Obtaining Climate Records

Marc Los Huertos

March 28, 2022

1 Terrestrial Meteorological Data

1.1 Selected History of Climate Science

Geologists have known the earth's 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 XX scientiest Arrenheus, 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.

2 NOAA Data Records

2.1 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.

3 Temperature Trends by State

3.1 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?"

3.2 Approach

3.2.1 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)

# Alabama
# ncdc_locs(locationid='FIPS:01', limit=52)
```

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.

3.2.2 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.

```
# alabama stations.. sorted by the most recent
# test <- ncdc_stations(datasetid='GHCND',
# datatypeid = c("TMAX", "TMIN"), locationid='FIPS:01',
# limit=1000, sortfield = 'maxdate', sortorder='desc')

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)
}
```

3.2.3 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.

```
fips = get_locationid(44); str(fips);

## 'data.frame': 1 obs. of  2 variables:
##  $ State: chr "Texas"
##  $ id   : chr "FIPS:48"
```

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$datacoverage > 0.95,]
GSOM_Longest =
   GSOM_Coverage[GSOM_Coverage$mindate == min(GSOM_Coverage$mindate),]
GSOM_Longest =
   GSOM_Longest[1,] #Pick first if more than one.
```

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

```
##    elevation    mindate    maxdate latitude                     name
## 83      83.8 1893-01-01 2022-02-01  29.4705 HALLETTSVILLE 2 N, TX US
##    datacoverage                id elevationUnit longitude
## 83       0.9794 GHCND:USC00413873        METERS  -96.9397
```

3.2.4 Download GSOM Data using rnoaa

```
## [1] 1893
```

3.2.5 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.

```r
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)
}
```

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.

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

# 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)

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

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

dfTbl_TMAX[1,]

##          Date           station  TMAX
## 1 1893-01-01 GHCND:USC00413873 63.59

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

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

3.3 Function to Evaluate Months

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.

```
MonthEvalStats <- function(GSOM) {
sumstats = NA
for (m in 1:12) {
# m=2
  TMAX.lm = lm(TMAX~Date, GSOM[GSOM$Month==m,])
  TMIN.lm = lm(TMIN~Date, GSOM[GSOM$Month==m,])
sumstats = rbind(sumstats, data.frame(Month = m,
         TMIN_Slope = coef(TMIN.lm)[2],
         TMIN_r2 = summary(TMIN.lm)$r.squared,
         TMIN_p_value= anova(TMIN.lm)$'Pr(>F)'[1],
         TMAX_Slope = coef(TMAX.lm)[2],
         TMAX_r2 = summary(TMAX.lm)$r.squared,
         TMAX_p_value= anova(TMAX.lm)$'Pr(>F)'[1]))
}

sumstats=data.frame(sumstats)[-1,]
rownames(sumstats)<-NULL
#sumstats <- subset(sumstats, TMAX_p_value <.05,
# select=c(Month, TMIN_Slope, TMIN_p_value,
# MAX_Slope, TMAX_p_value))
sumstats$TMAX_Symbol <- sumstats$TMIN_Symbol <- ""
sumstats$TMAX_Symbol[sumstats$TMAX_p_value < 0.05] = "*"
sumstats$TMAX_Symbol[sumstats$TMAX_p_value < 0.01] = "**"
sumstats$TMAX_Symbol[sumstats$TMAX_p_value < 0.001] = "***"
sumstats[,c(7,9)]
return(sumstats)
}

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

3.3.1 Determine Months with Biggest Changes

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 Month (TMAX overwrites)
sumstats = MonthEvalStats(GSOM)

maxmonth = sumstats$Month[sumstats$TMIN_Slope ==
                          max(sumstats$TMIN_Slope, na.rm=T)]

maxmonth = sumstats$Month[abs(sumstats$TMAX_Slope) ==
                          max(abs(sumstats$TMAX_Slope), na.rm=T)]
```

3.4 Plot Month with Biggest Changes

3.4.1 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 Signficant)"))
   #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=""))
   }
}
```

3.5 Extreme Temperture Events

3.5.1 Functions to Collect and Clean CHCND

```
GSOM_Longest$id

## [1] "GHCND:USC00413873"

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"))
   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] 46349
```

```
#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]   8.96 111.02

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]  24.98 111.02

CHCND$Year = as.numeric(format(as.Date(CHCND$Date), format = "%Y"))

#head(CHCND)
```

3.6 Determine Record Setting Temperatures

3.7 Iterate TMAX Boxplots

4 Plot Results

4.1 Static 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 Animation

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.

```r
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)
   GSOMsub <- GSOM[GSOM$Month==maxmonth & GSOM$Year<=i,]
   if(nrow(GSOMsub)<10) next
plot(TMIN~Date, GSOMsub[GSOMsub$Month==maxmonth,],
   col='gray70', pch=20, xlab="",
   main=paste("Mean", format(GSOMsub$Date,"%B")[1],
              "Min. Temp", GSOM_Longest$name))
GSOM.lm = lm(TMIN~Date, GSOMsub)
pred_dates <-data.frame(Date = GSOMsub$Date);

#Predits the values with confidence interval
ci <- predict(GSOM.lm, newdata = pred_dates,
              interval = 'confidence')
#   str(ci)
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)

# 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)

# TMAX --------------------------------
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)

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 )
print(nrow(pred_dates))#; pred_dates
}

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

The file is saved in the main directory.

```
#print(img)
```

```
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, "_GSOM.gif", sep=""))
```

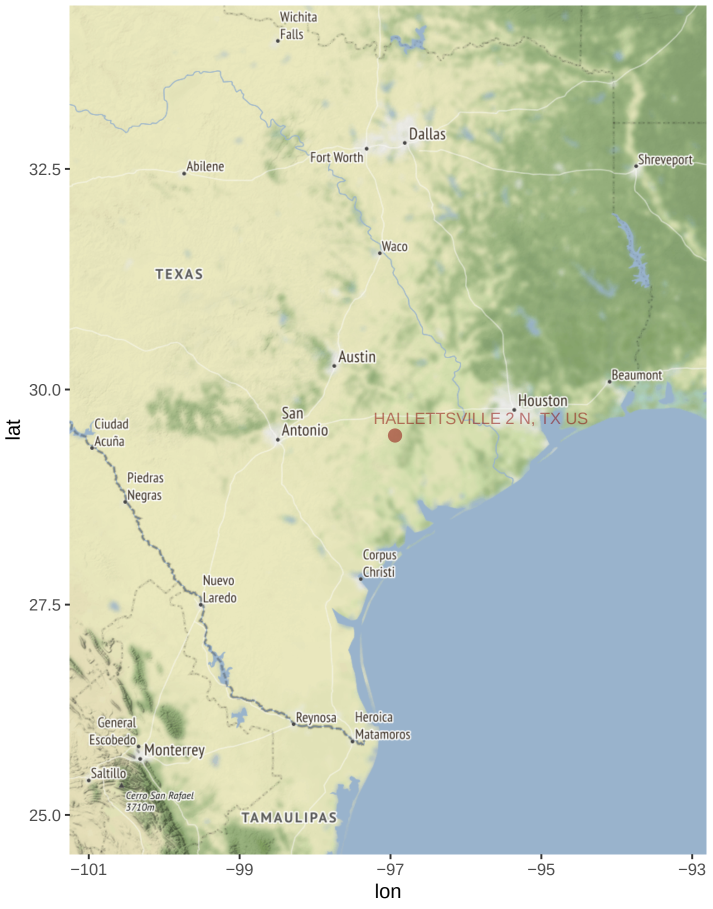
4.3 KISS

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

4.4 Show Map of Location

```r
library(ggmap)

## Google's Terms of Service:  https://cloud.google.com/maps-platform/terms/.
## Please cite ggmap if you use it!  See citation("ggmap") for details.

#API = "AIzaSyBfkMN5PYsBOA92RbOxo1bc51y-5aitKDI"
#register_google(key = API, write = TRUE)
#ggmap(myMap)+
#geom_point(aes(x = locus[1], y = locus[2]),
 #alpha = .5, color="darkred", size = 3)

GSOM_Longest$name

## [1] "HALLETTSVILLE 2 N, TX 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 -96.9
##  $ lat    : num 29.5
##  $ Station: Factor w/ 1 level "HALLETTSVILLE 2 N, TX 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=29.4705,-96.9397&zoom=7&
## Source :  http://tile.stamen.com/terrain/7/28/51.png
## Source :  http://tile.stamen.com/terrain/7/29/51.png
## Source :  http://tile.stamen.com/terrain/7/30/51.png
## Source :  http://tile.stamen.com/terrain/7/28/52.png
## Source :  http://tile.stamen.com/terrain/7/29/52.png
## Source :  http://tile.stamen.com/terrain/7/30/52.png
## Source :  http://tile.stamen.com/terrain/7/28/53.png
## Source :  http://tile.stamen.com/terrain/7/29/53.png
## Source :  http://tile.stamen.com/terrain/7/30/53.png
## Source :  http://tile.stamen.com/terrain/7/28/54.png
## Source :  http://tile.stamen.com/terrain/7/29/54.png
## Source :  http://tile.stamen.com/terrain/7/30/54.png
```

```r
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)
```



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

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

## pdf
##   2
```

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

incidents= read.csv('https://raw.githubusercontent.com/lgellis/MiscTutorial/master/ggmap/i2S

#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
```

4.5 OLD version

5 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 2010-05-01T00:00:00 TMAX     GHCND:USW0~   222 ""    ""    0     2400  celciu~
## 2 2010-05-02T00:00:00 TMAX     GHCND:USW0~   222 ""    ""    0     2400  celciu~
## 3 2010-05-03T00:00:00 TMAX     GHCND:USW0~   233 ""    ""    0     2400  celciu~
## 4 2010-05-04T00:00:00 TMAX     GHCND:USW0~   222 ""    ""    0     2400  celciu~
## 5 2010-05-05T00:00:00 TMAX     GHCND:USW0~   272 ""    ""    0     2400  celciu~
## 6 2010-05-06T00:00:00 TMAX     GHCND:USW0~   194 ""    ""    0     2400  celciu~
```

5.1 Evaluating Records

TBD

5.2 Export Options

TBD

6 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/`

```r
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"
```

```r
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, var£sst) 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
```

```r
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")
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")))
```

7 Satellite Data

TBD

8 Ice-Core Data

TBD