Evaluating Monthly Trends using CHCN-Daily

Marc Los Huertos

August 27, 2018

Contents

Abstract

This SOP will provide the tools to analyze the CDO climate data for long-term trends in the daily data. We will also create monthly averages and determine if some months have a stronger trend than others.

1 Introduction

1.1 Goals for This Document

This document provides EA students with the methods to analyze climate data based on monthly averages and evaluate if these data are reliable compared to the CHCN-Monthly and investigate sources of uncertainty.

1.2 To Begin

You should have R code that generates a plot of daily TMAX data for a site with a best fit line overlaid. If not, please go back to SOP85.

1.3 Generalized Steps

In this SOP you will ...

1. create new variables for date and month;
2. create a new dataframe with monthly averages;
3. model and estimate average trend for each month;
4. evaluate the validity of the models; and
5. interpret the trend data

1.4 Regression and Climate Change

One of the ourcomes of the linear regression is to estimate the best fit line

$$y = mx + b + \epsilon, \tag{1}$$

where ϵ is an estimate of the error. In addition, two other estimates are provided, one for the slope, m, and the y-intercept, b.

But these estimates are also hypotheses, where the null hypothesis is:

slope is zero Rejecting the null hypothesis would be support the alternative hypothesis, or the estimate of the slope.

y-intercept is zero Rejecting the null hypothesis would support the alternative hypothesis, the estimate of the y-intercept.

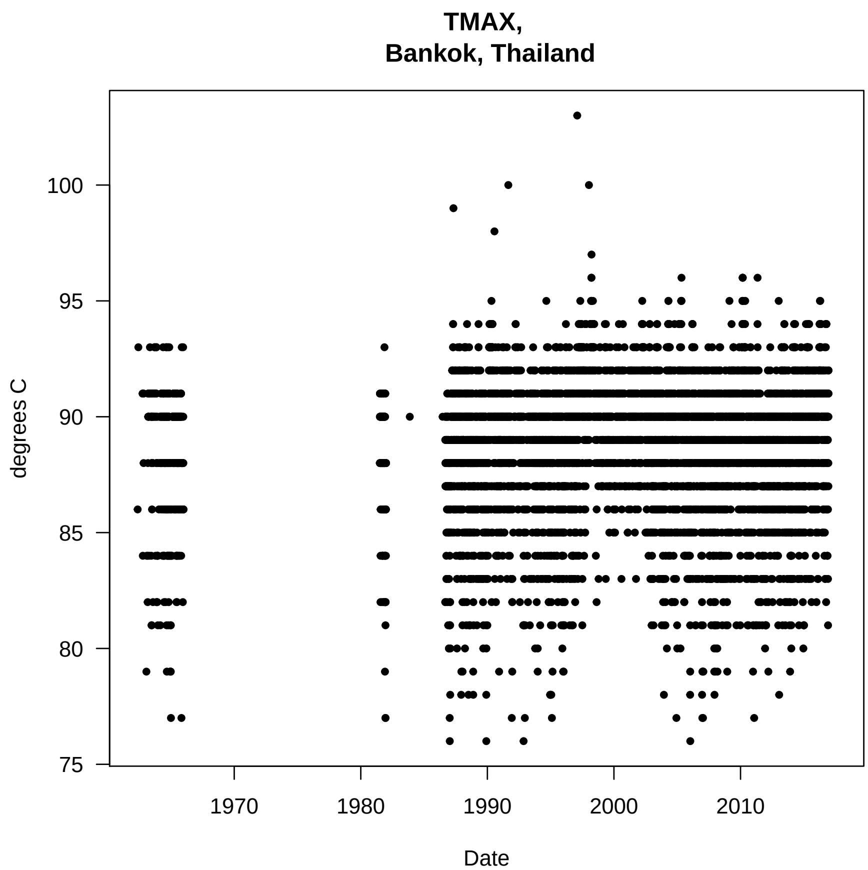
Okay, let's see if we can do this for our Bangkok data. Let's test if there is a significant change of daily maximum temperatures (TMAX) with time. Thus, in general terms, Maximum temperature is a function of time, or $TMAX = f(Time)$.

$$TMAX \sim \alpha + \beta * time + \epsilon \tag{2}$$

Translating this in R will take some additional tricks besides just getting the code figured out. First, we need to identify the predictor variable, 'NewDate', in the data frame which we created in SOP85.

Because these data are in a time series, they are serially correlated, meaning that the June sample will be more like the July sample than the August sample. In addition, the June 2010 sample will be similar to the June 2009 sample. These correlation violate the assumption of independence, but for now, we will ignore this violation and just create a linear model in bliss.

Figure 1: Maximum daily temperatures for Bangkok, Thailand.



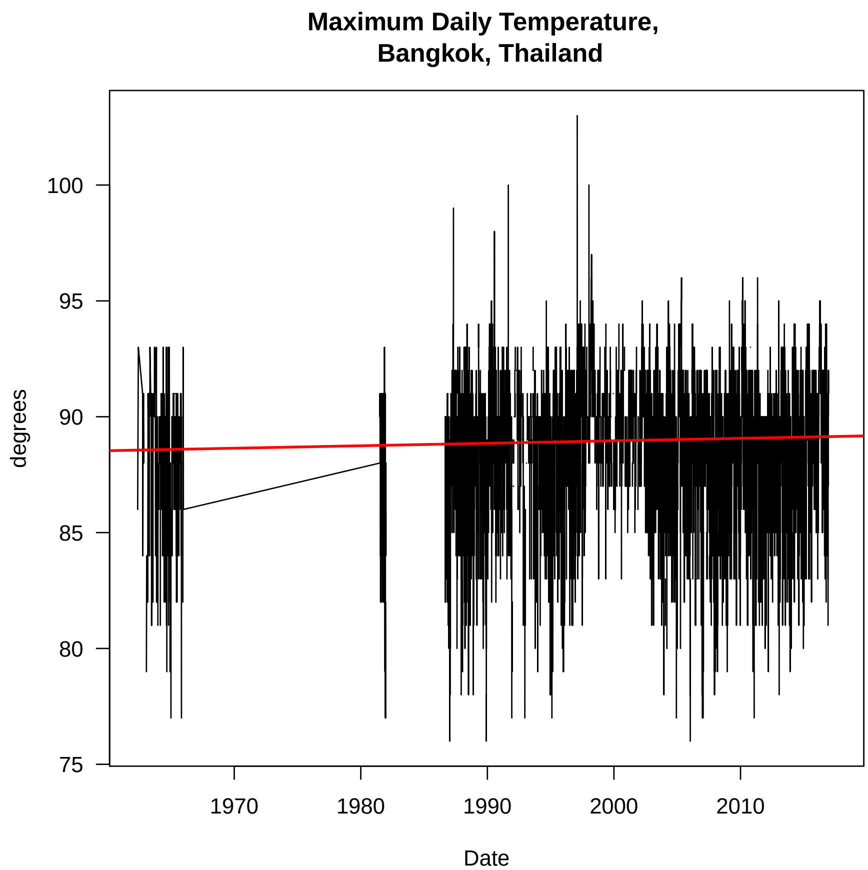
For the response variable, we will use the daily maximum temeratures, TMAX. Remember there are some missing data, it will be interesting to note how R deals with that.

First, let's create a plot of data using `plot()`, whose format is `plot(x, y)` or `plot(y   x)`. We will use the later for now,

We use the `lm()` function that arrange the results in-line with a regression model. This syntax is straight forward,

```
lm(TMAX ~ NewDate, data=climate_data)

##
## Call:
## lm(formula = TMAX ~ NewDate, data = climate_data)
##
```

Figure 2: Maximum Daily Temperatures in Bangkok, Thailand.




```
## Coefficients:
## (Intercept)      NewDate
##   8.864e+01    2.922e-05
```

From this model, we learn that the change in $TMAX$ is 0 degrees $year^{-1}$. Figure 1.4 shows a trend of increasing maximum temperatures.

Now determine test the null hypotheses and use the `summary()` function to display many of the important regression results.

```
summary(lm(TMAX ~ NewDate, data=climate_data))

##
## Call:
## lm(formula = TMAX ~ NewDate, data = climate_data)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -13.026  -1.080   0.859   1.859  14.069
##
## Coefficients:
##              Estimate Std. Error  t value Pr(>|t|)
## (Intercept) 8.864e+01  8.096e-02 1094.875  < 2e-16 ***
## NewDate     2.922e-05  6.841e-06    4.271 1.97e-05 ***
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.713 on 8221 degrees of freedom
##   (5481 observations deleted due to missingness)
## Multiple R-squared:  0.002214,Adjusted R-squared:  0.002093
## F-statistic: 18.24 on 1 and 8221 DF,  p-value: 1.967e-05
```

Based on the results, we reject the null hypotheses, i.e. the events that this might occur by chance is small: 2x10^{-16} for the slope is zero and p ¡ 2x10^{-16} for the y-intercept is zero.

In addition, we have some information on the residuals, and R^2 estimates, which are important to interpret the model.

For now, we can appreciate the the temperature is changing, i.e. increasing, with a slope of 2.7x10^{-5} degrees C per year.

1.4.1 Creating Monthly Averages of Daily Maximum Temperatures

One of the first things to note is how messy the data look and there are lots of sources of variation. For example, we expect months to respond differently to the climate change. To assess this, we will now analyze the data for monthly means of the maximum temperatures.

1.4.2 Creating Monthly Means

To create monthly means, we need to disagragate the NewDate variable into a month and year variables.

First we can use the `as.Date()` function to extract a portion of the date, where %m is for month and %Y is for a four digit year. Then, we create new variables in our dataframe, one for month and one for year.

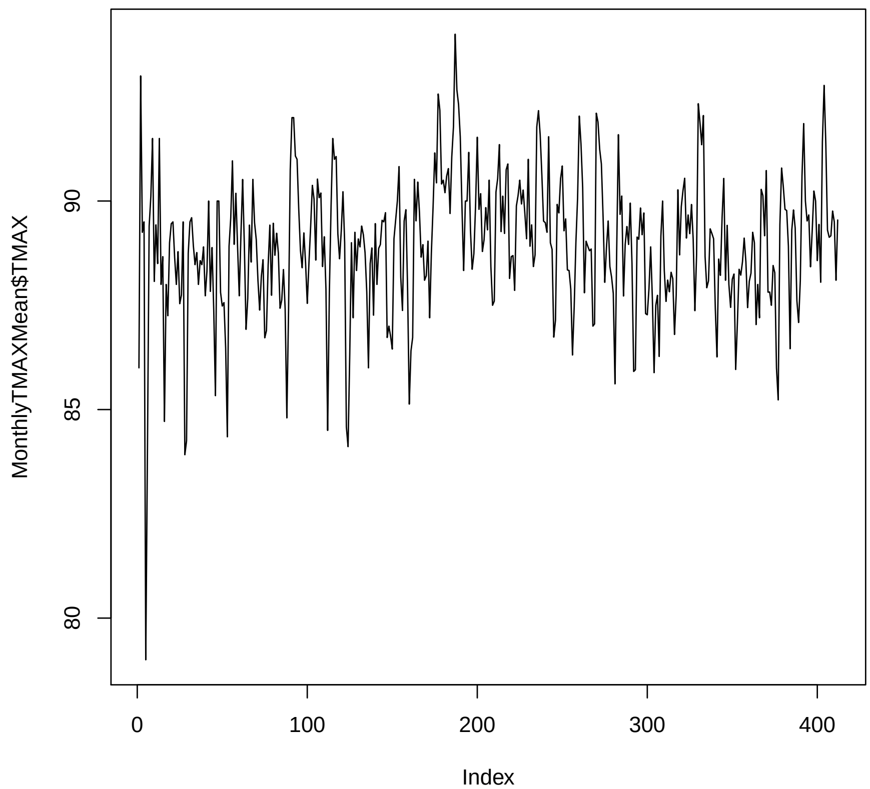
```
climate_data$Month = format(as.Date(climate_data$NewDate), format = "%m")
climate_data$Year = format(climate_data$NewDate, format="%Y")
```

After creating the month and year as separate variables, we can use them to caculate the mean using the `aggregate()` function. In the code below, we can also calculate the standard deviation too, although I haven't used this measure in this document, several students have asked for this for their analysis.

```
MonthlyTMAXMean = aggregate(TMAX ~ Month + Year, climate_data, mean)

MonthlyTMAXMean$YEAR = as.numeric(MonthlyTMAXMean$Year)
MonthlyTMAXMean$MONTH = as.numeric(MonthlyTMAXMean$Month)
```

```
str(MonthlyTMAXMean)

## 'data.frame': 412 obs. of  5 variables:
##  $ Month: chr  "05" "06" "10" "11" ...
##  $ Year : chr  "1962" "1962" "1962" "1962" ...
##  $ TMAX : num  86 93 89.2 89.5 79 ...
##  $ YEAR : num  1962 1962 1962 1962 1963 ...
##  $ MONTH: num  5 6 10 11 1 2 3 4 5 6 ...
```

```
plot(MonthlyTMAXMean$TMAX, ty='l')
```

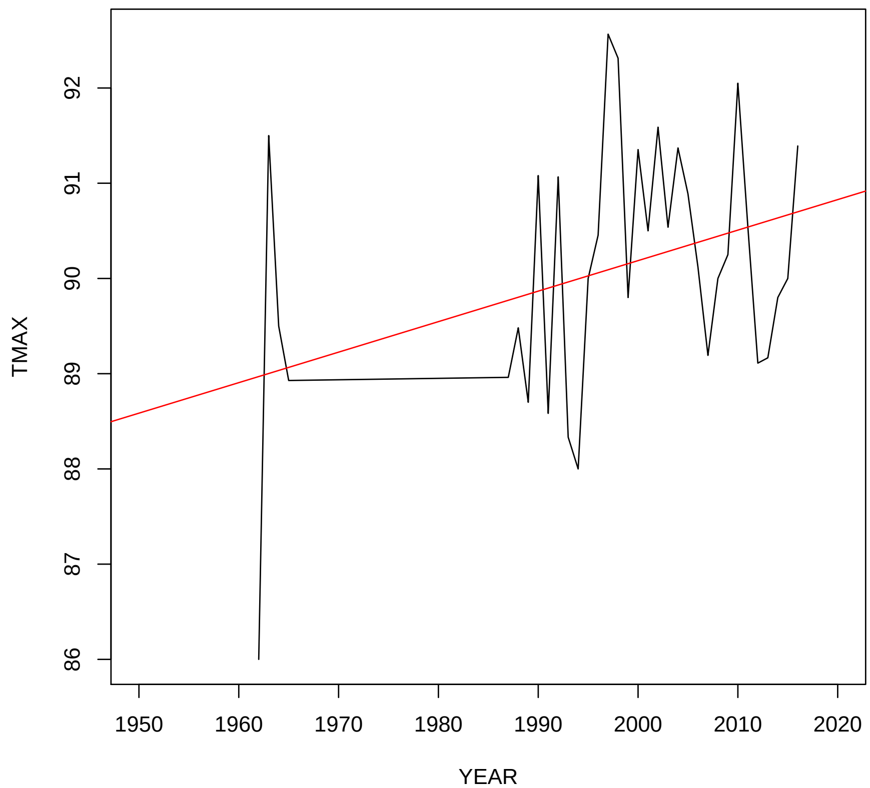


1.5 Selecting for 1 Month – May

Perhaps, we can get a better handle on this stuff if we analyze for just one month at a time – certainly easier to visualize!

```
#plot(MonthlyTMAXMean£TMAX[MonthlyTMAXMean£Month=="05"], ty='l')
plot(TMAX~YEAR, data=MonthlyTMAXMean[MonthlyTMAXMean$Month=="05",],
     ty='l', xlim=c(1950, 2020))
May.lm <- lm(TMAX~YEAR, data=MonthlyTMAXMean[MonthlyTMAXMean$Month=="05",])
summary(May.lm)

##
## Call:
## lm(formula = TMAX ~ YEAR, data = MonthlyTMAXMean[MonthlyTMAXMean$Month ==
##     "05", ])
##
```

```
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.96991 -0.82905 -0.08223  0.96417  2.49806
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 26.11965   30.36486   0.860   0.3961
## YEAR         0.03203    0.01520   2.107   0.0431 *
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.303 on 32 degrees of freedom
## Multiple R-squared:  0.1218,Adjusted R-squared:  0.09437
## F-statistic: 4.439 on 1 and 32 DF,  p-value: 0.04306

abline(coef(May.lm), col="red")
```



Now, the change is 0.032 degress C/year or 3.203 degress C/100 years with a probability of 0.0431. Although we can't reject the null hypothesis, we find the method to be fairly straightforward!

1.6 Testing all the Months

I think you should evaluate every month and see what happens. You might also consider looking at the TMIN as well. Could be important![1]

Below, I have create code to evaluate all of the months at once, but you may prefer to go through each month manually and change the number from 5 to other months of the year.

[1] What about multiple hypotheses in one dataset!

```
# First I create a vector of months
Months = c("January", "February", "March", "April", "May", "June",
  "July", "August", "September", "October", "November", "December")

# Create a panel so I can see all the figures at once.
par(mfrow=c(4,3), mar=c(5, 4, 3, 2) + 0.1)
TMAXresult <- NA
for (i in 1:12){
#plot(MonthlyTMAXMean£TMAX[MonthlyTMAXMean£Month==i], ty='l')
plot(TMAX~YEAR, data=MonthlyTMAXMean[MonthlyTMAXMean$MONTH==i,], ty='l', las=1, xlim=c(1940,
Month.lm <- lm(TMAX~YEAR, data=MonthlyTMAXMean[MonthlyTMAXMean$MONTH==i,])
summary(Month.lm)

abline(coef(Month.lm), col="red")

TMAXresult <- rbind(TMAXresult, cbind(Months[i], round(coef(Month.lm)[2], 4), round(summary(
}
```

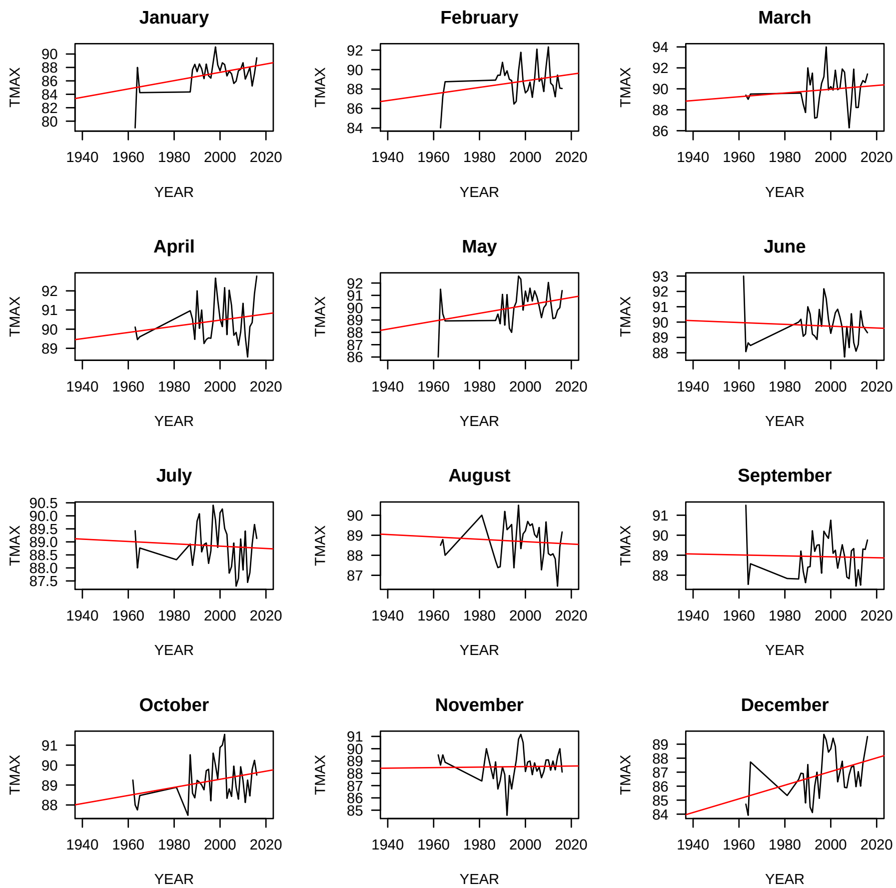
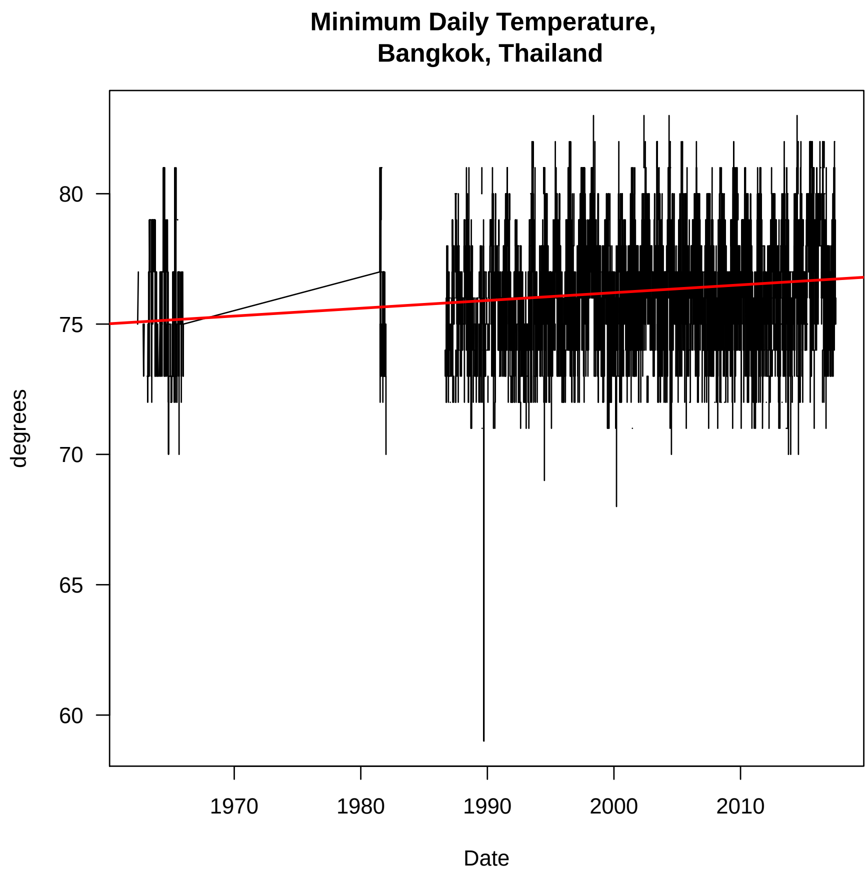


Figure 3: Minimum Daily Temperatures in Bangkok, Thailand.



1.7 Next Steps

1.7.1 Analyzing Minimum Daily Temperatures

Alternatively, it might be important to evaluate changes to the daily mininum temperatures. Following the same steps we used before but using the TMIN instead of TMAX, let's analyze the monthly average of daily minimum temperatures by following these steps:

1. First, let's plot the daily minimum temperatures, and as with the daily maximum temperatures, find tons of scatter (Table 1).

   There appears to be a trend, but it's clouded with lots of variation.

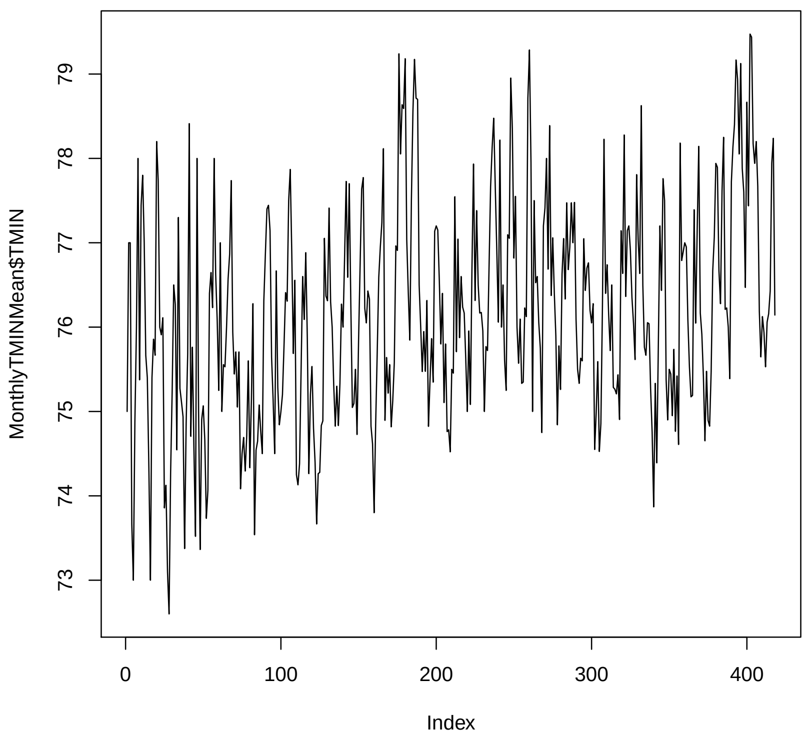
2. We create a monthly TMIN mean for each month.

```r
MonthlyTMINMean = aggregate(TMIN ~ Month + Year, climate_data, mean)

MonthlyTMINMean$YEAR = as.numeric(MonthlyTMINMean$Year)

# Fixing the Format of Month and Year as numeric
MonthlyTMINMean$YEAR = as.numeric(MonthlyTMINMean$Year)
MonthlyTMINMean$MONTH = as.numeric(MonthlyTMINMean$Month)
head(MonthlyTMINMean)
```

```
##   Month Year     TMIN YEAR MONTH
## 1    05 1962 75.00000 1962     5
## 2    06 1962 77.00000 1962     6
## 3    10 1962 77.00000 1962    10
## 4    11 1962 73.66667 1962    11
## 5    02 1963 73.00000 1963     2
## 6    03 1963 74.87500 1963     3
```

3. Create a plot of the monthly average of the daily minimum temperatures.

```r
plot(MonthlyTMINMean$TMIN, ty='l')
```



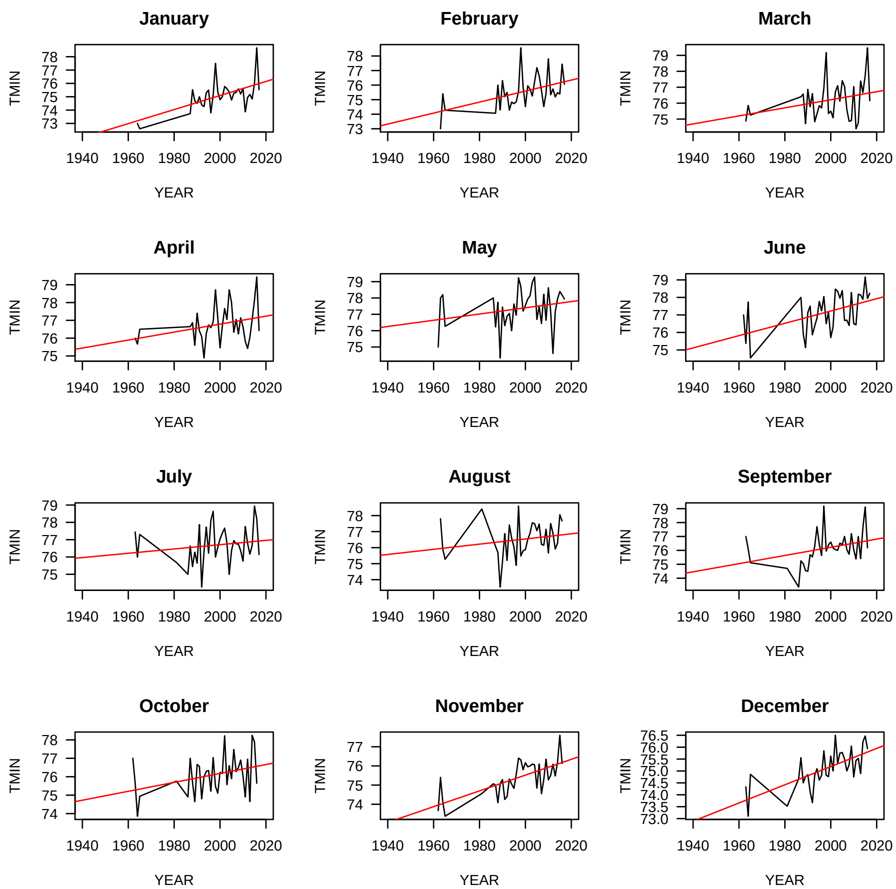
There is still lots of scatter and now we can subset our data by month.

4. Using the example above, we'll plot all 12 months at once to look for patterns (Table 4).

5. The change in minimum temperatures seems to be even more compelling than the maximum temperatures. To compare, look at the Table **??** to appreciate estimated slopes and their associated null hypothesis probabilities.

```
library(xtable)
Results <- data.frame(Month = TMINresult[c(2:13),1], TMINSlope = TMINresult[c(2:13),2],
Results$starTMIN = "NS"
Results$starTMIN[Results$TMIN_P <= .05] = "*"
Results$starTMIN[Results$TMIN_P < 0.01] = "**"
Results$starTMIN[Results$TMIN_P < 0.001] = "***"
Results$starTMAX = "NS"
Results$starTMAX[Results$TMAX_P < 0.05] = "*"
Results$starTMAX[Results$TMAX_P < 0.01] = "**"
```

Figure 4: Twelve Months of Monthly Average Daily Minimum Temperatures, Bangkok, Thailand




```
Results$starTMAX[Results$TMAX_P < 0.001] = "***"
Results$TMINslope=paste(Results$TMINSlope, Results$starTMIN)
Results$TMAXslope=paste(Results$TMAXSlope, Results$starTMAX)
colnames(Results) <- c("Month", "2", "3", "R^2", "5", "6",
                       "R^2", "8", "9", "Slope TMIN", "Slope TMAX")
print(xtable(Results[,c(1, 10, 4, 11, 7)]))
```

Based on the results above, the slopes are greatest during the dry season (starting in May) for the maximum temperatures – but the minimum temperatures show the largest slopes (change) and peaking between January and April.

In addition, the r^2 values signify the amount of the variance explained

	Month	Slope TMIN	R^2	Slope TMAX	R^2.1
1	January	0.0526 ***	0.363	0.0618 *	0.177
2	February	0.0378 **	0.223	0.0337 NS	0.079
3	March	0.0253 NS	0.085	0.018 NS	0.022
4	April	0.0224 NS	0.098	0.0161 NS	0.041
5	May	0.0191 NS	0.058	0.032 *	0.122
6	June	0.0351 **	0.236	-0.006 NS	0.006
7	July	0.0124 NS	0.028	-0.0045 NS	0.005
8	August	0.0161 NS	0.041	-0.0059 NS	0.008
9	September	0.0294 *	0.12	-0.0023 NS	0.001
10	October	0.0242 *	0.121	0.0203 NS	0.097
11	November	0.0412 ***	0.475	0.0022 NS	0.001
12	December	0.0382 ***	0.456	0.0488 *	0.183

by the predictor – in the case of TMIN, most of the values are over 20% meaning that over 20% of the variance is explained by time. While in March and April over time explains 50% of the variance.

This is very high for uncontrolled experiments. However, we should be cognizant that in many cases, especially for the maximum temperatures, it is less than 10%. This means the the variation in temperature are not predicted by time – thus, as a modeler, I would work very hard to capture other sources to better understand what is going on in Thailand.

Finally, we should also be very concerned about testing 2 dozen hypotheses with our little R code. It's easy to do, but based on change alone, with a critical value of 0.05, we should expect 1 in 20 tests to give us a Type I error, a signal when one doesn't exists. Since we did 12 tests, we should expect a good chance that one or more of our tests will reject the null hypothesis incorrectly. Yikes! Please keep this in mind and be careful to avoid this potential problem.

As we might expect, the a small amount of the variance is explained by the "Month." Many things predict temerpature, that year is one, is quite problematic.

6. What we have not determined is the cause. So, be careful when you describe the results, cause and effect cannot be analyzed using this method.

1.7.2 Precipitation: Departure from Mean

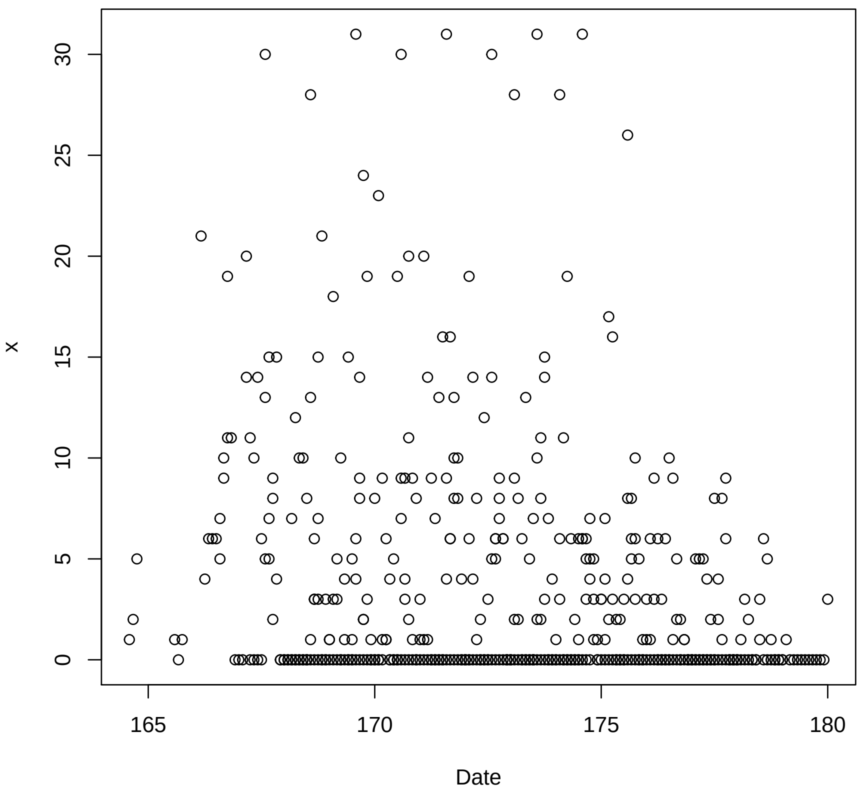
Precipiation might depend more on the departure from the mean (often referred as as normal, whatever that means!). I think it's worth pursuing, but haven't finished the analysis yet.

Precipitation is something that might increase or decrease due to climate change. So, to analyze this, we will evaluate how much precipitation has deveated from the mean, by plotting the rainfall and the mean in a time-series plot.

Second, we need to remove the missing values and evalaute which years have complete years. If you are missing rainy months, then the whole year should be thrown out – but what about partial years in the drought season? We'll need to be consistent – assuming that missing data are not zeros, we'll define complete years as over 300 days of data.

NOTE: The missing values have not been converted to NAs!

```
climate_data$PRCP[climate_data$PRCP==-9999] <- NA

Missing <- aggregate(is.na(climate_data$PRCP),
          list(climate_data$Month, climate_data$Year), sum)

# The aggregate command is used to create a simplified dataset. In this case
# we are creating a sum of PRCP based on each month and year.

Missing$Date = as.numeric(Missing$Group.1) + as.numeric(Missing$Group.2)/12

plot(x ~ Date, data=Missing)
```



Third, we will need to decide what level of aggredation – monthly, yearly, etc. Let's aggreate by month and year to get monthly totals.

There are loads of missing values in many months. Let's cut of the months that have more than 4 missing days.

```
TotalPPT <- aggregate(climate_data$PRCP,
            list(climate_data$Month, climate_data$Year), sum, na.rm=T)

# Check to see what you created.

names(TotalPPT) = c("Group.1", "Group.2", "ppt")

NonMissing <- Missing[Missing$x < 5, c(1:3)]
library(dplyr)

##
## Attaching package:  'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

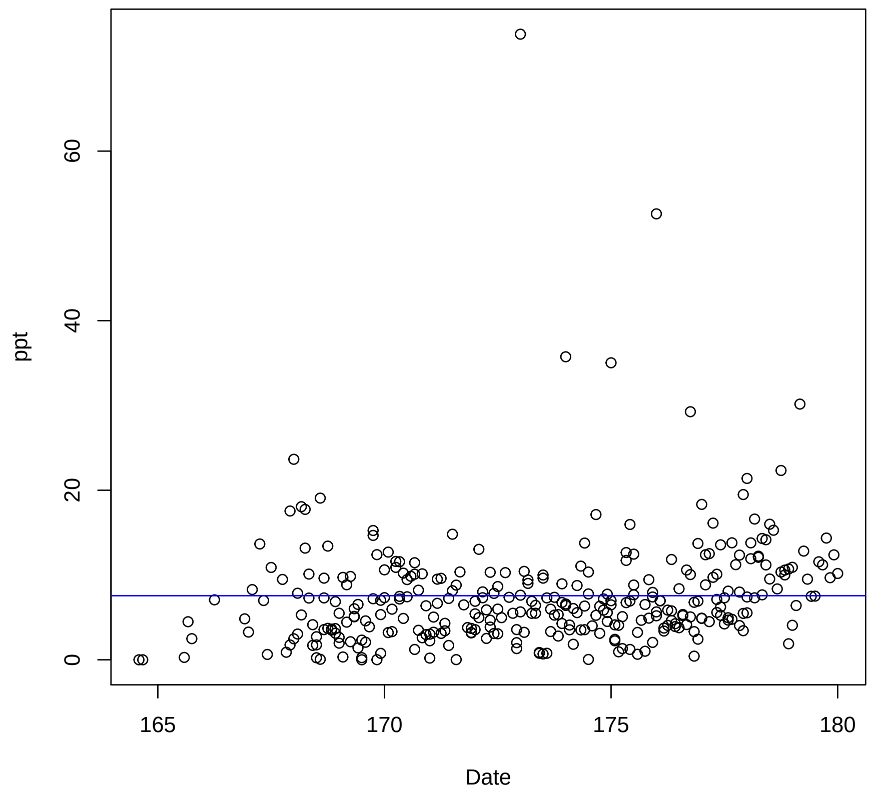
```
PPT <- merge(TotalPPT, NonMissing, all.y=TRUE)
PPT$Date <- as.numeric(PPT$Group.1) + as.numeric(PPT$Group.2)/12
head(PPT)
```

```
##   Group.1 Group.2  ppt x     Date
## 1      01    1963 0.00 1 164.5833
## 2      01    1964 0.00 2 164.6667
## 3      01    1983 7.07 4 166.2500
## 4      01    1991 4.83 0 166.9167
## 5      01    1992 3.27 0 167.0000
## 6      01    1993 8.28 0 167.0833
```

First, we need a "mean" – The IPCC uses 1961-1990 as a norm for temperature, I don't know what is the standard for rainfall or Thailand, so we should look that up. For now, we'll use our filtered records to generate a mean.

```
PRCP_mean = mean(PPT$ppt)
```

```
plot(ppt~Date, data=PPT)
abline(h=PRCP_mean, col="blue")
```

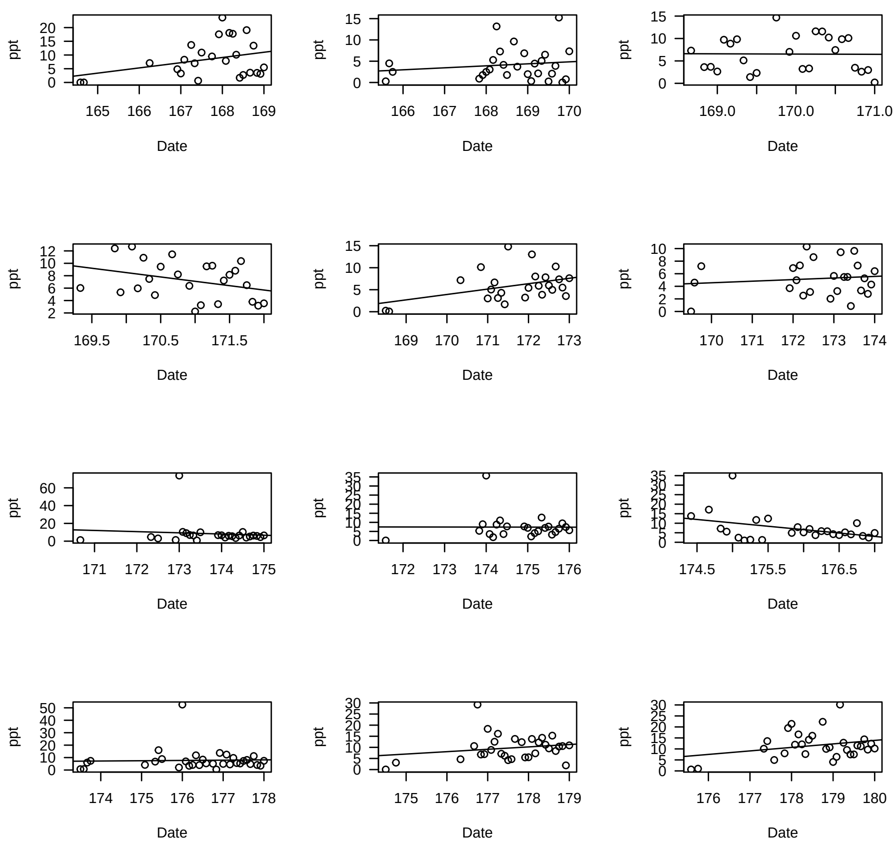


Wow, these data look terrible – the mean looks meaningless given the biased data set. I don't think we can do more analysis with this. But let's look at a few months and see what we can decipher.

```
#LosAngeles£PRCP[LosAngeles£PRCP==-9999] <- NA
#YearlySum = aggregate(PRCP ~ Year, LosAngeles, sum)
#YearlySum£YEAR = as.numeric(YearlySum£Year)
#YearlyMean = mean(YearlySum£PRCP)
```

A yearly mean, based on the annual sum for the entire records. Not sure this is appropriate.

Figure has points of the yearly sum of rainfall and the blue line mean. The greenline is the trend and red line is a five year running average, I think! I am still trying to understand what the code is doing.



ppt
Date
165 166 167 168 169
166 167 168 169 170
169.0 170.0 171.0
169.5 170.5 171.5
169 170 171 172 173
170 171 172 173 174
171 172 173 174 175
172 173 174 175 176
174.5 175.5 176.5
174 175 176 177 178
175 176 177 178 179
176 177 178 179 180

```
#plot(PRCP~YEAR, data=YearlySum, las=1, ty="p")
#abline(h=YearlyMean, col="blue")
#YearlySum.lm = lm(PRCP~YEAR, data=YearlySum)
#abline(coef(YearlySum.lm), col="green")

#n <- 5
#k <- rep(1/n, n)
#k

#y_lag <- stats::filter(YearlySum£PRCP, k, sides=1)
#lines(YearlySum£YEAR, y_lag, col="red")
```

```
#summary(YearlySum.lm)
```

1.8 Assumptions of the Linear Regression

Regression models, like all statistics, rely on certain assumptions. Violations of these assumptions reduces the validity of the model. If the violations are serious, then the model could be misleading or even incorrect.

TBD

1.8.1 Assumptions about ϵ

The error term should have

E(et) = 0 , zero mean

E(et) = s , constant variance

E(et, Xt) = 0 , no correlation with X

TBD E(e , e), no autocorrelation. t t-1

e **Normally distributed** (for hypothesis testing).

Assumption four is especially important and most likely not to be met when using time series data.

Autocorrelation.

1. It is not uncommon for errors to track themselves; that is, for the error a time t to depend in part on its value at t - m, where m is a prior time period.

1.8.2 Model Diagnostics

With every statistical test done, researchers validate their model in some way or anther. Often this entails the use of diagnostics, a standardize battery of procedures to check to see if the data are following the assumptions.

In R four plots are created by default. To see them all at the same time, we need to change the graphical parameters, using the par() function. In this case, we use `par(mfrow=c(2,2))` to create alter the graphics window expects four panels, in this case a 2 rows and two columns.

Try not to get bogged down in the code at this point. But noting this function can be handy in a number of ways to improve one's graphics.

To determine the validity of linear model assumptions (e.g. normality or heterogeneity of variance), you have probably used statistical tests; in contrast statisticians almost exclusively look at diagnostic plots. Why? When assumptions are violated the tests to determine violations do not perform well. So, let's see how to look at these assumptions graphically with these diagnostic plots. Linear models should have diagnostic plots that do not have any obvious structure or pattern. In this case, Figure 1.8.2 should show a great deal remaining structure in the residuals. Although for today, we are not going to try to interpret these figures, but you should notice there is a ton of unaccounted structure, i.e. variance, in the model. This is due, in part, to a violation of independence; these data are serially correlated and the model does not account for that and is inappropriate because of this. It also appears that a straight-line model does not fit well and a curvilinear should be investigated.

A properly specified model is shown in

Figure 5: Default diagnostic plots for a linear model in R.

```
par(mfrow=c(2,2))
plot(lm(TMIN ~ YEAR, data=MonthlyTMINMean[MonthlyTMINMean$MONTH==1,]))
```