828.05
## 5          996 120018981          Leech Lake    41824.352         344259.98
## 6          583 120019513 Lake Winnibigoshish    22566.124          86722.10
## 7           73 120019354          Rainy Lake    18522.551         660313.32
## 8         2554 105954753      Vermilion Lake    15736.590         509617.01
## 9         2161 120019371     Kabetogama Lake     9037.249         288750.31
## 10        3119 166868528           Cass Lake     8375.173          85326.14
##    nhd_fcode nhd_ftype iws_zoneid hu4_zoneid hu6_zoneid hu8_zoneid hu12_zoneid
## 1      39004       390  IWS_37547     HU4_26     HU6_36    HU8_468  HU12_13912
## 2      39004       390  IWS_34899     HU4_54     HU6_74    HU8_327  HU12_14600
## 3      39004       390  IWS_22933     HU4_25     HU6_73    HU8_344  HU12_10875
## 4      39004       390  IWS_33471     HU4_54     HU6_74    HU8_327  HU12_14204
## 5      39004       390  IWS_23572     HU4_25     HU6_35    HU8_332  HU12_14479
## 6      39004       390  IWS_22455     HU4_25     HU6_35    HU8_331  HU12_14543
## 7      39004       390  IWS_37542     HU4_26     HU6_36    HU8_473  HU12_13942
## 8      39004       390  IWS_36424     HU4_26     HU6_36    HU8_131  HU12_14405
## 9      39004       390  IWS_36301     HU4_26     HU6_36    HU8_130  HU12_14395
## 10     39004       390  IWS_21080     HU4_25     HU6_35    HU8_331  HU12_13957
##    edu_zoneid county_zoneid state_zoneid elevation_m                 geometry
## 1      EDU_56    County_435     State_14    323.5090 POINT (366706.2 464949.4)
## 2      EDU_16    County_455     State_14    358.1656 POINT (371974.2 341706.5)
## 3      EDU_43    County_484     State_14    381.7920 POINT (489582.1 157109.5)
```

```
## 4      EDU_16    County_455     State_14    358.3096 POINT (389013.3 360819.5)
## 5      EDU_42    County_424     State_14    395.2420 POINT (422409.7 255724.9)
## 6      EDU_42    County_424     State_14    396.1560   POINT (437872.1 286675)
## 7      EDU_55    County_446     State_14    338.0670 POINT (515833.6 420274.2)
## 8       EDU_3    County_446     State_14    414.1680 POINT (566966.7 347059.1)
## 9      EDU_55    County_446     State_14    339.2530 POINT (519199.2 408290.2)
## 10     EDU_42    County_424     State_14    396.7710 POINT (410563.2 281005.2)
```

```
  #mapview(.,zcol = 'lake_area_ha')
```

6.2 Part one

6.2.1 Show a map outline of Iowa and Illinois (similar to Minnesota map upstream)

```
Istates <- states %>%
  filter(name == 'Iowa'| name== 'Illinois') %>%
  st_transform(2163)
mapview(Istates, canvas = TRUE)
```

6.2.2 Subset LAGOS data to these sites, how many sites are in Illinois and Iowa combined? How does this compare to Minnesota?

```
Istates_lakes <- spatial_lakes[Istates,]

nrow(Istates_lakes)
```

```
## [1] 16466
```

```
Istates_count <- length(Istates_lakes$lagoslakeid)

nrow(minnesota_lakes)
```

```
## [1] 29038
```

```
Minn_count <- length(minnesota_lakes$lagoslakeid)
```

Iowa and Illinois have 16466 lakes combined, much less than the number of lakes that Minnesota alone has, 29038.

6.2.3 What is the distribution of lake size in Iowa vs. Minnesota?

- Here I want to see a histogram plot with lake size on x-axis and frequency on y axis (check out geom_histogram)

```
iowa <- states %>%
  filter(name == 'Iowa') %>%
  st_transform(2163)

iowa_lakes <- spatial_lakes[iowa,]

combined <- rbind(iowa_lakes, minnesota_lakes)

ggplot(combined, aes(x= lake_area_ha)) +
  ggthemes::theme_few() + theme(legend.position="bottom") +
  xlab("Lake Area (ha)") + ylab("Count") +
  scale_x_continuous(trans = "log10", labels = scales::comma) +
  geom_histogram(data = minnesota_lakes, color = "red", alpha = 0.2) +
  geom_histogram(data = iowa_lakes, color = "blue", alpha = 0.2) +
  scale_fill_manual(values=c("blue","red"), "State")
```

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

6.2.4 Make an interactive plot of lakes in Iowa and Illinois and color them by lake area in hectares

```
Istates_map = Istates_lakes %>%
  arrange(-lake_area_ha) %>%
    slice(1:1000)

mapview(Istates_map, zcol = 'lake_area_ha',  canvas = TRUE)
```

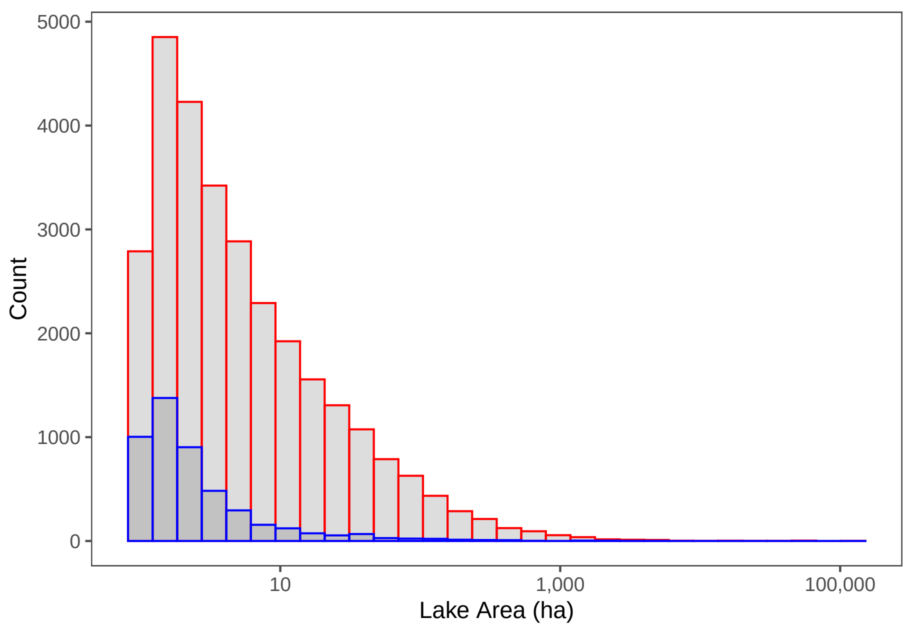


Figure 6.1: The number of lakes with a given area, in hectares, in Minnesota (red) and Iowa (blue).

6.2.5 What other data sources might we use to understand how reservoirs and natural lakes vary in size in these three states?

We might use the US Geological Survey (USGS) National Water Informational System (NWIS) and its National Water Dashboard as a data source, and look at gage height (indicating lake depth) as another parameter for lake size variation. The USGS National Hydrography Dataset (NHD) is another data source that would, similarly to Lagos, give us a surface area metric for lakes in the various states.

6.3 Part two

6.3.1 Subsets

6.3.1.1 Columns nutr to only keep key info that we want

```
clarity_only <- nutr %>%
  dplyr::select(lagoslakeid,sampledate,chla,doc,secchi) %>%
  mutate(sampledate = as.character(sampledate) %>% ymd(.))
```

6.3.1.2 Keep sites with at least 200 observations

```
#Look at the number of rows of dataset
#nrow(clarity_only)

chla_secchi <- clarity_only %>%
  filter(!is.na(chla),
         !is.na(secchi))

# How many observatiosn did we lose?
# nrow(clarity_only) - nrow(chla_secchi)


# Keep only the lakes with at least 200 observations of secchi and chla
chla_secchi_200 <- chla_secchi %>%
  group_by(lagoslakeid) %>%
  mutate(count = n()) %>%
  filter(count > 200)
```

6.3.1.3 Join water quality data to spatial data

```
spatial_200 <- inner_join(spatial_lakes,chla_secchi_200 %>%
                            distinct(lagoslakeid,.keep_all=T),
                          by='lagoslakeid')
```

6.3.2 Mean Chlorophyll A map

```
### Take the mean chl_a and secchi by lake

means_200 <- chla_secchi_200 %>%
  # Take summary by lake id
  group_by(lagoslakeid) %>%
  # take mean chl_a per lake id
  summarize(mean_chl = mean(chla,na.rm=T),
            mean_secchi=mean(secchi,na.rm=T)) %>%
  #Get rid of NAs
  filter(!is.na(mean_chl),
         !is.na(mean_secchi)) %>%
  # Take the log base 10 of the mean_chl
  mutate(log10_mean_chl = log10(mean_chl))

#Join datasets
mean_spatial <- inner_join(spatial_lakes,means_200,
                          by='lagoslakeid')

#Make a map
mapview(mean_spatial, zcol='log10_mean_chl', layer.name = "Mean Chlorophyll A Content")
```

6.3.3 What is the correlation between Secchi Disk Depth and Chlorophyll a for sites with at least 200 observations?

```
ggplot(means_200) +
  geom_point(aes(mean_secchi, mean_chl)) +
  ggthemes::theme_few() +
  xlab("Mean Secchi Disk Depth") + ylab("Mean Chlorophyll Content")
```

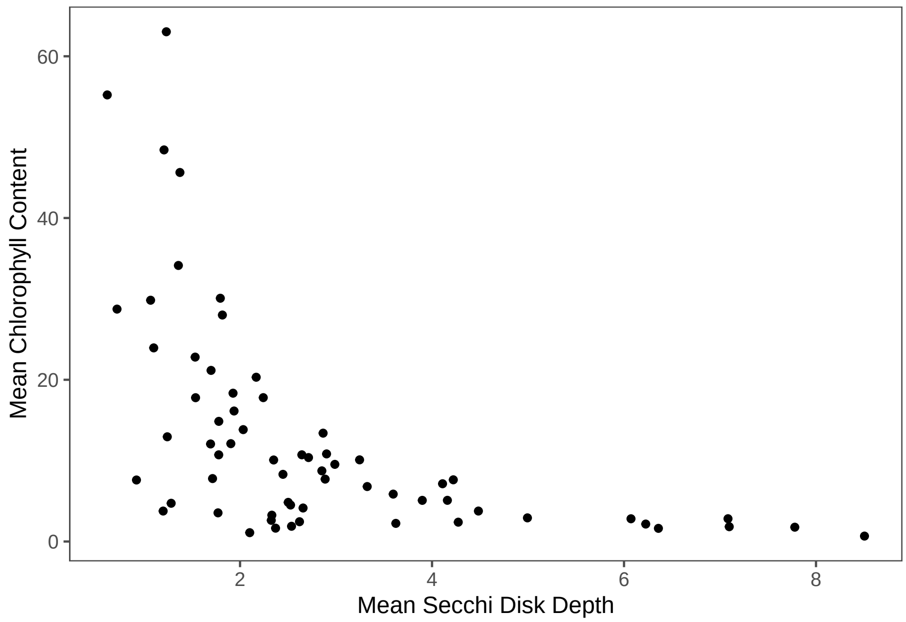


Figure 6.2: Chlorophyll content has a negative correlation with Secchi disk depth at sites with at least 200 observations.

6.3.3.1 Why might this be the case?

Secchi disks measure water clarity; the deeper the disk, the clearer the water (1). Chlorophyll content in lakes is generally a reliable marker of algae content, so that high chlorophyll values indicate high algal biomass and corresponding low water clarity (2). Additionally, chlorophyll may be used as a proxy for water quality, since high algal biomass is associated with high nutrient pollution in the process of eutrophication (2). High pollution may further decrease water clarity, so that the relationship between chlorophyll and Secchi disk depth may be expected.

1. "The Secchi Dip-in - What Is a Secchi Disk?" North American Lake Management Society (NALMS), https://www.nalms.org/secchidipin/monitoring-methods/the-secchi-disk/what-is-a-secchi-disk/.

2. Filazzola, A., Mahdiyan, O., Shuvo, A. et al. A database of chlorophyll and water chemistry in freshwater lakes. Sci Data 7, 310 (2020). https://doi-org.ezproxy2.library.colostate.edu/10.1038/s41597-020-00648-2

6.3.4 What states have the most data?

6.3.4.1 Make a lagos spatial dataset that has the total number of counts per site.

```
site_counts <- chla_secchi %>%
  group_by(lagoslakeid) %>%
  mutate(count = n())

lake_counts <- inner_join(site_counts, lake_centers, by= "lagoslakeid")%>%
  dplyr::select(lagoslakeid,nhd_long,nhd_lat, count, secchi, chla)

spatial_counts <- st_as_sf(lake_counts,coords=c("nhd_long","nhd_lat"),
                          crs=4326)
```

6.3.4.2 Join this point dataset to the us_boundaries data.

```
states <- us_states()

states_counts <- st_join(spatial_counts, states)
```

6.3.4.3 Group by state and sum all the observations in that state and arrange that data from most to least total observations per state.

```
sum_statecount <- states_counts %>%
  group_by(state_name) %>%
  summarize(sum = sum(count)) %>%
  arrange(desc(sum))

sumtable <- tibble(sum_statecount)

view(sumtable)

#ggplot(data = sumtable, aes(x=state_name, y=sum, fill=state_name)) +
#  geom_bar(stat = "identity", width = 0.3, position = "dodge") +
#  ggthemes::theme_few() +
#  xlab("State") + ylab(expression(paste("# of Observations")))
```

Minnesota has the most observations. Vermont, has the next most observations, but less than half of Minnesota's observations. South Dakota has the least number of observations in the dataset.

6.3.5 Is there a spatial pattern in Secchi disk depth for lakes with at least 200 observations?

```
mapview(mean_spatial, zcol='mean_secchi', layer.name = "Mean Secchi Disk Depth")
```

Chapter 7

Linear Regressions, Quadratic Fits, Residuals, and Spatial

> "How many data scientists does it take to change a light bulb? That depends. It is really a matter of power."

This assignment combined several methods to look at relationships between crop yields and weather data over time.

Data from USDA National Agricultural Statistical Service (NASS). Assignment by Dr. Matthew Ross and Dr. Nathan Mueller of Colorado State University.

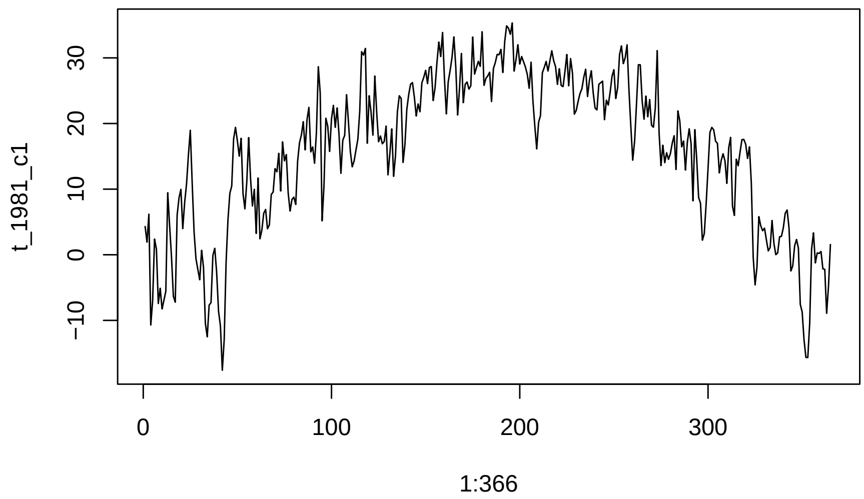
7.1 Weather Data Analysis

7.1.1 Load the PRISM daily maximum temperatures

```
# daily max temperature
# dimensions: counties x days x years
prism <- readMat("Data_sci_bookdown/data/prismiowa.mat")

# look at county #1
t_1981_c1 <- prism$tmaxdaily.iowa[1,,1] #first county, all days, first year
t_1981_c1[366] #check for leap year (366 days)
```

```
## [1] NaN
```

```
plot(1:366, t_1981_c1, type = "l") #base r plot
```




```
# assign dimension names to tmax matrix
dimnames(prism$tmaxdaily.iowa) <- list(prism$COUNTYFP, 1:366, prism$years) #add dimens

# converted 3d matrix into a data frame
tmaxdf <- as.data.frame.table(prism$tmaxdaily.iowa)

# relabel the columns
colnames(tmaxdf) <- c("countyfp","doy","year","tmax") #name columns
tmaxdf <- tibble(tmaxdf) #tidyverse table
```

7.1.2 Download NASS corn yield data

```
# set our API key with NASS
nassqs_auth(key = "B9113AF8-85C4-3CEE-8D93-6E885D49E24F") #Here put in API code from U

# parameters to query on
params <- list(commodity_desc = "CORN", util_practice_desc = "GRAIN", prodn_practice_d

# download
cornyieldsall <- nassqs_yields(params)
```

```
##   |                                                                              |
```

```
cornyieldsall$county_ansi <- as.numeric(cornyieldsall$county_ansi)
cornyieldsall$yield <- as.numeric(cornyieldsall$Value)

# clean and filter this dataset
cornyields <- select(cornyieldsall, county_ansi, county_name, yield, year) %>%
  filter(!is.na(county_ansi) & !is.na(yield))
cornyields <- tibble(cornyields)
```

7.2 Extract Winneshiek County corn yields, fit a linear time trend, make a plot. Is there a significant time trend?

```
winnecorn <- cornyields %>%
  filter(county_ansi == "191")

cornlm <- lm(yield ~ year, data = winnecorn)
summary(cornlm) #P=1.77e-13 R^2= 0.755
```

```
##
## Call:
## lm(formula = yield ~ year, data = winnecorn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.163  -1.841   2.363   9.437  24.376
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -4763.290    448.286  -10.63 4.46e-13 ***
## year            2.457      0.224   10.96 1.77e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.97 on 39 degrees of freedom
## Multiple R-squared:  0.7551, Adjusted R-squared:  0.7488
## F-statistic: 120.2 on 1 and 39 DF,  p-value: 1.767e-13
```

```
ggplot(winnecorn, mapping = aes(x = year, y = yield)) +
  geom_point() +
  theme_bw() +
  labs(x = "Year", y = "Corn Yield") +
  geom_smooth(method = lm, se=TRUE, color="#78917E", fill="#C5DDB3")
```

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

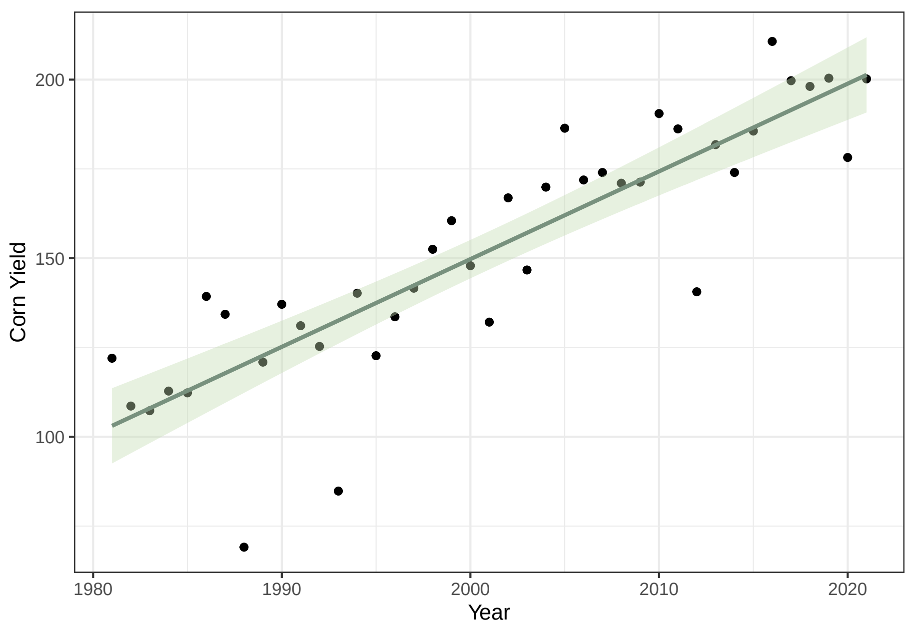


Figure 7.1: Linear regression of corn yields over time (years) in Winneshieck County, Iowa.

There is a significant positive correlation between corn yields and years in Winneshieck County, with an R-squared value of 0.755 and a P-value of 1.77e-13.

7.3 Fit a quadratic time trend (i.e., year + year^2) and make a plot. Is there evidence for slowing yield growth?

```r
winnecorn$yearsq <- winnecorn$year^2 #square explanatory variables for quadratic

lm_cornquad <- lm(yield ~ year + yearsq, winnecorn)
summary(lm_cornquad)
```

```
##
## Call:
## lm(formula = yield ~ year + yearsq, data = winnecorn)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -51.384  -3.115   1.388   9.743  25.324
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.583e+04  8.580e+04   0.301    0.765
## year        -2.812e+01  8.576e+01  -0.328    0.745
## yearsq       7.641e-03  2.143e-02   0.357    0.723
##
## Residual standard error: 17.17 on 38 degrees of freedom
## Multiple R-squared:  0.7559, Adjusted R-squared:  0.7431
## F-statistic: 58.84 on 2 and 38 DF,  p-value: 2.311e-12
```

```r
winnecorn$y_fitted <- lm_cornquad$fitted.values

#with the fitted values, create a non-linear trend
ggplot(winnecorn) +
  geom_point(mapping = aes(x = year, y = yield)) +
  geom_line(mapping = aes(x = year, y = y_fitted)) +
  theme_bw() +
  labs(x = "Year", y = "Corn Yield")
```

When we fit a quadratic line to the data, we find that it follows very closely to a linear regression, suggesting a fairly linear relationship between corn yields and years in Winneshieck County. There is no evidence of slowing yield growth in the model.

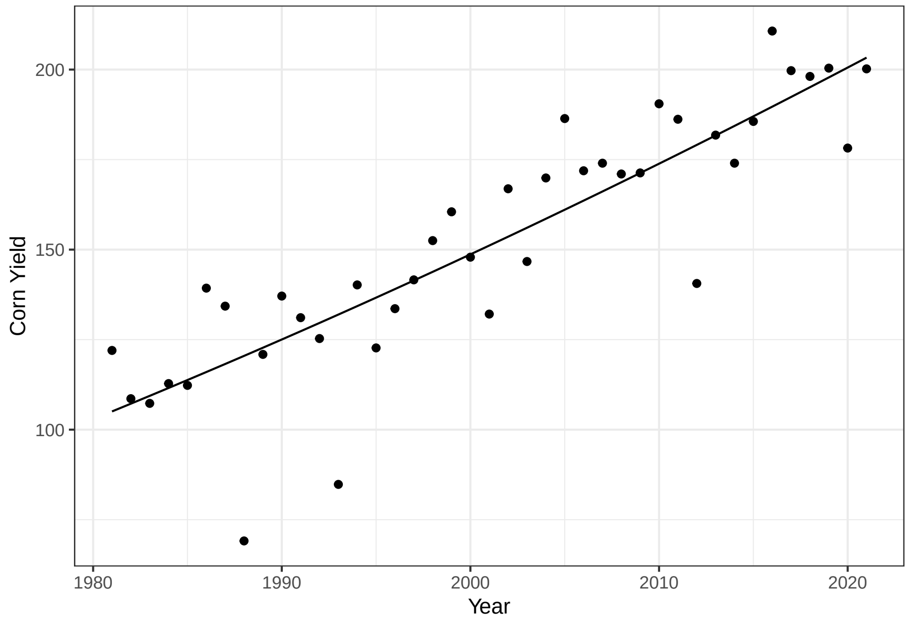


Figure 7.2: Quadratic fit of corn yields over time (years) in Winneshieck County, Iowa.

7.4 Time Series: Let's analyze the relationship between temperature and yields for the Winneshiek County time series. Use data on yield and summer avg Tmax. Is adding year or Tmax^2 to your model helpful? Make a plot and interpret the results.

```
# Winneshiek County summer temp maxes
tmaxdf$doy <- as.numeric(tmaxdf$doy)
tmaxdf$year <- as.numeric(as.character(tmaxdf$year))
tmaxdf$tmax <- as.numeric(tmaxdf$tmax)

winnesummer <- tmaxdf %>%
  filter(countyfp==191 & doy >= 152 & doy <= 243) %>% #day 152= June 1, 243= Aug 31
  group_by(year) %>%
  summarize(meantmax = mean(tmax))
```

```
lm_summertmax <- lm(meantmax ~ year, winnesummer)
summary(lm_summertmax) #not sig
```

```
##
## Call:
## lm(formula = meantmax ~ year, data = winnesummer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.5189 -0.7867 -0.0341  0.6859  3.7415
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 41.57670   36.44848   1.141    0.262
## year        -0.00747    0.01823  -0.410    0.684
##
## Residual standard error: 1.232 on 36 degrees of freedom
## Multiple R-squared:  0.004644,   Adjusted R-squared:  -0.02301
## F-statistic: 0.168 on 1 and 36 DF,  p-value: 0.6844
```

```
winnesummer$yearsq <- winnesummer$year^2 #square explanatory variables for quadratic
winnesummer$tmaxsq <- winnesummer$meantmax^2

lm_summerquad <- lm(meantmax ~ year + yearsq, winnesummer)
summary(lm_summerquad)
```

```
##
## Call:
## lm(formula = meantmax ~ year + yearsq, data = winnesummer)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.4617 -0.8812 -0.0530  0.7204  3.7308
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.618e+03  7.519e+03   0.481    0.633
## year        -3.585e+00  7.521e+00  -0.477    0.637
## yearsq       8.946e-04  1.881e-03   0.476    0.637
##
## Residual standard error: 1.246 on 35 degrees of freedom
## Multiple R-squared:  0.01104,    Adjusted R-squared:  -0.04547
## F-statistic: 0.1953 on 2 and 35 DF,  p-value: 0.8235
```

```
winnesummer$t_fitted <- lm_summerquad$fitted.values

# Join yield and temp data
winne <- inner_join(winnecorn, winnesummer)
```

```
## Joining, by = c("year", "yearsq")
```

```
lmwinne <- lm(yield ~ yearsq + tmaxsq, data = winne)
summary(lmwinne)
```

```
##
## Call:
## lm(formula = yield ~ yearsq + tmaxsq, data = winne)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.353  -7.496   2.089   9.806  27.874
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2.314e+03  2.557e+02  -9.047 1.09e-10 ***
## yearsq       6.274e-04  6.295e-05   9.968 9.22e-12 ***
## tmaxsq      -6.445e-02  4.245e-02  -1.518    0.138
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16.97 on 35 degrees of freedom
## Multiple R-squared:  0.7492, Adjusted R-squared:  0.7349
## F-statistic: 52.28 on 2 and 35 DF,  p-value: 3.074e-11
```

```
winne$allfit <- lmwinne$fitted.values

ggplot(winne) +
  geom_point(mapping = aes(x = year, y = yield)) +
  geom_line(mapping = aes(x = year, y = allfit, color="red")) +
  geom_line(mapping = aes(x = year, y = y_fitted, color="blue")) +
  theme_bw() +
  scale_colour_manual(name = "Model",
         values =c("red"="red","blue"="blue"), labels = c("Fit with Max Temp and Year"
  labs(x = "Year", y = "Corn Yield")
```

Adding maximum temperature trends to the model shows a similar trend, but peaks and dips in the fitted line highlight some of the outlying yield values and

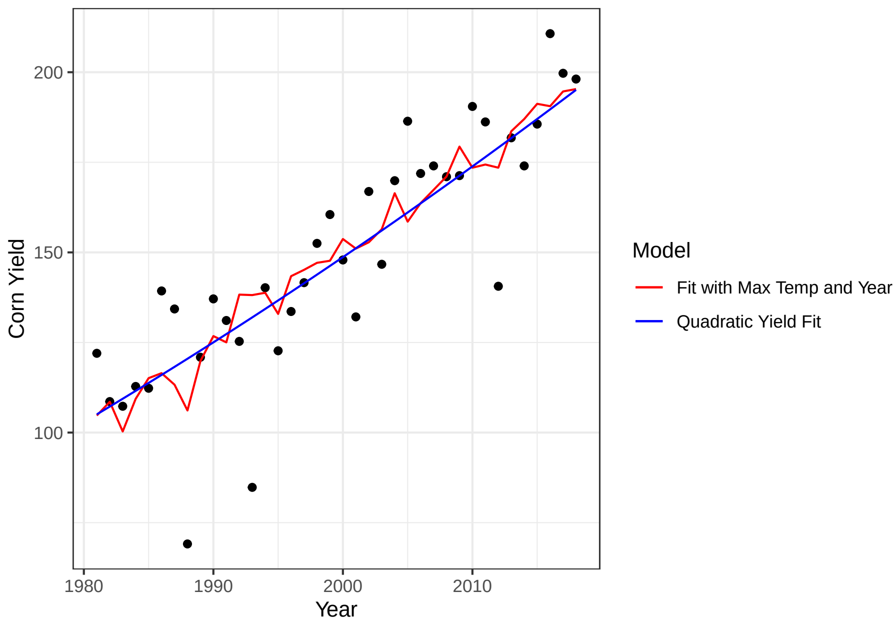


Figure 7.3: Comparative quadratic fit of corn yields over time (blue) and fitted line with maximum summer temperatures as well (red) in Winneshieck County, Iowa.

suggest an underlying relationship between maximum temperatures and yields. However, the relationship between squared maximum temperature and yield has a P-value of 0.14, compared with the year squared P-value of 9.22e-12, so it is clearly not the important driver of trends. This model has an R-squared value of 0.749, around the same as (slightly lower than) the simple linear regression model with only yield vs. year. Thus, adding temperature doesn't significantly add to our understanding of yield trends in Winneshieck County.

7.5 Cross-Section: Analyze the relationship between temperature and yield across all counties in 2018. Is there a relationship? Interpret the results.

```
corn2018 <- cornyields %>%
  filter(year == "2018") %>%
  mutate_at(vars(county_ansi), funs(factor))

tmax2018 <- tmaxdf %>%
  filter(year == "2018") %>%
  filter(doy >= 152 & doy <= 243) %>%
  group_by(countyfp) %>%
  rename("county_ansi" = "countyfp") %>%
  summarize(meantmax = mean(tmax))

yieldtemp_2018 <- inner_join(corn2018, tmax2018, by="county_ansi") %>%
  mutate(tmaxsq = (meantmax^2))

yt_lm <- lm(yield ~ meantmax + tmaxsq, data = yieldtemp_2018)
summary(yt_lm)
```

```
##
## Call:
## lm(formula = yield ~ meantmax + tmaxsq, data = yieldtemp_2018)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -44.221 -15.399   5.007  14.541  30.879
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5501.602   1860.830  -2.957  0.00397 **
```

```
## meantmax      406.789    131.493   3.094  0.00263 **
## tmaxsq         -7.256      2.321  -3.126  0.00239 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.75 on 90 degrees of freedom
## Multiple R-squared:  0.1317, Adjusted R-squared:  0.1124
## F-statistic: 6.827 on 2 and 90 DF,  p-value: 0.001736
```

```
yieldtemp_2018$ytfit <- yt_lm$fitted.values

ggplot(yieldtemp_2018) +
  geom_point(mapping = aes(x = meantmax, y = yield)) +
  geom_line(mapping = aes(x = meantmax, y = ytfit, color="red")) +
  theme_bw() + theme(legend.position="none") +
  labs(x = "Mean Max Temperature (C)", y = "Corn Yield")
```

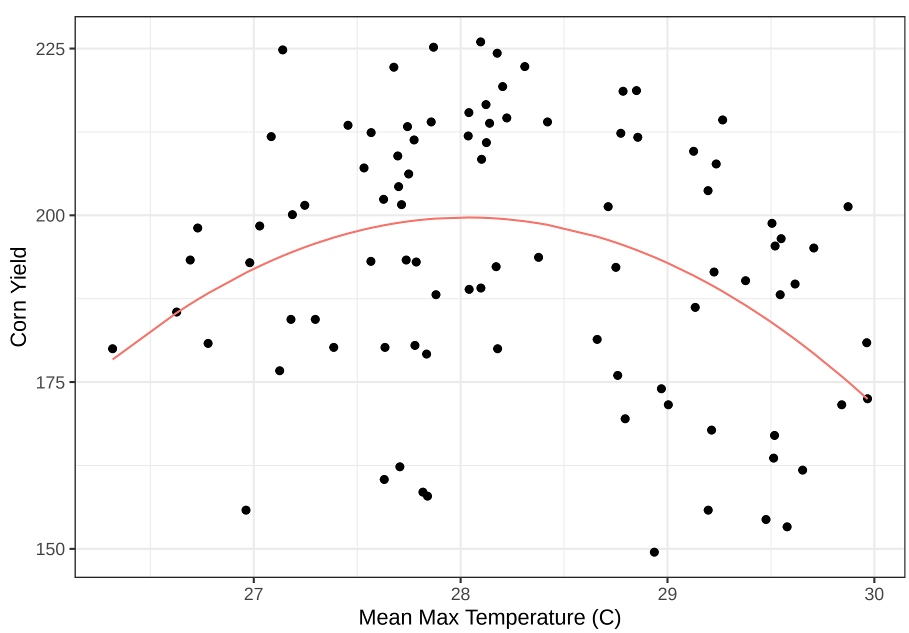


Figure 7.4: Quadratic fit of corn yields versus maximum summer temperatures (Degrees C) across Iowa.

There is a clear relationship with maximum temperatures and corn yields demonstrated in Figure 4. As we might expect, there appears to be a “sweet spot” in regard to temperature, with corn crops performing best at moderate temperatures and yields falling off at both low and high temperature years.

Lower mean maximum temperatures may indicate even lower temperatures that can shock crops, and high means are likely to cause high evaporation and withering. P<0.003 for the relationship between temperature and corn yield across Iowa.

7.6 Panel: One way to leverage multiple time series is to group all data into what is called a "panel" regression.

Convert the county ID code ("countyfp" or "county_ansi") into factor using as.factor, then include this variable in a regression using all counties' yield and summer temperature data. How does the significance of your temperature coefficients (Tmax, Tmax^2) change? Make a plot comparing actual and fitted yields and interpret the results of your model.

```
corn_all <- cornyields %>%
  mutate_at(vars(county_ansi), funs(factor))

tmax_all <- tmaxdf %>%
  filter(doy >= 152 & doy <= 243) %>%
  group_by(countyfp, year) %>%
  rename("county_ansi" = "countyfp") %>%
  summarize(meantmax = mean(tmax))
```

```
## `summarise()` has grouped output by 'county_ansi'. You can override using the
## `.groups` argument.
```

```
yieldtemp_all <- inner_join(corn_all, tmax_all) %>%
  mutate(tmaxsq = (meantmax^2)) %>%
  mutate(yearsq = year^2)
```

```
## Joining, by = c("county_ansi", "year")
```

```
ytc_lm <- lm(yield ~ year + meantmax + tmaxsq + county_ansi, data = yieldtemp_all)
summary(ytc_lm)
```

```
##
## Call:
## lm(formula = yield ~ year + meantmax + tmaxsq + county_ansi,
##     data = yieldtemp_all)
##
```

```
## Residuals:
##     Min      1Q  Median      3Q     Max
## -81.645  -9.720   1.924  13.232  40.409
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)    -5.826e+03  9.804e+01 -59.431  < 2e-16 ***
## year            2.203e+00  2.836e-02  77.664  < 2e-16 ***
## meantmax        1.182e+02  6.108e+00  19.352  < 2e-16 ***
## tmaxsq         -2.225e+00  1.085e-01 -20.503  < 2e-16 ***
## county_ansi3   -4.527e+00  4.321e+00  -1.048 0.294839
## county_ansi5    2.716e+00  4.343e+00   0.625 0.531743
## county_ansi7   -1.828e+01  4.350e+00  -4.203 2.70e-05 ***
## county_ansi9    5.068e+00  4.323e+00   1.172 0.241144
## county_ansi11   7.186e+00  4.325e+00   1.661 0.096732 .
## county_ansi13   7.289e+00  4.329e+00   1.684 0.092303 .
## county_ansi15   1.498e+01  4.323e+00   3.466 0.000534 ***
## county_ansi17   1.133e+01  4.332e+00   2.615 0.008966 **
## county_ansi19   7.651e+00  4.334e+00   1.765 0.077577 .
## county_ansi21   8.640e+00  4.328e+00   1.996 0.045974 *
## county_ansi23   9.089e+00  4.327e+00   2.100 0.035779 *
## county_ansi25   1.039e+01  4.326e+00   2.401 0.016400 *
## county_ansi27   9.666e+00  4.323e+00   2.236 0.025421 *
## county_ansi29   6.145e+00  4.321e+00   1.422 0.155092
## county_ansi31   1.579e+01  4.324e+00   3.651 0.000264 ***
## county_ansi33   4.582e+00  4.338e+00   1.056 0.290980
## county_ansi35   1.390e+01  4.325e+00   3.213 0.001325 **
## county_ansi37   2.169e+00  4.341e+00   0.500 0.617274
## county_ansi39  -2.404e+01  4.350e+00  -5.527 3.48e-08 ***
## county_ansi41   6.611e+00  4.329e+00   1.527 0.126809
## county_ansi43   8.864e+00  4.337e+00   2.044 0.041033 *
## county_ansi45   1.055e+01  4.325e+00   2.439 0.014756 *
## county_ansi47   6.528e+00  4.324e+00   1.510 0.131221
## county_ansi49   1.081e+01  4.321e+00   2.502 0.012386 *
## county_ansi51  -1.457e+01  4.352e+00  -3.349 0.000820 ***
## county_ansi53  -1.603e+01  4.350e+00  -3.686 0.000232 ***
## county_ansi55   9.423e+00  4.338e+00   2.172 0.029916 *
## county_ansi57   1.050e+01  4.321e+00   2.429 0.015186 *
## county_ansi59   2.906e+00  4.336e+00   0.670 0.502836
## county_ansi61   9.795e+00  4.340e+00   2.257 0.024059 *
## county_ansi63   7.232e+00  4.340e+00   1.666 0.095754 .
## county_ansi65   7.319e+00  4.341e+00   1.686 0.091905 .
## county_ansi67   4.791e+00  4.334e+00   1.106 0.269008
## county_ansi69   1.131e+01  4.330e+00   2.612 0.009035 **
## county_ansi71   1.358e+01  4.330e+00   3.136 0.001726 **
## county_ansi73   1.462e+01  4.321e+00   3.382 0.000727 ***
```

```
## county_ansi75   1.151e+01  4.328e+00   2.659 0.007863 **
## county_ansi77   3.379e+00  4.321e+00   0.782 0.434297
## county_ansi79   1.315e+01  4.324e+00   3.042 0.002370 **
## county_ansi81   8.706e+00  4.340e+00   2.006 0.044917 *
## county_ansi83   1.395e+01  4.326e+00   3.225 0.001271 **
## county_ansi85   6.891e+00  4.321e+00   1.595 0.110834
## county_ansi87   5.280e+00  4.321e+00   1.222 0.221864
## county_ansi89   9.433e-01  4.364e+00   0.216 0.828875
## county_ansi91   9.881e+00  4.334e+00   2.280 0.022661 *
## county_ansi93   1.186e+01  4.325e+00   2.743 0.006124 **
## county_ansi95   7.214e+00  4.322e+00   1.669 0.095161 .
## county_ansi97  -1.386e+00  4.330e+00  -0.320 0.748823
## county_ansi99   1.440e+01  4.322e+00   3.332 0.000871 ***
## county_ansi101  5.352e-01  4.325e+00   0.124 0.901510
## county_ansi103  4.380e+00  4.322e+00   1.013 0.310971
## county_ansi105  7.730e+00  4.328e+00   1.786 0.074158 .
## county_ansi107  2.203e+00  4.321e+00   0.510 0.610224
## county_ansi109  1.222e+01  4.335e+00   2.819 0.004839 **
## county_ansi111  1.779e+00  4.324e+00   0.411 0.680740
## county_ansi113  6.415e+00  4.326e+00   1.483 0.138218
## county_ansi115  7.330e+00  4.322e+00   1.696 0.089966 .
## county_ansi117 -2.168e+01  4.381e+00  -4.949 7.81e-07 ***
## county_ansi119  9.328e+00  4.325e+00   2.157 0.031063 *
## county_ansi121 -2.587e+00  4.321e+00  -0.599 0.549390
## county_ansi123  8.152e+00  4.321e+00   1.887 0.059302 .
## county_ansi125  1.919e+00  4.321e+00   0.444 0.656948
## county_ansi127  1.418e+01  4.326e+00   3.278 0.001055 **
## county_ansi129  1.023e+01  4.385e+00   2.332 0.019741 *
## county_ansi131  7.285e+00  4.352e+00   1.674 0.094242 .
## county_ansi133  7.987e-01  4.321e+00   0.185 0.853378
## county_ansi135 -1.585e+01  4.350e+00  -3.643 0.000273 ***
## county_ansi137  5.885e+00  4.322e+00   1.362 0.173381
## county_ansi139  8.283e+00  4.321e+00   1.917 0.055337 .
## county_ansi141  1.423e+01  4.328e+00   3.288 0.001018 **
## county_ansi143  8.743e+00  4.337e+00   2.016 0.043890 *
## county_ansi145 -3.674e-01  4.322e+00  -0.085 0.932261
## county_ansi147  7.261e+00  4.330e+00   1.677 0.093601 .
## county_ansi149  7.352e+00  4.322e+00   1.701 0.089007 .
## county_ansi151  1.150e+01  4.326e+00   2.659 0.007880 **
## county_ansi153  1.403e+01  4.321e+00   3.247 0.001178 **
## county_ansi155  1.127e+01  4.350e+00   2.590 0.009627 **
## county_ansi157  1.055e+01  4.322e+00   2.441 0.014702 *
## county_ansi159 -2.070e+01  4.321e+00  -4.792 1.72e-06 ***
## county_ansi161  9.390e+00  4.326e+00   2.170 0.030050 *
## county_ansi163  1.628e+01  4.323e+00   3.765 0.000169 ***
## county_ansi165  7.673e+00  4.323e+00   1.775 0.075966 .
```

```
## county_ansi167  1.558e+01  4.323e+00   3.603 0.000318 ***
## county_ansi169  1.122e+01  4.325e+00   2.593 0.009543 **
## county_ansi171  9.740e+00  4.325e+00   2.252 0.024387 *
## county_ansi173 -1.404e+01  4.350e+00  -3.228 0.001256 **
## county_ansi175 -1.155e+01  4.350e+00  -2.655 0.007967 **
## county_ansi177 -5.278e+00  4.329e+00  -1.219 0.222881
## county_ansi179 -3.220e+00  4.351e+00  -0.740 0.459267
## county_ansi181 -2.159e+00  4.321e+00  -0.500 0.617309
## county_ansi183  1.042e+01  4.321e+00   2.410 0.015981 *
## county_ansi185 -2.189e+01  4.350e+00  -5.033 5.07e-07 ***
## county_ansi187  1.421e+01  4.326e+00   3.285 0.001029 **
## county_ansi189  8.236e+00  4.344e+00   1.896 0.058035 .
## county_ansi191  4.567e+00  4.350e+00   1.050 0.293826
## county_ansi193  2.799e+00  4.321e+00   0.648 0.517252
## county_ansi195  6.123e+00  4.356e+00   1.406 0.159892
## county_ansi197  1.156e+01  4.329e+00   2.669 0.007634 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 18.83 on 3646 degrees of freedom
## Multiple R-squared:  0.7207, Adjusted R-squared:  0.7129
## F-statistic: 93.13 on 101 and 3646 DF,  p-value: < 2.2e-16
```

```r
yieldtemp_all$fittedyield <- ytc_lm$fitted.values

ggplot(yieldtemp_all, mapping = aes(x = fittedyield, y = yield)) +
  geom_point() +
  geom_smooth(method="lm") +
  theme_bw() + theme(legend.position="none") +
  labs(x = "Fitted Yield", y = "Actual Yield")
```

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

```r
par(mfrow=c(2,2))
plot(ytc_lm)
```

As a panel regression of all counties over all years, the statistical significance of year, mean maximum temperature, and squared maximum temperature as predictors of yield becomes stronger ($P<2e-16$ for each). The R squared value for the model is 0.721, indicating a pretty good fit, as is evident in Figure 5. However, the residuals for the model are pretty wide (Figures 5, 6) and the data may not be very normally distributed (Figure 6).

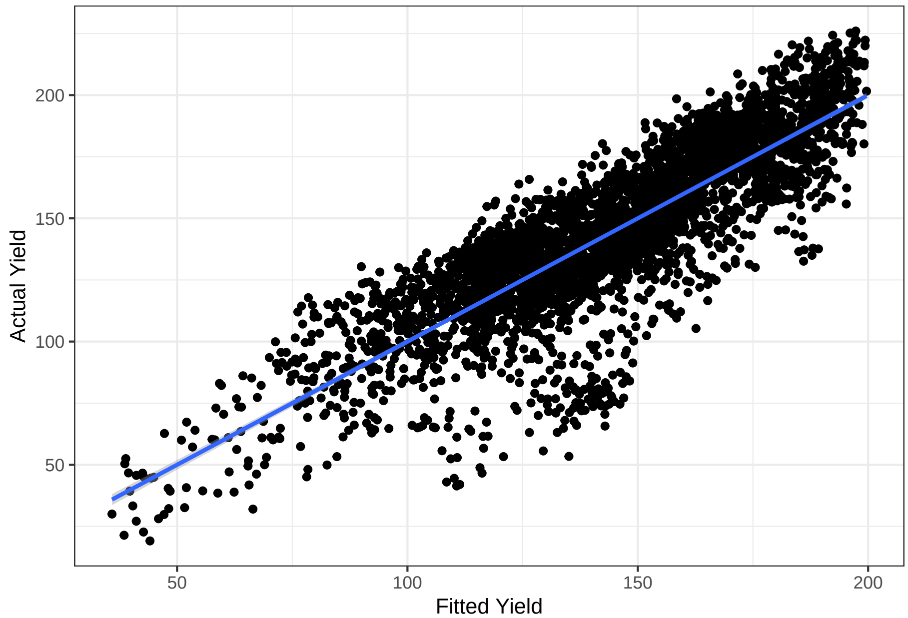


Figure 7.5: Fitted model yield values versus actual yield values for all counties of Iowa over all available years, from 1981 to 2018.

7.7 Soybeans: Download NASS data on soybean yields and explore either a time series relationship for a given county, the cross-sectional relationship for a given year, or a panel across all counties and years.

```
# parameters to query on
params2 <- list(commodity_desc = "SOYBEANS", prodn_practice_desc = "ALL PRODUCTION PRAC

# download
soyyieldsall <- nassqs_yields(params)
```

```
##   |                                                                               |
```

```
soyyieldsall$county_ansi <- as.numeric(soyyieldsall$county_ansi)
soyyieldsall$yield <- as.numeric(soyyieldsall$Value)
```

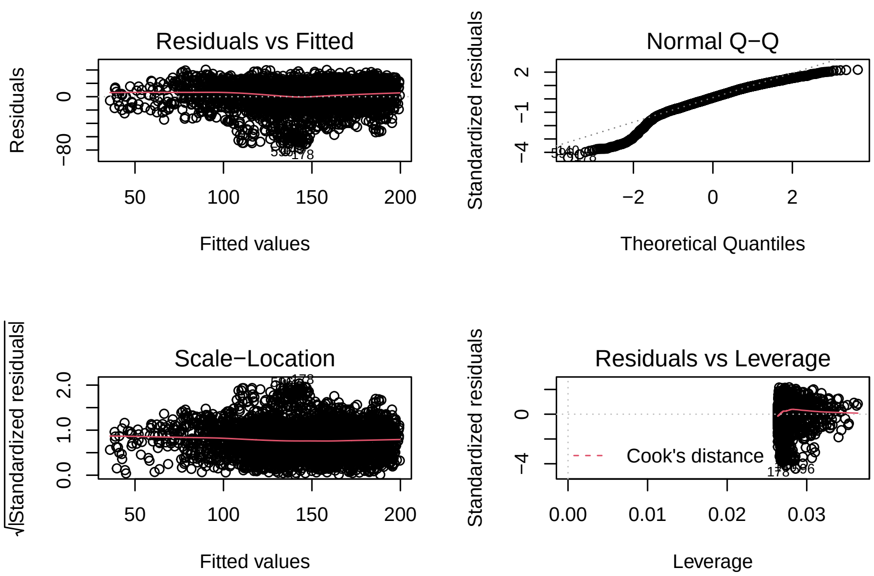


Figure 7.6: Residuals (top left), Normal Q-Q (top right), Scale-Location (bottom left), and Cook's Distance (bottom right) plots for the panel regression model.

```r
# clean and filter this dataset
soy <- select(soyyieldsall, county_ansi, county_name, yield, year) %>%
  filter(!is.na(county_ansi) & !is.na(yield))
soy <- tibble(soy)

soy_panel <- soy %>%
  mutate_at(vars(county_ansi), funs(factor)) %>%
  mutate(yearsq = year^2)

soypanel_lm <- lm(yield ~ year + yearsq + county_ansi, data = soy_panel)
summary(soypanel_lm)
```

```
##
## Call:
## lm(formula = yield ~ year + yearsq + county_ansi, data = soy_panel)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -100.865   -9.428    3.328   14.357   54.326
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      4.646e+04  1.090e+04   4.263 2.06e-05 ***
## year            -4.859e+01  1.089e+01  -4.461 8.38e-06 ***
## yearsq           1.272e-02  2.722e-03   4.672 3.09e-06 ***
## county_ansi3    -3.974e+00  4.809e+00  -0.826 0.408591
## county_ansi5     1.128e+01  4.749e+00   2.376 0.017550 *
## county_ansi7    -1.878e+01  4.778e+00  -3.931 8.61e-05 ***
## county_ansi9     9.877e+00  4.749e+00   2.080 0.037621 *
## county_ansi11    1.206e+01  4.749e+00   2.539 0.011162 *
## county_ansi13    1.391e+01  4.749e+00   2.930 0.003413 **
## county_ansi15    1.785e+01  4.749e+00   3.759 0.000173 ***
## county_ansi17    1.899e+01  4.749e+00   3.998 6.50e-05 ***
## county_ansi19    1.609e+01  4.749e+00   3.389 0.000709 ***
## county_ansi21    1.519e+01  4.749e+00   3.198 0.001396 **
## county_ansi23    1.612e+01  4.749e+00   3.394 0.000696 ***
## county_ansi25    1.603e+01  4.749e+00   3.375 0.000744 ***
## county_ansi27    1.435e+01  4.749e+00   3.023 0.002523 **
## county_ansi29    7.386e+00  4.749e+00   1.555 0.119953
## county_ansi31    1.942e+01  4.749e+00   4.088 4.43e-05 ***
## county_ansi33    1.250e+01  4.749e+00   2.632 0.008515 **
## county_ansi35    1.961e+01  4.778e+00   4.105 4.13e-05 ***
## county_ansi37    1.127e+01  4.749e+00   2.373 0.017672 *
## county_ansi39   -2.380e+01  4.841e+00  -4.916 9.19e-07 ***
## county_ansi41    1.302e+01  4.749e+00   2.742 0.006141 **
```

```
## county_ansi43   1.659e+01  4.749e+00   3.493 0.000483 ***
## county_ansi45   1.556e+01  4.749e+00   3.276 0.001061 **
## county_ansi47   1.300e+01  4.749e+00   2.737 0.006228 **
## county_ansi49   1.131e+01  4.749e+00   2.382 0.017259 *
## county_ansi51  -1.838e+01  4.778e+00  -3.847 0.000121 ***
## county_ansi53  -1.681e+01  4.841e+00  -3.472 0.000521 ***
## county_ansi55   1.775e+01  4.749e+00   3.738 0.000188 ***
## county_ansi57   9.205e+00  4.778e+00   1.926 0.054127 .
## county_ansi59   9.501e+00  4.749e+00   2.001 0.045507 *
## county_ansi61   1.838e+01  4.749e+00   3.871 0.000110 ***
## county_ansi63   1.503e+01  4.778e+00   3.145 0.001676 **
## county_ansi65   1.555e+01  4.749e+00   3.275 0.001065 **
## county_ansi67   1.283e+01  4.749e+00   2.702 0.006920 **
## county_ansi69   1.810e+01  4.749e+00   3.810 0.000141 ***
## county_ansi71   5.259e+00  4.778e+00   1.100 0.271183
## county_ansi73   1.644e+01  4.749e+00   3.461 0.000545 ***
## county_ansi75   1.853e+01  4.749e+00   3.901 9.75e-05 ***
## county_ansi77   5.347e+00  4.749e+00   1.126 0.260249
## county_ansi79   1.771e+01  4.778e+00   3.706 0.000214 ***
## county_ansi81   1.716e+01  4.749e+00   3.614 0.000306 ***
## county_ansi83   1.836e+01  4.749e+00   3.866 0.000112 ***
## county_ansi85   6.372e+00  4.809e+00   1.325 0.185252
## county_ansi87   2.204e+00  4.749e+00   0.464 0.642690
## county_ansi89   9.845e+00  4.749e+00   2.073 0.038239 *
## county_ansi91   1.758e+01  4.749e+00   3.703 0.000216 ***
## county_ansi93   1.808e+01  4.778e+00   3.784 0.000156 ***
## county_ansi95   9.245e+00  4.778e+00   1.935 0.053090 .
## county_ansi97   5.382e+00  4.749e+00   1.133 0.257218
## county_ansi99   1.823e+01  4.778e+00   3.815 0.000138 ***
## county_ansi101 -6.065e+00  4.749e+00  -1.277 0.201671
## county_ansi103  6.598e+00  4.778e+00   1.381 0.167447
## county_ansi105  1.373e+01  4.778e+00   2.874 0.004069 **
## county_ansi107  1.030e+00  4.749e+00   0.217 0.828258
## county_ansi109  1.968e+01  4.749e+00   4.145 3.47e-05 ***
## county_ansi111 -4.248e+00  4.749e+00  -0.894 0.371157
## county_ansi113  1.244e+01  4.778e+00   2.604 0.009251 **
## county_ansi115  4.608e+00  4.749e+00   0.970 0.331929
## county_ansi117 -2.150e+01  4.841e+00  -4.441 9.20e-06 ***
## county_ansi119  1.472e+01  4.749e+00   3.100 0.001952 **
## county_ansi121 -1.277e+00  4.749e+00  -0.269 0.788032
## county_ansi123  8.308e+00  4.749e+00   1.749 0.080294 .
## county_ansi125  1.995e+00  4.778e+00   0.418 0.676297
## county_ansi127  2.112e+01  4.778e+00   4.419 1.02e-05 ***
## county_ansi129  5.192e+00  4.809e+00   1.080 0.280353
## county_ansi131  1.591e+01  4.749e+00   3.349 0.000818 ***
## county_ansi133  1.767e-01  4.749e+00   0.037 0.970326
```

```
## county_ansi135 -1.612e+01  4.778e+00  -3.373 0.000751 ***
## county_ansi137  4.108e+00  4.749e+00   0.865 0.387051
## county_ansi139  8.369e+00  4.749e+00   1.762 0.078100 .
## county_ansi141  2.129e+01  4.749e+00   4.483 7.57e-06 ***
## county_ansi143  1.615e+01  4.749e+00   3.400 0.000680 ***
## county_ansi145 -2.245e+00  4.749e+00  -0.473 0.636402
## county_ansi147  1.413e+01  4.749e+00   2.975 0.002950 **
## county_ansi149  1.129e+01  4.778e+00   2.362 0.018212 *
## county_ansi151  1.801e+01  4.778e+00   3.770 0.000166 ***
## county_ansi153  1.401e+01  4.749e+00   2.949 0.003206 **
## county_ansi155  1.124e+01  4.778e+00   2.352 0.018734 *
## county_ansi157  1.181e+01  4.749e+00   2.487 0.012908 *
## county_ansi159 -2.035e+01  4.778e+00  -4.258 2.11e-05 ***
## county_ansi161  1.602e+01  4.749e+00   3.373 0.000751 ***
## county_ansi163  1.991e+01  4.749e+00   4.192 2.83e-05 ***
## county_ansi165  1.242e+01  4.778e+00   2.599 0.009393 **
## county_ansi167  2.019e+01  4.749e+00   4.252 2.17e-05 ***
## county_ansi169  1.555e+01  4.749e+00   3.275 0.001067 **
## county_ansi171  1.451e+01  4.749e+00   3.055 0.002266 **
## county_ansi173 -1.499e+01  4.778e+00  -3.137 0.001720 **
## county_ansi175 -1.023e+01  4.778e+00  -2.142 0.032273 *
## county_ansi177 -1.466e+01  4.749e+00  -3.086 0.002041 **
## county_ansi179 -5.652e+00  4.778e+00  -1.183 0.236907
## county_ansi181 -2.277e+00  4.809e+00  -0.473 0.635891
## county_ansi183  8.595e+00  4.778e+00   1.799 0.072137 .
## county_ansi185 -2.086e+01  4.778e+00  -4.365 1.30e-05 ***
## county_ansi187  2.017e+01  4.749e+00   4.247 2.21e-05 ***
## county_ansi189  1.687e+01  4.749e+00   3.552 0.000387 ***
## county_ansi191  1.306e+01  4.749e+00   2.750 0.005980 **
## county_ansi193  6.472e+00  4.749e+00   1.363 0.173049
## county_ansi195  1.503e+01  4.749e+00   3.164 0.001566 **
## county_ansi197  1.807e+01  4.749e+00   3.805 0.000144 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 21.37 on 3914 degrees of freedom
## Multiple R-squared:  0.6598, Adjusted R-squared:  0.6511
## F-statistic: 75.91 on 100 and 3914 DF,  p-value: < 2.2e-16
```

```
soy_panel$fittedyield <- soypanel_lm$fitted.values

ggplot(soy_panel, mapping = aes(x = year, y = yield)) +
  geom_point() +
  geom_line(mapping = aes(x = year, y = fittedyield, color="red")) +
  geom_smooth(method="lm") +
```

```
  theme_bw() + theme(legend.position="none") +
  labs(x = "Year", y = "Soy Yield")
```

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

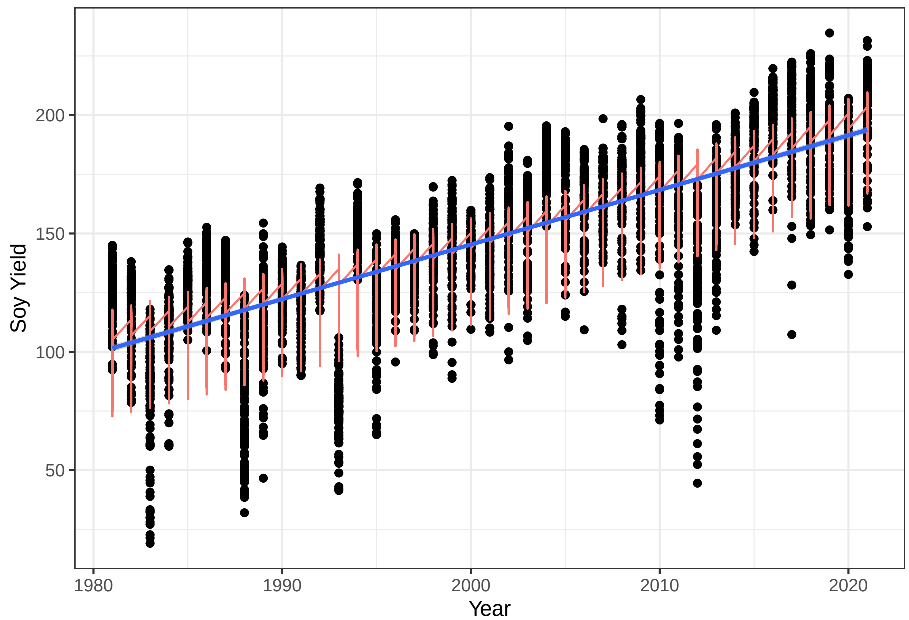


Figure 7.7: Soy yields over time (years) across all counties of Iowa, with a panel fit (orange), and linear fit (blue). Panel regression R squared is 0.660, with p<8.5e-06 for year and years squared versus yield.

Like with corn yields, soy yields in Iowa follow an upward trend over time, though with wide residuals (Figure 7).

7.8 Bonus: Find a package to make a county map of Iowa displaying some sort of information about yields or weather. Interpret your map.

```
library(sf) #Spatial package that can read and create shapefiles
library(mapview) #Interactive maps
```

```
library(USAboundaries) #USA states and counties

counties <- us_counties

Iowa_ct <- counties(states = 'iowa')
#str(Iowa_ct)
#mapview(Iowa_ct)

summer2018 <- tmaxdf %>%
  filter(doy >= 152 & doy <= 243, year==2018) %>% #day 152= June 1, 243= Aug 31
  group_by(countyfp, year) %>%
  summarize(meantmax = mean(tmax))
```

```
## `summarise()` has grouped output by 'countyfp'. You can override using the
## `.groups` argument.
```

```
Itemp_2018 <- merge(Iowa_ct, summer2018)

mapview(Itemp_2018, zcol ='meantmax', col.regions=brewer.pal(9, "OrRd"), layer.name = '
```

Chapter 8

Multivariate Statistics and Principle Components Analysis

A lot of data that we try to analyze is *multivariate*, meaning that the data has multiple records or observations with multiple variables. There are various ways to look at this type of data and describe associations of variables. We may describe associations through *covariance* or a *correlation coefficient*. *Principle Components Analysis (PCA)* is a useful tool for looking at correlation that uses orthogonal transformation to convert observations with potentially correlated variables into a set of values of linearly uncorrelated variables (called principal components). In order to explore PCA as a tool, we looked at wine and how variables of wine are correlated with one another.

Data and assignment provided by Dr. Michael Lefsky of Colorado State University.

8.1 Scatterplot matrix of variables

```
# generate a scatterplot of variables 2 to 6 where each cultivar is shown in a different color
pairs(wine[,2:6], main = "Wine Data -- 3 cultivars",
      pch = 21, bg = c("red", "green3", "blue")[unclass(wine$cultivar)])
```

From the above plot, alcohol and alkalinity don't seem to have a very strong linear relationship, but it appears that the weak correlation is negative, with alkalinity decreasing as alcohol increases and vice versa. Ash and alkalinity seem

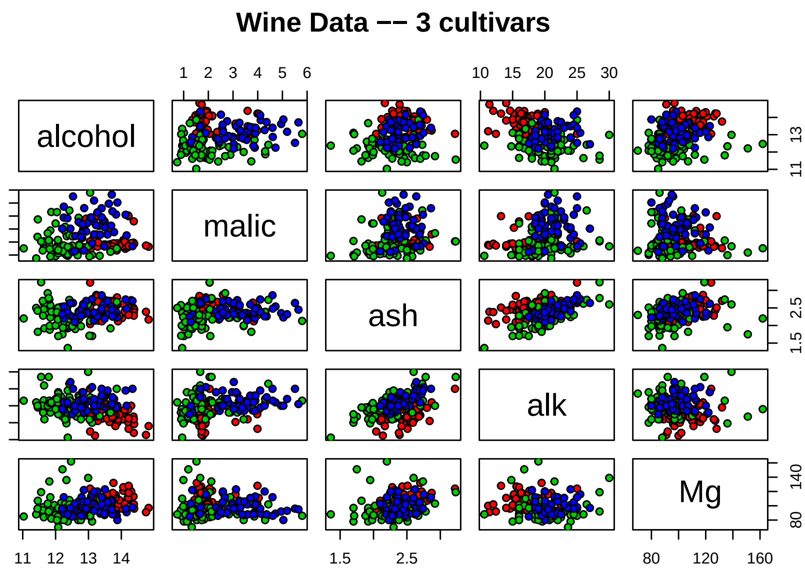


Figure 8.1: Scatterplot matrix demonstrating the relationships between five variables for three wine cultivars. Alcohol, malic acid (malic), ash, alkalinity of ash (alk), and magnesium (Mg) are plotted together on both x and y axes for cultivar 1 (red), cultivar 2 (green), and cultivar 3 (blue). Groupings demonstrate association between variables.

to have a slightly stronger linear relationship that is positive; as ash increases, so does alkalinity, and vice versa.

8.2 Correlation matrix

```
# Generate a correlation matrix between variables 2 to 14
cor(wine[,2:14])
```

```
##                alcohol       malic         ash         alk          Mg
## alcohol      1.00000000  0.09439694  0.211544596 -0.31023514  0.27079823
## malic        0.09439694  1.00000000  0.164045470  0.28850040 -0.05457510
## ash          0.21154460  0.16404547  1.000000000  0.44336719  0.28658669
## alk         -0.31023514  0.28850040  0.443367187  1.00000000 -0.08333309
## Mg           0.27079823 -0.05457510  0.286586691 -0.08333309  1.00000000
## totphen      0.28910112 -0.33516700  0.128979538 -0.32111332  0.21440123
## flavan       0.23681493 -0.41100659  0.115077279 -0.35136986  0.19578377
## nonflavphen -0.15592947  0.29297713  0.186230446  0.36192172 -0.25629405
## proantho     0.13669791 -0.22074619  0.009651935 -0.19732684  0.23644061
## color        0.54636420  0.24898534  0.258887259  0.01873198  0.19995001
## hue         -0.07174720 -0.56129569 -0.074666889 -0.27395522  0.05539820
## OD           0.07234319 -0.36871043  0.003911231 -0.27676855  0.06600394
## proline      0.64372004 -0.19201056  0.223626264 -0.44059693  0.39335085
##                totphen      flavan nonflavphen    proantho       color
## alcohol      0.28910112  0.2368149  -0.1559295  0.136697912  0.54636420
## malic       -0.33516700 -0.4110066   0.2929771 -0.220746187  0.24898534
## ash          0.12897954  0.1150773   0.1862304  0.009651935  0.25888726
## alk         -0.32111332 -0.3513699   0.3619217 -0.197326836  0.01873198
## Mg           0.21440123  0.1957838  -0.2562940  0.236440610  0.19995001
## totphen      1.00000000  0.8645635  -0.4499353  0.612413084 -0.05513642
## flavan       0.86456350  1.0000000  -0.5378996  0.652691769 -0.17237940
## nonflavphen -0.44993530 -0.5378996   1.0000000 -0.365845099  0.13905701
## proantho     0.61241308  0.6526918  -0.3658451  1.000000000 -0.02524993
## color       -0.05513642 -0.1723794   0.1390570 -0.025249931  1.00000000
## hue          0.43368134  0.5434786  -0.2626396  0.295544253 -0.52181319
## OD           0.69994936  0.7871939  -0.5032696  0.519067096 -0.42881494
## proline      0.49811488  0.4941931  -0.3113852  0.330416700  0.31610011
##                    hue          OD    proline
## alcohol     -0.07174720  0.072343187  0.6437200
## malic       -0.56129569 -0.368710428 -0.1920106
## ash         -0.07466689  0.003911231  0.2236263
## alk         -0.27395522 -0.276768549 -0.4405969
## Mg           0.05539820  0.066003936  0.3933508
## totphen      0.43368134  0.699949365  0.4981149
```

```
## flavan       0.54347857  0.787193902  0.4941931
## nonflavphen -0.26263963 -0.503269596 -0.3113852
## proantho     0.29554425  0.519067096  0.3304167
## color       -0.52181319 -0.428814942  0.3161001
## hue          1.00000000  0.565468293  0.2361834
## OD           0.56546829  1.000000000  0.3127611
## proline      0.23618345  0.312761075  1.0000000
```

Correlation matrix for variables for wine, including alcohol, malic acid (malic), ash, alkalinity of ash (alk), magnesium (Mg), total phenol (totphen), flavonoids (flavan), nonflavanoid phenols (nonflavphen), proanthocyanins (proantho), color intensity (color), hue, OD280/OD315 of diluted wines (OD), and proline.

8.3 Calculate variances

```
# Calculate variances for variables 2 to 14 for all variables together
var(wine[,2:14]) # the diagonal values in the matrix are variances
```

```
##                   alcohol        malic           ash          alk           Mg
## alcohol       0.65906233   0.08561131  0.0471151590   -0.8410929    3.1398781
## malic         0.08561131   1.24801540  0.0502770393    1.0763317   -0.8707795
## ash           0.04711516   0.05027704  0.0752646353    0.4062083    1.1229366
## alk          -0.84109290   1.07633171  0.4062082778   11.1526862   -3.9747604
## Mg            3.13987812  -0.87077953  1.1229365835   -3.9747604  203.9893354
## totphen       0.14688722  -0.23433772  0.0221455913   -0.6711491    1.9164699
## flavan        0.19203322  -0.45863037  0.0315347299   -1.1720828    2.7930870
## nonflavphen  -0.01575426   0.04073336  0.0063584714    0.1504219   -0.4555634
## proantho      0.06351752  -0.14114698  0.0015155780   -0.3771762    1.9328325
## color         1.02828254   0.64483818  0.1646543266    0.1450242    6.6205206
## hue          -0.01331344  -0.14332564 -0.0046821545   -0.2091181    0.1808513
## OD            0.04169782  -0.29244748  0.0007618358   -0.6562344    0.6693081
## proline     164.56718498 -67.54886657 19.3197390973 -463.3553450 1769.1586999
##                 totphen       flavan   nonflavphen     proantho        color
## alcohol       0.14688722   0.19203322  -0.015754260  0.063517520   1.02828254
## malic        -0.23433772  -0.45863037   0.040733362 -0.141146982   0.64483818
## ash           0.02214559   0.03153473   0.006358471  0.001515578   0.16465433
## alk          -0.67114915  -1.17208281   0.150421856 -0.377176220   0.14502419
## Mg            1.91646988   2.79308703  -0.455563385  1.932832476   6.62052061
## totphen       0.39168954   0.54047042  -0.035045125  0.219373345  -0.07999752
## flavan        0.54047042   0.99771867  -0.066867000  0.373147553  -0.39916863
## nonflavphen  -0.03504512  -0.06686700   0.015488634 -0.026059868   0.04012051
## proantho      0.21937334   0.37314755  -0.026059868  0.327594668  -0.03350392
```

```
## color       -0.07999752  -0.39916863   0.040120510 -0.033503918   5.37444938
## hue          0.06203888   0.12408197  -0.007471177  0.038664565  -0.27650580
## OD           0.31102128   0.55826225  -0.044469244  0.210932940  -0.70581258
## proline     98.17105726 155.44749222 -12.203586301 59.554333778 230.76748014
##                     hue            OD     proline
## alcohol     -0.013313443  0.0416978226   164.56718
## malic       -0.143325638 -0.2924474830   -67.54887
## ash         -0.004682155  0.0007618358    19.31974
## alk         -0.209118054 -0.6562343681  -463.35535
## Mg           0.180851266  0.6693080683  1769.15870
## totphen      0.062038876  0.3110212785    98.17106
## flavan       0.124081969  0.5582622548   155.44749
## nonflavphen -0.007471177 -0.0444692440   -12.20359
## proantho     0.038664565  0.2109329398    59.55433
## color       -0.276505801 -0.7058125762   230.76748
## hue          0.052244961  0.0917662439    17.00022
## OD           0.091766244  0.5040864089    69.92753
## proline     17.000223386 69.9275255507 99166.71736
```

By testing the variances of variables, we can look at whether they need to be standardized in order to get an accurate representation of variable influence in a PCA. In this case, we do need to standardize, because the variances are quite different between variables.

8.4 Standardizing variables

```
# You can standardize variables in R using the "scale()" function
wine.standardized <-as.data.frame(scale(wine[,2:14]))

sapply(wine.standardized, mean) # for calculating mean of all variables
```

```
##       alcohol         malic           ash           alk            Mg
## -8.591766e-16 -6.776446e-17  8.045176e-16 -7.720494e-17 -4.073935e-17
##       totphen        flavan   nonflavphen      proantho         color
## -1.395560e-17  6.958263e-17 -1.042186e-16 -1.221369e-16  3.649376e-17
##           hue            OD       proline
##  2.093741e-16  3.003459e-16 -1.034429e-16
```

```
sapply(wine.standardized, var) # for calculating variance of all variables
```

```
##     alcohol       malic         ash         alk          Mg     totphen
```

```
##           1           1           1           1           1           1
##      flavan nonflavphen    proantho       color         hue          OD
##           1           1           1           1           1           1
##     proline
##           1
```

8.5 PCA on standardized data

```
#Perform PCA on standardized data
wine.pca <- prcomp(wine.standardized)

# print summary of the PCA
summary(wine.pca)
```

```
## Importance of components:
##                          PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.169 1.5802 1.2025 0.95863 0.92370 0.80103 0.74231
## Proportion of Variance 0.362 0.1921 0.1112 0.07069 0.06563 0.04936 0.04239
## Cumulative Proportion  0.362 0.5541 0.6653 0.73599 0.80162 0.85098 0.89337
##                            PC8     PC9   PC10    PC11    PC12    PC13
## Standard deviation     0.59034 0.53748 0.5009 0.47517 0.41082 0.32152
## Proportion of Variance 0.02681 0.02222 0.0193 0.01737 0.01298 0.00795
## Cumulative Proportion  0.92018 0.94240 0.9617 0.97907 0.99205 1.00000
```

```
screeplot(wine.pca)
```

```
# Examine the loadings from the standardized data
wine.pca$rotation
```

```
##                      PC1          PC2         PC3         PC4         PC5
## alcohol     -0.144329395  0.483651548 -0.20738262  0.01785630 -0.26566365
## malic        0.245187580  0.224930935  0.08901289 -0.53689028  0.03521363
## ash          0.002051061  0.316068814  0.62622390  0.21417556 -0.14302547
## alk          0.239320405 -0.010590502  0.61208035 -0.06085941  0.06610294
## Mg          -0.141992042  0.299634003  0.13075693  0.35179658  0.72704851
## totphen     -0.394660845  0.065039512  0.14617896 -0.19806835 -0.14931841
## flavan      -0.422934297 -0.003359812  0.15068190 -0.15229479 -0.10902584
## nonflavphen  0.298533103  0.028779488  0.17036816  0.20330102 -0.50070298
## proantho    -0.313429488  0.039301722  0.14945431 -0.39905653  0.13685982
## color        0.088616705  0.529995672 -0.13730621 -0.06592568 -0.07643678
## hue         -0.296714564 -0.279235148  0.08522192  0.42777141 -0.17361452
```

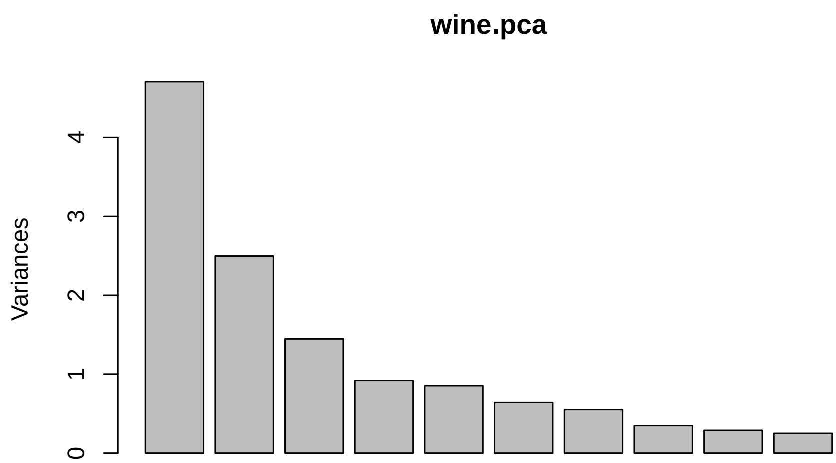


Figure 8.2: Screeplot summary of variances in standardized data.

```
## OD          -0.376167411 -0.164496193  0.16600459 -0.18412074 -0.10116099
## proline     -0.286752227  0.364902832 -0.12674592  0.23207086 -0.15786880
##                      PC6          PC7         PC8         PC9        PC10
## alcohol      0.21353865 -0.05639636  0.39613926 -0.50861912  0.21160473
## malic        0.53681385  0.42052391  0.06582674  0.07528304 -0.30907994
## ash          0.15447466 -0.14917061 -0.17026002  0.30769445 -0.02712539
## alk         -0.10082451 -0.28696914  0.42797018 -0.20044931  0.05279942
## Mg           0.03814394  0.32288330 -0.15636143 -0.27140257  0.06787022
## totphen     -0.08412230 -0.02792498 -0.40593409 -0.28603452 -0.32013135
## flavan      -0.01892002 -0.06068521 -0.18724536 -0.04957849 -0.16315051
## nonflavphen -0.25859401  0.59544729 -0.23328465 -0.19550132  0.21553507
## proantho    -0.53379539  0.37213935  0.36822675  0.20914487  0.13418390
## color       -0.41864414 -0.22771214 -0.03379692 -0.05621752 -0.29077518
## hue          0.10598274  0.23207564  0.43662362 -0.08582839 -0.52239889
## OD           0.26585107 -0.04476370 -0.07810789 -0.13722690  0.52370587
## proline      0.11972557  0.07680450  0.12002267  0.57578611  0.16211600
##                     PC11        PC12        PC13
## alcohol      0.22591696 -0.26628645  0.01496997
## malic       -0.07648554  0.12169604  0.02596375
## ash          0.49869142 -0.04962237 -0.14121803
## alk         -0.47931378 -0.05574287  0.09168285
## Mg          -0.07128891  0.06222011  0.05677422
```

```
## totphen     -0.30434119 -0.30388245 -0.46390791
## flavan       0.02569409 -0.04289883  0.83225706
## nonflavphen -0.11689586  0.04235219  0.11403985
## proantho     0.23736257 -0.09555303 -0.11691707
## color       -0.03183880  0.60422163 -0.01199280
## hue          0.04821201  0.25921400 -0.08988884
## OD          -0.04642330  0.60095872 -0.15671813
## proline     -0.53926983 -0.07940162  0.01444734
```

```
# Examine the biplot which shows the loadings in the first 2 principal components
biplot(wine.pca)
```

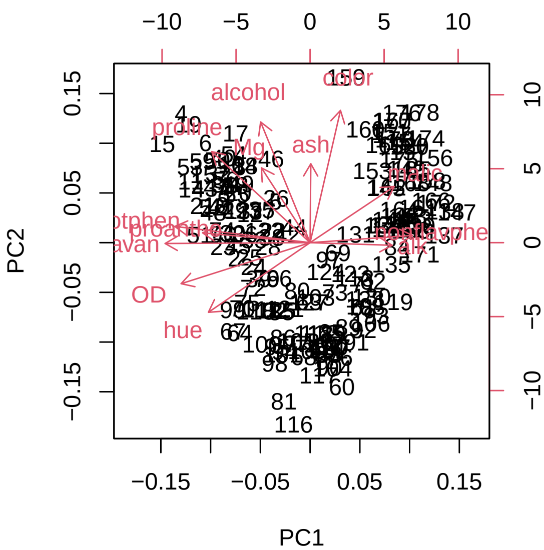


Figure 8.3: Biplot of the first two principal components for standardized data of three wine cultivars (1-59, 60-130, and 131-178), and loadings for variables alcohol, malic acid (malic), ash, alkalinity of ash (alk), magnesium (Mg), total phenol (totphen), flavonoids (flavan), nonflavanoid phenols (nonflavphen), proanthocyanins (proantho), color intensity (color), hue, OD280/OD315 of diluted wines (OD), and proline.

```
# Save the variable loadings to a .csv file
#write.csv(wine.pca$rotation, file="wine_pca_loadings.csv")
```

```
# Plot the scores from PC1 and PC2 and add labels
plot(wine.pca$x[,1], wine.pca$x[,2], main="Scores from PC1 and PC2")
text(wine.pca$x[,1], wine.pca$x[,2], wine$cultivar,cex=0.7,pos=4,col="red")    #add labels
```

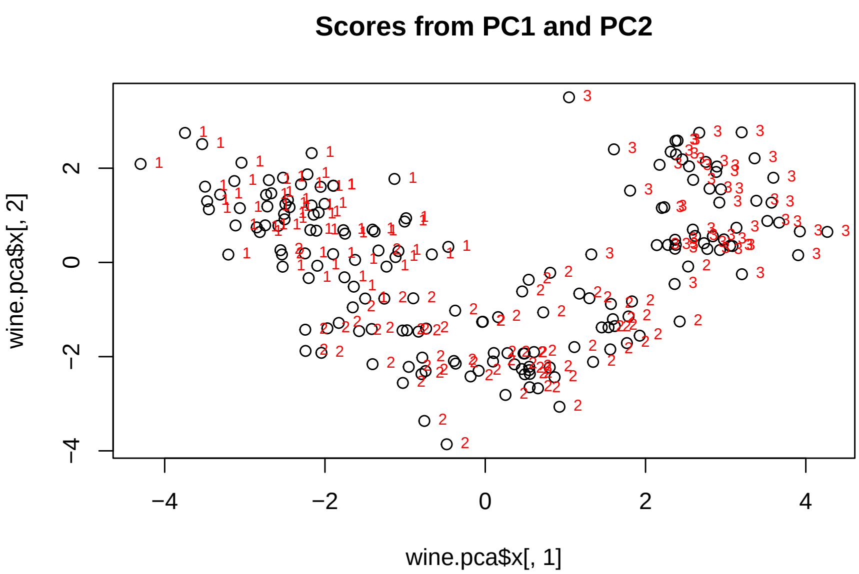


Figure 8.4: Scatterplot of principal component scores for standardized data of three wine cultivars (1, 2, 3). The first principal component scores are on the x axis, and the second principal component scores are on the y axis. Cultivar 1 is grouped in the upper left, with majority negative scores for principal component 1 and positive scores for principal component 2. Cultivar 2 is grouped with negative scores for principal component 2 and between -2 and 3 for principal component 1. Cultivar 3 is grouped in mostly positive scores for both components.

```
# Plot the scores from PC3 and PC4 and add labels
plot(wine.pca$x[,3], wine.pca$x[,4], main="Scores from PC3 and PC4")
text(wine.pca$x[,3], wine.pca$x[,4], wine$cultivar,cex=0.7,pos=4,col="red")    #add labels
```

8.6 PCA on raw data

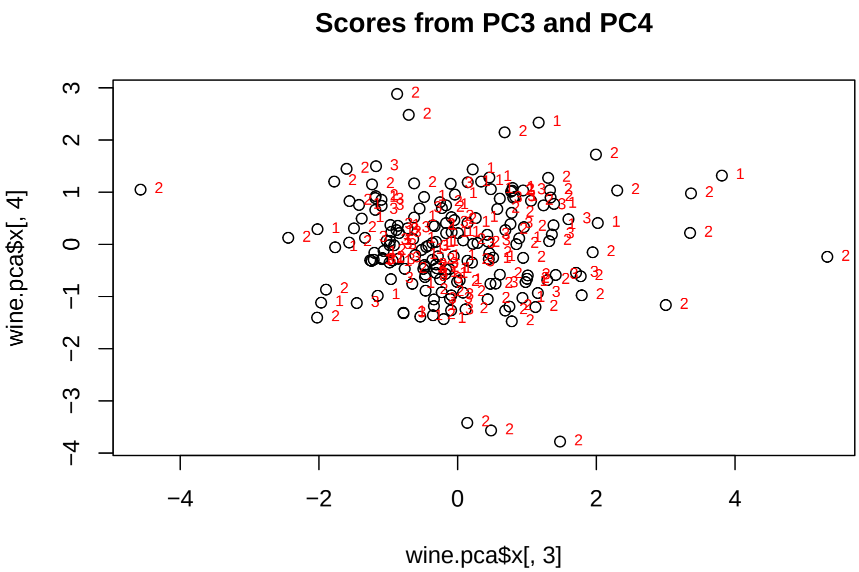


Figure 8.5: Scatterplot of principal component scores for three wine cultivars (1, 2, 3). The third principal component scores are on the x axis, and the fourth principal component scores are on the y axis. All three cultivars primarily group in the center, with some outliers, mostly from cultivar 2.

```
#Perform PCA on the raw data
wine.pca.raw <- prcomp(wine[,2:14])

# print summary of the Raw PCA
summary(wine.pca.raw)
```

```
## Importance of components:
##                             PC1      PC2     PC3     PC4     PC5     PC6    PC7
## Standard deviation     314.9632 13.13527 3.07215 2.23409 1.10853 0.91710 0.5282
## Proportion of Variance   0.9981  0.00174 0.00009 0.00005 0.00001 0.00001 0.0000
## Cumulative Proportion    0.9981  0.99983 0.99992 0.99997 0.99998 0.99999 1.0000
##                           PC8    PC9   PC10   PC11   PC12    PC13
## Standard deviation     0.3891 0.3348 0.2678 0.1938 0.1452 0.09057
## Proportion of Variance 0.0000 0.0000 0.0000 0.0000 0.0000 0.00000
## Cumulative Proportion  1.0000 1.0000 1.0000 1.0000 1.0000 1.00000
```

```
screeplot(wine.pca.raw)
```

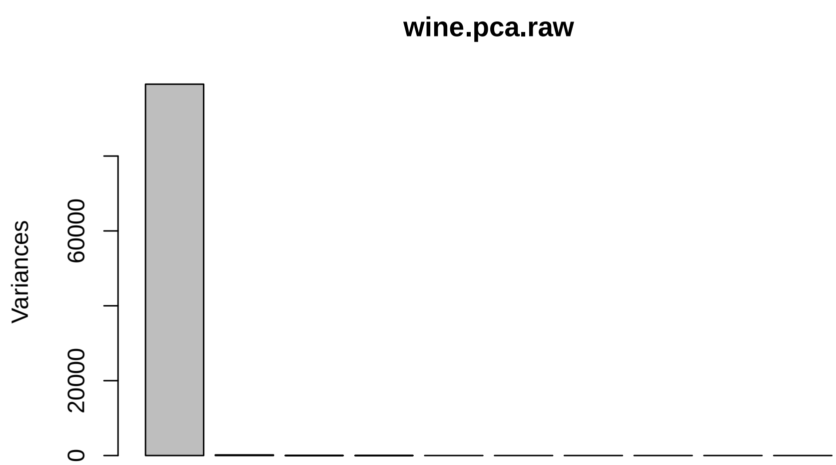


Figure 8.6: Screeplot summary of variances in raw data.

```
# Examine the loadings from the raw data
wine.pca.raw$rotation
```

```
##                       PC1           PC2          PC3          PC4          PC5
## alcohol     -0.0016592647 -1.203406e-03 -0.016873809  0.141446778  0.020336977
## malic        0.0006810156 -2.154982e-03 -0.122003373  0.160389543 -0.612883454
## ash         -0.0001949057 -4.593693e-03 -0.051987430 -0.009772810  0.020175575
## alk          0.0046713006 -2.645039e-02 -0.938593003 -0.330965260  0.064352340
## Mg          -0.0178680075 -9.993442e-01  0.029780248 -0.005393756 -0.006149345
## totphen     -0.0009898297 -8.779622e-04  0.040484644 -0.074584656  0.315245063
## flavan      -0.0015672883  5.185073e-05  0.085443339 -0.169086724  0.524761088
## nonflavphen  0.0001230867  1.354479e-03 -0.013510780  0.010805561 -0.029647512
## proantho    -0.0006006078 -5.004400e-03  0.024659382 -0.050120952  0.251182529
## color       -0.0023271432 -1.510035e-02 -0.291398464  0.878893693  0.331747051
## hue         -0.0001713800  7.626731e-04  0.025977662 -0.060034945  0.051524077
## OD          -0.0007049316  3.495364e-03  0.070323969 -0.178200254  0.260639176
## proline     -0.9998229365  1.777381e-02 -0.004528682 -0.003112916 -0.002298569
##                      PC6          PC7           PC8           PC9          PC10
## alcohol     -0.194120104  0.923280337 -2.848207e-01 -8.660061e-02  2.245000e-03
## malic       -0.742472963 -0.150109941  6.467447e-02 -1.566214e-02  1.850935e-02
## ash         -0.041752912  0.045009549  1.493395e-01 -7.364985e-02  8.679965e-02
## alk          0.024065303  0.031526583 -1.515391e-02 -2.044578e-03 -3.554028e-03
## Mg           0.001923782  0.001797363  3.552212e-03  1.963668e-03  4.051542e-05
## totphen     -0.278716809 -0.020185710  1.772379e-01 -2.556729e-01 -8.471951e-01
## flavan      -0.433597955 -0.038868518  2.481166e-01 -3.783067e-01  5.201384e-01
## nonflavphen  0.021952834 -0.004665483 -6.497968e-03 -3.675204e-02 -3.771319e-02
## proantho    -0.241884488 -0.309799487 -8.704332e-01  5.152017e-02 -9.722752e-03
## color       -0.002739609 -0.112836514  8.128692e-02  9.902908e-02  2.314712e-02
## hue          0.023776167  0.030819813  2.951904e-03 -3.306512e-02  3.846983e-02
## OD          -0.288912753  0.101973518  1.867145e-01  8.737465e-01 -1.701708e-02
## proline      0.001212255 -0.001076189 -1.034095e-05  7.255852e-05 -4.926638e-05
##                      PC11          PC12          PC13
## alcohol     -0.0149715080 -1.565141e-02  8.029245e-03
## malic       -0.0231876506  6.729555e-02 -1.109039e-02
## ash          0.9540106426 -1.320630e-01 -1.736857e-01
## alk         -0.0528216953  5.393806e-03  1.939563e-03
## Mg          -0.0030248882  6.208885e-04  2.284536e-03
## totphen      0.0088016070  3.882903e-03 -2.669144e-02
## flavan      -0.1332046120 -3.748803e-02  6.959853e-02
## nonflavphen  0.1991789841  1.475524e-01  9.664662e-01
## proantho     0.1356214601 -1.311883e-02 -1.760357e-02
## color       -0.0098196717  5.035557e-02 -4.632943e-03
## hue          0.0975106606  9.755619e-01 -1.665508e-01
## OD           0.0284851062  1.163025e-02  4.419224e-02
## proline     -0.0002404522 -9.999951e-05  3.626701e-05
```

```
# Save the variable loadings to a .csv file
#write.csv(wine.pca.raw$rotation, file="wine_pca_rawloadings.csv")

# Biplot which shows the loadings in the first 2 principal components (raw data)
biplot(wine.pca.raw)
```

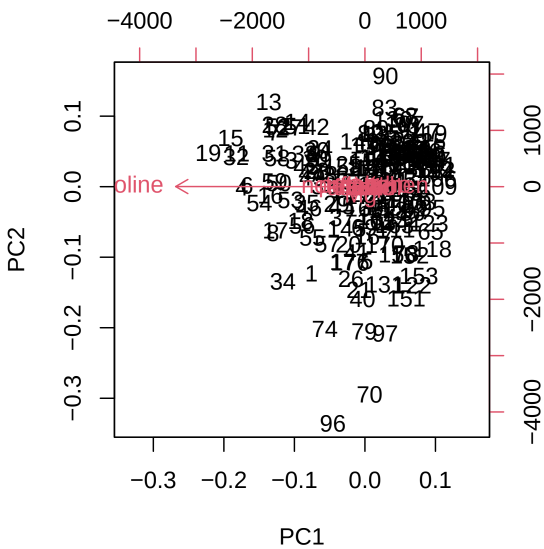


Figure 8.7: Biplot of the first two principal components of unstandardized (raw) data for comparison.