echo=FALSE, results='hide'

```
options(width=60, echo=FALSE, results='hide')
```

```
get_locationid <- function(FIPS){
   fips = ncdc_locs(locationcategoryid='ST', limit=55)
   temp <- data.frame(State = fips$data$name[FIPS],
              id = fips$data$id[FIPS])
   temp$id <- as.character(temp$id)
   temp$State <- as.character(temp$State)
   return(temp)
}
```

```
## 'data.frame': 1 obs. of  2 variables:
##  $ State: chr "Alabama"
##  $ id   : chr "FIPS:01"
```

Communicating Climate Change and Individual Weather Records–Alabama

Marc Los Huertos

May 27, 2022

1 Evaluating Terrestrial Meteorological Data

1.1 Selected History of Climate Science

Geologists have known the climate has been changing over the Earth's history. But what causes these changes has been a major research area for over 100 years. There are numerous drivers that contribute to changing climates – including the arrangement of the continents on the planet, the distance to the sun, energy generated by the sun, volanic activity, and the composition of the Earth's atmosphere.

It's the last one that we'll spend time because the Earth's temperature are changing pretty dramatically over the last 100 years and the cause is no mystery – the human activity that has released CO2 into the atmosphere. The two main sources of CO2 is from land use change, e.g. deforestration, and the burning of fossil fuels, e.g. coal, oil, and natural gas.

The first to propose the role of CO2 on the Earth's atmosphere was a Swedish scientist Svante Arrhenius, who figured out that CO2 absorbs infarred light. Moreover, he deduced that the Earth's temperature was actually warmer than it might otherwise be if CO2 was not part of the Earth's atmoshere.

1.2 Why look at individual stations?

I don't think there is a single, perfect way to analyze and communicate climate change. But the beauty of the network of stations in the USA and around the world is that these stations record weather as expecienced by local peoples. And while indiviudual stations may not represent the overall regional and global patterns well, this give us a mechanism to connect local experiences to regional or global processes.

Of course, some may fixate on the local pattern and remain unconvinced of the larger context and for those folks, there may be better ways to communicated climate data.

However, I would be remiss in failing to mention that some may fixate on local patterns and use these patterns to ignore or to dimiss the patterns in other regions.

Finally, the impacts of climate change are highly specific to the region in question. Thus, once someone understands the impacts on climate change in their region, they my not be able to appreciate how differnet the climate impacts might affect other peoples, who maybe more vulneratble, around the globe.

Thus, with these weaknessed in mind, I will pursue this project with an eye to address these other issues at later stages.

1.3 Approach

1.3.1 NOAA Data Records

The US National Oceanic and Atmospheric Adminstration (NOAA) maintains several sources of digital weather data from the USA and beyond. These data have been collected from stations around the country to support a wide range of human activities that include farming, aviation, shipping, and even armed conflict.

At various times, these records have been used to evaluate long-term climate change with varying success. Without a doubt, these data are not perfect, but they remain that foundation of an effective adn professionally maintained environmental monitoring program that engenders integrity, even when facing budget cuts.

1.3.2 rNOAA Package and R

R is an open source programming environment that has become one of the most popular tools for statiticians and data scientists. Capitalizing on the open source framework, a wide range of libraries or packages have been developed to faciliate data processing, analysis, and graphical displays. On such package is rNOAA developed to collect and display climate records stored on NOAA servers.

Using the package requires the use of a key. To maintain the integrity of the key, it's best to avoid posting the key in a public repository and to encryp the key to ensure it's not abused.

1.4 Selecting Weather Records by State

1.4.1 State Temperature Records

There are numerous ways to analyze temperature records, where stations can be analyzed individually or records could be sampled and analyzed in spatially in grids. Each of these are valid approaches depending on the question to be addressed.

In this case the question is "Based on the longest state meterological record, is there a temperature trend?"

1.4.2 List of Cities

rNOAA has a simple function to list for each of the states and the weather stations in each. We'll use ncdc_locs() functions to select each state and ncdc_station() to obtain the station ids with the longest records.

```
# List of States (alpha beta)
ncdc_locs(locationcategoryid='ST', limit=55)
```

The function queries the NOAA website and retrieves state codes, "FIPS:XX".

NOTE2: It would be nice to make a map of how concentrated the stations spatially.

1.4.3 Selection Stations

With the state ids, we can then, get metadata for all the weather stations, which will work to get the longest records, using `ncdc_stations()`.

First, we subset the data for stations that actively collecting data. Then we'll sort to the active stations to find the one with the longest records. We will use these stations for our analysis.

1.4.4 Select State

Using the rNOAA function ncdc_locs(), we can queary NOAA's database to identify station codes (FIPS) by state. With the states and some territories, there are 55 FIPS for US weather stations.

After

```
GSOM_Stations <- ncdc_stations(datasetid='GSOM',
                datatypeid = c("TMAX", "TMIN"),
                locationid=fips$id, limit=1000,
                sortfield = 'maxdate', sortorder='desc')

GSOM_Recent =
   GSOM_Stations$data[GSOM_Stations$data$maxdate>='2021-11-01',]

GSOM_Coverage =
   GSOM_Recent[GSOM_Recent$datacoverage > 0.92,]
GSOM_Sorted =  GSOM_Coverage[order(GSOM_Coverage$mindate),]
#GSOM_Longest =
 #  GSOM_Coverage[GSOM_Coverage£mindate == min(GSOM_Coverage£mindate),]
GSOM_Longest = GSOM_Sorted[1,] #Pick longest
# Second and Third for Comparisons
# GSOM_Longest = GSOM_Sorted[3,]
# GSOM_Longest = GSOM_Sorted[4,]
```

The record selected has the following metadata associated with it, which will be used for nameing, labeling, and mapping.

```
##     elevation    mindate    maxdate latitude
## 109     136.6 1888-02-01 2022-03-01  33.4163
##                 name datacoverage                id
## 109 TALLADEGA, AL US       0.9236 GHCND:USC00018024
##     elevationUnit longitude
## 109        METERS   -86.135
```

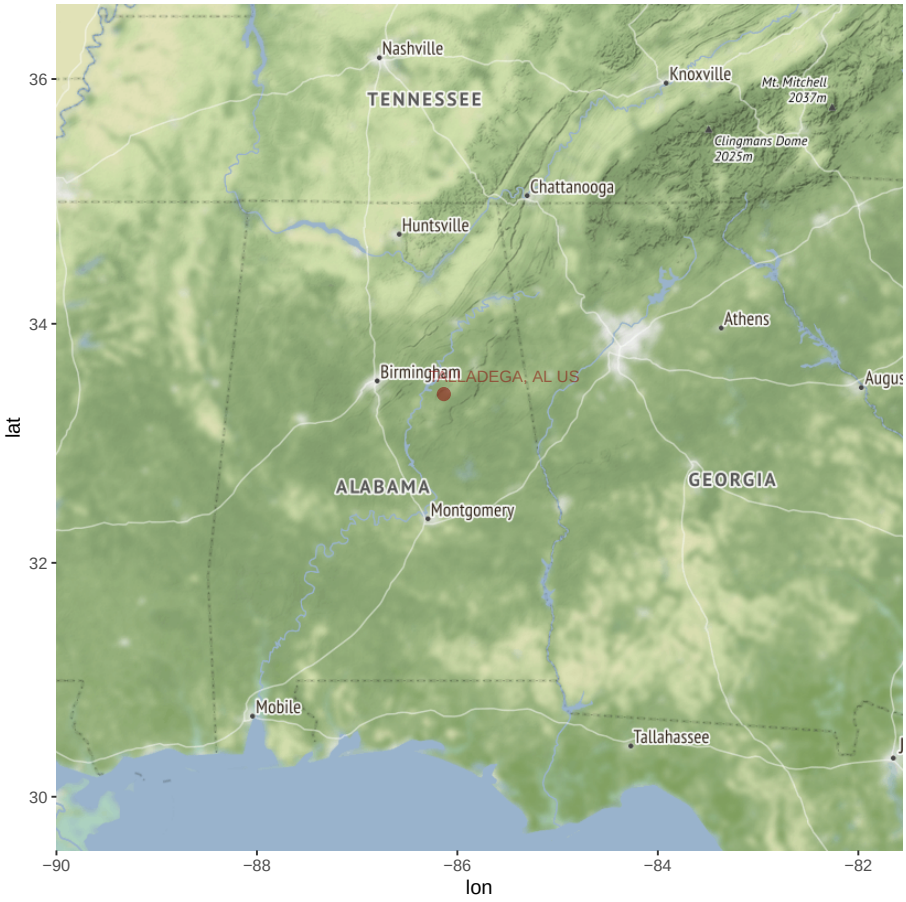
2 Download GSOM Data using rnoaa

```
## [1] 1888
```

2.1 Show Map of Location

```
GSOM_Longest$name

## [1] "TALLADEGA, AL US"

lat = GSOM_Longest$latitude
lon = GSOM_Longest$longitude
station = c(lon, lat)
station.df <- data.frame(lon = GSOM_Longest$longitude,
                         lat = GSOM_Longest$latitude,
                         Station = GSOM_Longest$name);
str(station.df)

## 'data.frame': 1 obs. of  3 variables:
##  $ lon    : num -86.1
##  $ lat    : num 33.4
##  $ Station: Factor w/ 1 level "TALLADEGA, AL US": 1

myMap <- get_map(location=station, zoom=7, scale =2,
source="stamen", maptype="terrain", messaging = FALSE, crop=FALSE)

## Source :  https://maps.googleapis.com/maps/api/staticmap?center=33.4163,-86.135&zoom=7&s
## Source :  http://tile.stamen.com/terrain/7/32/50.png
## Source :  http://tile.stamen.com/terrain/7/33/50.png
## Source :  http://tile.stamen.com/terrain/7/34/50.png
## Source :  http://tile.stamen.com/terrain/7/32/51.png
## Source :  http://tile.stamen.com/terrain/7/33/51.png
```

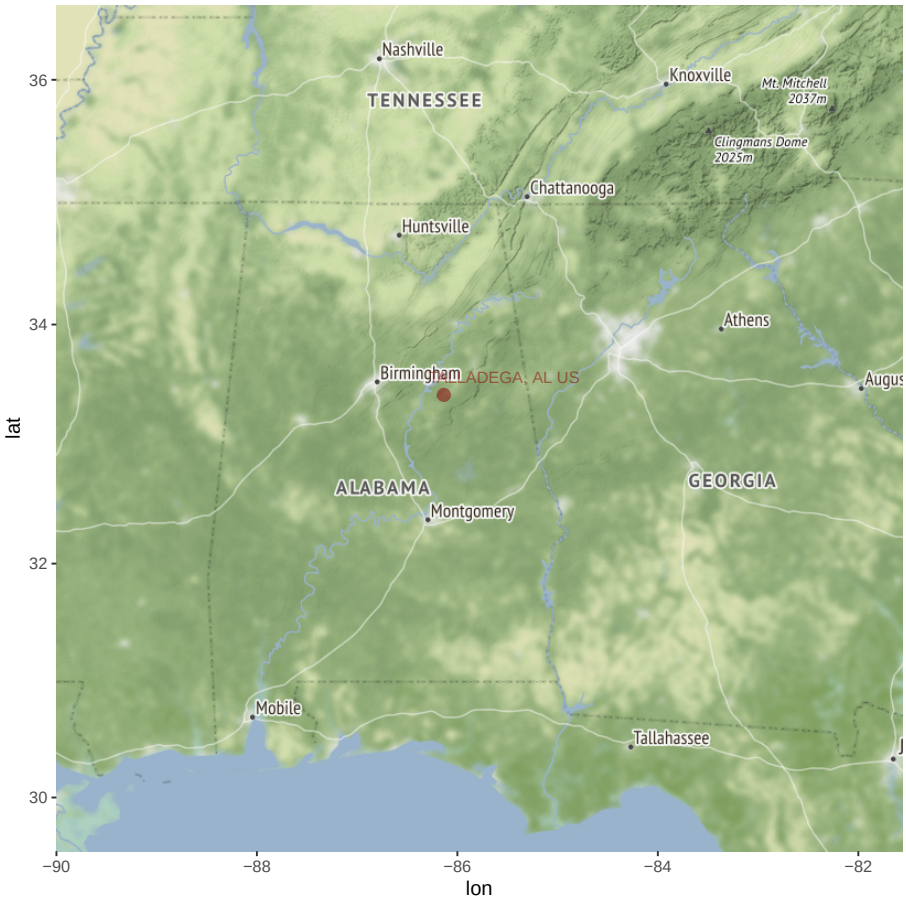
```
## Source :  http://tile.stamen.com/terrain/7/34/51.png
## Source :  http://tile.stamen.com/terrain/7/32/52.png
## Source :  http://tile.stamen.com/terrain/7/33/52.png
## Source :  http://tile.stamen.com/terrain/7/34/52.png

ggmap(myMap) + geom_point(aes(x = lon, y = lat),
      data = station.df, alpha = .5, color="darkred", size = 3) +
      geom_text(aes(x = lon, y = lat, label=Station),
      data = station.df, alpha = .5, color="darkred", size = 3,
      hjust=.1, vjust=-1)
```




```
#zoom = 11, scale = 2, maptype ='watercolor',

png(paste0("png//", fips$State, "-", stid, "-MAP.png"),
    width = 480, height = 480, units = "px",
    pointsize = 12, bg = "white")
```

```
## Error in paste0("png//", fips$State, "-", stid, "-MAP.png"):  object
'stid' not found
```

```
ggmap(myMap)+
geom_point(aes(x = lon, y = lat), data = station.df,
   alpha = .5, color="darkred", size = 3) +
   geom_text(aes(x = lon, y = lat, label=Station),
      data = station.df, alpha = .5, color="darkred",
      size = 3, hjust=.1, vjust=-1)
```




```
dev.off()
```

```
## null device
##           1
```

```r
#A) Download the main crime incident dataset

incidents= read.csv('https://raw.githubusercontent.com/lgellis/MiscTutorial/master/ggmap/i2

#B) Download the extra dataset with the most dangerous Seattle cities as per:

# https://housely.com/dangerous-neighborhoods-seattle/

n <- read.csv('https://raw.githubusercontent.com/lgellis/MiscTutorial/master/ggmap/n.csv', s

# Look at the data sets

dim(incidents)
head(incidents)
attach(incidents)

dim(n)
head(n)
attach(n)

# Create some color variables for graphing later
col1 = "#011f4b"; col2 = "#6497b1"; col3 = "#b3cde0"; col4 = "#CC0000"

#add year to the incidents data frame
incidents$ymd <-mdy_hms(Event.Clearance.Date)
incidents$year <- year(incidents$ymd)

#Create a more manageable data frame with only 2017 and 2018 data
i2 <- incidents %>% filter(year>=2017 & year<=2018)

#Only include complete cases
i2[complete.cases(i2), ]

#create a display label to the n data frame (dangerous neighbourhoods)
n$label <-paste(Rank, Location, sep="-")

##1) Create a map with all of the crime locations plotted.

p <- ggmap(get_googlemap(center = c(lon = -122.335167, lat = 47.608013),
                    zoom = 11, scale = 2,
                    maptype ='terrain',
                    color = 'color'))
p + geom_point(aes(x = Longitude, y = Latitude,  colour = Initial.Type.Group), data = i2, si
```

2.1.1 Functions to Collect and Clean GSOM

To collect the data, I used a short function, but the download time is painfully slow because only 1 year can be obtained at a time. Might want to get a work around for this at some point.

```
get_GSOM <- function(stid, datatype) {
   wtr<-list()  # create an empty list
   for (i in startyear:2021) {
      start_date <- paste0(i, "-01-01")
      end_date <- paste0(i, "-12-31")

#save data portion to the list (elements named for the year
      wtr[[as.character(i)]] <- ncdc(datasetid='GSOM',
         stationid=stid, datatypeid=datatype, startdate =
         start_date, enddate = end_date, limit=400)$data
   }
   #return the full list of data frames
   return(wtr)
}

stid = substr(GSOM_Longest$id, 7, 17)

get_GSOM2 <- function(stid){
   http.csv <- "https://www.ncei.noaa.gov/data/global-summary-of-the-month/access/"
   read.csv(paste(http.csv, stid, ".csv", sep=""))
}

# GSOM <- get_GSOM2(stid)
```

Functions to bin data into decades and scores.

```
floor_decade <- function(value){
   return(value - value %% 10) }
# Test Function
floor_decade(c(1883,1988))

## [1] 1880 1980

floor_score <- function(value){
   return(value - value %% 20) }

# Test Function
floor_score(c(1883,1893,1910, 1932,1940))

## [1] 1880 1880 1900 1920 1940
```

The function relies on two inputs, the station id and the measured parameter – TMAX and TMIN in this case. After that, the data needs to be clean up quite a bit.

Furthermore, I have converted units to Farenheit, which is not my favorite, but important for US consumption.

```
GSOM_TMAX <- get_GSOM(GSOM_Longest$id, 'TMAX')
GSOM_TMIN <- get_GSOM(GSOM_Longest$id, 'TMIN')
GSOM_PPT  <- get_GSOM(GSOM_Longest$id, 'PRCP')

# Bind the dataframes in the list
# together into one large dataframe

tbl_TMAX <- dplyr::bind_rows(GSOM_TMAX)
tbl_TMIN <- dplyr::bind_rows(GSOM_TMIN)
tbl_PPT <- dplyr::bind_rows(GSOM_PPT)

class(tbl_TMAX) # [1] "tbl_df"  "tbl" "data.frame"

## [1] "tbl_df"     "tbl"        "data.frame"

dfTbl_TMAX = as.data.frame(tbl_TMAX)
dfTbl_TMIN = as.data.frame(tbl_TMIN)
dfTbl_PPT = as.data.frame(tbl_PPT)

class(dfTbl_TMAX) # [1] "data.frame"

## [1] "data.frame"

dfTbl_TMAX$TMAX = dfTbl_TMAX$value*9/5+32
dfTbl_TMIN$TMIN = dfTbl_TMIN$value*9/5+32
dfTbl_PPT$PPT = dfTbl_PPT$value

dfTbl_TMAX$Date = as.Date(dfTbl_TMAX$date)
dfTbl_TMIN$Date = as.Date(dfTbl_TMIN$date)
dfTbl_PPT$Date = as.Date(dfTbl_PPT$date)

dfTbl_TMAX <- subset(dfTbl_TMAX, select=c(Date, station, TMAX))
dfTbl_TMIN <- subset(dfTbl_TMIN, select=c(Date, TMIN))
dfTbl_PPT <- subset(dfTbl_PPT, select=c(Date, PPT))

dfTbl_TMAX[1,]

##         Date           station  TMAX
## 1 1893-09-01 GHCND:USC00018024 83.75

GSOM <- merge(dfTbl_TMAX, dfTbl_TMIN, by="Date")
```

```
GSOM <- merge(GSOM, dfTbl_PPT, by="Date")

GSOM$Month = as.numeric(format(as.Date(GSOM$Date), format = "%m"))
GSOM$Year = as.numeric(format(as.Date(GSOM$Date), format = "%Y"))
GSOM$Decade = floor_decade(GSOM$Year)
GSOM$Score = floor_score(GSOM$Year)

#seq(min(GSOM£Year))

str(GSOM)

## 'data.frame': 1369 obs. of  9 variables:
##  $ Date   : Date, format: "1894-08-01" ...
##  $ station: chr  "GHCND:USC00018024" "GHCND:USC00018024" "GHCND:USC00018024" "GHCND:USC0
##  $ TMAX   : num  87.8 56.2 69 68.5 87.2 ...
##  $ TMIN   : num  69.8 34.5 49.5 45.7 59.7 ...
##  $ PPT    : num  213.4 19.6 99.8 127.6 7.6 ...
##  $ Month  : num  8 2 3 4 5 6 7 8 9 10 ...
##  $ Year   : num  1894 1898 1898 1898 1898 ...
##  $ Decade : num  1890 1890 1890 1890 1890 1890 1890 1890 1890 1890 ...
##  $ Score  : num  1880 1880 1880 1880 1880 1880 1880 1880 1880 1880 ...
```

2.1.2 Function to Evaluate Monthy Trends

Function to evaluate each month and determine if there is a trend. At somepoint, I'll have to the stats correcting for the autocorrelation.

Evaluate both TMAX and TMIN in GSOM by Year using MonthEvalStats() function.

```
MonthEvalStatsOLD <- 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

sumstats$Symbol = ""
sumstats$Symbol[sumstats$p_value < 0.05] = "*"
sumstats$Symbol[sumstats$p_value < 0.01] = "**"
sumstats$Symbol[sumstats$p_value < 0.001] = "***"
sumstats[,c(7,9)]
return(sumstats)
}

MonthEvalStats <- 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]))

} #end loop

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)
}

# test function
sumstats = MonthEvalStats(GSOM[500:4000,])
```

2.1.3 Function to report Probabilities

```
report_prob <-function(pvalue){
   if(pvalue > 0.05) return("> 0.05 (Not Significant)")
   if(pvalue < 0.05 & pvalue >= 0.001) return(
      paste("=", round(pvalue, 3), "(Statistically Significant)"))
   #if(pvalue < 0.01) print(round(pvalue, 4))
   if(pvalue < 0.001) return("< 0.001 (Statisically Significant)")
}

#test function
report_prob(0.0032)

report_prob2 <-function(lm){
   # lm=GSOM.lm
   if(anova(lm)$'Pr(>F)'[1] > 0.05){
         return("p-value > 0.05 (Not Significant)")
      }
   if(anova(lm)$'Pr(>F)'[1] < 0.05 &
      anova(lm)$'Pr(>F)'[1] >= 0.001){
         return(paste("Change ", round(coef(lm)[2]*356.25*100, 1),
         "/100 years, ", "p-value =", round(anova(lm)$'Pr(>F)'[1], 3),
         "(Statistically Significant)", sep=""))
      }
   if(anova(lm)$'Pr(>F)'[1] < 0.001) {
         return(paste("Change ", round(coef(lm)[2]*325.25*100, 1),
         "/100 years, ", "p-value < 0.001 (Statistically Significant)",
         sep=""))
   }
}

report_prob3 <-function(lm){
   #lm=Drought.run.lm
   temp = data.frame(change = NA, p_value=NA, note=NA)
   if(anova(lm)$'Pr(>F)'[1] > 0.05){
      temp[,1] <- "";
      temp[,2] <- "p-value > 0.05";
      temp[,3] <- "(Not Significant)"
      return(temp)
      }
   if(anova(lm)$'Pr(>F)'[1] < 0.05 &
      anova(lm)$'Pr(>F)'[1] >= 0.001){
      temp[,1] <- paste("Change: ",
         round(coef(lm)[2]*356.25*100, 1), "F/100 years", sep="");
      temp[,2] <- paste("p-value = ",
         round(anova(lm)$'Pr(>F)'[1], 3), sep="");
      temp[,3] <- " (Statistically Significant)"
```

```
      return(temp)
      }
   if(anova(lm)$'Pr(>F)'[1] < 0.001) {
      temp[,1] <- paste("Change: ", round(coef(lm)[2]*356.25*100, 1), "F/100 years", sep="")
      temp[,2] <- "p-value < 0.001";
      temp[,3] <- " (Statistically Significant)"
      return(temp)
   }
}
```

3 Communicating Long-term Weather Records

3.1 Complete Records vs. Post 1975 Trends

```
setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/Social_Media")
png(paste0("png//", fips$State, "-", stid, "-GSOMmonthly.png"), width = 480, height = 320, u
par(las=1, mfrow=c(1,1))
plot(TMAX~Date, GSOM, pch=20, cex=.5, col="grey", ylab="F", main=paste0(fips$State, "-", sti
GSOM.lm = lm(TMAX~Date, GSOM)
pred_dates <-data.frame(Date = GSOM$Date);
nrow(pred_dates);# pred_dates

## [1] 1369

#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="gray50")

# Post 1975
GSOM.lm = lm(TMAX~Date, GSOM[GSOM$Year>1975,])
pred_dates <-data.frame(Date = GSOM$Date[GSOM$Year>1975]);
nrow(pred_dates); #pred_dates

## [1] 510

#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="red")
lines(pred_dates$Date, as.numeric(ci[,2]), col="darkorange")
lines(pred_dates$Date, ci[,3], col="darkorange")
location_index = round(length(GSOM[GSOM$Year>1975,]$Date) * 0.99,0)
text(pred_dates$Date[location_index], ci[location_index,3],
```

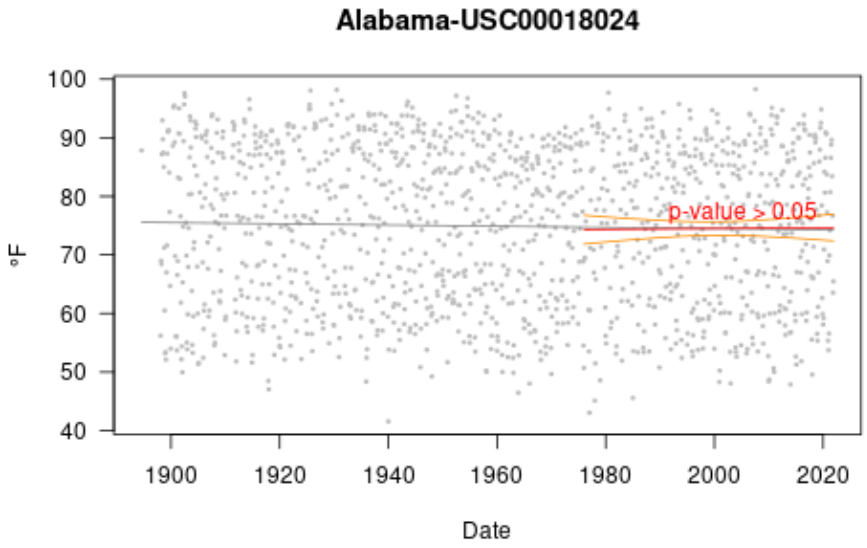


Figure 1: The climate trends from full record and post 1975 data. These data have a high level of variability due to seasonality effects that have not been filtered out.

```
     paste(report_prob3(GSOM.lm))[2], pos=2, cex=1.0, col="red")
#abline(coef(lm(TMAX~Date, GSOM)), col="black")
#abline(coef(lm(TMAX~Date, GSOM[GSOM$Year>1975,])), col="red")
dev.off()

## pdf
##   2
```

The noise in the data suggest that there is no trend, but it's tricky because the seasonal variation dominates the source of varition. Is there a way to "filter" out the seasonal effect?

3.2 Filtering Seasonal Effect

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 Februrary – 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 there is pretty good resolution.

Method 1 Filtering by Monthly Mean

```
TMAX.Monthly.means = aggregate(TMAX~Month, data=GSOM, mean)
names(TMAX.Monthly.means)=c("Month", "TMAXmean")
GSOM2 = merge(GSOM, TMAX.Monthly.means, by="Month")
GSOM2$TMAX.anom = GSOM2$TMAX - GSOM2$TMAXmean

TMIN.Monthly.means = aggregate(TMIN~Month, GSOM, mean)
names(TMIN.Monthly.means)=c("Month", "TMINmean")
GSOM2 = merge(GSOM2, TMIN.Monthly.means, by="Month")
GSOM2$TMIN.anom = GSOM2$TMIN - GSOM2$TMINmean

PPT.Monthly.means = aggregate(PPT~Month, GSOM, mean)
names(PPT.Monthly.means)=c("Month", "PPTmean")
GSOM2 = merge(GSOM2, PPT.Monthly.means, by="Month")
GSOM2$PPT.anom = GSOM2$PPT - GSOM2$PPTmean

# Sort by date
GSOM2 <- GSOM2[order(GSOM2$Date),]
```

```
png(paste0("png//", fips$State, "-", stid, "-GSOM-anomaly.png"),
    width = 480, height = 320, units = "px",
    pointsize = 12, bg = "white")
par(las=1, mfrow=c(1,1))
par(las=1)
plot(TMAX.anom~Date, GSOM2, pch=20, cex=.5,
     col="grey", ylab="Max. Temp (anomaly) F",
     main=paste0("Seasonally Filtered", fips$State, " (", stid, ")",
         report_prob3(GSOM.lm)[3]))
GSOM.lm = lm(TMAX.anom~Date, GSOM2)
pred_dates <-data.frame(Date = GSOM2$Date);
nrow(pred_dates); #pred_dates

## [1] 1369

#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="gray50")

ymax=max(GSOM2$TMAX.anom) - (max(GSOM2$TMAX.anom)-min(GSOM2$TMAX.anom))*.3
ymax2 <- ymax - (max(GSOM2$TMAX.anom)-min(GSOM2$TMAX.anom))*.1

location_index = round(length(GSOM2[GSOM2$Year>1975,]$Date) * 0.99,3)
text(pred_dates$Date[location_index], ymax,
     paste(report_prob3(GSOM.lm))[1], pos=2, cex=.9)
text(pred_dates$Date[location_index], ymax2,
```

```
     paste(report_prob3(GSOM.lm))[2], pos=2, cex=.9)

# Post 1975
GSOM.lm = lm(TMAX.anom~Date, GSOM2[GSOM2$Year>1975,])
pred_dates <-data.frame(Date = GSOM2$Date[GSOM2$Year>1975]);
nrow(pred_dates); #pred_dates

## [1] 510

#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="red")
lines(pred_dates$Date, as.numeric(ci[,2]), col="darkorange")
lines(pred_dates$Date, ci[,3], col="darkorange")

ymax=max(GSOM2$TMAX.anom) - (max(GSOM2$TMAX.anom)-min(GSOM2$TMAX.anom))*.7
ymax2 <- ymax - (max(GSOM2$TMAX.anom)-min(GSOM2$TMAX.anom))*.1

location_index = round(length(GSOM2[GSOM2$Year>1975,]$Date) * 0.99,0)
text(pred_dates$Date[location_index], ymax,
     paste(report_prob3(GSOM.lm))[1], pos=2, cex=.9, col="red")
text(pred_dates$Date[location_index], ymax2,
     paste(report_prob3(GSOM.lm))[2], pos=2, cex=.9, col="red")
dev.off()

## pdf
##   2
```

And to see what we created, see Figure 2.

3.2.1 Polynomial Filter

```
# 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
```

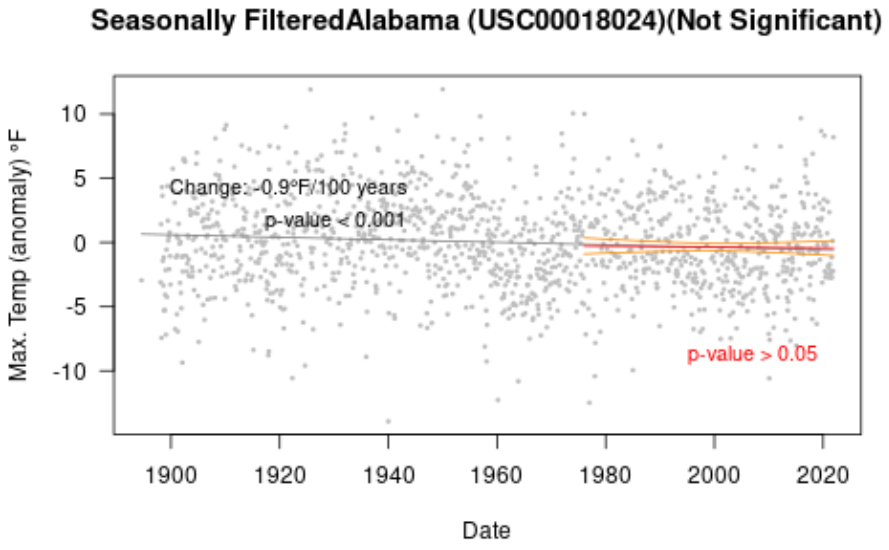


Figure 2: The changing in monthly temperature data.

3.3 Tables Temp Trends

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.

```
# Selecting Most Important Monthly Changes (TMAX overwrites)
#sumstats = MonthEvalStats(GSOM)

TMIN_Increase_month = with(sumstats[sumstats$Param=="TMIN",],
     Month[Slope==max(Slope, na.rm=T)])
TMIN_Decrease_month = with(sumstats[sumstats$Param=="TMIN",],
     Month[Slope==min(Slope, na.rm=T)])
TMAX_Increase_month = with(sumstats[sumstats$Param=="TMAX",],
     Month[Slope==max(Slope, na.rm=T)])
TMAX_Decrease_month = with(sumstats[sumstats$Param=="TMAX",],
     Month[Slope==min(Slope, na.rm=T)])
PPT_Increase_month = with(sumstats[sumstats$Param=="PPT",],
     Month[Slope==max(Slope, na.rm=T)])
PPT_Decrease_month = with(sumstats[sumstats$Param=="PPT",],
     Month[Slope==min(Slope, na.rm=T)])
```

3.3.1 Month with Biggest Changes

For this section, we'll look to see what months had the greatest changes for both TMIN and TMAX. By looking at significant slopes in whatever direction, we might learn if warming is really the dominiant pattern.

Table **??** summarizes the monthly trends for TMAX.

	Month	Slope100	r2	p_value	Symbol
1	1	-0.0193	0.08	0.0124	*
4	2	-0.0085	0.02	0.2033	
7	3	-0.0053	0.01	0.3335	
10	4	-0.0117	0.07	0.0217	*
13	5	-0.0045	0.02	0.2857	
16	6	0.0029	0.01	0.4076	
19	7	0.0047	0.03	0.1261	
22	8	0.0082	0.08	0.0136	*
25	9	-0.0008	0.00	0.8319	
28	10	-0.0019	0.00	0.7482	
31	11	-0.0068	0.02	0.2337	
34	12	-0.0009	0.00	0.8933	

Table 1: TMIN Trends

Table **??** summarizes the monthly trends for TMAX.

	Month	Slope100	r2	p_value	Symbol
2	1	-0.0094	0.02	0.1842	
5	2	-0.0098	0.03	0.1290	
8	3	0.0035	0.00	0.5561	
11	4	-0.0024	0.00	0.5682	
14	5	-0.0048	0.02	0.1847	
17	6	-0.0017	0.00	0.6589	
20	7	0.0012	0.00	0.7485	
23	8	-0.0013	0.00	0.7612	
26	9	0.0019	0.00	0.6612	
29	10	-0.0003	0.00	0.9322	
32	11	0.0009	0.00	0.8574	
35	12	0.0021	0.00	0.7278	

Table 2: TMAX Trends

PPT changes are tricky to capture and I'll have to keep working on this (Table **??**).

3.3.2 Defining TMAXmonth and TMINmonth

	Month	Slope100	r2	p_value	Symbol
3	1	-0.0332	0.00	0.7284	
6	2	0.1201	0.02	0.2631	
9	3	-0.0816	0.01	0.4707	
12	4	-0.0267	0.00	0.7942	
15	5	0.1415	0.03	0.1586	
18	6	0.0602	0.01	0.5136	
21	7	0.0177	0.00	0.8365	
24	8	0.0756	0.02	0.2859	
27	9	-0.0295	0.00	0.6989	
30	10	0.1426	0.05	0.0632	
33	11	0.0856	0.01	0.3628	
36	12	0.0658	0.01	0.4699	

Table 3: Caption PPT

```
TMINSlopeMax = max(abs(sumstats$Slope)[sumstats$Param=="TMIN"])
TMAXSlopeMax = max(abs(sumstats$Slope)[sumstats$Param=="TMAX"])

TMINmonthMax = as.numeric(subset(sumstats, select=Month, subset=abs(Slope)==TMINSlopeMax))

TMAXmonthMax = as.numeric(subset(sumstats, select=Month, subset=abs(Slope)==TMAXSlopeMax))
```

The greatest changes for Station GHCND:USC00018024

3.4 Functions to Collect and Clean CHCND

CHCND have been bias corrected...

```
GSOM_Longest$id

## [1] "GHCND:USC00018024"

stid = substr(GSOM_Longest$id, 7, 17)

CHCND.https <- "https://www.ncei.noaa.gov/data/global-historical-climatology-network-daily/a

get_CHCND <- function(stid) {
   #stid = "USC00013511"
   import <- read.csv(paste(CHCND.https, stid, ".csv", sep=""))
   selected = subset(import, select=c("DATE", "TMAX", "TMIN", "PRCP"))
   selected$TMAX = selected$TMAX/10*(9/5)+32
   selected$TMIN = selected$TMIN/10*(9/5)+32
   selected$Date = as.Date(selected$DATE)
   selected = selected[complete.cases(selected$TMAX),]
```

```
   selected
}

CHCND <- get_CHCND(stid); nrow(CHCND)

## [1] 44280

#str(CHCND)



CHCND$Month = as.numeric(format(as.Date(CHCND$Date), format = "%m"))
CHCND$Month.name = factor(format(as.Date(CHCND$Date), format = "%b"),
         levels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
                    "Aug", "Sep", "Oct", "Nov", "Dec"))
#levels(CHCND£Month.name)

range(CHCND$TMAX, na.rm=T)

## [1]  15.08 109.04

spread = sd(CHCND$TMAX, na.rm=T)*4
TMAX_mean = mean(CHCND$TMAX, na.rm=T)

CHCND$TMAX[complete.cases(CHCND$TMAX) &
              CHCND$TMAX > TMAX_mean+spread] <-NA
CHCND$TMAX[complete.cases(CHCND$TMAX) &
              CHCND$TMAX < TMAX_mean-spread] <-NA
range(CHCND$TMAX, na.rm=T)

## [1]  15.08 109.04

CHCND$Year = as.numeric(format(as.Date(CHCND$Date), format = "%Y"))
CHCND$YearDay = CHCND$Year + yday(CHCND$Date)/366
head(CHCND)

##            DATE  TMAX  TMIN PRCP       Date Month
## 1216 1893-09-01 89.96 75.92   NA 1893-09-01     9
## 1217 1893-09-02 86.00 84.02   NA 1893-09-02     9
## 1218 1893-09-03 86.00 87.08   NA 1893-09-03     9
## 1219 1893-09-04 87.08 84.92   NA 1893-09-04     9
## 1220 1893-09-05 78.98 87.08   NA 1893-09-05     9
## 1221 1893-09-06 86.00 75.92   NA 1893-09-06     9
##      Month.name Year  YearDay
## 1216        Sep 1893 1893.667
## 1217        Sep 1893 1893.669
## 1218        Sep 1893 1893.672
```

```
## 1219        Sep 1893 1893.675
## 1220        Sep 1893 1893.678
## 1221        Sep 1893 1893.680
```

3.5 Extreme Events

3.5.1 Functions for Rainfall Trends

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.

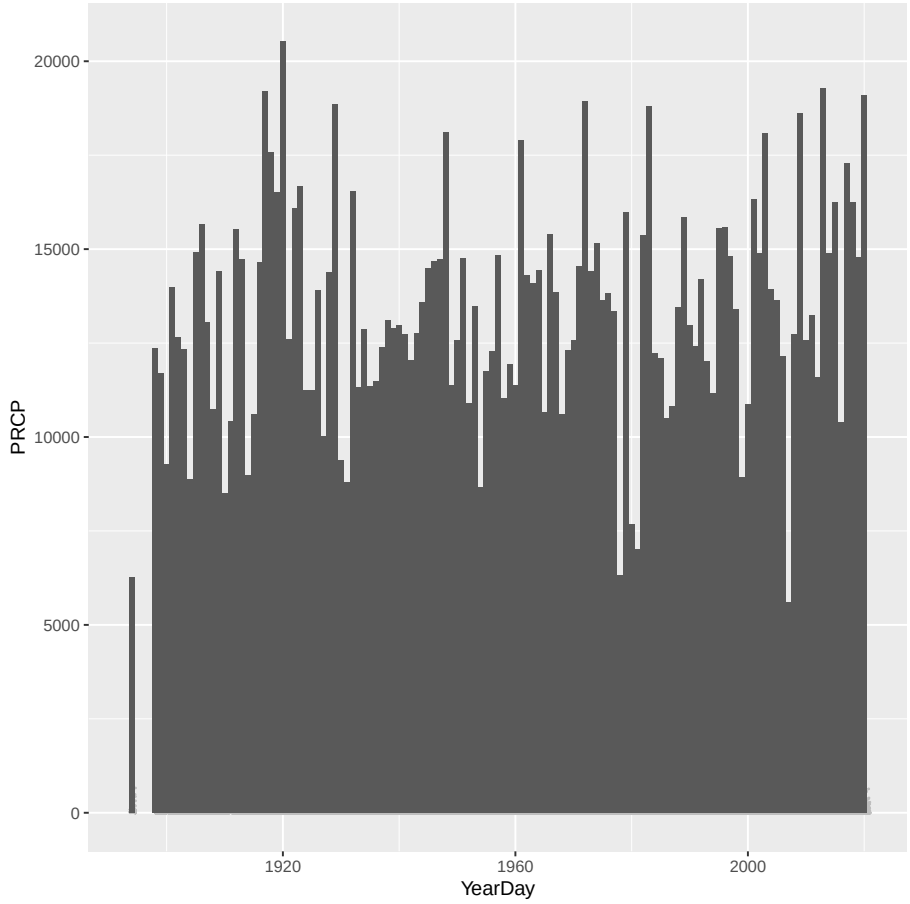
We can look for trends in monthly averages, number of days without rain (important in tropics), and/or extreme events based on daily or hourly data.

I don't know of a robust way to look at this for the entire globe.

```
CHCND$Season = "Winter"
CHCND$Season[CHCND$Month==4 | CHCND$Month==5 | CHCND$Month==6] = "Spring"
CHCND$Season[CHCND$Month==7 | CHCND$Month==8 | CHCND$Month==9] = "Summer"
CHCND$Season[CHCND$Month==10 | CHCND$Month==11 | CHCND$Month==12] = "Fall"


PRCP.Total = aggregate(PRCP~Year, data=CHCND, sum, na.rm=T)
PRCP.Season.Total = aggregate(PRCP~Season+Year, data=CHCND, sum, na.rm=T)
```

Rainfall...

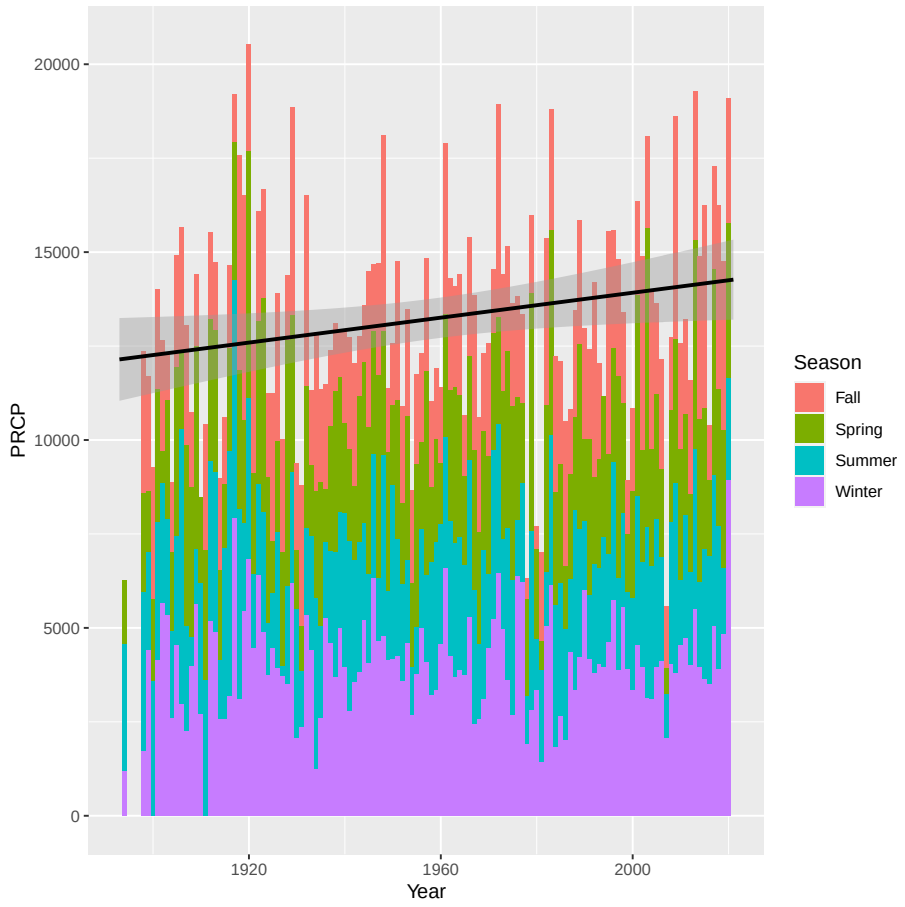
```
ggplot( ) +
   geom_point(data = CHCND,
      aes(y=PRCP, x=YearDay), size=.05, color="gray") +
   geom_bar(data = PRCP.Total,
      aes(x=Year, y=PRCP), stat="identity",
      position="identity") +
   xlim(min(CHCND$Year), max(CHCND$Year)-1)
```




```
   #ylab("Number of Extreme Temps") + # for the y axis label
```

Rainfall...

```
ggplot( ) +
   geom_bar(data = PRCP.Season.Total,
      aes(x=Year, y=PRCP, fill=Season), stat="identity") +
         xlim(min(CHCND$Year), max(CHCND$Year)-1) +
   #ylab("Number of Extreme Temps") + # for the y axis label
   geom_smooth(data = PRCP.Total,
      aes(y=PRCP, x=Year), method = "lm",
      se = T, color= "black")

## `geom_smooth()` using formula 'y ~ x'
```

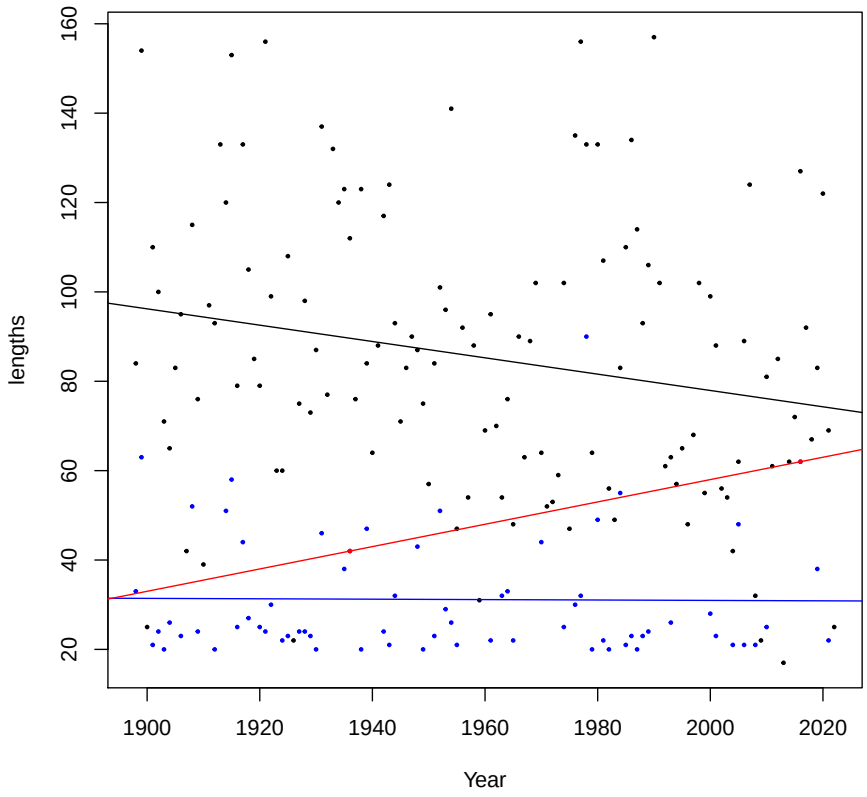



```
# + geom_smooth(data= PRCP.Season.Total, aes(x=Year, y = PRCP, color = Season, group=Season,
```

Days without rain...within a calendar year... bleed over between years isn't captured..

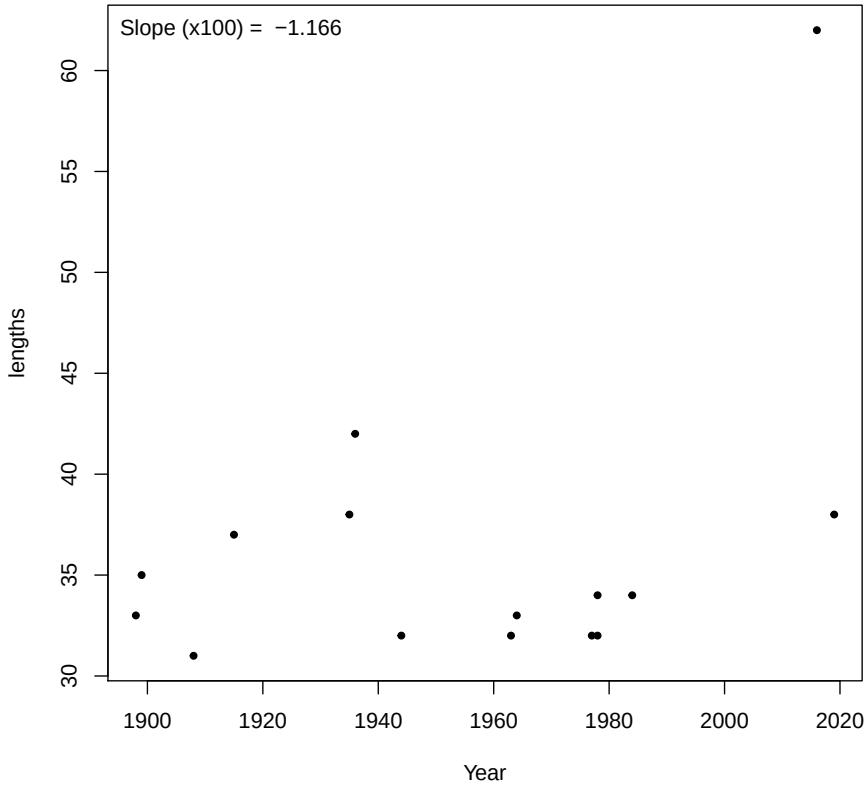
```
CHCND$PRCP.count = sequence(rle(CHCND$PRCP)$lengths)
Drought.run.temp <- data.frame(Year = NA, lengths=NA, values=NA)
for(i in min(CHCND$Year):max(CHCND$Year)){
   # print(i)
   run.length = rle(CHCND[CHCND$Year==i,]$PRCP)
   run.length.df = data.frame(Year = rep(i, length(run.length$values)),
         lengths = run.length$lengths,
         values = run.length$values)

   Drought.run.temp <- rbind(Drought.run.temp,
         run.length.df[run.length.df$values==0,])
```

```
}
Drought.run <- Drought.run.temp[-1,]
str(Drought.run)

## 'data.frame': 7732 obs. of  3 variables:
##  $ Year   : int  NA NA NA NA NA NA NA NA NA NA ...
##  $ lengths: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ values : int  NA NA NA NA NA NA NA NA NA NA ...

names(Drought.run)

## [1] "Year"    "lengths" "values"

# What is a drought 10 days, 20 days, 40 days?

Drought.run.10 = aggregate(lengths~Year,
   data=Drought.run[Drought.run$lengths>=10,], sum)
Drought.run.20 = aggregate(lengths~Year,
   data=Drought.run[Drought.run$lengths>=20,], sum)
Drought.run.40 = aggregate(lengths~Year,
   data=Drought.run[Drought.run$lengths>=40,], sum)
Drought.run.100 = aggregate(lengths~Year,
   data=Drought.run[Drought.run$lengths>=100,], sum)

## Error in aggregate.data.frame(mf[1L], mf[-1L], FUN = FUN, ...):
no rows to aggregate

plot(lengths~Year, Drought.run.10, pch=20, cex=.5)
points(lengths~Year, Drought.run.20, pch=20, col="blue", cex=.5)
points(lengths~Year, Drought.run.40, pch=20, col="red", cex=.5)
points(lengths~Year, Drought.run.100, pch=20, col="purple", cex=.5)

## Error in eval(m$data, eframe):  object 'Drought.run.100' not found

abline(lm(lengths~Year, Drought.run.10))
abline(lm(lengths~Year, Drought.run.20), col="blue")
abline(lm(lengths~Year, Drought.run.40), col="red")
```




```
abline(lm(lengths~Year, Drought.run.100), col="purple")

## Error in is.data.frame(data):  object 'Drought.run.100' not found

summary(lm(lengths~Year, Drought.run.100))

## Error in is.data.frame(data):  object 'Drought.run.100' not found

plot(lengths~Year, Drought.run[Drought.run$lengths>30,], pch=20)
plot(lengths~Year, Drought.run[Drought.run$lengths>30,], pch=20)


Drought.run.lm <- lm(lengths~Year, Drought.run[Drought.run$lengths>10,])
summary(Drought.run.lm)

##
## Call:
```

```
## lm(formula = lengths ~ Year, data = Drought.run[Drought.run$lengths >
##     10, ])
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -5.095 -3.585 -1.699  1.715 47.269
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 38.231403  11.717890   3.263  0.00117 **
## Year        -0.011657   0.005984  -1.948  0.05189 .
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.257 on 598 degrees of freedom
##   (954 observations deleted due to missingness)
## Multiple R-squared:  0.006305,Adjusted R-squared:  0.004643
## F-statistic: 3.794 on 1 and 598 DF,  p-value: 0.05189

text(min(Drought.run$Year, na.rm=T), max(Drought.run$lengths, na.rm=T),
     paste("Slope (x100) = ", round(coef(Drought.run.lm)[2]*100, 3)), pos=4)
```




```
#plot(PRCP.count ~ Year, data=CHCND[CHCND£PRCP==0,])
```

Probability Distributions by decade...

```
CHCND$Decade <- floor_decade(CHCND$Year)

PRCP.Decade <- aggregate(PRCP~Month+Decade, data=CHCND, sum)
head(PRCP.Decade)

##   Month Decade PRCP
## 1     1   1890 1504
## 2     2   1890 1772
## 3     3   1890 4060
## 4     4   1890 2966
## 5     5   1890  982
## 6     6   1890 2044
```

```
x <- PRCP.Decade$PRCP[PRCP.Decade$Decade==1900]
df <- approxfun(density(x))
plot(1:12, density(x))

## Error in xy.coords(x, y, xlabel, ylabel, log):  'x' and 'y' lengths
differ

xnew <- c(0.45,1.84,2.3)
points(xnew,df(xnew),col=2)

## Error in plot.xy(xy.coords(x, y), type = type, ...):  plot.new has
not been called yet
```

```
CHCND$Score <- floor_score(CHCND$Year)
```

3.6 Determine Record Setting Temperatures

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.

Crating a graphic of the results...

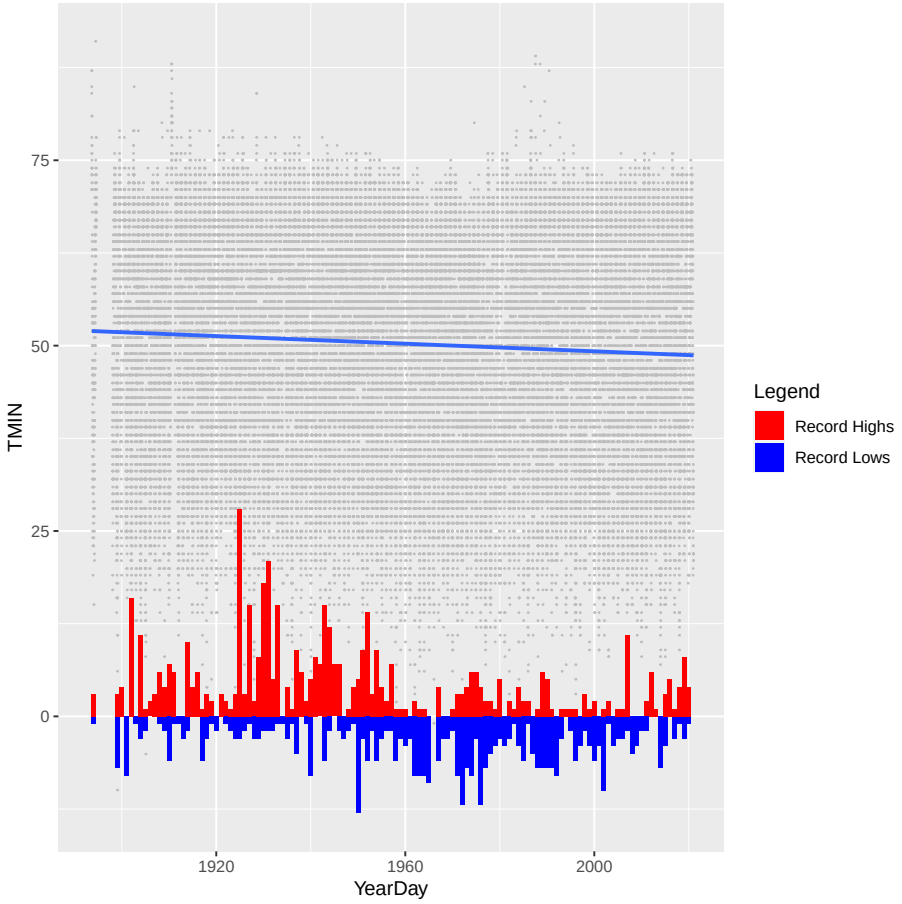
3.6.1 Number of Days with Records per year

This is a common way to communicate temperatures changes. I suspect we have a better sense of change when we notice "extreme" events...

```
names(CHCND)

##  [1] "DATE"       "TMAX"       "TMIN"       "PRCP"
##  [5] "Date"       "Month"      "Month.name" "Year"
##  [9] "YearDay"    "Season"     "PRCP.count" "Decade"
## [13] "Score"      "mmdd"       "minTMIN"    "maxTMAX"

minTMIN.length = aggregate(minTMIN~Year, data=CHCND, length)
minTMIN.length$group <- "Record Lows"
names(minTMIN.length) <- c("Year", "Num", "Group")
minTMIN.length$Num = -minTMIN.length$Num

maxTMAX.length = aggregate(maxTMAX~Year, data=CHCND, length);
maxTMAX.length$group <- "Record Highs"
names(maxTMAX.length) <- c("Year", "Num", "Group")

records = rbind(minTMIN.length, maxTMAX.length); # records
```

```r
ggplot( ) +
   geom_point(data = CHCND, aes(y=TMIN, x=YearDay),
      size=.05, color="gray") +
   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")) +
   geom_smooth(data = CHCND, aes(y=TMIN, x=YearDay), method = "lm", se = FALSE)
```

```
## `geom_smooth()` using formula 'y ~ x'
## Warning:  Removed 878 rows containing non-finite values
## (stat_smooth).
## Warning:  Removed 878 rows containing missing values (geom_point).
## Warning:  Removed 3 rows containing missing values (geom_bar).
```




```
#scale_y_continuous(
    # Features of the first axis
    # name = "Temperature (F )",
    # Add a second axis and specify its features
    # sec.axis = sec_axis(~.*Num, name="Number of Extreme Temps")
  #)
```

3.7 Iterate TMAX vs. Month Boxplots

4 Static Plot Results

4.1 Four Plots

To test the code, I have created graphics that can then be used in the animation process, i.e. try to create code that doesn't get too complicated and then fail!

```
## pdf
##   2
```

4.2 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.

First, TMIN and TMAX and change point analysis...
https://cran.r-project.org/web/packages/mcp/readme/README.html

```
library(mcp)
library(rlang)

# Simulate
set.seed(42)  # I always use 42; no fiddling
df = data.frame(
  x = 1:100,
  y = c(rnorm(30, 2), rnorm(40, 0), rnorm(30, 1))
)

# Plot it
plot(df)
abline(v = c(30, 70), col="red")

model = list(y~1, 1~1, 1~1)  # three intercept-only segments
fit_mcp = mcp(model, data = df, par_x = "x")

summary(fit_mcp)

library(patchwork)
plot(fit_mcp) + plot_pars(fit_mcp, pars = c("cp_1", "cp_2"), type = "dens_overlay")

model = list(
  price ~ 1 + ar(2),
  ~ 0 + time + ar(1)
)
ex = mcp_example("ar")
fit = mcp(model, ex$data)
summary(fit)
```

Let's create a figure that simplifies the narrative, if we can!

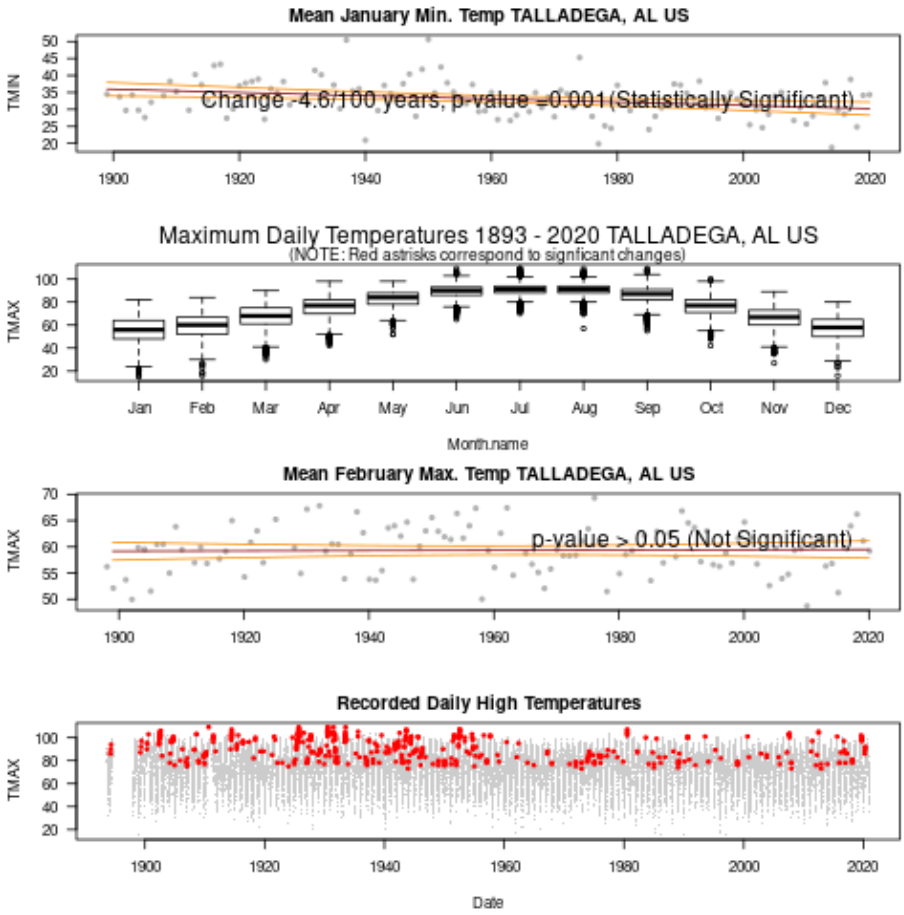


Figure 3: Climate can be analyzed using several types of lenses. In this case, we have analyzed show the months with the greatest changes. The first figure is monthly average of TMINs (daily low temperatures) with a best fit line. The second figure shows the monthly TMAX range and asterisks indicate singificant changes over the station record and the third figure is the trend for these TMAXs over time and includes the best fit line. The final figure shows the daily temperatures that have been the highest on record (in red). In some cases, climate change has created more records in the recent decades, while other stations seem don't show that trend.

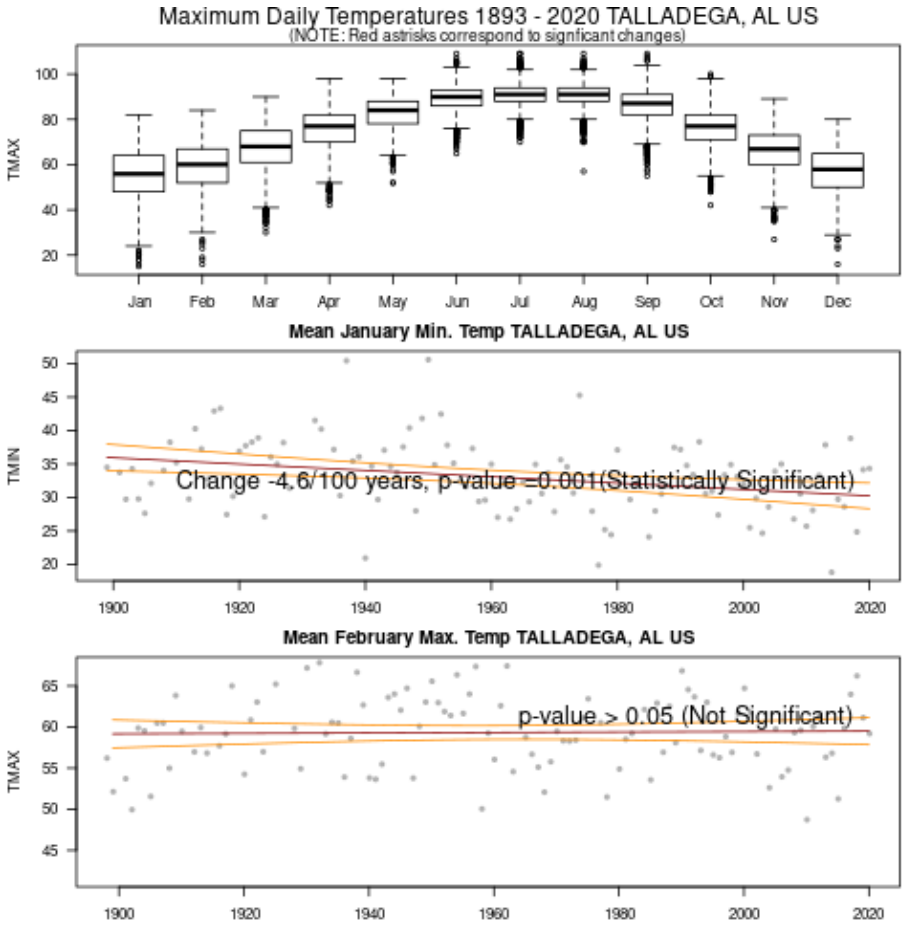


Figure 4: Keep it simple stupid!

```
## pdf
##   2
```

4.3 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 approrpriate with a Chi-Square test), or we can use the data to create a emperical distribution, which is my favored approach.

I started with decade bins, but used 20 years bins (scores) to simplify the graphics while keeping a pretty good temporal resolution.

```
library(wesanderson)

h.ramp <- rev(heat.colors(length(unique(GSOM2$Score))+1))[-1]
h.ramp <- wes_palette("Zissou1", length(unique(GSOM2$Score)),
      type = "continuous")[1:length(unique(GSOM2$Score))]
#TMAX.anomaly.Score = aggregate(TMAX.anom ~ Score, GSOM2, mean)
#TMAX.sd.anomaly.Score = aggregate(TMAX.anom ~ Score, GSOM2, sd)


# I hate list!
TMAX.anomaly.list = aggregate(TMAX.anom ~ Score, GSOM2,
   FUN = function(x) c(mean = mean(x), sd = sd(x)))
TMIN.anomaly.list = aggregate(TMIN.anom ~ Score, GSOM2,
   FUN = function(x) c(mean = mean(x), sd = sd(x)))
PPT.anomaly.list = aggregate(PPT.anom ~ Score, GSOM2,
   FUN = function(x) c(mean = mean(x), sd = sd(x)))


setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/Social_Media")
png(paste0("png//", fips$State, "-", stid, "-GSOM-normalPDF.png"),
    width = 480, height = 480, units = "px", pointsize = 12, bg = "white")
# TMIN
par(mfrow=c(2,2), las=1)
Anom.x = seq(min(GSOM2$TMIN.anom), max(GSOM2$TMIN.anom),by=.1)
Anom.y = max(dnorm(Anom.x,
   mean=TMIN.anomaly.list$TMIN.anom[1,1],
   sd=TMIN.anomaly.list$TMIN.anom[1,2]))*1.2

plot(Anom.x, dnorm(Anom.x,
   mean=TMIN.anomaly.list$TMIN.anom[1, 1],
   sd=TMIN.anomaly.list$TMIN.anom[1, 2]),
   ty="l", col=h.ramp[1], ylim=c(0, Anom.y),
   ylab="Density", xlab="TMIN Anomaly (F)", main="")

abline(v=TMIN.anomaly.list$TMIN.anom[1,1], col=h.ramp[1], lwd=2)
for(i in 2:nrow(TMIN.anomaly.list)){
lines(Anom.x, dnorm(Anom.x,
   mean=TMIN.anomaly.list$TMIN.anom[i, 1],
   sd=TMIN.anomaly.list$TMIN.anom[i, 2]), col=h.ramp[i])
}
abline(v=TMIN.anomaly.list$TMIN.anom[i,1], col=h.ramp[i], lwd=2)
Delta = TMIN.anomaly.list$TMIN.anom[i,1] - TMIN.anomaly.list$TMIN.anom[1,1]

text(TMIN.anomaly.list$TMIN.anom[i,1], Anom.y*.9,
   paste0("Change ", round(Delta, 1), "F"), pos=4)
```

```
mtext(paste0(fips$State, " (", GSOM_Longest$name, ")"), side=3, line=2)

# TMAX
Anom.x = seq(min(GSOM2$TMAX.anom), max(GSOM2$TMAX.anom),by=.1)
Anom.y = max(dnorm(Anom.x,
   mean=TMAX.anomaly.list$TMAX.anom[1,1],
   sd=TMAX.anomaly.list$TMAX.anom[1,2]))*1.2

plot(Anom.x, dnorm(Anom.x,
   mean=TMAX.anomaly.list$TMAX.anom[1, 1],
   sd=TMAX.anomaly.list$TMAX.anom[1, 2]),
   ty="l", col=h.ramp[1], ylim=c(0, Anom.y),
   ylab="Density", xlab="TMAX Anomaly (F)")

abline(v=TMAX.anomaly.list$TMAX.anom[1,1], col=h.ramp[1], lwd=2)
for(i in 2:nrow(TMAX.anomaly.list)){
lines(Anom.x, dnorm(Anom.x,
   mean=TMAX.anomaly.list$TMAX.anom[i, 1],
   sd=TMAX.anomaly.list$TMAX.anom[i, 2]), col=h.ramp[i])
}
abline(v=TMAX.anomaly.list$TMAX.anom[i,1], col=h.ramp[i], lwd=2)
Delta = TMAX.anomaly.list$TMAX.anom[i,1] - TMAX.anomaly.list$TMAX.anom[1,1]

text(TMAX.anomaly.list$TMAX.anom[i,1], Anom.y*.9,
   paste0("Change ", round(Delta, 1), "F"), pos=4)

# PPT
Anom.x = seq(min(GSOM2$PPT.anom), max(GSOM2$PPT.anom),by=.1)
Anom.y = max(dnorm(Anom.x,
   mean=PPT.anomaly.list$PPT.anom[1,1],
   sd=PPT.anomaly.list$PPT.anom[1,2]))*1.2

plot(Anom.x, dnorm(Anom.x,
   mean=PPT.anomaly.list$PPT.anom[1, 1],
   sd=PPT.anomaly.list$PPT.anom[1, 2]),
   ty="l", col=h.ramp[1], ylim=c(0, Anom.y),
   ylab="Density", xlab="PPT Anomaly")

abline(v=PPT.anomaly.list$PPT.anom[1,1], col=h.ramp[1], lwd=2)
for(i in 2:nrow(PPT.anomaly.list)){
lines(Anom.x, dnorm(Anom.x,
   mean=PPT.anomaly.list$PPT.anom[i, 1],
   sd=PPT.anomaly.list$PPT.anom[i, 2]), col=h.ramp[i])
}
abline(v=PPT.anomaly.list$PPT.anom[i,1], col=h.ramp[i], lwd=2)
```

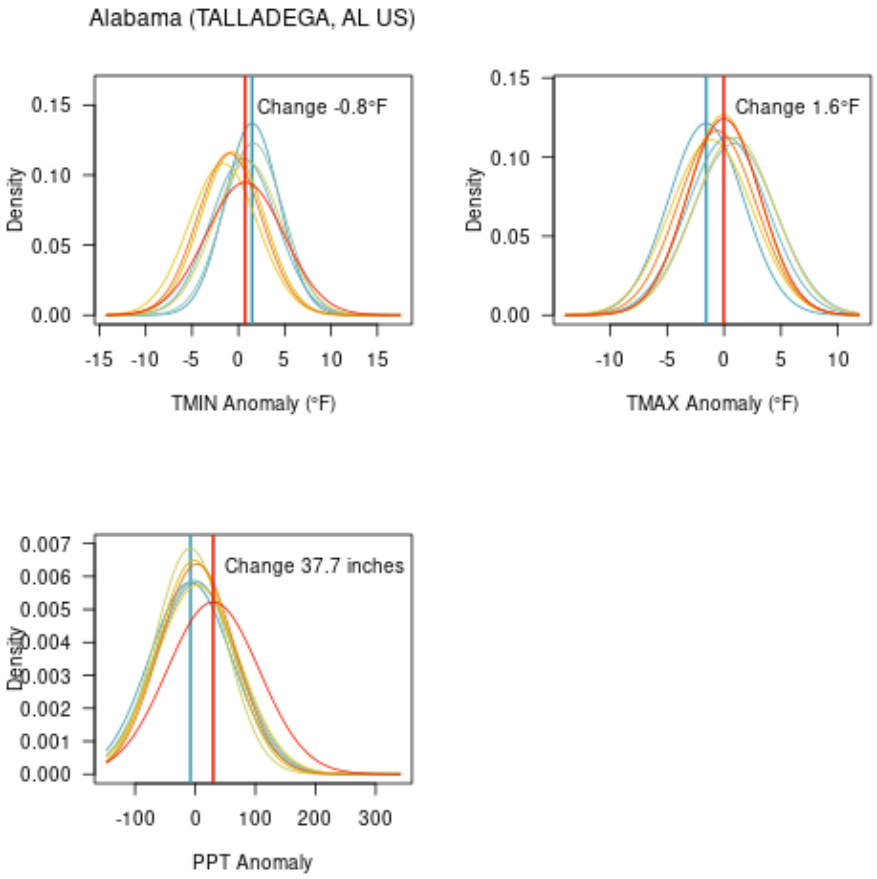


Figure 5: The changing in monthly temperature data, assuming a normal probability distribution.

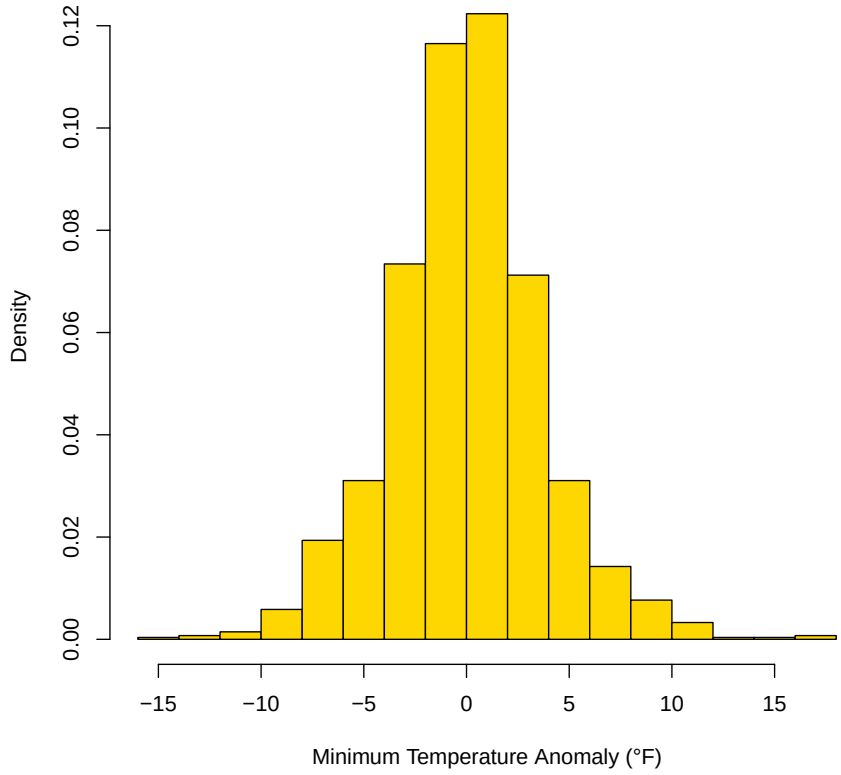
```
Delta = PPT.anomaly.list$PPT.anom[i,1] - PPT.anomaly.list$PPT.anom[1,1]

text(PPT.anomaly.list$PPT.anom[i,1], Anom.y*.9,
   paste0("Change ", round(Delta, 1), " inches"), pos=4)

dev.off()

## pdf
##   2
```

This figure is pretty effective, but still needs work.

4.4 Using library densEstBayes

Now, I used a screen split to look at the distribution of the temperate anomolies. First, we look at a simple histogram of the entire dataset.

```
par(mfrow=c(1,1))
hist(GSOM2$TMIN.anom, col = "gold",
     main = "", probability = TRUE,
     xlab = "Minimum Temperature Anomaly (F)")
```



The data center around zero, as expected, but are these normally distributed?

For TMAX there is a 0.031 probability that the distribution is the same as the normal distribution. For TMIN there is a 0 probability that the distribution is the same as the normal distribution. For PPT is a 0 probability that the distribution is the same as the normal distribution.

```
if(shapiro.test(GSOM2$TMAX.anom)$p.value<.05 |
   shapiro.test(GSOM2$TMIN.anom)$p.value<.05 |
   shapiro.test(GSOM2$PPT.anom)$p.value<.05) text="to avoid " else text="to use"
```

These values suggest that there is good reason to avoid the normal probability distribution.

Next we use a function to estimate the probability distribution using a markof chain the creates an estimated probability distribution. This doesn't always work when the distribution is not even and their only 10 years of data per slot. I suspect, I should make this by every 20 years. Plus that will go way faster and I think the data visualization will be more robust.

```
setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/Social_Media")

if(!file.exists(paste0("png//", fips$State, "-", stid, "-GSOM-estDensity.png"))){
   print("Creating estimated density distribution")

png(paste0("png//", fips$State, "-", stid, "-GSOM-estDensity.png"),
    width = 480, height = 320, units = "px", pointsize = 12, bg = "white")

# Split Screen TMAX Legend TMIN
# screen with values for left, right, bottom, and top.
split.screen(rbind(c(.01, 0.99, 0.86, 0.95),
                   c(0.01, 0.45, 0.01, 0.85),
                   c(0.45, 0.55, 0.01, 0.85),
                   c(0.55, 0.99, 0.01, 0.85)))

screen(1)
par(mar=c(0,0,0,0))
plot(NA, xaxt='n',yaxt='n',bty='n',ylab='',xlab='', xlim=c(0,10), ylim=c(0, 10))
mtext(paste0(fips$State, " (", GSOM_Longest$name, ")"), side=3, line=-1, cex=1.4)

screen(2)

# Determine xg (range)
dest <- densEstBayes(GSOM2$TMIN.anom, method = "NUTS"); dest$range.x

control = densEstBayes.control(range.x = dest$range.x,
      numBins = 401,
      numBasis = 50, sigmabeta = 1e5, ssigma = 1000,
      convToler = 1e-5, maxIter = 500, nWarm = NULL,
      nKept = NULL, nThin = 1, msgCode = 1)

#destSMFVB <- densEstBayes(GSOM2£TMIN.anom, method = "SMFVB", control = control)
#plot(destSMFVB, plotIt=T, xlab = "TMIN", main = "", setCol=h.ramp[i])
```

```
par(las=1, mar=c(4, 4, 0, 0) + 0.1)
for(i in 1:length(unique(GSOM2$Score))){
   # i = 13
   GSOM2sub = GSOM2[GSOM2$Score==sort(unique(GSOM2$Score))[i],]
   dest <- densEstBayes(GSOM2sub$TMIN.anom, method = "NUTS", control = control)
   xg = plot(dest, plotIt=FALSE)$xg
   densEstg = plot(dest, plotIt=FALSE)$densEstg

   if(i==1) plot(0, type = "n", bty = "l",
         xlim=range(xg), ylim=c(0,0.25),
         xlab = "TMIN anomaly (F)", main = "", ylab="Density")
   lines(xg, densEstg, col=h.ramp[i])
rug(jitter(GSOM2sub$TMIN.anom,amount = 0.2), col=h.ramp[i])
}

screen(3)
par(mar=c(0,0,1,0))
plot(NA,xaxt='n',yaxt='n',bty='n',ylab='',xlab='', xlim=c(0,10), ylim=c(0,10))
#
legend("topright", inset=c(0,0), bg="transparent", bty="n",
       legend=unique(GSOM2$Score),
       fill=h.ramp, horiz=FALSE, cex=0.85)

screen(4)
par(las=1, mar=c(4, 4, 0, 0) +0.1)
# Determine xg (range)
dest <- densEstBayes(GSOM2$TMAX.anom, method = "NUTS"); dest$range.x

control = densEstBayes.control(range.x = dest$range.x,
      numBins = 401,
      numBasis = 50, sigmabeta = 1e5, ssigma = 1000,
      convToler = 1e-5, maxIter = 500, nWarm = NULL,
      nKept = NULL, nThin = 1, msgCode = 1)

for(i in 1:length(unique(GSOM2$Score))){
   # i = 13
   GSOM2sub = GSOM2[GSOM2$Score==sort(unique(GSOM2$Score))[i],]
   dest <- densEstBayes(GSOM2sub$TMAX.anom, method = "NUTS", control = control)
   xg = plot(dest, plotIt=FALSE)$xg
   densEstg = plot(dest, plotIt=FALSE)$densEstg

   if(i==1) plot(0, type = "n", bty = "l",
         xlim=range(xg), ylim=c(0,.25),
         xlab = "TMAX anomaly (F)", main = "", ylab="Density")
   lines(xg, densEstg, col=h.ramp[i])
```

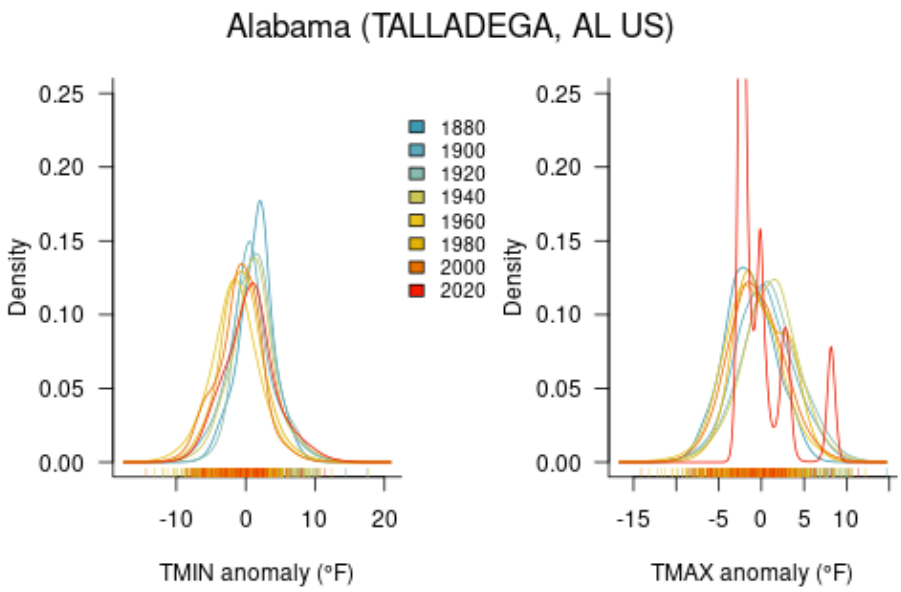


Figure 6: The changing in monthly temperature data.

```
rug(jitter(GSOM2sub$TMIN.anom,amount = 0.2), col=h.ramp[i])
}
#rug(jitter(GSOM2sub£TMIN.anom,amount = 0.2), col=h.ramp[i])

close.screen(all.screens = TRUE)
dev.off()
} else {
   print("Skipping estimated density distribution chunk")}
```

The process to create these figures is very time consuming, so in general, I need to come up with an if then statement to avoid creating these everytime!

5 Animated GIFs

So far, this creates a gif file, but I haven't been able get the gif in the pdf directly yet. I will need an additional package or create separate png that are combined. For now, we'll create a gif file to be used in separate documents.

5.1 Probability Distributions

```
setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/docs/")
if(!file.exists(paste0("Climate_gifs/",
   fips$State, "-", stid, "_GSOM-normalPDF.gif"))){
   print("Creating animated normal probability ")

# Define an image_graph size
img <- image_graph(600, 480, res = 96)

# START ----------------------------------------

ylim_new=NA
for(i in 1:length(unique(GSOM2$Decade)))
   {
# i = 9
decade=(unique(GSOM2$Decade))[order(unique(GSOM2$Decade))==i]

GSOM2sub <- GSOM2[GSOM2$Decade==decade,]
h.ramp <- rev(heat.colors(length(unique(GSOM2$Decade))+1))[-1]

# Determine Stats for PDFs
TMAX.mean.anomaly.decade = aggregate(TMAX.anom ~ Decade, GSOM2sub, mean)
TMAX.sd.anomaly.decade = aggregate(TMAX.anom ~ Decade, GSOM2sub, sd)
names(TMAX.sd.anomaly.decade)=c("Decade", "TMAX.sd.anom")
TMIN.mean.anomaly.decade = aggregate(TMIN.anom ~ Decade, GSOM2sub, mean)
TMIN.sd.anomaly.decade = aggregate(TMIN.anom ~ Decade, GSOM2sub, sd)
names(TMIN.sd.anomaly.decade)=c("Decade", "TMIN.sd.anom")

TMAX.temp = merge(TMAX.mean.anomaly.decade, TMAX.sd.anomaly.decade, by="Decade")

TMIN.temp = merge(TMIN.mean.anomaly.decade, TMIN.sd.anomaly.decade, by="Decade")

GSOM.Monthly.Anom.mean.sd = merge(TMAX.temp, TMIN.temp, by="Decade")

par(las=1, mfrow=c(1,2), mar= c(4, 4, 2, 1) + 0.1)

Anom.x = seq(min(GSOM2$TMAX.anom), max(GSOM2$TMAX.anom), by=.1)
plot(Anom.x, dnorm(Anom.x, mean=GSOM.Monthly.Anom.mean.sd$TMAX.anom[1],
   sd=GSOM.Monthly.Anom.mean.sd$TMAX.sd.anom[1]), ty="l", col=h.ramp[i], ylab="Density", xla
abline(v=mean(GSOM2$TMAX.anom[GSOM2$Decade==min(GSOM$Decade)]))
mtext(paste0(fips$State, " ", decade), side=3)

Anom.x = seq(min(GSOM2$TMIN.anom), max(GSOM2$TMIN.anom), by=.1)
plot(Anom.x, dnorm(Anom.x, mean=GSOM.Monthly.Anom.mean.sd$TMIN.anom[1],
   sd=GSOM.Monthly.Anom.mean.sd$TMIN.sd.anom[1]), ty="l", col=h.ramp[i], ylab="Density", xla
abline(v=mean(GSOM2$TMIN.anom[GSOM2$Decade==min(GSOM$Decade)]))
```

```
mtext(paste0(fips$State, " ", decade), side=3)
}

par(las=1, mfrow=c(1,2), mar= c(4, 4, 2, 1) + 0.1)

TMAX.anomaly.decade = aggregate(TMAX.anom ~ Decade, GSOM2,
   FUN = function(x) c(mean = mean(x), sd = sd(x)))
TMIN.anomaly.decade = aggregate(TMIN.anom ~ Decade, GSOM2,
   FUN = function(x) c(mean = mean(x), sd = sd(x)))


Anom.x = seq(min(GSOM2$TMIN.anom), max(GSOM2$TMIN.anom), by=.1)
plot(Anom.x, dnorm(Anom.x, mean=TMIN.anomaly.decade$TMIN.anom[[1,1]],
   sd=TMIN.anomaly.decade$TMIN.anom[[1,2]]), ty="l", col=h.ramp[1], ylab="Density", xlab="TM
mtext(paste0(fips$State, " ", decade), side=3)
for(i in 2:nrow(TMIN.anomaly.decade)){
lines(Anom.x, dnorm(Anom.x, mean=TMIN.anomaly.decade$TMIN.anom[[i,1]], sd=TMIN.anomaly.decad
}
abline(v=mean(GSOM2$TMIN.anom[GSOM2$Decade==min(GSOM$Decade)]), col="blue")
abline(v=mean(GSOM2$TMIN.anom[GSOM2$Decade==max(GSOM$Decade)]), col="red")

Anom.x = seq(min(GSOM2$TMAX.anom), max(GSOM2$TMAX.anom), by=.1)
plot(Anom.x, dnorm(Anom.x, mean=TMAX.anomaly.decade$TMAX.anom[[1,1]],
   sd=TMAX.anomaly.decade$TMAX.anom[[1,2]]), ty="l", col=h.ramp[1], ylab="Density", xlab="TM
mtext(paste0(fips$State, " ", decade), side=3)
for(i in 2:nrow(TMAX.anomaly.decade)){
lines(Anom.x, dnorm(Anom.x, mean=TMAX.anomaly.decade$TMAX.anom[[i,1]], sd=TMAX.anomaly.decad
}
abline(v=mean(GSOM2$TMAX.anom[GSOM2$Decade==min(GSOM$Decade)]), col="blue")
abline(v=mean(GSOM2$TMAX.anom[GSOM2$Decade==max(GSOM$Decade)]), col="red")


# END ---------------------------------------------------
dev.off()

GSOM_animation <- image_animate(img, fps = 1, loop=2, optimize = TRUE)
#print(GSOM_animation)
setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/docs/")
image_write(GSOM_animation, paste("Climate_gifs/",
   fips$State, "-", stid, "_GSOM-normalPDF.gif", sep=""))

} else {
   print("Skipping animated normal distribution chunk")}
```

The file is saved in the main directory.

5.2 4 Weather Trend Plots

```
setwd("/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/docs/")

if(!file.exists(paste0("Climate_gifs/", fips$State, "-", stid, "_GSOM-4plots.gif"))){
   print("Creating animated GSOM-4plots.gif")

img <- image_graph(600, 480, res = 96)
# START ----
ylim_new=NA
for(i in seq(min(GSOM$Year), max(GSOM$Year), by=2))
   {
par(las=1, mfrow=c(4,1), mar= c(4, 4, 2, 1) + 0.1)
# TMINmonthMax
   GSOMsub <- GSOM[GSOM$Month==TMINmonthMax & GSOM$Year<=i,]
   if(nrow(GSOMsub)<10) next
plot(TMIN~Date, GSOMsub[GSOMsub$Month==TMINmonthMax,],
   col='gray70', pch=20, xlab="",
   main=paste("Mean", format(GSOMsub$Date,"%B")[1],
               "Min. Temp", GSOM_Longest$name))
GSOM.lm = lm(TMIN~Date, GSOMsub)
pred_dates <-data.frame(Date = GSOMsub$Date);
nrow(pred_dates); pred_dates
#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
               interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="darkred")
lines(pred_dates$Date, as.numeric(ci[,2]), col="darkorange")
lines(pred_dates$Date, ci[,3], col="darkorange")
location_index = round(length(GSOMsub$Date) * 0.99,0)
text(pred_dates$Date[location_index], ci[location_index,3],
     paste(report_prob2(GSOM.lm)), pos=2, cex=1.5)

# Box Plot of TMAX by Month
CHCNDsub = subset(CHCND, CHCND$Year<=i,
      select=c(Month, Month.name, TMAX, TMIN))
boxplot(TMAX ~ Month.name, data=CHCNDsub, main="")
symbol.y = (par()$yaxp[2])-diff(par()$yaxp[1:2])*.99
#symbol.y = (par()£yaxp[2])
text(sumstats$Month, symbol.y, sumstats$TMAX_Symbol,
     col="red", cex=2)
mtext(paste("Maximum Daily Temperatures", min(CHCND$Year),
      "-", i, GSOM_Longest$name), line=1)
mtext("(NOTE: Red astrisks correspond to signficant changes)",
      line=0, cex=.7)
```

```
# TMAXmonthMax
GSOMsub <- GSOM[GSOM$Month==TMAXmonthMax & GSOM$Year<=i,]
ylim = range(GSOMsub$TMAX)
#if(!is.na(ylim_new)) ylim[2]=ylim_new
plot(TMAX~Date, GSOMsub, col='gray70', pch=20, xlab="",
     ylim=ylim,
     main=paste("Mean", format(GSOMsub$Date,"%B")[1],
                "Max. Temp", GSOM_Longest$name))
GSOM.lm = lm(TMAX~Date, GSOMsub)

ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="darkred")
lines(pred_dates$Date, as.numeric(ci[,2]), col="darkorange")
lines(pred_dates$Date, ci[,3], col="darkorange")

text(pred_dates$Date[location_index], ci[location_index,3],
     paste(report_prob2(GSOM.lm)), pos=2, cex=1.5)

# Record High Temperatures
plot(TMAX~Date, CHCND[CHCND$Year<=i,], pch='.', col="grey80",
     main="Recorded Daily High Temperatures")
points(maxTMAX~Date, data=CHCND[CHCND$Year<=i,], pch=20,
       col="red", cex=.8 )
}

# STOP ----
dev.off()

GSOM_animation <- image_animate(img, fps = 1, loop=2, optimize = TRUE)
image_write(GSOM_animation, paste0("Climate_gifs/", fips$State, "-", stid, "_GSOM-4plots.gif

} else {
   print("Skipping animated GSOM-4plots chunk")}
```

The file is saved in the main directory.

6 OLD Stuff

7 Other attempts...

```
ncdc_locs(locationcategoryid='CITY', sortfield='name',
          sortorder='desc')
```

```
# ncdc_locs(locationcategoryid='CITY',
#  locationid='FIPS:01', sortfield='name', sortorder='desc')

#ncdc_datasets(locationcategoryid='CITY',
#   locationid='FIPS:01', sortfield='name', sortorder='desc')


out <- ncdc(datasetid='NORMAL_DLY', stationid='GHCND:USW00014895',
            datatypeid='dly-tmax-normal', startdate = '2010-05-01',
            enddate = '2010-05-10')
```

```
with_units <- ncdc(datasetid='GHCND', stationid='GHCND:USW00014895',
                   datatypeid='TMAX', startdate = '2010-05-01',
                   enddate = '2010-10-31', limit=500, add_units = TRUE)
head( with_units$data )

## # A tibble: 6 x 9
##   date  datatype station value fl_m  fl_q  fl_so fl_t  units
##   <chr> <chr>    <chr>   <int> <chr> <chr> <chr> <chr> <chr>
## 1 2010~ TMAX     GHCND:~   222 ""    ""    0     2400  celc~
## 2 2010~ TMAX     GHCND:~   222 ""    ""    0     2400  celc~
## 3 2010~ TMAX     GHCND:~   233 ""    ""    0     2400  celc~
## 4 2010~ TMAX     GHCND:~   222 ""    ""    0     2400  celc~
## 5 2010~ TMAX     GHCND:~   272 ""    ""    0     2400  celc~
## 6 2010~ TMAX     GHCND:~   194 ""    ""    0     2400  celc~
```

7.1 Evaluating Records

TBD

7.2 Export Options

TBD

8 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/`

```
library(chron)
library(RColorBrewer)
library(lattice)
#library(ncdf)
library(ncdf4)
#library(greenbrown) # for gridded trend analysis

ersst.nc = "/home/CAMPUS/mwl04747/github/Climate_Change_Narratives/Data/FA19/ersst.v5.185401
Y1854 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1854.asc"
Y1864 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1864.asc"
Y1874 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1874.asc"
Y1884 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1884.asc"
Y1894 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1894.asc"
Y1904 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1904.asc"
Y1914 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1914.asc"
Y1924 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1924.asc"
Y1934 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1934.asc"
Y1944 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1944.asc"
Y1954 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1954.asc"
Y1964 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1964.asc"
Y1974 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1974.asc"
Y1984 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1984.asc"
Y1994 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.1994.asc"
Y2004 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.2004.asc"
Y2014 = "https://www1.ncdc.noaa.gov/pub/data/cmb/ersst/v5/ascii/ersst.v5.2014.asc"

temp = rbind(read.table(Y1854)[75,67], read.table(Y1864)[75,67], read.table(Y1874)[75,67],
read.table(Y1884)[75,67], read.table(Y1894)[75,67], read.table(Y1904)[75,67],
read.table(Y1914)[75,67], read.table(Y1924)[75,67], read.table(Y1934)[75,67],
read.table(Y1944)[75,67], read.table(Y1954)[75,67], read.table(Y1964)[75,67],
read.table(Y1974)[75,67], read.table(Y1984)[75,67], read.table(Y1994)[75,67],
read.table(Y2004)[75,67], read.table(Y2014)[75,67])

temp.df = data.frame(Temp = as.vector(temp)/100); temp.df
temp.df$Year = seq(1854, 2014, 10)
plot(Temp~ Year, temp.df)
abline(coef(lm(Temp~Year, data=temp.df)), col="red")
#automating this process!
```

```
directory = "/pub/data/cmb/ersst/v5/ascii"

B195401 = nc_open(ersst.nc)


# str(B195401)
# print(B195401)

ncin = B195401

print(ncin)
lon <- ncvar_get(ncin, "lon")
nlon <- dim(lon)
head(lon)

lat <- ncvar_get(ncin, "lat", verbose = F)
nlat <- dim(lat)
head(lat)

print(c(nlon, nlat))

t <- ncvar_get(ncin, "time")
tunits <- ncatt_get(ncin, "time", "units")
nt <- dim(t); nt

lat.sel = 67; lon.set = 75

#ncvar_get(ncin, sst) #object 'sst' not found

#ncvar_get(ncin, varfsst) object of type 'closure' is not subsettable
#ncvar_get(ncin, var) second argument to ncvar_get must be an object of type ncvar or ncdim

ncvar_get(ncin, "sst") #spits out the temperatures. but why the negative numbers!

# tmp.array <- ncvar_get(ncin, dname) # doesn't work...

tmp.array <- ncvar_get(ncin, "sst")
dim(tmp.array)

tmp.array[75, 67]

tmp.array[67,]

dlname <- ncatt_get(ncin, "sst", "long_name")
dunits <- ncatt_get(ncin, "sst", "units")
```

```r
fillvalue <- ncatt_get(ncin, "sst", "_FillValue")
dim(tmp.array)

title <- ncatt_get(ncin, 0, "title")
institution <- ncatt_get(ncin, 0, "institution")
datasource <- ncatt_get(ncin, 0, "source")
references <- ncatt_get(ncin, 0, "references")
history <- ncatt_get(ncin, 0, "history")
Conventions <- ncatt_get(ncin, 0, "Conventions")

# split the time units string into fields
tustr <- strsplit(tunits$value, " ")
tdstr <- strsplit(unlist(tustr)[3], "-")
tmonth = as.integer(unlist(tdstr)[2])
tday = as.integer(unlist(tdstr)[3])
tyear = as.integer(unlist(tdstr)[1])
chron(t, origin = c(tmonth, tday, tyear))

# tmp.array[tmp.array == fillvalue$value] <- NA

# length(na.omit(as.vector(tmp.array[, , 1])))

m <- 1
tmp.slice <- tmp.array[, , m]

image(lon, lat, tmp.array, col = rev(brewer.pal(10, "RdBu")))

# image(lon, lat, tmp.slice, col = rev(brewer.pal(10, "RdBu")))
```

9 Satellite Data

TBD

10 Ice-Core Data

TBD

11 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 4.2 minutes to process and compile. My next goal will be to optimize the process and streamline the time to analyze.