Guide 3: Analyzing & Visual Display of Climate Trends

Marc Los Huertos

February 15, 2024 (ver. 0.55)

Contents

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 mihght vary wildly from minute to minute, hour to hour, day to day, and season to season.

1.3 Approach

After Feb 13, I realized that I needed to provide a different structure to help you analyze and display the data. Instead of two separate documents, I have combined the code to analyze and display data in one document. Guide 4 is not completely focused on the script and video process.

Here's how I might start: Read the EPA summaries and see what stands out as compelling types of analyses. Then read news articles on the region and see what qualifies as "weather" news in the region.

I suggest you consider 3-5 questions that you think are interesting and then we can work together to see if we can answer those questions using the station data.

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!

Moreover, I haven't completed every possible analysis yet, but I tried to post the potential ones that I could think of. If you have some idea that is missing, let me know and we can figure out how to create an analysis for it!

1. Explore various methods to display climate data
2. Use the following Google Doc to suggest code and/or some functions that can be used to analyze your data. Marc and mentors will meet to see if we can help you with your code.

1.4 R Code with Custom Functions

From the Canvas page, go to the Guide3functions.R file and download the file to your computer. Then upload the file to Rstudio directory you are using for the project.

Open the file in Rstudio and run the code, using the "source". button near the top of the editor window.

Run the Guide3functions.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.

```
ls()
```

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

If you decide to use the knitting function in R, it turns out that the R environment that the console has is not availabe for the knitting function. So, I had to create a way to bring the data into the knitted environment. If you would like to do this, see Section 8.1.

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: Function: monthlyTrend.fun()

Example of how to use the function Here's an example using the list of station dataframes anomalies. Note: I have often spelled the word anamoly wrong!

```
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! You can use the Environment tab in Rstudio to see the trend results for all three variables withut needing any code.

	ELEMENT	MONTH	Estimate	Std. Error	t value	Pr(>|t|)	r.squared
1	TMAX	1	0.01	0.01	1.10	0.27	0.01
2	TMAX	2	-0.01	0.01	-1.13	0.26	0.01
3	TMAX	3	-0.01	0.01	-1.52	0.13	0.02
4	TMAX	4	-0.00	0.01	-0.56	0.58	0.00
5	TMAX	5	0.00	0.01	0.19	0.85	0.00
6	TMAX	6	0.00	0.00	0.68	0.50	0.00
7	TMAX	7	-0.01	0.00	-2.22	0.03	0.04
8	TMAX	8	-0.01	0.00	-1.67	0.10	0.02
9	TMAX	9	0.01	0.00	1.88	0.06	0.03
10	TMAX	10	0.00	0.01	0.68	0.50	0.00
11	TMAX	11	-0.01	0.01	-2.40	0.02	0.05
12	TMAX	12	-0.00	0.01	-0.60	0.55	0.00

Table 1: TMAX Trends

2.2 Plotting a Simple Trend Line

Here's an example of how to plot a simple trend line for TMIN for January. We are getting ahead of ourselves here, but it's a good example of how to plot a trend line and some of the back and forth we'll need to do between analysis and presentation.

I suggest you select the element/month with the strongest signal. I selected TMIN and January because it had a strong signal), i.e. highly significant with a p-value ¡¡ 0.05.

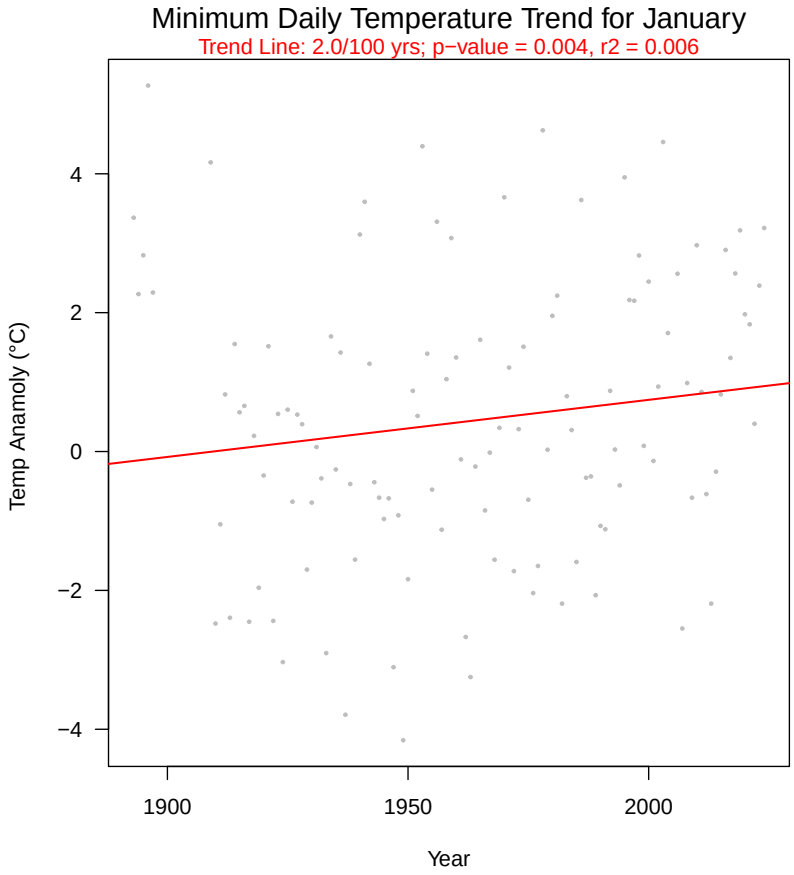
```
par(mar=c(5,5,3,4), las=1, mfrow=c(1,1))

plot(TMIN.a ~ Ymd, data=subset(USC00042294.anomalies$TMIN,
    MONTH==1), las=1, pch=20, cex=.5, col="grey",
    ylab="Temp Anamoly (C) ",
    main="",
    xlab="Year", frame.plot=TRUE)

  mtext("Minimum Daily Temperature Trend for January",
        cex=1.3, side=3, line=1)
  subtitle <- c("Trend Line: 2.0/100 yrs; p-value = 0.004, r2 = 0.006")
  mtext(subtitle, side=3, line=0, cex=1.0, col="red")

USC00042294.lm = lm(TMIN.a ~ Ymd, data=subset(USC00042294.anomalies$TMIN,
    MONTH==1))

abline(coef(USC00042294.lm), col="red", lwd=1.5)
```



2.3 Marc's Custom Function for Plotting Trends

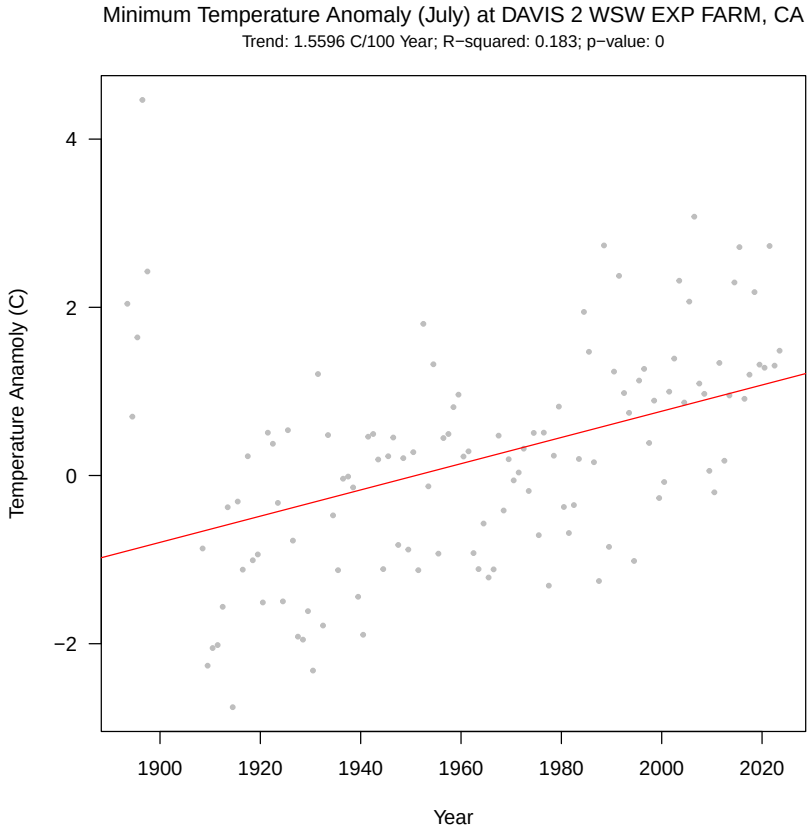
I have a custom function that will plot the trend line for you. It's a bit more complicated than the simple example, but it has a bug that I am still working to fix.

Plot Trends by Month I have created a basic function template, but we'll develop this together when we have a chance. This function will take the list of creates a plot based on three inputs: list of station dataframes, the element, and the month.

Function: **Function: plotTrend.fun()**

Example of how to use the function Here's an example using the list of station dataframes anomalies. Note: I have often spelled the word anomaly wrong!

```
plotTrend2.fun(USC00042294.anomalies, "TMIN", 7)
```



Explore Results I suggest you look carefully at the plots for hidden trends. In this case, look how the temperature really starts warming after the 1960s. I don't know that that means, but seems important!

```
plotTrend.fun(USC00042294.anomalies, "TMAX", 6)

## Error in plotTrend.fun(USC00042294.anomalies, "TMAX", 6):  could
not find function "plotTrend.fun"

plotTrend.fun(USC00042294.anomalies, "TMIN", 7)

## Error in plotTrend.fun(USC00042294.anomalies, "TMIN", 7):  could
not find function "plotTrend.fun"

plotTrend.fun(USC00042294.anomalies, "PRCP", 1)
```

```
## Error in plotTrend.fun(USC00042294.anomalies, "PRCP", 1):  could
not find function "plotTrend.fun"
```

2.4 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.

2.4.1 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.

2.4.2 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 + ... + 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. Exteme 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 regionaally 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 pattners 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.

3.3 Standardized Precipitation Evapotranspiration Index (SPEI)

The Standardized Precipitation Evapotranspiration Index (SPEI) is an extension of the widely used Standardized Precipitation Index (SPI). The SPEI is designed to take into account both precipitation and potential evapotranspiration (PET) in determining drought. Thus, unlike the SPI, the SPEI captures the main impact of increased temperatures on water demand.

I don't know if we can do this yet, but I am working on the code to develop the SPEI using a publish R package https://cran.r-project.org/web/packages/SPEI/.

3.3.1 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)
```

3.3.2 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 segmments 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.

3.4 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.

How to do this? These might be the steps!

1. First, we'll calculate the mean temperature for the entire period.

	Decade	Fall	Spring	Summer	Winter
1	1890.00	54.02	65.02	4.06	114.48
2	1900.00	32.56	60.62	2.86	127.27
3	1910.00	27.54	36.74	2.44	127.82
4	1920.00	34.28	38.41	2.16	92.32
5	1930.00	21.94	43.33	2.38	97.58
6	1940.00	33.71	54.42	1.59	105.02
7	1950.00	32.93	46.19	2.16	122.48
8	1960.00	40.64	44.20	3.00	107.50
9	1970.00	50.57	44.26	3.61	103.29
10	1980.00	49.78	54.81	1.82	112.22
11	1990.00	39.47	48.95	3.22	138.97
12	2000.00	33.30	47.23	1.33	136.12
13	2010.00	31.17	65.72	2.96	109.66
14	2020.00	0.83	21.07	0.22	28.06

2. We might sort the data set, by highest/lowest temperature for each day of the year.

3. Then we can see if the frequency of the highest/lowest temperatures is increasing over time.

4. We'll then create a new column that will be the minimum temperature for each day.

5. We'll then create a new column that will be the maximum temperature for each day, but only if it is the maximum temperature for that day.

6. We'll then create a new column that will be the minimum temperature for each day, but only if it is the minimum temperature for that day.

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.

```
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")

    )
```

3.5 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.

3.5.1 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).

3.6 Temp & Precipitation Probability

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

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

3.7 Using library densEstBayes

These values suggest that there is good reason

3.8 Probability Distributions

Thining more abou this one...

3.9 Evaluating Records

TBD

3.10 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 Marc's Custom Functions

7.1 monthlyTrend.fun

```
## 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: 0x9ed4460>
```

7.1.1 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 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)
## }
```

8 Troubleshooting

8.1 Loading and Reading Data into the Knit Environment

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. If you clean up the environment, you might need to read.csv data into the R environment again. Again, check the environment to see if the objects are there.

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

Loading Saved RData files

If you have saved the RData files, you can load them into the R environment using the following code:

DOESN'T WORK YET!

```
LoadData.fun(datapath)

## [1] "RData file does not exist (Not if you have all functions in one Rmd file!)."
```

```
ls()

##  [1] "datapath"                "droughtCount.fun"        "floor_decade"
##  [4] "LoadData.fun"            "MonthEvalStats.fun"      "monthlyTrend.fun"
##  [7] "plotTrend2.fun"          "PRCP.Decade.mean"        "PRCP.Decade.sd"
## [10] "read_and_load_data.fun" "spi.fun"                 "State_names"
## [13] "StateIDs"               "States"                  "station"
## [16] "station_names"          "Stations"                "subtitle"
## [19] "USC00040693.anomalies"  "USC00042294.anomalies"   "USC00042294.lm"
## [22] "USC00042294.trends"
```

9 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 compexity.

The document took

10 Mapping (GIS) and Weather Stations

10.1 Simple Mapping of Stations

10.2 More Complex Mapping of Stations

```
library(geodata)
d <- worldclim_country(country = "USA", var = "tmin",
```

```
                        path = tempdir())
terra::plot(mean(d), plg = list(title = "Min. temperature (C)"))
```

```
library(here)
library(xtable)

stations.active.oldest = read.csv(
  here("04_Regional_Climate_Trends", "stations.active.oldest.csv"))

# OR
# use file.choose() to select the file
# filename = "MY.PATH/04_Regional_Climate_Trends/stations.active.oldest.csv"
# stations.active.oldest = read.csv(filename)
```

10.3 Map US Weather Stations

Here's map that has been transformed using a bunch of libraries that I don't really know how to use, but I found webpage that told me how to make this!

It would be nice to add Canada and Mexico so the USA is not floating in space. I'll work on that later.

```
library(usmap)
library(ggplot2)
library(sf)

## Linking to GEOS 3.7.2, GDAL 3.0.4, PROJ 6.3.2; sf_use_s2() is TRUE

library(tidyverse)

## -- Attaching core tidyverse packages ------------------------ tidyverse
2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v lubridate 1.9.3     v tibble    3.2.1
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts()
--
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to
force all conflicts to become errors

library("rnaturalearth")
library("rnaturalearthdata")
```

```
##
## Attaching package:  'rnaturalearthdata'
##
## The following object is masked from 'package:rnaturalearth':
##
##     countries110
```

```
library(usmap)

world <- ne_countries(scale = "medium", returnclass = "sf")
class(world)
```

```
## [1] "sf"         "data.frame"
```

```
usa <- subset(world, admin == "United States of America")

(mainland <- ggplot(data = usa) +
     geom_sf(fill = "cornsilk") +
     #geom_sf(data = subset(sites), size = 3, shape = 21, fill = "gray70") +
     geom_sf(data = subset(sites.2163), size = 1, shape = 21, fill = "red") +
    coord_sf(crs = st_crs(2163), xlim = c(-2500000, 2500000), ylim = c(-2300000,
        730000)))
```

```
## Error in subset(sites.2163):  object 'sites.2163' not found
```

```
(alaska <- ggplot(data = usa) +
     geom_sf(fill = "cornsilk") +
    geom_sf(data = subset(sites.2163), size = 1, shape = 21, fill = "red") +
     coord_sf(crs = st_crs(3467), xlim = c(-2400000, 1600000), ylim = c(200000,
        2500000), expand = FALSE, datum = NA))
```

```
## Error in subset(sites.2163):  object 'sites.2163' not found
```

```
(hawaii  <- ggplot(data = usa) +
     geom_sf(fill = "cornsilk") +
     geom_sf(data = subset(sites.2163), size = 1, shape = 21, fill = "red") +
     coord_sf(crs = st_crs(4135), xlim = c(-161, -154), ylim = c(18,
        23), expand = FALSE, datum = NA))
```

```
## Error in subset(sites.2163):  object 'sites.2163' not found
```

```
(world.sf <- ggplot(data =world) +
     geom_sf() +
     #geom_sf(data = subset(sites), size = 3, shape = 21, fill = "gray70") +
     geom_sf(data = subset(sites.2163), size = 1, shape = 21, fill = "red") +
    coord_sf(crs = st_crs(2163), xlim = c(-2500000, 2500000), ylim = c(-2300000,
        730000)))
```

```
## Error in subset(sites.2163):  object 'sites.2163' not found
```

```
mainland +
 annotation_custom(
      grob = ggplotGrob(alaska),
      xmin = -2750000,
      xmax = -2750000 + (1600000 - (-2400000))/2.5,
      ymin = -2450000,
      ymax = -2450000 + (2500000 - 200000)/2.5
  ) +
  annotation_custom(
      grob = ggplotGrob(hawaii),
      xmin = -1250000,
      xmax = -1250000 + (-154 - (-161))*120000,
      ymin = -2450000,
      ymax = -2450000 + (23 - 18)*120000
  )
```

```
## Error in eval(expr, envir, enclos):  object 'mainland' not found
```

11 Mapping Long-term Weather Records

11.1 Creating A Basic Trend Plot

Plot Trends by Month I have created a basic function template, but we'll develop this together when we have a chance. This function will take the list of creates a plot based on the following inputs: list of station dataframes, the element, and the month.

Function: Function: plotTrend2.fun()

Example of how to use the function Here's an example using the list of station dataframes anomalies. Note: I have often spelled the word anomaly wrong!

```
plotTrend.fun(USC00042294.anomalies, "TMIN", 7)
```

```
## Error in plotTrend.fun(USC00042294.anomalies, "TMIN", 7):  could
not find function "plotTrend.fun"
```

Explore Results I suggest you look carefully at the plots for hidden trends. In this case, look how the temperature really starts warming after teh 1960s. I don't know that that means, but seems important!

12 Extreme Events–Using Daily Records