**PROJECT-REPORT**

**CS 6500/MAS 6500 P1**

**GROUP-4**

**ANALYSIS ON WEATHER DATA**

**SUBMITTED BY:**

**Ravali Bongoni**

**Sreelatha Yarragudi**

**INTRODUCTION**

In this project, we analyzed weather data from the National Oceanic and Atmospheric Administration’s (NOAA) National Climatic Data Center(NCDC). This is a group project in which we are assigned 4 Tasks to perform on this dataset from the year 1901 to 1940 using Map-Reduce Programming Model for distributed processing. This Dataset contains ‘\*’ which means that the elements are not reported. In this project, we are concerned with three variables: USAF(Station-

ID), ‘YR—MODAHRMN’ (year, month, day, and time), TEMP(Temperature). For every Task, we used Map-Reduce programs for analyzing the weather data. This report contains detailed information about each Task, how much time taken to import the data, description of the problem occurred during execution.

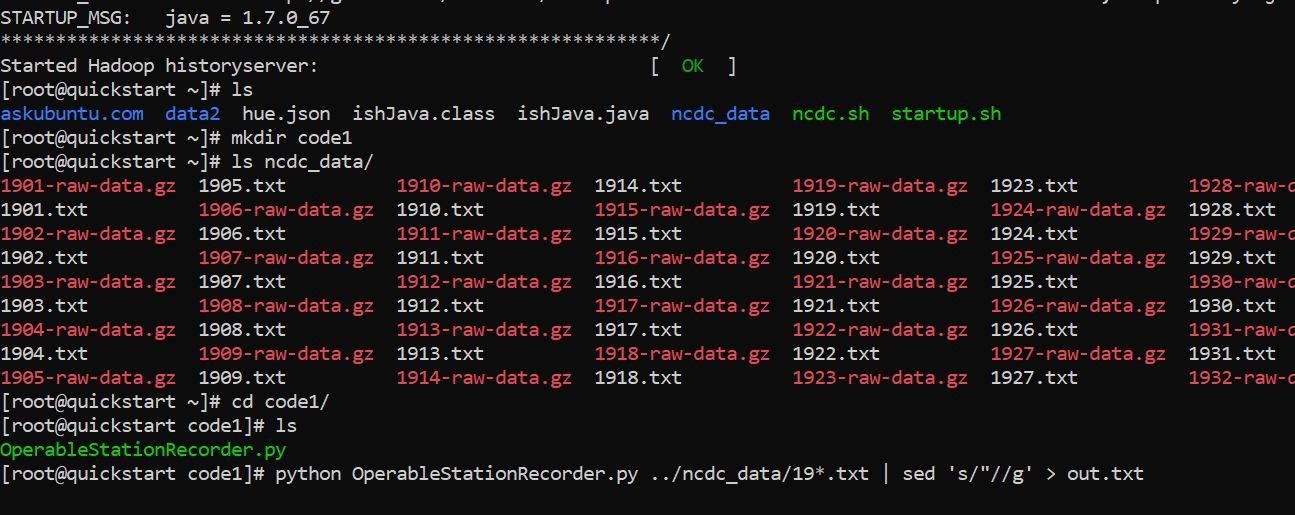
For this project, we copy the docker container into our local machine because the data is lost whenever we exit from the container and to save the data even after exiting, we always have to commit changes we make to the container. So, basically, we only need the data in our local so that we can write code for each task and to analyze the output. We copied one specific folder from the docker container and copied to the local. The command we used for copying the data folder from container is:

docker cp my\_container:/root/ncdc\_data data

In this command, my\_container is the container ID from where you want to copy the folder.

**TASK0- FILTERING:**

For this task, the main idea is to understand the meaning of ‘operable’. So, if a weather station reported at-least more than two observations per month during that year then that particular station is considered to be ‘operable’. This task is only concerned with Station-ID, year and month but our output consists of operable stations and all the remaining information related to that station as it is. Below is the snapshot of the list of data files and commands written to run the python code for this task.



We came up with the code OperableStationRecorder.py mentioned below. This code is splitting the data by spaces and yielding Station-ID as key and remaining line as value. Then the year and month are counted based on its repetition and if the count exceeds 2 then it returns that number otherwise it returns 2. If 2 is returned then the station is 'not operable' and if it 2 is returned then it is considered operable and the output of this code given out all the columns but only for operable stations. The original data provided by the doctor dyer is used as input for this code.

Code for Task-0 from mrjob.job import MRJob from mrjob.step import MRStep

import re import time import copy class NcOperableStations(MRJob): MINIMUM\_REPORTS=2 def steps(self):

return [

MRStep(mapper=self.mapper\_get\_stations, reducer=self.reducer\_count\_ratings)

]

def mapper\_get\_stations(self, \_, line):

str=line.split(' ') stationId=str[0] timeCollected=str[2] if stationId == 'USAF': return yield stationId, (line) def reducer\_count\_ratings(self, key, values):

initValues=(list(values))[:] count=self.get\_reportingStationCount(list(initValues)) if count>= self.MINIMUM\_REPORTS: for v in (list(initValues)): yield (str(v)).replace('"', ''), '' def get\_reportingStationCount(self, reportPeriods): periodString=''.join(reportPeriods) count=0 minCount=2 for p in reportPeriods: timeVal=p[13:19]

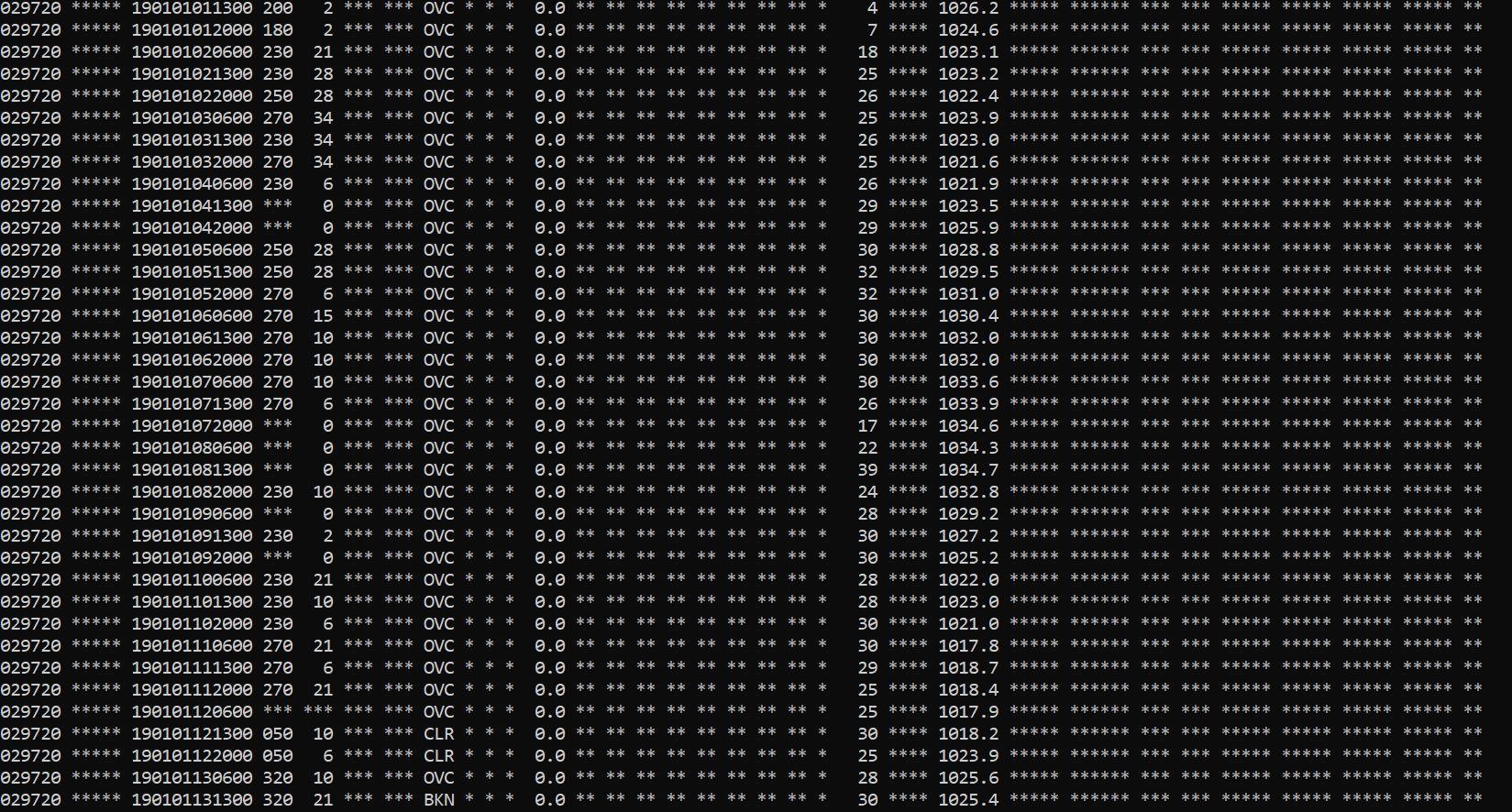
occurances = re.findall(timeVal, periodString) count=len(occurances) if self.MINIMUM\_REPORTS > count:

return count; return minCount def reducer\_sorted\_output(self, count, val): for val in val:

yield val, count if \_\_name\_\_ == '\_\_main\_\_': NcOperableStations.run()

Output:

The output of python file OperableStationRecorder.py is the text file which consists of the list of weather stations that are operable and all the remaining information. This output text file is displaying the information required for Task-0. We can view this output anytime by using:- vim out.txt and this command will display the data contained in that file as it will be used for performing the remaining tasks.



***Time taken to complete this task:*** In this task, we copied the data from docker container to local and it took few hours to figure out how to copy the specific folder from docker and the entire command took few minutes for copying. It took almost 3-4 hours for 3daysi.e., around 12 hours to come up with the correct Map-Reduce Program as the output for this task will be used in upcoming tasks.

***Challenged faced:*** In the first attempt we wrote a code but the output we got was not having temperature feature in it. But it is required in remaining tasks so we came up with another code which met all the requirements.

**TASK1- ANALYSIS 1:**

In this task, we used the output of Task-0 as input as we need to find if there are any weather stations that are operable for 80% or more of the entire time period 1920-1940.

The Map-Reduce program is written for this task to come up with an accurate output. The output for this task should contain all the weather stations which are operable between 1920-1940 time period. And if there are no weather stations fulfilling these requirements then we have to list the top fifty stations and rank them by the greatest number of years they are operable.

Code for Task1:

import sys from mrjob.job import MRJob class Task1(MRJob):

def mapper(self,key,line): if "\*\*\*" in line:

stationID=line[1:7]+line[7:11]+line[84:87] year= line[7:11] if int(year)>=1920: yield(stationID,1) def reducer(self,key,values):

yield(key,sum(values))

if \_\_name\_\_=='\_\_main\_\_':

Task1.run()

Output:

After running this code we found that there is no record of stations that are operable for at-least 80% or more for the entire time period 1920-1940. So we listed the top fifty stations with the number of years they are operable in the 1920-1940 time period. This task was accomplished without any issues and running this program does not take much time and ran without any errors. Below is the list of top 50 weather stations.

STATION-ID NUMBER OF YEARS

|  |  |
| --- | --- |
| "103380" | 15 |
| "029110" | 14 |
| "029750" | 14 |
| "108650" | 15 |
| "104190" | 12 |
| "106370" | 13 |
| "101200" | 11 |
| "101270" | 11 |
| "103840" | 11 |
| "267020" | 11 |
| "028360" | 10 |
| "100190" | 10 |
| "100670" | 10 |
| "100910" | 10 |
| "101310" | 10 |
| "101470" | 10 |
| "101700" | 10 |
| "103610" | 10 |
| "104100" | 10 |
| "104160" | 10 |
| "104270" | 10 |
| "104330" | 10 |
| "104380" | 10 |
| "104530" | 10 |

***Time taken to complete this task:*** In this task, it took only 2hours to come up with the MapReduce code as in this we just have to extract the data for 1920-1940.

***Challenged faced:*** It was a bit confusing to analyze the data obtained from this task but we figured it out after discussing it with all the group members and apart from this no significant issues were faced.

**TASK2:**

In this Task, we need to do some descriptive statistical analysis of temperature in each year. We have to calculate the mean, median, minimum and maximum temperature for each year. To calculate this we wrote a python map-reduce program. The output of Task0 is taken as input for this task. Moreover, in this task, we also have to graph all the results and then have to apply linear regression to fit a line to each graph with R2 for each fit.

The output of Task0 is taken as input for this Task and writing map-reduce programs for this task was not a problem as it took very little time to process that. But writing R-Code for this task was a bit time taking because it was difficult and confusing to come up with linear regression. Eventually, we completed this task without any major issues.

Python code for calculating mean.

For the calculating mean of temperatures, we imported the statistics module in python. Statistics module consists of the mean() function which returns the mean of numerical data. import statistics from mrjob.job import MRJob class Task2(MRJob): def mapper(self,\_,line): if '\*' in line:

year=line[13:17] temp= line[84:87] if temp!='\*\*\*\*':

itemp= int(temp) yield(year,itemp) def reducer(self,key,values):

mean= statistics.mean(values) yield(key,mean)

if \_\_name\_\_=='\_\_main\_\_': Task2.run()

Python code for calculating median.

For calculating median of temperatures, we imported statistics module in python. Statistics module consist of median() function which returns the median of numerical data.

import statistics from mrjob.job import MRJob class Task2(MRJob): def mapper(self,\_,line): if '\*' in line:

year=line[13:17] temp= line[84:87] if temp!='\*\*\*\*': itemp= int(temp) yield(year,itemp) def reducer(self,key,values): median= statistics.median(values) yield(key,median)

if \_\_name\_\_=='\_\_main\_\_': Task2.run()

Python code for calculating maximum.

To calculate maximum temperature for each year, we used python max() function. This max() function will return highest of all the temperature for each year.

from mrjob.job import MRJob class Task2(MRJob): def mapper(self,\_,line): if '\*' in line:

year=line[13:17] temp= line[84:87] if temp!='\*\*\*\*': itemp= int(temp) yield(year,itemp) def reducer(self,key,values): maximum= max(values)

yield(key,maximum)

if \_\_name\_\_=='\_\_main\_\_': Task2.run()

Python code for calculating minimum.

To calculate maximum temperature for each year, we used python min() function. This min() function will return lowest of all the temperature for each year.

from mrjob.job import MRJob class Task2(MRJob): def mapper(self,\_,line): if '\*' in line:

year=line[13:17] temp= line[84:87] if temp!='\*\*\*\*':

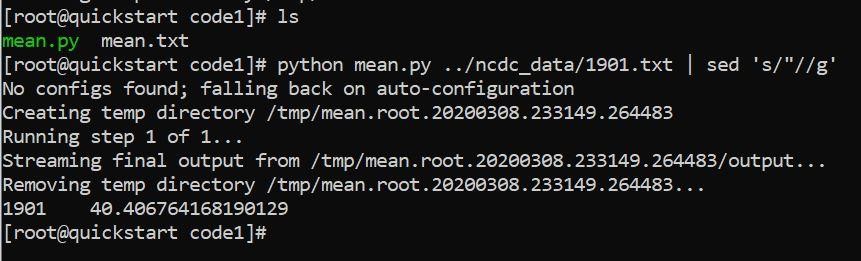
itemp= int(temp) yield(year,itemp) def reducer(self,key,values): minimum= min(values)

yield(key,minimum)

if \_\_name\_\_=='\_\_main\_\_':

Task2.run()

We ran all these python programs to calculate mean, median, minimum and maximum respectively and got output for each year separately. Below is the example of code we ran on HDFS for calculating mean for year 1901.



Output:

**Year Mean Median Maximum Minimum**

1. 40.40 40.0 89 -28
2. 35.89 36.0 76 -27
3. 40.68 38.0 84 -23

1904 37.99 38.0 78 -21 1905 39.79 38.0 83 -27

1906 40.47 38.0 85 -13

1907 37.71 38.0 83 -31

1. 37.18 36.0 84 -36
2. 36.78 36.0 82 -36
3. 38.39 37.0 85 -35
4. 37.52 36.0 87 -36

1912 35.02 35.0 90 -42

1913 37.39 37.0 86 -35

1. 37.36 37.0 92 -36
2. 32.91 35.0 85 -42
3. 36.12 37.0 89 -54
4. 37.64 39.0 90 -49
5. 36.96 36.0 100 -45
6. 39.83 39.0 85 -30
7. 38.14 38.0 83 -43
8. 36.86 37.0 82 -40
9. 36.82 38.0 85 -39
10. 38.89 39.0 85 -50

1925 38.82 37.0 89 -36

1. 38.92 40.0 120 -44
2. 45.65 46.0 120 -40
3. 45.87 45.0 100 -11
4. 49.05 49.0 91 -40
5. 51.69 51.0 104 -24

1932 48.16 49.0 120 -58

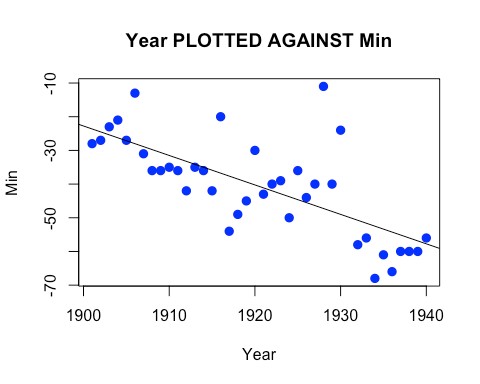
1933 46.89 49.0 120 -56

1. 47.66 49.0 120 -68
2. 46.97 48.0 120 -61
3. 47.24 48.0 131 -66
4. 47.77 50.0 120 -60
5. 52.15 52.0 120 -60
6. 52.60 52.0 120 -60
7. 51.09 54.0 120 -56

For this analysis in R in task 2 we have taken the output of task 0 which are the operable station for year 1901 to 1940. Considering years including data from all operation stations for that year we analyzed the data with descriptive statistics like mean, median, mode, minimum and maximum temperatures of the data in R which is generated from the MapReduce function of task 2. We graphed the results of these descriptive statistics with respect to each year. We also have fitted the regression line for each year and regression line equations and R square values are noted and compared.

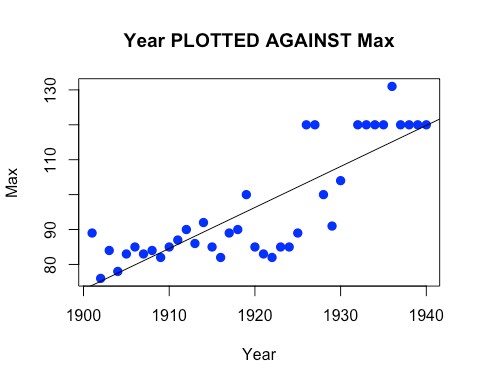
*#We tried to fit a linear regression model to these graphed plots using lm function.*

|  |
| --- |
| Task2 <- **read.csv**("~/Downloads/Task2.csv")  **attach**(Task2)  Min.lm <- **lm**(Min**~**Year, data = Task2) **summary**(Min.lm)  ##  ## Call:  ## lm(formula = Min ~ Year, data = Task2)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -16.366 -5.866 -2.494 3.133 36.261  ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) 1640.1226 277.7273 5.906 8.43e-07 \*\*\* ## Year -0.8752 0.1446 -6.051 5.35e-07 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 10.45 on 37 degrees of freedom  ## Multiple R-squared: 0.4974, Adjusted R-squared: 0.4838 ## F-statistic: 36.62 on 1 and 37 DF, p-value: 5.351e-07  **plot**(Year, Min, pch = 16, cex = 1.3, col = "blue", main = "Year PLOTTED AGAIN  ST Min", xlab = "Year", ylab = "Min")  **abline**(1640.1226,**-**0.8752) |



From the summary output we can find that the multiple R-squared is 0.4974. The fit line of min temperature and year’s equation is Min = 1640.1226 - 0.8752\*Year.

|  |
| --- |
| Max.lm <- **lm**(Max**~**Year, data = Task2) **summary**(Max.lm)  ##  ## Call:  ## lm(formula = Max ~ Year, data = Task2)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -16.616 -4.704 1.303 5.041 16.686  ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -2158.7276 245.2086 -8.804 1.31e-10 \*\*\* ## Year 1.1745 0.1277 9.198 4.26e-11 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  ## |
| ## Residual standard error: 9.223 on 37 degrees of freedom  ## Multiple R-squared: 0.6957, Adjusted R-squared: 0.6875 ## F-statistic: 84.59 on 1 and 37 DF, p-value: 4.264e-11  **plot**(Year, Max, pch = 16, cex = 1.3, col = "blue", main = "Year PLOTTED AGAIN  ST Max", xlab = "Year", ylab = "Max") **abline**(**-**2158.7276, 1.1745) |



From the summary output we can find that the multiple R-squared is 0.6957. The fit line of max temperature and year’s equation is Max = -2158.7276 + 1.1745\*Year.

Mean.lm <- **lm**(Mean**~**Year, data = Task2) **summary**(Mean.lm)

## ## Call:

## lm(formula = Mean ~ Year, data = Task2)

##

## Residuals:

## Min 1Q Median 3Q Max

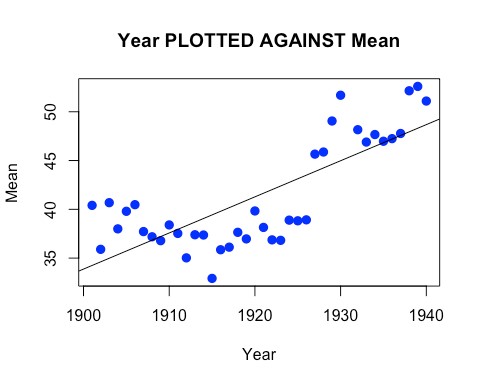
## -6.5121 -3.4019 0.1985 2.4258 6.7113

##

## Coefficients:

## Estimate Std. Error t value Pr(>|t|)

|  |  |
| --- | --- |
| ## (Intercept) -669.50546 95.58418 -7.004 2.80e-08 \*\*\* ## Year 0.37020 0.04978 7.437 7.47e-09 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 3.595 on 37 degrees of freedom  ## Multiple R-squared: 0.5992, Adjusted R-squared: 0.5884 ## F-statistic: 55.31 on 1 and 37 DF, p-value: 7.47e-09  **plot**(Year, Mean, pch = 16, cex = 1.3, col = "blue", main = "Year PLOTTED AGAI NST Mean", xlab = "Year", ylab = "Mean") | |
| **abline**(**-**669.50546, 0.37020) |  |



From the summary output we can find that the multiple R-squared is 0.5992. The fit line of max temperature and year’s equation is Mean = -669.50546 + 0.37020\*Year.

Median.lm <- **lm**(Median**~**Year, data = Task2) **summary**(Median.lm)

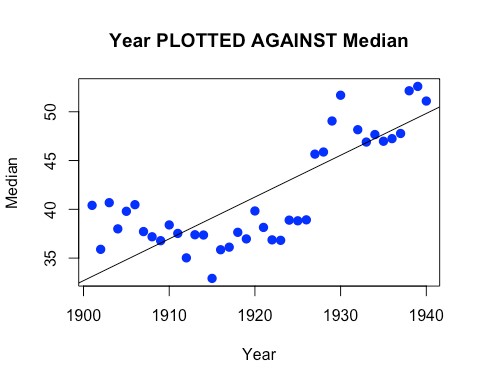
## ## Call:

## lm(formula = Median ~ Year, data = Task2)

##

## Residuals:

|  |
| --- |
| ## Min 1Q Median 3Q Max  ## -6.4029 -2.9030 0.0255 2.5972 6.8826  ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -781.57685 89.44629 -8.738 1.59e-10 \*\*\* ## Year 0.42856 0.04658 9.201 4.23e-11 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 3.364 on 37 degrees of freedom  ## Multiple R-squared: 0.6958, Adjusted R-squared: 0.6876 ## F-statistic: 84.65 on 1 and 37 DF, p-value: 4.228e-11  **plot**(Year, Mean, pch = 16, cex = 1.3, col = "blue", main = "Year PLOTTED AGAI  NST Median", xlab = "Year", ylab = "Median")  **abline**(**-**781.57685, 0.42856) |



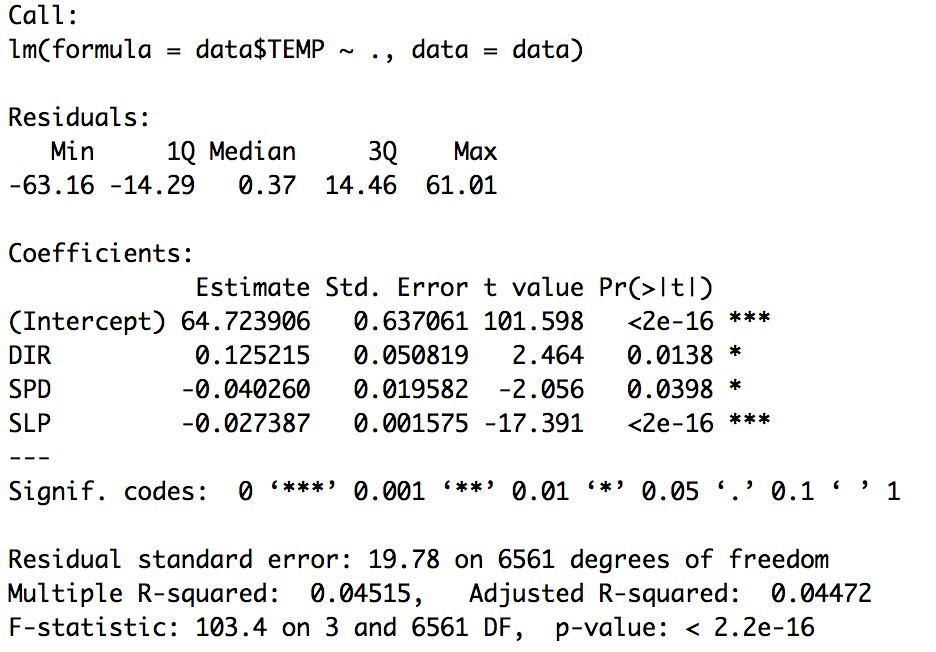
From the summary output we can find that the multiple R-squared is 0.6958. The fit line of median temperature and year’s equation is Median = -781.57685 + 0.42856\*Year.

**Task – 2 Part B:**

We did multi-linear Regression fit to all the 40 years 1901-1940 considering few variables in the data. We have taken Temperature as target variable and SLP-sea level pressure, DIR- direction of wind, SPD- Wind speed as regressor variables. Hence, we will predict the temperature for any new observation of DIR, SPD and SLP. R-square from the summary of multi-linear regression is captured and shown in the below out table.

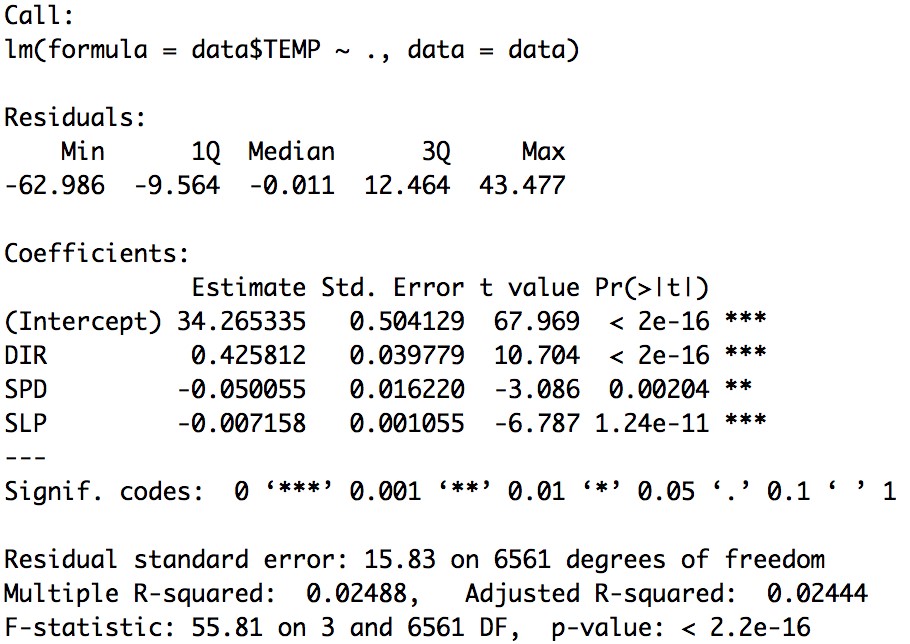
R-Code for fitting a regression model for each year.

data\_file <- read.csv("~/Downloads/Output\_task/Output\_task0/out1901.txt", sep="") data <- data\_file[,-c(25:33)] data <- data[,-c(13:21)] data <- data[,-c(0,1,2,3,6,7,8,9,10,11,12,14)] cols.num <- c("DIR", "SPD","TEMP", "SLP") data[cols.num] <- sapply(data[cols.num],as.numeric) sapply(data, class) mlr <- lm(data$TEMP~ .,data=data) summary(mlr) Ouput:



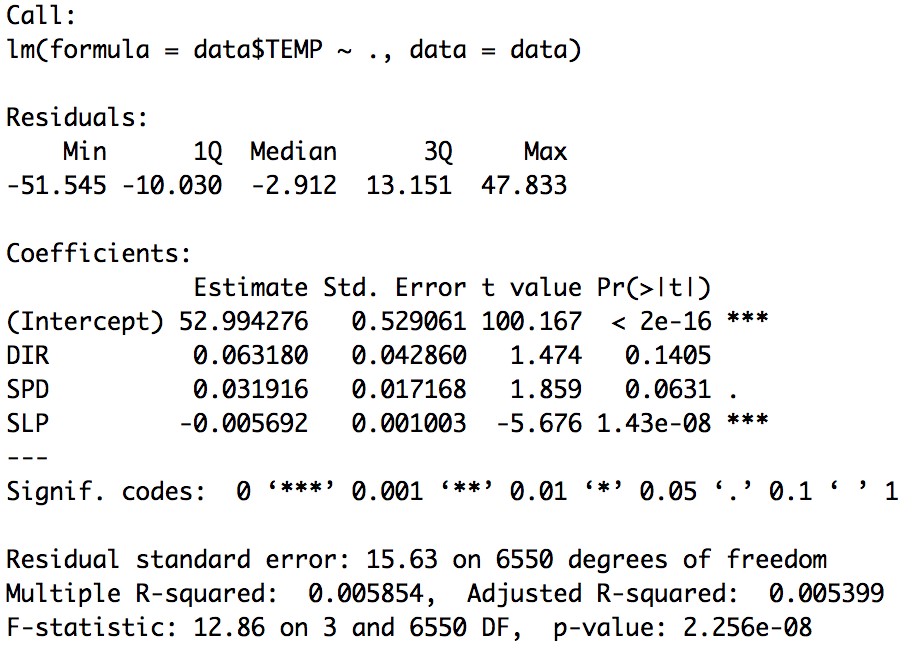
data\_file <- read.csv("~/Downloads/Output\_task/Output\_task0/out1902.txt", sep="")

data <- data\_file[,-c(25:33)] data <- data[,-c(13:21)] data <- data[,-c(0,1,2,3,6,7,8,9,10,11,12,14)] cols.num <- c("DIR", "SPD","TEMP", "SLP") data[cols.num] <- sapply(data[cols.num],as.numeric) sapply(data, class) mlr <- lm(data$TEMP~ .,data=data) summary(mlr) Ouput:



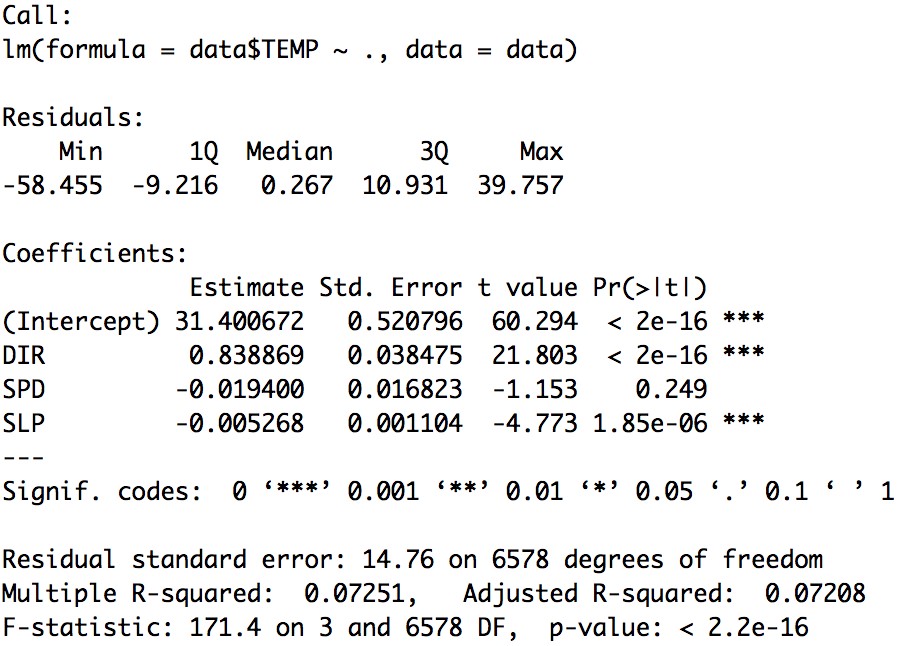
data\_file <- read.csv("~/Downloads/Output\_task/Output\_task0/out1903.txt", sep="") data <- data\_file[,-c(25:33)] data <- data[,-c(13:21)] data <- data[,-c(0,1,2,3,6,7,8,9,10,11,12,14)] cols.num <- c("DIR", "SPD","TEMP", "SLP") data[cols.num] <- sapply(data[cols.num],as.numeric) sapply(data, class) mlr <- lm(data$TEMP~ .,data=data) summary(mlr)

Ouput:



data\_file <- read.csv("~/Downloads/Output\_task/Output\_task0/out1904.txt", sep="") data <- data\_file[,-c(25:33)] data <- data[,-c(13:21)] data <- data[,-c(0,1,2,3,6,7,8,9,10,11,12,14)] cols.num <- c("DIR", "SPD","TEMP", "SLP") data[cols.num] <- sapply(data[cols.num],as.numeric) sapply(data, class) mlr <- lm(data$TEMP~ .,data=data) summary(mlr)

Ouput:



We have attached the R-code and the corresponding output of regression fit in the appendix. Here we have just given the code and corresponding output for 4 years for reference.

Below table gives the Corresponding fit and R square for each year.

* R-square value is high in 1904 of 0.7251 and minimum for year 1937 of 0.001751 value.
* Any new observation with these regressor could predict the temperature value.

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

***Time Taken:*** It took around 1 hour to write a map reduce program to find out mean, median, mode, minimum and maximum temperatures of the data with respect to all years. For graphing all the results, we got it took around one more hour. It took 2 hours of time for us to decide on which variables to fit in regression since there should be a response variable depending on other independent variables.

***Data Refinement:*** Removed the variables which are not required and included only TEMP, SLP, SPD and DIR. Removed all the blank and NA observations before performing the analysis

***Challenges faced:*** We decided to fit regression line for each year. To do that we tried to write a function to execute each year in a row. But got stuck there as our files are not in a single dataframe structure. Then we tried to append files to list and run that list. As it was taking a lot of time in doing this, we run the command for each year separately.

## Task3

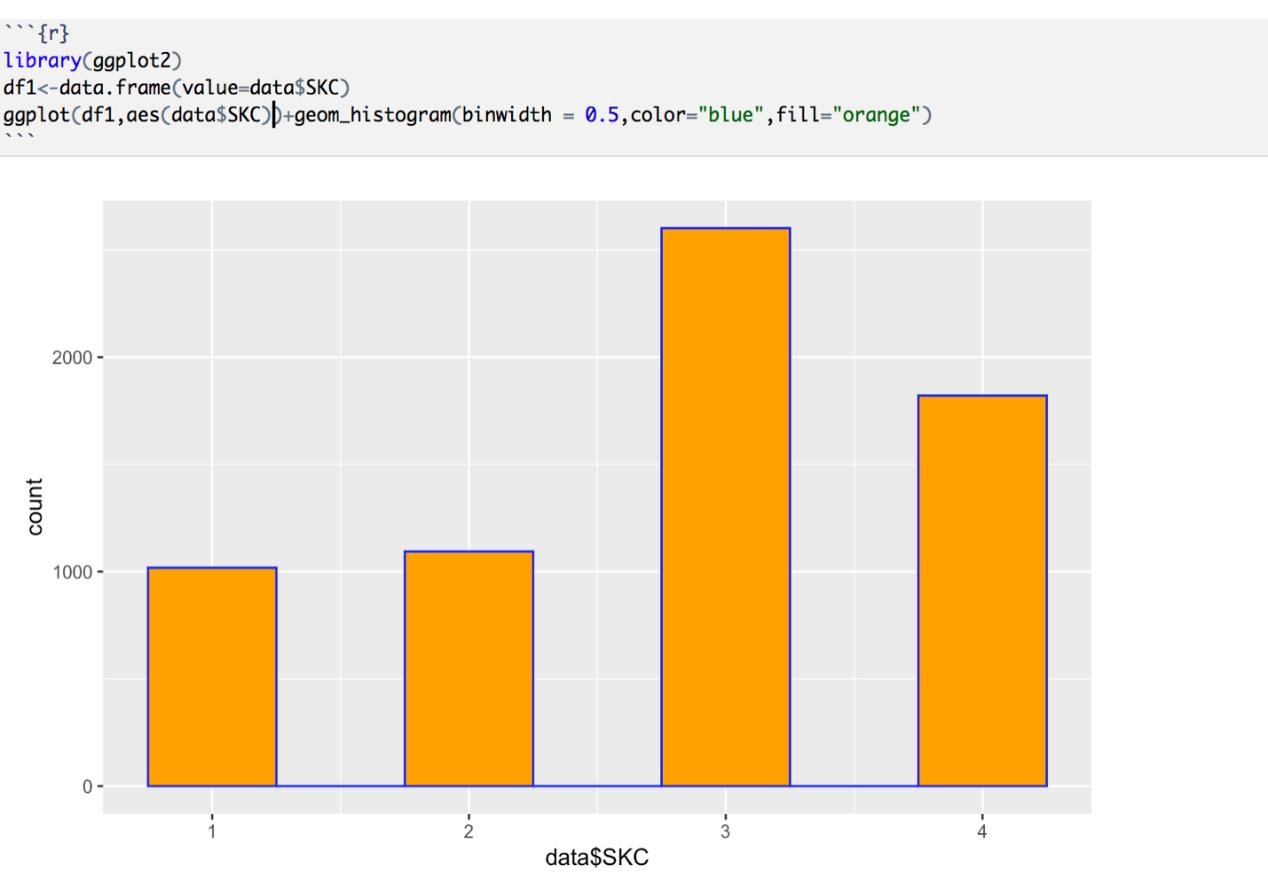
As part of Task 3, We have taken the data of the year 1901 operable stations output file of task 0 and fitted a regression model for Temperature with SKC which is categorical variable with different levels and continuous variable SLP. For a categorical variable cannot fit a normal linear regression model as it is having different levels and will not be an appropriate fit for the dataset. When a categorical variable is observed with multiple levels, we will create dummy variables for analysis. For m levels we create m-1 dummy variables.

In this dataset, there are four different levels in SKC variable throughout the year therefore 3 **Indicator variables** **– dummy variables** will be created. As we have one more regressor variable there would be an interaction between the generated dummy variables and continuous variables. Therefore 3 interaction variables will also be observed in this model.

As analysis in this task involves only the TEMP, SKC and SLP we have filtered out all other variables from the dataset and omitted all blank and NA observations from the dataset.

Here SKC is the type of sky cover which includes CLR- Clear sky, SCT- Scattered, BKN- Broken, OVC- Overcast. In this model we will predict the temperature values when the sky covers are varying.

*#Histogram for SKC variables to know its occurrences*



1-CLR, 2-OVC, 3-SCT, 4-BKN

From the above histogram we can see SCT – scattered sky is observed highly, CLR - clear sky is observed less times. BKN – Broken Sky is observed as the second highest for year 1901.

*#Fitting a model & checking the collinearity between variables.*

|  |  |
| --- | --- |
| **library**(car)  ## Loading required package: carData  data <- **read.csv**("~/Downloads/out1901.csv") cols.num <- **c**( "TEMP", "SLP")  data[cols.num] <- **sapply**(data[cols.num],as.numeric) **sapply**(data, class)  ## SKC TEMP SLP ## "factor" "numeric" "numeric"  mlr <- **lm**(data**$**TEMP**~** .,data=data) **vif**(mlr) | |
| ## GVIF Df GVIF^(1/(2\*Df))  ## SKC 1.114923 3 1.018296  ## SLP 1.114923 1 1.055899 |  |

*There is no multi-collinearity between the regressor variables which is a good thing*.

|  |
| --- |
| *#regression*  slr <- **lm**(data**$**TEMP **~**.,data=data) **summary**(slr)  ##  ## Call:  ## lm(formula = data$TEMP ~ ., data = data)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -73.779 -12.286 0.276 13.908 47.652  ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -270.48682 17.89971 -15.111 <2e-16 \*\*\*  ## SKCCLR -7.34415 0.79740 -9.210 <2e-16 \*\*\*  ## SKCOVC -7.99282 0.67451 -11.850 <2e-16 \*\*\*  ## SKCSCT 0.11854 0.71073 0.167 0.868 ## SLP 0.31246 0.01773 17.620 <2e-16 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 18.11 on 6530 degrees of freedom  ## Multiple R-squared: 0.1017, Adjusted R-squared: 0.1012  ## F-statistic: 184.8 on 4 and 6530 DF, p-value: < 2.2e-16 |

Here we can see “SKC” as categorical variable whose 4 levels are, CLR, OVC,SCT,BKN.

hence taking dummy variables, dummy1, dummy2, dummy3

|  |
| --- |
| data**$**dummy1 <- **ifelse**(data**$**SKC **==** "OVC", 1, 0) data**$**dummy2 <- **ifelse**(data**$**SKC **==** "SCT", 1,0) data**$**dummy3 <- **ifelse**(data**$**SKC**==** "BKN",1,0) data**$**SKC<- **ifelse**(data**$**SKC **==** "CLR", 1, 0) cols.num <- **c**("SKC","dummy1","dummy2","dummy3") data[cols.num] <- **sapply**(data[cols.num],as.numeric) **sapply**(data, class)  ## SKC TEMP SLP dummy1 dummy2 dummy3 ## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"  mlr.dummy <- **lm**(data**$**TEMP **~** SLP**+** dummy1 **+** dummy2**+**dummy3 , data=data) **summary**(mlr.dummy)  ##  ## Call:  ## lm(formula = data$TEMP ~ SLP + dummy1 + dummy2 + dummy3, data = data)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -73.779 -12.286 0.276 13.908 47.652 |
| ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -277.83097 18.01665 -15.421 <2e-16 \*\*\*  ## SLP 0.31246 0.01773 17.620 <2e-16 \*\*\*  ## dummy1 -0.64867 0.68241 -0.951 0.342  ## dummy2 7.46270 0.69578 10.726 <2e-16 \*\*\* ## dummy3 7.34415 0.79740 9.210 <2e-16 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 18.11 on 6530 degrees of freedom  ## Multiple R-squared: 0.1017, Adjusted R-squared: 0.1012  ## F-statistic: 184.8 on 4 and 6530 DF, p-value: < 2.2e-16 |

Dummy1’s p-value is greater than 0.05. Hence dummy1 is not significant variable.

Equation of regression lines for different levels of SKC is, Temp = -277.83097+ 0.31246 SLP -

0.64867 dummy1+ 7.46270 dummy2+7.34415 dummy3

When SKC= CLR, Temp = -277.83097+ 0.31246 SLP

When SKC= OVC, Temp = (-277.83097-0.64867) 0.31246 SLP

When SKC= SCT, Temp = (-277.83097+7.46270)+ 0.31246 SLP

When SKC= BKN, Temp = (-277.83097+7.34415)+ 0.31246 SLP

(-0.64867) is the estimated difference in mean response Temperature when SKC changes from CLR to OVC (7.46270) is the estimated difference in mean response Temperature when SKC changes from CLR to SCT (7.34415) is the estimated difference in mean response Temperature when SKC changes from CLR to BKN.

#with interaction:

|  |
| --- |
| data**$**intrcn1 <- data**$**SLP**\***data**$**dummy1 data**$**intrcn2 <- data**$**SLP**\***data**$**dummy2 data**$**intrcn3 <- data**$**SLP**\***data**$**dummy3  mlr.interaction<- **lm**(TEMP **~** SLP **+** dummy1 **+** dummy2 **+** dummy3 **+** intrcn1 **+** intrcn  2 **+** intrcn3, data=data) **summary**(mlr.interaction)  ##  ## Call:  ## lm(formula = TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 +  ## intrcn2 + intrcn3, data = data)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -74.219 -12.015 -0.252 14.081 48.423 ## |
| ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -493.78217 52.13453 -9.471 < 2e-16 \*\*\*  ## SLP 0.52511 0.05133 10.229 < 2e-16 \*\*\*  ## dummy1 326.42146 57.57421 5.670 1.49e-08 \*\*\*  ## dummy2 80.43638 65.02470 1.237 0.21613  ## dummy3 199.99558 70.36560 2.842 0.00449 \*\*  ## intrcn1 -0.32330 0.05681 -5.691 1.32e-08 \*\*\*  ## intrcn2 -0.07135 0.06411 -1.113 0.26577 ## intrcn3 -0.18956 0.06949 -2.728 0.00639 \*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 18.04 on 6527 degrees of freedom  ## Multiple R-squared: 0.1088, Adjusted R-squared: 0.1078  ## F-statistic: 113.8 on 7 and 6527 DF, p-value: < 2.2e-16 |

From the summary, Equation of regression lines for different levels of SKC with interaction is,

Temp = -493.78217+ 0.52511 SLP+ 326.42146 dummy1+ 80.43638 dummy2+199.99558 dummy3-0.32330 intrcn1-0.07135 intrcn2-0.18956 intrcn3

When SKC= CLR, Temp = -493.78+ 0.52 SLP

When SKC= OVC, Temp = (-493.78+326.42) + (0.52-0.323) SLP

When SKC= SCT, Temp = (-493.78+80.436) + (0.52-0.0713) SLP When SKC= BKN, Temp = (-493.78+199.99) + (0.52-0.189) SLP

* 326.421 is the estimated difference in mean response Temperature when SKC changes from CLR to OVC
* 80.436 is the estimated difference in mean response Temperature when SKC changes from CLR to SCT
* 199.99 is the estimated difference in mean response Temperature when SKC changes from CLR to BKN
* -0.323 is the estimated difference in the slope of the regression lines associating Temp and SLP when SKC changes from CLR to OVC
* -0.071 is the estimated difference in the slope of the regression lines associating Temp and SLP when SKC changes from CLR to SCT
* -0.189 is the estimated difference in the slope of the regression lines associating Temp and SLP when SKC changes from CLR to BKN

we can observe the p-value for dummy2 and interaction 2 is greater than alpha 0.05 which is not significant. Therefore we will refit the model remove the non-significant variables

*#remove dummy2 and interaction 2 based on p-value*

|  |  |
| --- | --- |
| mlr.interaction2<- **lm**(TEMP **~** SLP **+** dummy1 **+** dummy3 **+** intrcn1 **+** intrcn3, data= data) **summary**(mlr.interaction2)  ##  ## Call:  ## lm(formula = TEMP ~ SLP + dummy1 + dummy3 + intrcn1 + intrcn3,  ## data = data)  ##  ## Residuals:  ## Min 1Q Median 3Q Max  ## -71.171 -12.030 -0.382 14.520 46.674  ##  ## Coefficients:  ## Estimate Std. Error t value Pr(>|t|)  ## (Intercept) -384.68685 31.06790 -12.382 < 2e-16 \*\*\*  ## SLP 0.42252 0.03066 13.781 < 2e-16 \*\*\*  ## dummy1 217.32614 39.67495 5.478 4.47e-08 \*\*\*  ## dummy3 90.90027 56.95464 1.596 0.111  ## intrcn1 -0.22071 0.03929 -5.618 2.02e-08 \*\*\* ## intrcn3 -0.08697 0.05638 -1.543 0.123  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 ##  ## Residual standard error: 18.23 on 6529 degrees of freedom | |
| ## Multiple R-squared: 0.0904, Adjusted R-squared: 0.0897  ## F-statistic: 129.8 on 5 and 6529 DF, p-value: < 2.2e-16 |  |

Adj R-square value in this model has decreased so we will consider the original model.

*#Hypothesis Testing for model variables:*

Hypothesis testing is used to verify the variables in the model are significant or not and to determine the probability that a given hypothesis is true.

**linearHypothesis**(mlr.interaction, **c**("dummy1 = 0","dummy2 = 0","dummy3=0","int rcn1=0","intrcn2=0","intrcn3=0"))

## Linear hypothesis test

##

## Hypothesis:

## dummy1 = 0

## dummy2 = 0

## dummy3 = 0

## intrcn1 = 0

## intrcn2 = 0

## intrcn3 = 0

##

## Model 1: restricted model

|  |
| --- |
| ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6533 2239938 ## 2 6527 2125024 6 114914 58.826 < 2.2e-16 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

H0: β2=β3=β4=β5=β6=β7=0 vs H1:atleast one of them !=0 Test statistic = 58.826 p-value = 2.2e-16

Decision:Reject H0. The regression lines associating Temperature and SLP differ by SKC.

|  |
| --- |
| **linearHypothesis**(mlr.interaction, **c**("dummy1 = 0", "intrcn1 = 0"))  ## Linear hypothesis test  ##  ## Hypothesis:  ## dummy1 = 0  ## intrcn1 = 0  ##  ## Model 1: restricted model  ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6529 2136582 ## 2 6527 2125024 2 11558 17.75 2.051e-08 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

Ho2: β2=β5=0 vs H12: atleast one of them is zero Test statistic = 17.75 p-value = 2.051e-08

Decision: Reject Ho2. There is a significant difference in the regression lines between Temperature and SLP when SKC changes from CLR to OVC. Since Ho2 is rejected,

we need to do the following tests:

Ho3: β2=0 vs β2 != 0 Test statistic = 5.670 p-value = 1.49e-08

Decision: Reject Ho3. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to OVC is not significant.

Ho4: β5=0 vs β5 != 0 Test statistic =-5.691 p-value = 1.32e-08

Decision: Reject Ho4. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to OVC is not significant.

|  |
| --- |
| **linearHypothesis**(mlr.interaction, **c**("dummy2 = 0", "intrcn2 = 0"))  ## Linear hypothesis test  ##  ## Hypothesis:  ## dummy2 = 0  ## intrcn2 = 0  ##  ## Model 1: restricted model  ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6529 2168799 ## 2 6527 2125024 2 43775 67.227 < 2.2e-16 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

Ho5=: β3=β6=0 vs H15: atleast one of them is zero

Test statistic = 67.227 p-value = 2.2e-16 decision: Reject Ho5. There is a significant difference in the regression lines between Temperature and SLP when SKC changes from CLR to SCT. Since Ho5 is rejected

we need to do the following tests:

Ho6: β3=0 vs β3 != 0 Test statistic = 1.237 p-value = 0.21613 decision: Failed to reject Ho6. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to SCT is significant.

Ho7: β6=0 vs β6 != 0 Test statistic =-1.113 p-value = 0.26577

Decision: Failed to reject Ho7. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to SCT is significant.

**linearHypothesis**(mlr.interaction, **c**("dummy3 = 0", "intrcn3 = 0"))

## Linear hypothesis test

##

|  |
| --- |
| ## Hypothesis: ## dummy3 = 0  ## intrcn3 = 0  ##  ## Model 1: restricted model  ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6529 2159077 ## 2 6527 2125024 2 34053 52.297 < 2.2e-16 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

Ho8=: β4=β7=0 vs H18: atleast one of them is zero

Test statistic = 52.297 p-value = 2.2e-16

Decision: Reject Ho8. There is a significant difference in the regression lines between Temperature and SLP when SKC changes from CLR to BKN. Since Ho8 is rejected,

we need to do the following tests:

Ho9: β4=0 vs β4 != 0 Test statistic = 2.842 p-value = 0.00449

Decision: Reject Ho9. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to BKN is not significant.

Ho10: β7=0 vs β7 != 0 Test statistic =-2.728 p-value = 0.00639 decision: Reject Ho10. The difference in the intercept of the regression lines between Temperature and SLP when SKC changes from CLR to BKN is not significant.

#test for equal intercept for category OVC and SCT

**linearHypothesis**(mlr.interaction, **c**("dummy1 = dummy2"))

## Linear hypothesis test

##

## Hypothesis:

## dummy1 - dummy2 = 0

##

## Model 1: restricted model

## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##

## Res.Df RSS Df Sum of Sq F Pr(>F)

## 1 6528 2134374

|  |  |
| --- | --- |
| ## 2 6527 2125024 1 9349.9 28.718 8.658e-08 \*\*\* ## --- | |
| ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |  |

Ho11: β2=β3 vs β2 != β3 Test statistic = 28.718 p-value = 8.658e-08

Decision: Reject Ho11. There is a significant difference in the intercept of the regression lines between Temp and SLP when SKC changes from OVC to SCT.

#test for equal slopes for category OVC and SCT

|  |
| --- |
| **linearHypothesis**(mlr.interaction, **c**("intrcn1 = intrcn2"))  ## Linear hypothesis test  ##  ## Hypothesis:  ## intrcn1 - intrcn2 = 0  ##  ## Model 1: restricted model  ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6528 2135026 ## 2 6527 2125024 1 10001 30.719 3.099e-08 \*\*\*  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

Ho12: β5=β6 vs β5 != β6 Test statistic = 30.719 p-value = 3.099e-08

Decision: Reject Ho12. There is a significant difference in the intercept of the regression lines between Temp and SLP when SKC changes from OVC to SCT.

*#test for equal intercept for category SCT and BKN*

**linearHypothesis**(mlr.interaction, **c**("dummy2 = dummy3"))

## Linear hypothesis test

##

## Hypothesis:

## dummy2 - dummy3 = 0

##

## Model 1: restricted model

## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3

|  |  |
| --- | --- |
| ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6528 2126267 ## 2 6527 2125024 1 1243.2 3.8184 0.05073 .  ## --- | |
| ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |  |

Ho12: β3=β4 vs β3 != β4 Test statistic = 3.8184 p-value = 0.05073

Decision: Reject Ho12. There is a significant difference in the intercept of the regression lines between Temp and SLP when SKC changes from SCT to BKN.

*#test for equal slopes for category SCT to BKN*

|  |
| --- |
| **linearHypothesis**(mlr.interaction, **c**("intrcn2 = intrcn3"))  ## Linear hypothesis test  ##  ## Hypothesis:  ## intrcn2 - intrcn3 = 0  ##  ## Model 1: restricted model  ## Model 2: TEMP ~ SLP + dummy1 + dummy2 + dummy3 + intrcn1 + intrcn2 + intrc n3 ##  ## Res.Df RSS Df Sum of Sq F Pr(>F)  ## 1 6528 2126264 ## 2 6527 2125024 1 1240 3.8086 0.05103 .  ## ---  ## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1 |

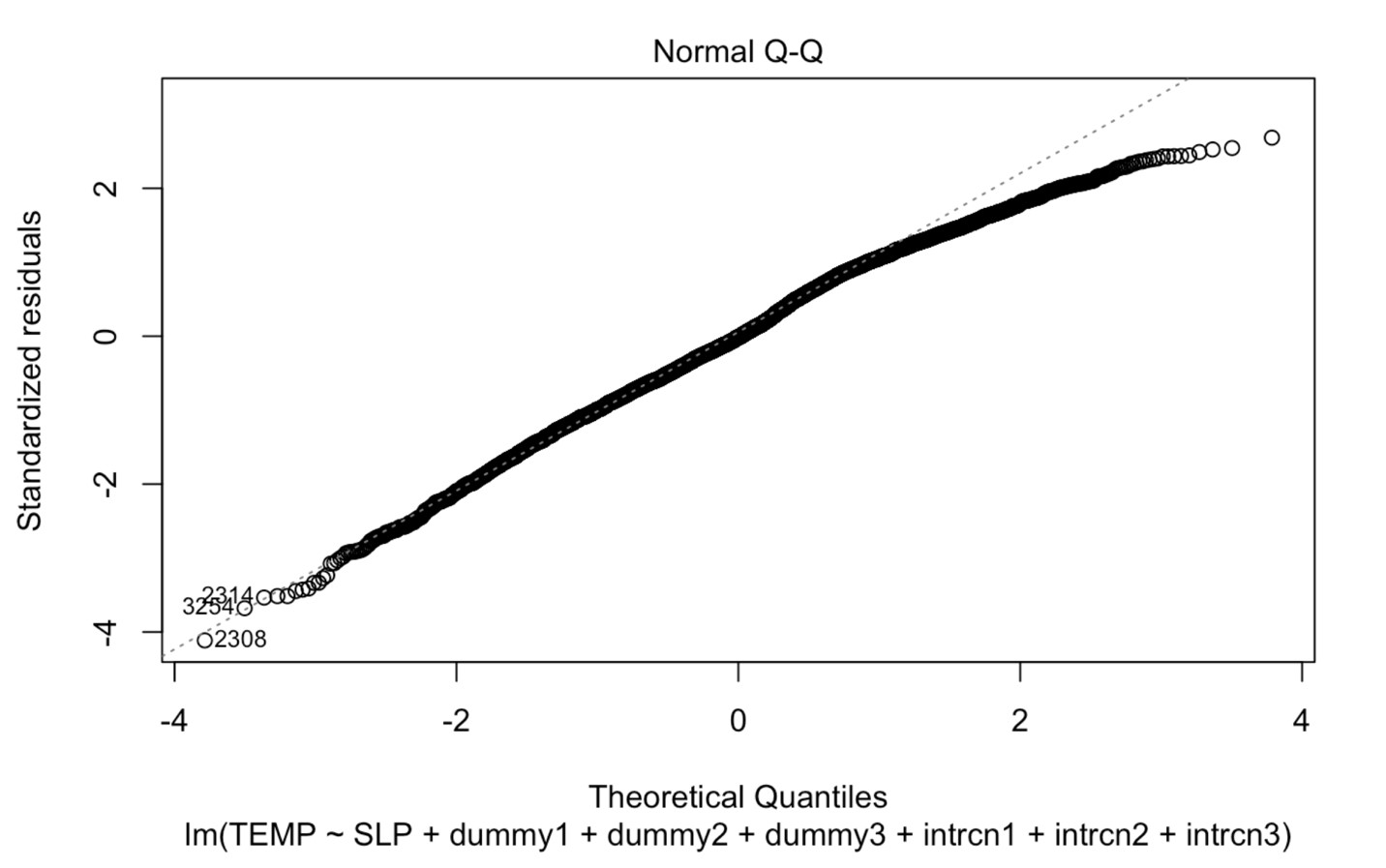
Ho13: β6=β7 vs β6 != β7 Test statistic = 3.8086 p-value = 0.05103

Decision: Reject Ho13. There is a significant difference in the intercept of the regression lines between Temp and SLP when SKC changes from SCT to BKN.

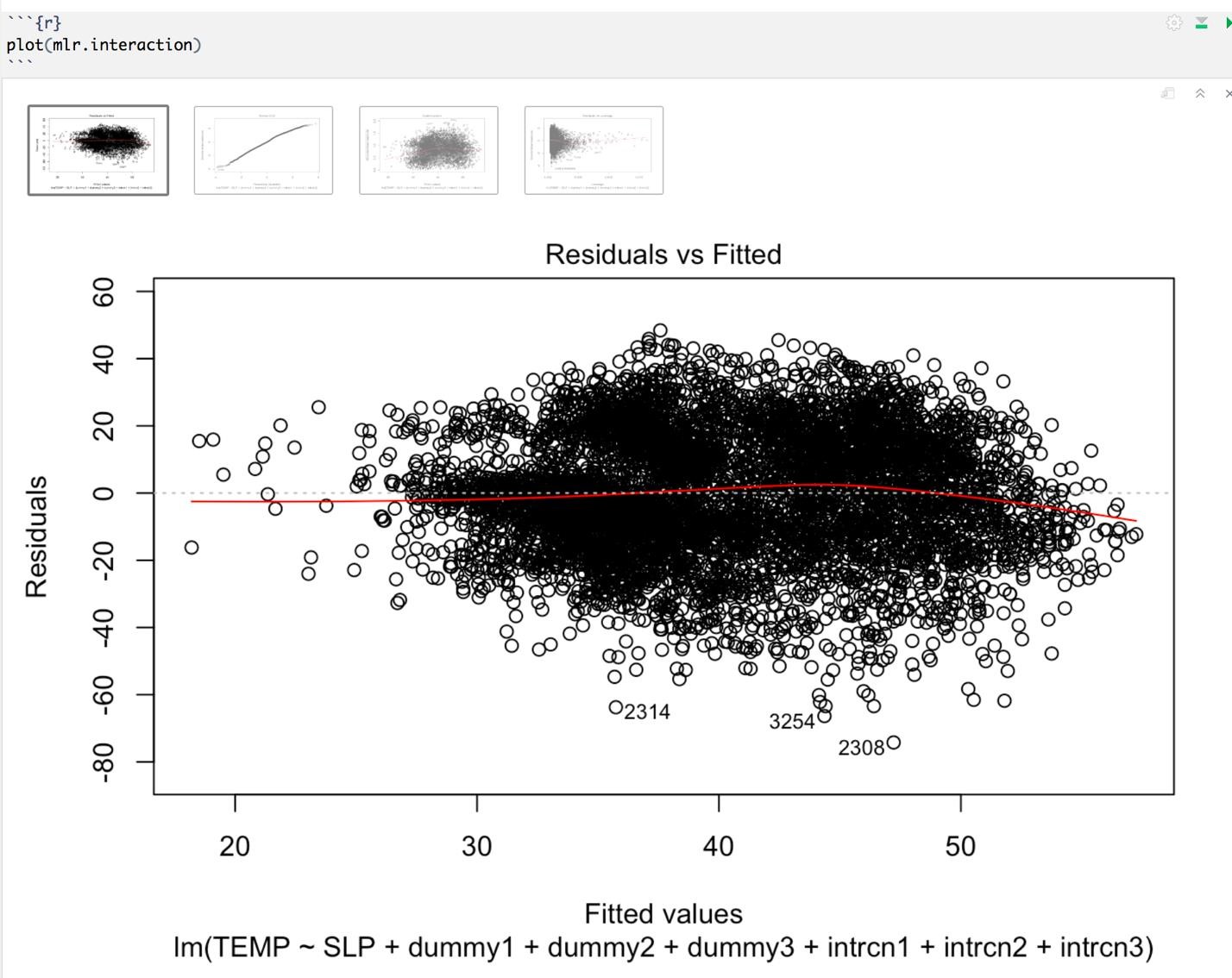
From the hypothesis testing, dummy 2 and interaction 2 are not significant in the model.

**Plots:**

*#Qq plot for Residuals:*



This plot shows if the data is meeting the assumption of normality. Given dotted line is the normal line and data points on plot refers our fit. This shows that our data is following normal distribution. Here data is standardized residuals.



We can see that the mean residuals is almost zero throughout the plot. This indicates that our fit is appropriate as sum of mean residuals is always zero.

***Time Taken:*** 30 minutes for data cleaning, 1 hour for fitting the models, 1 hours performing hypothesis testing and 20 minutes for working on report.

***Data Refinement***: Removed the variables which are not required and included only TEMP, SLP and SKC. Removed all the blank and NA observations before performing the analysis. After data cleaning the dataset consists of 6535 observations and 3 variables. Subsequently as part of this task we have generated dummy1, dummy2, dummy 3, intrcn1, intrcn2, intrcn 3 into data frame.

***Challenges faced:*** We got stuck while creating dummy variables as SKC data is in ‘factor’ datatype. It took time for us to figure out this. We tried and realized that we need to change the datatype. After changing the data type, we could perform remaining analysis. Firstly, when we tried the fit to model, we could see the coefficients values as NA and didn’t get the output as expected. After removing the blank values, we were able to fit the model.

## Task 4

In task 4, We have chosen the data file of year 1928 for performing regression analysis. The reason behind in selecting this particular year is that we observed lowest mean temperature is recorded for year 1928 and to see how the model would be fitted for predicting any new observations. This task involved only TEMP, SPD, VSB and SLP variables and filtered out all rest of the variables from dataset for analysis. Removed blank and NA observations to avoid conflicts while fitting the model.

Here we used Knn model which mean **K-nearest neighbors**. Our goal is to predict the temperature when regressor variables are continuous. Usually Knn model can be used both for classification and regression variables. As our data set size is large there would be overfitting when a model is fitted. Overfitting in model can be avoided by partitioning the data and developing the model using the partitions. In partitioning we divide the data into training, validation and test datasets.

Training data is largest partition containing the data used for building the models that will be examined. In Knn model we use validation data for choosing optimal k value.

Data<- **read.csv**("C:/Users/ysree/Downloads/project/out1928.csv") Data<- Data[,**-**4]

*##function for partitioning the data into training, validationa and test data*

|  |
| --- |
| partition.3 <- **function**(data, prop.train, prop.val){  *# select a random sample of size = prop.train % of total records*  selected1 <- **sample**(1**:nrow**(data), **round**(**nrow**(data)**\***prop.train), replace = F ALSE)  *# create training data which has prop.train % of total records*  data.train <- data[selected1,]  *# select a random sample of size = prop.val % of the total records* rest <- **setdiff**(1**:nrow**(data), selected1)  selected2 <- **sample**(rest, **round**(**nrow**(data)**\***prop.val), replace = FALSE)  *# create validation data which has prop.val % of total records* data.val <- data[selected2,]  *# create testing data with the remaining records* data.test <- data[**setdiff**(rest, selected2),]  **return**(**list**(data.train=data.train, data.test=data.test, data.val=data.val)) |

}

*#Partitioning the 1928 observations into training, validation and test data sets with proportion rate of 60%, 20% and 20%.*

p3 <- **partition.3**(Data, 0.6, 0.2) mydata.train <- p3**$**data.train mydata.val <- p3**$**data.val mydata.test <- p3**$**data.test

*#To equalize the scales of the various predictors we are rescaling the variables*

|  |
| --- |
| **library**(scales)  train.norm <- p3**$**data.train val.norm <- p3**$**data.val test.norm <- p3**$**data.test **for** (i **in** 2**:**4){  train.norm[,i] <- **rescale**(p3**$**data.train[,i])  val.norm[,i] <- **rescale**(p3**$**data.val[,i], from = **c**(**min**(p3**$**data.train[,i]), **m ax**(p3**$**data.train[,i])))  test.norm[,i] <- **rescale**(p3**$**data.test[,i], from = **c**(**min**(p3**$**data.train[,i]), **max**(p3**$**data.train[,i])))  } |

*#Fitting a KNN model for the training and validation data*

|  |  |
| --- | --- |
| **library**(FNN)  Knn = **knn.reg**(train = train.norm[,2**:**4], test = val.norm[,2**:**4], | |
| y=train.norm[,1], k=4) |  |

*#finding mean square error on validation data*

val.mse <- **mean**((Knn**$**pred **-** val.norm[,1])**^**2) val.mse

## [1] 143.1085

*#fit k-nn model with k values from the following list to find the optimal k value*

|  |
| --- |
| klist <- **c**(1, 5, 10, 20, 100, 300, 500, 800, 1000, 2000) val.mse <- **rep**(NA, **length**(klist)) **for** (i **in** 1**:length**(klist)){  Knn = **knn.reg**(train = train.norm[,2**:**4], test = val.norm[,2**:**4], y=train.norm [,1], k=klist[i])  *### find mse on validation data*  val.mse[i] <- **mean**((Knn**$**pred **-** val.norm[,1])**^**2) **cat**("K", klist[i], "Validation MSE", val.mse[i], "\n") }  ## K 1 Validation MSE 228.6494  ## K 5 Validation MSE 137.6653  ## K 10 Validation MSE 131.8767  ## K 20 Validation MSE 130.0481  ## K 100 Validation MSE 136.8682  ## K 300 Validation MSE 142.8984 ## K 500 Validation MSE 148.57  ## K 800 Validation MSE 155.2114  ## K 1000 Validation MSE 159.3986  ## K 2000 Validation MSE 172.122 |

We need to choose the k value with least MSE for better fit.

From the K values list, less mse value is observed for K=20, therefore optimal k is taken as 20.

Test data is used for assessing the performance of the chosen model with new data. Now we will evaluate the model performance on test data

Knn = **knn.reg**(train = train.norm[,2**:**4], test = test.norm[,2**:**4], y=train.norm[,1], k=20)

*#finding MSE on test data*

test.mse <- **mean**((Knn**$**pred **-** test.norm[,1])**^**2) test.mse

## [1] 140.463

Now we have tested the performance of the model on the test data where we can see the MSE value as 131.52 which is less.

From this model, the predicted response value for a new record can be computed by taking the average of the response values nearest to the 20 neighbors.

***Time Taken:*** 45 minutes for data exploring, data cleaning figure out what analysis to perform, 2 hours for data partitioning and fitting the models and 15 minutes for working on report.

***Data Refinement:*** Removed the variables which are not required and included only TEMP, SLP, SPD and VSB. Removed all the blank and NA observations before performing the analysis. After data cleaning the dataset consists of 5992 observations and 4 variables.

***Challenges faced:*** Faced issue while writing the function for partitioning of data. Figured out the solution after multiple tries and discussing among team members.

**Summary & Efforts Breakdown:**

### TASK-0

In this task, the step was to get the data on our local so that all the tasks can be performed without issues. After getting all the data we wrote Map-reduce programs and faced some issues as it was not giving some of the required features in the output. Eventually, we came up with the correct Map-Reduce program which gave the required output.

### TASK-1

In this task, the output of task0 was used as input here as we only needed filtered data. The MapReduce program was written for this task with any problem and it gave the desired results.

### TASK-2

In this task, it took one hour to do the R analysis. After we got the MapReduce result, we combine 4 txt files to one CSV file. We use that CSV file to do the linear regression analysis and we generated the plots and R^2

### TASK-3

In this task, we have taken Task 0 output file of specific year 1901. We have explored the variables in dataset and decided to work on SKC which is categorical variables with 4 levels which involves the concept of **Indicator variables – dummy variables**. We have fitted the regression model for the data which states the relation and effect of multiple levels of SKC variables on Temp variable.

### TASK-4

In this task, we have taken the output file of year specific 1928 which is recorded with low mean in the output of Task 2. We have explored the variables in the data and came up with fitting model using KNN model which involves the data partitioning and finding the optimal K values. From the model we can predict the response value of a new record by taking average response values with optimal K value.