

Analyzing Climate Trends

Marc Los Huertos

February 6, 2024 (ver. 0.42)

Contents

1	Introduction	2
1.1	Goals	2
1.2	Weather vs. Climate	2
1.3	Approach	2
1.4	R Code with Custom Functions	2
1.5	Before Starting the Process	3
1.5.1	Reading csv and loading R.data	3
2	Analyzing Monthly Trends	3
2.1	Linear Regression: TMAX, TMIN, and PRCP YEAR, by=YEAR	3
2.1.1	Deprecated Loop Code	4
2.2	Filtering Seasonal Effect	6
2.2.1	Method 1: Filtering by Monthly Mean	6
2.2.2	Method 2: Polynomial Filter	6
3	Extreme Events—Using Daily Records	6
3.1	Complicated Nature of Rainfall Patterns	6
3.2	Drought	7
3.2.1	Days in a Row without Rain	7
3.2.2	Changes in the Rainfall Probability Distributions	8
3.3	Record Setting Temperature Records	8
3.4	KISS	10
3.4.1	Change Point Analysis	10
3.5	Temp & Precipitation Probability	10
3.6	Using library densEstBayes	10
3.7	Probability Distributions	10
3.8	Evaluating Records	10
3.9	Export Options	10
4	Sea Surface Temperature Data – SURP PROJECT WAITING TO HAPPEN	10
5	Satellite Data	11

6 Ice-Core Data	11
7 Conclusions	11

1 Introduction

1.1 Goals

We will use basic linear regression models, i.e. the `lm()` function to determine if there are trends in the data. We will also consider the possibility of non-linear trends, but we will start with the simplest approach. We will evaluate data for extreme events, i.e. the tails of the distribution, and we will also consider the possibility of changes in the distribution over time and records "highs", "lows" and precipitation, and drought.

1.2 Weather vs. Climate

The difference between weather and climate is a measure of time. Weather is what conditions of the atmosphere are over a short period of time, and climate is how the atmosphere "behaves" over relatively long periods of time.

In general, understanding the climate is a question of averages and ranges, of statistical likelihoods, and of repeated or predictable patterns. Weather, on the other hand, is what we experience on a day-to-day basis, and it might vary wildly from minute to minute, hour to hour, day to day, and season to season.

1.3 Approach

We need to address several things that might get in the way of our analysis:

1. Determine if there are trends by months
2. Determine if there if trends are more common in recent years
3. Evaluate distribution changes by decade / score
4. Evaluate extreme events

In contrast to Guide 1 and 2, I couldn't separate the explanation of code from the code and statistical analysis. If you have suggestions of how to stream line this, I am all ears!

1.4 R Code with Custom Functions

Run the **Guide3.R** code and the functions will be loaded into your environment automatically and are designed to help you analyze your statio data.

As we get further into the project, these code chunks may require tweaking because of the questions you are interested for your location falls outside the design of the custom functions. Please let Marc or the mentors know to get assistance tweaking the code!

1.5 Before Starting the Process

Before you begin, make sure you have the stations to read into R. Look at the R environment to see that the stations have been loaded in to R. If not, please Slack Marc and mentors, so we help you get these files into the R environment, which might mean running the previous guide again.

1.5.1 Reading csv and loading R.data

Read and Load Data This function will read the csv files and load the data into R. The function will also return a list of dataframes.

```
## function (x)
## {
##   print("This function might not be needed, waiting to see how the class does")
## }
```

2 Analyzing Monthly Trends

2.1 Linear Regression: TMAX, TMIN, and PRCP YEAR, by=YEAR

In this section, we'll discuss how we might analyze the trends in the data by month. In other words, is there a trend in January, February, March, etc. Moreover, we might find the trends differ dramatically between months.

Analyze Stations for Trends by Month This function will take the list of dataframes and return a list of linear models for each month. The function will also return a summary of the linear models.

```
## function (station)
## {
##   TMAX = station$TMAX
##   TMIN = station$TMIN
##   PRCP = station$PRCP
##   TMAX.lm = lapply(1:12, function(i) {
##     lm(TMAX.a ~ YEAR, data = subset(TMAX, MONTH == i))
##   })
##   TMIN.lm = lapply(1:12, function(i) {
##     lm(TMIN.a ~ YEAR, data = subset(TMIN, MONTH == i))
##   })
##   PRCP.lm = lapply(1:12, function(i) {
##     lm(PRCP.a ~ YEAR, data = subset(PRCP, MONTH == i))
##   })
##   TMAX.summary <- lapply(TMAX.lm, summary)
```

```
## TMIN.summary <- lapply(TMIN.lm, summary)
## PRCP.summary <- lapply(PRCP.lm, summary)
## TMAX.stats <- lapply(TMAX.summary, function(x) {
##   c(r.squared = x$r.squared, x$coefficients[2, 1:4])
## })
## TMIN.stats <- lapply(TMIN.summary, function(x) {
##   c(r.squared = x$r.squared, x$coefficients[2, 1:4])
## })
## PRCP.stats <- lapply(PRCP.summary, function(x) {
##   c(r.squared = x$r.squared, x$coefficients[2, 1:4])
## })
## TMAX.stats <- lapply(TMAX.stats, function(x) x[c(1:5)])
## TMAX <- as.data.frame(do.call(rbind, TMAX.stats))
## TMAX$MONTH <- 1:12
## TMAX$ELEMENT <- "TMAX"
## TMIN.stats <- lapply(TMIN.stats, function(x) x[c(1:5)])
## TMIN <- as.data.frame(do.call(rbind, TMIN.stats))
## TMIN$MONTH <- 1:12
## TMIN$ELEMENT <- "TMIN"
## PRCP.stats <- lapply(PRCP.stats, function(x) x[c(1:5)])
## PRCP <- as.data.frame(do.call(rbind, PRCP.stats))
## PRCP$MONTH <- 1:12
## PRCP$ELEMENT <- "PRCP"
## return(rbind(TMAX[, c(7, 6, 2:5, 1)], TMIN[, c(7, 6, 2:5,
##   1)], PRCP[, c(7, 6, 2:5, 1)]))
## }
## <bytecode: 0x3800cd0>
```

Example of how to use the function Here's an example using the list of station dataframes anomalies.

```
USC00042294.trends <- monthlyTrend.fun(USC00042294.anomalies)
```

Explore Results Table 1 summarizes the monthly trends for TMAX. Admittedly, determining the months with the biggest changes isn't a very good approach for hypothesize testing – it's more like a fishing expedition, but as long as we understand the difference between an a priori hypothesis and an exploratory analysis, we should be okay if we make appropriate conclusions.

We would want to evaluate the trends for TMIN and PRCP too!

2.1.1 Deprecated Loop Code

In case your data downloaded as Montly data, here's the code I created two years ago before the rNOAA library started failing and I had to spend all week

	ELEMENT	MONTH	Estimate	Std. Error	t value	Pr(> t)	r.squared
1	TMAX	1	-0.01	0.01	-0.80	0.43	0.02
2	TMAX	2	-0.02	0.01	-1.31	0.20	0.05
3	TMAX	3	-0.01	0.01	-0.71	0.48	0.02
4	TMAX	4	-0.01	0.01	-0.48	0.64	0.01
5	TMAX	5	-0.02	0.01	-1.29	0.21	0.05
6	TMAX	6	-0.02	0.01	-1.90	0.07	0.10
7	TMAX	7	-0.00	0.01	-0.49	0.63	0.01
8	TMAX	8	-0.01	0.01	-1.57	0.13	0.07
9	TMAX	9	0.00	0.01	0.39	0.70	0.00
10	TMAX	10	-0.01	0.01	-0.73	0.47	0.02
11	TMAX	11	-0.01	0.01	-1.06	0.30	0.03
12	TMAX	12	-0.01	0.01	-1.48	0.15	0.06

Table 1: TMAX Trends

redoing the code!

Station data frames were called GSOM (Monthly data). See if you can interpret each of the steps. Evaluate both TMAX and TMIN in GSOM by Year using MonthEvalStats() function.

```
## function (GSOM)
## {
##     sumstats = NA
##     for (m in 1:12) {
##         TMIN.lm = lm(TMIN ~ Date, GSOM[GSOM$Month == m, ])
##         TMAX.lm = lm(TMAX ~ Date, GSOM[GSOM$Month == m, ])
##         PPT.lm = lm(PPT ~ Date, GSOM[GSOM$Month == m, ])
##         sumstats = rbind(sumstats, data.frame(Month = m, Param = "TMIN",
##             Slope = coef(TMIN.lm)[2], r2 = summary(TMIN.lm)$r.squared,
##             p_value = anova(TMIN.lm)$"Pr(>F)"[1]), data.frame(Month = m,
##             Param = "TMAX", Slope = coef(TMAX.lm)[2], r2 = summary(TMAX.lm)$r.squared,
##             p_value = anova(TMAX.lm)$"Pr(>F)"[1]), data.frame(Month = m,
##             Param = "PPT", Slope = coef(PPT.lm)[2], r2 = summary(PPT.lm)$r.squared,
##             p_value = anova(PPT.lm)$"Pr(>F)"[1]))
##     }
##     sumstats = data.frame(sumstats)[-1, ]
##     rownames(sumstats) <- NULL
##     head(sumstats)
##     sumstats$Symbol = ""
##     sumstats$Symbol[sumstats$p_value < 0.05] = "*"
##     sumstats$Symbol[sumstats$p_value < 0.01] = "**"
##     sumstats$Symbol[sumstats$p_value < 0.001] = "***"
##     return(sumstats)
## }
```

2.2 Filtering Seasonal Effect

Since we are looking at trends over years, it might be useful to filter out the seasonal effects.

There are several ways to filter out seasonal effects. The easiest way is subtract the mean value for each date, but that's tricky because every four years there is an extra day in February – although there are ways to deal with this, a more straight forward way is to use mean monthly values to capture the seasonality for each month. With 12 months, this is a pretty good approach because the pretty good resolution is pretty good when the station has complete records.

2.2.1 Method 1: Filtering by Monthly Mean

One way of doing this is creating a matrix of values for each month and then subtracting the mean value for each month, for the whole record, not just the 1960-1990 “normals”. If this is of interest, we can help you with this.

2.2.2 Method 2: Polynomial Filter

Another method is to use a polynomial filter. Perhaps, someday, I'll work on this. Great student project to be followed up with.

```
# fit polynomial:  $x^2*b1 + x*b2 + \dots + bn$ 

# create time series object
#X = [i%365 for i in range(0, len(series))]
# y = series.values

# degree = 4
#coef = polyfit(X, y, degree)
# print('Coefficients: %s' % coef)
# create curve
```

3 Extreme Events—Using Daily Records

3.1 Complicated Nature of Rainfall Patterns

Rainfall trends are tough. Extreme events can occur in 24 hours or over long periods that might result in floods or droughts. Each region might have different patterns, so developing a consistent approach is tough.

We can look for trends in monthly averages, number of days without rain (important in tropics), and/or extreme events based on daily or hourly data.

In addition, the definition of extreme events is highly regionally specific. Thus, if this is of interest, let us know and we'll help you develop code!

Rainfall totals by season might be a useful way to think about changes, because the rainfall is often seasonal, I wonder if we can see patterns by season.

3.2 Drought

Days without rain...within a calendar year... bleed over between years isn't captured.. This is screwed up, Drought.run needs work.

3.2.1 Days in a Row without Rain

One proxy for drought is a long period without rain. This is a bit tricky because it's not just the number of days without rain, but the number of days without rain in a row. So, I found a function call `rle()` that counts the number of days in a row without rain.

```
## function (station = USC00040693, threshold = 0.1)
## {
##   drought.df <- subset(station, ELEMENT == "PRCP")
##   x <- drought.df$VALUE
##   length(x)
##   drought = x < threshold
##   str(drought)
##   drought = rle(drought)
##   str(drought)
##   rle.values <- as.data.frame(do.call(cbind, drought))
##   str(rle.values)
##   drought.df$Drought = as.logical(rep(rle.values$values, times = rle.values$lengths))
##   head(drought.df)
##   drought.df$Length = rep(rle.values$lengths, rle.values$lengths)
##   head(drought.df)
##   if ("Ymd" %in% names(drought.df) == FALSE) {
##     drought.df$Ymd = as.Date(as.character(drought.df$DATE),
##       format = "%Y%m%d")
##     drought.df$MONTH = as.numeric(format(drought.df$Ymd,
##       "%m"))
##     drought.df$YEAR = as.numeric(format(drought.df$Ymd, "%Y"))
##   }
##   drought.df$WY = ifelse(drought.df$MONTH < 10, drought.df$YEAR,
##     drought.df$YEAR + 1)
##   str(drought.df)
##   aggregate(drought.df$Drought, by = list(YEAR = drought.df$YEAR,
##     MONTH = drought.df$MONTH), FUN = sum)
##   DroughtperYear = aggregate(drought.df$Drought, by = list(WY = drought.df$WY),
##     FUN = sum)
##   RecordperYear = aggregate(drought.df$DATE, by = list(WY = drought.df$WY),
##     FUN = length)
```

```
## drought = merge(RecordperYear, DroughtperYear, by = "WY")
## head(drought)
## drought$DroughtPerYear = round(drought$x.y/drought$x.x *
## 100)
## head(drought)
## return(drought)
## }
```

Example of how to use the function Here's an example using the station dataframe.

```
USC00040693.drought <- droughtCount.fun(USC00040693, threshold=0.1)
```

3.2.2 Changes in the Rainfall Probability Distributions

In this example, we could either use `rnorm` or a kernel density estimate, for each 10 years or 20 years segments and see if there is a difference over time. In general, these can be pretty compelling. But it's the mean that is really driving the analysis, so that might be something to look at by decade to see if there is a trend.

##		Ymd	ID	ELEMENT	VALUE	MONTH	YEAR	Decade
##	57285	2014-07-04	USC00044412	SNWD	0.0	7	2014	2010
##	56855	2014-05-01	USC00044412	PRCP	0.0	5	2014	2010
##	19983	1991-10-11	USC00044412	TMAX	28.9	10	1991	1990
##	3563	1908-12-20	USC00044412	PRCP	0.0	12	1908	1900
##	60299	2016-04-17	USC00044412	TOBS	17.2	4	2016	2010
##	13901	1988-11-11	USC00044412	SNOW	0.0	11	1988	1980
##	31160	1999-12-16	USC00044412	SNWD	0.0	12	1999	1990
##	36669	2003-01-22	USC00044412	TOBS	8.3	1	2003	2000
##	58804	2015-06-30	USC00044412	TMAX	32.2	6	2015	2010
##	27546	1997-11-05	USC00044412	TMAX	25.0	11	1997	1990
##	[1]	"ID"	"DATE"	"ELEMENT"	"VALUE"	"M.FLAG"	"Q.FLAG"	
##	[7]	"S.FLAG"	"OBS.TIME"	"Ymd"	"MONTH"	"YEAR"	"Decade"	

3.3 Record Setting Temperature Records

In many cases, people seem to "feel" how temperature has been changing over time, and new records seem to capture the attention in the media. So, we'll create a updated record of maximum temperatures and display them.

This is a common way to communicate temperatures changes. I suspect we have a better sense of change when we notice "extreme" events, and this also fits the news media's need for "new" stories.

	Decade	Fall	Spring	Summer	Winter
1	1890.00	1.47	2.41	0.46	4.16
2	1900.00	1.05	1.66	0.79	4.92
3	1910.00	1.22	2.48	0.47	4.80
4	1920.00	1.71	2.78	0.73	5.26
5	1940.00	0.71			0.90
6	1950.00	0.01	1.42	0.56	2.64
7	1980.00	0.61	1.33	0.00	2.09
8	1990.00	0.76	1.86	0.37	4.11
9	2000.00	1.10	1.23	0.17	3.07
10	2010.00	1.29	1.52	0.27	3.95
11	2020.00	1.27	2.64	0.48	3.30

```
ggplot( ) +
  geom_bar(data = records, aes(x=Year, y=Num, fill=Group),
    stat="identity", position="identity") +
  xlim(min(CHCND$Year), max(CHCND$Year)-1) +
  ylab("Number of Extreme Temps") + # for the y axis label
  scale_fill_manual("Legend",
    values = c("Record Highs" = "red", "Record Lows" = "blue"))
```

```
#TMIN in the March, at station USC00042294
par(mfrow=c(1,1))
station = subset(USC00042294.anomalies[[2]], subset=(MONTH==3))

plot(TMIN.a ~ YEAR, data= station, type="p", col="grey", pch=19,
  xlab="Year", ylab="Temperature Anomaly (C)", main="TMIN Data")

TMIN.March.lm = lm(TMIN.a ~ YEAR, data= station)
abline(coef(TMIN.March.lm), col="red" )

%\item Box Plot of TMAX.

station = subset(USC00042294.anomalies[[1]])

boxplot(TMAX ~ MONTH, data= station, xlab="Month", ylab="Temperature Anomaly (C)",
  main="TMAX Data", col="grey")

)
```

3.4 KISS

Keeping it simple is critical in communicating scientific information. In this section, I try to come up with a consistent message for every state and a simple graphic.

3.4.1 Change Point Analysis

First, TMIN and TMAX and change point analysis, I need to do some background in this, but I used this method in a recent paper with co-author, but I didn't do that part of the analysis, so more reading is needed (<https://cran.r-project.org/web/packages/mcp/readme/README.html>).

3.5 Temp & Precipitation Probability

To highlight the patterns of change, it might be useful to analyze how the probability distribution might change – we can use a normal probability distribution as a theoretical distribution (and we can check if this distribution is appropriate with a Chi-Square test), or we can use the data to create an empirical distribution, which is my favored approach.

We might start with decade bins, or 20 years bins (scores) to simplify the analysis.

3.6 Using library densEstBayes

These values suggest that there is good reason

3.7 Probability Distributions

Thinking more about this one...

3.8 Evaluating Records

TBD

3.9 Export Options

TBD

4 Sea Surface Temperature Data – SURP PROJECT WAITING TO HAPPEN

In contrast to terrestrial data, sea surface temperature (SST) is quite difficult to obtain and process. There are numerous tools to access the data, but they often require knowledge of complex software tools that are not easy to set up or programming experience with python or others.

<https://climexp.knmi.nl/select.cgi?id=someone@somewhere&field=ersstv5>

There are, however, a few tools build for R users that seem to accomplish all that we need.

https://rda.ucar.edu/index.html?hash=data_user&action=register

<https://rda.ucar.edu/datasets/ds277.9/>

Alternatively, we can download flat ascII tables of gridded data:

<https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/>

5 Satellite Data

TBD

6 Ice-Core Data

TBD

7 Conclusions

Developing a robust method to analyze weather stations is both time consuming and difficult to justify the outcome. In part because the data suggest that each station (region) requires different types of analysis, based on the expected patterns of temperature and rainfall. As climate scientists have known for decades, the terminology of global warming is not very useful. Not because scientists are trying to hide something or promote some biased agenda, but that even as warming of the global average is well documented, the impacts of climate change on each region is highly specific, requiring specificity in the analysis.

Hopefully, this little analysis has created some mechanism for others to appreciate this complexity.

The document took