Rochester Weather Patterns Since 1926

Cole Durham

1. Introduction

In this project, we will explore the trends exhibited in snowfall and maximum temperature data from Rochester, New York, from January of 1926 until October, 2025. The full dataset contains columns for the daily maximum and minimum temperatures, the amount of precipitation, the amount of snowfall, and the total depth of snow measured on the ground.

```
Rochester <- read.csv("Rochester.csv")
head(Rochester)
```

```
        Date TAVG..Degrees.Fahrenheit. TMAX..Degrees.Fahrenheit.
1 1926-01-02                         NA                        33
2 1926-01-03                         NA                        35
3 1926-01-04                         NA                        42
4 1926-01-05                         NA                        49
5 1926-01-06                         NA                        38
6 1926-01-07                         NA                        29
  TMIN..Degrees.Fahrenheit. PRCP..Inches. SNOW..Inches. SNWD..Inches.
1                        28          0.02           0.3             3
2                        32          0.00           0.0             2
3                        35          0.14           0.0             0
4                        38          0.00           0.0             0
5                        29          0.00           0.0             0
6                        16          0.04           0.4             0
```

We will convert the columns `TMAX..Degrees.Fahrenheit.` and `SNOW..Inches.` to separate objects and identify any missing values.

```r
#Create snowfall set, then rename col
Rochester.snowfall <- Rochester %>% select("Date", "SNOW..Inches.")

Rochester.snowfall <- Rochester %>%
  select(Date, SNOW..Inches.) %>%
  rename(snowfall = SNOW..Inches.)

#Create max temp set, rename col
Rochester.Tmax <- Rochester %>% select("Date", "TMAX..Degrees.Fahrenheit.")

Rochester.Tmax <- Rochester %>%
select(Date, TMAX..Degrees.Fahrenheit.) %>%
rename(tmax = TMAX..Degrees.Fahrenheit.)

#Make Date col a true datetime object
Rochester.snowfall$Date <- as.Date(Rochester.snowfall$Date, format = "%Y-%m-%d")
Rochester.Tmax$Date     <- as.Date(Rochester.Tmax$Date, format = "%Y-%m-%d")

#Make Year, Month, Day Columns: Should've done before splitting
Rochester.snowfall$Month <- month(Rochester.snowfall$Date)
Rochester.snowfall$Year <- year(Rochester.snowfall$Date)
Rochester.snowfall$Day <- day(Rochester.snowfall$Date)


Rochester.Tmax$Month <- month(Rochester.Tmax$Date)
Rochester.Tmax$Year <- year(Rochester.Tmax$Date)
Rochester.Tmax$Day <- day(Rochester.Tmax$Date)


#Determine dates (rows) with snowfall entered as NA
(snow.na <- Rochester.snowfall[is.na(Rochester.snowfall$snowfall), ])
```

```
            Date snowfall Month Year Day
55    1926-02-25       NA     2 1926  25
1377  1929-10-09       NA    10 1929   9
1378  1929-10-10       NA    10 1929  10
1379  1929-10-11       NA    10 1929  11
1380  1929-10-12       NA    10 1929  12
1381  1929-10-13       NA    10 1929  13
1382  1929-10-14       NA    10 1929  14
1383  1929-10-15       NA    10 1929  15
1384  1929-10-16       NA    10 1929  16
```

```
2389  1932-07-17       NA     7 1932  17
5430  1940-11-13       NA    11 1940  13
5431  1940-11-14       NA    11 1940  14
5432  1940-11-15       NA    11 1940  15
5433  1940-11-16       NA    11 1940  16
5434  1940-11-17       NA    11 1940  17
5435  1940-11-18       NA    11 1940  18
5436  1940-11-19       NA    11 1940  19
5437  1940-11-20       NA    11 1940  20
5438  1940-11-21       NA    11 1940  21
5439  1940-11-22       NA    11 1940  22
5440  1940-11-23       NA    11 1940  23
5441  1940-11-24       NA    11 1940  24
5442  1940-11-25       NA    11 1940  25
5443  1940-11-26       NA    11 1940  26
5444  1940-11-27       NA    11 1940  27
5445  1940-11-28       NA    11 1940  28
5446  1940-11-29       NA    11 1940  29
5447  1940-11-30       NA    11 1940  30
7987  1947-11-14       NA    11 1947  14
25750 1996-07-02       NA     7 1996   2
25753 1996-07-05       NA     7 1996   5
25754 1996-07-06       NA     7 1996   6
25779 1996-07-31       NA     7 1996  31
```

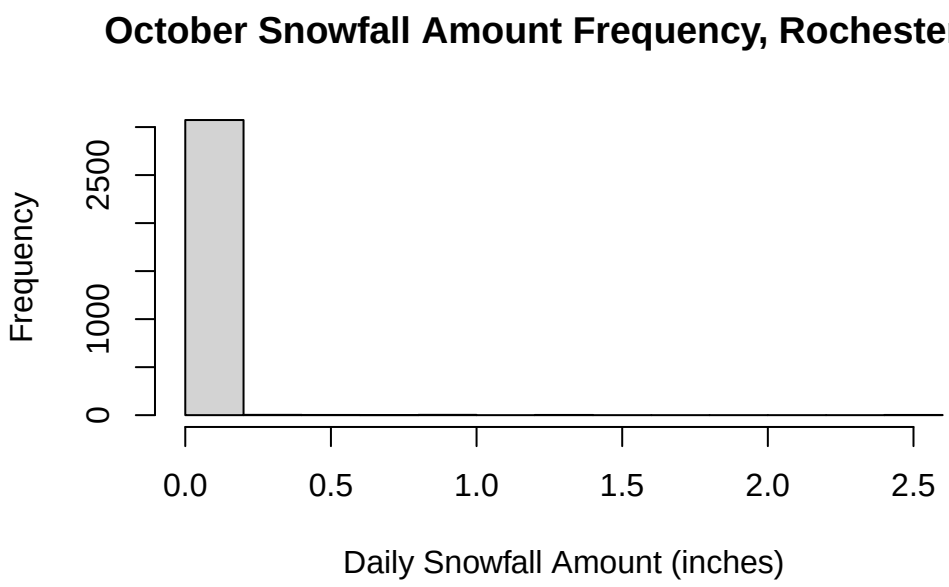
```
#Determine dates (rows) with max temp entered as NA
(Tmax.na <- Rochester.Tmax[is.na(Rochester.Tmax$tmax),])
```

```
           Date tmax Month Year Day
2389 1932-07-17   NA     7 1932  17
6259 1943-02-20   NA     2 1943  20
```

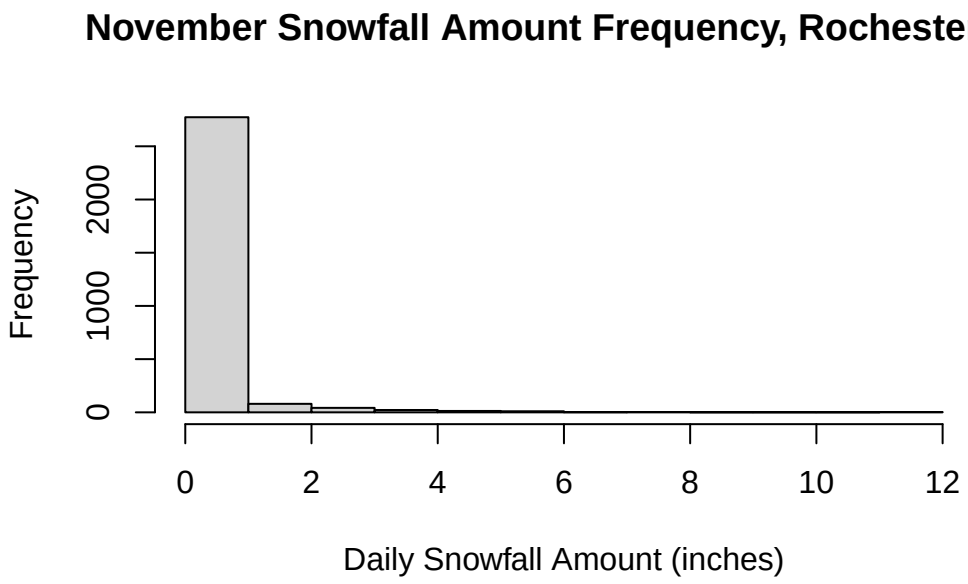
Unfortunately, we are presented with an immediate issue in the snowfall data: eight consecutive missing values in October, 1929 and almost twenty consecutive missing values in November, 1940. If these were from the warmer months of March-August, we could impute zeros with no hesitation; however, Rochester weather can be unruly in autumn. There are two ways to fix the largest gaps in the data, the first of which is to simply drop the portion of the observed values prior to 1941. An alternative, since standard techniques (e.g. interpolation, sample mean, last-observation-carry-forward) are generally insufficient for such large gaps, is to randomly sample from the same month in adjacent years.

To get a sense of the daily snowfall amounts and maximum temperatures in October and November over the years, we can plot histograms.

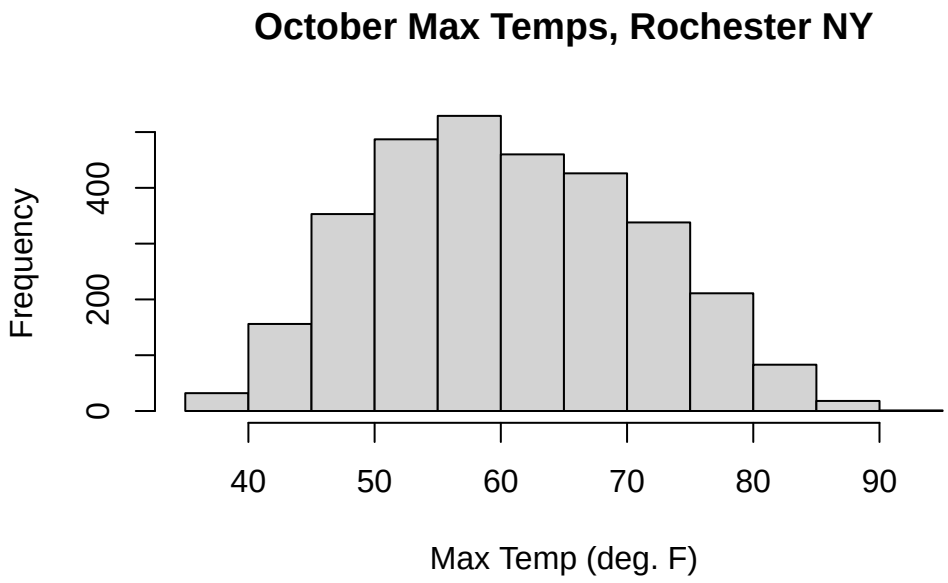
```r
hist(Rochester.snowfall[grepl("-10-", Rochester.snowfall$Date), ]$snowfall, main = "October S
     xlab = "Daily Snowfall Amount (inches)")
```



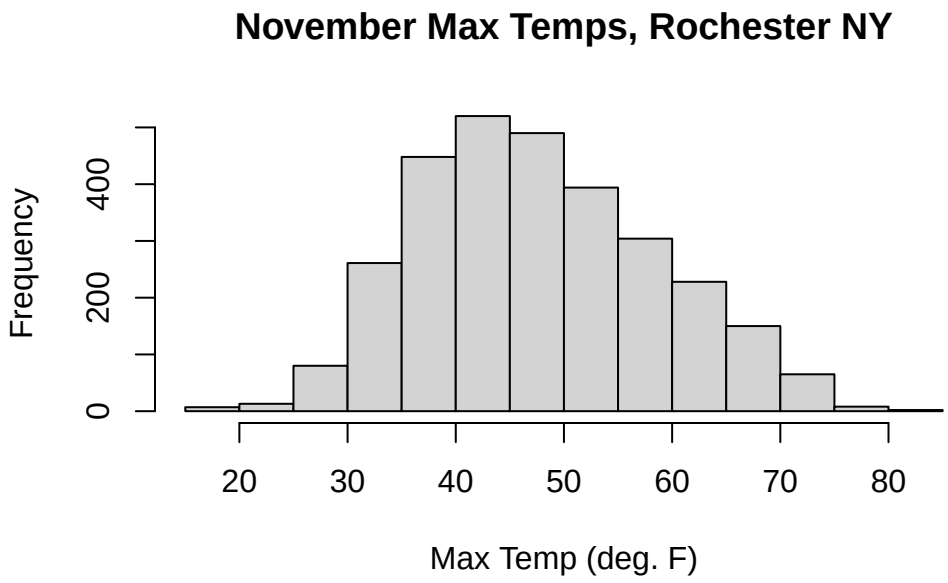

```r
hist(Rochester.snowfall[grepl("-11-", Rochester.snowfall$Date), ]$snowfall, main = "November
     xlab = "Daily Snowfall Amount (inches)")
```




```
hist(Rochester.Tmax[grepl("-10-", Rochester.Tmax$Date), ]$tmax, main = "October Max Temps, Ro
     xlab = "Max Temp (deg. F)")
```



```
hist(Rochester.Tmax[grepl("-11-", Rochester.Tmax$Date), ]$tmax, main = "November Max Temps, R
     xlab = "Max Temp (deg. F)")
```



From the histograms for snowfall in October and November, it would not be unreasonable to simply enter zeros for all missing days, but we will take the extra step to sample from the same month in adjacent years.

2. Data Cleaning and Organization

The largest gaps in the snowfall data occur in October, 1929 and November, 1940. These values will be imputed by sampling with replacement from the same month in adjacent years.

```
#Identify NA rows from October, 1929
oct29_idx <- which(
  Rochester.snowfall$Month == 10 &
  Rochester.snowfall$Year == 1929 &
  is.na(Rochester.snowfall$snowfall)
)

#Identify NA rows from November, 1940
nov40_idx <- which(
  Rochester.snowfall$Month == 11 &
```

```
  Rochester.snowfall$Year == 1940 &
  is.na(Rochester.snowfall$snowfall)
)

#Gather same months from adjacent years (Oct 1927/30, Nov 1939/41)
adj_oct <- Rochester.snowfall %>%
filter(format(Date, "%Y-%m") %in% c("1927-10", "1930-10")) %>%
filter(!is.na(snowfall)) %>%
pull(snowfall)

adjacent_nov <- Rochester.snowfall %>%
filter(format(Date, "%Y-%m") %in% c("1939-11", "1941-11")) %>%
filter(!is.na(snowfall)) %>%
pull(snowfall)

#Set seed to reproduce; sample with replacement from the adjacent yr numeric vectors
set.seed(42)
Rochester.snowfall$snowfall[oct29_idx] <- sample(
  adj_oct,
  size = length(oct29_idx),
  replace = TRUE
)

Rochester.snowfall$snowfall[nov40_idx] <- sample(
  adjacent_nov,
  size = length(nov40_idx),
  replace = TRUE
)

#Check for NA again; should be greatly reduced
(snow.na.1 <- Rochester.snowfall[is.na(Rochester.snowfall$snowfall), ])
```

```
            Date snowfall Month Year Day
55    1926-02-25       NA     2 1926  25
2389  1932-07-17       NA     7 1932  17
7987  1947-11-14       NA    11 1947  14
25750 1996-07-02       NA     7 1996   2
25753 1996-07-05       NA     7 1996   5
25754 1996-07-06       NA     7 1996   6
25779 1996-07-31       NA     7 1996  31
```

Now the missing values from the month of July in 1932 and 1996 will be imputed as 0.

```r
#Identify NA rows from July, 1996
jul96_idx <- which(
  Rochester.snowfall$Month == 7 &
  Rochester.snowfall$Year == 1996 &
  is.na(Rochester.snowfall$snowfall)
)

jul32_idx <- which(
  Rochester.snowfall$Month == 7 &
  Rochester.snowfall$Year == 1932 &
  is.na(Rochester.snowfall$snowfall)
)

#Replace
Rochester.snowfall$snowfall[jul96_idx] <- 0
Rochester.snowfall$snowfall[jul32_idx] <- 0

#Another check of remaining NA rows
(snow.na.1 <- Rochester.snowfall[is.na(Rochester.snowfall$snowfall), ])
```

```
           Date snowfall Month Year Day
55   1926-02-25       NA     2 1926  25
7987 1947-11-14       NA    11 1947  14
```

The remaining two missing data points will be filled according to LOCF (last-observed-carry-forward) method.

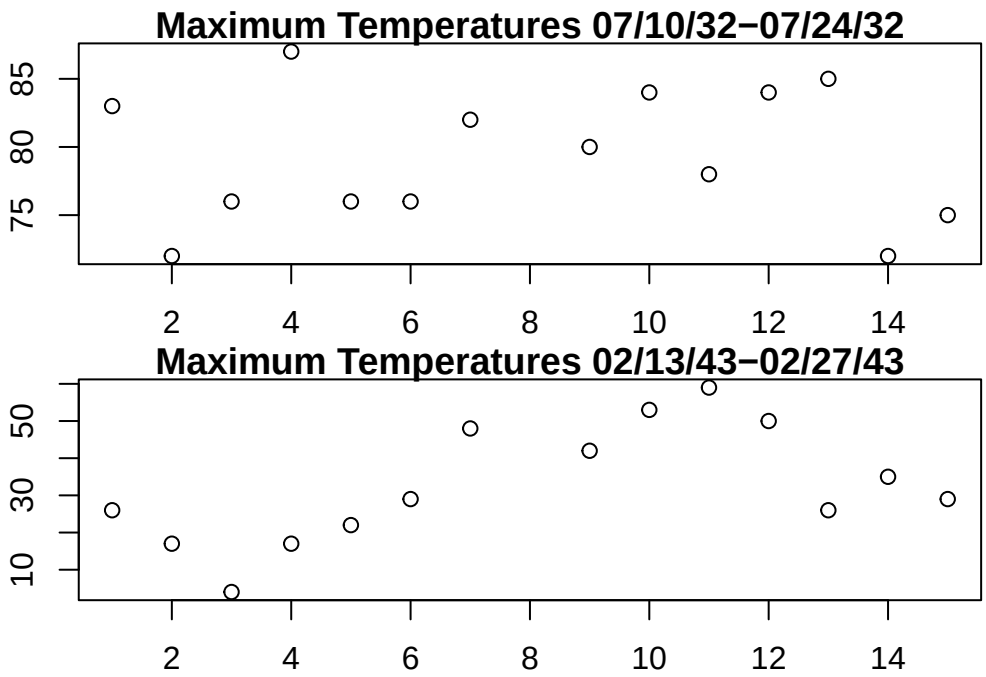
```r
feb25_26_idx <- which(
  Rochester.snowfall$Month == 2 &
  Rochester.snowfall$Year == 1926 &
  Rochester.snowfall$Day == 25 &
  is.na(Rochester.snowfall$snowfall)
)

feb24_26_idx <- which(
  Rochester.snowfall$Month == 2 &
  Rochester.snowfall$Year == 1926 &
  Rochester.snowfall$Day == 24)

nov14_47_idx <- which(
  Rochester.snowfall$Month == 11 &
```

```
  Rochester.snowfall$Year == 1947 &
  Rochester.snowfall$Day == 14 &
  is.na(Rochester.snowfall$snowfall)
)

nov13_47_idx <- which(
  Rochester.snowfall$Month == 11 &
  Rochester.snowfall$Year == 1947 &
  Rochester.snowfall$Day == 13
)

Rochester.snowfall$snowfall[feb25_26_idx] <- Rochester.snowfall$snowfall[feb24_26_idx]
Rochester.snowfall$snowfall[nov14_47_idx] <- Rochester.snowfall$snowfall[nov13_47_idx]

#Hopefully, final check
(snow.na.2 <- Rochester.snowfall[is.na(Rochester.snowfall$snowfall), ])
```

```
[1] Date     snowfall Month    Year     Day
<0 rows> (or 0-length row.names)
```

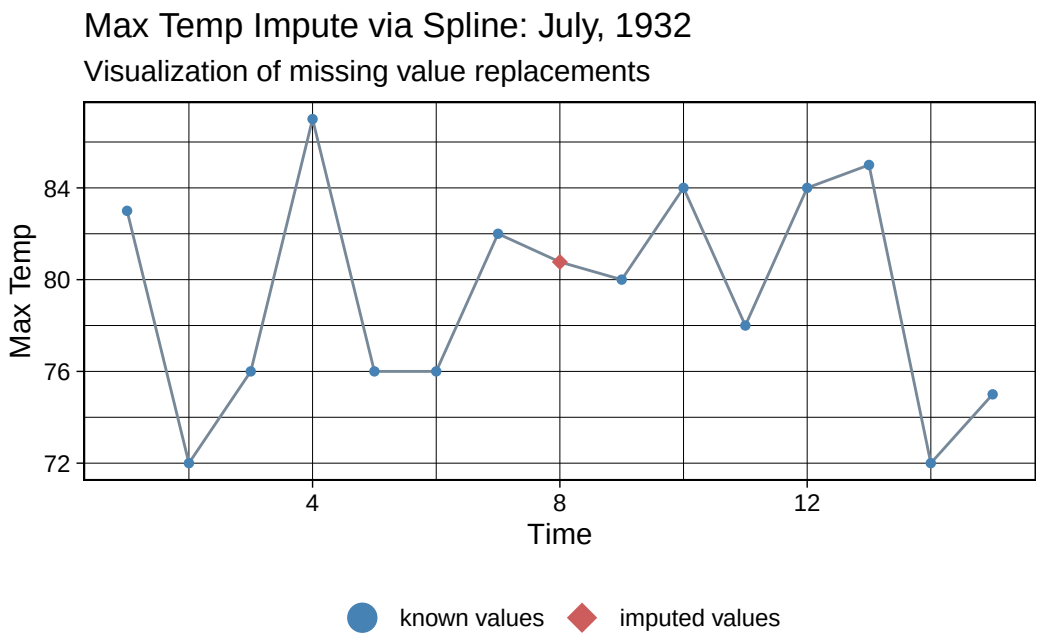
The snowfall data has been successfully cleaned. Now we will look to address the missing maximum temperature entries. In order to get a sense of what imputation method will work best, we need to zoom in on the days near July 17, 1932, and February 20, 1943.

```
par(mfrow = c(2,1), mar = c(2,3,1,1))
missing_temp_32 <- Rochester.Tmax[Rochester.Tmax$Date >= as.Date("1932-07-10") &
                                  Rochester.Tmax$Date <= as.Date("1932-07-24"), ]
missing_temp_43 <- Rochester.Tmax[Rochester.Tmax$Date >= as.Date("1943-02-13") &
                                  Rochester.Tmax$Date <= as.Date("1943-02-27"), ]
plot(missing_temp_32$tmax, main="Maximum Temperatures 07/10/32-07/24/32",
     ylab = "Max Temp")
plot(missing_temp_43$tmax, main="Maximum Temperatures 02/13/43-02/27/43",
     ylab = "Max Temp")
```

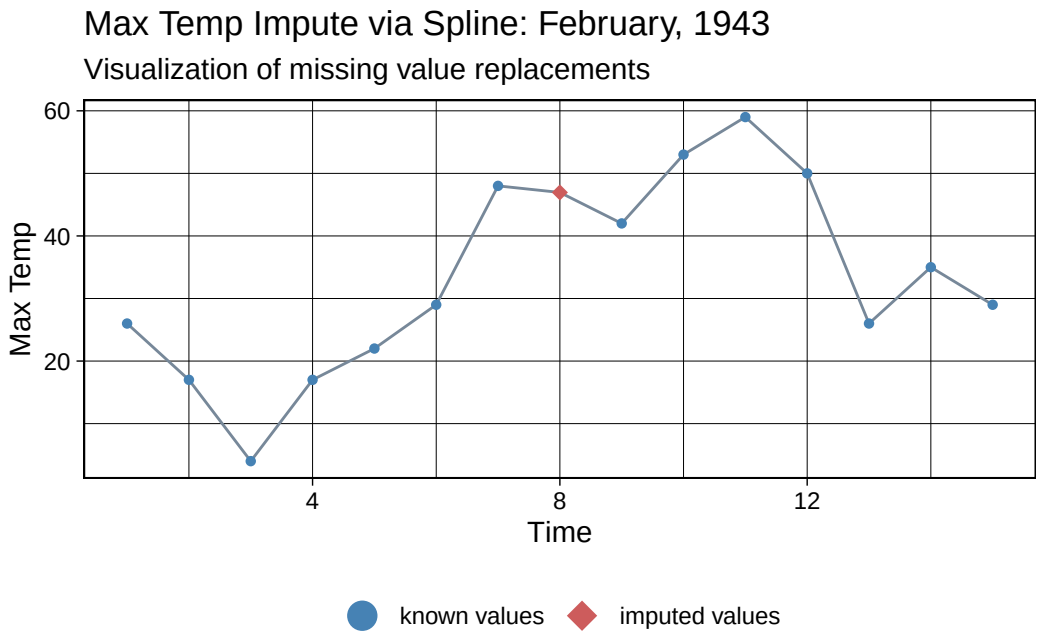


The best available option is to use a spline to interpolate and impute. This is automated using the `imputeTS` package. We will impute and look locally at the values to make sure everything is reasonably done before imputing in the actual series.

```
par(mfrow=c(2,1), mar = c(2,4,1,1))
#Spline interps
missing_temp_32$tmax_interp <- na_interpolation(missing_temp_32$tmax, option = "spline")
missing_temp_43$tmax_interp <- na_interpolation(missing_temp_43$tmax, option = "spline")


ggplot_na_imputations(x_with_imputations = missing_temp_32$tmax_interp,
                      x_with_na = missing_temp_32$tmax,
                      title = "Max Temp Impute via Spline: July, 1932",
                  xlab = "Time",
                  ylab = "Max Temp")
```




```
ggplot_na_imputations(x_with_imputations = missing_temp_43$tmax_interp,
                      x_with_na = missing_temp_43$tmax,
                      title = "Max Temp Impute via Spline: February, 1943",
                  xlab = "Time",
                  ylab = "Max Temp")
```



These are reasonable visually, though this is a nebulous assessment at best. We will impute using this method.

```
#Replace NA temp values via spline interpolation
Rochester.Tmax$tmax <- na_interpolation(Rochester.Tmax$tmax, option = "spline")
(Tmax.na1 <- Rochester.Tmax[is.na(Rochester.Tmax$tmax),])
```

```
[1] Date  tmax  Month Year  Day
<0 rows> (or 0-length row.names)
```

Now we will compile our time series to contain the following information (noting we do not have January 1st, 1926):

- Average monthly maximum temperature
- Total annual snowfall deviation: compare total annual snowfall for each year to the yearly average across 1991-2020. In particular, take $S_i - R_S$ where T_i is the annual total snowfall for year i and R_S is the reference snowfall. This quantity will be non-negative when at least as much snow fell in year i compared to the reference amount, and negative otherwise.
- Annual maximum temperature deviation: compare the average maximum temperature over a given year to the average maximum temperature across 1991-2020. In particular, take $T_i - R_T$ where T_i is the average maximum temperature for year i and R_T is the

reference temperature. This quantity will be nonnegative when the average maximum temperature in a given year is lower than the reference.

```
#Construct monthly snowfall totals
monthly_snowfall <- Rochester.snowfall %>%
  group_by(Year, Month) %>%
  summarize(
    tot_snow = sum(snowfall, na.rm = TRUE)
  )
```

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

```
#Construct average monthly max temp
monthly_tmax <- Rochester.Tmax %>%
  group_by(Year, Month) %>%
  summarize(
    mean_tmax = mean(tmax, na.rm = TRUE)
  )
```

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

```
#Calculate yearly snowfall totals from 1991-2020
monthly_snow_9120 <- Rochester.snowfall[Rochester.snowfall$Year >= 1991 &
                                Rochester.snowfall$Year <= 2020, ]
annual_snow_9120 <- monthly_snow_9120 %>%
  group_by(Year) %>%
  summarize(
    tot_snow = sum(snowfall)
  )

#Reference annual snowfall amount: average from 1991-2020
annual_snow_reference <- mean(annual_snow_9120$tot_snow)

#Reference temp: average maximum temp over 1991-2020
ref_temp <- Rochester.Tmax %>%
  filter(Year >= 1991, Year <= 2020) %>%
  summarize(ref_temp = mean(tmax, na.rm = TRUE)) %>%
  pull(ref_temp)
```

```r
#Construct annual snowfall totals
annual_snow <- monthly_snowfall %>%
  group_by(Year) %>%
  summarize(
    tot_snow = sum(tot_snow)
  )

#Construct average annual max temperatures
annual_tmax <- Rochester.Tmax %>%
  group_by(Year) %>%
  summarize(
    tmax_avg = mean(tmax)
  )

#Construct annual snowfall deviation: annual snowfall total - annual_snow_reference
annual_snow_deviations <- annual_snow
annual_snow_deviations$tot_snow <- annual_snow_deviations$tot_snow - annual_snow_reference

#Construct annual temperature deviations
annual_tmax_deviations <- annual_tmax
annual_tmax_deviations$tmax_avg <- annual_tmax_deviations$tmax_avg - ref_temp
```

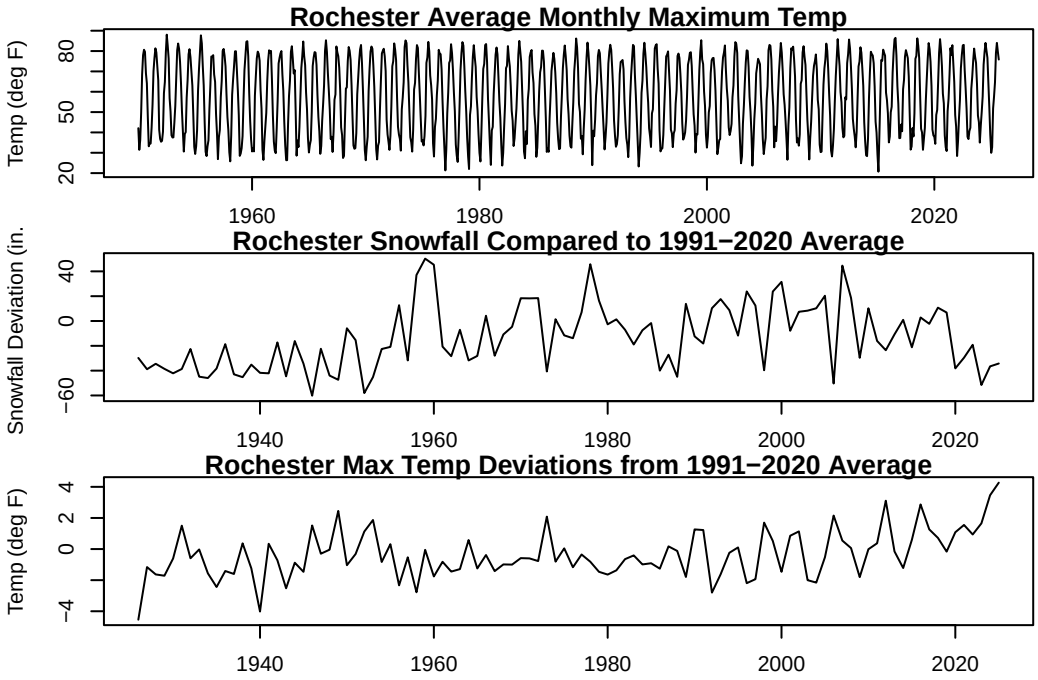
Before making proper time series objects, we have to make sure the data is ordered properly by year and month.

```r
monthly_tmax <- monthly_tmax[order(monthly_tmax$Year, monthly_tmax$Month), ]
annual_snow_deviations <- annual_snow_deviations[order(annual_snow_deviations$Year), ]
```

Finally, our time series objects can be created. The average monthly maximum temperature series will be limited to January, 1950 and beyond.

```r
par(mfrow=c(3,1), mar = c(2,4,1,1))

#Restrict max temp average by month to Jan 1950 - Sep 2025
max_temps <- ts(monthly_tmax$mean_tmax, start = c(1926, 1), frequency = 12)
max_temps <- window(max_temps, start = c(1950, 1), end = c(2025, 9))

snowfall_devs <- ts(annual_snow_deviations$tot_snow, start = 1926, frequency = 1)
max_tempdevs <- ts(annual_tmax_deviations$tmax_avg, start = 1926, frequency = 1)

ts.plot(max_temps, ylab = "Temp (deg F)", main = "Rochester Average Monthly Maximum Temp")
```

```
ts.plot(window(snowfall_devs, start = 1926), ylab = "Snowfall Deviation (in.)", main = "Roche
ts.plot(window(max_tempdevs, start = 1926), ylab = "Temp (deg F)", main = "Rochester Max Temp
```



By construction, the latter two time series are more similar in nature to each other than to the first. The seasonal nature of the maximum temperature averaged over each month provides us with a different sort of forecasting opportunity, which we explore in the next section.

3. Analysis of Average Monthly Maximum Temperature

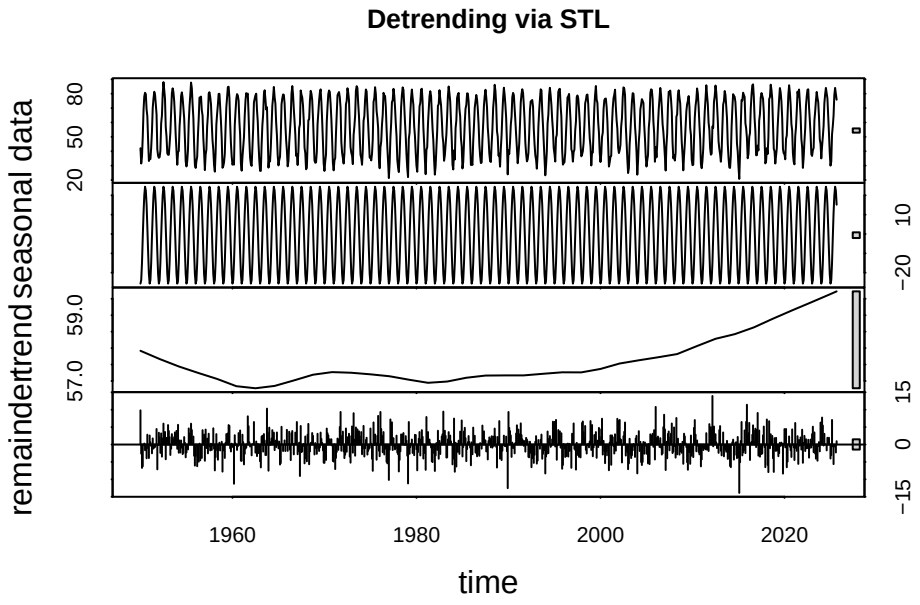
Our main goal in modeling the average monthly maximum temperature is to be able to forecast future movement of the series due to trend and seasonal factors.

3.1 Preliminary Analysis

The time series `max_temps` exhibits regular seasonal behavior with stable amplitude and limited discernible trend. We will examine these claims more closely, beginning with two different methods of trend estimation.

One nonparametric method we may use to estimate trend to avoid assuming any functional form is Season-Trend LOESS, a form of detrending via local regression.

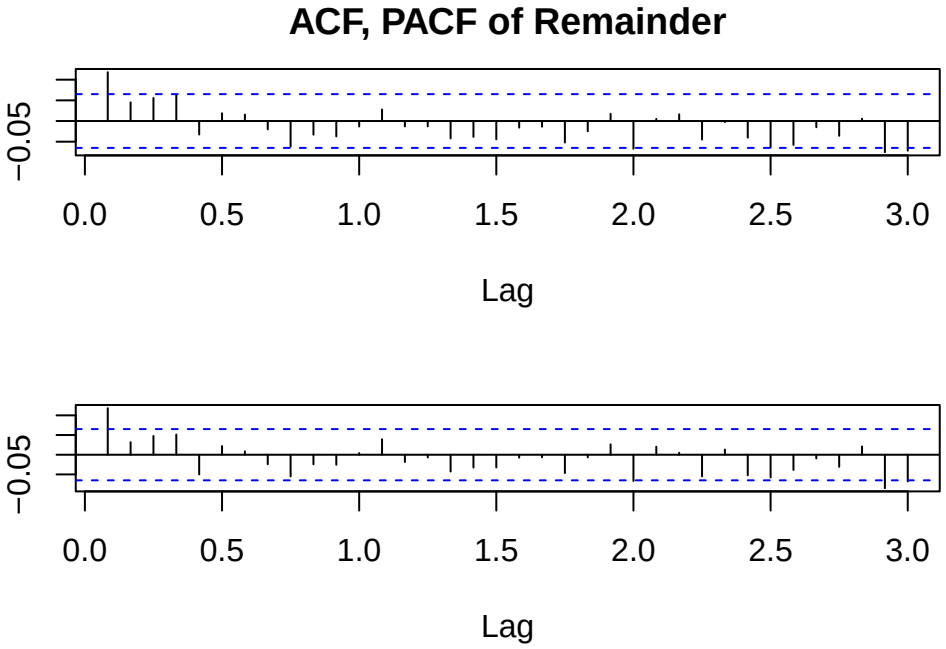
```
stl_detrending <- stl(max_temps, s.window = "periodic", t.window = 241)
plot(stl_detrending, main="Detrending via STL")
```



We use a window of length 241 months to extract the trend over approximately twenty year rolling periods, as this was the minimum window size at which predominantly low-frequency movement was captured. A trend is present which accounts for a roughly 2 degree Fahrenheit swing between a minimum of approximately 57 degrees in the early 1960's and a maximum of 59 degrees in 2024. This is consistent with long-term regional climate change.

To confirm that the above process leaves a remainder which resembles white noise, i.e. temporal dependence has been removed, we can inspect the ACF and PACF.

```
par(mfrow=c(2,1), mar = c(4,3,2.5,2))
acf(stl_detrending$time.series[,"remainder"], main = "ACF, PACF of Remainder", lag.max = 36)
pacf(stl_detrending$time.series[,"remainder"], main="", lag.max = 36)
```



The sample ACF does not have consistent significant spikes at positive lags within a three year window, suggesting the remainder does resemble a white noise process. Our next step is to create a baseline model by using polynomial regression to model the trend component and monthly indicator variables to model the seasonal component.

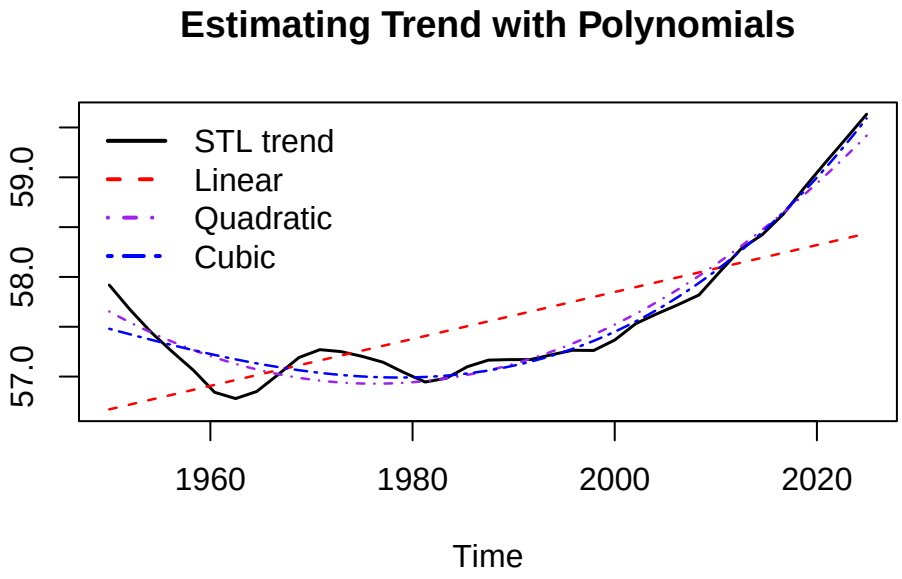
3.2 Baseline Model

To train and then evaluate our initial model, we will create a training set from January, 1950 until December, 2024, and a forecast set from January, 2025 until September, 2025.

```
#Create train/forecast sets from actual data and output from stl().
motemps.train <- window(max_temps, start = c(1950, 1), end = c(2024,12))
motemps.fore <- window(max_temps, start = c(2025,1), end = c(2025, 9))

motemps_trend.train <- window(stl_detrending$time.series[,"trend"],
                              start = c(1950, 1), end = c(2024,12))
motemps_trend.fore <- window(stl_detrending$time.series[,"trend"],
                              start = c(2025,1), end = c(2025, 9))

motemps_seas.train <- window(stl_detrending$time.series[,"seasonal"],
                              start = c(1950, 1), end = c(2024,12))
motemps_seas.fore <- window(stl_detrending$time.series[,"seasonal"],
```

```
                                start = c(2025,1), end = c(2025, 9))

n_ttrain <- length(motemps.train)
n_tfore <- length(motemps.fore)

#Training time data
ttrain_trend_df <- tibble(
  t_fit = 1:n_ttrain,
  t_sqfit = t_fit^2,
  t_cubefit = t_fit^3
)

#Forecasting time data
tfore_trend_df <- tibble(
  t_fit = (n_ttrain+1):(n_ttrain+n_tfore),
  t_sqfit = t_fit^2,
  t_cubefit = t_fit^3
)

mlr.line <- lm(motemps_trend.train ~ t_fit, data = ttrain_trend_df)
mlr.quadr <- lm(motemps_trend.train ~ t_fit + t_sqfit, data = ttrain_trend_df)
mlr.cubic <- lm(motemps_trend.train ~ t_fit + t_sqfit + t_cubefit, data = ttrain_trend_df)
```

```
trend_pred_quad <- cbind(motemps_trend.train, mlr.line$fitted, mlr.quadr$fitted, mlr.cubic$fi
ts.plot(trend_pred_quad, col=c("black", "red", "purple", "blue"), lty = c(1,2,4,6), main = "

legend("topleft",
       legend = c("STL trend", "Linear", "Quadratic", "Cubic"),
       col = c("black", "red", "purple", "blue"),
       lty = c(1, 2, 4, 6),
       lwd = 2,
       bty = "n")
```



It is visually evident that the linear model is too simple to capture the growth pattern while the quadratic and cubic models have comparable performance. These statements are supported by the model summaries:

```
summary(mlr.line)
```

```
Call:
lm(formula = motemps_trend.train ~ t_fit, data = ttrain_trend_df)

Residuals:
    Min      1Q  Median      3Q     Max
-0.5378 -0.4032 -0.1183  0.2206  1.2473

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.667e+01  3.139e-02 1805.10   <2e-16 ***
t_fit       1.963e-03  6.037e-05   32.52   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4705 on 898 degrees of freedom
Multiple R-squared:  0.5408,    Adjusted R-squared:  0.5403
```

```
F-statistic:  1057 on 1 and 898 DF,  p-value: < 2.2e-16
```

```
summary(mlr.quadr)
```

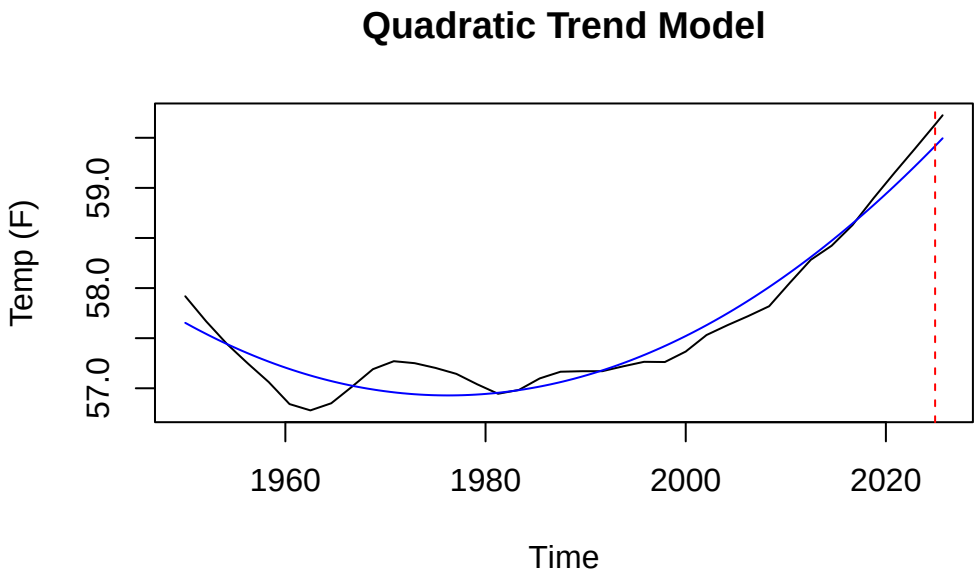
```

Call:
lm(formula = motemps_trend.train ~ t_fit + t_sqfit, data = ttrain_trend_df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.35003 -0.12015 -0.00895  0.10619  0.31150

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  5.766e+01  1.642e-02 3512.43   <2e-16 ***
t_fit       -4.613e-03  8.414e-05  -54.82   <2e-16 ***
t_sqfit      7.298e-06  9.043e-08   80.71   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1638 on 897 degrees of freedom
Multiple R-squared:  0.9444,    Adjusted R-squared:  0.9443
F-statistic:  7620 on 2 and 897 DF,  p-value: < 2.2e-16
```

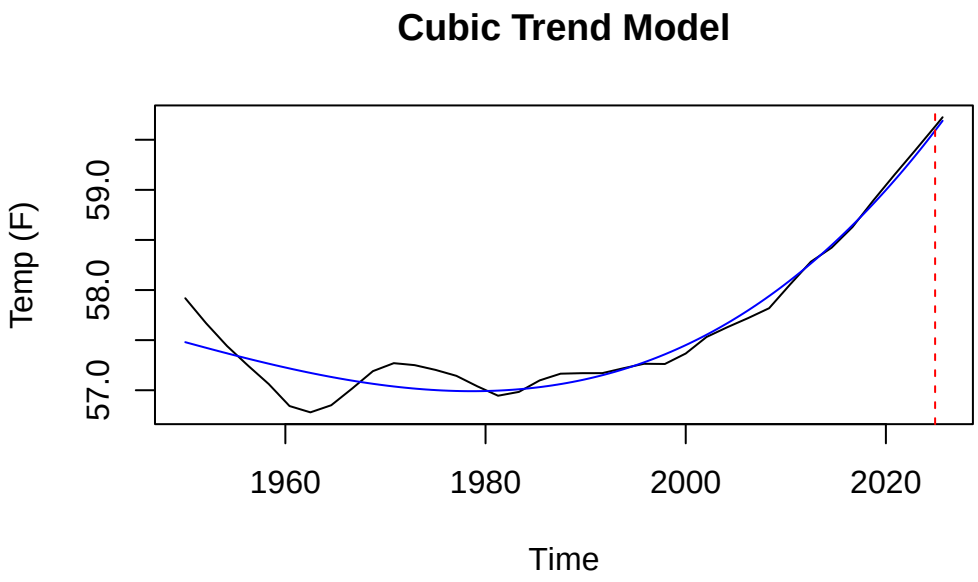
```
summary(mlr.cubic)
```

```

Call:
lm(formula = motemps_trend.train ~ t_fit + t_sqfit + t_cubefit,
    data = ttrain_trend_df)

Residuals:
     Min       1Q   Median       3Q      Max
-0.39380 -0.06223  0.00192  0.07010  0.43855

Coefficients:
              Estimate Std. Error  t value Pr(>|t|)
(Intercept)  5.748e+01  2.008e-02 2862.572   <2e-16 ***
t_fit       -2.281e-03  1.929e-04  -11.826   <2e-16 ***
t_sqfit      8.327e-07  4.974e-07    1.674   0.0944 .
t_cubefit    4.784e-09  3.629e-10   13.184   <2e-16 ***
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.15 on 896 degrees of freedom
Multiple R-squared:  0.9534,    Adjusted R-squared:  0.9533
F-statistic:  6117 on 3 and 896 DF,  p-value: < 2.2e-16
```

Retaining only the quadratic and cubic models for consideration, we wish to compare their performance on the out-of-sample data. The plots give the initial impression that the actual trend of the data grows too quickly for the quadratic model to keep pace, while the cubic model does a more reasonable job.





To confirm the superiority of the cubic model, we calculate standard forecast metrics.

```
# Forecast/Prediction errors: Observed - Predicted
efore.quad <- motemps_trend.fore - pfore.quad$fit
efore.cub <- motemps_trend.fore - pfore.cub$fit

# Forecast evaluation criteria
me.quad <- mean(efore.quad)                          # Mean Error
mpe.quad <- 100*(mean(efore.quad/motemps_trend.fore))          # Mean Percent Error
mse.quad <- sum(efore.quad**2)/n_tfore               # Mean Squared Error
mae.quad <- mean(abs(efore.quad))                    # Mean Absolute Error
mape.quad <- 100*(mean(abs((efore.quad)/motemps_trend.fore)))  # Mean Absolute Percent Error
fec.quad <- data.frame(quadratic = rbind(me.quad, mpe.quad,
                                         mse.quad, mae.quad, mape.quad),
                       row.names = c("me", "mpe", "mse", "mae", "mape"))

# Forecast evaluation criteria
me.cube <- mean(efore.cub)                           # Mean Error
mpe.cube <- 100*(mean(efore.cub/motemps_trend.fore))           # Mean Percent Error
mse.cube <- sum(efore.cub**2)/n_tfore                # Mean Squared Error
mae.cube <- mean(abs(efore.cub))                     # Mean Absolute Error
mape.cube <- 100*(mean(abs((efore.cub)/motemps_trend.fore)))  # Mean Absolute Percent Error
fec.cube <- data.frame(cubic = rbind(me.cube, mpe.cube,
```

```
                                        mse.cube, mae.cube, mape.cube),
                      row.names = c("me", "mpe", "mse", "mae", "mape"))

round(fec.quad, digits = 4)
```

```
     quadratic
me      0.2232
mpe     0.3740
mse     0.0499
mae     0.2232
mape    0.3740
```

```
round(fec.cube, digits = 4)
```

```
      cubic
me   0.0382
mpe  0.0641
mse  0.0015
mae  0.0382
mape 0.0641
```

As expected, the cubic trend model has significantly better forecast performance than the quadratic model. We may also consider information criterion when performing model selection, though our forecasting purposes place more weight on the above metrics.

```
k_2 <- 2
k_3 <- 3

AIC.quad <- AIC(mlr.quadr, k=k_2)
AIC.cubic <- AIC(mlr.cubic, k=k_3)

AIC_vals <- data.frame(AIC = rbind(AIC.quad,
                                   AIC.cubic),
                       row.names = c("Quadratic", "Cubic"))
AIC_vals
```

```
                AIC
Quadratic -697.4424
Cubic     -850.0111
```

This confirms that the increased complexity of the cubic model has merit. We will move forward with the long-term temperature trend modeled by the polynomial in t, measured in months since December, 1949:

$$T_{trend}(t) = 57.48 - 0.0023t + (8.327 \times 10^{-7})t^2 + (4.784 \times 10^{-9})t^3.$$