ECS 132 Spring 2019 : Term Project

Hoseung Lee (914001975, hoslee), Yi Lian (998370114, yilian)

December 13, 2019

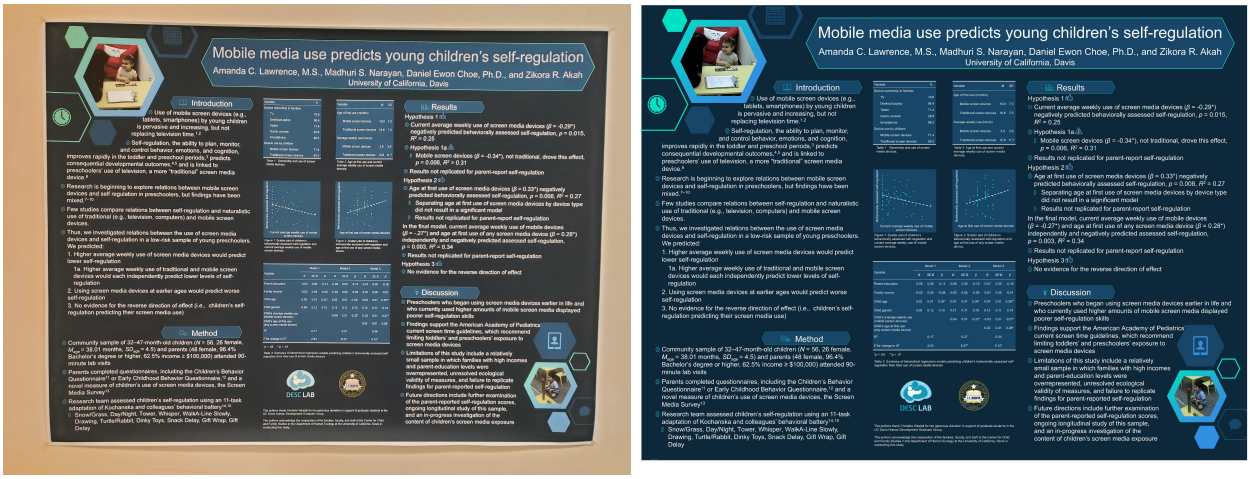
Contents

1 Problem A

For problem A of the Term Project, students are required to each choose a poster that mentioned p-values or any equivalent. We went to Hart Hall to collect a variety of posters that we deemed were appropriate for Problem A. The following two subsections are on the posters we favored the most.

The subsections will primarily be addressing the existence of p-values in our respective posters. For a significant amount of time, p-values were used in scientific reports as a measure of significance. However, experts are now debating how p-values are being badly misused. This has led to the American Statistical Association releasing a statement (https://www.amstat.org/asa/News/ASA-P-Value-Statement-Viewed-150000-Times.aspx) on how "Well-reasoned statistical arguments contain much more than the value of a single number and whether that number exceeds an arbitrary threshold. The ASA statement is intended to steer research into a 'post $p < 0.05$ era.". We will attempt to address this concern with our respective analysis on our posters.

1.1 Hoseung Lee (914001975, hoslee)



(a) Hoseung's Poster Choice at Hart Hall

(b) Online Version for Visual Clarity

Figure 1: "Mobile media use predicts young children's self-regulation"

The poster listed in the figure above (a) was found in Hart Hall on November 27, 2019. This is one of the 7 posters that Yi and I discovered in Hart Hall. This particular research poster was done under the UC Davis Human Development Graduate Group. This research discusses the impact of Mobile Media usage on children's ability to self-regulate (take care of themselves). The researchgate site for this poster can be found here: https://www.researchgate.net/publication/336196787_Mobile_media_use_predicts_young_children's_self-regulation

A sample of 56 32-47-month-old children were tested for self-regulation using a 11-task adaptation of a behavioral test. The age of first screen media device exposure and the average weekly use of mobile screen devices were also measured.

The poster has three hypotheses that are tested:

1. Higher average weekly use of screen media devices would predict lower self-regulation
2. Using screen media devices at earlier ages would predict worse self-regulation
3. No evidence for the reverse direction of effect (i.e., children's self-regulation predicting their screen media use)

I will only be discussing hypotheses 1 and 2 as these are the primary hypotheses.

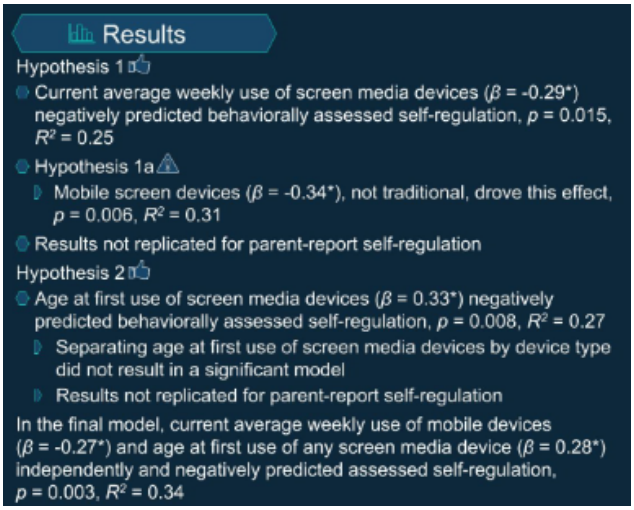


Figure 2: Results of Experiment

The hypotheses that "Higher average weekly use of screen media devices would predict lower self-regulation" and "Using screen media devices at earlier ages would predict worse self-regulation" were both accepted by this poster. The first hypothesis had a p-value of 0.015 and the second hypothesis had a p-value of 0.008. According to the Significance test method that was used here, the negative effect that screen media device usage has is "very significant" since the p-values for both statistics are significantly less than 0.05, the standard α value. However, this number is very misleading if you look at the other data available in the poster.

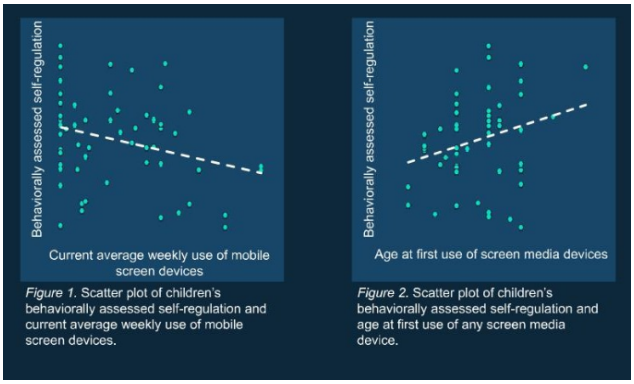
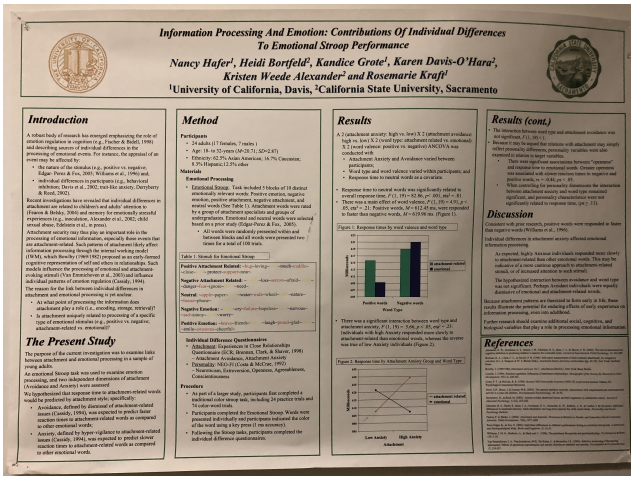


Figure 3: Graph of Experiment Data

Right next to the p-value of the different hypotheses is the R^2 value of the hypotheses. The R^2 is essentially a measure of how close the data is to the regression line. The higher the R^2 (maximum value of 1) the better the model fits the data. The R^2 of hypotheses 1 and 2 are 0.25 and 0.31 respectively. This is a very poor fit as can be seen visually in the data. It can be easily seen that the data points in both Figures for the two experiments are all over the place. Without the regression lines that are visible in the graphics, it wouldn't be very easy to find an obvious trend. Of course, this contrasts greatly with the p-value of these two experiments which suggest that the media device usage had a significant effect on children's self-regulation.

Replacing the p-value with a confidence interval for this research poster would greatly increase the clarity of the poster's findings. For instance, lets say that the confidence interval indicated that there is a 95% chance that there is an extraordinarily small negative effect on children's self-regulation when they are exposed to mobile media. If this is the case, one can't really come to the conclusion that their hypothesis "children's exposure to mobile media has a negative impact on self-regulation" is significantly correct. This is the type of information that a single p-value lacks. Such nature of the p-value and the low R^2 values that are in the model make the conclusion that "Preschoolers who began using screen media devices earlier in life and who currently used higher amounts of mobile screen media displayed poorer self-regulation skills" a questionable claim.

1.2 Yi Lian (998370114, yilian)



(a) Yi's Poster Choice at Hart Hall

Figure 4: "Information processing and emotion: Contributions of Individual difference To Emotional stroop Performance"

The poster listed in the figure above (a) was found in Hart Hall on November 27, 2019. This is one of the 7 posters that Yi and I discovered in Hart Hall. This particular research poster was done under the UC Davis Human Development Graduate Group. This research discusses the individual influence of attachments and emotional process. The researchers want to figure out the connection between attachment and emotional processing.

A sample of 24 adults including 17 females and 7 males aged 18 to 32 years old. The mean is 20.71 and the standard deviation is 2.87. The sample is investigated using a questionnaire to examine the individual level of attachment (attachment avoidance and attachment anxiety) and personality (neuroticism, extroversion, openness, agreeableness, conscientiousness) followed by a stroop task and the respond time was recorded and further analyzed.

The poster provides two hypothesis. I will call them hypothesis 1 and 2:

1. Anxiety, defined by hyper-vigilance to attachment-related issue, was expected to predict slower reaction times to attachment-related words as compared to other emotional words.
2. Avoidance, defined by dismissal of attachment-related issue, was expected to predict faster reaction times to attachment-related words as compared to other emotional words.
3. Although the study did not set a null hypothesis, it is implied that the null hypothesis for both hypothesis 1 and 2 suggests that being attachment avoidance or attachment anxiety do not have an impact on the respond time of attachment-related words compared to other emotional words.

It worth to notice that besides those to hypothesis that the researchers wants to analyse, there are also other hypothesis suggested and examined in poster. I will call them sub-hypothesis 1 and 2

1. The respond time to positive words will be smaller than that to the negative words.
2. The implied null hypothesis is that there is no significant relationship between respond time of the word and the positiveness/negativeness of that word.

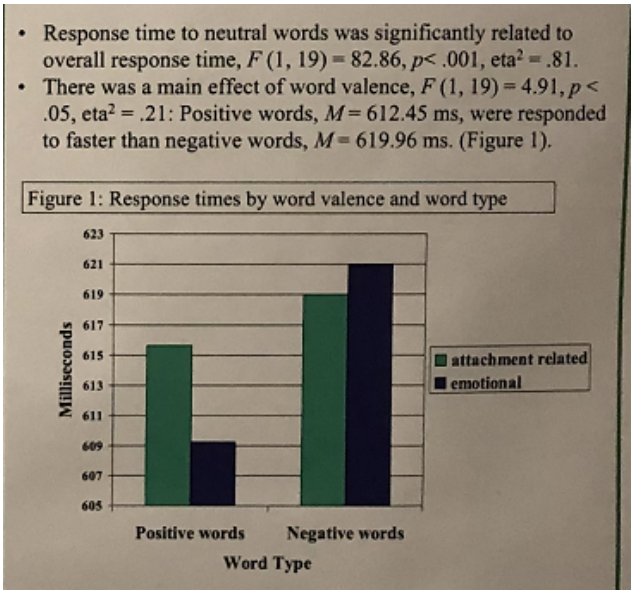


Figure 5: "Results of Experiment regarding sub-hypothesis"

Regarding the sub-hypothesis, the p value is less than 0.05. That means we are more likely to reject the null sub-hypothesis that "There is no significant relationship between respond time of the word and the word valence" at a significant level of 0.05.

In addition, the hypothesis the researcher set is directed. The interpretation of p value only gives a binary conclusion(Yes or No) about the null hypothesis, which do not evaluates the direction and the strength. If the study is trying to claim that "The respond time to a positive word will be smaller than the respond time to a negative word", then confidence interval should also be analyzed to draw the conclusion that "There was a main effect (the respond time) on word valence".

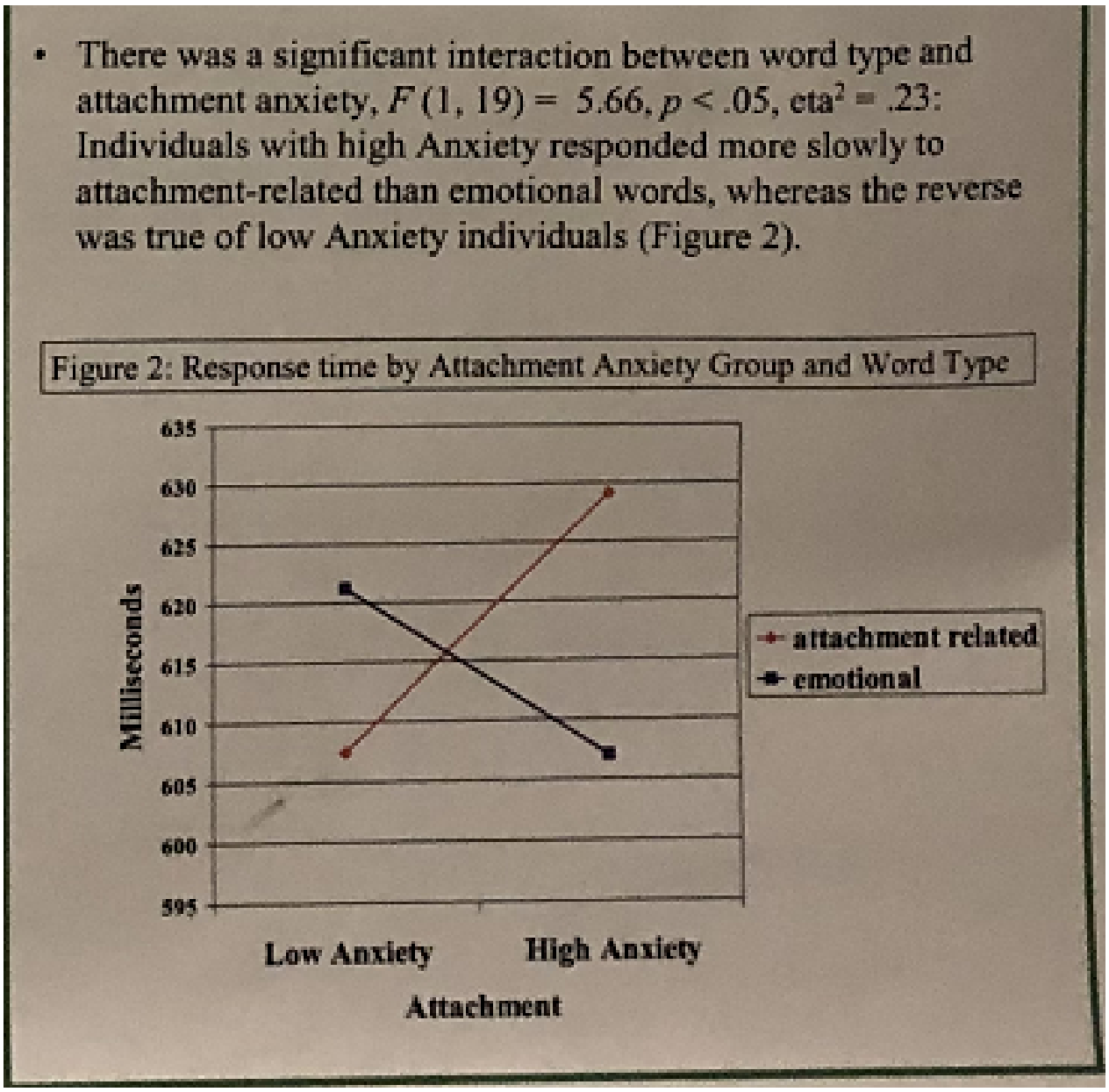


Figure 6: "Results of Experiment regarding hypothesis1"

Regarding the hypothesis 1, the p value is less than 0.05. That means we are more likely to reject the null hypothesis that "Being attachment anxiety do not have an impact on the respond time of attachment-related words compared to other emotional words."

The experiment do show the trend that individuals with high Anxiety responded more slowly to attachment-related than emotional words, whereas the reserve was true of low Anxiety individuals. However, this evidence is not enough for the study to draw the conclusion that "Attachment anxiety people will have will a less respond time to attachment-related words as compared to other emotional words". If the study want to figure out the strength and the direction of a hypothesis, confidence interval should be introduced.

The author seems lack of a comprehensive understanding about the difference between p value and confidence interval. P value mainly helped with decision making. That is to reject or accept the null hypothesis, which is a binary process without the ability to evaluate the direction (which hypothesis is stronger) and the strength (how confident can we trust the hypothesis). For this poster, if the authors to study the strength and direction, confidence interval should be introduced instead.

2 Problem B

2.1 Problem B.1 - KNN Method

2.1.1 Problem B.1 - KNN Method; Step 1 (Casual Bikers)

Just to test out the efficacy of the KNN Model in weather prediction, I started by trying to predict the weather just by using the number of casual bike users in the dataset. This was deemed to be the most important single data column because there was an assumption that there would be less casual passengers on days that have less favorable weather conditions. The weathersit variable is already approximately ordered from favorable to less favorable weather:

1. Clear, Few Clouds, Partly Cloudly
2. Mist + Cloudy, Mist + Broken Clouds, Mist + Few Clouds, Mist
3. Light Snow, Light Rain + Thunderstorm + Scattered Clouds, Light Rain + Scattered Clouds
4. Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

Theoretically, higher casual passenger numbers should correlate to lower weathersit numbers. Since we are assuming people will bike more on days that have good weather conditions.

It may be strange that we didn't just use the total passengers count instead of JUST the casual passenger count, but there is good reasoning for this. We assume registered passengers are people that registered to the service because they have a good reason to consistently use a bike (for a job or such). So unlike a casual passenger, registered passengers would generally still be forced to bike regardless of the severity of the weather conditions. Ultimately, we assumed registered passengers wouldn't be as useful in predicting the weather.

I trained my first KNN-model on the first 631 data points in the day1 file. The remaining 100 was used as a the testing set. I only used the casual passenger numbers in my KNN-model (along with base K-value of 5, these same setting where used for all of the following models unless mentioned). With this, I was able to get a 53% accuracy rate.

```
> BKNNStep1()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.53
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 7: "KNNStep1 Results"

However, as can be seen in the image above, if a person just guessed that everyday would be of weathersit type "1" (Which is the most common weather type in this given data set), they would be correct 54% of the time, which is slightly better than the model we have created! Casual passenger count alone clearly isn't enough to create a good weather prediction model.

2.1.2 Problem B.1 - KNN Method; Step 2 (Temperature)

We know that the temperature effects rain and cloud formation and that rain/snow can also effect the temperature. There are also obvious correlations between the weather type and temperature such as snow formation. If the temperature is below freezing, we know that we will experience snow rather than rain. While the day1 dataset rarely contains such extreme cold temperatures, it still serves as an example of how the temperature can effect the weather type. More about the impacts of temperature on the weather type can be seen in this article here: https://www.thoughtco.com/what-determines-rain-temperature-3443616. This is why I decided to use temperature (only) as the parameter for developing my KNN-Model for step 2. Everything was identical with Step 1, except the data extraction for temperature. The temperature data in step 2 is located in the 10-th column of the day1 data matrix.

```
> BKNNStep2()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.54
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 8: "KNNStep2 Results"

Unfortunately, as can be seen in the results above, the KNN-model based off of the temperature data barely outperformed the casual passenger based KNN-model. This time, the KNN-model was tied with the "model" where you simply assume every weather type is going to be of type 1, at 54%.

2.1.3 Problem B.1 - KNN Method; Step 3 (Casual Passengers + Temperature)

In hopes of improving the classification percentage, I decided to use both the Casual Passengers data and the Temperature data. Unfortunately, the classification percentage was even worse, at a 49% classification rate as opposed to the 53% and 54% that the casual passengers-KNN and the temperature-KNN had respectively. This is potentially due to the small data set we have for this assignment. There are only 731 data sets overall in day1 and only 631 for the training set, this small data set may cause randomness that impacts the accuracy of our classification. There also is the possibility that the pairing of only casual passengers and temperature inherently make the classification worse.

```
> BKNNStep3()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.49
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 9: "KNNStep3 Results"

2.1.4 Problem B.1 - KNN Method; Step 4 (Day)

After basing my KNN model off of the Casual Passenger count and my Temperature data, I only got sub optimal results. (With every single iteration of these KNN-models doing worse than simply assuming that the weather on a given day is going to be type 1) At this point, I assumed that the Casual Passenger count and the temperature data poorly predicted weather type.

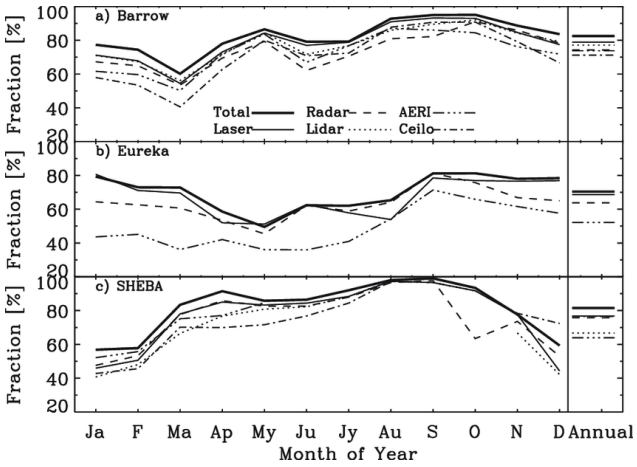


Figure 10: Cloud Cycles in the Arctic Results https://www.researchgate.net/figure/Annual-cycles-of-monthly-mean-cloud-occurrence-fraction-at-the-a-Barrow-b-Eureka_fig4_252812195

I determined that the day (out of the 365 possible days) would be a good measure for weather types. As can be seen in the figure, there are yearly cloud cycles that remain somewhat consistent year after year. Of course, the cloud cycles in Virginia (Which is primarily where the cycling data was collected) may not be as stable as the cloud cycle that are illustrated in the figure above. However, it still does illustrate a point that cloud formation has a yearly cycle. In addition, certain days of the year are more hotter/colder than others. Theoretically, the day of the year should be able to capture both (and maybe even more) of these weather predicting cycles and be able to accurately predict the weather for any given day.

I first created a new column in the day1 dataset called "Dayof365". For every single row of the dataset, I converted the dteday to the day of the year. For instance, if the dteday in this row was "2011-01-09", the corresponding day value would be "9". However, this is when I realized there was a problem. If the days are categorized from 1 to 365, the model would think day "1" should behave radically differently from day "365". This is clearly incorrect because after day "365", the day counter loops back to "1". This means that "365" and "1" should have very similar weather behaviors. There were two possible solutions to this problem:

- Just get the day value and subtract it from "180", the magnitude of this value would replace our day values and it would alleviate the problem that we experience by simply using a 1 to 365 scale.
- Imagine a round table where the numbers 1 to 365 are laid clockwise around the table in order. In this format, the distance between the days would be shown appropriately.

I wasn't sure how to implement the second solution, so I went with the first possible solution which was relatively simple to implement.

```
for (i in 1:731) {
  if (day == 366) {
    day <- 1
    #reset 1 after you hit 366
  }
  day1Mod[i,17] = day
  day1Mod[i,17] = abs(day1Mod[i,17] - 180)
  day <- day + 1
}
```

Figure 11: Solution to day numbering problem

Unfortunately, the results for the day-based KNN model was also mediocre, with a 46% classification rate. However, this doesn't mean the solution to the 1 to 365 day scale problem was unsuccessful. When the KNN-model was implemented without the day scale fix, the classification rate was 45%, slightly worse than after the fix was applied.

```
> BKNNStep4()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.46
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 12: Weather classification results with the day scale fix applied

```
> BKNNStep4()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.45
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 13: Weather classification results without the day scale fix applied

Of course, while the day scale problem was alleviated, the classification rate was still way too low. I had to look for other KNN-models.

2.1.5 Problem B.1 - KNN Method; Step 5 (Temperature, Humidity, Windspeed, Casual Passengers)

After discussion with my groupmate, I decided to use several different predictors (rather than the one or two predictors I used before). I decided that the Temperature, Humidity, Windspeed, and the number of Casual Passengers were the top four important predictors so I created a KNN-model based off of these four predictors. This model was very successful with a 74% classification rate, it beat out all of the models that I had created before by margins of almost 20%

```
> BKNNStep5()
[1] "The correct percentage (100 data points tests) is: "
[1] 0.78
[1] "The correct percentage, by just assuming the weather will be category 1 (100 data points tests) is: "
[1] 0.54
```

Figure 14: Weather Classification results with Temperature, Humidity, Windspeed, Casual Passenger Predictors. k-value was adjusted to 21 from the standard 5

Because of the success of this model, I decided to finally adjust the k-value of the model. After adjusting the k-value to get the maximum classification success rate, I achieved a 78% prediction rate after changing the k-value to 21 from the standard k-value of 5 that was used in the prior models. This was another 4% boost to the classification success rate. It is very important to find an ideal k-value because a low k-value means that your model will be prone to noise. A k-value that is too high will simply make your classification closer to the mean of the entire dataset, which will also lower classification accuracy.

2.2 Problem B.2 - Linear Model Method

The idea of linear regression model is to treat the predictors as a variable in a linear function $f(x) = ax_1 + b \cdot x_2 + ...$, where $x_1, x_2...$ are our predictors, and our goal is to correctly determine which factors provided have a significant impact on the result, and the coefficient of that factor.

The data set *day*1 from package regtools will be the data set we use here. It contains the information about a given day's date, season, month, weekday, temperature, feeling temperature humidity, wind speed, and so on. The weather is indicated by index 1-4 in the weathersit column, where:

- Clear, Few Clouds, Partly Cloudly
- Mist + Cloudy, Mist + Broken Clouds, Mist + Few Clouds, Mist
- Light Snow, Light Rain + Thunderstorm + Scattered Clouds, Light Rain + Scattered Clouds
- Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog

To begin with, let's first take a look at the general graph generated by

```
plot(day1)
```

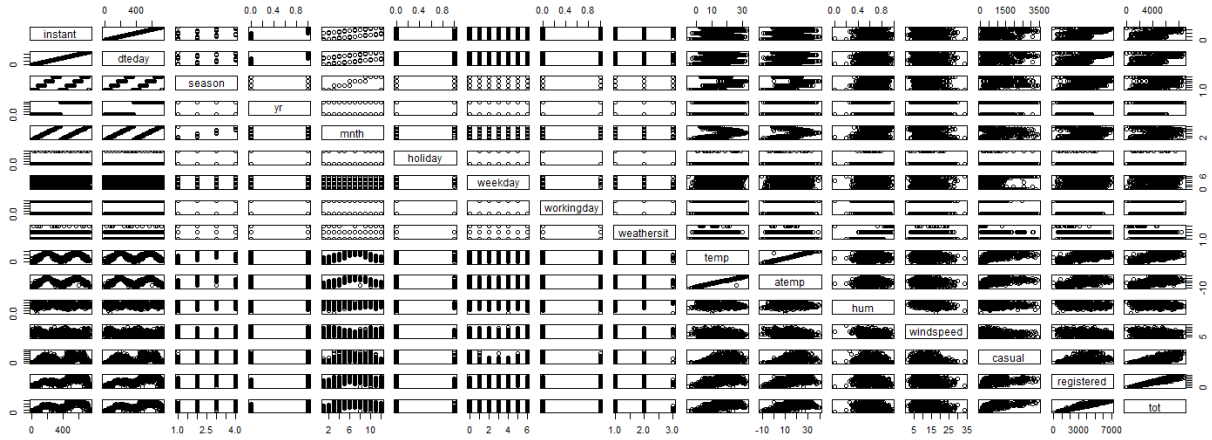


Figure 15: "16 by 16 pair-wise exclusive plot of data-set day1"

Here we want to have a general idea about the correlations between two any columns in day1. As we can see, the graph generated by data in column temp and atemp(feeling temperature) is nearly a straight line, that means they are more likely to have the same influence towards our dummy variable weathersit(weather). Another "almost straight line" relationship could be found between registered and tot, which is the registered bike rental and total bike rental of a given day. So when we try to build a linear model, variables with a high correlation should only be considered once in most of the cases.

2.2.1 Predict weather by temperature

Intuitively, when talking about weather, the first thing comes to my mind is the temperature. So, let's start our model by first consider the relationship between weather and temperature. We will first generate our mode by

```
weather_by_temp = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)]))
```

We get a data-set called *weather_by_temp* that stores the information of our first linear model generate based on the first 631 entries from the column weathersit and temp. Let's take a look at it

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	1.5008	0.0444	33.83	0.0000
day1$temp[c(1:631)]	-0.0075	0.0024	-3.10	0.0020

We now have our first linear function that $weathersit = 1.500839 - 0.007539 \cdot temp$

Notice that the P value for both estimate is pretty small, which means both the intercept and the temperature will have a significant influence in the weather.

We will then apply this model to the last 100 days to figure out the accuracy.

```
test_weather_temp = 1.500839 - 0.007539 * day1$temp[c(632:731)]
diff_temp = test_weather_temp - day1$weathersit[c(632:731)]
diff_temp_round=round(diff_temp)
sum(diff_temp_round == 0)
```

What we are doing here is to apply the last 100 entries of data in the temp column to this model, which gives a vector with 100 elements representing our estimation. We then round them to the nearest integer, subtract the answer and see how many 0's are inside. Each 0 represents a correct prediction. For this model, 54 out of 100 predictions is correct. Our prediction is half accurate, but we are not satisfied.

2.2.2 Predict weather by temperature and humidity

Now let's introduce a new factor, humidity. We know that higher the humidity, the greater probability that it would rain. So this should also have some impact on weather. Let's first generate the model by

```
weather_by_temp_hum = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
    day1$hum[c(1:631)]))
```

We get a data-set called *weather_by_temp_hum* that stores the information of our second linear model generated based on the first 631 entries from the column weathersit, temp and humidity. Here is the detailed information:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.1552	0.0784	1.98	0.0483
day1$temp[c(1:631)]	-0.0124	0.0019	-6.37	0.0000
day1$hum[c(1:631)]	2.2753	0.1185	19.21	0.0000

The linear function generated is is $weathersit = 0.155168 - 0.012417 \cdot temp + 2.275260 \cdot hum$

The P value for each estimated coefficient is relative low, which means this model is reasonable. Let's apply this model to the last 100 entries of data-set

```
test_weather_temp_hum = 0.155168 - 0.012417 * day1$temp[c(632:731)] + 2.275260 *
    day1$hum[c(632:731)]
diff_temp_hum = test_weather_temp_hum - day1$weathersit[c(632:731)]
diff_temp_hum_round = round(diff_temp_hum)
sum(diff_temp_hum_round == 0)
```

This model gives 74 correct prediction out of 100, which is a big progress. But as we know weather is a very complex model that can influenced by almost everything. We will introduce more predictors in the following sections.

2.2.3 Predict weather by temperature, humidity and interaction term

From the previous temperature-humidity model, we are assuming that the effect of temperature on weather is same at all humidity levels. But is it true in real life? Let's generate a new model studying those relationships.

```
weather_by_temp_hum_new = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] *
    day1$hum[c(1:631)]))
```

We get a data-set called *weather_by_temp_hum* that stores the information of our second linear model generated based on the first 631 entries from the column weathersit, temp, humidity and interaction of temperature and humidity. Here is the detailed information:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.1087	0.1464	0.74	0.4578
day1$temp[c(1:631)]	-0.0090	0.0092	-0.99	0.3245
day1$hum[c(1:631)]	2.3555	0.2442	9.65	0.0000
day1$temp[c(1:631)]:day1$hum[c(1:631)]	-0.0057	0.0151	-0.38	0.7072

We now have a new model that $weathersit = 0.108737 - 0.009048 \cdot temp + 2.355487 \cdot hum + -0.005668 \cdot temp \cdot hum$

The P value for this model is unacceptably high, so we are not expecting a good prediction based on this model. Let's look at its performance

```
 test_weather_temp_hum_new = 0.108737 - 0.009048 * day1$temp[c(632:731)] + 2.355487 *
     day1$hum[c(632:731)] - 0.005668 * day1$hum[c(632:731)] * day1$temp[c(632:731)]
diff_temp_hum_new = test_weather_temp_hum_new - day1$weathersit[c(632:731)]
diff_temp_hum_new_round = round(diff_temp_hum_new)
sum(diff_temp_hum_new_round == 0)
```

This model gives 74 correct predictions among 100 total predictions. The result is the same as the model which interaction is not introduced. However, the P value is too high. This model is not the best choice.

2.2.4 Predict weather by temperature, humidity and wind speed

Wind speed will also affect certain day's weather. We will therefore generate a new model to evaluate its impact.

```
weather_by_temp_hum_wind = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
    day1$hum[c(1:631)] + day1$windspeed[c(1:631)]))
```

The information of the model is:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1701	0.1010	-1.68	0.0926
day1$temp[c(1:631)]	-0.0110	0.0019	-5.67	0.0000
day1$hum[c(1:631)]	2.4088	0.1193	20.18	0.0000
day1$windspeed[c(1:631)]	0.0170	0.0034	4.98	0.0000

The function generated is $weathersit = -0.170059 - 0.010964 \cdot temp + 2.408763 \cdot humidity + 0.017007 \cdot windspeed$

Notice that the P value are all relatively small, so this model has the potential to perform well.

We will apply this model to the data-set to see its performance:

```
test_weather_temp_hum_wind = -0.170059 - 0.010964 * day1$temp[c(632:731)] + 2.408763 *
    day1$hum[c(632:731)] + 0.017007 * day1$windspeed[c(632:731)]
diff_temp_hum_wind = test_weather_temp_hum_wind - day1$weathersit[c(632:731)]
diff_temp_hum_wind_round = round(diff_temp_hum_wind)
sum(diff_temp_hum_wind_round == 0)
```

It turns out that this model gives 77 correct prediction among the 100 outcomes, which is the best model we get so far.

2.2.5 Predict weather by temperature, humidity, wind speed and exclusive interaction term

Till now we have linked the relationship between weather and our independent variables wind speed, humidity and temperature. So what can we tell about their interactions? Intuitively, humidity's influence on weather under low temperature and wind speed is different compare with the humidity under high temperature and wind speed. So let's try introduce their interaction exclusively.

```
weather_by_temp_hum_wind_new = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] *
    day1$hum[c(1:631)] * day1$windspeed[c(1:631)]))
```

The information of the model is:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	0.4445	0.4185	1.06	0.2886
day1$temp[c(1:631)]	-0.0342	0.0261	-1.31	0.1904
day1$hum[c(1:631)]	1.1734	0.6677	1.76	0.0793
day1$windspeed[c(1:631)]	-0.0340	0.0271	-1.25	0.2101
day1$temp[c(1:631)]:day1$hum[c(1:631)]	0.0536	0.0416	1.29	0.1972
day1$temp[c(1:631)]:day1$windspeed[c(1:631)]	0.0022	0.0018	1.21	0.2263
day1$hum[c(1:631)]:day1$windspeed[c(1:631)]	0.1045	0.0448	2.33	0.0201
day1$temp[c(1:631)]:day1$hum[c(1:631)]:day1$windspeed[c(1:631)]	-0.0050	0.0030	-1.68	0.0927

In general, the P value for this model is high, so this is not a safe prediction. The fucntion generated is $weather = 0.444522 \cdot -0.034163 \cdot temp + 1.173436 \cdot hum - 0.033980 * windspeed + 0.053640 \cdot temp \cdot hum + 0.002233 \cdot temp \cdot windspeed + 0.104494 \cdot hum \cdot windspeed - 0.005036 \cdot temp \cdot hum \cdot windspeed$

Let's apply this function to our data-set and see the result:

```
test_weather_temp_hum_wind_new = 0.444522 - 0.034163 * day1$temp[c(632:731)] + 1.173436 *
    day1$hum[c(632:731)] - 0.033980 * day1$windspeed[c(632:731)] + 0.053640 *
    day1$temp[c(632:731)] * day1$hum[c(632:731)] + 0.002233 * day1$temp[c(632:731)] *
    day1$windspeed[c(632:731)] + 0.0104494 * day1$hum[c(632:731)] * day1$windspeed[c(632:731)] -
    0.005036 * day1$temp[c(632:731)]*day1$hum[c(632:731)] * day1$windspeed[c(632:731)]
diff_temp_hum_wind_new = test_weather_temp_hum_wind_new - day1$weathersit[c(632:731)]
diff_temp_hum_wind_new_round = round(diff_temp_hum_wind_new)
sum(diff_temp_hum_wind_new_round == 0)
```

This model correctly predicts 40 outcomes out of 100. By comparing the results of the previous model, we can tell that the interaction between temperature, humidity and wind speed, if it do exists, should not

be treated like this.

So how should we manipulate this to give a more suitable model?

2.2.6 Predict weather by temperature, humidity, winds peed and feeling temperature

Notice that we do have the data for atemp, which describes the feeling temperature of a given day. By a little surfing on the Internet, we find that the feeling temperature is influenced by temperature, wind speed and humidity. So we will use atemp as our interaction between wind speed, humidity and temperature. Let's generate a new model based on what we discussed.

```
weather_by_temp_hum_wind_atemp = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
    day1$atemp[c(1:631)] + day1$windspeed[c(1:631)] + day1$hum[c(1:631)]))
```

The information of the generated model is here:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.2180	0.1085	-2.01	0.0450
day1$temp[c(1:631)]	0.0065	0.0147	0.45	0.6563
day1$atemp[c(1:631)]	-0.0142	0.0118	-1.20	0.2292
day1$windspeed[c(1:631)]	0.0163	0.0035	4.73	0.0000
day1$hum[c(1:631)]	2.4179	0.1195	20.23	0.0000

The function generated is $We getanewmodelhere : weather = -0.217957 + 0.006529 \cdot temp - 0.014185 \cdot atemp + 0.016347 \cdot windspeed + 2.417899 \cdot hum$

Some of the P values generated is too high, so if this model's performance is not the best, this model is hardly the optimise model we will choose.

Let's see the performance of our new model.

```
test_weather_temp_hum_wind_atemp = -0.217957 + 0.006529 * day1$temp[c(632:731)] - 0.014185 *
    day1$atemp[c(632:731)] + 0.016347 * day1$windspeed[c(632:731)] + 2.417899 *
    day1$hum[c(632:731)]
diff_temp_hum_wind_atemp = test_weather_temp_hum_wind_atemp - day1$weathersit[c(632:731)]
diff_temp_hum_wind_atemp_round = round(diff_temp_hum_wind_atemp)
sum(diff_temp_hum_wind_atemp_round == 0)
```

This model gives 78 correct predictions out of 100. It gives one more correct prediction compares to the model generated by Section 2.2.4. But we are still not sure which model to choose since the lead is tiny, and the P value of this model is high.

2.2.7 Predict weather by temperature, humidity, wind speed, feeling temperature, seasons and month

Till now we have our model involving temperature, feeling temperature, wind speed and humidity. The performance of our model is OK, but we still want to improve it. Seasons will also influence weather. For example, in my hometown, it usually rains during March and April. So we can assume seasons and month will also act as a predictor.

```
weather_by_temp_hum_wind_atemp_season_month = summary(lm(day1$weathersit[c(1:631)] ~
    day1$temp[c(1:631)] + day1$hum[c(1:631)] + day1$windspeed[c(1:631)] + day1$season[c(1:631)] +
    day1$mnth[c(1:631)]))
```

The information of the generated model is

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1799	0.1035	-1.74	0.0829
day1$temp[c(1:631)]	-0.0108	0.0022	-4.97	0.0000
day1$hum[c(1:631)]	2.4356	0.1210	20.12	0.0000
day1$windspeed[c(1:631)]	0.0165	0.0034	4.77	0.0000
day1$season[c(1:631)]	0.0567	0.0341	1.66	0.0964
day1$mnth[c(1:631)]	-0.0230	0.0105	-2.19	0.0289

The P value is relatively small, which is something we are expecting. The function generated by this model is $weather = -0.179861 - 0.010750 \cdot temp + 2.435579 \cdot hum + 0.016459 \cdot windspeed + 0.056707 \cdot season - 0.023030 \cdot month$

Let's apply this to the data-set to see the results:

```
test_weather_temp_hum_wind_atemp_season_month = -0.179861 - 0.010750 * day1$temp[c(632:731)] +
    2.435579 * day1$hum[c(632:731)] + 0.016459 * day1$windspeed[c(632:731)] + 0.056707 *
    day1$season[c(632:731)] - 0.023030 * day1$mnth[c(632:731)]
diff_temp_hum_wind_atemp_season_month = test_weather_temp_hum_wind_atemp_season_month -
    day1$weathersit[c(632:731)]
diff_temp_hum_wind_atemp_season_month_round = round(diff_temp_hum_wind_atemp_season_month)
sum(diff_temp_hum_wind_atemp_season_month_round == 0)
```

As the result, this model gives 74 correct predictions among 100. But this model is not as good as the model in Section 2.2.4. We will continue our journey towards the best fir model.

2.2.8 Predict weather by temperature, humidity, winds peed and month

Recall that, at the beginning of Section 2, we noticed that the graph representing the relationships of temperature and feeling temperature is almost a straight line, this tells us that maybe we only need one of the two predictors. Besides, Month and Season also follows that relationship. We know what month it is, we know the season of that month.

We will now generate a model to estimate weather based on temperature, humidity, winds peed and month with the following command:

```
weather_by_temp_hum_wind_month = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
    day1$mnth[c(1:631)] + day1$hum[c(1:631)] + day1$windspeed[c(1:631)]))
```

Here is the information of the model generated:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1486	0.1020	-1.46	0.1454
day1$temp[c(1:631)]	-0.0098	0.0021	-4.70	0.0000
day1$mnth[c(1:631)]	-0.0086	0.0059	-1.44	0.1492
day1$hum[c(1:631)]	2.4400	0.1212	20.14	0.0000
day1$windspeed[c(1:631)]	0.0163	0.0035	4.72	0.0000

Notice that the P value for this model is acceptable, though it is not the best P value we are expecting. The linear function generated by this model is $weather = -0.148649 - 0.009815 \cdot temp - 0.008568 \cdot month + 2.439992 \cdot humdity + 0.016282 \cdot windspeed$

We will apply this function to our data-set

```
test_weather_temp_hum_wind_month = -0.148649 - 0.009815 * day1$temp[c(632:731)] -0.008568 *
    day1$mnth[c(632:731)] + 2.439992 * day1$hum[c(632:731)] + 0.016282 * day1$windspeed[c(632:731)]
diff_temp_hum_wind_month = test_weather_temp_hum_wind_month - day1$weathersit[c(632:731)]
diff_temp_hum_wind_month_round = round(diff_temp_hum_wind_month)
sum(diff_temp_hum_wind_month_round == 0)
```

As the result, this model correctly predicts 74 out of total 100 cases. Till now, models introduced in Section 2.2.4 is still the best model we got.

2.2.9 Predict weather by temperature, humidity, wind speed, month and casual bikers

Will people's action reflect the weather? Consider the case of bike rentals. The registered rentals will always happen, since people make the appointment before knowing the weather of the coming day. But the casual bike rental do not always happen. People will be less likely to rent bikes when weather is not suitable for biking. So we will introduce casual bike rental as our new predictor.

We generate the model estimate weather based on temperature, humidity, wind speed, month and casual bike rental

```
weather_by_temp_hum_wind_month_casual = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)]
    + day1$windspeed[c(1:631)] + day1$hum[c(1:631)] + day1$casual[c(1:631)]))
```

The information of generated model is:

	Estimate	Std. Error	t value	Pr(>|t|)
(Intercept)	-0.1065	0.1039	-1.03	0.3057
day1$temp[c(1:631)]	-0.0078	0.0023	-3.33	0.0009
day1$windspeed[c(1:631)]	0.0158	0.0034	4.60	0.0000
day1$hum[c(1:631)]	2.3490	0.1214	19.35	0.0000
day1$casual[c(1:631)]	-0.0001	0.0000	-2.43	0.0153

The p value of generated model is acceptable. The linear function generated is $weather = -0.1065 - 0.007766 * temp + 0.001582 * windspeed + 2.349 * hum - 0.00007253 * causal$

We then apply this function to see its performance

```
test_weather_temp_hum_wind_month_casual = -0.1065 - 0.007766 * day1$temp[c(632:731)] + 0.01582 *
    day1$windspeed[c(632:731)] + 2.349 * day1$hum[c(632:731)] - 0.00007253 *
    day1$casual[c(632:731)]
diff_temp_hum_wind_month_casual = test_weather_temp_hum_wind_month_casual -
    day1$weathersit[c(632:731)]
diff_temp_hum_wind_month_casual_round = round(diff_temp_hum_wind_month_casual)
sum(diff_temp_hum_wind_month_casual_round == 0)
```

This model correctly predicts 79 outcomes out of 100. Consider its relatively low P value and its best performance. We conclude that this model is our most-fit model to predict weather.

2.2.10 Conclusion

In conclusion, the models introduced in Section 2.2.9 is the best model to predict weather. The linear function is $weather = -0.1065 - 0.007766 * temp + 0.001582 * windspeed + 2.349 * hum - 0.00007253 * causal$

3 Contributions

3.1 Hoseung Lee (914001975, hoslee)

For Problem B, I was in charge of everything related to the KNN Model. Section 2.1.5 contains my best KNN-based model. At the very beginning of the project, we also discussed the relevance of the different predictors, since that is relevant regardless of the classification method we used.

3.2 Yi Lian (998370114, yilian)

For Problem A, Hoseung Lee and I scavenged over the campus together to find posters to analyzed. We found 7 posters in Hart Hall that could be used by our project. For Problem B, I was in charge of everything related to linear regression model. The key point to construct a linear regression model is to choose the correct predictor. So I tried to introduce difference models by different predictors. In my points of view, the models in Section 2.2.9 is the best fit model towards our goal.

A 2.1 - KNN Model

```
BKNNStep1 <- function() {
 #Step 1 - Only casual users
  data(day1)
  #Use the first 600 users, weather and casual passengers
  day1Mod <- day1[1:631,c(9,14)]
  day1Modx <- day1Mod[,2:2]
  weathersit1 <- day1Mod$weathersit
  #day1Modx is entire data except for the weathersit column
  #weathersit1 is just the weather column
  #newx is vector of features for a new wcase to be predicted
  #k :number of nearest neighbors

  #knnout <- basicKNN(day1Modx,weathersit1,c(20),5)
  #knnout$regests

  #Get the actual weather data to compare
  day1ModTest <- day1[632:731,c(9,14)]
  dayTestWeather <- day1ModTest[,1:1]
  dayTestPass <- day1ModTest[,2:2]
  dayTestWeatherKNN <- replicate(100,0)
  dayTestWeatherJust1s <- replicate(100,1)
  #apply knn from start to end of the testing set
  for(i in 1:100) {
    knnout <- basicKNN(day1Modx,weathersit1,c(dayTestPass[i]),5)
    weatherNum <- round(knnout$regests)
    dayTestWeatherKNN[i] <- weatherNum
  }
  #compare our weather predictions with the actual weather
  correctCount <- 0
  for(i in 1:100) {
    if (dayTestWeatherKNN[i] == dayTestWeather[i])
      correctCount <- correctCount + 1
  }
  print("The correct percentage (100 data points tests) is: ")
  print(correctCount/100)

  correctCount1 <- 00
  for(i in 1:100) {
    if (dayTestWeatherJust1s[i] == dayTestWeather[i])
      correctCount1 <- correctCount1 + 1
  }
  print("The correct percentage, by just assuming the weather will be category 1 (100 data points
      tests) is: ")
  print(correctCount1/100)
}

BKNNStep2 <- function() {
  #Step 2 - Only normal temperature (temp)
  data(day1)
  #Use the first 600 users, weather and temperature
  day1Mod <- day1[1:631,c(9,10)]
  day1Modx <- day1Mod[,2:2]
```

```
  weathersit1 <- day1Mod$weathersit
  #day1Modx is entire data except for the weathersit column (just the temp)
  #weathersit1 is just the weather column
  #newx is vector of features for a new wcase to be predicted
  #k :number of nearest neighbors

  #knnout <- basicKNN(day1Modx,weathersit1,c(20),5)
  #knnout$regests

  #Get the actual weather data to compare
  day1ModTest <- day1[632:731,c(9,10)]
  dayTestWeather <- day1ModTest[,1:1]
  dayTestTemp <- day1ModTest[,2:2]
  dayTestWeatherKNN <- replicate(100,0)
  dayTestWeatherJust1s <- replicate(100,1)
  #apply knn from start to end of the testing set
  for(i in 1:100) {
    knnout <- basicKNN(day1Modx,weathersit1,c(dayTestTemp[i]),5)
    weatherNum <- round(knnout$regests)
    dayTestWeatherKNN[i] <- weatherNum
  }
  #compare our weather predictions with the actual weather
  correctCount <- 0
  for(i in 1:100) {
    if (dayTestWeatherKNN[i] == dayTestWeather[i])
      correctCount <- correctCount + 1
  }
  print("The correct percentage (100 data points tests) is: ")
  print(correctCount/100)

  correctCount1 <- 00
  for(i in 1:100) {
    if (dayTestWeatherJust1s[i] == dayTestWeather[i])
      correctCount1 <- correctCount1 + 1
  }
  print("The correct percentage, by just assuming the weather will be category 1 (100 data points
      tests) is: ")
  print(correctCount1/100)
}

#casual passengers AND temperature
BKNNStep3 <- function() {
  #Step 3 - Only temperature and casual passengers
  data(day1)
  #Use the first 600 users, weather and temperature
  day1Mod <- day1[1:631,c(9,10,14)]
  day1Modx <- day1Mod[,2:3]

  weathersit1 <- day1Mod$weathersit
  #day1Modx is entire data except for the weathersit column (just the temp)
  #weathersit1 is just the weather column
  #newx is vector of features for a new wcase to be predicted
  #k :number of nearest neighbors

  #knnout <- basicKNN(day1Modx,weathersit1,c(20),5)
```

```
  #knnout$regests

  #Get the actual weather data to compare
  day1ModTest <- day1[632:731,c(9,10,14)]
  dayTestWeather <- day1ModTest[,1:1]
  dayTestPassTemp <- day1ModTest[,2:3]
  dayTestWeatherKNN <- replicate(100,0)
  dayTestWeatherJust1s <- replicate(100,1)
  #apply knn from start to end of the testing set
  for(i in 1:100) {
    knnout <- basicKNN(day1Modx,weathersit1,c(dayTestPassTemp[i,1], dayTestPassTemp[i,2]),5)
    weatherNum <- round(knnout$regests)
    dayTestWeatherKNN[i] <- weatherNum
  }
  #compare our weather predictions with the actual weather
  correctCount <- 0
  for(i in 1:100) {
    if (dayTestWeatherKNN[i] == dayTestWeather[i])
      correctCount <- correctCount + 1
  }
  print("The correct percentage (100 data points tests) is: ")
  print(correctCount/100)

  correctCount1 <- 00
  for(i in 1:100) {
    if (dayTestWeatherJust1s[i] == dayTestWeather[i])
      correctCount1 <- correctCount1 + 1
  }
  print("The correct percentage, by just assuming the weather will be category 1 (100 data points
      tests) is: ")
  print(correctCount1/100)
}


# dates only no modification
BKNNStep4 <- function() {
  #Step 4 - Only Use Dates, create days going from 1 to 365 instead of individual dates (convert
      dteday)
  data(day1)
  day1Mod <- cbind(day1,replicate(731, 0))
  colnames(day1Mod)[17] <- "Dayof365"
  day <- 1
  for (i in 1:731) {
    if (day == 366) {
      day <- 1
      #reset 1 after you hit 366
    }
    day1Mod[i,17] = day
    day <- day + 1
  }
  #Use the first 600 users, day
  day1ModOrig <- day1Mod;
  day1Mod <- day1Mod[1:631,c(9,17)]
  day1Modx <- day1Mod[,2:2]
```

```
  weathersit1 <- day1Mod$weathersit
  #day1Modx is entire data except for the weathersit column (just the temp)
  #weathersit1 is just the weather column
  #newx is vector of features for a new wcase to be predicted
  #k :number of nearest neighbors

  #knnout <- basicKNN(day1Modx,weathersit1,c(20),5)
  #knnout$regests

  #Get the actual weather data to compare
  day1ModTest <- day1ModOrig[632:731,c(9,17)]
  dayTestWeather <- day1ModTest[,1:1]
  dayTestDay <- day1ModTest[,2:2]
  dayTestWeatherKNN <- replicate(100,0)
  dayTestWeatherJust1s <- replicate(100,1)
  #apply knn from start to end of the testing set
  for(i in 1:100) {
    knnout <- basicKNN(day1Modx,weathersit1,c(dayTestDay[i]),5)
    weatherNum <- round(knnout$regests)
    dayTestWeatherKNN[i] <- weatherNum
  }
  #compare our weather predictions with the actual weather
  correctCount <- 0
  for(i in 1:100) {
    if (dayTestWeatherKNN[i] == dayTestWeather[i])
      correctCount <- correctCount + 1
  }
  print("The correct percentage (100 data points tests) is: ")
  print(correctCount/100)

  correctCount1 <- 00
  for(i in 1:100) {
    if (dayTestWeatherJust1s[i] == dayTestWeather[i])
      correctCount1 <- correctCount1 + 1
  }
  print("The correct percentage, by just assuming the weather will be category 1 (100 data points
      tests) is: ")
  print(correctCount1/100)


}

# temp, hum, win, casual
BKNNStep5 <- function() {
  #Step 5 - # temp, hum, win, casual
  data(day1)

  day1Mod <- day1[1:631,c(9, 10,12,13,14)]
  day1Modx <- day1Mod[,c(2:5)]

  weathersit1 <- day1Mod$weathersit
  #day1Modx is entire data except for the weathersit column (just the temp)
  #weathersit1 is just the weather column
  #newx is vector of features for a new wcase to be predicted
  #k :number of nearest neighbors
```

```
  #knnout <- basicKNN(day1Modx,weathersit1,c(20),5)
  #knnout$regests

  #Get the actual weather data to compare
  day1ModTest <- day1[632:731,c(9, 10,12,13,14)]
  dayTestWeather <- day1ModTest[,1:1]
  dayTestVar <- day1ModTest[,c(2:5)]
  dayTestWeatherKNN <- replicate(100,0)
  dayTestWeatherJust1s <- replicate(100,1)
  #apply knn from start to end of the testing set
  for(i in 1:100) {
    knnout <- basicKNN(day1Modx,weathersit1,c(dayTestVar[i,1], dayTestVar[i,2], dayTestVar[i,3],
        dayTestVar[i,4]),21)
    weatherNum <- round(knnout$regests)
    dayTestWeatherKNN[i] <- weatherNum
  }
  #compare our weather predictions with the actual weather
  correctCount <- 0
  for(i in 1:100) {
    if (dayTestWeatherKNN[i] == dayTestWeather[i])
      correctCount <- correctCount + 1
  }
  print("The correct percentage (100 data points tests) is: ")
  print(correctCount/100)

  correctCount1 <- 00
  for(i in 1:100) {
    if (dayTestWeatherJust1s[i] == dayTestWeather[i])
      correctCount1 <- correctCount1 + 1
  }
  print("The correct percentage, by just assuming the weather will be category 1 (100 data points
      tests) is: ")
  print(correctCount1/100)
}
```

A 2.2 - Linear Regression Model

```
data("day1")
## For the dataset day1, there are columns months, holiday, weekday, working day, weathersit
## which indicates the weather of the day
##    - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
##    - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
##    - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
##    - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog,
## temperature, atemperature which is the feeling temperature.
weather_temp <- function() {
## I will first introduce the relationship between weather and temperature. Cause intuitively,
## those two factors are most related to me.
  weather_by_temp = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)]))

## By looking at the weather_by_temp dataset, we see that the system gives a model that
## weathersit = 1.500839 - 0.007539 * temp.

## Let's apply this relationship to see if it is accurate enough
  test_weather_temp = 1.500839 - 0.007539 * day1$temp[c(632:731)]
  diff_temp = test_weather_temp - day1$weathersit[c(632:731)]
  diff_temp_round=round(diff_temp)
  sum(diff_temp_round == 0)
  print(sum(diff_temp_round == 0))
## The ideally result for the diff should be all, which indicating that our prediction is accurate.
## As we can see from the sum functions, there are 54 0's. That is our prediction is half accurate,
## but we are not satisfied.
}

## Let's introduce the humidity.
weather_temp_hum <- function() {
  weather_by_temp_hum = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
      day1$hum[c(1:631)]))
## The generated model by lm function is weathersit = 0.155168 - 0.012417 * temp + 2.275260 * hum
## Let's test it on the last 100 set of data.
  test_weather_temp_hum = 0.155168 - 0.012417 * day1$temp[c(632:731)] + 2.275260 *
      day1$hum[c(632:731)]
  diff_temp_hum = test_weather_temp_hum - day1$weathersit[c(632:731)]
  diff_temp_hum_round = round(diff_temp_hum)
  sum(diff_temp_hum_round == 0)
  print(sum(diff_temp_hum_round == 0)
)
}
## After introducing the humidity, we got 74 entries that the difference is close to 0, which means
## our model is right in 74/100 cases for the last 100 days. But weather can be influenced by
    almost
## everything, so lets introduce more.
weather_temp_hum_new <- function() {
## Since the humidity change will incluence the weather differenety, let's introduce the
    interaction
## between humidity and temperature.
  weather_by_temp_hum_new = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] *
      day1$hum[c(1:631)]))
## We now have a new model that weathersit = 0.108737 - 0.009048 * temp + 2.355487 * hum +
```

```
    -0.005668 * temp * hum
  test_weather_temp_hum_new = 0.108737 - 0.009048 * day1$temp[c(632:731)] + 2.355487 *
      day1$hum[c(632:731)] - 0.005668 * day1$hum[c(632:731)] * day1$temp[c(632:731)]
  diff_temp_hum_new = test_weather_temp_hum_new - day1$weathersit[c(632:731)]
  diff_temp_hum_new_round = round(diff_temp_hum_new)
  sum(diff_temp_hum_new_round == 0)
  print(sum(diff_temp_hum_new_round == 0))
}
## After introducing the indicator, we get 74/100 correct prediction, we can see the accuracy of
    our estimation is almost the same. In addition, the coeeficient for the interaction is
    relatively low, which
## indicats that the interaction between humidity and temperature do not have a great influence on
    our model. We will discard this interaction later.
weather_temp_hum_wind <- function() {
## Now lets introduce the windspeed
  weather_by_temp_hum_wind = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
      day1$hum[c(1:631)] + day1$windspeed[c(1:631)]))
## The model generated are now weathersit = -0.170059 - 0.010964 * temp + 2.408763 * humidity +
    0.017007 * windspeed
## Let's try this model on the rest 100 entries

  test_weather_temp_hum_wind = -0.170059 - 0.010964 * day1$temp[c(632:731)] + 2.408763 *
      day1$hum[c(632:731)] + 0.017007 * day1$windspeed[c(632:731)]
  diff_temp_hum_wind = test_weather_temp_hum_wind - day1$weathersit[c(632:731)]
  diff_temp_hum_wind_round = round(diff_temp_hum_wind)
  sum(diff_temp_hum_wind_round == 0)
  print(sum(diff_temp_hum_wind_round == 0)
)
}
## The model related to temperature, humidity and windspeed gives 77 correct prediction out of 100
    entries. This is a little progress compare with the last model that
## gives 75 correct predictions. And notice that, the P value for these 3 estimated cooeficients
    are relatively small now (<0.01) compare with the old P value that are around 0.5.
## This means our model is now more confident about the cooeficient it generated.

## Till now we have linked the relationship between weather and our independent variables
    windspeed, humidity and temperature. So what can we tell about their interactions?
## Intuitively, humidity's influence on weather under low temperature and windspeed is different
    compare with the humidity under high tempareture and windspeed. So let's try
## introduce their interaction exclusively
weather_temp_hum_wind_new <- function () {
  weather_by_temp_hum_wind_new = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] *
      day1$hum[c(1:631)] * day1$windspeed[c(1:631)]))
## We get a new model with 7 terms: weather = 0.444522 * -0.034163 * temp + 1.173436 * hum
    -0.033980 * windspeed + 0.053640 * temp * hum + 0.002233 * temp * windspeed
## + 0.104494 * hum * windspeed - 0.005036 * temp * hum * windspeed

## Notice that tht P value is relatively large, which implied this model is not precise enough.
    Let's give it a try anyway
  test_weather_temp_hum_wind_new = 0.444522 - 0.034163 * day1$temp[c(632:731)] + 1.173436 *
      day1$hum[c(632:731)] - 0.033980 * day1$windspeed[c(632:731)] + 0.053640 *
      day1$temp[c(632:731)] * day1$hum[c(632:731)] + 0.002233 * day1$temp[c(632:731)] *
      day1$windspeed[c(632:731)] + 0.0104494 * day1$hum[c(632:731)] * day1$windspeed[c(632:731)] -
      0.005036 * day1$temp[c(632:731)]*day1$hum[c(632:731)] * day1$windspeed[c(632:731)]
  diff_temp_hum_wind_new = test_weather_temp_hum_wind_new - day1$weathersit[c(632:731)]
```

```
  diff_temp_hum_wind_new_round = round(diff_temp_hum_wind_new)
  sum(diff_temp_hum_wind_new_round == 0)
  print(sum(diff_temp_hum_wind_new_round == 0))
}
## We get 40 correct predictions based on this model. By camparing the results draw by previous
    models, we can tell that this model is not the best model. So the interaction
## between windspeed, humidity and temperature should not be done like this.
## So how should we manipulate this to give a more suitable model?
## Notice that we do have the data for atemp, which describes the feeling temperature of a given
    day. By a little surfing on the Internet, we find that the feeling temperature
## is influenced by temperature, windspeed and humidity. So we will use atemp as our interaction
    between windspeed, humidity and tempareture.
## Let's generate a new model based on what we discussed
weather_temp_hum_wind_atemp <- function () {
  weather_by_temp_hum_wind_atemp = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
      day1$atemp[c(1:631)] + day1$windspeed[c(1:631)] + day1$hum[c(1:631)]))
## We get a new model here: weather = -0.217957 + 0.006529 * temp - 0.014185 * atemp + 0.016347 *
    windspeed + 2.417899 * hum
##Let's try this on the last 100 entried to see if the accuracy get improved
  test_weather_temp_hum_wind_atemp = -0.217957 + 0.006529 * day1$temp[c(632:731)] - 0.014185 *
      day1$atemp[c(632:731)] + 0.016347 * day1$windspeed[c(632:731)] + 2.417899 *
      day1$hum[c(632:731)]
  diff_temp_hum_wind_atemp = test_weather_temp_hum_wind_atemp - day1$weathersit[c(632:731)]
  diff_temp_hum_wind_atemp_round = round(diff_temp_hum_wind_atemp)
  sum(diff_temp_hum_wind_atemp_round == 0)
  print(sum(diff_temp_hum_wind_atemp_round == 0))
}
## We now have 78/100 predictions that are correct for the last 100 days. So far this is the model
    with the best preformance. But we are still not satisfied.

## Till now we have our model involving temperature, feeling temperature, windspeed and humidity.
    The performance of our model is ok, but we still want to improve it.
## Seasons will also influence weather. For example, in my hometown, it usually rainy during March
    and April. So we can assume seasons and month will act as a predictor.
## Let's try it
weather_temp_hum_wind_atemp_season_month <- function() {
  weather_by_temp_hum_wind_atemp_season_month = summary(lm(day1$weathersit[c(1:631)] ~
      day1$temp[c(1:631)] + day1$hum[c(1:631)] + day1$windspeed[c(1:631)] + day1$season[c(1:631)]
      + day1$mnth[c(1:631)]))
## The model generated is weather = -0.179861 - 0.010750 * temp + 2.435579 * hum + 0.016459 *
    windspeed + 0.056707 * season - 0.023030 * month
## Apply this model to test the accuracy of prediction
  test_weather_temp_hum_wind_atemp_season_month = -0.179861 - 0.010750 * day1$temp[c(632:731)] +
      2.435579 * day1$hum[c(632:731)] + 0.016459 * day1$windspeed[c(632:731)] + 0.056707 *
      day1$season[c(632:731)] - 0.023030 * day1$mnth[c(632:731)]
  diff_temp_hum_wind_atemp_season_month = test_weather_temp_hum_wind_atemp_season_month -
      day1$weathersit[c(632:731)]
  diff_temp_hum_wind_atemp_season_month_round = round(diff_temp_hum_wind_atemp_season_month)
  sum(diff_temp_hum_wind_atemp_season_month_round == 0)
  print(sum(diff_temp_hum_wind_atemp_season_month_round == 0))
}
## The accuracy for this model is 74/100. Let's try remove season and atemp
weather_temp_hum_wind_month <- function() {
  weather_by_temp_hum_wind_month = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
      day1$mnth[c(1:631)] + day1$hum[c(1:631)] + day1$windspeed[c(1:631)]))
```

```
## The model generated is weather = -0.148649 - 0.009815 * temp - 0.008568 * month + 2.439992 *
    humdity + 0.016282 * windspeed
## Let's see its performance
  test_weather_temp_hum_wind_month = -0.148649 - 0.009815 * day1$temp[c(632:731)] -0.008568 *
      day1$mnth[c(632:731)] + 2.439992 * day1$hum[c(632:731)] + 0.016282 *
      day1$windspeed[c(632:731)]
  diff_temp_hum_wind_month = test_weather_temp_hum_wind_month - day1$weathersit[c(632:731)]
  diff_temp_hum_wind_month_round = round(diff_temp_hum_wind_month)
  sum(diff_temp_hum_wind_month_round == 0)
  print(sum(diff_temp_hum_wind_month_round == 0))
}
## This model correctly predicts 74 outcomes out of 100.

## Let's try introduce the registered user, and the working day
weather_temp_hum_wind_month_casual <- function() {
  weather_by_temp_hum_wind_month_casual = summary(lm(day1$weathersit[c(1:631)] ~
      day1$temp[c(1:631)] + day1$windspeed[c(1:631)] + day1$hum[c(1:631)] + day1$casual[c(1:631)]))
## the function is weather = -0.1065 - 0.007766 * temp + 0.001582 * windspeed + 2.349 * hum -
    0.00007253 * causal
  test_weather_temp_hum_wind_month_casual = -0.1065 - 0.007766 * day1$temp[c(632:731)] + 0.01582 *
      day1$windspeed[c(632:731)] + 2.349 * day1$hum[c(632:731)] - 0.00007253 *
      day1$casual[c(632:731)]
  diff_temp_hum_wind_month_casual = test_weather_temp_hum_wind_month_casual -
      day1$weathersit[c(632:731)]
  diff_temp_hum_wind_month_casual_round = round(diff_temp_hum_wind_month_casual)
  sum(diff_temp_hum_wind_month_casual_round == 0)
  print(sum(diff_temp_hum_wind_month_casual_round == 0))
}

weather_temp_hum_wind_casual <- function() {
  weather_by_temp_hum_wind_casual = summary(lm(day1$weathersit[c(1:631)] ~ day1$temp[c(1:631)] +
      day1$hum[c(1:631)] + day1$windspeed[c(1:631)] + day1$casual[c(1:631)]))
  test_weather_temp_hum_wind_casual = -0.1065 - 0.007766 * day1$temp[c(632:731)] + 2.349 *
      day1$hum[c(632:731)] + 0.01582 * day1$windspeed[c(632:731)] - 0.00007253 *
      day1$casual[c(632:731)]
  diff_temp_hum_wind_casual = test_weather_temp_hum_wind_casual - day1$weathersit[c(632:731)]
  diff_temp_hum_wind_casual_round = round(diff_temp_hum_wind_casual)
  sum(diff_temp_hum_wind_casual_round == 0)
  print(sum(diff_temp_hum_wind_casual_round == 0))
}
```