Predict Rain Capstone

Robert Clark

Introduction

In this project, I am using a weather dataset from Australia with the goal of seeing how accurately I can predict whether it will rain tomorrow. I have always thought weather is really interesting. I lived in an area in the USA called the Midwest, and in that area, very strong thunderstorms are a fairly common occurance. Eventually I grew to love them, and I always enjoyed listening to the rain and thunder as my family played games together inside. I grew to absolutely love watching the big Midwest thunderstorms roll in across the sky. This is what drew me to try to apply machine learning to this dataset.

This dataset is called Rain in Australia, downloaded from the Kaggle website, found at this URL: https://www.kaggle.com/jsphyg/weather-dataset-rattle-package (https://www.kaggle.com/jsphyg/weather-dataset-rattle-package)

Overview

The Rain in Australia dataset contains the following variables:
Date: date of recorded weather, considered “today”
Location: city in Australia
MinTemp: minimum temperature in Celcius
MaxTemp: maximum temperature in Celcius
Rainfall: amount of recorded rainfall for the day in mm (this includes all forms of precipitation)
Evaporation: Class A pan evaporation in mm
Sunshine: number of hours of sunshine in the day
WindGustDir: direction of the strongest wind gust that day
WindGustSpeed: speed of the strongest wind gust in km/h
WindDir9am: direction of wind recorded at 9am
WindDir3pm: direction of wind recorded at 3pm
WindSpeed9am: speed of wind recorded at 9am
WindSpeed3pm: speed of wind recorded at 3pm
Humidity9am: humidity level, as a percentage, recorded at 9am
Humidity3pm: humidity level, as a percentage, recorded at 3pm
Pressure9am: atmospheric pressure (hpa) recorded at 9am
Pressure3pm: atmospheric pressure (hpa) recorded at 3pm
Cloud9am: fraction of sky covered obscured by cloud, in “oktas” (scale from 0 to 8) recorded at 9am
Cloud3pm: fraction of sky covered obscured by cloud, in “oktas” (scale from 0 to 8) recorded at 3pm
Temp9am: temperature in Celcius recorded at 9am
Temp3pm: temperature in Celcius recorded at 3pm
RainToday: binary variable saying whether it rained today or not. If Rainfall > 1, then “Yes”, else “No”
RISK_MM: amount of recorded rain that fell tomorrow.
RainTomorrow: binary variable whether it rained tomorrow or not. If RISK_MM > 1, then “Yes”, else “No”

The **RainTomorrow** variable is the target variable that I will try to predict.

Because the target variable is binary, I thought that using the k-nearest neighbors would be a great place to start. Before I start though, I need to examine the data and start discovering what relationships might exist between the variables. It is likely that not all the variables will be helpful in predicting rain tomorrow. Some variables could even decrease the accuracy.

One of the challenges that I will face in this project is the prevlance. Because this data comes from Australia, which is mostly desert, there are a lot more days in the dataset with no rain tomorrow than there was rain tomorrow. This is going to make it hard to get an accurate model. This also means that the overall accuracy can be fairly deceiving. 22% of our data had rain the next day. This means that I could predict no rain tomorrow every day and have an overall accuracy of 78%. For this reason, I will be focusing on the model's specificty to really show how accurate my predictions are. I will go over specificity in more detail later in this presentation.

Let's go ahead and import the data and get the data ready to start analyzing.

Import Dataset

```
library(tidyverse)
```

```
## Registered S3 methods overwritten by 'ggplot2':
##   method         from
##   [.quosures     rlang
##   c.quosures     rlang
##   print.quosures rlang
```

```
## Registered S3 method overwritten by 'rvest':
##   method            from
##   read_xml.response xml2
```

```
## ── Attaching packages ─────────────────────────────────────────────────────
──────────── tidyverse 1.2.1 ──
```

```
## ✔ ggplot2 3.1.1     ✔ purrr   0.3.2
## ✔ tibble  2.1.1     ✔ dplyr   0.8.1
## ✔ tidyr   0.8.3     ✔ stringr 1.4.0
## ✔ readr   1.3.1     ✔ forcats 0.4.0
```

```
## ── Conflicts ────────────────────────────────────────────────────────────────
────── tidyverse_conflicts() ──
## ✖ dplyr::filter() masks stats::filter()
## ✖ dplyr::lag()    masks stats::lag()
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
library(mice)
```

```
##
## Attaching package: 'mice'
```

```
## The following object is masked from 'package:tidyr':
##
##     complete
```

```
## The following objects are masked from 'package:base':
##
##     cbind, rbind
```

```
library(VIM)
```

```
## Loading required package: colorspace
```

```
## Loading required package: grid
```

```
## Loading required package: data.table
```

```
##
## Attaching package: 'data.table'
```

```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

```
## The following object is masked from 'package:purrr':
##
##     transpose
```

```
## Registered S3 methods overwritten by 'car':
##   method                          from
##   influence.merMod                lme4
##   cooks.distance.influence.merMod lme4
##   dfbeta.influence.merMod         lme4
##   dfbetas.influence.merMod        lme4
```

```
## VIM is ready to use.
##  Since version 4.0.0 the GUI is in its own package VIMGUI.
##
##           Please use the package to use the new (and old) GUI.
```

```
## Suggestions and bug-reports can be submitted at: https://github.com/alexkowa/VIM/issu
es
```

```
##
## Attaching package: 'VIM'
```

```
## The following object is masked from 'package:datasets':
##
##     sleep
```

```
library(rpart)
library(randomForest)
```

```
## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.
```

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
library(Rborist)
```

```
## Rborist 0.1-17
```

```
## Type RboristNews() to see new features/changes/bug fixes.
```

```
# import data
weather <- read.csv("~/Documents/R Studio/EDX Courses/weatherAUS.csv")

# change class of date column from factor to date
weather <- weather %>% mutate(Date = as.Date(Date))

# examine data
head(weather)
```

```
##         Date Location MinTemp MaxTemp Rainfall Evaporation Sunshine
## 1 2008-12-01   Albury    13.4    22.9      0.6          NA       NA
## 2 2008-12-02   Albury     7.4    25.1      0.0          NA       NA
## 3 2008-12-03   Albury    12.9    25.7      0.0          NA       NA
## 4 2008-12-04   Albury     9.2    28.0      0.0          NA       NA
## 5 2008-12-05   Albury    17.5    32.3      1.0          NA       NA
## 6 2008-12-06   Albury    14.6    29.7      0.2          NA       NA
##   WindGustDir WindGustSpeed WindDir9am WindDir3pm WindSpeed9am
## 1           W            44          W        WNW           20
## 2         WNW            44        NNW        WSW            4
## 3         WSW            46          W        WSW           19
## 4          NE            24         SE          E           11
## 5           W            41        ENE         NW            7
## 6         WNW            56          W          W           19
##   WindSpeed3pm Humidity9am Humidity3pm Pressure9am Pressure3pm Cloud9am
## 1           24          71          22      1007.7      1007.1        8
## 2           22          44          25      1010.6      1007.8       NA
## 3           26          38          30      1007.6      1008.7       NA
## 4            9          45          16      1017.6      1012.8       NA
## 5           20          82          33      1010.8      1006.0        7
## 6           24          55          23      1009.2      1005.4       NA
##   Cloud3pm Temp9am Temp3pm RainToday RISK_MM RainTomorrow
## 1       NA    16.9    21.8        No     0.0           No
## 2       NA    17.2    24.3        No     0.0           No
## 3        2    21.0    23.2        No     0.0           No
## 4       NA    18.1    26.5        No     1.0           No
## 5        8    17.8    29.7        No     0.2           No
## 6       NA    20.6    28.9        No     0.0           No
```

Just from looking at the first 6 rows, it looks like we have some variables that have a lot of missing data. Before I start exploring the data, I want to see how messy our dataset is.

```
# How many variables do we have that have missing values (NA's)?
colnames(weather)[colSums(is.na(weather)) > 0]
```

```
##  [1] "MinTemp"       "MaxTemp"       "Rainfall"      "Evaporation"
##  [5] "Sunshine"      "WindGustDir"   "WindGustSpeed" "WindDir9am"
##  [9] "WindDir3pm"    "WindSpeed9am"  "WindSpeed3pm"  "Humidity9am"
## [13] "Humidity3pm"   "Pressure9am"   "Pressure3pm"   "Cloud9am"
## [17] "Cloud3pm"      "Temp9am"       "Temp3pm"       "RainToday"
```

```
colnames(weather)[colSums(is.na(weather)) == 0]
```

```
## [1] "Date"         "Location"     "RISK_MM"      "RainTomorrow"
```

I have a lot of variables with missing data. I think for now I will just ignore the missing values and continue on exploring the data. As I explore the data, I will just filter out the missing values to see what trends exist now. I will address the missing data and properly fix it before I begin modeling.

Data Exploration

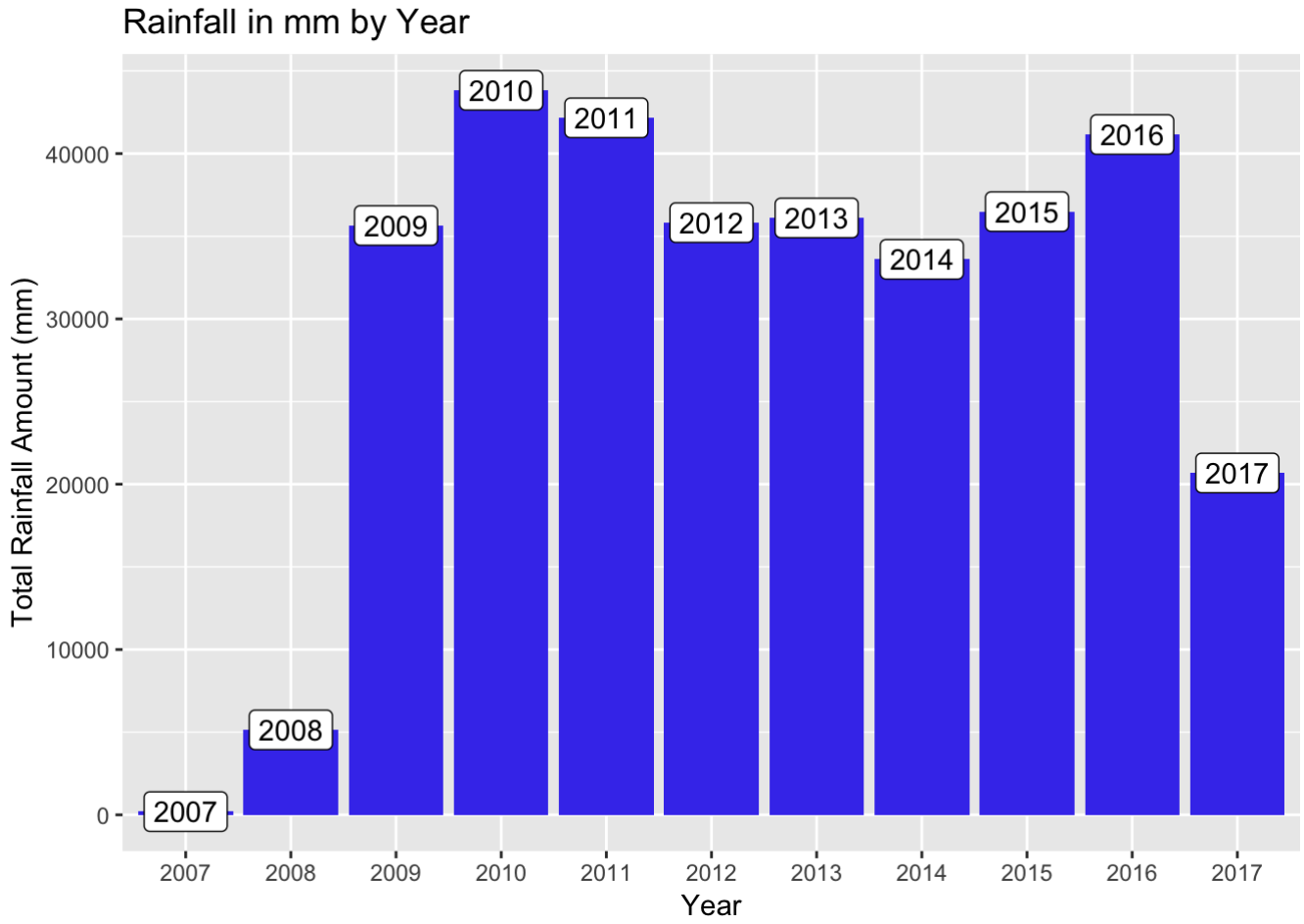
Let's start exploring the data. There are a lot of variables in this dataset, and I want to see how they interact with the RainTomorrow variable. The interesting thing with this dataset is that for some of the variables, such as pressure and humidity, there are measures from the morning and the afternoon. This will allow me to look at how this data changes during the day and see if there are any relationships there with the RainTomorrow variable.

The first thing I will look at is the rainfall. What are some of the trends with rainfall in Australia? Are some locations more rainy than others? Does is rain more during one part of the year than the rest? I will answer these questions, and others, in this next section.

```
# What percent of the days in our dataset were rainy days?
(weather %>% filter(RainTomorrow == "Yes") %>% nrow()) / (weather %>% nrow())
```

```
## [1] 0.2241812
```

```
# view how much rainfall was recorded in each year? To make further calculations easier,
I will add "year" as a column
weather <- weather %>%
  mutate(year = format(Date, "%Y"))
weather %>%
  group_by(year) %>%
  filter(!is.na(Rainfall)) %>%
  summarize(rainfall = sum(Rainfall)) %>%
  ggplot(aes(year, rainfall)) +
  geom_bar(stat = "identity", fill = "#4542EC") +
  geom_label(aes(year, rainfall, label = year)) +
  xlab("Year") +
  ylab("Total Rainfall Amount (mm)") +
  ggtitle("Rainfall in mm by Year")
```

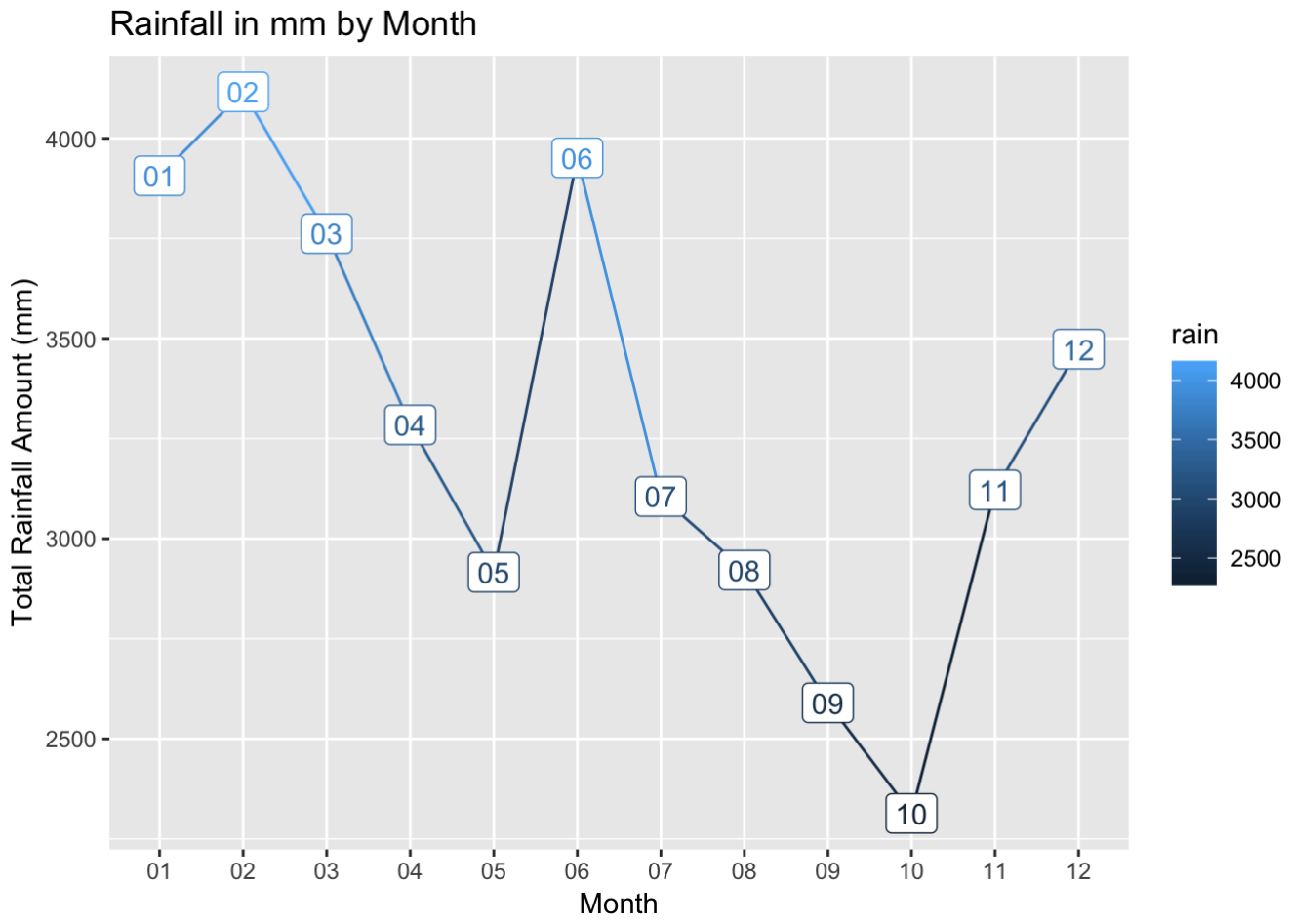



```
# how many days of rain did we get per year?
weather %>% filter(Rainfall > 0) %>% group_by(year) %>% summarize(days_with_rain = n())
```

```
## # A tibble: 11 x 2
##    year  days_with_rain
##    <chr>          <int>
##  1 2007              28
##  2 2008             814
##  3 2009            5687
##  4 2010            6259
##  5 2011            5858
##  6 2012            5423
##  7 2013            5509
##  8 2014            5740
##  9 2015            5900
## 10 2016            6454
## 11 2017            2840
```

It looks like we have drastically less days of rain in 20017, 2008 and 2017. We probably have incomplete data for those years. Besides the years with incomplete data, there isn't a lot of variation between the amount of rainfall in those years. Let's look at rainfall by month to see if there are rainy seasons that might help predicting rain tomorrow. Since 2007, 2008 and 2017 have incomplete data (not a full year's worth of data), I will exclude them from this exploration, since I really want to see how the rain changes throughout the whole year.

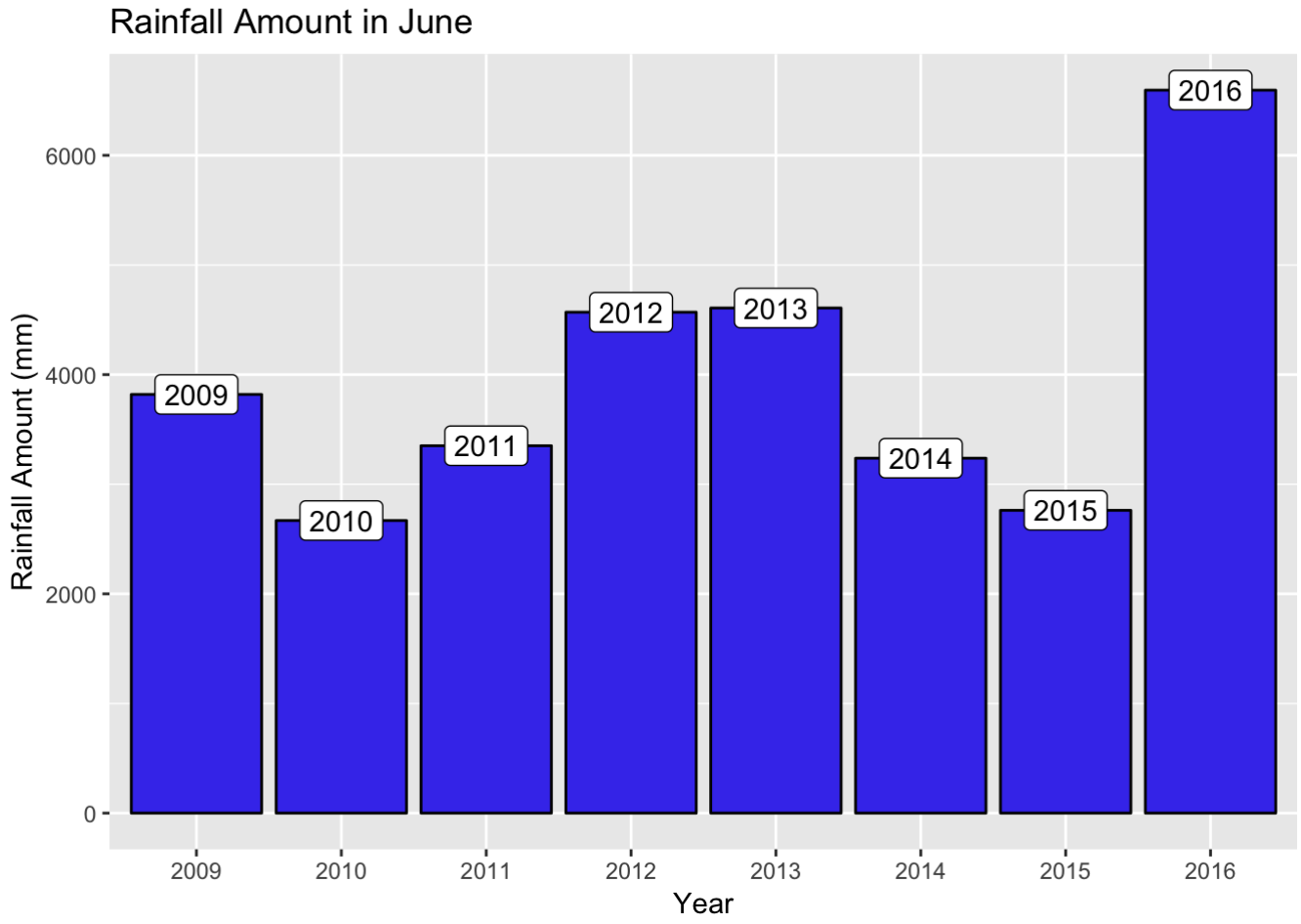
```
# Like year, I'll add month as a column for easier future calculations.
# Get the month, the year, and the amount of rainfall
years <- c(2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016)
monthly_rainfall <- weather %>%
  filter(year %in% years, !is.na(Rainfall)) %>%
  mutate(month = format(Date, "%m")) %>%
  group_by(month, year) %>%
  summarize(rainfall = sum(Rainfall))

# Create plot
monthly_rainfall %>%
  group_by(month) %>%
  summarize(rain = mean(rainfall)) %>%
  ggplot(aes(month, rain, group = 1)) +
  geom_line(aes(month, rain, color = rain)) +
  geom_point(aes(month, rain, color = rain)) +
  geom_label(aes(month, rain, label = month, color = rain)) +
  xlab("Month") +
  ylab("Total Rainfall Amount (mm)") +
  ggtitle("Rainfall in mm by Month")
```



From the graph, we can see that the rainy season tends to run from about December to about March, with February being the wettest. It looks like the month could be a good predictor variable. If we are in February, the chances it will rain tomorrow are much higher than October.

I'm also not sure why June is so wet. Do they get monsoons or big storms during that time, generally? Or mabye one year had an extra wet June for some reason, and it is throwing that month off it's normal mean a bit. Let's explore that month to see if one year stands out more than others.

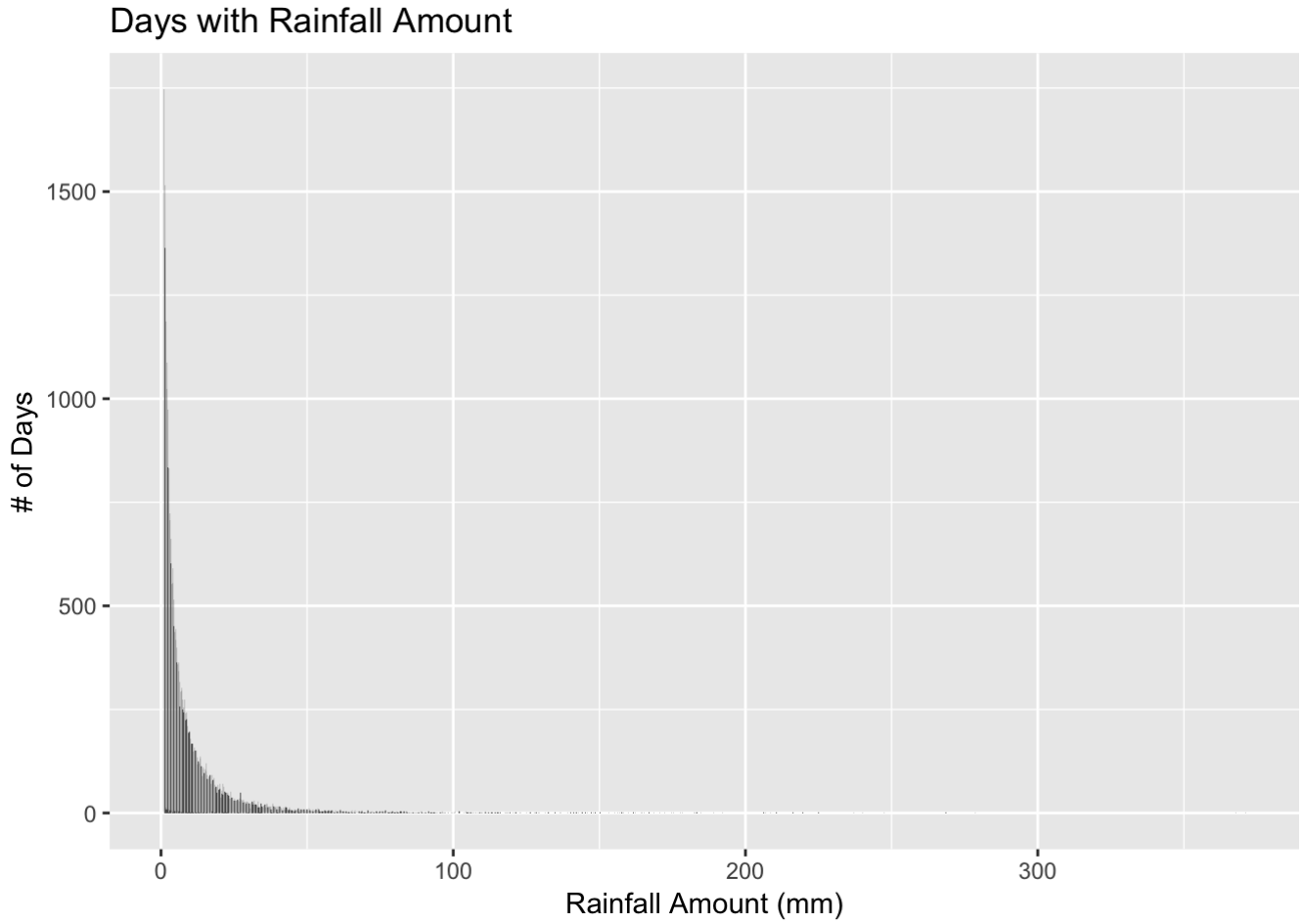
```
# Look at the month of June for each year
monthly_rainfall %>%
  filter(month == "06") %>%
  ggplot(aes(year, rainfall)) +
  geom_bar(stat="identity", color = "black", fill = "#4542EC") +
  geom_label(aes(year, rainfall, label = year)) +
  xlab("Year") +
  ylab("Rainfall Amount (mm)") +
  ggtitle("Rainfall Amount in June")
```



2016 was definitely the wettest, but all the other months are fairly wet too, compared to the other non-rainy season months. Let's look at some other points about rainfall, such as when it rains, how much does it commonly rain? If it rains one day, does it tend to rain the next day? Do really rainy days mean that it it likely to rain again tomorrow?

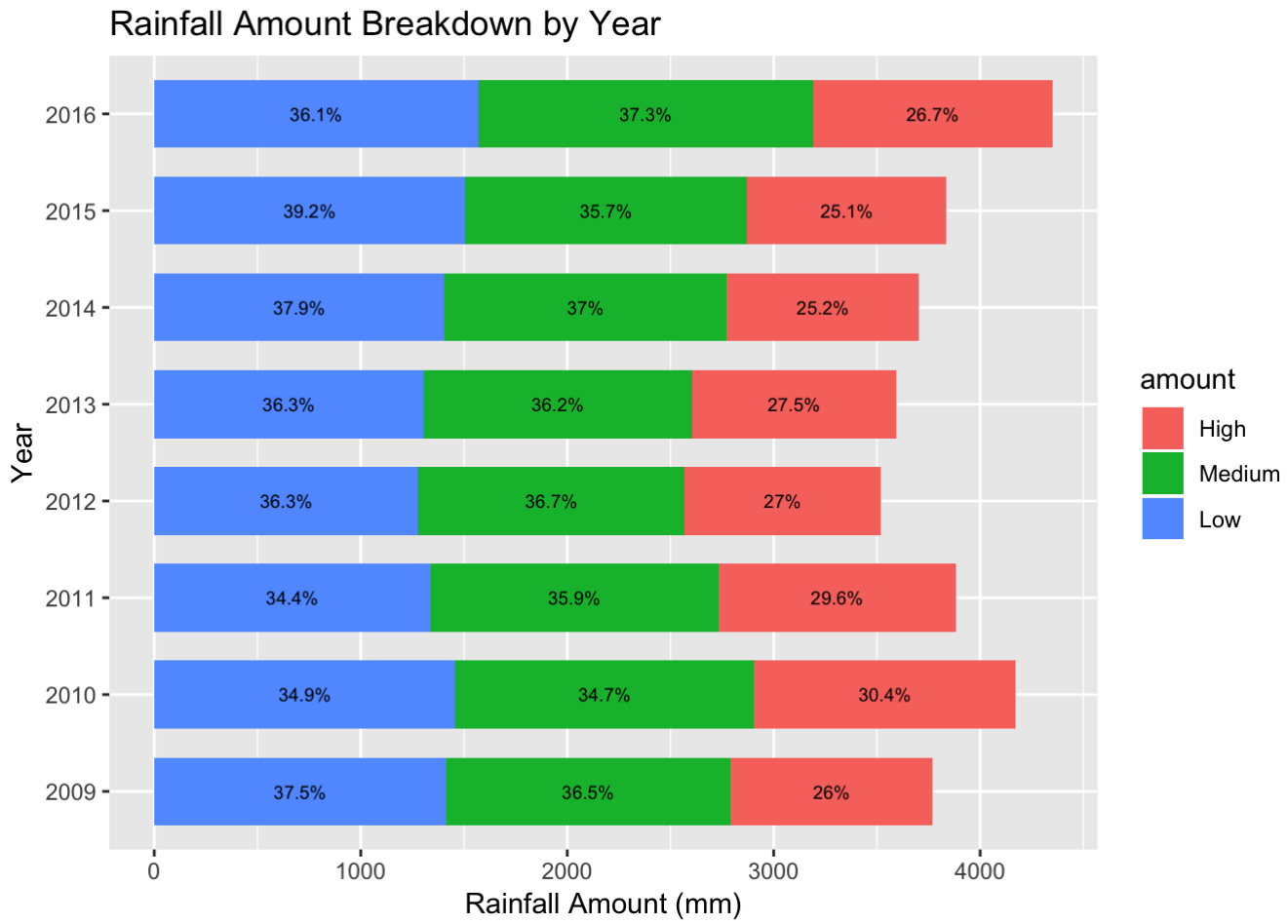
```
# Let's look at distribution of rainy days
weather %>%
  filter(Rainfall >= 1.0) %>%
  group_by(Rainfall) %>%
  summarize(count = n()) %>%
  ggplot(aes(Rainfall, count)) +
  geom_histogram(stat = "identity") +
  xlab("Rainfall Amount (mm)") +
  ylab("# of Days") +
  ggtitle("Days with Rainfall Amount")
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



It is easy to see that most of the data has rainfall amounts less than 50mm, but with just the large spread of data, it is hard to see anything else. I'm going to group the rainfall amounts into low rain, medium rain, and high rain, and count the days for each category. At the same time, I'll break it out by year to get a better picture of the distribution of the categories.

```
# Categorize rainfall amounts as low, medium, and high, and then show the number of days
for each category and year
# Store categories into separate dataset
rainfall_categorized <- weather %>%
  filter(Rainfall >= 1.0, year %in% years) %>%
  # values for category cutoffs were simply chosen after examination of data distributio
n
  mutate(amount = ifelse(Rainfall <= 3, "Low", ifelse(Rainfall <= 10, "Medium", "High"
))) %>%
  group_by(year, amount) %>%
  summarize(count = n())

# I want the graph to sort the categories from low to high, which requires factoring and
arranging the categories
rainfall_categorized$amount <- factor(rainfall_categorized$amount, levels = c("High", "M
edium", "Low"))
rainfall_categorized <- arrange(rainfall_categorized, amount)

# Create horizontal bar chart. Each year gets a bar, each bar is divided up by rain amou
nt category. Each category is labeled with the percentage that category takes up in each
year.
rainfall_categorized %>%
  group_by(year) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(year, count, fill = amount)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip()  +
  xlab("Year") +
  ylab("Rainfall Amount (mm)") +
  ggtitle("Rainfall Amount Breakdown by Year")
```



This graph just shows us the breakdown of rainfall in a single year. Now what we want to know is how today's rainfall affects whether it rains tomorrow or not. Does it tend to rain tomorrow if it rained today?

```
# Does it tend to rain tomorrow if it rained today?
# First, let's look at the number of days in our dataset where it rained tomorrow compar
ed to not raining tomorrow overall
weather %>%
  group_by(RainTomorrow) %>%
  summarize(count = n())
```

```
## # A tibble: 2 x 2
##   RainTomorrow  count
##   <fct>         <int>
## 1 No           110316
## 2 Yes           31877
```

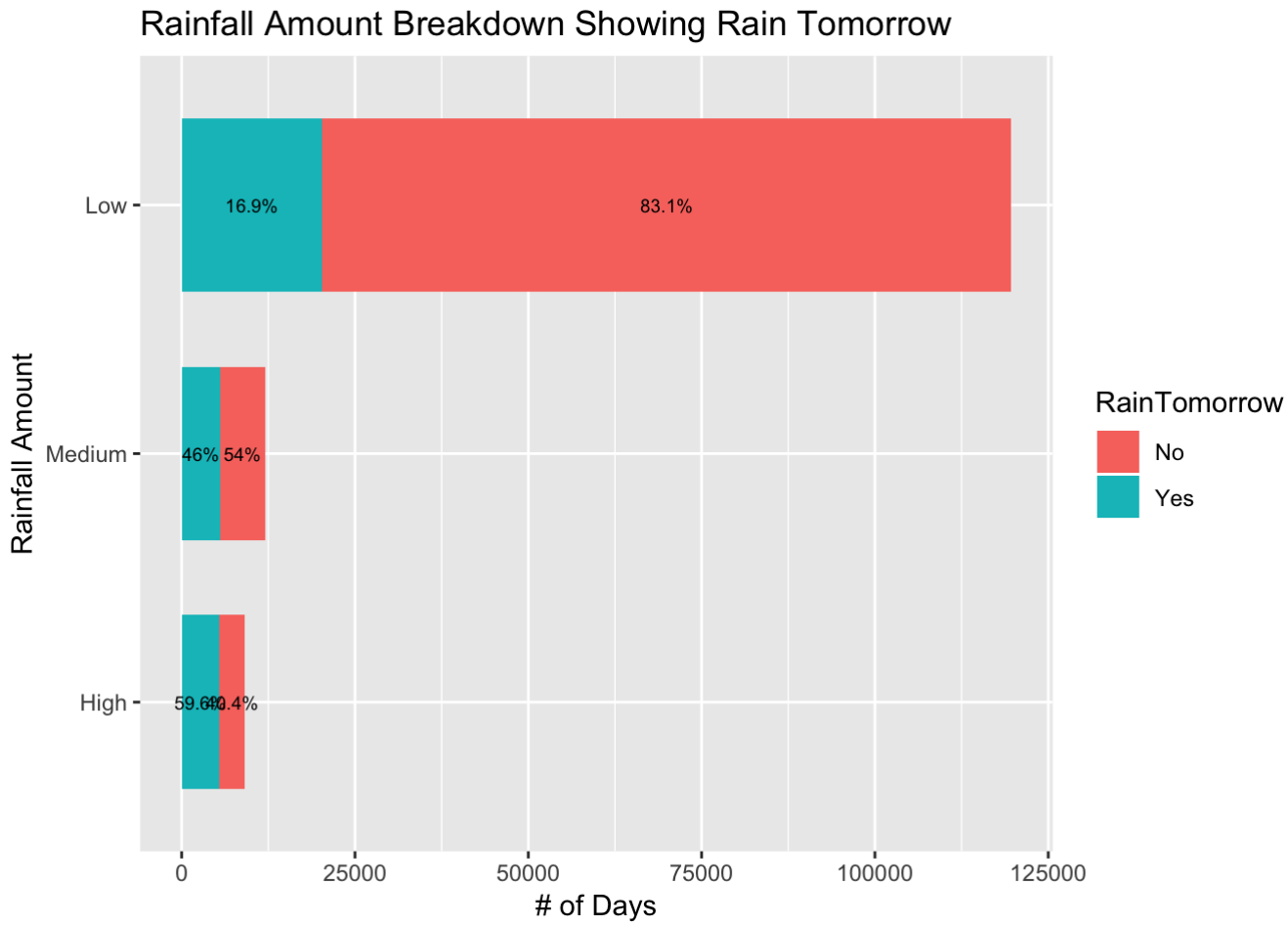
```
# Now let's look at that same data, but only show days where it rained today
weather %>%
  filter(RainToday == "Yes") %>%
  group_by(RainTomorrow) %>%
  summarize(count = n())
```

```
## # A tibble: 2 x 2
##   RainTomorrow count
##   <fct>        <int>
## 1 No           16858
## 2 Yes          14597
```

When we filter the data to only rows where it rained today, we see that the proportion of days where it rained tomorrow compared to no rain tomorrow is a lot more even than compared to the unfiltered data. So this shows us that there is a little bit of a relationship where if it rains today, the chances of it raining tomorrow are a lot higher than if it didn't rain today. If there was no relationship there, the proportion would be a lot closer to the overall proportion.

Since we have the rain categorized as low, medium, and high, let's look at whether days of high rain tend to rain tomorrow as well compared to days of low rain.

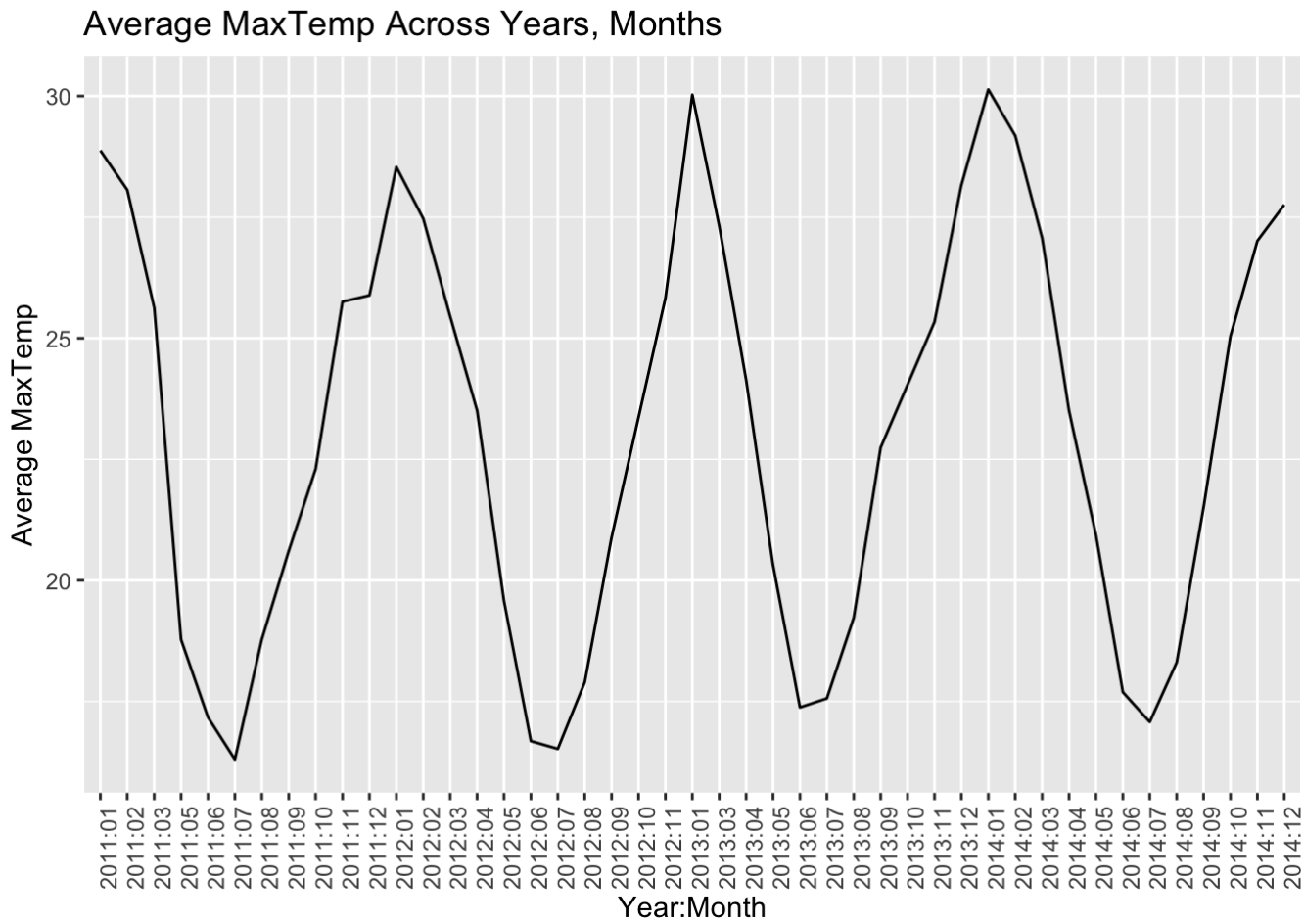
```
# Group into low, medium, high again with same definition
rainfall_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Rainfall)) %>%
  mutate(Rainfall = ifelse(Rainfall <= 3, "Low", ifelse(Rainfall <= 10, "Medium", "High"
))) %>%
  group_by(Rainfall, RainTomorrow) %>%
  summarize(count = n())

# Factor again for proper ordering in barchart
rainfall_categorized$Rainfall <- factor(rainfall_categorized$Rainfall, levels = c("High"
, "Medium", "Low"))
rainfall_categorized <- arrange(rainfall_categorized, Rainfall)

# Create bar chart
rainfall_categorized %>%
  group_by(Rainfall) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Rainfall, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip()  +
  xlab("Rainfall Amount") +
  ylab("# of Days") +
  ggtitle("Rainfall Amount Breakdown Showing Rain Tomorrow")
```



There certainly is a relationship here, where when it rains a lot, it tends to rain the next day too. There are other factors that could be at play here too, like it tends to rain more in the rainy season. But there is definitely a relationship here. It could be hard to get a lot of value out of this relationship though due to the fact that there is so much more data with low rainfall amounts compared to high rainfall amounts, and the high rainfall amounts only have a slightly better than 50% chance of raining tomorrow.

Let's start looking at some of the other variables in the dataset and their relationship with RainTomorrow. I will first look at MaxTemp.

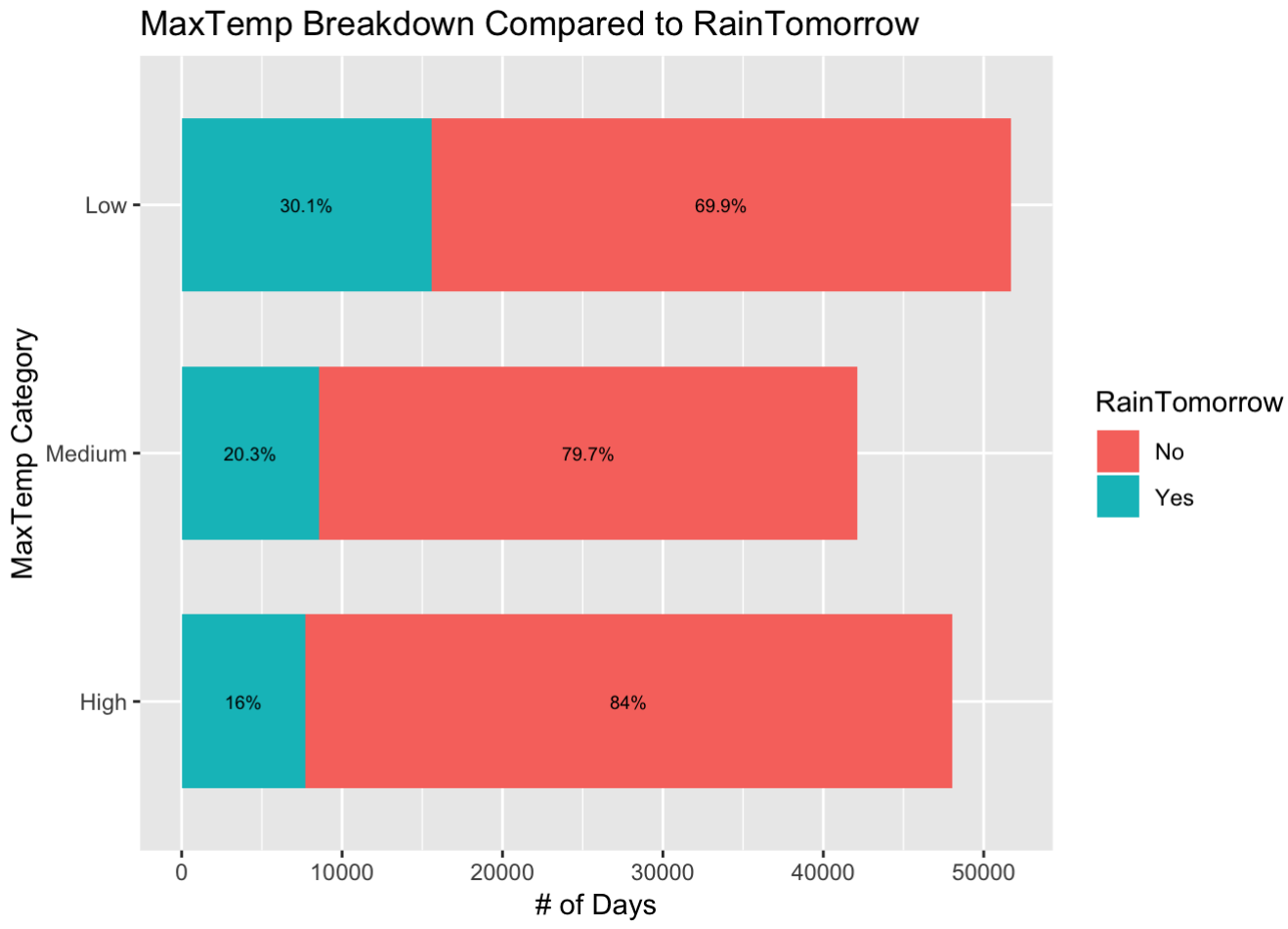
```
# This shows the average max temp for each month over a few years
weather %>%
  mutate(MaxTemp = ifelse(is.na(MaxTemp), 0, MaxTemp)) %>%
  filter(year %in% c(2011, 2012, 2013, 2014)) %>%
  mutate(month = format(Date, "%m")) %>%
  mutate(full_month = paste(year, month, sep = ":")) %>%
  group_by(full_month) %>%
  summarize(mean_max = mean(MaxTemp)) %>%
  ggplot(aes(full_month, mean_max, group = 1)) +
  geom_line() +
  theme(axis.text.x = element_text(angle = 90, hjust = 1))  +
  xlab("Year:Month") +
  ylab("Average MaxTemp") +
  ggtitle("Average MaxTemp Across Years, Months")
```



We can see our max temp range from about 17 degrees Celcius during the winters to 28, 29 in the summer. We already have seen that the rainiest months are during the summer, from December to March, and the winter, in June, July. So the months with the highest and lowest max temps are also the wettest months. Since it appears that MaxTemp follows really close with the month with how much rain falls during that month, we probably only need month or MaxTemp in the final model instead of both.

Let's look at MaxTemp more, especially its relationship to RainTomorrow. I am going to do the same thing with MaxTemp that I did for Rainfall, which is categorize the variable into low, medium and high. I won't be categorizing this for the final model because it isn't necessary, but categorizing the variables like this now just helps me to visualize and understand the relationship between the variables better than just looking at numbers.

```
# filter data, get count of days, categorize variable
rain_temp <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(MaxTemp)) %>%
  mutate(MaxTemp = ifelse(MaxTemp <= 20, "Low", ifelse(MaxTemp <= 26, "Medium", "High"
))) %>%
  group_by(MaxTemp, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange for correct visualization
rain_temp$MaxTemp <- factor(rain_temp$MaxTemp, levels = c("High", "Medium", "Low"))
rain_temp <- arrange(rain_temp, MaxTemp)

# build visualization
rain_temp %>%
  group_by(MaxTemp) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(MaxTemp, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("MaxTemp Category") +
  ylab("# of Days") +
  ggtitle("MaxTemp Breakdown Compared to RainTomorrow")
```



When we look at days where it rained tomorrow, there doesn't appear to be that significant change in high temperatures and low temperatures. This probably won't be that useful of a variable to include in the final model.

Before moving on, let's see if there is any relationship between days that have a big difference in MaxTemp and MinTemp. Do days that have more temperature fluctuation tend to rain tomorrow? To see this, I will find the difference between the MinTemp and the MaxTemp, and then take the average of these differences for days where it rained tomorrow and days where it didn't rain tomorrow.

```
weather %>%
  filter(year %in% years, !is.na(RainTomorrow), !is.na(MinTemp), !is.na(MaxTemp)) %>%
  mutate(temp_change = MaxTemp - MinTemp) %>%
  group_by(RainTomorrow) %>%
  summarize(average_change = mean(temp_change))
```

```
## # A tibble: 2 x 2
##   RainTomorrow average_change
##   <fct>                 <dbl>
## 1 No                     11.9
## 2 Yes                     7.92
```

There is a little bit of a difference, but I don't think it is anything really worth keeping in our model. Let's move on to the next variable and look at Pressure.

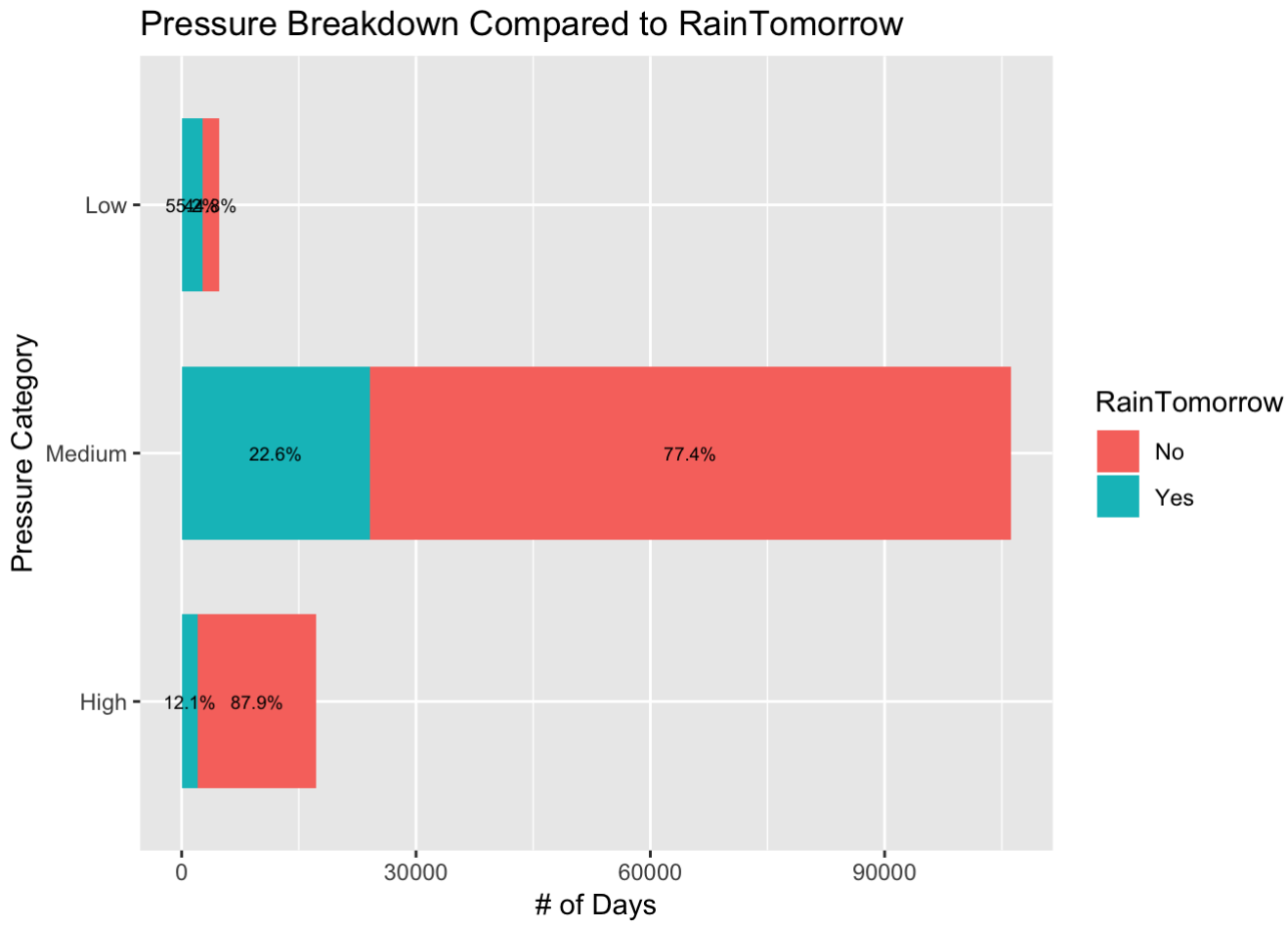
This time I am going to start with looking at the change in pressure from 9am to 3pm. Does a bigger change in pressure mean it is going to rain tomorrow?

```
# filter out the NA's and get the average pressure change for rain tomorrow and no rain
 tomorrow
weather %>%
  filter(year %in% years, !is.na(Pressure9am), !is.na(Pressure3pm), !is.na(RainTomorro
w)) %>%
  mutate(pressure_change = Pressure3pm - Pressure9am) %>%
  group_by(RainTomorrow) %>%
  summarize(average_pressure_change = mean(pressure_change))
```

```
## # A tibble: 2 x 2
##   RainTomorrow average_pressure_change
##   <fct>                          <dbl>
## 1 No                             -2.48
## 2 Yes                            -2.10
```

There is not nearly as much of a difference there as I expected. Let's just look at high and low pressure and see if one or the other tends to mean rain tomorrow.

```
# categorize the pressure
pressure_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Pressure3pm)) %>%
  mutate(Pressure3pm = ifelse(Pressure3pm <= 1003, "Low", ifelse(Pressure3pm <= 1023, "M
edium", "High"))) %>%
  group_by(Pressure3pm, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange for visualization
pressure_categorized$Pressure3pm <- factor(pressure_categorized$Pressure3pm, levels = c(
"High", "Medium", "Low"))
pressure_categorized <- arrange(pressure_categorized, Pressure3pm)

# build visualization
pressure_categorized %>%
  group_by(Pressure3pm) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Pressure3pm, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Pressure Category") +
  ylab("# of Days") +
  ggtitle("Pressure Breakdown Compared to RainTomorrow")
```

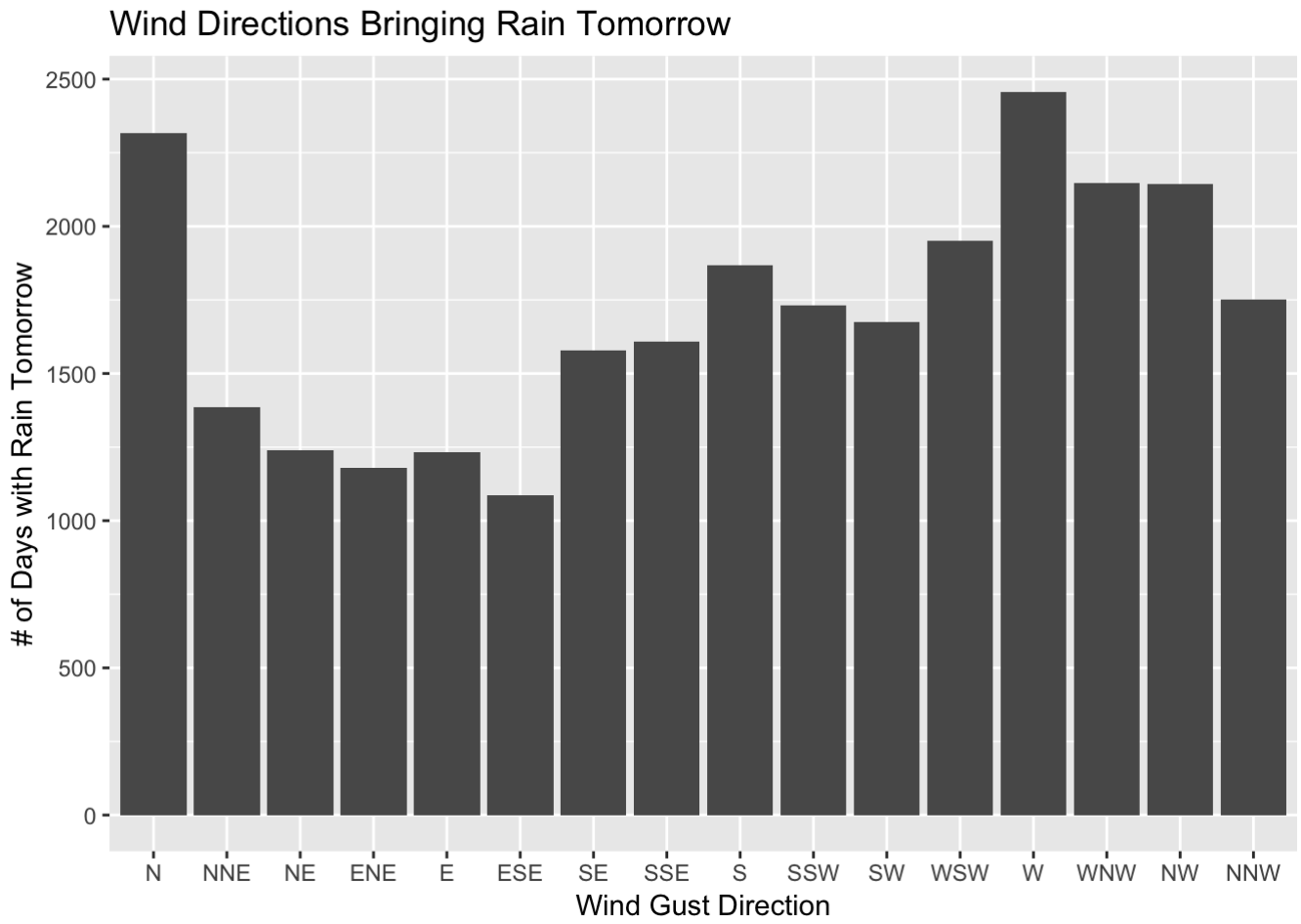


There definitely is a change in the percentage, where low pressure days lead to rain tomorrow more often than high pressure days. Probably worth keeping in the model, but I still don't know how much it will really add to the accuracy because again the percentage is only about about over 50%.

Let's go ahead and move on to the next variable, WindGustDir. We have wind direction data from 9am and 3pm, but I think I will only look at the wind gust direction. The wind gust will be the direction the wind was going in when it was blowng the hardest, which is probably the best indicator of where the wind really was coming from.

I will look at each wind diretion and see the number of days that it rained tomorrow when the wind was blowing in that direction.

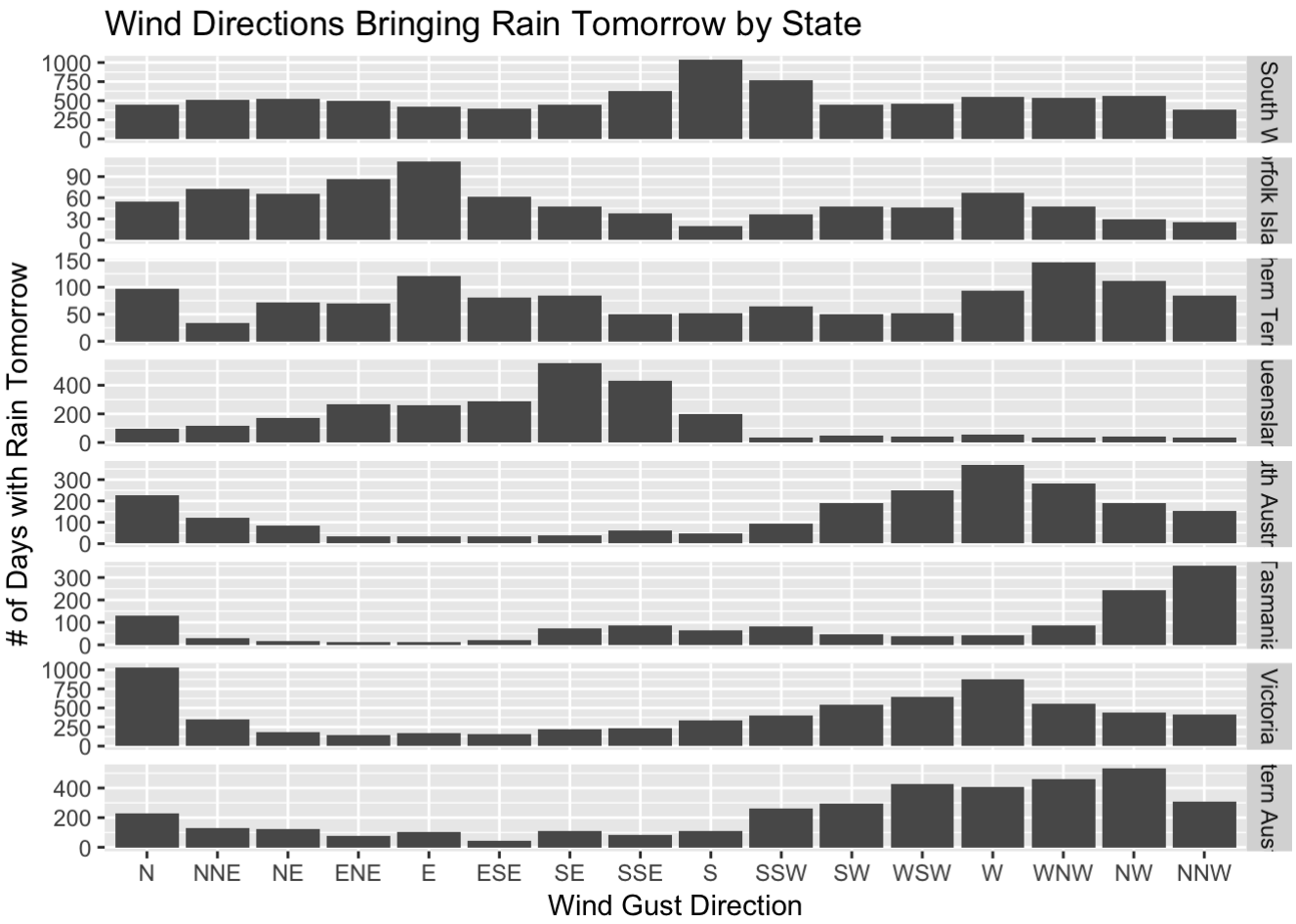
```
# store weather in another variable
wind_weather <- weather

# factor and arrange direction
wind_weather$WindGustDir <- factor(weather$WindGustDir, levels = c("N", "NNE", "NE", "EN
E", "E", "ESE", "SE", "SSE", "S", "SSW", "SW", "WSW", "W", "WNW", "NW", "NNW"))
wind_weather <- arrange(wind_weather, WindGustDir)

# view number of days wind blew from direction with rain tomorrow
wind_weather %>%
  filter(year %in% years, !is.na(WindGustDir), RainTomorrow == "Yes") %>%
  group_by(WindGustDir) %>%
  summarize(days = n()) %>%
  ggplot(aes(WindGustDir, days)) + geom_bar(stat = "identity") +
  xlab("Wind Gust Direction") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Wind Directions Bringing Rain Tomorrow")
```



It does look like West and North are the directions that wind brings rain tomorrow the most. One thing I need to consider when looking at this graph is the location. We have cities from all over Australia, and it is very possible that different cities will have wind blow in rain from different locations. We could also have a big difference in the number of cities in a similar location vs another.

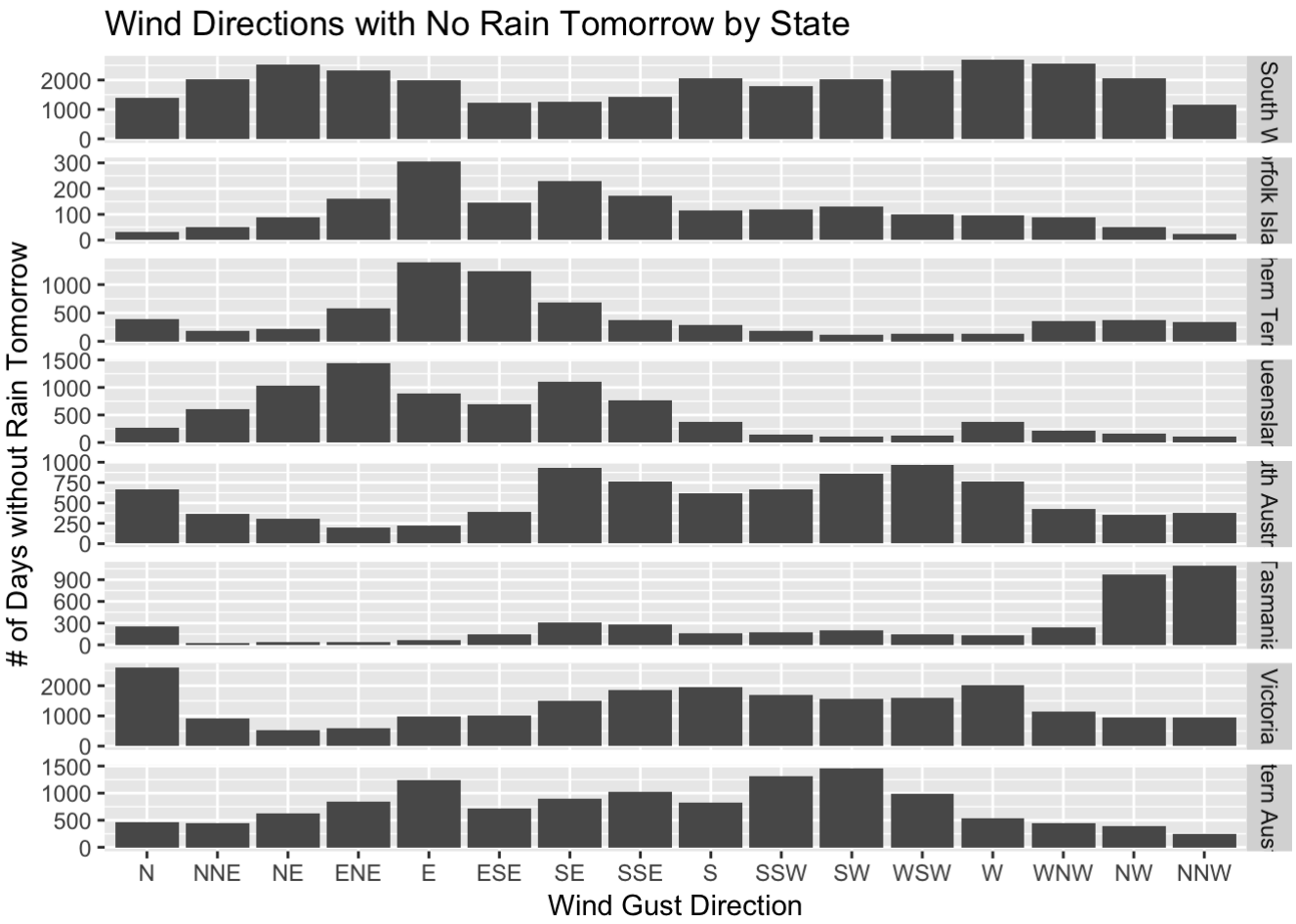
To see this better, let's view the same graph, only broken out by location. There are about 40 different cities in the dataset, which would be too many graphs so I am going to create a new variable called State, and assign each location the state in Australia that they belong in. There are about 7 states in Australia, so that would make it much easier to see the wind direction broken out by state.

```
# create States variable
states_abbv <- c("NSW", "NI", "NT", "Q", "SA", "T", "V", "WA")
states <- c("New South Wales", "Norfolk Island", "Northern Territory", "Queensland", "So
uth Australia", "Tasmania", "Victoria", "Western Australia")

# rename "Location" to "City" and assign each city their appropriate state
state_weather <- wind_weather %>%
  mutate(City = Location) %>%
  mutate(State =
           ifelse(Location %in% c("Albury", "BadgerysCreek", "Canberra", "Cobar", "Coffs
Harbour", "Moree", "MountGinini", "Newcastle", "NorahHead", "Penrith", "Sydney", "Sydney
Airport", "Tuggeranong", "WaggaWagga", "Williamtown", "Wollongong"), states[1],
                  ifelse(Location  %in% c("NorfolkIsland"), states[2],
                         ifelse(Location %in% c("AliceSprings", "Darwin", "Katherine",
"Uluru"), states[3], ifelse(Location %in% c("Brisbane", "Cairns", "GoldCoast", "Townsvil
le"), states[4],

ifelse(Location %in% c("Adelaide", "MountGambier", "Nuriootpa", "Woomera"), states[5],

ifelse(Location %in% c("Hobart", "Launceston"), states[6],

ifelse(Location %in% c("Ballarat", "Bendigo", "Dartmoor", "Melbourne", "MelbourneAirpor
t", "Mildura", "Nhil", "Portland", "Richmond", "Sale", "Watsonia"), states[7],

ifelse(Location %in% c("Albany", "PearceRAAF", "Perth", "PerthAirport", "SalmonGums", "W
alpole", "Witchcliffe"), states[8], "*******ERROR********")))))))) %>%
  select(-Location)

# create same wind direction graph, but create individual graphs for each state
# let the y-axis change based on amount of data
state_weather %>%
  filter(year %in% years, !is.na(WindGustDir), RainTomorrow == "Yes") %>%
  group_by(State, WindGustDir) %>%
  summarize(days = n()) %>%
  ggplot(aes(WindGustDir, days)) +
  geom_bar(stat = "identity") +
  facet_grid(State~., scales = "free_y", ) +
  xlab("Wind Gust Direction") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Wind Directions Bringing Rain Tomorrow by State")
```



This graph is much more helpful. We see the different winds that bring rain tomorrow for the different states. South Australia tends to have western winds bring rain. Queensland has south eastern winds bring rain. This graph shows the number of days where it rained tomorrow. What does the graphs look like without rain tomorrow? Is it different? Maybe the wind tends to blow in the same direction whether it rains tomorrow or not. Let's look at the same chart but with RainTomorrow = No. If the charts are the same, that will tell us that wind doesn't tell us much in regard to rain tomorrow.

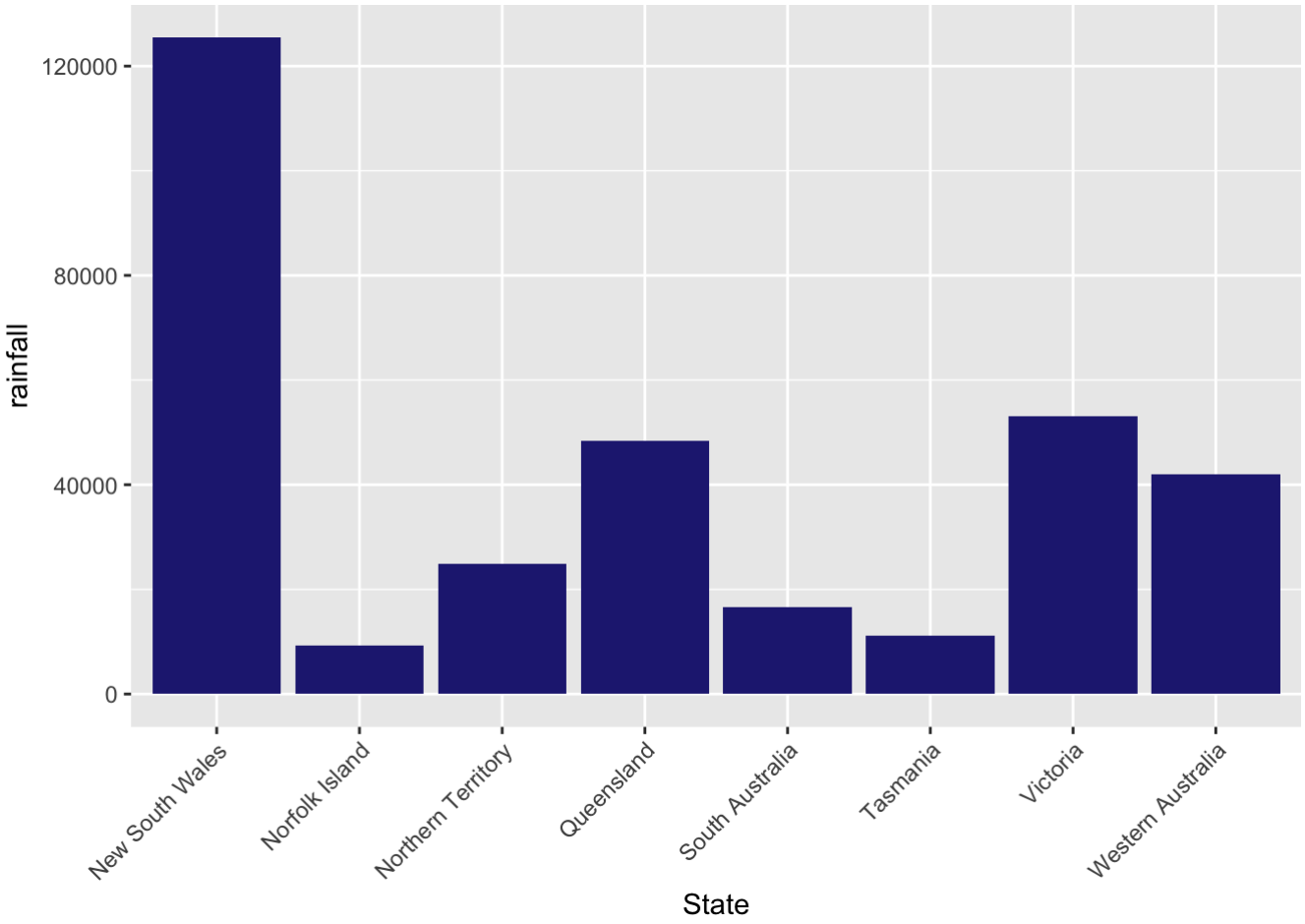
```
# create same chart, but for RainTomorrow = No
state_weather %>%
  filter(year %in% years, !is.na(WindGustDir), RainTomorrow == "No") %>%
  group_by(State, WindGustDir) %>%
  summarize(days = n()) %>%
  ggplot(aes(WindGustDir, days)) +
  geom_bar(stat = "identity") +
  facet_grid(State~., scales = "free_y", ) +
  xlab("Wind Gust Direction") +
  ylab("# of Days without Rain Tomorrow") +
  ggtitle("Wind Directions with No Rain Tomorrow by State")
```



Yes, those wind directions look fairly different than the other graph that has rain tomorrow, so this could be a good variable to include in the final model.

I also want to look at just the location. Do some cities/states get more rain than others?

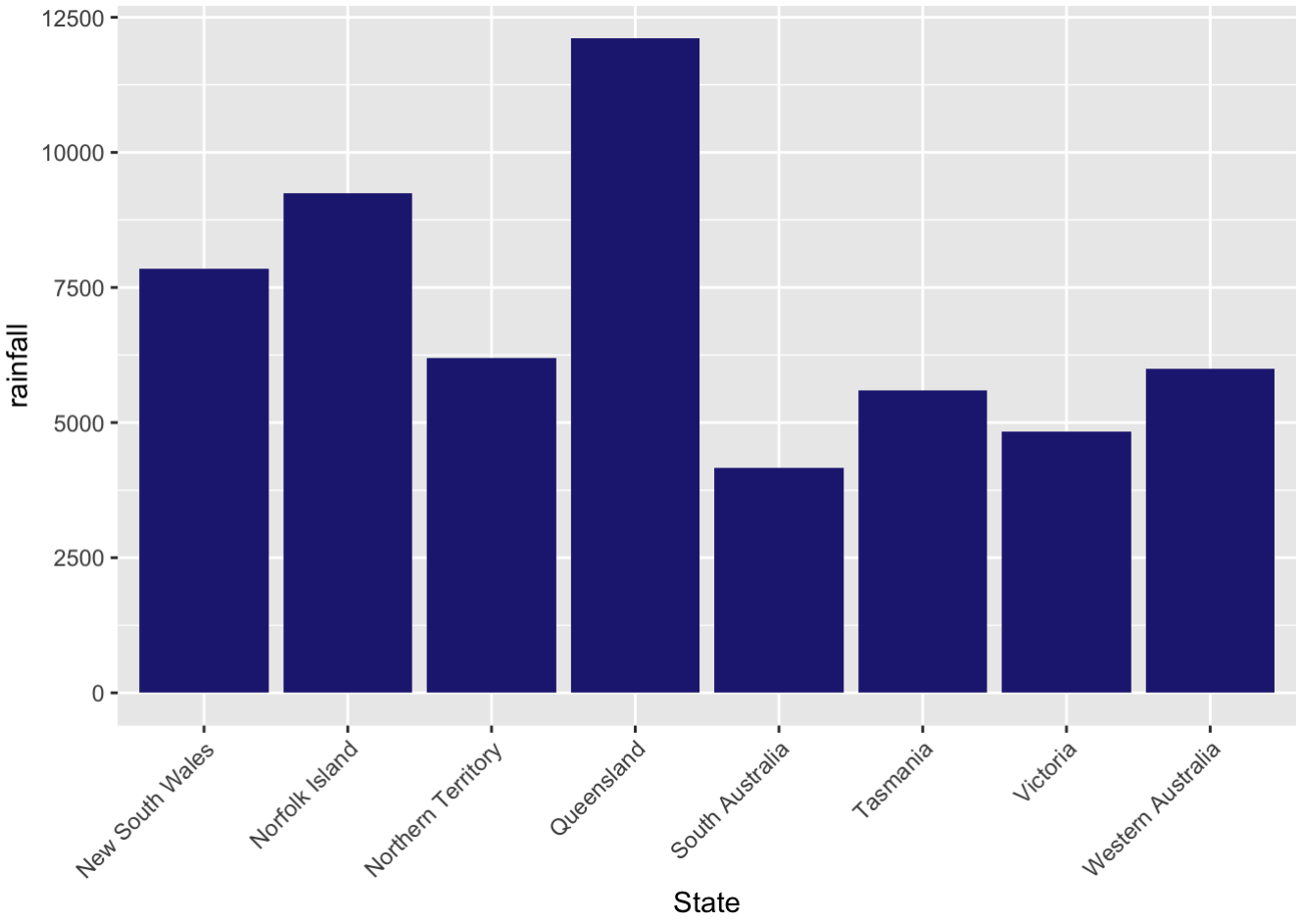
```
states_abbv <- c("NSW", "NI", "NT", "Q", "SA", "T", "V", "WA")
states <- c("New South Wales", "Norfolk Island", "Northern Territory", "Queensland", "So
uth Australia", "Tasmania", "Victoria", "Western Australia")
state_weather <- weather %>%
  mutate(City = Location) %>%
  mutate(State = ifelse(Location %in% c("Albury", "BadgerysCreek", "Canberra", "Cobar",
"CoffsHarbour", "Moree", "MountGinini", "Newcastle", "NorahHead", "Penrith", "Sydney",
"SydneyAirport", "Tuggeranong", "WaggaWagga", "Williamtown", "Wollongong"), states[1], i
felse(Location  %in% c("NorfolkIsland"), states[2], ifelse(Location %in% c("AliceSpring
s", "Darwin", "Katherine", "Uluru"), states[3], ifelse(Location %in% c("Brisbane", "Cair
ns", "GoldCoast", "Townsville"), states[4], ifelse(Location %in% c("Adelaide", "MountGam
bier", "Nuriootpa", "Woomera"), states[5], ifelse(Location %in% c("Hobart", "Launceston"
), states[6], ifelse(Location %in% c("Ballarat", "Bendigo", "Dartmoor", "Melbourne", "Me
lbourneAirport", "Mildura", "Nhil", "Portland", "Richmond", "Sale", "Watsonia"), states[
7], ifelse(Location %in% c("Albany", "PearceRAAF", "Perth", "PerthAirport", "SalmonGums"
, "Walpole", "Witchcliffe"), states[8], "*******ERROR********"))))))))) %>%
  select(-Location)
state_weather %>%
  filter(!is.na(Rainfall)) %>%
  group_by(State) %>%
  summarize(rainfall = sum(Rainfall)) %>%
  ggplot(aes(State, rainfall)) +
  geom_bar(stat = "identity", fill = "#252580") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```



From this graph, you would think that New South Wales is the rainiest state. This could also just mean that we have more locations in New South Wales than any other state, thus raising the rainfall amount a lot. So we need to normalize this by the number of cities we have in each state.

```
# put in number of cities for each state
city_count <- data.frame(State = states, count = ifelse(states == "New South Wales", 16,
ifelse(states == "Norfolk Island", 1, ifelse(states == "Northern Territory", 4, ifelse(s
tates == "Queensland", 4, ifelse(states == "South Australia", 4, ifelse(states == "Tasma
nia", 2, ifelse(states == "Victoria", 11, ifelse(states == "Western Australia", 7, 1
))))))))))

# recreate the graph with rainfall normalized
state_weather %>%
  filter(!is.na(Rainfall)) %>%
  group_by(State) %>%
  summarize(rainfall = sum(Rainfall)) %>%
  left_join(city_count, by="State") %>%
  mutate(rainfall = rainfall/count) %>%
  ggplot(aes(State, rainfall)) +
  geom_bar(stat = "identity", fill = "#252580") +
  theme(axis.text.x = element_text(angle = 45, hjust = 1))
```

```
## Warning: Column `State` joining character vector and factor, coercing into
## character vector
```



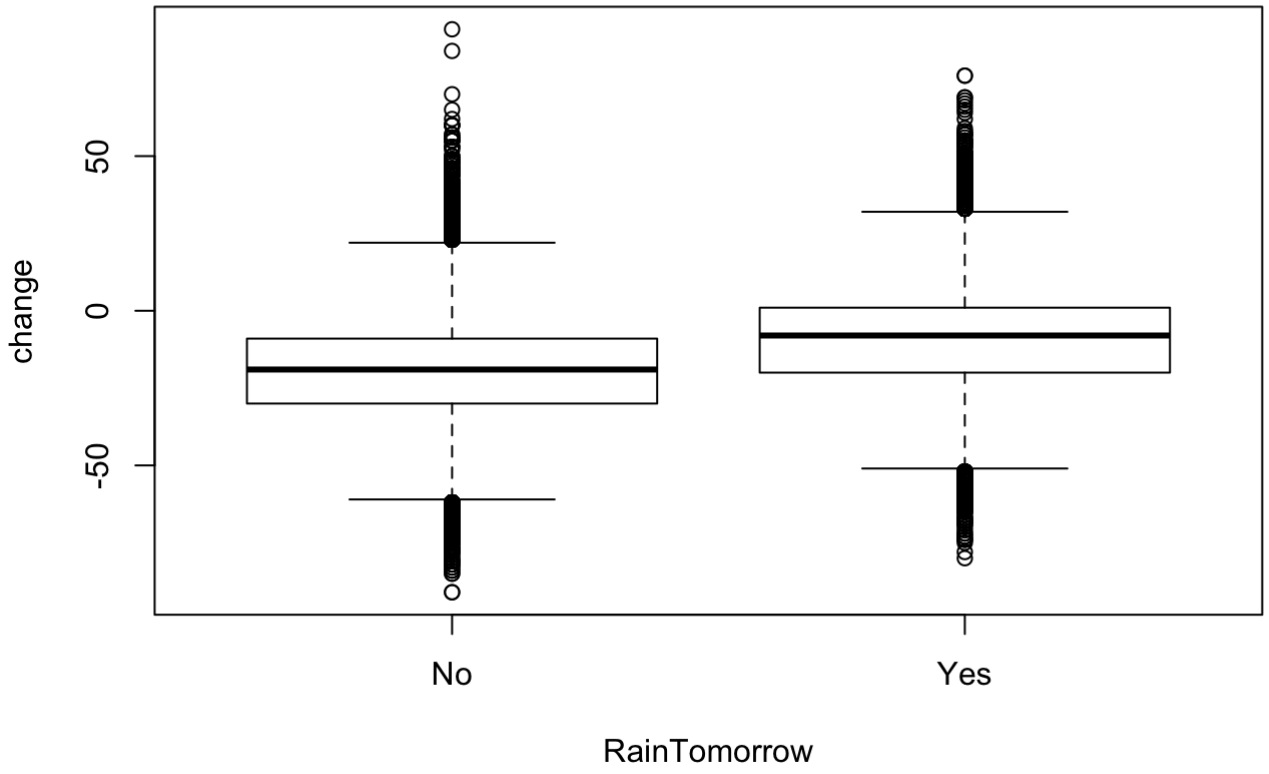
It looks like Queensland is our wettest state now. After looking online, Wikipedia says that Victoria and Tasmania are the wettest states. Maybe I didn't normalize my counts correctly? Do some cities have more days of data than others? I'm not sure, but it doesn't look like the location alone is going to be a big help in predicting rain. I may want to combine the city and wind gust direction into a single variable, that might yield better predicting results.

The next variable I will look at is the humidity. Do humid days tend to mean rain is coming tomorrow? Does a big change in humidity from 9am to 3pm mean anything? I will be following the same format of finding the average change and categorizing the variable as well.

```
# find average change in humidity for days that rained tomorrow and didn't rain tomorrow
weather %>%
  filter(year %in% years, !is.na(RainTomorrow), !is.na(Humidity9am), !is.na(Humidity3p
m)) %>%
  mutate(humidity_change = Humidity3pm - Humidity9am) %>%
  group_by(RainTomorrow) %>%
  summarize(average_change = mean(humidity_change))
```

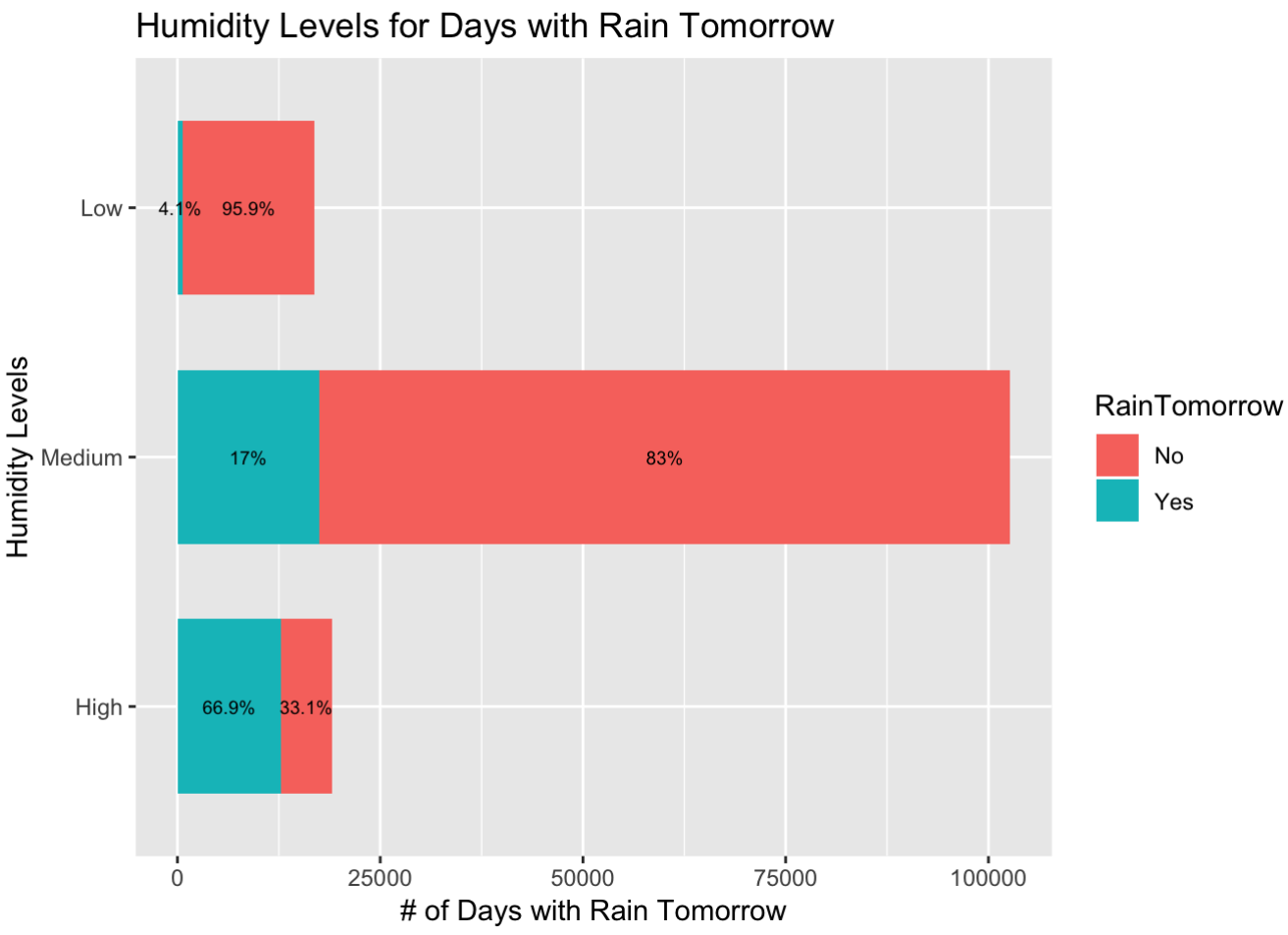
```
## # A tibble: 2 x 2
##   RainTomorrow average_change
##   <fct>                 <dbl>
## 1 No                    -19.6
## 2 Yes                    -9.18
```

```
# save data into variable for boxplot chart
humidity_change <- weather %>%
  filter(year %in% years, !is.na(RainTomorrow), !is.na(Humidity9am), !is.na(Humidity3p
m)) %>%
  mutate(change = Humidity3pm - Humidity9am) %>%
  select(RainTomorrow, change)

# create boxplot
boxplot(change ~ RainTomorrow, data=humidity_change)
```



That's a decent difference., in the mean, but we still have a lot of overlap in our boxplots. We might want to include the change in humidity in our model. Let's look at low vs high humidity.

```
# Does high humidity mean rain tomorrow?
humidity_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Humidity3pm)) %>%
  mutate(Humidity3pm = ifelse(Humidity3pm <= 25, "Low", ifelse(Humidity3pm <= 74, "Mediu
m", "High"))) %>%
  group_by(Humidity3pm, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange
humidity_categorized$Humidity3pm <- factor(humidity_categorized$Humidity3pm, levels = c(
"High", "Medium", "Low"))
humidity_categorized <- arrange(humidity_categorized, Humidity3pm)

# create visualization
humidity_categorized %>%
  group_by(Humidity3pm) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Humidity3pm, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Humidity Levels") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Humidity Levels for Days with Rain Tomorrow")
```



That's a better percentage for the high humidity than the low pressure sefction had. Humidity will probably be a good variable to keep.

Let's check the cloud variable next.

```
# there are a lot of NA's in the cloud data. How many rows do we have with complete 9am
 and 3pm data?
(weather %>% filter(!is.na(Cloud9am), !is.na(Cloud3pm)) %>% summarize(count = n())) / (w
eather %>% summarize(count = n()))
```

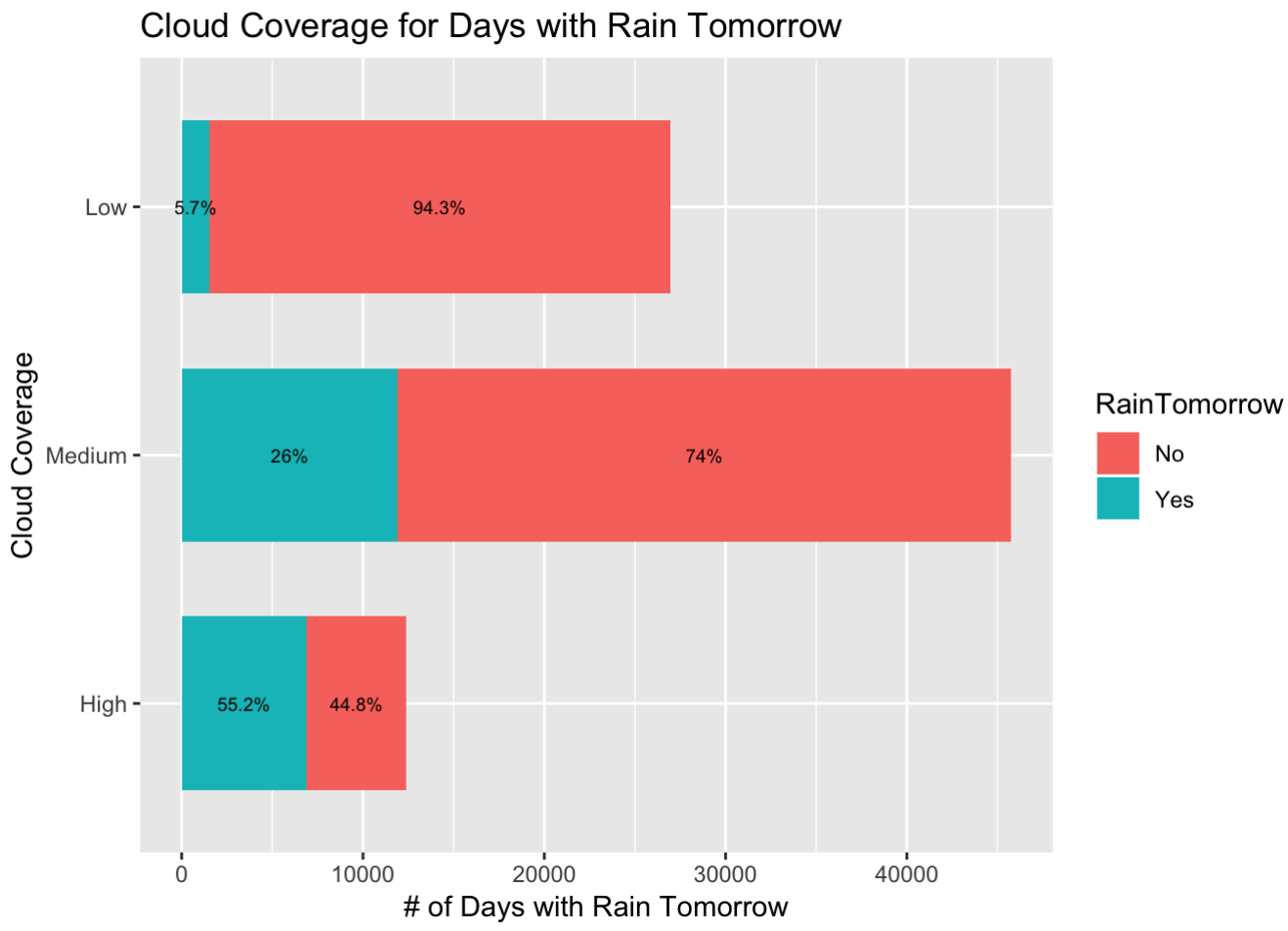
```
##       count
## 1 0.5704852
```

```
# only just over half of all days in this dataset have data for the 9am and 3pm cloud da
ta. I would assume that the data is missing at random, I have no reason to suggest other
wise. For the purpose of data exploration, I will just remove the missing values
# let's look at how cloud coverage changes with regards to whether it rains tomorrow or
 not
weather %>%
  filter(!is.na(Cloud9am), !is.na(Cloud3pm)) %>%
  mutate(cloud_change = Cloud3pm - Cloud9am) %>%
  group_by(RainTomorrow) %>%
  summarize(average_change = mean(cloud_change))
```

```
## # A tibble: 2 x 2
##   RainTomorrow average_change
##   <fct>                 <dbl>
## 1 No                  -0.00420
## 2 Yes                  0.270
```

There isn't much a change at all. Let's look for very cloudy vs sunny days. Again, we will categorize them into low, medium and high values.

```
# Does very cloudy mean rain tomorrow?
cloud_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Cloud3pm)) %>%
  mutate(Cloud3pm = ifelse(Cloud3pm <= 2, "Low", ifelse(Cloud3pm <= 7, "Medium", "High"
))) %>%
  group_by(Cloud3pm, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange
cloud_categorized$Cloud3pm <- factor(cloud_categorized$Cloud3pm, levels = c("High", "Med
ium", "Low"))
cloud_categorized <- arrange(cloud_categorized, Cloud3pm)

# build visualization
cloud_categorized %>%
  group_by(Cloud3pm) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Cloud3pm, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Cloud Coverage") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Cloud Coverage for Days with Rain Tomorrow")
```

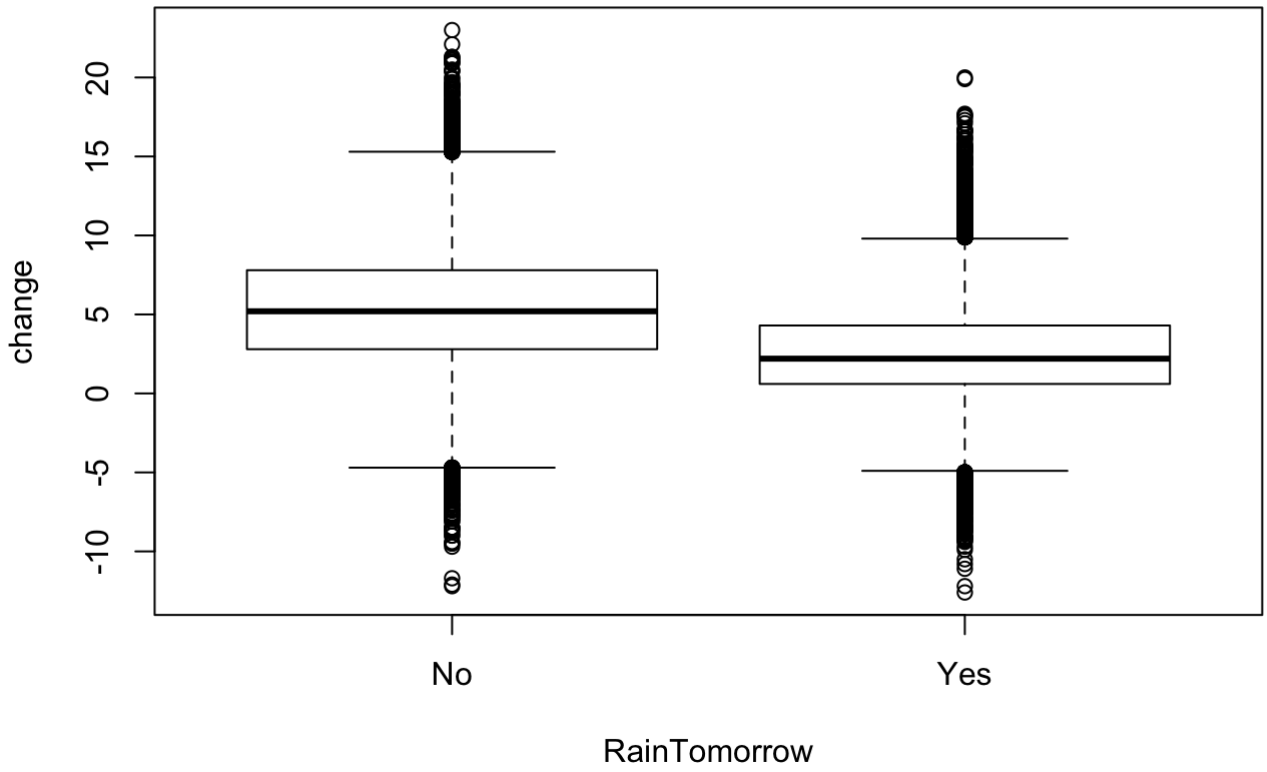


Not as good as I was hoping. Very cloudy days in the afternoon have a little over 50% chance of raining tomorrow. Cloud coverage is probably a variable I can leave out. Let's move on to temperature. We've already seen that temperature changes with the season and doesn't show much in regards to rain tomorrow. Let's just look at the change in temperature from 9am to 3pm.

```
# Temp for RainTomorrow
weather %>%
  filter(!is.na(Temp9am), !is.na(Temp3pm), !is.na(RainTomorrow)) %>%
  mutate(temp_change = Temp3pm - Temp9am) %>%
  group_by(RainTomorrow) %>%
  summarize(average_change = mean(temp_change))
```

```
## # A tibble: 2 x 2
##   RainTomorrow average_change
##   <fct>                 <dbl>
## 1 No                     5.40
## 2 Yes                    2.59
```

```
# check boxplot distributions
temp_change <- weather %>%
  filter(!is.na(Temp9am), !is.na(Temp3pm), !is.na(RainTomorrow)) %>%
  mutate(change = Temp3pm - Temp9am) %>%
  select(RainTomorrow, change)
boxplot(change ~ RainTomorrow, data=temp_change)
```



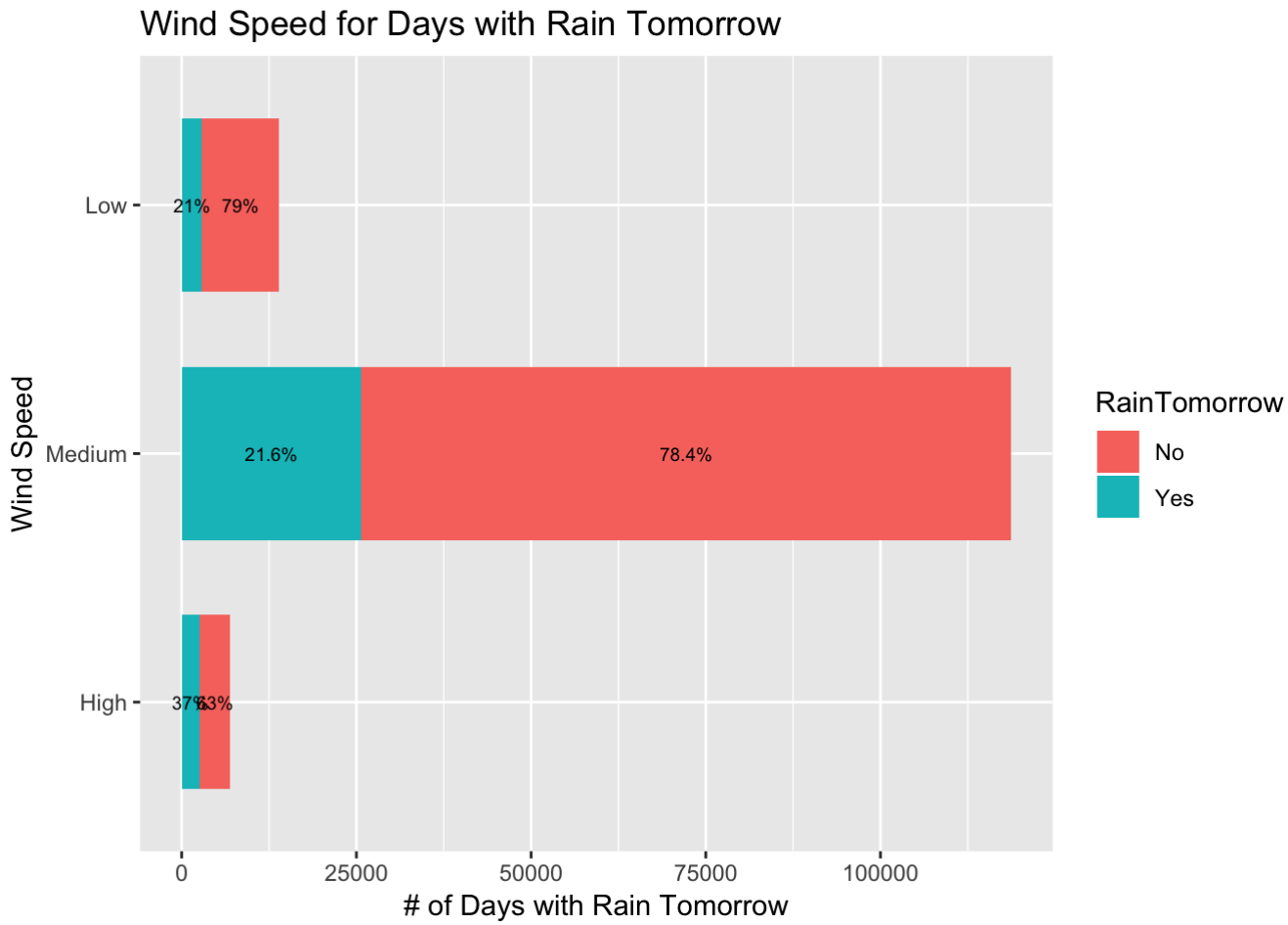
Not sure if the change in temperature will be helpful. There is a difference, on days where it rains tomorrow, it tends to not go up in temperature quite as much. Move on to wind speed.

```
# Does wind speed change have a relationship whether it rained tomorrow or not?
weather %>%
  filter(!is.na(WindSpeed9am), !is.na(WindSpeed3pm), !is.na(RainTomorrow)) %>%
  mutate(wind_change = WindSpeed3pm - WindSpeed9am) %>%
  group_by(RainTomorrow) %>%
  summarize(average_change = mean(wind_change))
```

```
## # A tibble: 2 x 2
##   RainTomorrow average_change
##   <fct>                 <dbl>
## 1 No                     4.63
## 2 Yes                    4.55
```

That's a pretty even split, it sure doesn't seem like that plays a part in it.

```
# Do very windy days indicate rain tomorrow?
wind_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(WindSpeed3pm)) %>%
  mutate(WindSpeed3pm = ifelse(WindSpeed3pm <= 7, "Low", ifelse(WindSpeed3pm <= 33, "Med
ium", "High"))) %>%
  group_by(WindSpeed3pm, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange
wind_categorized$WindSpeed3pm <- factor(wind_categorized$WindSpeed3pm, levels = c("High"
, "Medium", "Low"))
wind_categorized <- arrange(wind_categorized, WindSpeed3pm)

# build visualization
wind_categorized %>%
  group_by(WindSpeed3pm) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(WindSpeed3pm, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Wind Speed") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Wind Speed for Days with Rain Tomorrow")
```

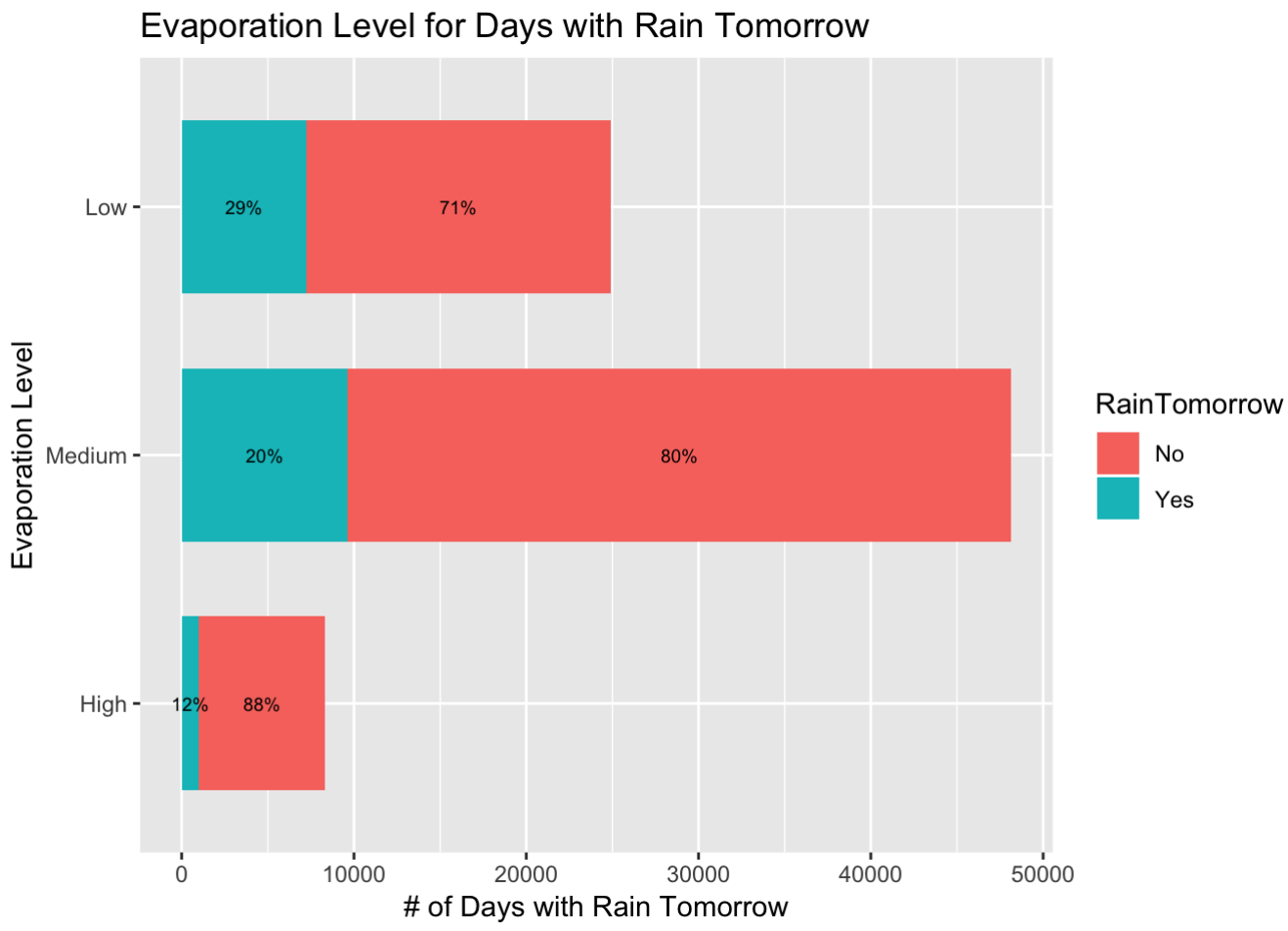


That doesn't show much at all. Let's just move right on to evaporation.

```
# How much evaporation data do we have?
(weather %>% filter(!is.na(Evaporation)) %>% summarize(count = n())) / (weather %>% summ
arize(count = n()))
```

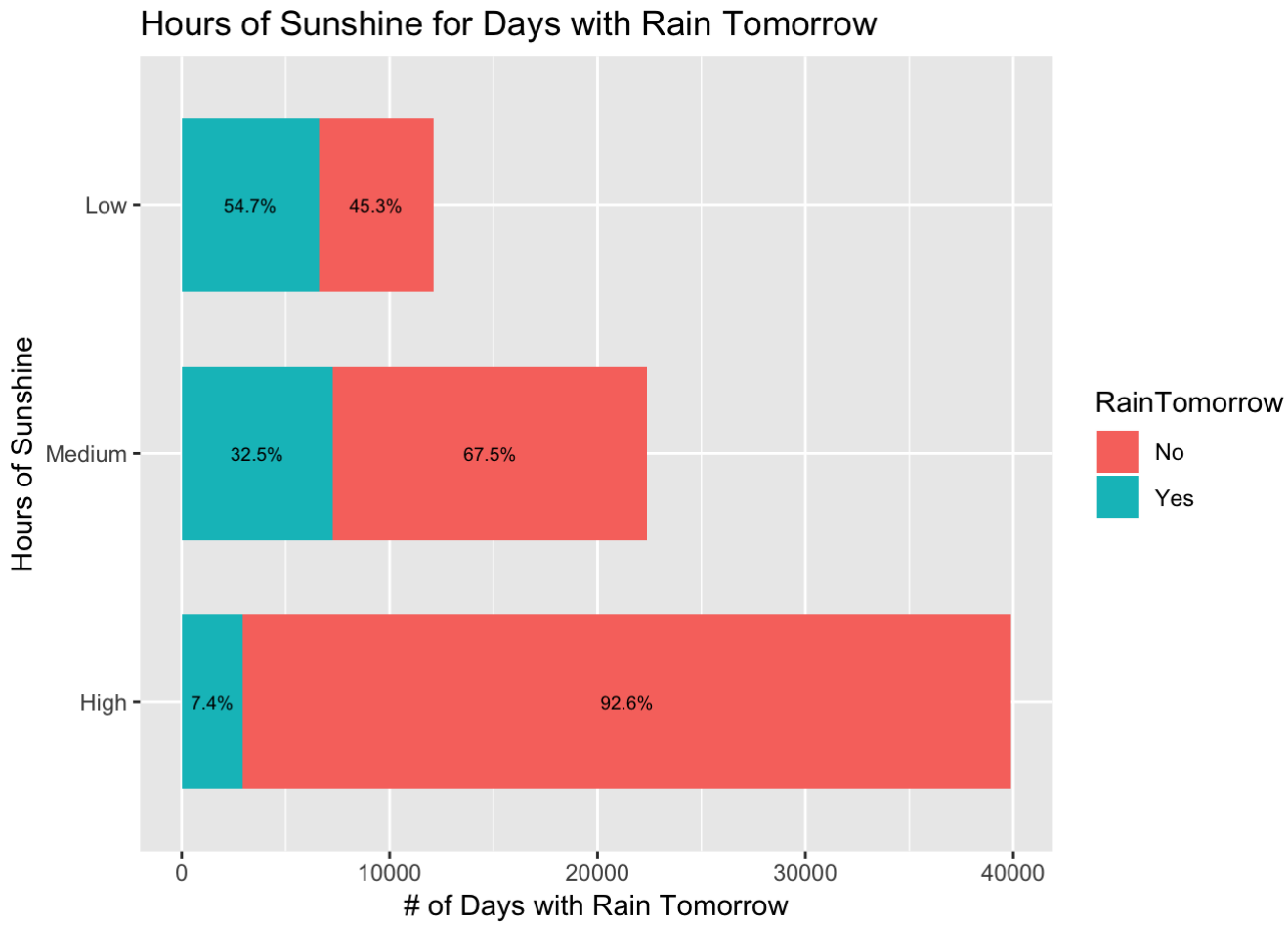
```
##       count
## 1 0.5721097
```

```
# let's categorize it
evaporation_categorized <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Evaporation)) %>%
  mutate(Evaporation = ifelse(Evaporation <= 3, "Low", ifelse(Evaporation <= 10, "Mediu
m", "High"))) %>%
  group_by(Evaporation, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange
evaporation_categorized$Evaporation <- factor(evaporation_categorized$Evaporation, level
s = c("High", "Medium", "Low"))
evaporation_categorized <- arrange(evaporation_categorized, Evaporation)

# build visualization
evaporation_categorized %>%
  group_by(Evaporation) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Evaporation, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Evaporation Level") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Evaporation Level for Days with Rain Tomorrow")
```



Again, not much there. evaporation might not be that good of an indicator. Next variable is Sunshine.

```
# categorize data
sunshine_evaporation <- weather %>%
  filter(!is.na(RainTomorrow), !is.na(Sunshine)) %>%
  mutate(Sunshine = ifelse(Sunshine <= 3, "Low", ifelse(Sunshine <= 8, "Medium", "High"
))) %>%
  group_by(Sunshine, RainTomorrow) %>%
  summarize(count = n())

# factor and arrange
sunshine_evaporation$Sunshine <- factor(sunshine_evaporation$Sunshine, levels = c("High"
, "Medium", "Low"))
sunshine_evaporation <- arrange(sunshine_evaporation, Sunshine)

# build visualization
sunshine_evaporation %>%
  group_by(Sunshine) %>%
  mutate(percent = count/sum(count) * 100) %>%
  mutate(label = paste(round(percent, digits = 1), "%", sep = "", collapse = NULL)) %>%
  ggplot(aes(Sunshine, count, fill = RainTomorrow)) +
  geom_bar(position = position_stack(), stat = "identity", width = .7) +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5), size = 2.5) +
  coord_flip() +
  xlab("Hours of Sunshine") +
  ylab("# of Days with Rain Tomorrow") +
  ggtitle("Hours of Sunshine for Days with Rain Tomorrow")
```



This is really similar to the cloudy variable graph, which makes sense. High cloudy days are the same as days with low amounts of sunshine.

Analysis

After data exploration, let's review the variables that looked like might add value to the model. These were the best variables I think we saw:
Month
Rainfall
Pressure 3pm
City + WindGustDir
Humidity 3pm
Humidity Change
Sunshine

These variables might add value too:
MaxTemp
Evaporation
Cloudy 3pm
Temp Change

It is time to start getting ready for the modeling. The first thing we need to do is clean up the data and fix all the missing values. We've already decided that this data is missing at random. We don't want to just exclude the missing data, since we have way too large of a section of our data with missing values. We want to replace the missing values with plausible data. One of the more popular methods for doing this is called imputation. The

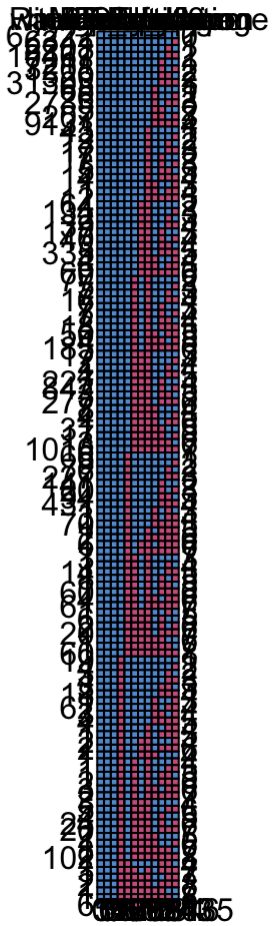
MICE package in R will implement the imputation on our data. This package generates Multivariate Imputations by Chained Equations, or MICE. For further explanation on the method, please refer online to the documentation. Basically, the method looks at the data that is there and decides what values would be plausible to go into the missing spaces, based on the relationships between the data. (It will more likely add a low Sunshine amount when the Cloud 3pm is a 7 or 8, instead of a high Sunshine amount.)

Before running the imputation, I will create the dataset we will use for modeling, removing the superfluous variables.

```
# create modeling dataset. I will start with all variables that might add value
# we need to put the month in there, combine city and windgustdir
weather_model <- weather %>%
  mutate(month = format(Date, "%m"), wind_location = apply(.,1 ,function(x) paste(toStri
ng(x[2]), toString(x[8]))), humidity_change = Humidity3pm - Humidity9am, temp_change = T
emp3pm - Temp9am) %>%
  select(month, Rainfall, Pressure3pm, wind_location, Humidity3pm, humidity_change, Suns
hine, MaxTemp, Evaporation, Cloud3pm, temp_change, RainTomorrow)
```

Now we can begin the imputation. To really confirm that our data is missing at random, there are some functions in the MICE and VIM packages that help visualy confirm this.

```
# check how many records are missing for which variables.
md.pattern(weather_model)
```



```
##       month wind_location RainTomorrow MaxTemp Rainfall temp_change
## 62222     1             1            1       1        1           1
## 6341      1             1            1       1        1           1
## 2208      1             1            1       1        1           1
## 10467     1             1            1       1        1           1
## 7211      1             1            1       1        1           1
## 3208      1             1            1       1        1           1
## 1520      1             1            1       1        1           1
## 31303     1             1            1       1        1           1
## 65        1             1            1       1        1           1
## 26        1             1            1       1        1           1
## 2733      1             1            1       1        1           1
## 5         1             1            1       1        1           1
## 207       1             1            1       1        1           1
## 9131      1             1            1       1        1           1
## 43        1             1            1       1        1           1
## 12        1             1            1       1        1           1
## 3         1             1            1       1        1           1
## 17        1             1            1       1        1           1
## 17        1             1            1       1        1           1
## 3         1             1            1       1        1           1
## 5         1             1            1       1        1           1
## 14        1             1            1       1        1           1
## 1         1             1            1       1        1           1
## 1         1             1            1       1        1           1
## 12        1             1            1       1        1           1
## 64        1             1            1       1        1           1
## 134       1             1            1       1        1           1
## 2         1             1            1       1        1           1
## 39        1             1            1       1        1           1
## 177       1             1            1       1        1           1
## 40        1             1            1       1        1           1
## 3         1             1            1       1        1           1
## 335       1             1            1       1        1           1
## 4         1             1            1       1        1           1
## 6         1             1            1       1        1           1
## 60        1             1            1       1        1           1
## 77        1             1            1       1        1           0
## 5         1             1            1       1        1           0
## 7         1             1            1       1        1           0
## 16        1             1            1       1        1           0
## 7         1             1            1       1        1           0
## 7         1             1            1       1        1           0
## 2         1             1            1       1        1           0
## 18        1             1            1       1        1           0
## 50        1             1            1       1        1           0
## 4         1             1            1       1        1           0
## 185       1             1            1       1        1           0
## 17        1             1            1       1        1           0
## 2         1             1            1       1        1           0
## 1         1             1            1       1        1           0
## 4         1             1            1       1        1           0
## 227       1             1            1       1        1           0
```

```
## 842      1              1              1              1              1              0
## 3        1              1              1              1              1              0
## 277      1              1              1              1              1              0
## 8        1              1              1              1              1              0
## 4        1              1              1              1              1              0
## 1        1              1              1              1              1              0
## 31       1              1              1              1              1              0
## 17       1              1              1              1              1              0
## 13       1              1              1              1              1              0
## 1016     1              1              1              1              1              0
## 66       1              1              1              1              0              1
## 9        1              1              1              1              0              1
## 76       1              1              1              1              0              1
## 227      1              1              1              1              0              1
## 141      1              1              1              1              0              1
## 20       1              1              1              1              0              1
## 94       1              1              1              1              0              1
## 431      1              1              1              1              0              1
## 1        1              1              1              1              0              1
## 3        1              1              1              1              0              1
## 70       1              1              1              1              0              1
## 1        1              1              1              1              0              1
## 2        1              1              1              1              0              1
## 6        1              1              1              1              0              1
## 1        1              1              1              1              0              1
## 3        1              1              1              1              0              0
## 1        1              1              1              1              0              0
## 14       1              1              1              1              0              0
## 8        1              1              1              1              0              0
## 1        1              1              1              1              0              0
## 60       1              1              1              1              0              0
## 4        1              1              1              1              0              0
## 61       1              1              1              1              0              0
## 1        1              1              1              1              0              0
## 6        1              1              1              1              0              0
## 2        1              1              1              1              0              0
## 20       1              1              1              1              0              0
## 1        1              1              1              1              0              0
## 1        1              1              1              1              0              0
## 60       1              1              1              1              0              0
## 19       1              1              1              0              1              1
## 4        1              1              1              0              1              1
## 3        1              1              1              0              1              1
## 4        1              1              1              0              1              1
## 3        1              1              1              0              1              1
## 18       1              1              1              0              1              1
## 1        1              1              1              0              1              1
## 62       1              1              1              0              1              1
## 3        1              1              1              0              1              1
## 4        1              1              1              0              1              1
## 1        1              1              1              0              1              1
## 1        1              1              1              0              1              1
## 2        1              1              1              0              1              1
## 2        1              1              1              0              1              1
```

```
## 1          1              1              1        0         1             1
## 1          1              1              1        0         1             0
## 1          1              1              1        0         1             0
## 1          1              1              1        0         1             0
## 2          1              1              1        0         1             0
## 3          1              1              1        0         1             0
## 2          1              1              1        0         1             0
## 3          1              1              1        0         1             0
## 5          1              1              1        0         1             0
## 4          1              1              1        0         1             0
## 24         1              1              1        0         1             0
## 20         1              1              1        0         1             0
## 2          1              1              1        0         1             0
## 1          1              1              1        0         1             0
## 8          1              1              1        0         1             0
## 102        1              1              1        0         1             0
## 1          1              1              1        0         0             1
## 3          1              1              1        0         0             0
## 1          1              1              1        0         0             0
## 2          1              1              1        0         0             0
## 1          1              1              1        0         0             0
## 1          1              1              1        0         0             0
## 6          1              1              1        0         0             0
##            0              0              0      322      1406          3277
##       Humidity3pm humidity_change Pressure3pm Cloud3pm Evaporation
## 62222           1               1           1        1           1
## 6341            1               1           1        1           1
## 2208            1               1           1        1           0
## 10467           1               1           1        1           0
## 7211            1               1           1        0           1
## 3208            1               1           1        0           1
## 1520            1               1           1        0           0
## 31303           1               1           1        0           0
## 65              1               1           0        1           1
## 26              1               1           0        1           1
## 2733            1               1           0        1           0
## 5               1               1           0        0           1
## 207             1               1           0        0           1
## 9131            1               1           0        0           0
## 43              1               0           1        1           1
## 12              1               0           1        1           1
## 3               1               0           1        1           0
## 17              1               0           1        1           0
## 17              1               0           1        0           1
## 3               1               0           1        0           1
## 5               1               0           1        0           0
## 14              1               0           1        0           0
## 1               1               0           0        1           1
## 1               1               0           0        1           0
## 12              1               0           0        0           0
## 64              0               0           1        1           1
## 134             0               0           1        1           1
## 2               0               0           1        1           0
## 39              0               0           1        1           0
```

```
## 177  0  0  1  0  1
## 40   0  0  1  0  1
## 3    0  0  1  0  0
## 335  0  0  1  0  0
## 4    0  0  0  1  1
## 6    0  0  0  1  0
## 60   0  0  0  0  0
## 77   1  0  1  1  1
## 5    1  0  1  1  1
## 7    1  0  1  1  0
## 16   1  0  1  1  0
## 7    1  0  1  0  1
## 7    1  0  1  0  1
## 2    1  0  1  0  0
## 18   1  0  1  0  0
## 50   1  0  0  1  0
## 4    1  0  0  0  1
## 185  1  0  0  0  0
## 17   0  0  1  1  1
## 2    0  0  1  1  1
## 1    0  0  1  1  0
## 4    0  0  1  1  0
## 227  0  0  1  0  1
## 842  0  0  1  0  1
## 3    0  0  1  0  0
## 277  0  0  1  0  0
## 8    0  0  0  1  1
## 4    0  0  0  1  1
## 1    0  0  0  1  0
## 31   0  0  0  0  1
## 17   0  0  0  0  1
## 13   0  0  0  0  0
## 1016 0  0  0  0  0
## 66   1  1  1  1  1
## 9    1  1  1  1  1
## 76   1  1  1  1  0
## 227  1  1  1  1  0
## 141  1  1  1  0  1
## 20   1  1  1  0  1
## 94   1  1  1  0  0
## 431  1  1  1  0  0
## 1    1  1  0  1  0
## 3    1  1  0  1  0
## 70   1  1  0  0  0
## 1    1  0  1  0  0
## 2    1  0  0  0  0
## 6    0  0  1  0  0
## 1    0  0  0  0  0
## 3    1  0  1  1  1
## 1    1  0  1  1  0
## 14   1  0  1  1  0
## 8    1  0  1  0  1
## 1    1  0  1  0  0
## 60   1  0  1  0  0
```

```
## 4                1                0             0             1             0
## 61               1                0             0             0             0
## 1                0                0             1             1             0
## 6                0                0             1             0             1
## 2                0                0             1             0             1
## 20               0                0             1             0             0
## 1                0                0             0             0             1
## 1                0                0             0             0             0
## 60               0                0             0             0             0
## 19               1                1             1             1             1
## 4                1                1             1             1             1
## 3                1                1             1             1             0
## 4                1                1             1             1             0
## 3                1                1             1             0             0
## 18               1                1             1             0             0
## 1                1                1             0             1             1
## 62               1                1             0             1             0
## 3                1                1             0             0             1
## 4                1                1             0             0             0
## 1                1                0             1             1             0
## 1                1                0             1             0             0
## 2                0                0             1             1             1
## 2                0                0             1             0             0
## 1                0                0             0             1             1
## 1                1                0             1             1             1
## 1                1                0             1             1             0
## 1                1                0             1             0             0
## 2                1                0             1             0             0
## 3                1                0             0             1             0
## 2                1                0             0             0             0
## 3                0                0             1             1             1
## 5                0                0             1             1             0
## 4                0                0             1             0             1
## 24               0                0             1             0             1
## 20               0                0             1             0             0
## 2                0                0             0             1             1
## 1                0                0             0             1             1
## 8                0                0             0             0             0
## 102              0                0             0             0             0
## 1                1                1             1             0             0
## 3                1                0             1             0             0
## 1                0                0             1             1             0
## 2                0                0             1             0             0
## 1                0                0             0             0             1
## 1                0                0             0             0             0
## 6                0                0             0             0             0
##              3610             4286         13981         57094         60843
##       Sunshine
## 62222        1      0
## 6341         0      1
## 2208         1      1
## 10467        0      2
## 7211         1      1
## 3208         0      2
```

```
## 1520         1       2
## 31303        0       3
## 65           1       1
## 26           0       2
## 2733         0       3
## 5            1       2
## 207          0       3
## 9131         0       4
## 43           1       1
## 12           0       2
## 3            1       2
## 17           0       3
## 17           1       2
## 3            0       3
## 5            1       3
## 14           0       4
## 1            0       3
## 1            0       4
## 12           0       5
## 64           1       2
## 134          0       3
## 2            1       3
## 39           0       4
## 177          1       3
## 40           0       4
## 3            1       4
## 335          0       5
## 4            0       4
## 6            0       5
## 60           0       6
## 77           1       2
## 5            0       3
## 7            1       3
## 16           0       4
## 7            1       3
## 7            0       4
## 2            1       4
## 18           0       5
## 50           0       5
## 4            0       5
## 185          0       6
## 17           1       3
## 2            0       4
## 1            1       4
## 4            0       5
## 227          1       4
## 842          0       5
## 3            1       5
## 277          0       6
## 8            1       4
## 4            0       5
## 1            0       6
## 31           1       5
## 17           0       6
```

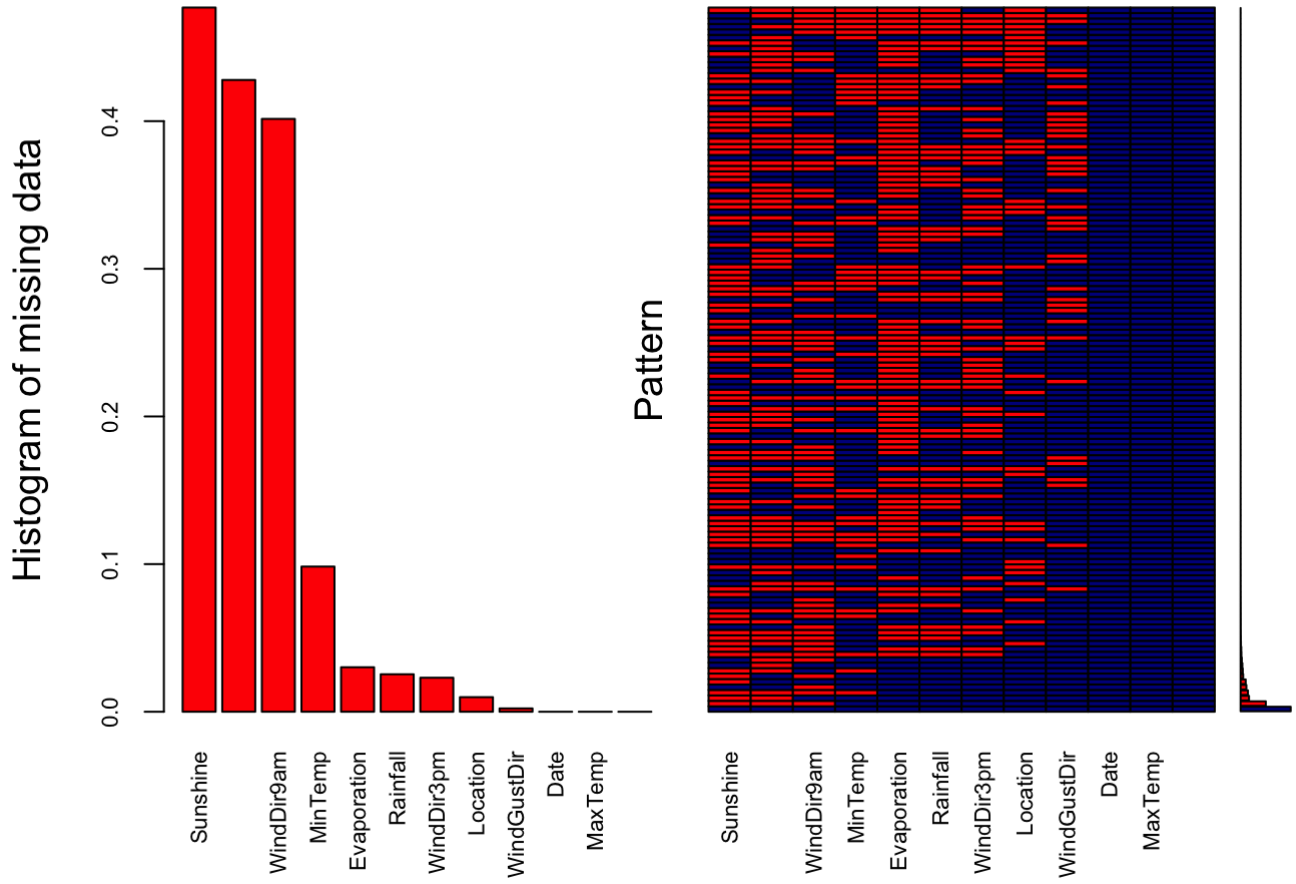
```
## 13           1       6
## 1016         0       7
## 66           1       1
## 9            0       2
## 76           1       2
## 227          0       3
## 141          1       2
## 20           0       3
## 94           1       3
## 431          0       4
## 1            1       3
## 3            0       4
## 70           0       5
## 1            0       5
## 2            0       6
## 6            0       6
## 1            0       7
## 3            1       3
## 1            1       4
## 14           0       5
## 8            0       5
## 1            1       5
## 60           0       6
## 4            0       6
## 61           0       7
## 1            1       5
## 6            1       5
## 2            0       6
## 20           0       7
## 1            1       6
## 1            1       7
## 60           0       8
## 19           1       1
## 4            0       2
## 3            1       2
## 4            0       3
## 3            1       3
## 18           0       4
## 1            0       3
## 62           0       4
## 3            0       4
## 4            0       5
## 1            0       4
## 1            0       5
## 2            0       4
## 2            0       6
## 1            1       4
## 1            0       4
## 1            0       5
## 1            1       5
## 2            0       6
## 3            0       6
## 2            0       7
## 3            1       4
```

```
## 5             0      6
## 4             1      5
## 24            0      6
## 20            0      7
## 2             1      5
## 1             0      6
## 8             1      7
## 102           0      8
## 1             1      4
## 3             0      7
## 1             0      7
## 2             0      8
## 1             1      7
## 1             1      8
## 6             0      9
##           67816 212635
```

From this, we can see that only 62,222 rows have complete data. 6,341 rows miss only the Sunshine variable, 2,208 miss only the Evaporation variable, and 10,467 miss both the Sunshine and Evaporation variables.

```
# Let's look at a more helpful visualization from the VIM package:
aggr(weather_model, col=c('darkblue','red'), numbers=TRUE, sortVars=TRUE, labels=names(w
eather), cex.axis=.7, gap=1, ylab=c("Histogram of missing data","Pattern"))
```

```
## Warning in plot.aggr(res, ...): not enough vertical space to display
## frequencies (too many combinations)
```

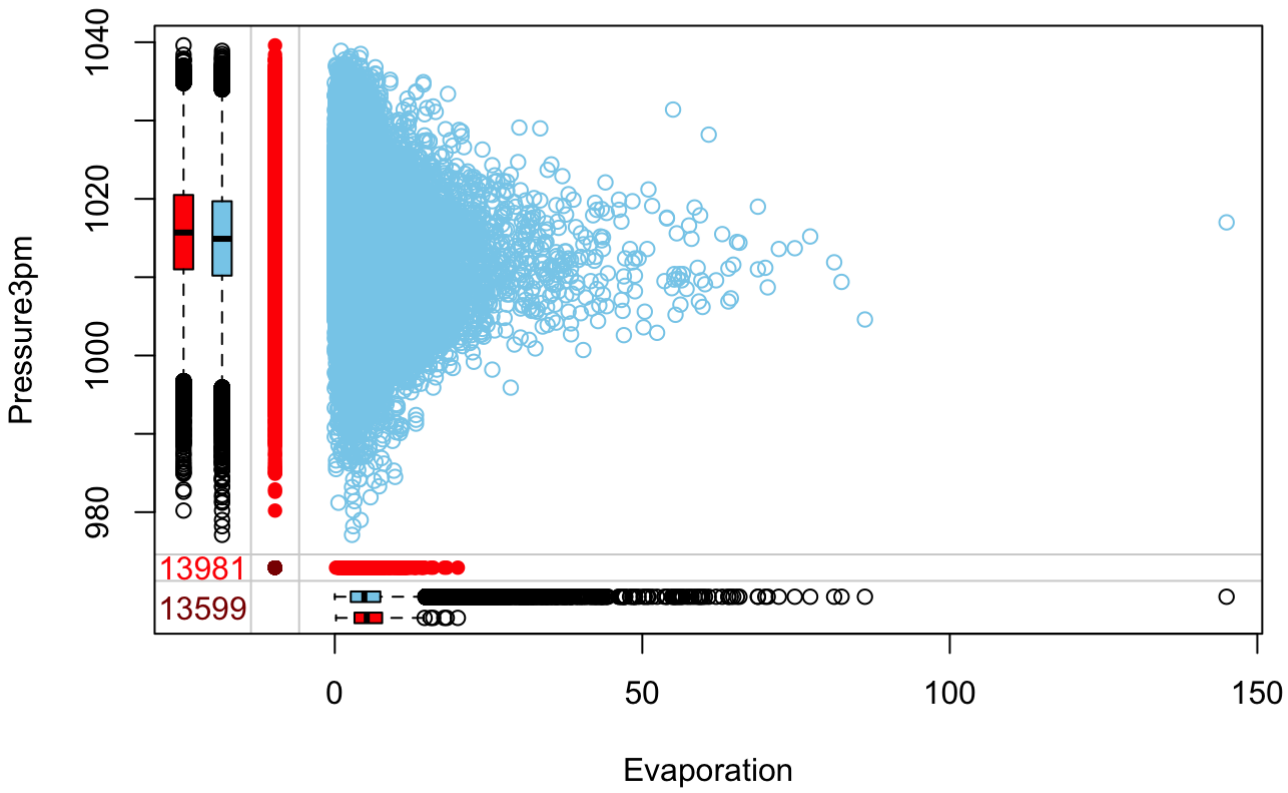



```
##
##  Variables sorted by number of missings:
##       Variable       Count
##       Sunshine 0.476929244
##  WindGustSpeed 0.427890262
##     WindDir9am 0.401524688
##        MinTemp 0.098324109
##    Evaporation 0.030142131
##       Rainfall 0.025388029
##     WindDir3pm 0.023046142
##       Location 0.009887969
##    WindGustDir 0.002264528
##           Date 0.000000000
##        MaxTemp 0.000000000
##   WindSpeed9am 0.000000000
```

Visually we can see how many records are missing. Just by looking at it, there doesn't seem to be any kind of pattern among what data is missing.

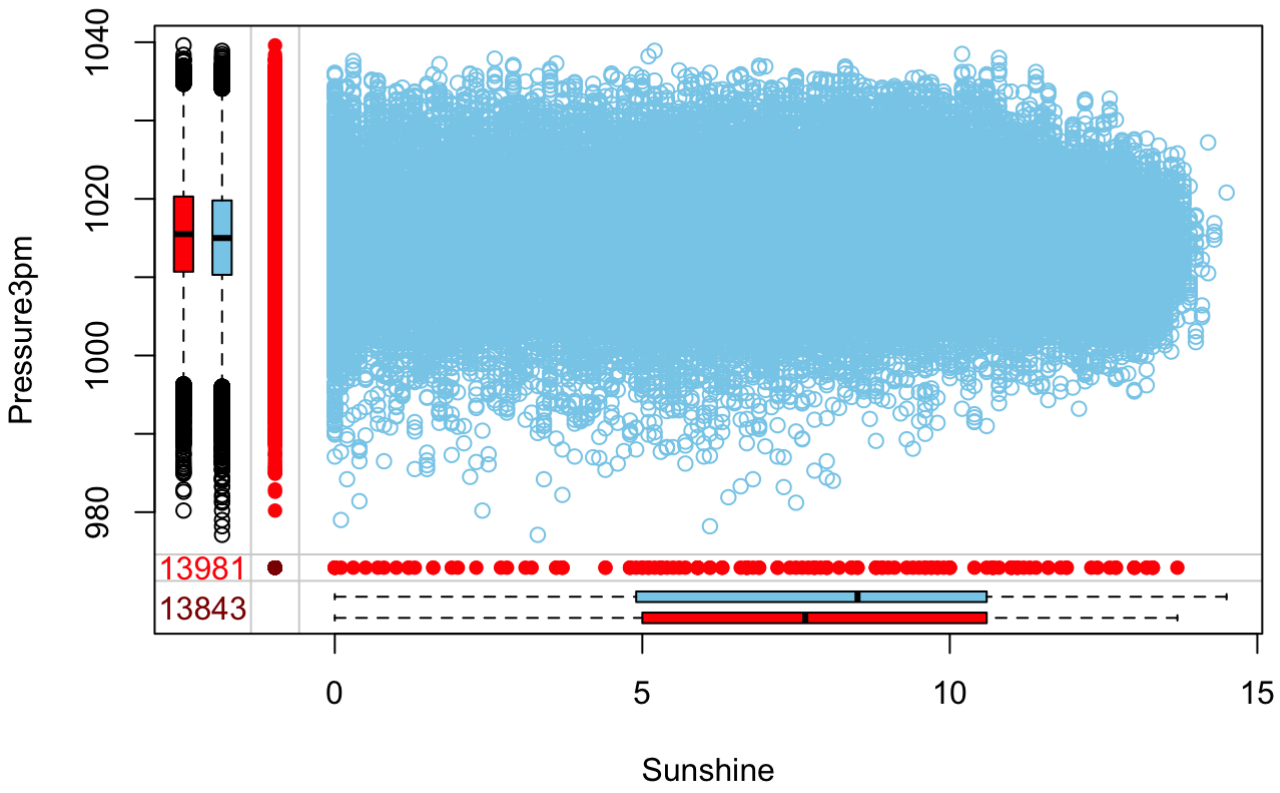
There is one more visualization that we can check. This approach only let's us see two variables at a time, but it will be a good confirmation that those values are missing at random.

```
# This next visualization will help us understand if Evaporation is missing at random. I
will plot it against pressure
marginplot(weather_model[c(9,3)])
```

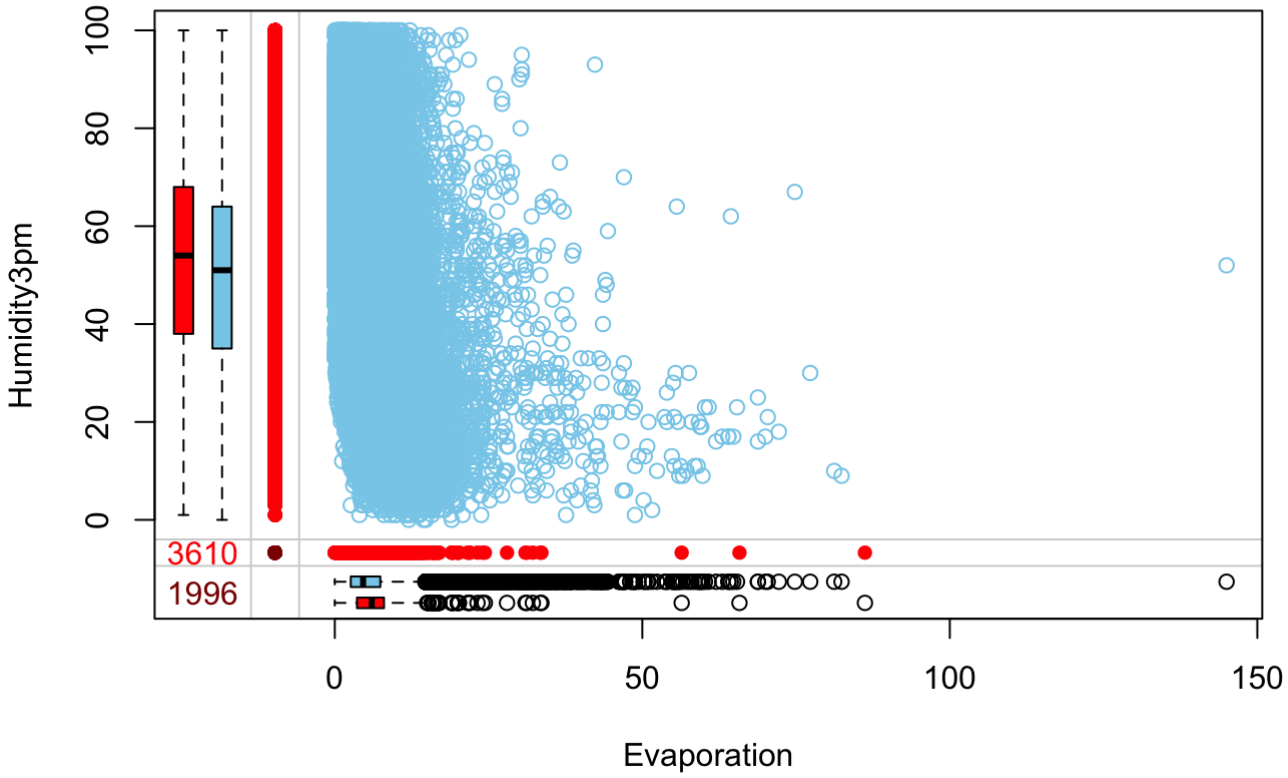


In this visualization we see blue and red boxplots. The red boxplot shows us the distribution of Pressure3pm with Evaporation missing. The blue boxplot shows the distribution of the remaining datapoints. If our assumption of missing at random is correct, we expect the red and blue box plots to be very similar. In this case, they are. Let's check a few other variable combinations just to make sure.

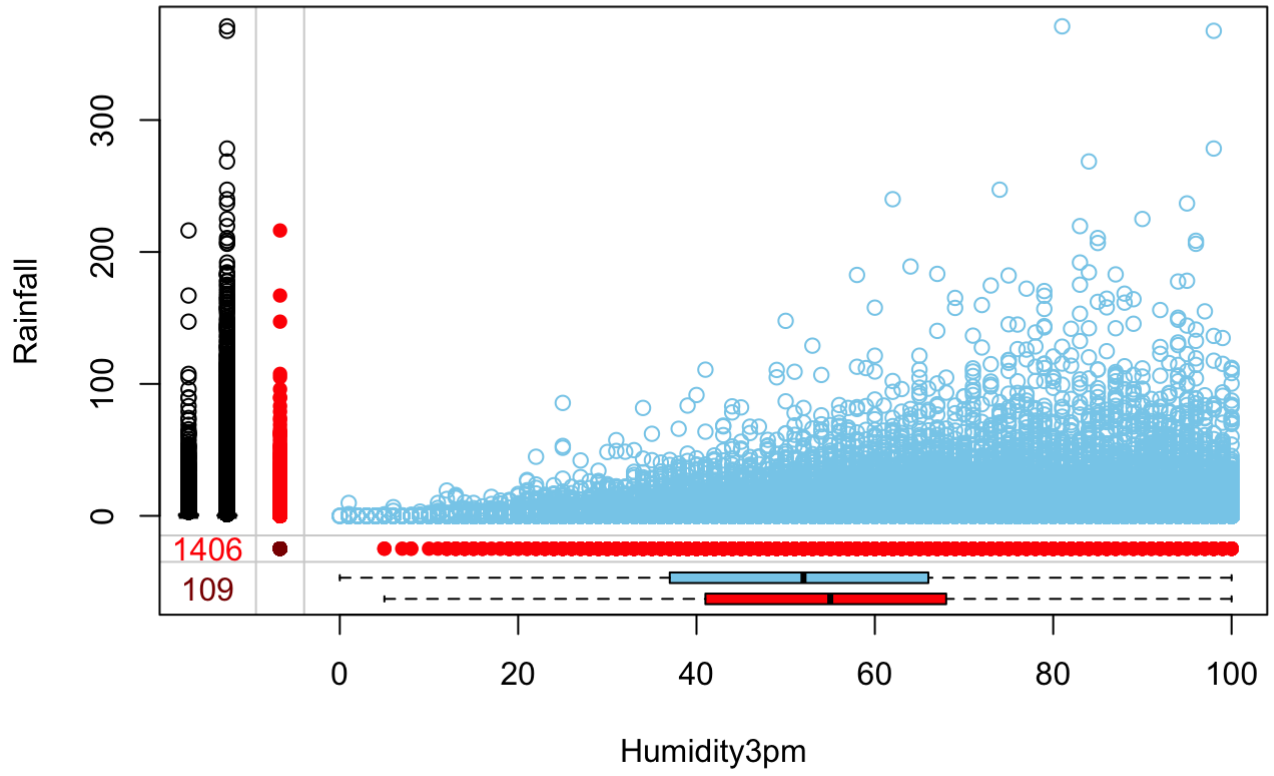
```
# This next visualization will help us understand if Evaporation is missing at random. I
will plot it against pressure
marginplot(weather_model[c(7,3)])
```



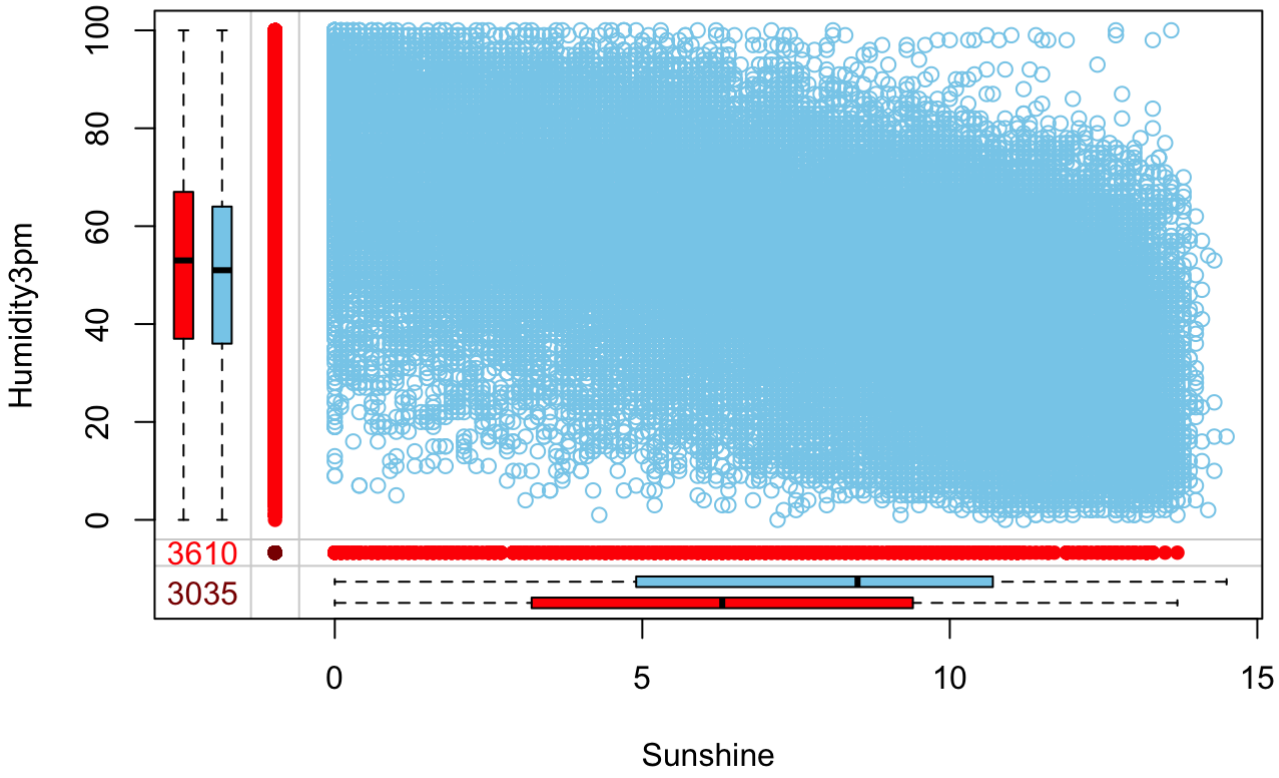

```
marginplot(weather_model[c(9,5)])
```




```
marginplot(weather_model[c(5,2)])
```




```
marginplot(weather_model[c(7,5)])
```



I don't see anything that really raises any big flags. The sunshine and humidity graph doesn't look completely random, but it does seem like it is still random enough.

I think we can feel safe that our assumption of missing at random is correct. We can now use imputation to fill in the missing data to prepare for modeling.

```
# run imputation. This will run 3 iterations with 5 imputations each. The method "pmm" m
eans "predictive mean matching"
tempData <- mice(weather_model, m=5, maxit=1, meth='pmm', seed=500)
```

```
##
##  iter imp variable
##    1   1  Rainfall  Pressure3pm  Humidity3pm  humidity_change  Sunshine  MaxTemp  Evap
oration  Cloud3pm  temp_change
##    1   2  Rainfall  Pressure3pm  Humidity3pm  humidity_change  Sunshine  MaxTemp  Evap
oration  Cloud3pm  temp_change
##    1   3  Rainfall  Pressure3pm  Humidity3pm  humidity_change  Sunshine  MaxTemp  Evap
oration  Cloud3pm  temp_change
##    1   4  Rainfall  Pressure3pm  Humidity3pm  humidity_change  Sunshine  MaxTemp  Evap
oration  Cloud3pm  temp_change
##    1   5  Rainfall  Pressure3pm  Humidity3pm  humidity_change  Sunshine  MaxTemp  Evap
oration  Cloud3pm  temp_change
```

```
## Warning: Number of logged events: 2
```

```
# summary for looking at which variables were imputed with which method
summary(tempData)
```

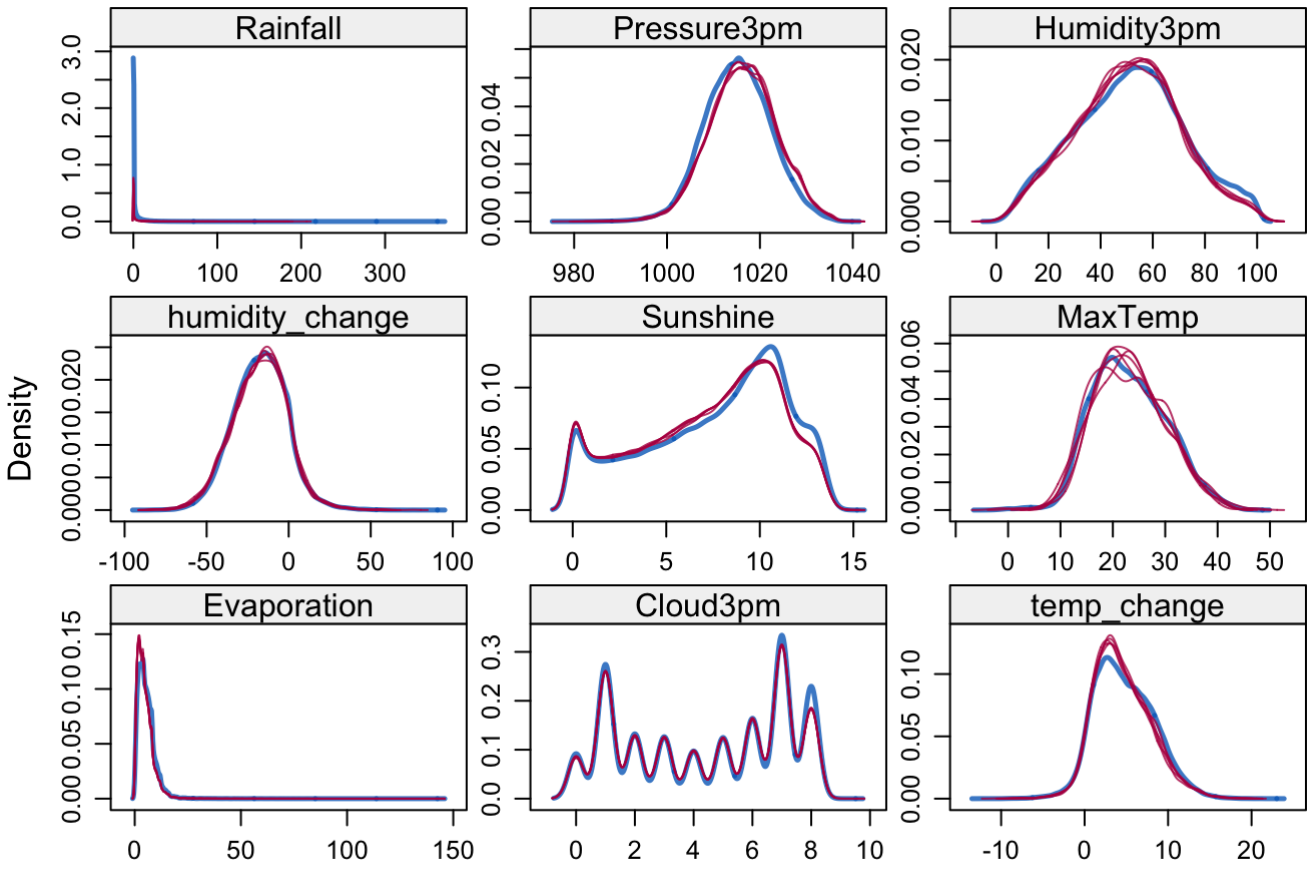
```
## Class: mids
## Number of multiple imputations:  5
## Imputation methods:
##           month        Rainfall     Pressure3pm   wind_location
##              ""           "pmm"           "pmm"              ""
##     Humidity3pm humidity_change        Sunshine         MaxTemp
##           "pmm"           "pmm"           "pmm"           "pmm"
##     Evaporation        Cloud3pm     temp_change    RainTomorrow
##           "pmm"           "pmm"           "pmm"              ""
## PredictorMatrix:
##                 month Rainfall Pressure3pm wind_location Humidity3pm
## month               0        1           1             0           1
## Rainfall            0        0           1             0           1
## Pressure3pm         0        1           0             0           1
## wind_location       0        1           1             0           1
## Humidity3pm         0        1           1             0           0
## humidity_change     0        1           1             0           1
##                 humidity_change Sunshine MaxTemp Evaporation Cloud3pm
## month                         1        1       1           1        1
## Rainfall                      1        1       1           1        1
## Pressure3pm                   1        1       1           1        1
## wind_location                 1        1       1           1        1
## Humidity3pm                   1        1       1           1        1
## humidity_change               0        1       1           1        1
##                 temp_change RainTomorrow
## month                     1            1
## Rainfall                  1            1
## Pressure3pm               1            1
## wind_location             1            1
## Humidity3pm               1            1
## humidity_change           1            1
## Number of logged events:  2
##   it im dep     meth          out
## 1  0  0     constant         month
## 2  0  0     constant wind_location
```

```
# to access the imputed data for a single variable, you would run this code:
# tempData$imp$Evaporation
```

With the predictive mean matching, the imputation has generated values for all the missing data in the entire dataset. Now we need to look at the imputed data to see if they really are plausible values, good enough to use in modeling. We do this by looking at density plots. MICE has a densityplot function that plots the imputed values against the original values in density plots. If all the density plots align fairly well, then we can assume that the imputed values are plausible, and we can then use them to fill in the missing data in our dataset and move on to modeling. The imputed datasets show in red, the original, observed data in blue.

There are plenty of ways that imputation could introduce incorrect data in our dataset. Circular dependence between two variables due to correlation, nonsensical data, impossible combinations, and collinearity or empty cells if you have a lot of columns and not very many rows. I believe that in this case, we will not need to worry about these. Let's check the density plots to see how our imputations did.

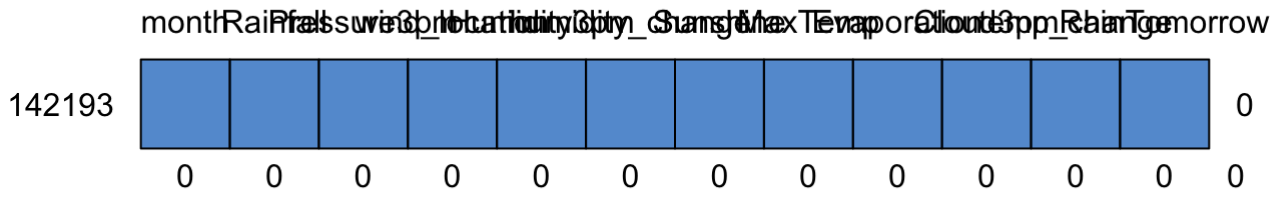
```
# run density plots to check for plausibility
densityplot(tempData)
```



The density plots looks great. Let's add the imputed data into the final dataset.

```
# add the imputed values back into the original dataset, using values from the first ite
ration.
completed_weather <- complete(tempData, 1)

# check to see if mice did its work
md.pattern(completed_weather)
```

```
##  /\     /\
## {  `---'  }
## {  O   O  }
## ==>  V <==  No need for mice. This data set is completely observed.
##  \  \|/  /
##   `-----'
```




```
##        month Rainfall Pressure3pm wind_location Humidity3pm
## 142193     1        1           1             1           1
##            0        0           0             0           0
##        humidity_change Sunshine MaxTemp Evaporation Cloud3pm temp_change
## 142193               1        1       1           1        1           1
##                      0        0       0           0        0           0
##        RainTomorrow
## 142193            1 0
##                   0 0
```

We have verified that we don't have any missing data anymore, so it is time to move on to the modeling section. As mentioned before, I will start with k-nearest neighbors and see how this model performs. To begin, I need to create the

```
# due to complexities I was getting from data classified as characters, I have dropped t
he month and wind_location variables.
completed_weather <- completed_weather %>% select(-month, -wind_location)

# change RainTomorrow to be binary 1 and 0, not "Yes" and "No"
completed_weather <- completed_weather %>% mutate(RainTomorrow = as.factor(ifelse(RainTo
morrow == "Yes", 1, 0)))
summary(completed_weather)
```

```
##     Rainfall        Pressure3pm      Humidity3pm     humidity_change
##  Min.   :  0.000   Min.   : 977.1   Min.   :  0.00   Min.   :-91.00
##  1st Qu.:  0.000   1st Qu.:1010.6   1st Qu.: 37.00   1st Qu.:-28.00
##  Median :  0.000   Median :1015.4   Median : 52.00   Median :-17.00
##  Mean   :  2.358   Mean   :1015.4   Mean   : 51.47   Mean   :-17.38
##  3rd Qu.:  0.800   3rd Qu.:1020.2   3rd Qu.: 66.00   3rd Qu.: -6.00
##  Max.   :371.000   Max.   :1039.6   Max.   :100.00   Max.   : 91.00
##     Sunshine         MaxTemp       Evaporation        Cloud3pm
##  Min.   : 0.000   Min.   :-4.80   Min.   :  0.000   Min.   :0.000
##  1st Qu.: 4.700   1st Qu.:17.90   1st Qu.:  2.400   1st Qu.:2.000
##  Median : 8.200   Median :22.60   Median :  4.400   Median :5.000
##  Mean   : 7.472   Mean   :23.23   Mean   :  5.189   Mean   :4.473
##  3rd Qu.:10.500   3rd Qu.:28.20   3rd Qu.:  7.000   3rd Qu.:7.000
##  Max.   :14.500   Max.   :48.10   Max.   :145.000   Max.   :9.000
##   temp_change      RainTomorrow
##  Min.   :-12.600   0:110316
##  1st Qu.:  2.100   1: 31877
##  Median :  4.400
##  Mean   :  4.764
##  3rd Qu.:  7.200
##  Max.   : 23.000
```

```
# find how many rows that 90% of the data is, 10% of the data
set.seed(1234)
n <- nrow(completed_weather)
ntrain <- round((n*.90),0)
ntest <-  round((n*.10),0)

# random sample the data to get the train and test data.
i_train <- sample(1:n, ntrain, replace=FALSE)
train_set <- completed_weather[i_train, ]
test_set <- completed_weather[-i_train, ]
```

Now that I have the train and test set, I can begin the modeling with k-nearest neighbors.

```
# run knn in sapply to see which k-value maximizes the accuracy
accuracies <- sapply(seq(5, 15, 1), function(k){
    knn_fit <- knn3(RainTomorrow ~ ., data = train_set, k = k)
    y_hat_knn <- predict(knn_fit, test_set, type = "class")
    confusionMatrix(data = y_hat_knn, reference = test_set$RainTomorrow)$overall["Accura
cy"]
})

# get value of k that produced highest accuracy and re-run knn to produce full confusion
matrix
k <- which.max(accuracies)+4

# re-run knn to produce full confusion matrix
knn_fit <- knn3(RainTomorrow ~ ., data = train_set, k = k)
y_hat_knn <- predict(knn_fit, test_set, type = "class")
confusionMatrix(data = y_hat_knn, reference = test_set$RainTomorrow)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10428  1652
##          1   634  1505
##
##                Accuracy : 0.8392
##                  95% CI : (0.8331, 0.8452)
##     No Information Rate : 0.778
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.474
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9427
##             Specificity : 0.4767
##          Pos Pred Value : 0.8632
##          Neg Pred Value : 0.7036
##              Prevalence : 0.7780
##          Detection Rate : 0.7334
##    Detection Prevalence : 0.8496
##       Balanced Accuracy : 0.7097
##
##        'Positive' Class : 0
##
```

```
knn_conf <- confusionMatrix(data = y_hat_knn, reference = test_set$RainTomorrow)

# create results model comparison dataframe
model_results <- data_frame(method = "K-Nearest Neighbors", Accuracy = knn_conf$overall[
"Accuracy"], Specificity = knn_conf$byClass["Specificity"])
```

```
## Warning: `data_frame()` is deprecated, use `tibble()`.
## This warning is displayed once per session.
```

Now that we have run k-nearest neighbors, let's examine how accurate our model is. Our final accuracy used a k-value of 13, which produced an accuracy of 9, which just by itself, looks to be ok. But as I mentioned above, let's look at our Specificity, which I believe is going to be our real metric of how well our model is doing.

Notice that the Sensitivity is high, but the Specificity is quite low, only at 50%. So what is the specificity? The specificity is our ability to correctly guess the positive. In other words, when the data says that it is going to rain tomorrow, how well are we doing at correctly predicting yes?

To get a better understanding of specificity, let's look at a different way to look at the confusion matrix. For each prediction, there are 4 possible outcomes:
True Positive: We predicted it would rain tomorrow when it actually rained tomorrow.
False Positive: We predicted it would rain tomorrow when it actually did not rain tomorrow.
False Negative: We predicted it would not rain tomorrow when it actually rained tomorrow.
True Negative: We predicted it would not rain tomorrow when it actually did not rain tomorrow.

To put it a little more clearly, a false positive means that the positive we predicted was false, and the actual result was negative. False negative means the negative we predicted was false, and the actual result was positive. Let's look at it in matrix form.

```
##                       Predicted Rain Predicted No Rain
## Actually Rained                 1505              1652
## Actually Didn't Rain             634             10428
```

Here we can get a better understanding really of how our model is really doing, and why the specificity is so low. We see that when it actually rained tomorrow, so the RainTomorrow variable = "Yes", that of those 3157 times, we correctly predicted it would rain only about half the time. This is how we get the specificity from these numbers:

$$\text{Specificity } = \frac{\text{True Positive}}{\text{True Positive } + \text{False Negative}}$$

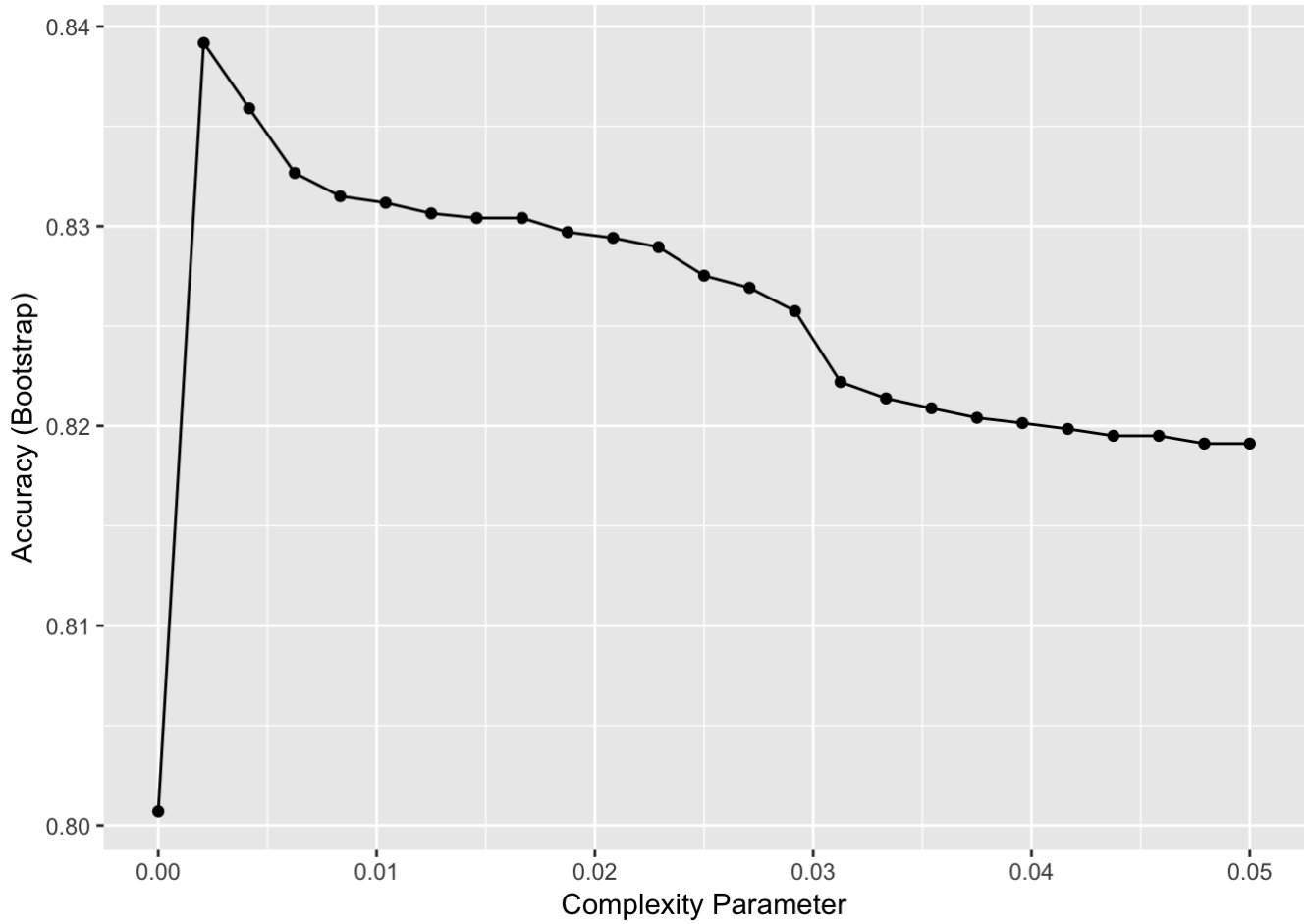
Following this formula, this is how we get our specificity that was reported in the original confusion matrix that came out of knn. Plugging in the numbers, we get 0.4767184, which is exactly the specificity reported earlier. We can see why the Sensitivity is so high too. Of the times that it did not rain tomorrow, which was 11062, we correctly guessed it would not rain 10428 times.

So while we have an overall accuracy in the mid-80's, this is how well our model really does: when it is actually going to rain tomorrow, our model will only correctly guess it 50% of the time. Which, in the end, is not all that valuable of a model.

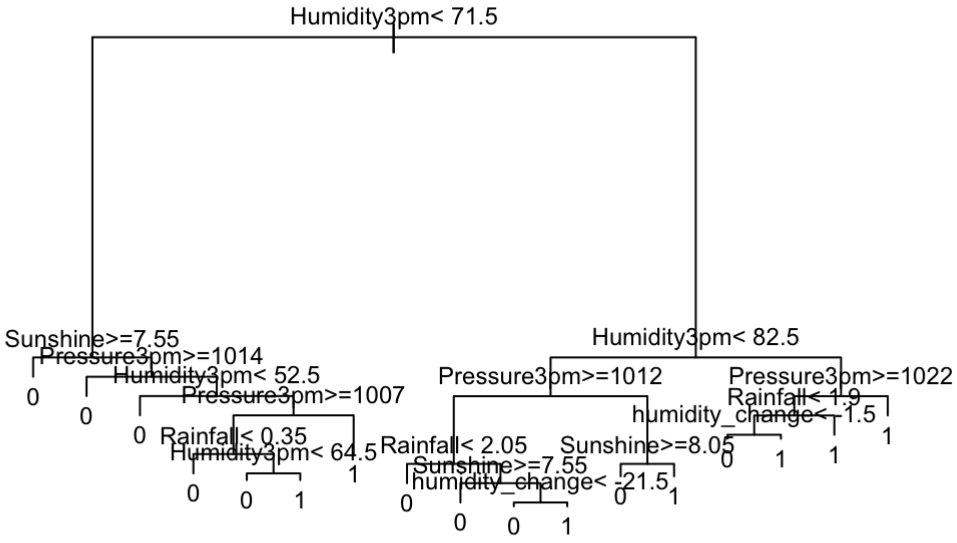
I think a lot of this comes from the fact that in my data exploration, I didn't find anything that really showed a big indicator of whether it would rain or not. When I grouped those variables into high, medium, or low, the best indicators of rain tomorrow were only in the mid 50's, low 60's, which I think is why I'm not seeing accuracy (specificity) results above that.

With that in mind, it is worth checking out a different modeling algorithm to see if we can't get that specificity up higher. So I will move on to try out decision trees and random forests, and hopefully that might return some better results.

```
# run CART
train_rpart <- train(RainTomorrow ~ .,
                     method = "rpart",
                     tuneGrid = data.frame(cp = seq(0, 0.05, len = 25)),
                     data = train_set)

# we can see from this plot what complexity parameter provides the highest accuracy, fou
nd through cross-validation
ggplot(train_rpart)
```




```
# plot decision tree visualization
plot(train_rpart$finalModel, margin = 0.1)
text(train_rpart$finalModel, cex = 0.75)
```



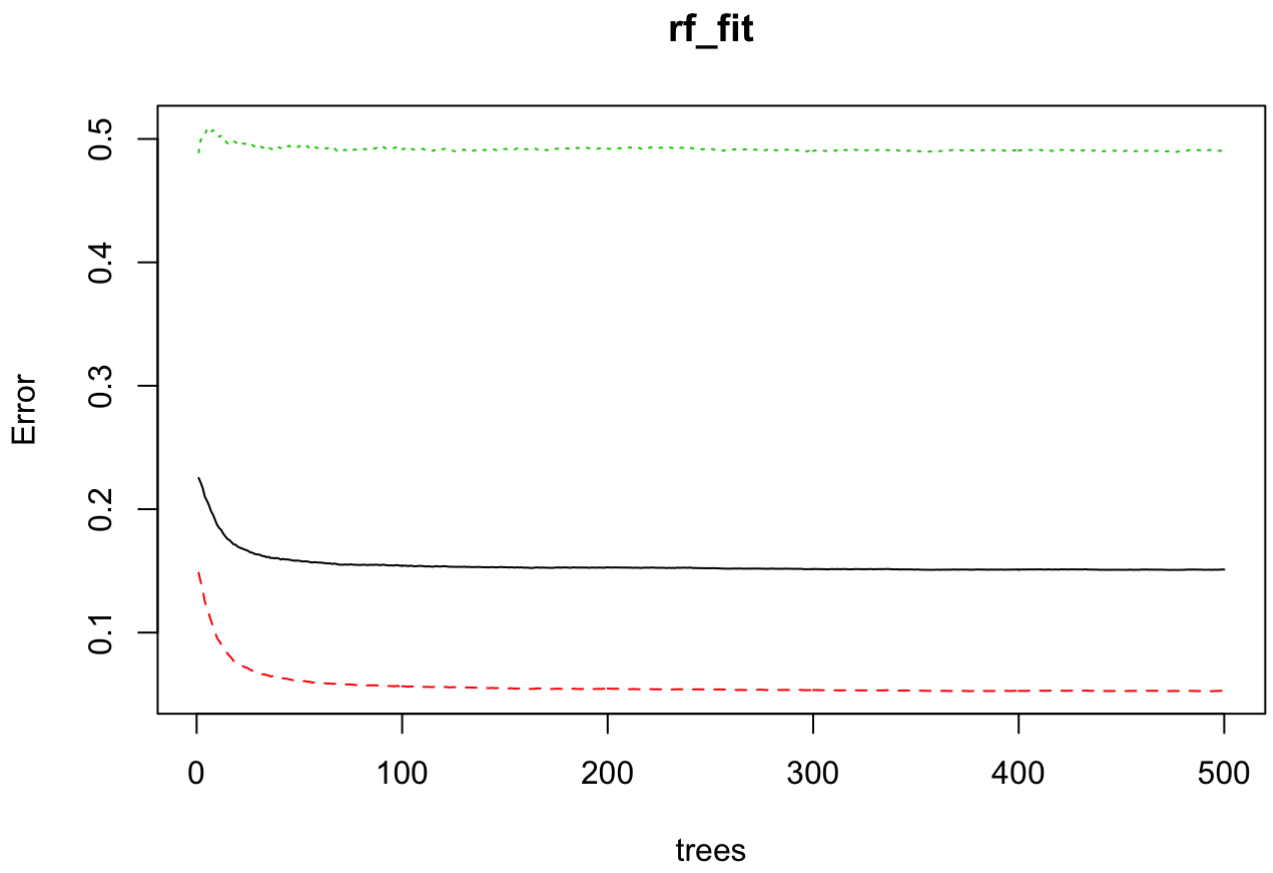

```
# check confusion matrix for accuracy
confusionMatrix(predict(train_rpart, test_set), test_set$RainTomorrow)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10535  1733
##          1   527  1424
##
##                Accuracy : 0.8411
##                  95% CI : (0.8349, 0.847)
##     No Information Rate : 0.778
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4672
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9524
##             Specificity : 0.4511
##          Pos Pred Value : 0.8587
##          Neg Pred Value : 0.7299
##              Prevalence : 0.7780
##          Detection Rate : 0.7409
##    Detection Prevalence : 0.8628
##       Balanced Accuracy : 0.7017
##
##        'Positive' Class : 0
##
```

```
cart_conf <- confusionMatrix(predict(train_rpart, test_set), test_set$RainTomorrow)


# add CART results to results dataframe
model_results <- bind_rows(model_results,
                           data_frame(method="CART",
                                      Accuracy = cart_conf$overall["Accuracy"],
                                      Specificity = cart_conf$byClass["Specificity"]))
```

We achieved about the same accuracy and specificity when using decision trees. Let's check random forests to see if that does any better.

```
# try raindom forest on model 2
rf_fit <- randomForest(RainTomorrow ~ ., data = train_set)

# this plot shows the error versus the number of trees. After about 50 trees, the error
 does not change much.
plot(rf_fit)
```




```
# check accuracy
confusionMatrix(predict(rf_fit, test_set), test_set$RainTomorrow)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction     0     1
##          0 10502  1561
##          1   560  1596
##
##                Accuracy : 0.8508
##                  95% CI : (0.8449, 0.8567)
##     No Information Rate : 0.778
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.513
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.9494
##             Specificity : 0.5055
##          Pos Pred Value : 0.8706
##          Neg Pred Value : 0.7403
##              Prevalence : 0.7780
##          Detection Rate : 0.7386
##    Detection Prevalence : 0.8484
##       Balanced Accuracy : 0.7275
##
##        'Positive' Class : 0
##
```

```
rf_conf <- confusionMatrix(predict(rf_fit, test_set), test_set$RainTomorrow)

model_results <- bind_rows(model_results,
                           data_frame(method="Random Forest",
                                      Accuracy = rf_conf$overall["Accuracy"],
                                      Specificity = rf_conf$byClass["Specificity"]))
```

Once again, we are still stuck at about 50% specificity. It does not look like we will get much more accurate than that.

Side note: I did try multiple variations of the dataset in the different machine learning algorithms I have used in this presentation. I ran through multiple iterations of knn and random forests with different data models, dropping and adding variables, but no matter what I tried I never achieved a significant rise in specificity.

Results

For the final results, the best performing model was the random forest, but not by much. The three models I tried all produced very similar results. Here are the results in a table format.

```
## # A tibble: 3 x 3
##   method              Accuracy Specificity
##   <chr>                  <dbl>       <dbl>
## 1 K-Nearest Neighbors    0.839       0.477
## 2 CART                   0.841       0.451
## 3 Random Forest          0.851       0.506
```

So we see that the Random Forest algorithm produced the highest specificity.

Conclusion

In the end, this is not a very good, reliable model for predicting whether it will rain tomorrow in Australia. Knowing that it will rain tomorrow, the model could do no better than a 50-50 guess at predicting correctly. If this was the weather forecasting model that your country used, you would not be able to trust it very much. If the model predicted no rain tomorrow, about 9 times out of 10, you would get no rain tomorrow. But if the model predicted rain tomorrow, that really leaves you with no indication one way or the other if it actually will rain tomorrow.

Obviously, current weather models are able to predict weather much more accurately than I am able to do here. I tried looking online for some hints at what data they use or what forecasting models they use, but I wasn't able to find anything. So clearly there are better ways to do this, and the methods and data that I was trying was not very helpful at all.