Project MM923
Hassan Abdirauf Ahmed
Weather
Project Tasks:

Section 1: Data Management

The data provided is not in a *tidy* format, because it does not satisfy three conditions.

- Each observation must have its own row,
- Each variable must have its own column,
- Each value must have its own cell. [1]

We can see that the variable "Measure_of_Temp" has three different variables which violates the second condition above. The three variables "forecast_hours_before", "forecast_temp" and "observed_temp". Each of these three variables must have their own columns. This is done by using using calling "pivot.wider() function, this ensures that each variable has its own column [1]. Another violation of the conditions mention before is the variable "State_City". This variable has the code for the state and city both separated by ";". Therefore, the second condition is not met meaning the "State" must have its own column and "City" must have its own as well. This is done by using the "separate()" function. The code is,

```
weather_forecast %>%
  pivot_wider(names_from = Measure_of_Temp, values_from = Temp) %>%
  separate(State_City, c("State", "City"), ":") %>%
  separate(date, c("Year", "Month", "Day"),"-")-> weather_forecast_tidy
```

These turn the "weather_forecast" data into a tidy data called "weather_forecast_tidy". From here we will be using "weather_forecast_tidy" strictly.
The last line in the code separates the "Year", "Month" and "Day" to allow for visualization trends. "Year", "Month" and "Day" will have their own columns.

Section 2: Data Cleaning

Since we are interested in measurements which provide reliability, we must remove observations which may contain errors. The last column in "weather_forecast_tidy" is "possible errors" this has observations which are either "none" meaning no errors or the name of the possible variable that's causing it. The code that removes these errors is,

```
weather_forecast_tidy %>%
  filter(possible_error == "none") ->weather_forecast_tidy
```

This uses the filter() function which takes the observations which have only 'none' and removes the ones with the possible errors.

There are many missing observations in "observed_percip". We also have been informed that any missing observations "NA" mean that there was no rain recorded on that that day. Meaning that we replace them with "0.00". The code to replace them is,

```
weather_forecast_tidy %>%
  mutate(observed_precip=ifelse(is.na(observed_precip), 0.00, observed_precip))->
weather_forecast_tidy
```

This uses the mutate function where the "observed_percip" is equal to an "ifelse" statement which has the condition that if we find any "NA" value in "observed_percip" we replace it 0.00 and else leave it as it is. This mutates/changes the "observed_percip" variable in our "weather_forecasts_tidy" data.

<u>Section 3 Exploring Data</u>

<u>Part 1</u>

The mean of "observed_temp" is 48.0 and the variance of "observed_temp" is 365 which are given by this line of code,

```
weather_forecast_tidy %>%
  summarise(mean(observed_temp, na.rm = TRUE))
weather_forecast_tidy %>%
  summarise(var(observed_temp, na.rm = TRUE))
```

This is done using the summarise function which finds the mean and variance of the "observed_temp" whilst removing any "NA" or missing observations.
Figure 1 shows the histogram of observed temperature, the histogram can be said to be symmetrical, this gives us an indication of the distribution of the observed temperature [7].

<u>Part 2</u>

The coefficient matrix of "observed_precip", "observed_temp" and "forecast_temp" is,

$$\begin{array}{l c}
 & \text{observed\_precip observed\_temp forecast\_temp} \\
\begin{array}{l}\text{observed\_precip}\\ \text{observed\_temp}\\ \text{forecast\_temp}\end{array} & \begin{bmatrix} 1 & 0.04356879 & 0.04543923 \\ 0.04356879 & 1 & 0.98675205 \\ 0.04543923 & 0.98675205 & 1 \end{bmatrix}
\end{array}$$

This shows that "observed_precip" and "observed_temp" is 0.04356879 which is not linearly correlated, as well as "observed_precip" and "forecast_temp" which is 0.04543923 which not linearly correlated. However, "observed_temp" and "forecast_temp" is 0.98675205 which means that they are highly positively correlated. The code used is,

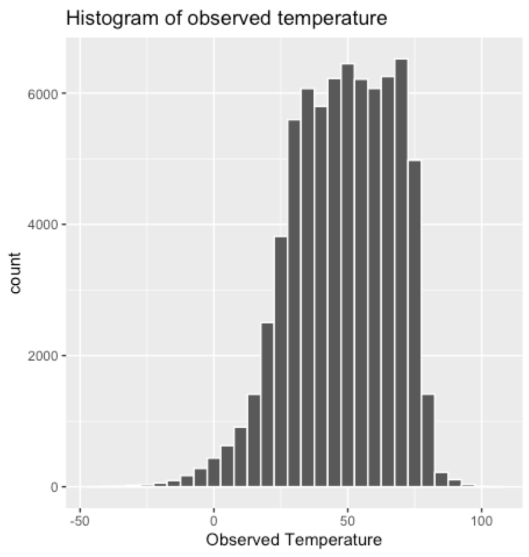

Figure 1: Histogram of observed temperature

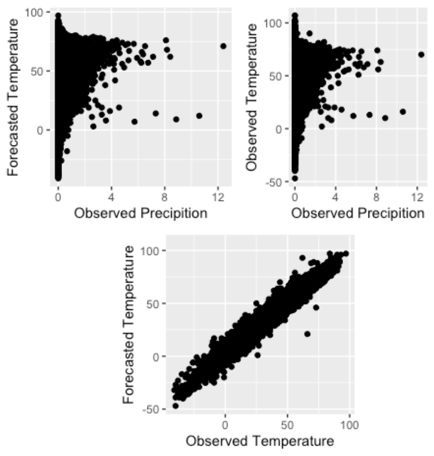

Figure 2: Three scatter plot of forecasted temperature (top left) and observed precipitation, observed temperature and observed precipitation (top right), forecasted temperature and observed temperature (bottom).

```
my_corr_data <- weather_forecast_tidy[c(7,11,12)]
cor(my_corr_data,use="complete.obs", method= "spearman")
```

This stores my three variables of interest in a new variable and then uses the "cor()" function to find the Spearman's correlations with the removal of any missing observations ("NA")

This is also summarised by Figure 3, which clearly visualises the correlations between the three different variables. The code used is,

```
ggplot(weather_forecast_tidy)+
  geom_point(aes(observed_precip, forecast_temp))+
  labs(x= "Observed Precipition", y= "Forecasted Temperature")  -> p1
ggplot(weather_forecast_tidy)+
  geom_point(aes(observed_precip, observed_temp))+
  labs(x= "Observed Precipition", y= "Observed Temperature")  ->p2
ggplot(weather_forecast_tidy)+
  geom_point(aes(forecast_temp, observed_temp))+
  labs(x= "Observed Temperature", y= "Forecasted Temperature") ->p3
layout <- "
AABB
#CC#
"
p1+p2+p3+plot_layout(design=layout)
```

This uses the "ggplot()" and "geom_point() "to plot scatter graphs of the 3 variables then give them each different variables such as "p1", "p2" and "p3" such that we can see the plots together with a specific design which is shown using "layout".

Part 3

The mean and standard deviation of observed precipitation in New York is 0.117 and 0.367 respectively. This is done by,

```
weather_forecast_tidy %>%
  group_by(State) %>%
  filter(State== "NY") %>%
  summarise(Mean=mean(observed_precip), Standard_Deviation= sd(observed_precip))
```

This groups the data by its “State” filters the state needed “NY” and then summarises the mean and standard deviations.

<u>Part 4</u>

The average observed temperature of the tri-state area (New York, New Jersey, Connecticut) are shown by three different graphs in the Figure 3. The top line graph is the average observed temperature against the year. The bottom graphs are both show the average observed temperature against the months, but the bottom left is in 2021 and the bottom right is 2022.
The year graph shows a decrease in average observed temperature between 2021 and 2022 in all three states, they seem to also be parallel meaning the are all decreasing at the same rate. However, this may not provide much information, so we proceed to look at each year in more depth and they both show a similar trend where their minimum is in January and both peak in August in the tri-state area. But Connecticut observations stopping in May. In all three-line graphs we see that New Jersey with the highest average observed temperature and New York with the lowest average observed temperature.

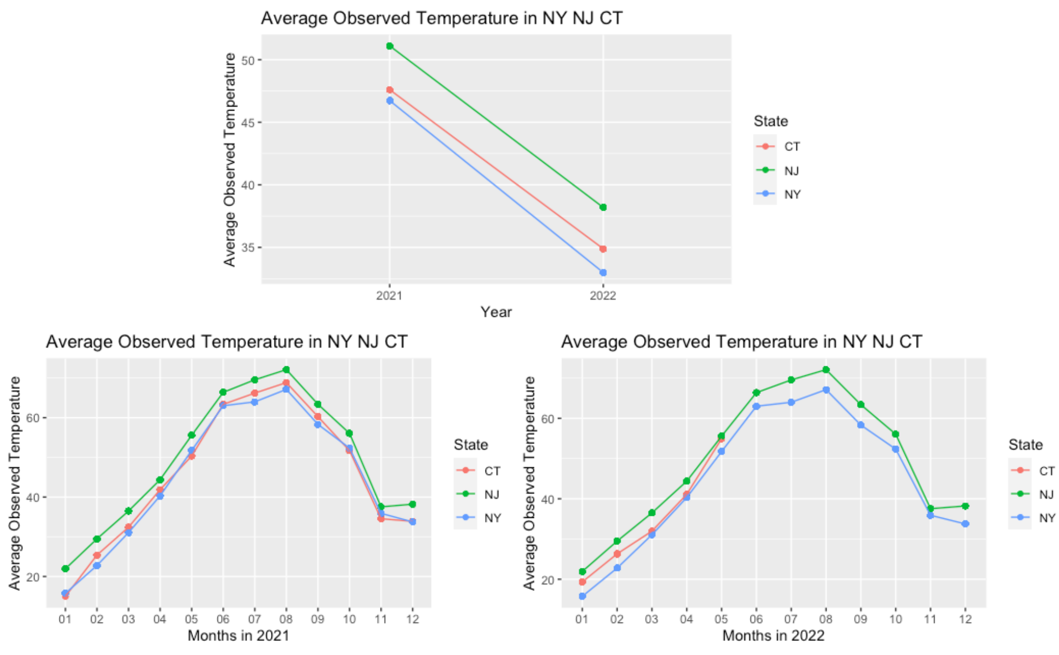

Figure 3: Average Observed Temperature in the tri-state area in Years (top), Months of 2021 (bottom left) and Months in 2022 (bottom right)

The code for this part is,

```
#Average observed temperature in Years
weather_forecast_tidy %>%
  group_by(State,Year) %>%
  filter(State=="NY"|State=="NJ"|State=="CT") %>%
  mutate(mean1= mean(observed_temp, na.rm = TRUE)) %>%
  ggplot(aes(Year, mean1, colour=State , group= State))+
  geom_point()+
  geom_line()+
  labs(x="Year",
       y="Average Observed Temperature",
       title = "Average Observed Temperature in NY NJ CT") ->p4

#Average observed temperature in Months in 2021
weather_forecast_tidy %>%
  group_by(State, Month) %>%
  filter(State=="NY"|State=="NJ"|State=="CT" & Year == 2021) %>%
  mutate(mean1= mean(observed_temp, na.rm = TRUE)) %>%
  ggplot(aes(Month, mean1, colour=State , group= State))+
  geom_point()+
  geom_line()+
  labs(x="Months in 2021",
       y="Average Observed Temperature",
       title = "Average Observed Temperature in NY NJ CT") ->p5

#Average observed temperature in Months in 2022
weather_forecast_tidy %>%
  group_by(State, Month) %>%
  filter(State=="NY"|State=="NJ"|State=="CT" & Year == 2022) %>%
  mutate(mean1= mean(observed_temp, na.rm = TRUE)) %>%
  ggplot(aes(Month, mean1, colour=State , group= State))+
  geom_point()+
  geom_line()+
  labs(x="Months in 2022",
       y="Average Observed Temperature",
       title = "Average Observed Temperature in NY NJ CT")->p6

layout1 <- "
#AA#
BBCC
"
p4+p5+p6+plot_layout(design=layout1)
```

The procedure is the same in all three, first use group.by() to group the states ,the year, the months in 2021 and 2022. Then filter out the three states using the filter() function with the year if looking at the specific year. So “Year == 2021” for months in 2021 and “Year == 2022” for months in 2022. Using mutate() to find the mean of the observed temperature without any missing values. Then use “ggplot()” to plot both points (“geom_point()”) and lines (“geom_line()”) of each state using “colour=State”. Finally like in Figure 3 we use layout() for a better visualization of the plots.

Part 5

The function called “weather_summary” which takes the “weather_forecast_tidy” as an input and returns a named list of the average forecast and observed temperature over time. Which is then tested by the full “weather_forecast_tidy” data and the “weather_forecast_tidy”data in year 2021. The code below shows the function “weather_summary”.

```
weather_summary <- function(x){
  mean_forecast_temp = mean(x$forecast_temp, na.rm = TRUE)
  mean_observed_temp = mean(x$observed_temp, na.rm = TRUE)
  date1 = min(x$Year)
  date2 = max(x$Year)
  time_interval = as.integer(date2) - as.integer(date1)
  if(time_interval < 1){
    warning("The interval of the year is not greater than 1.")
  }else{
  mean_forecast_temp_time = mean_forecast_temp / time_interval
  mean_observed_temp_time =  mean_observed_temp / time_interval
  list("Average observed temperature over time = " = mean_observed_temp_time,
      "Average forecast temperature over time = " =mean_forecast_temp_time)
  }
}
#Test with full weather_forecast_tidy
weather_summary(weather_forecast_tidy)

#Test with full wather_forecast_tidy in 2021 only.
Year_2021 <-weather_forecast_tidy[weather_forecast_tidy$Year == "2021",]
weather_summary(Year_2021)
```

Firstly, we take the mean forecast temperature and the mean observed temperature, whilst removing the missing variables. Then we find the mean forecasted temperature over time which is chosen to be the yearly. Similarly, we do the same with the mean observed temperature over time. However, if we find that the year is less than one, we provide a “warning()” that the time interval is not greater than one as the denominator cannot be less than 1.
This is then tested on the “weather_forecast_tidy” data which gives these results.

```
$`Average observed temperature over time = `
[1] 48.02578

$`Average forecast temperature over time = `
[1] 46.54
```

While a warning message is given if we only look at year 2021.

```
Warning message:
In weather_summary(Year_2021) :
The interval of the year is not greater than 1.
```

Section 4: Building a model.

The fitted linear regression model for observed temperature based on "high_or_low", "forecast_hours_before", "observed_precip" and "forecast_temp" is,

$$observed\_temp = \beta_0 + \beta_1 \times \ highorlow \ + \beta_2 \times \ forecasthoursbefore + \beta_3 \times observedprecip + \beta_4 \times forecasttemp \qquad (1)$$

Since the variable "high_or_low" is a categorical we make a dummy variable or an indicator variable [2] where low will be 0 and high will be 1. This new variable will be named "new_high_low, the code is shown in Appendix 1,
we use the "mutate()" function then a ifelse statement where if low we make it 0 or else we make it a 1 meaning high will be 1.
Then we have 2 cases, when "new_high_low" = 1

$$observedtemp = \beta_0 + \beta_1 \times newhighlow \ + \beta_2 \times \ forecasthoursbefore + \beta_3 \times observedprecip + \beta_4 \times forecasttemp + \beta_5 \times \ forecasthoursbefore \times newhighlow \ + \beta_6 \times observedprecip \times newhighlow \ + \beta_7 \times forecasttemp \times newhighlow \qquad (2)$$

meaning they have different slopes and different intercepts.

When "new_high_low" = 0

$$observedtemp = \beta_0 + \beta_2 \times \ forecasthoursbefore + \beta_3 \times observedprecip + \beta_4 \times forecasttemp \qquad (3)$$

Meaning we would have the same intercept, β_0, but different slopes, β_2, β_3 and β_4.

The formatted ANOVA is shown below in the Table 1. This shows our F is 380467.443, with the regression having 7 degrees of freedom
We will also test if the slope coefficient of the forecasted temperature is equal to 1 using a 2-sided t-test [3].
First, we must set up our hypothesis test [3]:

$$H_0: \beta_4 = 1$$
$$H_A: \beta_4 \neq 1$$

With a 95% confidence interval. This is done very quickly with R,
where $\hat{\beta}_5 = 0.9812159$, $\mathrm{ese}(\hat{\beta}_5) = 0.0006112$, $\frac{\alpha}{2} = 0.025$, $\mathrm{n - p - 1 =}\ 72183$

```
t_star <- (0.9812159 - 1)/0.0006112
t_star
[1] -30.73315
t_stat <- qt(1-0.025, 72186)
t_stat
[1] 1.959997
```

Now we must test if ,

$$|t^*| > t_{72183,0.025} \qquad [3]$$

Therefore,

$$|-30.73315| > 1.959997$$

Table 1: The ANOVA of the observed temperature

Subject	Degrees of Freedom	Sum of squares	Mean sum of Squares	F
Regression	7	25489126	3641303.714	380467.443
Residual	72183	690835	9.570605267	
Total	72190	26179961		

Thus, we reject our null hypothesis in favour to our alternative, meaning that the coefficient of the forecasted temperature, β_5, is not equal to 1.

Variable Selection is done by taking our most complicated model (M4) and comparing it to M3a. This is done either with a t-test or an F test. We will be using the F-test and drop1() function [3].

We look to compare model M4 which has all the interactions with M3a and M3b as shown in Appendix 2,this code results in removing "new_high_low" which has p value of 0.50177, which is our indicator variable, "forecast_hours_before*new_high_low" which has p value of 0.95571, "observed_precip*new_high_low" which has p value of 0.06636, and "forecast_temp*new_high_low"which has a p value of 0.69785.
This gives us a model M2b which is,

$$observedtemp = \beta_0 + \beta_2 \times\ forecasthoursbefore + \beta_3 \times observedprecip + \beta_4 \times forecasttemp \qquad (4)$$

Therefore, we do the F test again and compare it to M1 (as shown in Appendix 3). All independent variables have p-values less than 0.05 meaning all are significant. This means that our model M2b is the best compared to the M1 model. Which is also confirmed using a stepwise selection process. As shown in Appendix 4

In addition, we now look at any transformations that could improve our model. We first look at the,

- Errors are normally distributed; this is done by looking at a QQ-Plot
- Errors have constant variance; this is done by looking a Residuals vs Fitted Plot
- Errors are independent; this is done if again looking at a Residuals vs Fitted Plot [4]

The QQ-Plot in Figure 4 shows both tails deviating from the line, tending to be a heavy tailed, therefore not normal. However there seems to be no real pattern meaning that it has a constant variance and independence.

Looking at each independent variable and trying to improve the normality of the errors.
From the top plots in Figure 5 shows that the errors are quite normal, but there is a clear grouping in the residuals vs fitted plot meaning no constant variance or independence. The middle plots show that the errors are normal but again there is a clear pattern of funnelling in as majority of the points are on the left and very few on the right. The bottom plots show the same patterns as Figure 4.

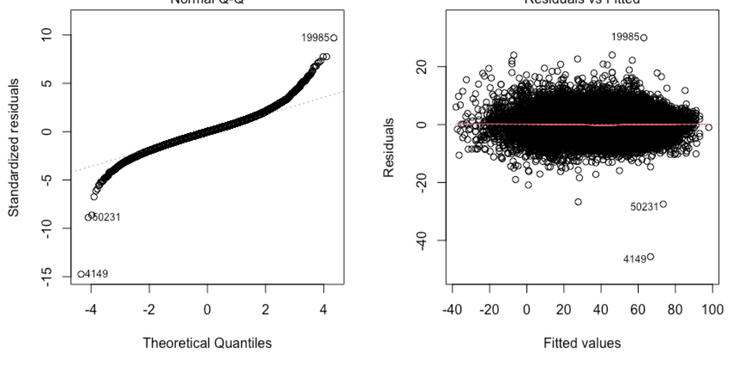

Figure 4: Normal QQ-plot (left) and Residuals Vs Fitted (right)

Therefore, no normality, constant variance, or independence.
We now look at any transformations that could improve our assumption. However, no transformations such as sine, cosine, natural logs, exponentials, roots, and powers had any effect on the plots for the better. Meaning there was no improvement.

Comparing the original model (Equation 2) and our model (Equation 4) we see that it is the when the categorical variable has the observation low meaning zero. The adjusted R^2 of the full model is 0.9736095 whereas our final model the adjusted R^2 is 0.9736068 where the difference is 0.0000027 which is very small. But our final model simpler than of our full model using Occam's Razor the need for simplicity in the model [5].

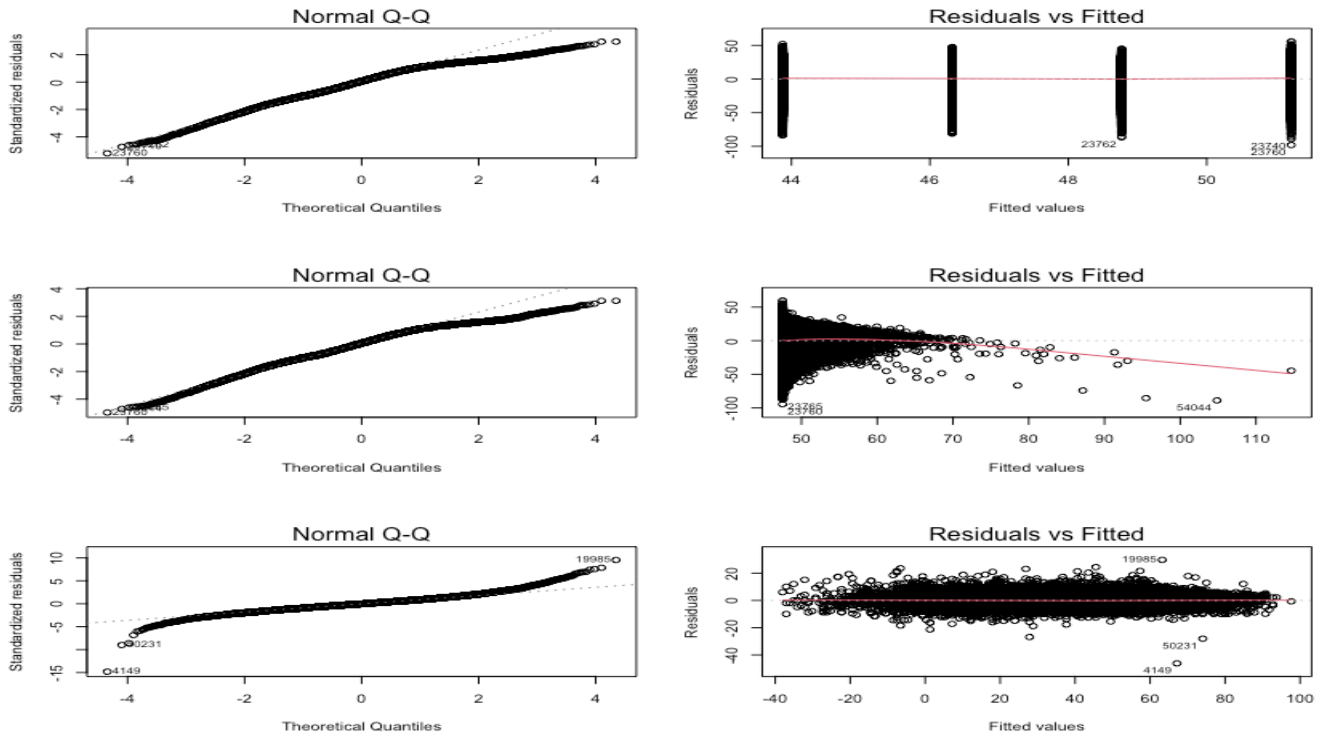

Figure 5: Top Normal QQplot (left) and Residuals Vs Fitted (right) for "forecast_hours_before" Middle Normal QQplot (left) and Residuals Vs Fitted (right) for "observed_precip" Bottom Normal QQplot (left) and Residuals Vs Fitted (right) for "forecast_temp"

Section 5: Interpreting the model

Part 1

The goodness of fit of the model is tested by using the Prediction Sum of Squares (PRESS), this measures how unseen data fit with our regression model [6]. The Lower the PRESS, the better the prediction or lower prediction error [6]. We will also use Akaike's Information Criteria (AIC) which also assess the models fit for prediction [5]. We are looking to minimise the AIC. The adjusted R^2 can also be used to assess our models [5].

We will use these three tools to assess the goodness of fit of our model (Equation 4) and compare that to the first full model (Equation 2) we started with. This is done by a function created,

```
Press_adjR2_AIC <- function(x){
  PRESS <- sum(x$resid^2/(1-hatvalues(x))^2)
  Summary1<- summary(x)$adj.r.squared
  AIC1<- AIC(x)

  list("PRESS"=PRESS, "AdjustedR2"=Summary1, "AIC"=AIC1)

}
Press_adjR2_AIC(ot_lm1)
Press_adjR2_AIC(ot_lm)
```

Table 2 perfectly summarises the PRESS, Adjusted R^2 and AIC of two models. We can compare the two and see that press of the final model is lower than that of the full model. The adjusted R^2 is also lower but again given there are three more variables in the full model we should expect the adjusted R^2 to increase as we increase the number of variables in the model. The AIC is also higher by 3.2 units. Overall, we would conclude that the final model is a good fit of model.

The final model of with coefficients are,

$$\begin{aligned} observedtemp = 1.5316121 + 0.0268948 \times forecasthoursbefore \\ - 0.2997679 \times observedprecip + 0.9811933 \times forecasttemp \end{aligned}$$

Table 2: PRESS, Adjusted R^2 and AIC of two models.

Model	PRESS	Adjusted R^2	AIC
Final Model with 4 variables (Equation 4)	691022.6	0.9736068	367940
Full Model with 7 variables (Equation 2)	691296.3	0.9736095	367936.8

Part 2

The intercept is the value we have when the independent variable equal to zero meaning that our intercept being 1.5316121°F means that our observed temperature without any influence from the independent variables will be 1.5316121°F.
The coefficient for "forecast_hours_before" is 0.0268948, which means as increase the forecast between observed temperature and the forecasted, the observed temperature will also increase by 0.0268948 making the temperature warmer.
The coefficient for observed precipitation is -0.2997679, which means if i increase the observed precipitation the cooler or decrease in observed temperature. Which is reasonable as the rain cools down temperatures.
The coefficient of forecast temperature is 0.9811933, as we increase the forecast temperature the warmer or increase in the observed temperature.

The 95% confidence interval for the:

- intercept is (1.45709858,1.60612556),
- "forecast_hours_before" is (0.02527993,0.02850967),
- observed precipitation is (-0.37019756, -0.22933825),
- forecast temperature is (0.97999600, 0.98239063).

This is done by using the confint() function.

```
confint(ot_lm1)
```

Part 3

If the forecasted temperature is 40°F then the expected temperature is 40.77934°F given that the observed precipitation is 0 and the forecast hours before is also 0.
This is done using predict() function.

```
predict(ot_lm1, newdata = data.frame(forecast_hours_before= 0, observed_precip = 0, forecast_temp= 40) ,
      interval = "confidence", level = 0.95)
```

Part 4

The plausible range of values for the temperature when the forecasted temperature 53°F and the observed precipitation is 0.01 is (47.46775, 59.59597) with a 95% significance level. Meaning it may be as low as 47.47°F and as high as 59.60°F. This is also assuming the forecast hours before is also 0.
We use the predict() function again.

```
predict(ot_lm1, newdata = data.frame(forecast_hours_before= 0, observed_precip = 0.01, forecast_temp= 53) ,
      interval = "predict", level = 0.95).
```

Reference

[1] Stewart, R.S 07/02/2023, *4.4 Using tidyr to reshape data* [PDF Online], Last Accessed 02/04/2023.
[2] Stewart, R.S 28/03/2023, *10.12 Week 10 Summary* [PDF Online], Last Accessed 02/04/2023.
[3] Stewart, R.S 07/03/2023, *7.17 Week 7 Summary* [PDF Online], Last Accessed 02/04/2023.
[4] Stewart, R.S 21/03/2023, *9.14 Week 9 Summary* [PDF Online], Last Accessed 02/04/2023.
[5] Pyper, K.P 07/03/2023, *8.5 Model Selection Criterion* [PDF Online], Last Accessed 02/04/2023.
[6] Pyper, K.P 15/02/2023, 8.9 Cross Validation [PPT Online], Last Accessed 02/04/2023.
[7] Stewart, R.S 19/02/2023, *5.5 Histograms* [PDF Online], Last Accessed 02/04/2023.

Appendix

A1:
```
weather_forecast_tidy %>%
  mutate(new_high_low = ifelse(high_or_low== "low", 0, 1 )) -> weather_forecast_tidy
```

A2:
```
t_lm<-lm(observed_temp ~ new_high_low+forecast_hours_before+observed_precip
+forecast_temp +
         forecast_hours_before*new_high_low+observed_precip*new_high_low
+forecast_temp*new_high_low,
       data = weather_forecast_tidy)
drop1(ot_lm, test = "F", scope = ~.)
```

A3:
```
ot_lm1<-lm(observed_temp ~ forecast_hours_before+observed_precip +forecast_temp,
        data = weather_forecast_tidy)
drop1(ot_lm1, test = "F", scope = ~.)
```

A4:
```
step(ot_lm1,test= "F" , scope= ~forecast_hours_before+observed_precip+forecast_temp,
    direction = "both")
```