StatComp Project 2: Scottish weather

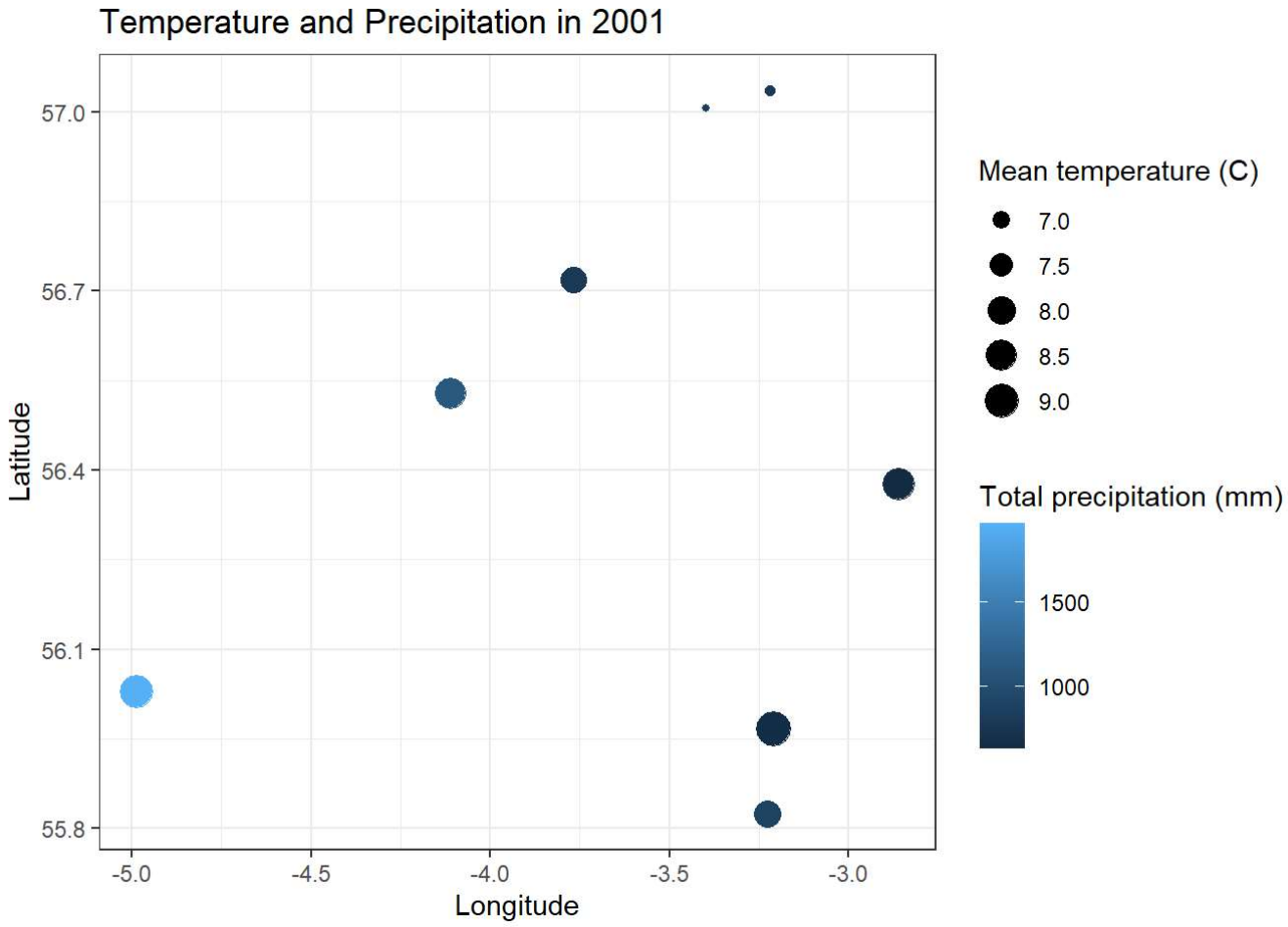
Zongsheng Liu (s2097920)

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

Seasonal variability

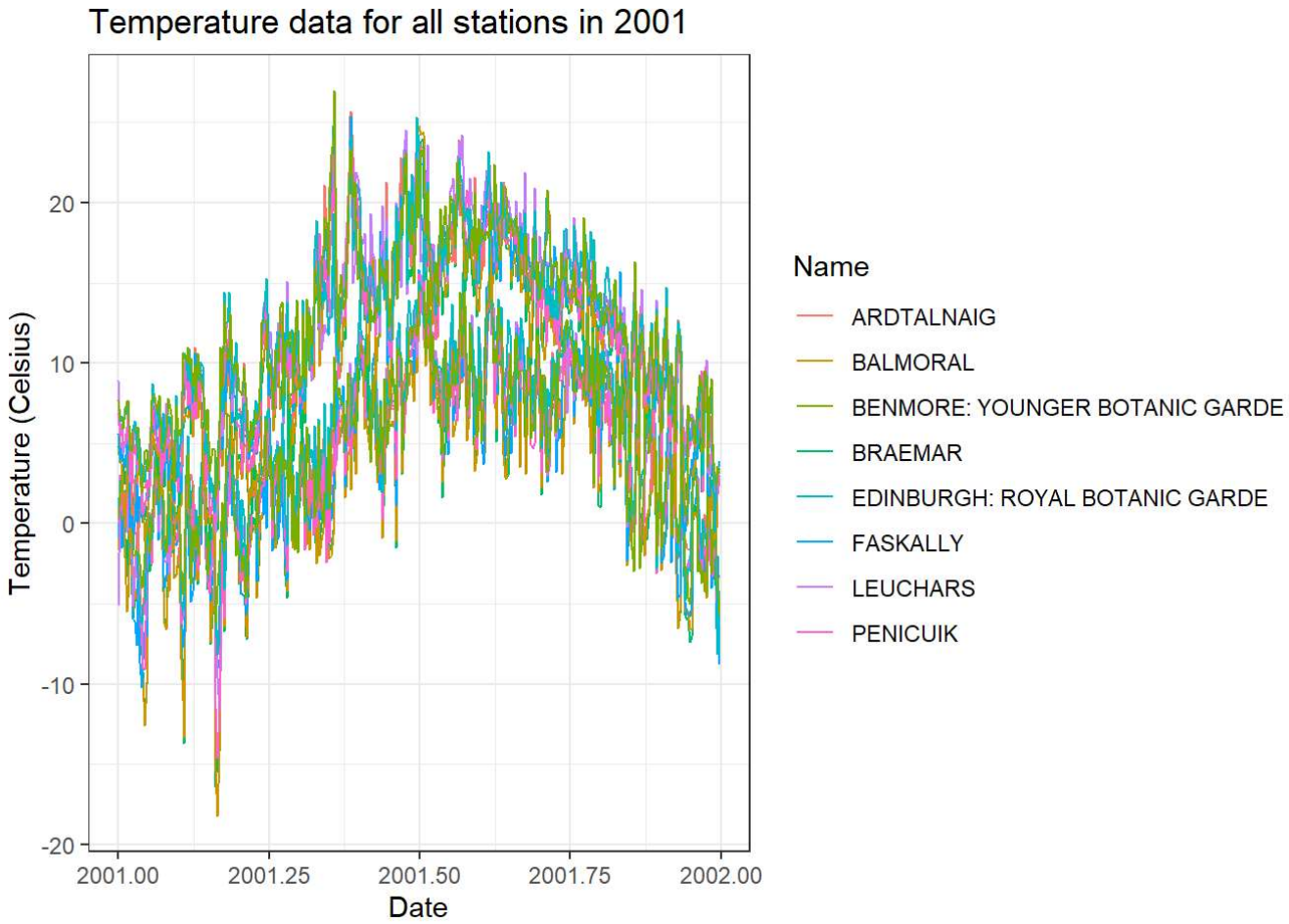
Data Visualization

To have a brief idea of the seasonal variability, firstly, start loading the two data sets $ghcnd\_stations$ and $ghcnd\_values$ and try to plot it. Since the data set is relatively vast and massive to present, eyesight would be narrowed down to only 2001 for the following plot. In order to visualize the data set, use the internal function $filter$, $group\_by$ and $mutate$ to construct well-organized data to plot the following results.

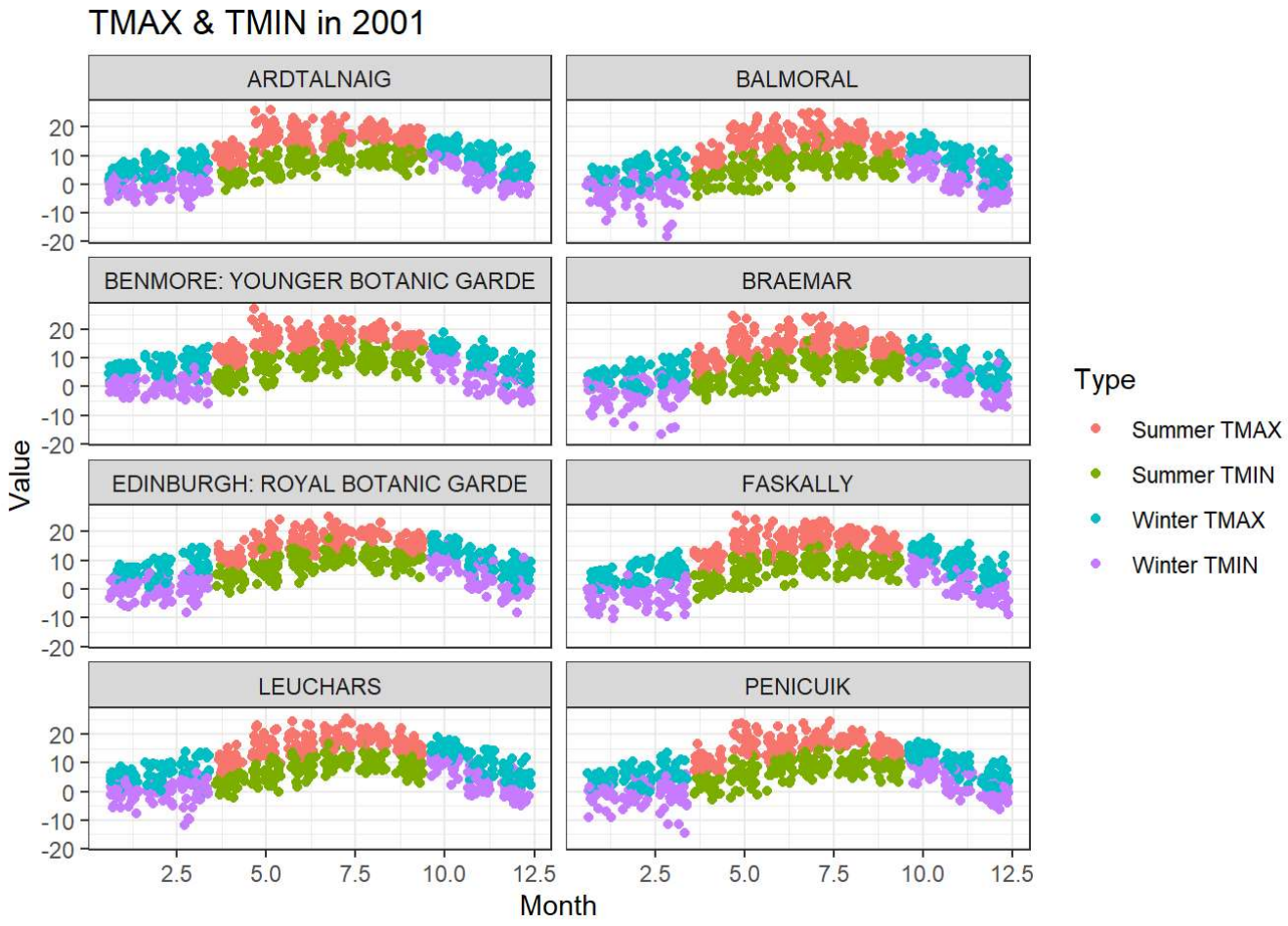


The first plot combined all the information; It took longitude as the x-axis and latitude as the y-axis; each dot would represent an individual station based on its geographic information. The size of each dot would be proportional to the mean temperature, and the color would indicate the total precipitation in 2001; a higher total value would lead to a darker color. As a result suggest, it followed the intuition from the geographic perspective. The mean temperature would decrease as latitude increases, and the total precipitation would increase as its longitude is close to the east coast.

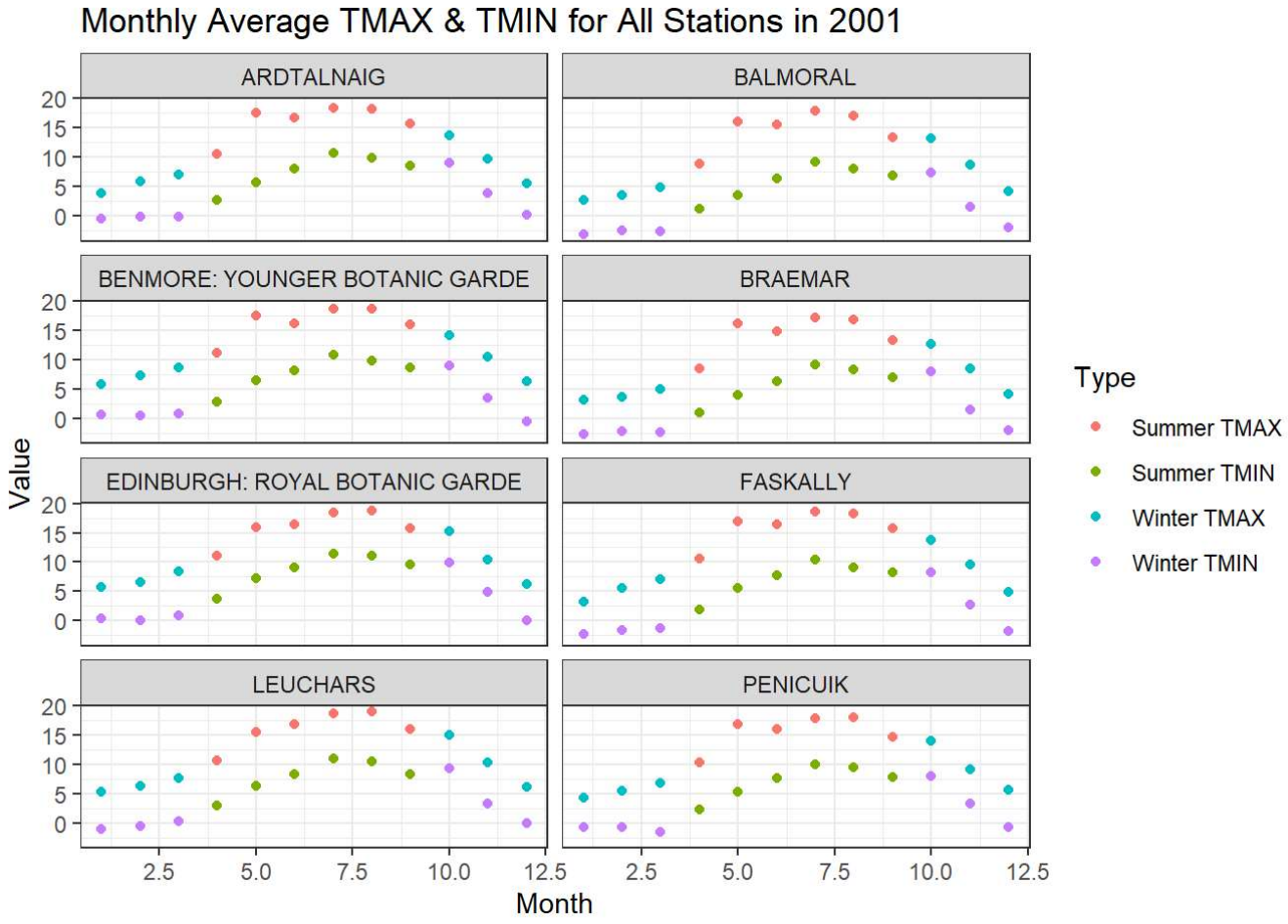
To exam the specific seasonal variability on temperature, firstly plot for TMAX and TMIN for all stations during 2001. It trends to both TMAX and TMIN asymptotically followed a normal distribution with a peak at the middle of the year. Since the plot is still massive due to the volume of the data set.



In order to verify the seasonal variability, the above plot could be further grouped by ID and Seasonal information.

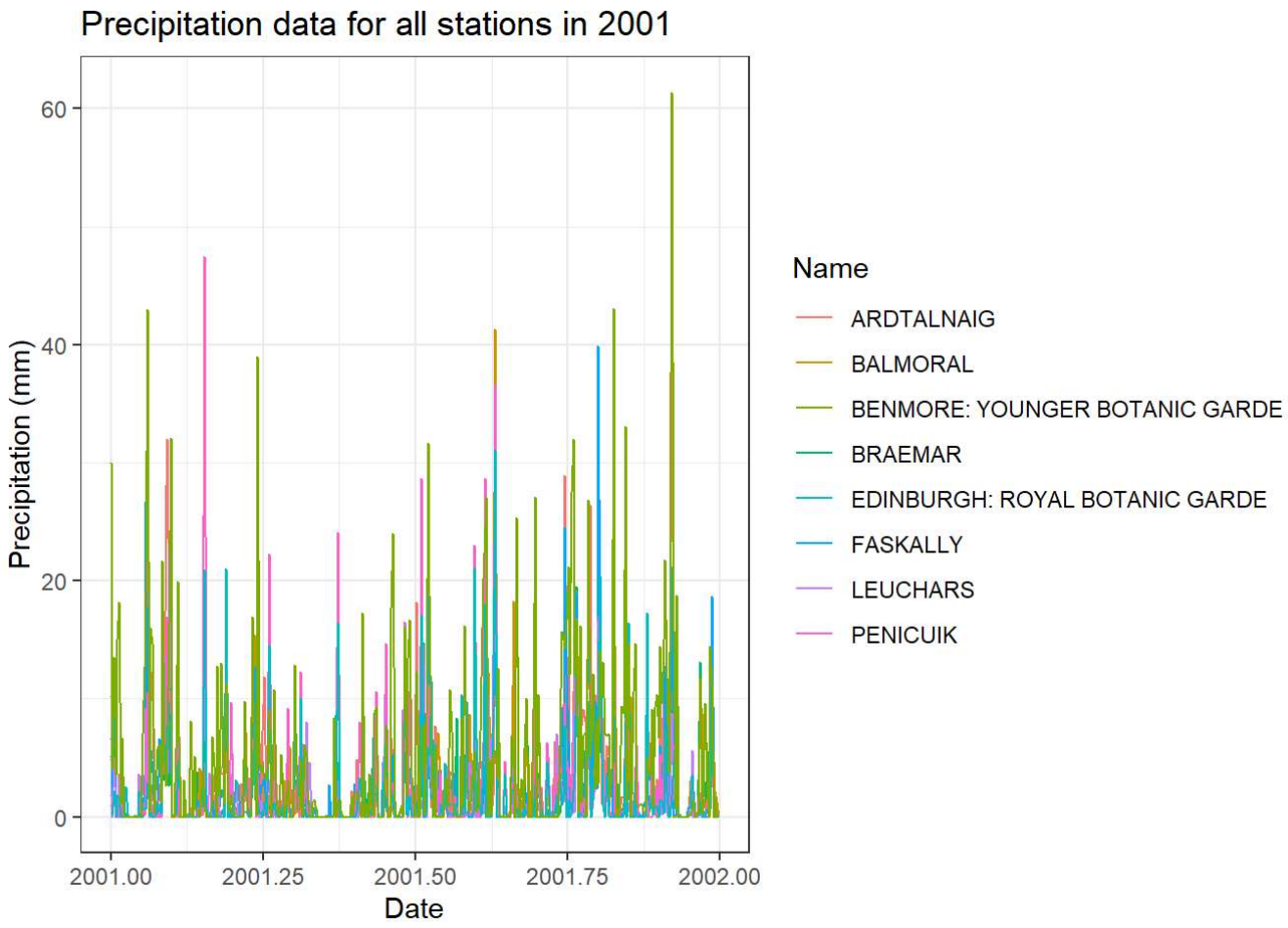


Then, introduce a new plot showing each station's monthly seasonal averages (from Jan 2001 to Dec 2001) of TMIN and TMAX separately.

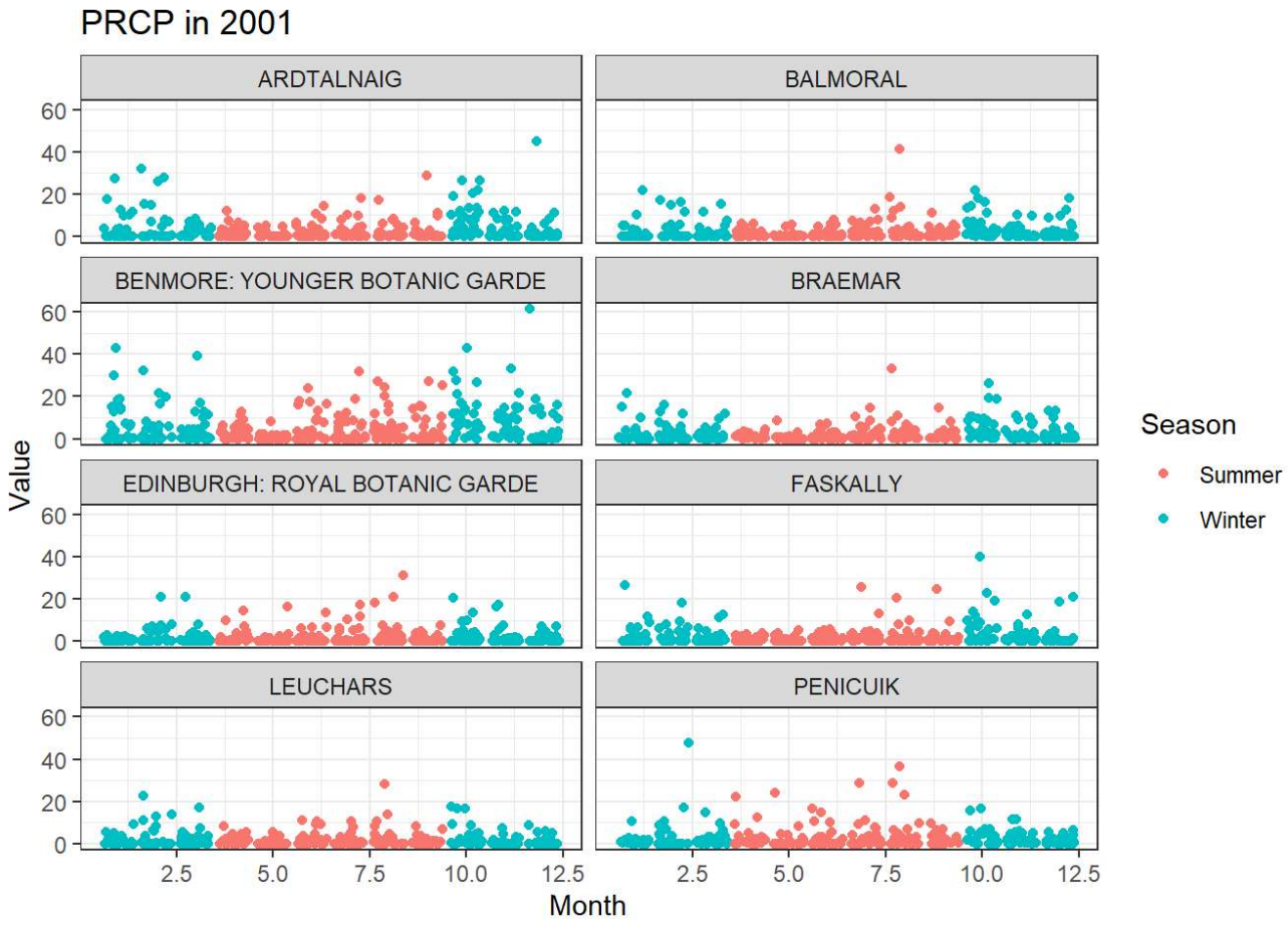


It would clearly follow the same pattern expected above; in this case, the monthly seasonal averages clearly show the seasonal pattern. The shapes are similar bell shapes for all stations, but may vary a bit in the average and amplitude.

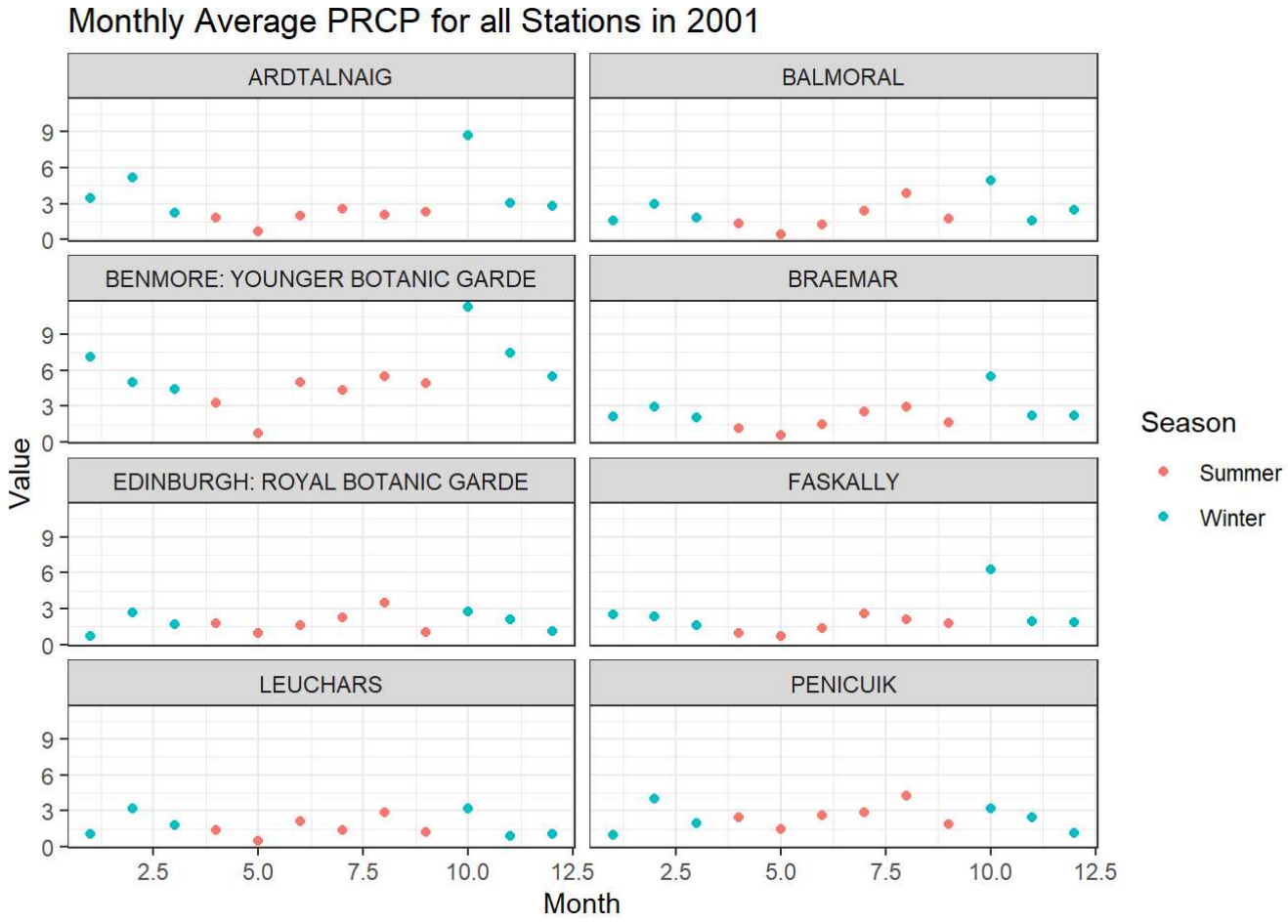
Similarly, all stations' daily precipitation during 2001 could be plotted as follow.



Similarly, plotting $PRCP$, which were separated by ID and $season$ would be helpful to examine the seasonal variability.



Since the data set contains some 0s, the plot would be hard to read for examining the seasonal pattern. It could be separated into some subplots corresponding to each station with monthly average precipitation.



Believe that the monthly average precipitation would not be an excellent representation of the data due to the number of 0s in the data set. In this case, it does not have a typical pattern for each station's seasonal variability. However, the minimum monthly average would occur around May, with different amplitudes for each station.

Hypothesis Testing

Since the first hypothesis test would focus on seasonal variability, it would be helpful to group the data into two seasonal categories in advance. Let winter be {Jan, Feb, Mar, Oct, Nov, Dec}, and let summer be {Apr, May, Jun, Jul, Aug, Sep}. Using $mutate$ and $ifelse$ allowed adding a new column named $Season$ and identifying the seasonal group for every row. By applying the same techniques, a new column named $Summer$ was added to the data that is $TRUE$ for data in the defined summer months. It would make the following effect for the $ghcnd\_values$ as shown below.

```
## # A tibble: 6 × 9
##   ID           Year Month   Day DecYear Element Value Season Summer
##   <chr>       <int> <int> <int>   <dbl> <chr>   <dbl> <chr>  <lgl>
## 1 UKE00105874  1960     1     1   1960  TMAX      3.9 Winter FALSE
## 2 UKE00105874  1960     1     1   1960  TMIN      3.3 Winter FALSE
## 3 UKE00105874  1960     1     2   1960. TMAX      7.2 Winter FALSE
## 4 UKE00105874  1960     1     2   1960. TMIN     -8.3 Winter FALSE
## 5 UKE00105874  1960     1     3   1960. TMAX      6.7 Winter FALSE
## 6 UKE00105874  1960     1     3   1960. TMIN     -6.7 Winter FALSE
```

Then, introduce a Monte Carlo permutation test to test the following hypothesis:

$$H0 : \text{The rainfall distribution is the same in winter as in summer.}$$

$$H1 : \text{The winter and summer distributions have different expected values.}$$

This test was designed to evaluate the significance of differences in rainfall distributions between two seasonal groups. Firstly, introduce the test statistic T as follow:

$$T = |\text{winter average} - \text{summer average}|$$

In this type of test, the observed data are randomly shuffled between the two groups, and the test statistic was calculated for each permuted sample. The above process would be repeated N times by considering the relatively large volume of the existing data set and keeping a more accurate formulation, N would be set to $N = 10000$ in this case. Recall the standard definition of $p\text{-}value$, and it denoted the probability of obtaining results at least as extreme as the observed results. In the above Monte Carlo permutation test, the $p\text{-}value$ is defined by the proportion of times the test statistic was at least as extreme as the observed test statistic. Specifically, the $p\text{-}value$ is computed as the number of times the absolute difference between the winter and summer average rainfall was more significant than or equal to the observed absolute difference, divided by the total number of permutations.

To test the above hypothesis, set $\alpha = 0.05$. Then, the 95% Confidence Intervals would be constructed. In general, it is a range of values that it is 95% confident contains the true value of the parameter being estimated. In this case, the difference between the winter and summer rainfall distributions is an interval estimate. In theory, the above Monte Carlo permutation test is theoretically constructed by finding the values that bound the central 95% of the simulated permutation test statistics. In terms of percentiles, the upper and lower bounds of the 95% confidence interval correspond to the 2.5_{th} percentile and the 97.5_{th} percentile of the simulated T values, respectively. However, 2 cases are considered. While most observed counts are zero for p-value, as $Project2Hints$ suggested, $95\%CI$ would be constructed as follow:

$$CI_p = (0, 1 - 0.025^{\frac{1}{N}})$$

. However, for such an interval, width needs to be considered carefully. It need to satisfy

$$1 - 0.025^{\frac{1}{N}} \leq \epsilon$$

and

$$N \geq \frac{log(0.025)}{log(1-\epsilon)}.$$

For $p - value > 0$, it is a trivial case such that $95\% CI$ is computed by

$$CI_p = \tilde{p} \pm z_{0.975}\sqrt{\frac{\tilde{p}(1-\tilde{p})}{N}}$$

where $\sqrt{\frac{\tilde{p}(1-\tilde{p})}{N}}$ was just clarified as the standard deviation of $p\text{-}value$. In this case, running the Monte Carlo Permutation Test with N=10000 would return the following results:

```
##            ID p_value standard_deviations CI_lower_bound CI_upper_bound
## 1 UKE00105874  0.0000         0.000000000     0.00000000   0.0003688199
## 2 UKE00105875  0.0000         0.000000000     0.00000000   0.0003688199
## 3 UKE00105884  0.0000         0.000000000     0.00000000   0.0003688199
## 4 UKE00105885  0.0000         0.000000000     0.00000000   0.0003688199
## 5 UKE00105886  0.0305         0.001719586     0.02712967   0.0338703261
## 6 UKE00105887  0.0000         0.000000000     0.00000000   0.0003688199
## 7 UKE00105888  0.6603         0.004736073     0.65101747   0.6695825332
## 8 UKE00105930  0.0000         0.000000000     0.00000000   0.0003688199
```

Based on the test result above, most of the weather stations have a small p-value. For stations $\mathrm{UKE00105874\ BRAEMAR}$, $\mathrm{UKE00105875\ BALMORAL}$, $\mathrm{UKE00105884\ ARDTALNAIG}$, $\mathrm{UKE00105885\ FASKALLY}$, $\mathrm{UKE00105887\ PENICUIK}$ and $\mathrm{UKE00105930\ BENMORE}$, above 6 stations had p-value=0, it suggested strong evidence against the null hypothesis that the rainfall distribution is the same in winter as in summer. For station $\mathrm{UKE00105886\ LEUCHARS}$, it has a p_value of 0.031, which is still below the $\alpha = 0.05$. It suggested some evidence against the null hypothesis, but it is not as strong as other stations with a p-value of 0; therefore, the null hypothesis is still rejected. $\mathrm{UKE00105888\ EDINBURGH}$ was the only station who had a p-value greater than α. Therefore, the null hypothesis for this station failed to reject. In other words, there is not enough evidence to conclude that there is a significant difference in the rainfall distribution between summer and winter at this station.

The confidence intervals from the above results showed the lower and upper bounds for the true difference in rainfall between winter and summer for each weather station. It is 95% confident that the true difference in rainfall distribution between winter and summer lies between the lower and upper bounds of the interval for each station. For stations mentioned above with $p\text{-}value = 0$, the lower bound of CI is 0, which indicates that the difference in rainfall distribution between winter and summer is statistically significant. The CI upper bound is 0.0036 which is relatively small, indicating a narrow range of possible values for the true population parameter. To sum up, the test results suggested strong evidence of a difference in the expected rainfall between winter and summer for most of the weather stations except for $\mathrm{UKE00105888\ EDINBURGH}$.

Spatial weather prediction

Model Estimation

In this section, models for the monthly averaged precipitation values in Scotland will be estimated. Knowing that the precipitation data are very skewed and sensitive, with variance increasing with the mean value. Thus, the following model would estimate the square root of the monthly averaged precipitation values. In order to construct the above model, a new data frame named $ghcnd\_lm$ would be constructed as follow: it contained average monthly precipitation, the monthly average of $DecYear$, Longitude, Latitude and Elevation as columns, and grouped by ID, $Year$ and $Month$. The first several rows could be shown as follow.

```
## # A tibble: 6 × 8
## # Groups:   ID, Year [1]
##   ID           Year Month Value_sqrt_avg DecYear_avg Longitude Latitude Elevat…¹
##   <chr>       <int> <int>          <dbl>       <dbl>     <dbl>    <dbl>    <dbl>
## 1 UKE00105874  1961     1          1.52        1961.     -3.40     57.0      339
## 2 UKE00105874  1961     2          1.37        1961.     -3.40     57.0      339
## 3 UKE00105874  1961     3          0.900       1961.     -3.40     57.0      339
## 4 UKE00105874  1961     4          1.31        1961.     -3.40     57.0      339
## 5 UKE00105874  1961     5          1.09        1961.     -3.40     57.0      339
## 6 UKE00105874  1961     6          0.918       1961.     -3.40     57.0      339
## # … with abbreviated variable name ¹Elevation
```

Let M_0 be a basic model for the square root of monthly average precipitation, and it will be defined as:

$$M_0 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear$$

In order to reflect the seasonal variability, introduce the covariates $cos(2\pi kt)$ and $sin(2\pi kt)$ for $k = 1, 2, \ldots$. Thus, the model M_k could be defined as

$$M_k = M_0 + \sum_{k=1}^{K}[\gamma_{c,k}cos(2\pi kt) + \gamma_{s,K}sin(2\pi kt)]$$

.

For the above expression, $\gamma_{s,k}$ and $\gamma_{c,k}$ was defined as the model coefficients, and t was defined to be the average monthly $\mathrm{DecYear}$ discussed above.

In order to have a meaningful interpretation on the following result table for every individual model, $t\text{-}value$, $Pr(>|t|)$ and $Signif.\ codes$ should be introduced. $t\text{-}value$ is associated with testing the parameter's significance in the first column. A relatively large $t\text{-}value$ would suggest that the estimated coefficient is more likely to be different from zero, and the predictor variable is likely to be important in explaining the variation in the response variable. $Pr(>|t|)$ denoted the $p\text{-}value$ associated with the t-statistic discussed previously. If the p-value is smaller than α, the null hypothesis (the true value of the coefficient is zero) would be rejected. Similarly, if the p-value is larger than α, it fails to reject the null hypothesis, and there is insufficient evidence to conclude that the true coefficient value is different from zero. Also, the asterisks at the most right column denoted the level of significance; more asterisks would suggest a smaller $p\text{-}value$ and a higher level of significance. $Signif.codes$ would just provide a visually accessible way of assessing whether the statistic met various α criteria.

Coefficients of M_0, M_1, M_2, M_3 and M_4 could be estimated by $lm()$ as follow:

M_0 :

$$M_0 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear$$

```
##
## Call:
## lm(formula = Value_sqrt_avg ~ Longitude + Latitude + Elevation +
##     DecYear_avg, data = ghcnd_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.94594 -0.32642 -0.02307  0.30481  1.95082
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  2.477e+00  1.394e+00   1.777   0.0757 .
## Longitude   -5.464e-01  1.073e-02 -50.930  < 2e-16 ***
## Latitude    -1.360e-01  2.018e-02  -6.737 1.79e-11 ***
## Elevation    4.022e-04  7.689e-05   5.231 1.75e-07 ***
## DecYear_avg  2.411e-03  3.968e-04   6.076 1.31e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4902 on 5442 degrees of freedom
## Multiple R-squared:  0.3424, Adjusted R-squared:  0.3419
## F-statistic: 708.3 on 4 and 5442 DF,  p-value: < 2.2e-16
```

It is easy to check that for the above 5 parameters, only the coefficient of $Intercept$ seemed not statistically significant (0 asterisks) compared to the rest of the 4 estimated coefficients having 3 asterisks which represented high levels of significance.

M_1 :

$$M_1 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear + cos(2\pi DecYear) + sin(2\pi DecYear).$$

```
##
## Call:
## lm(formula = Value_sqrt_avg ~ Longitude + Latitude + Elevation +
##     DecYear_avg + cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg),
##     data = ghcnd_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.98638 -0.30492 -0.01175  0.29335  1.89615
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.490e+00  1.319e+00   1.888   0.0591 .
## Longitude                  -5.467e-01  1.015e-02 -53.862  < 2e-16 ***
## Latitude                   -1.350e-01  1.909e-02  -7.069 1.76e-12 ***
## Elevation                   4.040e-04  7.274e-05   5.554 2.92e-08 ***
## DecYear_avg                 2.376e-03  3.754e-04   6.329 2.67e-10 ***
## cos(2 * pi * DecYear_avg)   1.828e-01  8.902e-03  20.536  < 2e-16 ***
## sin(2 * pi * DecYear_avg) -1.320e-01  8.870e-03 -14.885  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4637 on 5440 degrees of freedom
## Multiple R-squared:  0.4117, Adjusted R-squared:  0.411
## F-statistic: 634.5 on 6 and 5440 DF,  p-value: < 2.2e-16
```

Compared to M_0, M_1 was just added two more cos and sin terms. The coefficients for the first 5 common variables are similar to those in model M_0. Furthermore, the additional sin and cos terms are both significant(3 asterisks) and may indicate a clear annual pattern in the data.

M_2 :

$$M_2 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear + cos(2\pi DecYear) + sin(2\pi DecYear) + cos(4\pi DecYear) + sin(4\pi DecYear).$$

```
##
## Call:
## lm(formula = Value_sqrt_avg ~ Longitude + Latitude + Elevation +
##     DecYear_avg + cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
##     cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg), data = ghcnd_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.98846 -0.30883 -0.01187  0.29641  1.89402
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.477e+00  1.318e+00   1.879 0.060239 .
## Longitude                  -5.467e-01  1.014e-02 -53.911  < 2e-16 ***
## Latitude                   -1.349e-01  1.907e-02  -7.075 1.69e-12 ***
## Elevation                   4.042e-04  7.267e-05   5.562 2.80e-08 ***
## DecYear_avg                 2.382e-03  3.750e-04   6.351 2.31e-10 ***
## cos(2 * pi * DecYear_avg)   1.829e-01  8.894e-03  20.567  < 2e-16 ***
## sin(2 * pi * DecYear_avg) -1.321e-01  8.862e-03 -14.903  < 2e-16 ***
## cos(4 * pi * DecYear_avg)   3.088e-02  8.889e-03   3.474 0.000516 ***
## sin(4 * pi * DecYear_avg)   7.686e-04  8.866e-03   0.087 0.930922
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4633 on 5438 degrees of freedom
## Multiple R-squared:  0.413,  Adjusted R-squared:  0.4121
## F-statistic: 478.3 on 8 and 5438 DF,  p-value: < 2.2e-16
```

Similarly, M_2 had 2 more additional sin and cos terms compared to M_1. In this case, the first 7 common terms would have the same situation in terms of asterisks compared to M_1. However, only the additional term $cos(4\pi DecYear)$ is significant(3 asterisks) compared to another additional term $sin(4\pi DecYear)$, which had 0 asterisks.

M_3 :

$$\begin{aligned} M_3 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear \\ + cos(2\pi DecYear) + sin(2\pi DecYear) \\ + cos(4\pi DecYear) + sin(4\pi DecYear) \\ + cos(6\pi DecYear) + sin(6\pi DecYear). \end{aligned}$$

```
##
## Call:
## lm(formula = Value_sqrt_avg ~ Longitude + Latitude + Elevation +
##     DecYear_avg + cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
##     cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) + cos(6 *
##     pi * DecYear_avg) + sin(6 * pi * DecYear_avg), data = ghcnd_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.01405 -0.30872 -0.01268  0.29559  1.86840
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.470e+00  1.317e+00   1.876 0.060694 .
## Longitude                  -5.467e-01  1.013e-02 -53.958  < 2e-16 ***
## Latitude                   -1.349e-01  1.906e-02  -7.080 1.62e-12 ***
## Elevation                   4.044e-04  7.261e-05   5.570 2.67e-08 ***
## DecYear_avg                 2.385e-03  3.747e-04   6.364 2.12e-10 ***
## cos(2 * pi * DecYear_avg)   1.828e-01  8.887e-03  20.568  < 2e-16 ***
## sin(2 * pi * DecYear_avg) -1.321e-01  8.855e-03 -14.915  < 2e-16 ***
## cos(4 * pi * DecYear_avg)   3.072e-02  8.882e-03   3.459 0.000546 ***
## sin(4 * pi * DecYear_avg)   5.964e-04  8.859e-03   0.067 0.946329
## cos(6 * pi * DecYear_avg) -7.072e-03  8.868e-03  -0.797 0.425214
## sin(6 * pi * DecYear_avg)   2.854e-02  8.872e-03   3.217 0.001305 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4629 on 5436 degrees of freedom
## Multiple R-squared:  0.4142, Adjusted R-squared:  0.4131
## F-statistic: 384.3 on 10 and 5436 DF,  p-value: < 2.2e-16
```

Regarding statistical significance level(asterisks), the first 9 common terms would have exactly the same results compared to M_2. In addition, only the additional term $sin(6\pi DecYear)$ is relatively significant(2 asterisks) compared to another additional term $cos(6\pi DecYear)$, which had 0 asterisks.

M_4 :

$$\begin{aligned} M_4 : Value\_sqrt\_avg \sim Intercept + Longitude + Latitude + Elevation + DecYear \\ + cos(2\pi DecYear) + sin(2\pi DecYear) \\ + cos(4\pi DecYear) + sin(4\pi DecYear) \\ + cos(6\pi DecYear) + sin(6\pi DecYear) \\ + cos(8\pi DecYear) + sin(8\pi DecYear). \end{aligned}$$

```
##
## Call:
## lm(formula = Value_sqrt_avg ~ Longitude + Latitude + Elevation +
##     DecYear_avg + cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
##     cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) + cos(6 *
##     pi * DecYear_avg) + sin(6 * pi * DecYear_avg) + cos(8 * pi *
##     DecYear_avg) + sin(8 * pi * DecYear_avg), data = ghcnd_lm)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.00162 -0.30627 -0.01235  0.29501  1.88086
##
## Coefficients:
##                              Estimate Std. Error t value Pr(>|t|)
## (Intercept)                 2.471e+00  1.317e+00   1.877  0.06056 .
## Longitude                  -5.467e-01  1.013e-02 -53.968  < 2e-16 ***
## Latitude                   -1.350e-01  1.906e-02  -7.083 1.59e-12 ***
## Elevation                   4.042e-04  7.259e-05   5.568 2.70e-08 ***
## DecYear_avg                 2.385e-03  3.747e-04   6.366 2.09e-10 ***
## cos(2 * pi * DecYear_avg)   1.829e-01  8.886e-03  20.578  < 2e-16 ***
## sin(2 * pi * DecYear_avg) -1.320e-01  8.853e-03 -14.916  < 2e-16 ***
## cos(4 * pi * DecYear_avg)   3.070e-02  8.880e-03   3.457  0.00055 ***
## sin(4 * pi * DecYear_avg)   6.781e-04  8.857e-03   0.077  0.93897
## cos(6 * pi * DecYear_avg) -7.045e-03  8.867e-03  -0.795  0.42691
## sin(6 * pi * DecYear_avg)   2.849e-02  8.870e-03   3.212  0.00133 **
## cos(8 * pi * DecYear_avg)   1.356e-02  8.900e-03   1.524  0.12763
## sin(8 * pi * DecYear_avg)   1.223e-02  8.838e-03   1.384  0.16656
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4628 on 5434 degrees of freedom
## Multiple R-squared:  0.4146, Adjusted R-squared:  0.4134
## F-statistic: 320.8 on 12 and 5434 DF,  p-value: < 2.2e-16
```

The last model M_4 would obtain the most significant number of cos and sin terms. The first 11 common terms compared to M_3 obviously had the preciously same result in terms of asterisks. However, the last two additional terms would both be identified as insignificant since both of them had 0 asterisks.

In order to compare the above model, an important concept $Adjusted\ R\text{-}squared$ should be interpreted. $Adjusted\ R\text{-}squared$ could be seen as a penalized version of $R\text{-}squared\ value$.It would provide a measure of how well the model will perform on new data and also adjusted on the number of predictor variables in the model. General speaking, a higher $adjusted\ R\text{-}squared\ value$ indicates a better fit of the model. To sum up from the above investigation, $R\text{-}squared\ value$ was keeping increasing while more cos and sin terms were added. In other words, M_4 had the highest $R\text{-}squared\ value$, which means it may have a better fit of the model to the data. Another auxiliary variable would be consider to examine was $F\text{-}statistic$. It would measure the overall significance of the model. In this case, M_4 also had the highest $F\text{-}statistic$, which also suggests as a better fit. However, by the simple investigation, the most of additional terms added in M_2, M_3 and M_4 would be identified as insignificant by their $P\text{-}value$. Although, $adjusted\ R\text{-}squared\ value$ and F-statistic would both increase by adding more terms and , they may lead to a new risk that a more complex model would overfit the data in this case.

Stratified Cross-Validation

In order to compare which model would predict precipitation better at new locations, then stratified cross-validation that groups the data by weather station would be introduced. In theory, it would be followed as a similar idea for $K\text{-}fold\ Cross\ Vaildation$. Instead of splitting the whole data set into K subsets and doing K times cross-validation, for each station, data set $ghcnd\_lm$ would be separated into two parts; the first part, which only contains the data from this specifically tested station would be used as a validation set, and the second part is just the rest of data which had already excluded the data from this tested station, it would be used for training the above 5 models. In order to examine the performance of each model, after each validation at each station, scores could be computed as a key performance indicator. To consider the above 5 models from different perspectives, there were two scores $Dawid\text{-}Sebastiani\ Score\ S_{DS}$ and $Squared\ Error\ Score\ S_{SE}$ that would be computed.

It would be computed as follow:

$$S_{SE}(F, y) = (y - \hat{y}_F)^2$$

$$S_{DS}(F, y) = \frac{(y - \mu_F^2)^2}{\sigma_F^2} + log(\sigma_F^2)$$

Where $\hat{y}_F$ is a point estimate under prediction distribution F; similarly, μ_F and σ_F were point expectation and standard deviation respectively under the prediction distribution F.

For each validation data set corresponding to each station, assuming it would have a cardinality N, there would be precisely the number of N scores $S(F, y_i)$. However, another objective for this task is to examine whether the prediction accuracy across the whole year is the same. In this case, grouping each score $S(F, y_i)$ by the $Month$ category associated with each validated point in advance would be helpful. Furthermore, the mean score would be computed as follow:

$$\hat{S} = \frac{1}{N} \sum_{i=1}^{N} S(F, y_i).$$

In other words, it would take the mean value of all score values for each validation point. Thus, the score result would be obtained as follow.

$DS\ Score\ Result$

```
##                     M0          M1         M2         M3         M4
## UKE00105874 -0.67154282 -0.73860300 -0.7612443 -0.7732837 -0.7807061
## UKE00105875 -0.65442012 -0.70053381 -0.7157929 -0.7233330 -0.7279975
## UKE00105884  0.01379104 -0.08935839 -0.1246451 -0.1436464 -0.1551673
## UKE00105885 -0.44964572 -0.49702276 -0.5146089 -0.5242613 -0.5300673
## UKE00105886 -0.23723171 -0.19667239 -0.1843903 -0.1782761 -0.1747920
## UKE00105887 -0.66510384 -0.69093351 -0.7005041 -0.7053148 -0.7082705
## UKE00105888 -0.66045147 -0.64226546 -0.6373893 -0.6349692 -0.6336080
## UKE00105930  1.81807021  1.71628519  1.6866975  1.6719833  1.6634857
```

$SE\ Score\ Result$

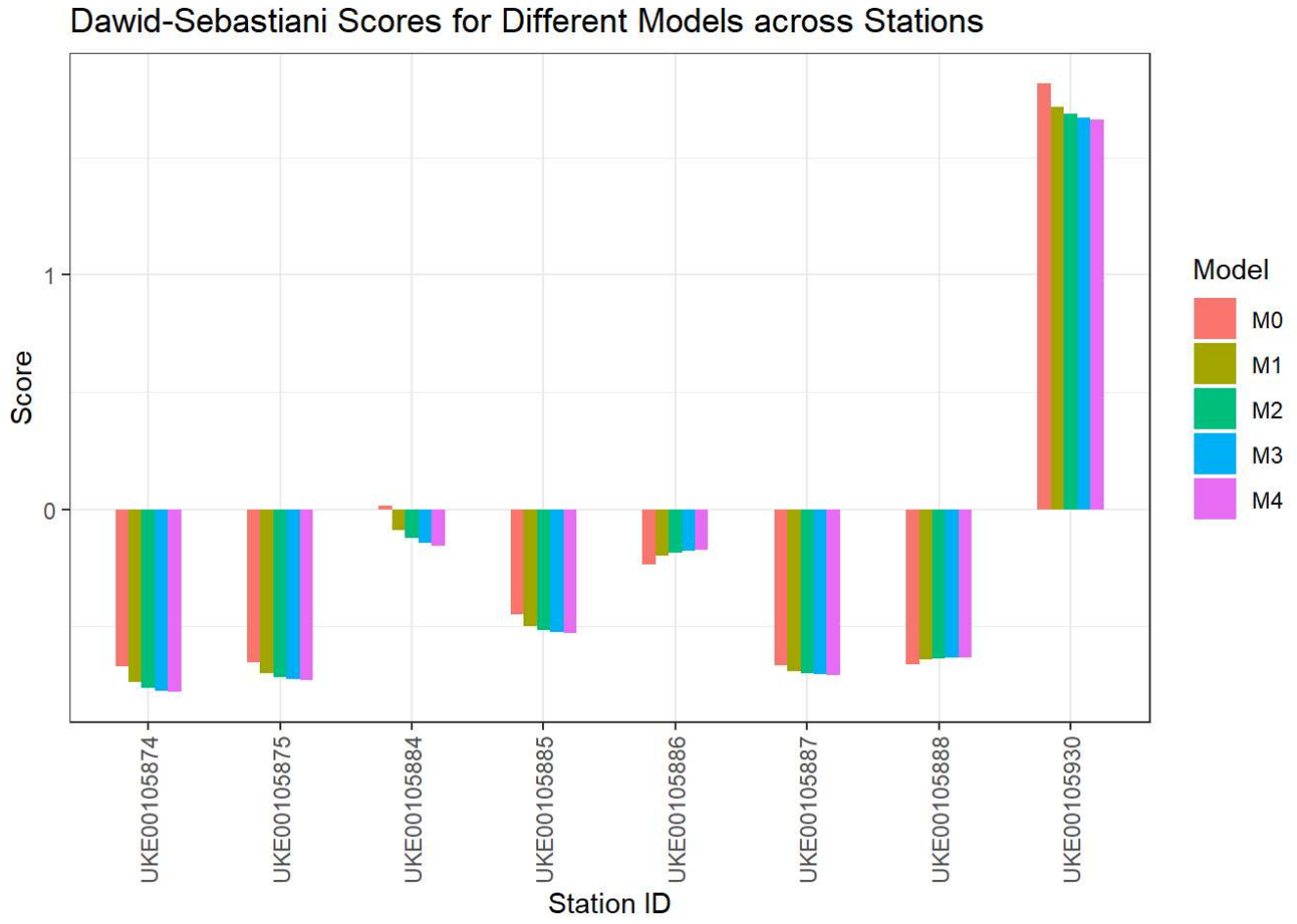
```
##                    M0        M1        M2        M3        M4
## UKE00105874 0.1788729 0.1667628 0.1626964 0.1605197 0.1591676
## UKE00105875 0.1830942 0.1756277 0.1731938 0.1720167 0.1712776
## UKE00105884 0.3394895 0.3126104 0.3034292 0.2985463 0.2955919
## UKE00105885 0.2345134 0.2239609 0.2200554 0.2179175 0.2166318
## UKE00105886 0.2867391 0.2920788 0.2935619 0.2942712 0.2946582
## UKE00105887 0.1800883 0.1774780 0.1764140 0.1758933 0.1755652
## UKE00105888 0.1810967 0.1884164 0.1906042 0.1917030 0.1923422
## UKE00105930 0.6791761 0.6403603 0.6277927 0.6213674 0.6175527
```

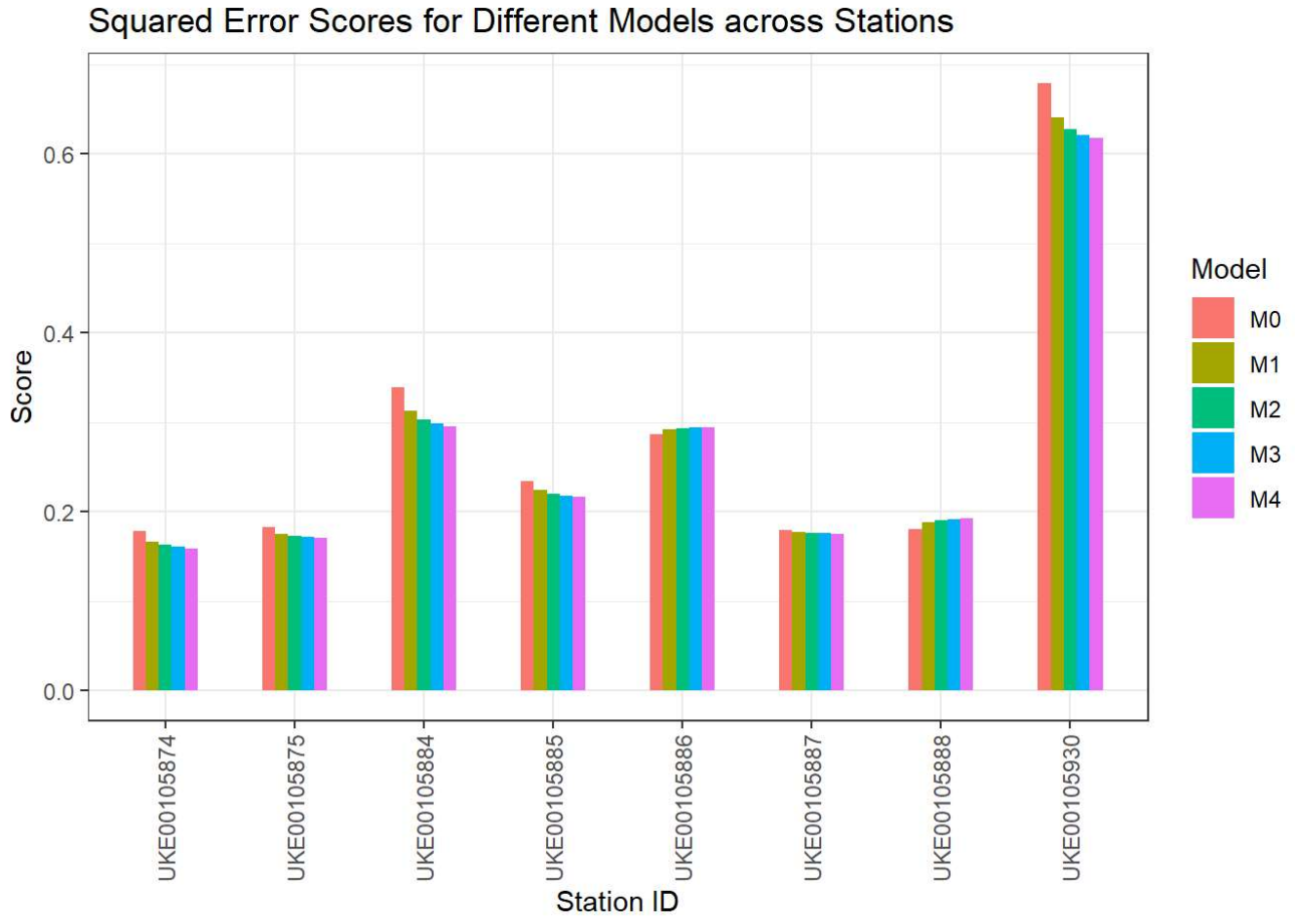
Since the above two scoring rules were made by negative orientation. In other words, a lower score would indicate a better-fitting performance. When the following two conditions are satisfied

$$\sigma_F^2 < 1 \textbf{ and } \mu_F - \sqrt{-\sigma_F^2 log(\sigma_F^2)} < y < \mu_F + \sqrt{-\sigma_F^2 log(\sigma_F^2)},$$

it would lead to a negative $Dawid\text{-}Sebastiani\ Score$. However, the same rule would be applied that compares the scores computed on the number line with lower values as better.

The above two tables would be easily visualized by two bar-charts.





Knowing that $Dawid\text{-}Sebastiani\ Score$ considers both the bias and variance of a model, it would provide a better measure of prediction accuracy, particularly for generalizing to new data. In this case, $Dawid\text{-}Sebastiani\ Score$ would be a helpful tool to examine the prediction accuracy at different stations. By observing $Dawid\text{-}Sebastiani\ Score$ from the first table, it would be obvious that most of the stations($UKE00105874\ BRAEMAR$, $UKE00105875\ BALMORAL$, $UKE00105884\ ARDTALNAIG$, $UKE00105885\ FASKALLY$. $UKE00105887\ PENICUIK$ and $UKE00105930\ BENMORE$) would have the relatively lowest $DS\ Score$ from M_4, it indicated M_4 would be suggested as a better model. For $UKE00105886\ LEUCHARS$ and $UKE00105888\ EDINBURGH$ would have obtained the smallest $DS\ Score$ from M_0, which lead to an opposite result that M_0 had better performance in this case.

Similarly, the same analysis algorithm could be done for results from $Squread\ Error\ Score$. The result had an astonishing similarity; the lowest score value of M_4 would be obtained from the same stations discussed above $UKE00105874\ BRAEMAR$, $UKE00105875\ BALMORAL$, $UKE00105884\ ARDTALNAIG$, $UKE00105885\ FASKALLY$. $UKE00105887\ PENICUIK$ and $UKE00105930\ BENMORE$. Furthermore, $UKE00105886\ LEUCHARS$ and $UKE00105888\ EDINBURGH$ would also had the least score values from M_0.

In summary, models are not equally good at predicting the different stations. In general, $ M_4 $ is a better model for most stations. However, it may also suggest that M_0 is better for few stations such as $UKE00105886\ LEUCHARS$ and $UKE00105888\ EDINBURGH$.

Since each individual score $S(F, y_i)$ had already been grouped by $month$, the focus would be easily narrowed down to monthly mean scores for each model and station across the whole year. By applying the same technique, the monthly mean scores could be computed by the mean value of the set grouped by ID, $Month$ and $Model$. Then, results could be obtained by a data frame with size 96×12. The first two columns indicate ID and $Month$, the following 5 columns would illustrate S_{DS} over 5 models, and last 5 columns were filled by S_{SE} over 5 models. Since there are 8 stations and score values would be grouped into 12 months; thus, this data frame would contain precisely $8 \times 12 = 96$ rows. The first several rows of this data frame could be visualized as follow (Full details can be found in Code Appendix).

```
##            ID month              M0_DS              M1_DS              M2_DS
## 1 UKE00105874     1  -0.42386296930423 -0.672156255640566 -0.701325543986491
## 2 UKE00105874     2 -0.771280974708621 -0.784021775923047  -0.78046423263305
## 3 UKE00105874     3 -0.987656236745094  -1.06865296369845  -1.06513712328886
## 4 UKE00105874     4 -0.579230315136535 -0.810826719405428 -0.823227106934836
## 5 UKE00105874     5 -0.785456853937001  -0.92439025530766 -0.923431182214247
## 6 UKE00105874     6 -0.843020709660528  -1.01933917652282  -1.02503905603877
##                M3_DS              M4_DS              M0_SE              M1_SE
## 1 -0.712289927144644 -0.723006198893593 0.240631628392496  0.184565230066764
## 2 -0.756762659086561 -0.770004179287415  0.15400191281537  0.159497802872058
## 3  -1.06198865589765  -1.06014340141016 0.100056436306682 0.0957365097182405
## 4 -0.827430878931835 -0.825622638330334 0.201888020169839   0.15349326957422
## 5 -0.931795458165218 -0.923060118186526 0.150471460166394  0.128056852083531
## 6  -1.02580145411332  -1.02534391736023 0.136116922233999  0.106787384728047
##                M2_SE              M3_SE              M4_SE
## 1  0.178100267723097  0.175695271548145 0.173295288766188
## 2  0.160391368566891  0.165751703352831  0.16278581094919
## 3 0.0967160525237215 0.0975452436314947 0.097943564324042
## 4  0.150825104864066  0.149957363553642 0.150352646992871
## 5  0.128415348445563   0.12663983900391  0.12858013353864
## 6  0.105687894242941  0.105634299909767 0.105722533874279
```

Then, above data frame could be further summarized as the monthly mean score for all stations. It could be represented as the following two tables depend on its scoring method.

DS Score Result

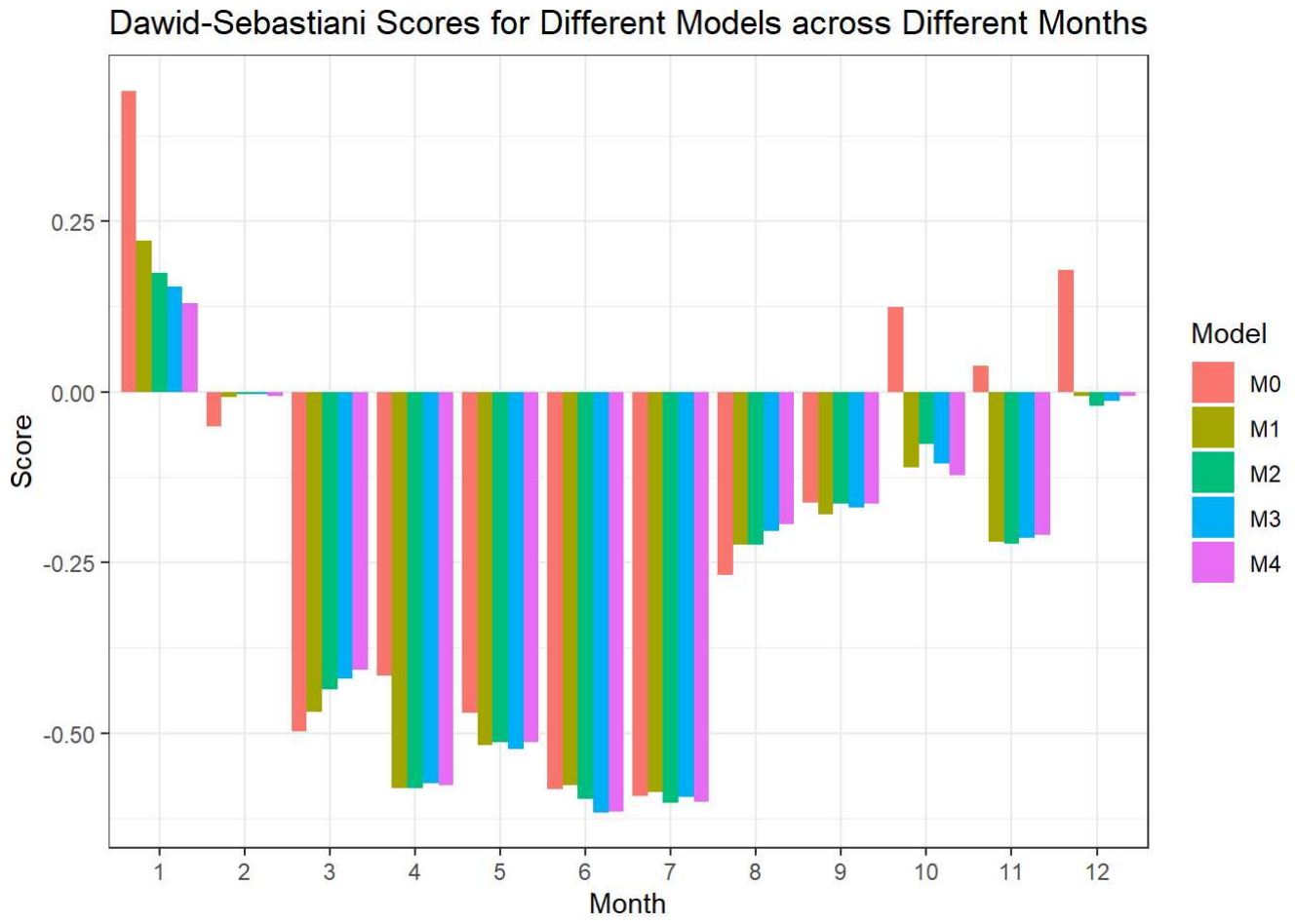
```
##            M0           M1           M2           M3           M4
## 1   0.4413907  0.221915183  0.173982632  0.154636762  0.129554103
## 2  -0.0503851 -0.008052729 -0.003203456 -0.003193911 -0.006877864
## 3  -0.4973583 -0.468919645 -0.436180003 -0.420446317 -0.408113801
## 4  -0.4166440 -0.580489980 -0.580298860 -0.573474248 -0.576219327
## 5  -0.4708118 -0.517874521 -0.513267998 -0.523342039 -0.513014102
## 6  -0.5826806 -0.576720029 -0.597217453 -0.616064525 -0.615005219
## 7  -0.5926212 -0.586151048 -0.602467905 -0.594123964 -0.601649868
## 8  -0.2682899 -0.223437014 -0.223859917 -0.203809238 -0.193991683
## 9  -0.1626464 -0.179965486 -0.163227233 -0.169643694 -0.163904641
## 10  0.1246962 -0.111334845 -0.076103857 -0.105132222 -0.122205711
## 11  0.0376633 -0.219984591 -0.223322307 -0.214438825 -0.209364440
## 12  0.1778855 -0.006496048 -0.020970118 -0.014125388 -0.006022949
```

SE Score Result

```
##           M0        M1        M2        M3        M4
## 1  0.4151712 0.3536541 0.3441055 0.3403913 0.3357776
## 2  0.3103258 0.3118834 0.3125109 0.3130576 0.3119986
## 3  0.2090948 0.2156752 0.2216202 0.2245575 0.2268042
## 4  0.2398814 0.2031036 0.2030351 0.2044030 0.2038990
## 5  0.2220588 0.2134897 0.2144625 0.2122486 0.2145144
## 6  0.2018153 0.2063394 0.2019009 0.1977561 0.1979971
## 7  0.1989819 0.2034511 0.1999361 0.2017490 0.2001122
## 8  0.2726760 0.2772730 0.2769680 0.2809943 0.2830602
## 9  0.2886359 0.2805546 0.2831044 0.2819955 0.2829303
## 10 0.3522729 0.2931995 0.2996408 0.2940202 0.2909285
## 11 0.3283321 0.2691176 0.2683576 0.2696340 0.2704126
## 12 0.3553591 0.3095394 0.3073360 0.3077392 0.3090589
```

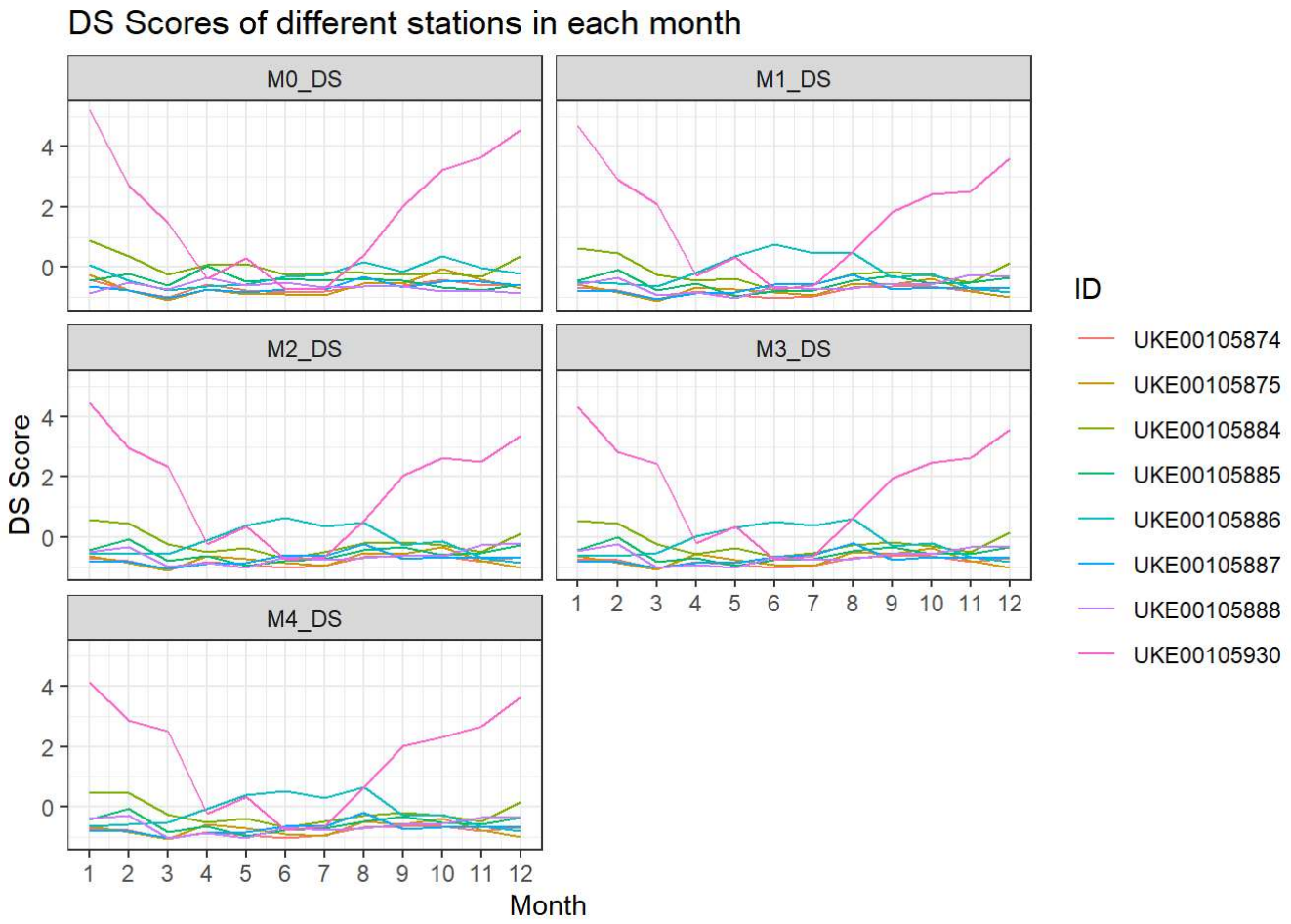
Furthermore, the above two tables could make a more general observation of each month's average score values. Since $Squread\ Error\ Score$ was computed as an average of the squared differences between the predicted and actual values, it would provide a good measure of the overall fit between predictions and the actual values. In this case, it may be helpful to introduce $Squread\ Error\ Score$ as a main indicator scoring method to examine the overall fit performance for each model in each month.

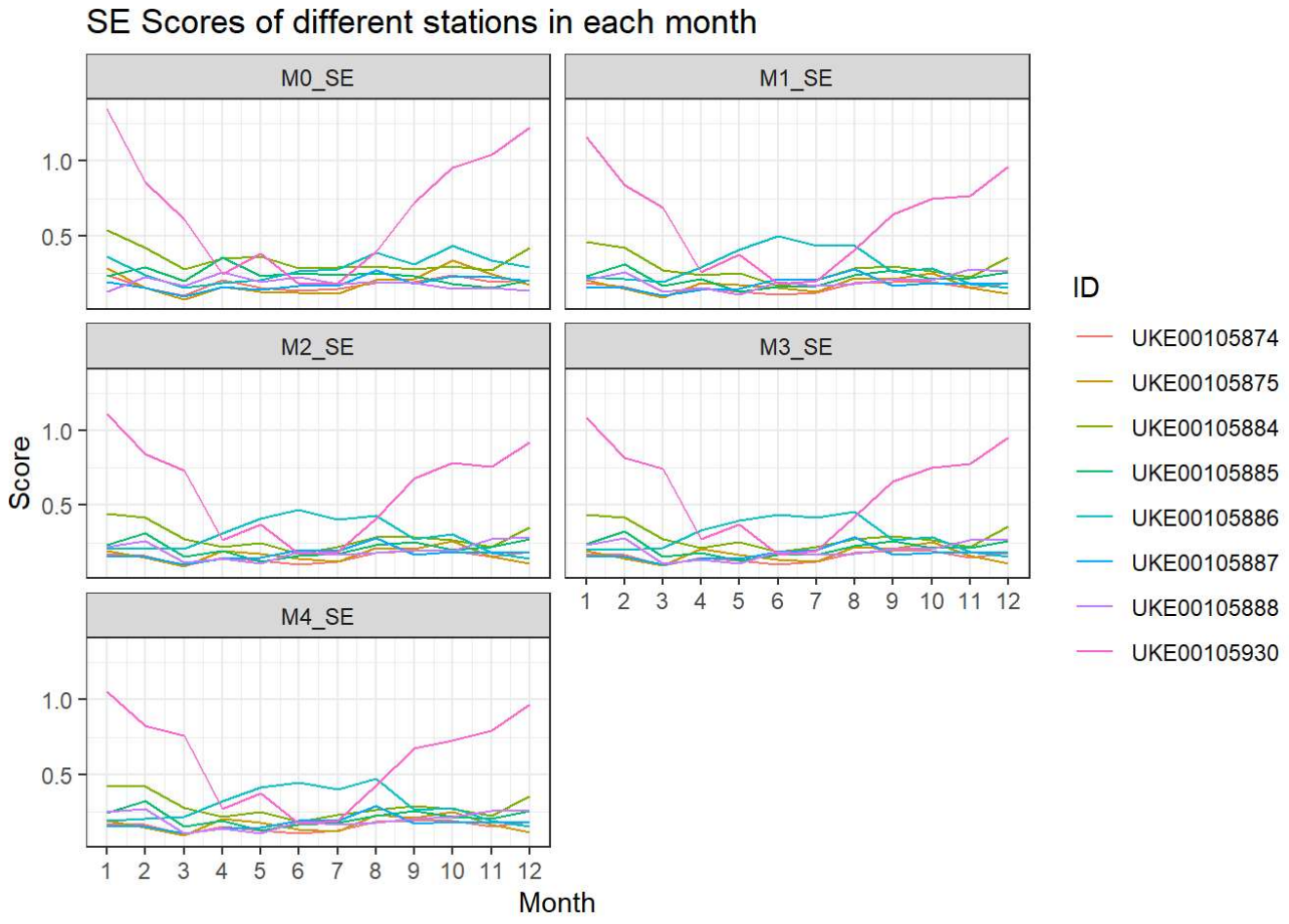
Then, it could be plotted as two bar-chart for comparing scores.



By observing the results of $Squread\ Error\ Score$, it was difficult to identify which model would always have a dominant lowest value across most months. For Jan, Apr, Jun, Nov and Dec, it would seemed to have scores of M_4 which were identified as the least one. For Feb, Mar, Jul and Aug, they suggested M_0 was better because of its lowest score values. Furthermore, for the rest of the months, scores from M_2 or M_3 may sometimes take the lowest value. A similar observation could be done for results of $Dawid\text{-}Sebastiani\ Score$ score. A similar conclusion would be drawn; there would not be any models that had the smallest score values across most of the months. From the above discussion, it is believed that the prediction accuracy or overall fit performance is not the same across the whole year for every model.

Then, score values in each month could be further investigated at every station. It could be visualized as two plots below. In this case, since each subplot would contain 8 values from all 12 months, it would be challenging to examine 96 bars in total if it was a bar chart. For better visualization, the following two plots would be designed as two line plots instead.





In this case, each plot has five subplots grouped by model type. The two plots would generally have the same trend; for each model, it would be observed that general score performance at each station was different, especially at station $UKE00105930\ BENMORE$ and $UKE00105886\ LEUCHARS$. The above two

plots could also be viewed from monthly scores variation; for each model, each station’s score values for all months would not be consistent; it had some amount of variations. The above observation would further validate the previous conclusion that the models were not equally good at predicting the different stations, even though the prediction accuracy was not the same for every month.

Code appendix

Full Details of Score Results

```
##             ID month               M0_DS               M1_DS
## 1  UKE00105874     1   -0.42386296930423  -0.672156255640566
## 2  UKE00105874     2  -0.771280974708621  -0.784021775923047
## 3  UKE00105874     3  -0.987656236745094   -1.06865296369845
## 4  UKE00105874     4  -0.579230315136535  -0.810826719405428
## 5  UKE00105874     5  -0.785456853937001   -0.92439025530766
## 6  UKE00105874     6  -0.843020709660528   -1.01933917652282
## 7  UKE00105874     7  -0.802581401248321  -0.958022591149753
## 8  UKE00105874     8    -0.6371811139519  -0.675064632553239
## 9  UKE00105874     9  -0.620883687203387  -0.629445307919127
## 10 UKE00105874    10  -0.427461214291655  -0.629165398714115
## 11 UKE00105874    11  -0.607735603160342  -0.816099773324778
## 12 UKE00105874    12  -0.572162788096072  -0.680773375062049
## 13 UKE00105875     1  -0.257596168779421  -0.568623816897187
## 14 UKE00105875     2   -0.78079046344973  -0.843213889302671
## 15 UKE00105875     3   -1.08624522480219   -1.12160505209717
## 16 UKE00105875     4  -0.739527496692207  -0.665776978278724
## 17 UKE00105875     5  -0.891246528193678  -0.732742654816807
## 18 UKE00105875     6  -0.908959096462405  -0.828997488405851
## 19 UKE00105875     7  -0.937858052945392  -0.920874465524347
## 20 UKE00105875     8  -0.555885997164922  -0.530229265302398
## 21 UKE00105875     9  -0.533205475012939  -0.565225250919072
## 22 UKE00105875    10 -0.0539968904433351  -0.375702069496599
## 23 UKE00105875    11  -0.408606092085872  -0.802625146010129
## 24 UKE00105875    12   -0.69912392282115   -1.00415394520501
## 25 UKE00105884     1   0.894735059981212   0.641265844058935
## 26 UKE00105884     2   0.363176553814971    0.44997375108153
## 27 UKE00105884     3  -0.261867601716065   -0.25776376291263
## 28 UKE00105884     4  0.0595413672491986  -0.434899271115994
## 29 UKE00105884     5  0.0984780799055369  -0.366138788094822
## 30 UKE00105884     6   -0.23473016464053  -0.783319821995905
## 31 UKE00105884     7  -0.200009993382451  -0.570843575200092
## 32 UKE00105884     8  -0.186962572514871   -0.20736306170718
## 33 UKE00105884     9  -0.265398640930928  -0.146340237779018
## 34 UKE00105884    10  -0.172534778265859  -0.286008104530386
## 35 UKE00105884    11  -0.299506822869904  -0.497101574320118
## 36 UKE00105884    12   0.370571963203463   0.148444758233304
## 37 UKE00105885     1   -0.45330850351424  -0.457578919669015
## 38 UKE00105885     2  -0.213755505682426 -0.0909657053022989
## 39 UKE00105885     3  -0.597480383270309  -0.764284027400959
## 40 UKE00105885     4  0.0363254610387591  -0.556644226970723
## 41 UKE00105885     5  -0.468968192439407  -0.949897859625099
## 42 UKE00105885     6  -0.389439699992378  -0.808482500817602
## 43 UKE00105885     7  -0.444672640585872  -0.758343115463668
## 44 UKE00105885     8  -0.381842627722916  -0.443580020962662
## 45 UKE00105885     9  -0.455315318213933  -0.295277399584023
## 46 UKE00105885    10  -0.678722025979314   -0.55532122437033
## 47 UKE00105885    11  -0.774908002168037   -0.52000730951656
## 48 UKE00105885    12  -0.573661199073093  -0.332415237767757
## 49 UKE00105886     1  0.0597263745731535  -0.483587782075512
## 50 UKE00105886     2  -0.448703308409669  -0.557763186502708
## 51 UKE00105886     3   -0.77318653072962  -0.638026636663294
## 52 UKE00105886     4  -0.640339931378019  -0.189533887185943
## 53 UKE00105886     5  -0.581323411637759   0.353032131367143
## 54 UKE00105886     6  -0.320438610361874   0.765266025906994
```

```
## 55 UKE00105886     7  -0.265045188694703    0.46897148143418
## 56 UKE00105886     8   0.167996625445748   0.481879117493327
## 57 UKE00105886     9  -0.148061938791927  -0.342533121288678
## 58 UKE00105886    10   0.360500805481718  -0.206051381057169
## 59 UKE00105886    11 -0.0302603096535238  -0.696300764111316
## 60 UKE00105886    12   -0.22764505108064  -0.828708970330141
## 61 UKE00105887     1  -0.626283640530385  -0.806009745576308
## 62 UKE00105887     2  -0.775292407146139  -0.817683314687536
## 63 UKE00105887     3   -1.01640162775275   -1.06410962826142
## 64 UKE00105887     4  -0.736612816845033  -0.882310476273223
## 65 UKE00105887     5   -0.83244475496925  -0.849350826653379
## 66 UKE00105887     6  -0.733044017620299  -0.560644620153184
## 67 UKE00105887     7  -0.725670690152829  -0.585162923364669
## 68 UKE00105887     8   -0.31295146944564  -0.245181842968552
## 69 UKE00105887     9  -0.678440939605053  -0.744859345729316
## 70 UKE00105887    10  -0.469592532786323  -0.673522045899714
## 71 UKE00105887    11  -0.477462687309048  -0.691671031590306
## 72 UKE00105887    12  -0.597048469022037  -0.680652287448821
## 73 UKE00105888     1  -0.881379097311782  -0.584085115366832
## 74 UKE00105888     2  -0.494309357230136  -0.335578138861592
## 75 UKE00105888     3  -0.725411963236346  -0.929420882285176
## 76 UKE00105888     4   -0.35416562683954   -0.82211591420537
## 77 UKE00105888     5  -0.621935776973582   -1.02265682099678
## 78 UKE00105888     6  -0.500178365822276   -0.65095219516024
## 79 UKE00105888     7  -0.670230977478998  -0.747597093555821
## 80 UKE00105888     8   -0.62784897927325  -0.700343339536867
## 81 UKE00105888     9  -0.637242950217835  -0.550936722430121
## 82 UKE00105888    10  -0.793646425179072  -0.578954583236226
## 83 UKE00105888    11  -0.767282786870325  -0.260665023583984
## 84 UKE00105888    12  -0.851785349393709  -0.305647445803975
## 85 UKE00105930     1    5.21909493121202     4.7060972554971
## 86 UKE00105930     2    2.71787469792479    2.91483043130037
## 87 UKE00105930     3    1.46938284567107    2.09250579224351
## 88 UKE00105930     4  -0.379142873529031  -0.281812362819001
## 89 UKE00105930     5   0.316403079132927   0.349148908510861
## 90 UKE00105930     6  -0.731634086094251  -0.727290455319137
## 91 UKE00105930     7   -0.69490071182825  -0.617336102745057
## 92 UKE00105930     8   0.388356766694337   0.532386936937662
## 93 UKE00105930     9    2.03737739786609    1.83489350079762
## 94 UKE00105930    10     3.2330229095194    2.41404604620465
## 95 UKE00105930    11    3.66706870789721    2.52459389478907
## 96 UKE00105930    12    4.57393881510213    3.63193812084786
##                  M2_DS               M3_DS               M4_DS
## 1   -0.701325543986491  -0.712289927144644  -0.723006198893593
## 2    -0.78046423263305  -0.756762659086561  -0.770004179287415
## 3    -1.06513712328886   -1.06198865589765   -1.06014340141016
## 4   -0.823227106934836  -0.827430878931835  -0.825622638330334
## 5   -0.923431182214247  -0.931795458165218  -0.923060118186526
## 6    -1.02503905603877   -1.02580145411332   -1.02534391736023
## 7   -0.950445551010939  -0.955451915346863  -0.948925459664829
## 8   -0.674016443524688  -0.678340146249407  -0.678044995087948
## 9   -0.639516785245777  -0.635634388562709  -0.636948035357932
## 10  -0.613969035454372  -0.627951712685808  -0.633743453353156
## 11  -0.817146367372362  -0.818246910608172   -0.81747854287831
## 12  -0.664604052372644  -0.681126616537633  -0.682430694074739
## 13   -0.61375397355914  -0.632393740817164  -0.648640977871723
```

```
## 14  -0.845719237484328  -0.850220067642363  -0.848966806235423
## 15   -1.09787804367319   -1.08313551417139   -1.08183787904614
## 16  -0.616689901976139  -0.563988366792333  -0.593295192060158
## 17  -0.718567829062361  -0.745282728608875  -0.716220065247537
## 18  -0.866207610629958  -0.903326305479153   -0.90202057608518
## 19  -0.954651581134475  -0.942112524186335  -0.956654539244222
## 20  -0.537746244320339  -0.501189234702157  -0.478972875338968
## 21   -0.56513167133753  -0.565825295671215  -0.565810370412148
## 22  -0.331840717396897  -0.371981625915379  -0.390074674956717
## 23  -0.798615304781302  -0.787184430452602   -0.77256325041406
## 24   -1.00893230675855   -1.00479829798731   -1.00481096919447
## 25   0.571215420103886   0.544433248233058   0.500766765206781
## 26   0.451077374639788   0.451971547128114   0.451213280984985
## 27  -0.249394860790073   -0.24685378499302  -0.238099383687614
## 28  -0.500270012772445  -0.550691457103626  -0.510436152294641
## 29   -0.36901555164323   -0.35285650691086  -0.370024180928006
## 30  -0.725920312960631  -0.667563193546124  -0.688814626434728
## 31  -0.492389993998681  -0.522111746921324  -0.465894730464716
## 32  -0.202905207551079   -0.26480839992179  -0.292622500902114
## 33  -0.180389342898959  -0.166475389123774  -0.176010128201919
## 34  -0.280092965959932   -0.28506954930987  -0.285625376772009
## 35  -0.496956582540389  -0.495310813407152  -0.492418070936552
## 36   0.132420964185526   0.147531373993871   0.152955098956241
## 37  -0.439692984287022  -0.428049964580853  -0.408506222704155
## 38 -0.0866872380711952 -0.0238950579062396 -0.0586368523523501
## 39  -0.801600787411434  -0.818742261539453  -0.827179140324979
## 40  -0.629104229233642  -0.698433459860655  -0.647620152698533
## 41  -0.952639090595151  -0.939635013138021  -0.954936780788727
## 42  -0.783561698283993  -0.749729369445071  -0.757946796162817
## 43  -0.720087117589684  -0.739201998029165  -0.707890476315829
## 44  -0.440945799655265  -0.479106197494646  -0.496974952252246
## 45   -0.34758122805968   -0.31987620926827  -0.330996899038082
## 46  -0.592772893121894  -0.553764261168887  -0.521154255435351
## 47  -0.522955492319348  -0.551097194926493  -0.580813345698717
## 48  -0.279745731912048  -0.337091713561426  -0.346837529532894
## 49  -0.559882480657377  -0.594309944847152  -0.637457774274131
## 50  -0.565603097342013  -0.610794376440904  -0.589586335706861
## 51  -0.573297171166768  -0.537602335592619  -0.524016978491573
## 52 -0.0921605812431852  0.0178619388618442 -0.0504735753378586
## 53   0.378639609908212   0.322494307534108   0.388400261234422
## 54   0.644358593524493   0.497648738828856    0.52017767417116
## 55   0.339338366864384   0.397229877476282   0.314877402490565
## 56   0.464259022234598   0.593843989561636   0.657318011333342
## 57  -0.275181526974871  -0.311992476898345  -0.302025724194023
## 58  -0.125512670890736  -0.208933679198547  -0.256986148452369
## 59  -0.689287169571965  -0.663844341192081  -0.634838033396388
## 60  -0.863585157830764  -0.820801029217732  -0.815658236065056
## 61   -0.81167069788508  -0.809590772827256  -0.807421700449126
## 62  -0.817921927869587  -0.809838326061946  -0.815134580543488
## 63   -1.04490809451533   -1.02909181256075   -1.02307502751483
## 64  -0.875178522120148  -0.862261418777509  -0.872447698215185
## 65  -0.847136114052873  -0.867073930143926  -0.852836161137731
## 66   -0.60658407532862  -0.648975477976955  -0.640571266620277
## 67   -0.63114632825106  -0.602346289795817  -0.632700295817198
## 68  -0.249453823255365  -0.215591884655068  -0.199469355969837
## 69   -0.73857166310839  -0.743969122484687  -0.742847279878596
```

```
## 70  -0.661951154661157    -0.6735937157231  -0.678029304000809
## 71  -0.692119989171404  -0.693363431797373  -0.692155584691164
## 72  -0.659101992354345  -0.681263537906366  -0.684431940724613
## 73  -0.507020772006829  -0.452150361219188  -0.391006616005924
## 74  -0.332784419741298  -0.245668337739766   -0.28854698869351
## 75    -0.9788117824403   -1.00775398914443   -1.01824759444642
## 76  -0.868342883162739  -0.910427902237891  -0.882931374050808
## 77   -1.02371684731411   -1.01621443583706   -1.02279444149328
## 78  -0.660497751936841  -0.664617268328112  -0.664267155082347
## 79  -0.758361507544584  -0.752966228936145  -0.759326636328087
## 80  -0.700063409675965  -0.711727595695378  -0.714686960435485
## 81  -0.596339903839864  -0.567062359226048  -0.576545014370926
## 82  -0.634320762080634  -0.576660767870094  -0.533365255194648
## 83   -0.26115813683958  -0.316842570487401  -0.358141594699545
## 84  -0.210226805416886  -0.310415783524985  -0.328095549139555
## 85    4.45399208977478    4.32144556000548    4.15170555241084
## 86    2.95247513079639    2.81965598585314    2.86463955076106
## 87    2.32158783621206    2.42159781962171    2.50768899626606
## 88  -0.237417643172478  -0.192422441849236  -0.226927834625466
## 89   0.349723019100279   0.343627454636523   0.347358671197955
## 90  -0.754287709951212  -0.766151872824512  -0.761255084443313
## 91   -0.65199952714938  -0.636030882935559  -0.656684204942614
## 92   0.549992571217839   0.626445565402749   0.651520165020459
## 93    2.03689425755071    1.95368569316065    2.01994632667922
## 94    2.63162934190844    2.45689753632021    2.32133277660351
## 95    2.49166058928536    2.61037908891561    2.67349290192894
## 96    3.38601413787471    3.57496250336517     3.6611262270308
##                M0_SE              M1_SE              M2_SE              M3_SE
## 1   0.240631628392496  0.184565230066764  0.178100267723097  0.175695271548145
## 2    0.15400191281537  0.159497802872058  0.160391368566891  0.165751703352831
## 3   0.100056436306682 0.0957365097182405 0.0967160525237215 0.0975452436314947
## 4   0.201888020169839   0.15349326957422  0.150825104864066  0.149957363553642
## 5   0.150471460166394  0.128056852083531  0.128415348445563   0.12663983900391
## 6   0.136116922233999  0.106787384728047  0.105687894242941  0.105634299909767
## 7   0.146194730648032  0.120519745848753  0.122368738887987  0.121350416950244
## 8   0.187435973773344  0.183904662126873  0.184199088524363   0.18327225694496
## 9   0.191503698751669  0.194128927524204  0.191920542332572  0.192819804797235
## 10  0.239722238171872  0.194188092200466  0.197630881756716  0.194533083574236
## 11  0.194781997211286  0.152309418185469  0.152182682788863  0.152007846100735
## 12  0.203669908002056  0.182645438581523  0.186323604317008  0.182667714967222
## 13  0.282113146564068  0.207836772109883  0.197820211850941   0.19371523742292
## 14  0.151559518339257  0.146638410045132  0.146207430897751  0.145320629337521
## 15 0.0753444592414178 0.0845982206268442 0.0901055216695646  0.093575370903633
## 16  0.161854639009579  0.186181948357745   0.19716368627042  0.208909187715111
## 17  0.124003338035432   0.17125971394684  0.174498562339109  0.168634051250921
## 18  0.119580608996942  0.149810284053553  0.141651745937346  0.133523729194992
## 19  0.112364197124771   0.12933238745608  0.121970203201477  0.124903834711222
## 20  0.207686388190761  0.216398624541769  0.214736247549186  0.222870093500488
## 21  0.213333643336625  0.208587089583698  0.208630868626311  0.208497217547595
## 22  0.332909702572127  0.250828360318382  0.260540165995242  0.251566721552368
## 23  0.244432238401717  0.155683276467442  0.156686131286105  0.159322650348403
## 24  0.171949031467728  0.110779878698901  0.109900617633196  0.110984126248342
## 25  0.539067240705544  0.457369348513525  0.442634180179879  0.436973623906372
## 26  0.418625101983436  0.417966476562807  0.417907883051434  0.417945723162249
## 27  0.277052483379672  0.272302028041247  0.273931681768458  0.274406944863942
## 28  0.349856080862799  0.235838821820204  0.222360787198055  0.211991589473117
```

```
## 29  0.358660072272081  0.249980958643288  0.249330435395869  0.252620382988006
## 30  0.283188379590793   0.16411608618658  0.175970837095562  0.187978697011551
## 31  0.291046654248652  0.207846402118586  0.223970393183189   0.21785248922635
## 32   0.29399535417708  0.282656599238324  0.283470363008383   0.27070226390654
## 33  0.276249112101183  0.295240322884081  0.288120895437942  0.290923230032554
## 34   0.29727917431112   0.26648647386072  0.267620524605887  0.266556945635163
## 35  0.268518801512745   0.22303628981403  0.223042900292717  0.223368945893549
## 36   0.42033512333226  0.355935400108651  0.352442463457721  0.355449273227558
## 37  0.233633113183542  0.232261501867347  0.236126525633934  0.238629788992213
## 38  0.291765463770329   0.31180804639509  0.312650508839386  0.326156700293747
## 39  0.198644576052354    0.1656994592219  0.157652527105959  0.153997137055453
## 40  0.352474384289625  0.210746735716081  0.195040435176889  0.180043709388396
## 41  0.229816813804335  0.125397334823581  0.124884397268588  0.127791552496562
## 42  0.249122147476321     0.156094963226   0.16155222875475  0.168934680539822
## 43  0.235708803883618   0.16696676850551  0.175305888311919  0.171206124598129
## 44  0.250956962180113  0.235274953165099  0.235829904916917  0.227545606307223
## 45  0.233140038916213   0.26747800547026  0.256091422454388  0.262054165896724
## 46  0.178906208035154  0.211039903491147  0.202923805130683  0.211388059997881
## 47  0.155557890614486  0.218699960859252  0.218059135124271  0.211961673356175
## 48  0.204434249242752  0.259432525574587  0.270815095754554  0.258340513035128
## 49  0.359970261100078  0.226576744994951  0.210079295876073  0.202657529119659
## 50  0.234577104019623  0.210520326522344   0.20883234476948  0.199089823248624
## 51  0.154561844869607  0.193161443368221  0.207173134353107  0.214887444716422
## 52  0.187326979524373   0.29018575790964  0.311147279548551  0.334764889302362
## 53  0.201881290710865  0.407553344984952  0.412880865077056  0.400504905040891
## 54  0.266225625994026  0.496739007982754  0.470309950055145  0.438311607574469
## 55  0.279894719069205  0.432658536566812  0.404412405442949  0.416656573654307
## 56  0.386697883496146  0.435457247891457   0.43141315677231  0.459096924902891
## 57  0.308713584345124  0.257076460018124  0.271585799401026  0.263568181077111
## 58  0.434135580617151  0.286606931169662  0.303933661783985  0.285815255737138
## 59      0.33779115274  0.180570720382812  0.182124656750971  0.187660400067047
## 60  0.289093191721655   0.15191441146334  0.144446940865999  0.153772857915903
## 61  0.189815850275836  0.154992615536756  0.153818219815398  0.154377097061549
## 62   0.15244963607355  0.152388279041089  0.152422564325385  0.154316532434031
## 63 0.0920004054352581 0.0975266269779466  0.101950942476536  0.105632140601508
## 64  0.162151821552604  0.138001569986973  0.139691843784149  0.142677041394847
## 65  0.138129426721176  0.145348623536144  0.145937231642824  0.141617048631768
## 66  0.163048627948034  0.209619188783693   0.19941981623155  0.190038613255739
## 67  0.164900558226977  0.204170191499231  0.193966673368954  0.200402211472581
## 68  0.268408725648608  0.279887136516514  0.278866272453992  0.286306731265086
## 69  0.176745970956955  0.168613283467724  0.170077789687566  0.168953186456209
## 70  0.229107186147494  0.184494805426875   0.18711409620067  0.184578584415098
## 71  0.227144596909983   0.18045465136386  0.180406941459435  0.180189941221725
## 72  0.197156800465068  0.182916318958998  0.187758345788649  0.182884538650724
## 73   0.12557305866978  0.204562158340442  0.221567847851523   0.23364439474111
## 74  0.222846628805927  0.259396806565448  0.259968662914515  0.279087558976539
## 75  0.164779016123743  0.128359371531529  0.117575483787215  0.111349564809171
## 76  0.258070009870868   0.15203348393726   0.14192009374686  0.132767098823792
## 77  0.190777338090176  0.107777058857289  0.107669329611193  0.109479491764578
## 78  0.221379609596567  0.189808364258265  0.187738650338312   0.18687787410217
## 79  0.178633159109043  0.168475911269739  0.166160563408196  0.167424871319603
## 80  0.189285636508368  0.178903124285669  0.179011434739325  0.176503160374316
## 81  0.186927514629649  0.211874833236907  0.201876076784035  0.208346863117891
## 82  0.147622637048285  0.205698109835153  0.193509583782648  0.206240260826517
## 83  0.154250601804356  0.275931806229799  0.275761513660489  0.263426020351178
## 84  0.133015141009763  0.266012745315057  0.286996664786138  0.264848940018603
```

```
## 85   1.35056548886427   1.16106833000784   1.11269734648023   1.08743716141859
## 86  0.856780929336309  0.836851414658086  0.841706417616919   0.81679219585354
## 87  0.610319240171571  0.688018058545919   0.72785646145534  0.745066483333707
## 88  0.245429338729944  0.258347552448225  0.266131937023478  0.274112824609882
## 89  0.382730787661212  0.372543378773005  0.372084205951018  0.370701622053606
## 90  0.175860115380694  0.177740133254499   0.17287639690841   0.17074900316957
## 91  0.183112547253824  0.197639157602701  0.191333781155218  0.194195172049515
## 92  0.396941192822215  0.405701973575541  0.408217150325768    0.4216571548689
## 93   0.72247363788971  0.641437827259066  0.676531964755013   0.66080132712469
## 94  0.958500419592112  0.746253135670379  0.783853435405321  0.751482579081579
## 95   1.04417959530359  0.766254420334911  0.758597058881108  0.779134593242714
## 96   1.22321954188529  0.966678828174883  0.920004231402793  0.952965277669252
##                 M4_SE
## 1   0.173295288766188
## 2    0.16278581094919
## 3   0.097943564324042
## 4   0.150352646992871
## 5    0.12858013353864
## 6   0.105722533874279
## 7   0.122796492071917
## 8   0.183333506927745
## 9   0.192522604074928
## 10  0.193236094603246
## 11  0.152170902913646
## 12  0.182370780548946
## 13  0.190101141362281
## 14  0.145587951641568
## 15 0.0938455961613114
## 16  0.202395384407566
## 17  0.175083186676208
## 18  0.133800531174758
## 19  0.121657784488282
## 20  0.227807132416911
## 21  0.208498512767341
## 22  0.247551003228999
## 23  0.162561847224776
## 24  0.110963546570568
## 25  0.428003589480115
## 26   0.41778922096541
## 27  0.276204827147694
## 28  0.220261115914709
## 29  0.249094202703599
## 30  0.183613096983672
## 31   0.22940074867972
## 32   0.26498925274851
## 33  0.288964486337983
## 34  0.266442988495088
## 35  0.223963236228955
## 36  0.356565370612052
## 37  0.242862535088045
## 38  0.318626217356902
## 39  0.152172274481486
## 40  0.191051887451544
## 41  0.124480515304145
## 42  0.167156767571452
## 43  0.177990111300232
```

```
## 44  0.223675079106385
## 45  0.259643251421475
## 46  0.218452466817301
## 47   0.20552484026542
## 48  0.256228175774946
## 49  0.193344379549044
## 50  0.203665477877168
## 51  0.217819257912906
## 52  0.320026640759147
## 53  0.414743254229418
## 54  0.443192564314775
## 55  0.398898980130878
## 56    0.4728167907042
## 57  0.265723140453492
## 58  0.275450506601403
## 59  0.193918420817848
## 60  0.154876586560978
## 61  0.154857457987949
## 62  0.153138311541355
## 63  0.106966745561694
## 64   0.14041263909035
## 65  0.144776209406907
## 66  0.191904153704154
## 67  0.193660375564043
## 68  0.289889047606939
## 69  0.169201215068273
## 70  0.183593199711139
## 71  0.180456966862464
## 72  0.182179303607178
## 73  0.247104541674957
## 74  0.269650755393557
## 75  0.109036653515703
## 76  0.138816355796724
## 77   0.10802702600926
## 78  0.186953700945109
## 79   0.16602230187032
## 80   0.17585063357735
## 81  0.206259056462377
## 82  0.215770903817469
## 83  0.254336647202335
## 84  0.260959203911031
## 85   1.05665176272071
## 86  0.824745024178828
## 87  0.760444643262836
## 88  0.267875268347906
## 89   0.37133059915505
## 90  0.171633467684166
## 91  0.190470925674503
## 92  0.426119854709016
## 93  0.672630335089782
## 94  0.726930825790303
## 95  0.790368258696943
## 96  0.968328578573009
```

Function definitions

```
# Zongsheng Liu (s2097920)
data(ghcnd_stations, package = "StatCompLab")
data(ghcnd_values, package = "StatCompLab")

library(dplyr)
library(ggplot2)
library(tidyr)
library(StatCompLab)



set.seed(20011016)

#combine ghcnd-values and ghcnd_sttions, group the m by ID
ghcnd <- left_join(ghcnd_values, ghcnd_stations, by = "ID")

#define seasonal infor
winter_months <- c(1, 2, 3, 10, 11, 12)
summer_months <- c(4, 5, 6, 7, 8, 9)

year_of_interest <- 2001
# Filter data for the year 2001
ghcnd_values_year <- ghcnd_values %>%
  filter(Year == year_of_interest)
# Compute mean temperature and total precipitation by station ID
ghcnd_values_agg <- ghcnd_values_year %>%
  group_by(ID) %>%
  summarize(mean_temp = mean((Value[Element == "TMAX"]+Value[Element == "TMIN"])/2),
            total_precip = sum(Value[Element == "PRCP"]))

# group mean temperature and total precipitation by Id and also add geographic info
ghcnd_data <- left_join(ghcnd_values_agg, ghcnd_stations, by = "ID")

#select 2001 as year of interest and filter the data
ghcnd_filtered <- ghcnd_values %>%
  filter(Year == year_of_interest)
# Merge the ghcnd_filtered data with the ghcnd_stations data to get the station location info
rmation
ghcnd_merged <- ghcnd_filtered %>%
  left_join(ghcnd_stations, by = "ID")

# filter prcp data for the plots below
precip_data <- ghcnd_merged %>%
  filter(Element == "PRCP")



# Plot mean temperature and total precipitation  in 2001 as a map
plot1 <-function(){
  ggplot(ghcnd_data, aes(x = Longitude, y = Latitude)) +
    geom_point(aes(size = mean_temp, color = total_precip)) +
    scale_size_continuous(name = "Mean temperature (C)") +
    scale_color_continuous(name = "Total precipitation (mm)") +
    labs(title = "Temperature and Precipitation in 2001")
}
```

```
# Plot all temperature data points in one plot
plot2 <- function(){
  temp_data <- ghcnd_merged %>%
    filter(Element == "TMIN" | Element == "TMAX") %>%
    pivot_wider(names_from = "Element", values_from = "Value")
  ggplot(temp_data, aes(x = DecYear, y = TMIN, group = ID)) +
    geom_line(aes(color = Name)) +
    geom_line(aes(y = TMAX, color = Name)) +
    labs(x = "Date", y = "Temperature (Celsius)") +
    ggtitle(paste("Temperature data for all stations in 2001")) +
    theme_bw()
}

# 8 subplots which grouped by ID for monthly average TMAX and TMIn
plot3 <- function(){
  ghcnd %>%
    filter(Element %in% c("TMIN", "TMAX"),Year == year_of_interest) %>%
    group_by(ID, Name, Element, Month) %>%
    summarise(Value = mean(Value), .groups = "drop") %>%
    mutate(Season = ifelse(Month %in% winter_months, "Winter", "Summer")) %>%
    mutate(Type = ifelse(Element == "TMIN" & Season == "Winter", "Winter TMIN ",  # separate
data by season on the plot
                         ifelse(Element == "TMIN" & Season == "Summer", "Summer TMIN",
                                ifelse(Element == "TMAX" & Season == "Winter", "Winter TMAX",
"Summer TMAX")))) %>%
    ggplot(aes(Month, Value, colour = Type)) +
    ggtitle("Monthly Average TMAX & TMIN for All Stations in 2001") +
    geom_point() +
    facet_wrap(~ Name, ncol=2)
}


# Plot all precipitation data in 2001 for all stations in 1 plot
plot4 <- function(){
  ggplot(precip_data, aes(x = DecYear, y = Value, group = ID)) +
    geom_line(aes(color = Name)) +
    labs(x = "Date", y = "Precipitation (mm)") +
    ggtitle(paste("Precipitation data for all stations in 2001")) +
    theme_bw()
}



# 8 subplots for monthly average PRCP
plot5 <- function(){
  precip_data %>%
    filter(Element %in% "PRCP",Year == year_of_interest) %>%
    group_by(ID, Name, Element, Month) %>%
    summarise(Value = mean(Value), .groups = "drop") %>%
    mutate(Season = if_else(Month %in% winter_months, "Winter", "Summer")) %>%  #plot data in
summer or winter sperately
    ggplot(aes(Month, Value, colour = Season)) +
    ggtitle("Monthly Average PRCP for all Stations in 2001") +
    geom_point() +
    facet_wrap(~ Name, ncol=2)
```

```
}

# 8 subplots for all TMAX and TMIN data whuch grouped by season
plot6 <- function(){
  ghcnd %>%
    filter(Element %in% c("TMIN", "TMAX"), Year == year_of_interest, Month %in% c(winter_mont
hs, summer_months)) %>%
    mutate(Season = ifelse(Month %in% winter_months, "Winter", "Summer")) %>%
    mutate(Type = ifelse(Element == "TMIN" & Season == "Winter", "Winter TMIN ",   # group th
em by season info
                         ifelse(Element == "TMIN" & Season == "Summer", "Summer TMIN",
                                ifelse(Element == "TMAX" & Season == "Winter", "Winter TMAX",
"Summer TMAX")))) %>%
    ggplot(aes(Month, Value, colour = Type)) +
    ggtitle("TMAX & TMIN in 2001") +
    geom_jitter() +
    facet_wrap(~ Name, ncol=2)
}


# 8 subplots for all TMAX and TMIN data whuch grouped by season
plot7 <- function() {
  precip_data %>%
    filter(Element == "PRCP",Year == year_of_interest) %>%
    group_by(ID, Name, Year, Month) %>%
    mutate(Season = if_else(Month %in% winter_months, "Winter", "Summer")) %>% # group them b
y season info
    ggplot(aes(Month, Value, color = Season)) +
    ggtitle("PRCP in 2001") +
    geom_jitter() +
    facet_wrap(~ Name, ncol=2)
}

#plot1()
#plot2()
#plot3()
#plot4()
#plot5()
#plot6()
#plot7()

# Add season information to the ghcnd_values data
ghcnd_values <- ghcnd_values %>%
  mutate(Season = ifelse(Month %in% winter_months, "Winter", "Summer"))

#Add summer column
ghcnd_values <- ghcnd_values %>%
  mutate(Summer = ifelse(Month %in% summer_months, TRUE, FALSE))

# compute T for getting p_value
test_stat <- function(data){
  winter_data <- subset(data, Month %in% c(1, 2, 3, 10, 11, 12))
  summer_data <- subset(data, Month %in% c(4, 5, 6, 7, 8, 9))
  return(abs(mean(winter_data$Value) - mean(summer_data$Value)))
}
```

```r
#compute CI for p-value
p_value_CI <- function(p_value, sd_p, N, alpha){
  #case 1: p-value =0
  if (p_value==0){
    upper <- 1- (alpha/2)^(1/N)
    CI <- c(0,upper)
  }
  #case 2: p-value != 0
  else{
    z <- qnorm(1 - alpha/2) #compute z socre correspond to different alpha value
    lower <- p_value - z*sd_p
    upper <- p_value + z*sd_p
    CI <- c(lower, upper)
  }
  CI <- pmax(0, pmin(1, CI))
  return(CI)
}

#Construct a Monte Carlo permutation test
mc_permutation <- function(data, N, alpha){
  #N: the number of permutation would perform
  #alpha: alpha value for hypothesis testing
  #data: ghcnd joined by ghcnd_values and ghcnd_stations and grouped by ID
  p_value <- c()
  sd <- c()
  CI_l <- c()
  CI_u <- c()
  prcp_data <- data %>%  #filter out PRCP data
    filter(Element %in% "PRCP")
  # use for loop to loop over all stations
  for (i in unique(prcp_data$ID)){
    rain_s <- prcp_data %>% filter(ID==i)
    t_stats <- test_stat(data=rain_s) #compute T
    t_p <- replicate(N,
                     {rain_s$Value <- sample(rain_s$Value)
                     test_stat(rain_s)}) #make permutation N times by replicate
    p <- mean(t_p >= t_stats) #compute p-value  by (the number of times t_p >= t_stats )/(N)
    p_value <- c(p_value,p)
    sd_p <- sqrt(p*(1-p)/N) # compute sd for p_value
    sd <- c(sd, sd_p)
    CI_p <- p_value_CI(p,sd_p,N,alpha) #construct CI
    CI_l <- c(CI_l, CI_p[1]) #get lower bound
    CI_u <- c(CI_u, CI_p[2]) #get upper bound
  }
  #form required restults as a data frame
  test_result <- data.frame(ID = unique(prcp_data$ID),
                            p_value = p_value,
                            standard_deviations = sd,
                            CI_lower_bound = CI_l,
                            CI_upper_bound = CI_u)
  return(test_result)
}

#run the above MC permutation test with N=1000
table1 <- mc_permutation(ghcnd, 10000, 0.05)
```

```r
# make a new data frame for the model estimation tsak below which contains  ID, Longitude, La
titude, Elevation, Value_sqrt_avg, DecYear_avg
ghcnd_lm <- ghcnd_values %>%
  filter(Element == "PRCP") %>%
  group_by(ID, Year, Month) %>%
  summarise(Value_sqrt_avg = sqrt(mean(Value)),
            DecYear_avg = mean(DecYear)) %>%
  group_by(ID, Year, Month) %>%
  summarise(Value_sqrt_avg = mean(Value_sqrt_avg),
            DecYear_avg = mean(DecYear_avg)) %>%
  left_join(select(ghcnd_stations, ID, Longitude, Latitude, Elevation), by = "ID")

#define all 5 models as follow:

# Model M0
M0 <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg, data = ghcnd_lm)

# Model M1
M1 <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
           cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg),
           data = ghcnd_lm)

# Model M2
M2 <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
           cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
           cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg),
           data = ghcnd_lm)

# Model M3
M3 <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
           cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
           cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) +
           cos(6 * pi * DecYear_avg) + sin(6 * pi * DecYear_avg),
           data = ghcnd_lm)

# Model M4
M4 <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
           cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
           cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) +
           cos(6 * pi * DecYear_avg) + sin(6 * pi * DecYear_avg) +
           cos(8 * pi * DecYear_avg) + sin(8 * pi * DecYear_avg),
           data = ghcnd_lm)

#summary(M0)
#summary(M1)
#summary(M2)
#summary(M3)
#summary(M4)

#combine above models into a single function for later
Model <- function(data, freq = 4){
  # data: data set for model training
  #freq: defined as k in model
  # it would return  model summary based on the freq and training data
  if (freq == 0){   #Model m0
    M <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg, data = data)
```

```
  }else if(freq == 1){  #Model m1
    M <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
                cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg),
              data = data)
  }else if(freq == 2){ #Model M2
    M <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
                cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
                cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg),
              data = data)
  }else if(freq == 3){   #Model M3
    M <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
                cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
                cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) +
                cos(6 * pi * DecYear_avg) + sin(6 * pi * DecYear_avg),
              data = data)
  } else{   #Model m3
    M <- lm(Value_sqrt_avg ~ Longitude + Latitude + Elevation + DecYear_avg +
                cos(2 * pi * DecYear_avg) + sin(2 * pi * DecYear_avg) +
                cos(4 * pi * DecYear_avg) + sin(4 * pi * DecYear_avg) +
                cos(6 * pi * DecYear_avg) + sin(6 * pi * DecYear_avg) +
                cos(8 * pi * DecYear_avg) + sin(8 * pi * DecYear_avg),
              data = data)
  }
  return (M)
}


#perform cross_validation
cv_station <- function(data){
  #use ghcnd_lm again as input
  all_result <-c()
  ds_table1 <-c()
  se_table1 <-c()
  score_table3 <- matrix(nrow = 12, ncol=5)
  score_table4 <- matrix(nrow = 12, ncol=5)
  # loop over all the stations
  for(station_id in unique(data$ID)){
    monthly_score <- matrix(0, nrow = 12, ncol = 10)
    mean_ds <- c()
    mean_se <- c()
    #loop over all the models
    for(i in 0:4){
      for(month in 1:12){
        #loop over all the month to get a well organized data
        fit <- Model(data=data %>% filter(ID != station_id), freq = i)
        pred <- predict(fit, newdata =data %>% filter(ID == station_id, Month == month), se.f
it = TRUE)
        residual_var <- sum(fit$residuals^2)/fit$df.residual
        sd_pred <- sqrt(pred$se.fit^2 + residual_var)
        #compute ds score
        ds_score <- proper_score(type = "ds", obs = data %>% filter(ID == station_id, Month =
= month)  %>% pull(Value_sqrt_avg), mean = pred$fit, sd = sd_pred)
        #compute se score
        se_score <- proper_score("se", obs = data %>% filter(ID == station_id, Month == mont
h)  %>% pull(Value_sqrt_avg), mean = pred$fit)
        #construct a data frame with size contains all result for each station
```

```
        monthly_score[month, i+1] <- mean(ds_score)
        monthly_score[month, i+6] <- mean(se_score)
        #compute mean prediction scores
        mean_ds <- c(mean_ds, mean(ds_score))
        mean_se <- c(mean_se, mean(se_score))
      }
      #organize two mean distribution scores
      ds_table1 <- c(ds_table1, mean(mean_ds))
      se_table1 <- c(se_table1, mean(mean_se))
    }
    #finally form a data frame with size 96*10 contains all the results
    all_result <-rbind(all_result, monthly_score)
  }

  #use for loop to get values for overall mean scores for each month
  for(i in 1:12){
    for(j in 1:5){
      score_table3[i,j] = mean(all_result[seq(i, 96, by = 12),j])
      score_table4[i,j] = mean(all_result[seq(i, 96, by = 12),5+j])

    }
  }
  #organize all tables and adding row names, column names
  score_table1 <- matrix(ds_table1, nrow = 8, byrow = TRUE)
  row.names(score_table1) <- unique(ghcnd_lm$ID)
  colnames(score_table1) <- c("M0", "M1", "M2", "M3", "M4")

  score_table2 <- matrix(se_table1, nrow = 8, byrow = TRUE)
  row.names(score_table2) <- unique(ghcnd_lm$ID)
  colnames(score_table2) <- c("M0", "M1", "M2", "M3", "M4")

  row.names(score_table3) <- unique(ghcnd_lm$Month)
  colnames(score_table3) <- c("M0", "M1", "M2", "M3", "M4")

  row.names(score_table4) <- unique(ghcnd_lm$Month)
  colnames(score_table4) <- c("M0", "M1", "M2", "M3", "M4")


  colnames(all_result) <- c("M0_DS", "M1_DS", "M2_DS", "M3_DS", "M4_DS","M0_SE", "M1_SE", "M2
_SE", "M3_SE", "M4_SE")
  ID <- rep(unique(ghcnd_lm$ID),each=12)
  month <- rep(1:12,8)
  all_result <- as.data.frame(cbind(ID,month, all_result))


  return(list(score_table1 = score_table1, score_table2 = score_table2, score_table3 = score_
table3, score_table4 = score_table4 ,all_result=all_result))
}
cv_station_result <- cv_station(ghcnd_lm)
#cv_station_result$all_result
#cv_station_result$score_table1
#cv_station_result$score_table2
#cv_station_result$score_table3
#cv_station_result$score_table4

#bar chart to show ds scores at each station
```

```r
score_plot1 <- function(){
  score_df <- as.data.frame(cv_station_result$score_table1)
  score_df$ID <- row.names(cv_station_result$score_table1)
  # Reshape the data frame to long format
  score_long <- tidyr::gather(score_df, key = "Model", value = "Score", -ID)
  # Generate the bar chart
  ggplot(score_long, aes(x = ID, y = Score, fill = Model)) +
    geom_bar(stat = "identity", position = "dodge", width = 0.5) +
    ggtitle("Dawid-Sebastiani Scores for Different Models across Stations") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
    labs(x = "Station ID", y = "Score")
}

#bar chart to show se scores at each station
score_plot2 <- function(){
  score_df <- as.data.frame(cv_station_result$score_table2)
  score_df$ID <- row.names(cv_station_result$score_table2)
  # Reshape the data frame to long format
  score_long <- tidyr::gather(score_df, key = "Model", value = "Score", -ID)
  # Generate the bar chart
  ggplot(score_long, aes(x = ID, y = Score, fill = Model)) +
    geom_bar(stat = "identity", position = "dodge", width = 0.5) +
    ggtitle("Squared Error Scores for Different Models across Stations") +
    theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1)) +
    labs(x = "Station ID", y = "Score")
}

#bar chart to show overall ds scores in each month
score_plot3 <- function(){
  # Convert row names to factor with ordered levels
  months <- unique(ghcnd_lm$Month)
  score_table3 <- as.data.frame(cv_station_result$score_table3)
  score_table3$month_factor <- factor(row.names(score_table3), levels=months, ordered=TRUE)
  # Generate the bar chart
  score_table3 %>%
    pivot_longer(cols=-month_factor, names_to="Model", values_to="Score") %>%
    ggplot(aes(x=month_factor, y=Score, fill=Model)) +
    geom_col(position="dodge") +
    xlab("Month") +
    ylab("Score") +
    ggtitle("Dawid-Sebastiani Scores for Different Models across Different Months") +
    theme(plot.title = element_text(hjust = 0.5))
}


#bar chart to show overall ds scores in each month
score_plot4 <- function(){
  # Convert row names to factor with ordered levels
  months <- unique(ghcnd_lm$Month)
  score_table4 <- as.data.frame(cv_station_result$score_table4)
  score_table4$month_factor <- factor(row.names(score_table4), levels=months, ordered=TRUE)
  # Generate the bar chart
  score_table4 %>%
    pivot_longer(cols=-month_factor, names_to="Model", values_to="Score") %>%
    ggplot(aes(x=month_factor, y=Score, fill=Model)) +
    geom_col(position="dodge") +
```

```
    xlab("Month") +
    ylab("Score") +
    ggtitle("Squared Error Scores for Different Models across Each Months") +
    theme(plot.title = element_text(hjust = 0.5))
}


#construct a new data set that ds score in each month for every station
ds_table <- cv_station_result$all_result %>%
  gather(key = "Model_type", value = "value", M0_DS:M4_DS) %>%
  select(Model_type, ID, month, value) %>%
  mutate(month = as.numeric(month),
         value = as.numeric(value))

#construct a new data set that se score in each month for every station
se_table <- cv_station_result$all_result %>%
  gather(key = "Model_type", value = "value", M0_SE:M4_SE) %>%
  select(Model_type, ID, month, value) %>%
  mutate(month = as.numeric(month),
         value = as.numeric(value))

#plot ds score grouped by model for every month every station
score_plot5 <- function(){
  ggplot(ds_table, aes(month, value, color = ID)) +
    geom_line() +
    facet_wrap(~Model_type, ncol=2) + # split the plot based on the Model_type
    ggtitle("DS Scores of different stations in each month") + # add title
    xlab("Month") + # change the name of x-axis
    ylab("DS Score") + # change the name of y-axis
    scale_x_continuous(limits = c(1, 12), breaks = seq(1, 12, by = 1)) # set x-axis limits an
d ticks
}

#construct a new data set that se score in each month for every station
score_plot6 <- function(){
  ggplot(se_table, aes(month, value, color = ID)) +
    geom_line() +
    facet_wrap(~Model_type, ncol=2) + # split the plot based on the Model_type
    ggtitle("SE Scores of different stations in each month") + # add title
    xlab("Month") + # change the name of x-axis
    ylab("Score") + # change the name of y-axis
    scale_x_continuous(limits = c(1, 12), breaks = seq(1, 12, by = 1)) # set x-axis limits an
d ticks
}

#score_plot1()
#score_plot2()
#score_plot3()
#score_plot4()
#score_plot5()
#score_plot6()
```