# Analyzing Climate Trends

Marc Los Huertos

February 4, 2024 (ver. 0.29)

# 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.

## 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

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 distirbution changes by decade / score

# 2 Functions to Analyze Monthly Trends

```
## Error in source(here("04_Regional_Climate_Trends", "Guides", "Guide3.R")):
/home/mwl04747/RTricks/04_Regional_Climate_Trends/Guides/Guide3.R:23:1:
unexpected '@'
## 22:
## 23:  @
##      ^
```

## 2.1 Before Starting the Process

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

## 2.2 R Code with Custom Functions

Run the Guide3.R code and the functions will be loaded into your environment automatically.

As we get futhrer into the project, these code chunks may require tweaking because of the questions you are interested in your location. Marc and the mentors will help you with this.

## 2.3 Function Descriptions and Use

The following functions are used to read the weather stations data into R, clean and pre-process data so we can analyze the data with the next guide.

## 2.4 Reading csv and loading R.data

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

```
## Error in print(read_and_load_data.fun): object 'read_and_load_data.fun'
not found
```

enddescription

## 2.5 Function to Evaluate Trends by Month

**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.

```
## Error in print(monthlyTrend.fun): object 'monthlyTrend.fun'
not found
```

**Example of how to use the function**

```
USC00042294.trends <- monthlyTrend.fun(USC00042294.a
## Error in monthlyTrend.fun(USC00042294.anomalies): could
not find function "monthlyTrend.fun"
```

**Explore Results** bwzr

Table 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.

```
## Error in xtable(USC00042294.trends[USC00042294.trends$ELEMENT
== "TMAX", :  object 'USC00042294.trends' not found
```

### 2.5.1 Example of Loop Code from Provious Interation

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.

```
## Error:  <text>:2:0:  unexpected end of input
## 1:  print(MonthEvalStats.fun
##     ^
```

## 2.6 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.

### 2.6.1 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),]
```

```
GSOM_anomaly_1975.png = paste0(fips$State2, "_", stid, "_GSOM_anomaly_1975.png")

png(paste0(png_private, GSOM_anomaly_1975.png),
    width = 480, height = 320, units = "px",
    pointsize = 12, bg = "white")
par(las=1, mfrow=c(1,1))

GSOM.lm = lm(TMAX.anom~Date, GSOM2)

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]))

pred_dates <-data.frame(Date = GSOM2$Date);
nrow(pred_dates); #pred_dates
#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="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$Year>1975]);
nrow(pred_dates); #pred_dates
#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
lines(pred_dates$Date, as.numeric(ci[,1]), col="red")
lines(pred_dates$Date, as.numeric(ci[,2]), col="darkorange")
```

Figure 1: The changing in monthly temperature data.

```
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()
```

And to see what we created, see Figure 1.

#### 2.6.2 Method 2: Polynomial Filter

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.

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

```
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 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.

```
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(data= PRCP.Season.Total, aes(x=Year, y = PRCP, color = Season, group=Se
```

## 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.

```
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)
names(Drought.run)

# 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)

if(Drought.run$lengths>100) {
   Drought.run.100 = aggregate(lengths~Year,
   data=Drought.run[Drought.run$lengths>=100,], sum)
}

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)

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


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$lengths>10,])
summary(Drought.run.lm)
text(min(Drought.run$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,])
```

Rainfall Probability Distributions by decade... to be developed.

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

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

x <- PRCP.Decade$PRCP[PRCP.Decade$Decade==1900]
df <- approxfun(density(x))
plot(1:12, density(x))
xnew <- c(0.45,1.84,2.3)
points(xnew,df(xnew),col=2)
```

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

### 3.3 Record Setting Temperature Records

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

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

```
names(CHCND)

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


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)
```

```
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"))
```

I tried to use a for loop and in then statements and it was painfully slow, so I converted the data to a matrix that can be used by barplots with much more effeciency!

Create the matrix

```
library(lubridate)
str(CHCND)

TMAX.mat.noleap <- matrix(NA, nrow=366, ncol=max(CHCND$Year) - min(CHCND$Year)+1)
TMIN.mat <- matrix(NA, nrow=366, ncol=max(CHCND$Year) - min(CHCND$Year)+1)
#TMAX.mat.leap <- matrix(NA, nrow=1, ncol=max(CHCND£Year) - min(CHCND£Year))

# Dumb Method, fraction of year might be better...

CHCND.noleap = subset(CHCND, select=c(Date, Year, yday, TMAX, TMIN, PRCP),
                      subset=(mmdd!="02-29"))

## Add yday for leap year
CHCND.noleap$yday[CHCND.noleap$yday>=60 & !leap_year(CHCND.noleap$Date)]<-CHCND.noleap$

## Create leap year dataframe
CHCND.leap  = subset(CHCND, select=c(Date, Year, yday, TMAX, TMIN, PRCP),
                     subset=(mmdd=="02-29"))
names(CHCND.noleap)
years = seq(min(CHCND$Year), max(CHCND$Year), by=1)
year.seq = data.frame(Year = years, Col = seq_len(length(seq(min(CHCND$Year), max(CHCND

for(i in min(CHCND.noleap$Year):max(CHCND.noleap$Year)){
      for(j in c(1:59, 61:366)){
      # i=2016; j = 50;
         TMAX.mat.noleap[j, year.seq$Col[year.seq$Year==i]] <-
         CHCND.noleap$TMAX[CHCND.noleap$Year==i & CHCND.noleap$yday == j]
         TMIN.mat[j, year.seq$Col[year.seq$Year==i]] <-
         CHCND.noleap$TMIN[CHCND.noleap$Year==i & CHCND.noleap$yday == j]
      }
   if(leap_year(i)){
      TMAX.mat.noleap[60, year.seq$Col[year.seq$Year==i]] <- CHCND.leap$TMAX[CHCND.leap
      TMIN.mat[60, year.seq$Col[year.seq$Year==i]] <- CHCND.leap$TMIN[CHCND.leap$Year==
      print(paste0("Added Leap Year", i))
   } else {print(paste0("Process non-leap year ", i))}
}

dim(TMAX.mat.noleap)

max(CHCND$yday[CHCND$Year==max(CHCND$Year)])
```

```r
# subset(CHCND, select=c(Date, yday, Year, TMAX), subset=(Year >= max(CHCND$Year)-1 & y

# TMAX.mat.noleap[140:144,134:135]
```

```r
records.png = paste0(fips$State2, "_", stid, "_CHCND_Temp_Records.png")

png(paste0(png_private, records.png), width = 480, height = 280, units = "px", pointsiz

results<-NULL
decades <- years[years/10 == floor(years/10)]
i = max(CHCND$Year)
# START LOOP
   j = which(years %in% i)
   if(sum(is.na(TMAX.mat.noleap[,j]))==366) next
TMAX1 = apply(TMAX.mat.noleap[,1:j], 1, function (x) which.max(x));
is.na(TMAX1) <- lengths(TMAX1) == 0
TMAX1 <- unlist(TMAX1)
TMAX1 <- count(TMAX1)
#str(TMAX1)
names(TMAX1)=c("Year", "TMAX")
TMAX_na = data.frame(Year=1:j)
TMAX <- merge(TMAX_na, TMAX1, all.x=TRUE, by="Year")

if(sum(is.na(TMIN.mat[,j]))==366) next
   # Select Minimum and Change to Negative Value
TMIN1 = apply(TMIN.mat[,1:j], 1, function (x) which.min(x));
is.na(TMIN1) <- lengths(TMIN1) == 0
TMIN1 <- unlist(TMIN1)
TMIN1 <- count(TMIN1) # Max Counts Negagive
#str(TMIN1)
names(TMIN1)=c("Year", "TMIN")
TMIN_na <- data.frame(Year=1:j)
TMIN <- merge(TMIN_na, TMIN1, all.x=TRUE, by="Year")

R1 <- merge(TMAX, TMIN, by="Year")
R1$Index = rep(j, nrow(R1))
#results = rbind(results, R)
R1$TMIN = -R1$TMIN
## Sorting out X Axis
tic.no <- 4
rowskip = round(nrow(R1)/tic.no, 0)
row_numb <- seq_len(nrow(R1)) %% rowskip
row.sel = which(row_numb %in% c(1))
index.year <- years[row.sel]
```

Figure 2: Daily temperatures that have been the highest on record (in red) and lowest on record (in blue). In some cases, climate change has created more records in the recent decades, while other stations seem don't show that trend.

```
# switch to decades?

xtics = row.sel
xlabs = index.year

yrange = range(c(R1$TMIN, R1$TMAX), na.rm=T)
ytics = floor(seq(yrange[1], yrange[2], length.out=tic.no))
ylabs = as.character(abs(ytics))

par(las=1, xpd=TRUE)
plot(c(1,nrow(R1)), c(yrange[1], yrange[2]), ty="n", xaxt='n', yaxt='n', xlab="Year", y
axis(2,at=ytics, labels=ylabs)
axis(1,at=xtics, labels=xlabs)
barplot(height = R1$TMAX, space=0, add = TRUE, axes = FALSE, col="red")
barplot(height = R1$TMIN, space=0, add = TRUE, axes = FALSE, col="blue")
# END LOOP

dev.off()
```

The patterns of record temperatures often shows increasing number of new high temperature records and fewer record low temperatures more recently, but as usual, it depends on the location (Figure 2).

### 3.4 Iterate TMAX vs. Month Boxplots

Evaluating the changes in TMAX and Monthly temperatures might be useful, but for now, I think it's hard to see the patterns.

### 3.5 Four Plots Compelling Figures

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!

### 3.6 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.

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) and the lowest minimum temperatures (in blue). In some cases, climate change has created more records in the recent decades, while other stations seem don't show that trend.

### 3.6.1 Change Point Analysis

First, TMIN and TMAX and change point analysis...

https://cran.r-project.org/web/packages/mcp/readme/README.html

```
dyn.load('/opt/jags/4.3.1/lib/libjags.so.4')
library(mcp)

# Define the model
model = list(
  response ~ 1,  # plateau (int_1)
  ~ 0 + time,    # joined slope (time_2) at cp_1
  ~ 1 + time     # disjoined slope (int_3, time_3) at cp_2
)

# Get example data and fit it
ex = mcp_example("demo")
fit = mcp(model, data = ex$data)

summary(fit)

# 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
```

Figure 4: Keep it simple stupid!

```
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")

ex$data$time;
fit = mcp(model, ex$data)
summary(fit)

plot(fit) + plot_pars(fit, pars = c("cp_1"), type = "dens_overlay")

ex$data$time;
fit = mcp(model, ex$data)
summary(fit)

GSOM$TMAX
test.df = data.frame(TMAX = GSOM$TMAX , time=1:1523)
model2 = list(
  price ~ 1 + ar(2),
  ~ 0 + time + ar(1)
)
fit2 = mcp(model2, test.df)
plot(fit2) + plot_pars(fit2, pars = c("cp_1"), type = "dens_overlay")
```

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

### 3.7 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)))

GSOM_dnorm.png <- paste0(fips$State2, "_", stid, "_GSOM_dnorm.png")

png(paste0(png_private, GSOM_dnorm.png),
    width = 480, height = 300, units = "px", pointsize = 12, bg = "white")
# TMIN
par(mfrow=c(1, 3), las=1, mar = c(5, 4, 4, 0.2) + 0.1, xpd=FALSE)
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*.95,
   paste0("Change ", round(Delta, 1), "F"), pos=3, cex=.9)



# 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*.96,
   paste0("Change ", round(Delta, 1), "F"), pos=3, cex=0.9)

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

# 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)){
```

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

```
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$PPT.anom[i,1], col=h.ramp[i], lwd=2)
Delta = PPT.anomaly.list$PPT.anom[i,1] - PPT.anomaly.list$PPT.anom[1,1]

text(PPT.anomaly.list$PPT.anom[i,1], Anom.y*.96,
   paste0("Change ", round(Delta, 1), " inches"), pos=3, cex=0.9)

dev.off()
```

This figure is pretty effective, but still needs work.

### 3.8 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?

```
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

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.

```
GSOM_estPDF.png = paste0(fips$State2, "_", stid, "_GSOM_estPDF.png")

if(!file.exists(paste0(png_private, GSOM_estPDF.png))){
   print("Creating estimated density distribution")

png(paste0(png_private, GSOM_estPDF.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])
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")}
```

Figure 6: The changing in monthly temperature data.

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!

# 4 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.

## 4.1 Probability Distributions

```
GSOM_dnorm.gif = paste0(fips$State2, "_", stid, "_GSOM_dnorm.gif")

if(!file.exists(paste0(gif_private, GSOM_dnorm.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"
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"
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", xla
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.
}
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", xla
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.
```

```
}
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)

image_write(GSOM_animation, paste0(gif_private, GSOM_dnorm.gif))

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

The file is saved in the main directory.

## 4.2 4 Weather Trend Plots

```
panel4.gif = paste0(fips$State2, "_", stid, "_4panel.gif")

if(!file.exists(paste0(gif_private, panel4.gif))){
   print("Creating animated 4panel.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(2, 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$Year<=i,
      select=c(Month, Month.name, TMAX, TMIN))
boxplot(TMAX ~ Month.name, data=CHCNDsub, xlab="", main="")
symbol.y = (par()$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$TMAX)
#if(!is.na(ylim_new)) ylim[2]=ylim_new
plot(TMAX~Date, GSOMsub, col='gray70', pch=20,
     ylim=ylim, xlab="",
     main=paste("Mean", format(GSOMsub$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
# START
   j = which(years %in% i)
   if(sum(is.na(TMAX.mat.noleap[,j]))==366) next
TMAX1 = apply(TMAX.mat.noleap[,1:j], 1, function (x) which.max(x));
is.na(TMAX1) <- lengths(TMAX1) == 0
```

```
TMAX1 <- unlist(TMAX1)
TMAX1 <- count(TMAX1)
#str(TMAX1)
names(TMAX1)=c("Year", "TMAX")
TMAX_na = data.frame(Year=1:j)
TMAX <- merge(TMAX_na, TMAX1, all.x=TRUE, by="Year")

if(sum(is.na(TMIN.mat[,j]))==366) next
   # Select Minimum and Change to Negative Value
TMIN1 = apply(TMIN.mat[,1:j], 1, function (x) which.min(x));
is.na(TMIN1) <- lengths(TMIN1) == 0
TMIN1 <- unlist(TMIN1)
TMIN1 <- count(TMIN1) # Max Counts Negagive
#str(TMIN1)
names(TMIN1)=c("Year", "TMIN")
TMIN_na <- data.frame(Year=1:j)
TMIN <- merge(TMIN_na, TMIN1, all.x=TRUE, by="Year")

R1 <- merge(TMAX, TMIN, by="Year")
R1$Index = rep(j, nrow(R1))
#results = rbind(results, R)
R1$TMIN = -R1$TMIN
## Sorting out X Axis
tic.no <- 4
rowskip = round(nrow(R1)/tic.no, 0)
row_numb <- seq_len(nrow(R1)) %% rowskip
row.sel = which(row_numb %in% c(1))
index.year <- years[row.sel]
# switch to decades?

xtics = row.sel
xlabs = index.year

yrange = range(c(R1$TMIN, R1$TMAX), na.rm=T)
ytics = floor(seq(yrange[1], yrange[2], length.out=tic.no))
ylabs = as.character(abs(ytics))

par(las=1, xpd=TRUE)
plot(c(1,nrow(R1)), c(yrange[1], yrange[2]), ty="n", xaxt='n', yaxt='n', ylab="No. of R
axis(2,at=ytics, labels=ylabs)
axis(1,at=xtics, labels=xlabs)
barplot(height = R1$TMAX, space=0, add = TRUE, axes = FALSE, col="red")
barplot(height = R1$TMIN, space=0, add = TRUE, axes = FALSE, col="blue")
# END
}
```

```
# STOP ----
dev.off()

GSOM_animation <- image_animate(img, fps = 1, loop=2, optimize = TRUE)
image_write(GSOM_animation, paste0(gif_private, panel4.gif))

} else {
   print("Skipping animated GSOM_4plots chunk")}
```

The file is saved in the main directory.

### 4.3 Evaluating Records

TBD

### 4.4 Export Options

TBD

## 5 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/

## 6 Satellite Data

TBD

# 7 Ice-Core Data

TBD

# 8 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