Obtaining and Cleaning NOAA Weather Station Data

Marc Los Huertos

January 30, 2024 (ver. 0.75)

1 Introduction

1.1 Goals

Using a list of active weather stations in the United States, you will download and process the data to create a time series of temperature anomalies.

1.2 Read Data

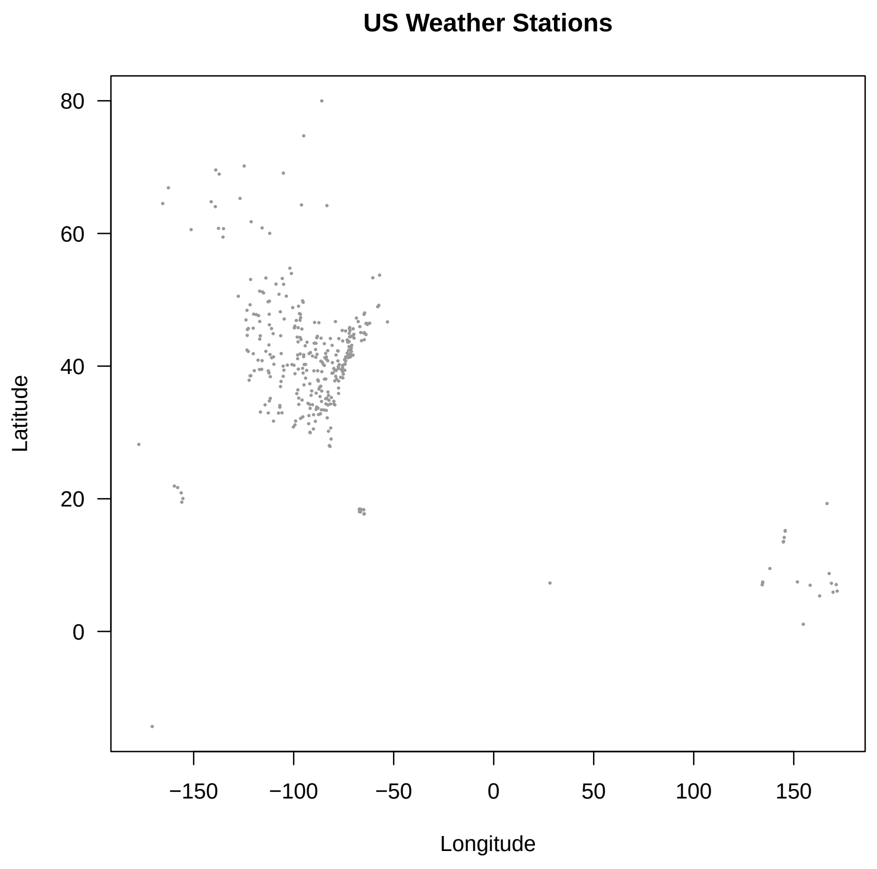
First, we install some packages and read in the data. I suggest you create a folder for the project (I created one called ”04_Regional_Climate_Trends”) and then used the function here() to get the working directory and read the csv into R. This might be easier than the file.choose() option, but you can use that if you prefer.

```
library(here)

## here() starts at /home/mwl04747/RTricks

library(xtable)

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

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

1.3 Map US Weather Stations

Here's a map of the weather stations in the dataset. Pretty lame map! We'll make a better one later.[1]

```r
plot(stations.active.oldest$LONGITUDE,
     stations.active.oldest$LATITUDE,
     xlab = "Longitude",ylab = "Latitude",
     pch=20, cex=0.3, col='gray60', las=1,
     main = "US Weather Stations")
```



1.4 Select and Evaluate State Data

[1]Evelyn/Brody: This is a good change to see how to use R for map making! First we need to transform that data.

```
stations.unique =
  unique(stations.active.oldest[,c("STATE", "STATE_NAME")])

xtab = xtable(stations.unique)
```

The each of you will select a state – see the Google Sheet sign up so we have a diverse set of states.

```
my.state = "CA" # change the "CA" to your state
```

2 Download Data from NOAA

2.1 Subset Station Data by State

This uses the stations.active.oldest file to download the data from the NOAA website based on the state you have choose.

```
my.stations = subset(stations.active.oldest, STATE == my.state)

# Download Updated Station Data
i=1
here::here("04_Regional_Climate_Trends", my.stations$ID[i])

## [1] "/home/mwl04747/RTricks/04_Regional_Climate_Trends/USC00043157"
```

The loop is not ideal – loops are slow in R and prone to break – but it should work for our purposes.[2]

```
for(i in 1:nrow(my.stations)){
  url = paste0("https://www.ncei.noaa.gov/pub/data/ghcn/daily/by_station/",
               my.stations$ID[i],
               ".csv.gz")

print(i) # Print Index Number
download.file(url, paste0(here::here("04_Regional_Climate_Trends",
                                     "Data",
                                     "SP24/"),
                          my.stations$ID[i],
                          ".csv.gz"),
              quiet = FALSE, mode = "w", cacheOK = TRUE)

assign(paste0("station", i),
```

[2]Evelyn/Brody: I wonder if we can test if the file exists so we don't have to downwload every time!

```
       read.csv(gzfile(paste0(here::here("04_Regional_Climate_Trends",
            "Data", "SP24/"),my.stations$ID[i], ".csv.gz")),
            header=FALSE))

# can't get the header named in loop! Grrr...
#names(paste0("station",i)) <- c("ID", "DATE", "ELEMENT",
# "VALUE", "M-FLAG", "Q-FLAG", "S-FLAG", "OBS-TIME")

} # LOOP END

## [1] 1
## [1] 2
## [1] 3
## [1] 4
## [1] 5

# Fix Variable Names based on NOAA Documentation
names(station1) <- c("ID", "DATE", "ELEMENT", "VALUE",
                     "M-FLAG", "Q-FLAG", "S-FLAG", "OBS-TIME")
names(station3) <- names(station2) <- names(station1)
names(station5) <- names(station4) <- names(station1)

# NAMES OF VARIABLES ARE INCORRECT for some STATIONS??

  #ID = 11 character station identification code
  #YEAR/MONTH/DAY = 8 character date in YYYYMMDD format
  #                     (e.g. 19860529 = May 29, 1986)
  #ELEMENT = 4 character indicator of element type
  #DATA VALUE = 5 character data value for ELEMENT
  #M-FLAG = 1 character Measurement Flag
  #Q-FLAG = 1 character Quality Flag
  #S-FLAG = 1 character Source Flag
  #OBS-TIME = 4-character time of observation in hour-minute format
  #                     (i.e. 0700 =7:00 am); if no ob time information
 #is available, the field is left empty
```

3 Process and Clean Data

I have created a ”function” that can process and clean the data, if the data are consistent! If not, we’ll trouble shoot together.

Here’s the data structure, using str(), but if you have something different, please let me know and we’ll sort out how to fix it.

```
str(station1)

## 'data.frame': 224921 obs. of  8 variables:
##  $ ID      : chr  "USC00043157" "USC00043157" "USC00043157" "USC00043157" ...
##  $ DATE    : int  18670601 18670602 18670603 18670604 18670605 18670606 18670607 1867060
##  $ ELEMENT : chr  "PRCP" "PRCP" "PRCP" "PRCP" ...
##  $ VALUE   : int  0 0 0 0 0 0 0 0 0 0 ...
##  $ M-FLAG  : chr  "" "" "" "" ...
##  $ Q-FLAG  : chr  "" "" "" "" ...
##  $ S-FLAG  : chr  "F" "F" "F" "F" ...
##  $ OBS-TIME: int  NA NA NA NA NA NA NA NA NA NA ...
```

3.1 Clean Data

First, I tested each line on station1. I will then create a function to clean the data and apply it to each station.

```
station1$VALUE = station1$VALUE/10  # Correct Values Units
# Fix Date format
station1$Ymd = as.Date(as.character(station1$DATE), format = "%Y%m%d")
str(station1)

## 'data.frame': 224921 obs. of  9 variables:
##  $ ID      : chr  "USC00043157" "USC00043157" "USC00043157" "USC00043157" ...
##  $ DATE    : int  18670601 18670602 18670603 18670604 18670605 18670606 18670607 1867060
##  $ ELEMENT : chr  "PRCP" "PRCP" "PRCP" "PRCP" ...
##  $ VALUE   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ M-FLAG  : chr  "" "" "" "" ...
##  $ Q-FLAG  : chr  "" "" "" "" ...
##  $ S-FLAG  : chr  "F" "F" "F" "F" ...
##  $ OBS-TIME: int  NA NA NA NA NA NA NA NA NA NA ...
##  $ Ymd     : Date, format: "1867-06-01" "1867-06-02" ...

station1$MONTH = as.numeric(format(station1$Ymd, "%m"))
station1$YEAR = as.numeric(format(station1$Ymd, "%Y"))
station1.monthly = aggregate(VALUE ~ MONTH + YEAR,
                   data = subset(station1, ELEMENT == "TMAX"), mean)

# create baseline (normals) dataset
station1.normals = subset(station1,
                          Ymd >= "1961-01-01" & Ymd <= "1990-12-31")
station1.normals.monthly = aggregate(VALUE ~ MONTH,
                   data = subset(station1.normals, ELEMENT == "TMAX"), mean)
names(station1.normals.monthly) <- c("MONTH", "NORMALS")
```

```
station1.anomaly = merge(station1.monthly,
                         station1.normals.monthly, by = "MONTH")
station1.anomaly$ANOMALY =
  station1.anomaly$VALUE - station1.anomaly$NORMALS
```

3.2 Clean Data Function

Function is probably senstive to missing values, need to work on that!

```
cleandataframe.fun <- function(x){
  x$VALUE = x$VALUE/10
  x$Ymd = as.Date(as.character(x$DATE), format = "%Y%m%d")
  x$MONTH = as.numeric(format(x$Ymd, "%m"))
  x$YEAR = as.numeric(format(x$Ymd, "%Y"))
  x.monthly = aggregate(VALUE ~ MONTH + YEAR,
                   data = subset(x, ELEMENT == "TMAX"), mean)
  x.normals = subset(x, Ymd >= "1961-01-01" & Ymd <= "1990-12-31")
  x.normals.monthly = aggregate(VALUE ~ MONTH,
                   data = subset(x.normals, ELEMENT == "TMAX"), mean)
  names(x.normals.monthly) <- c("MONTH", "NORMALS")
  x.anomaly = merge(x.monthly, x.normals.monthly, by = "MONTH")
  x.anomaly$ANOMALY = x.anomaly$VALUE - x.anomaly$NORMALS
  return(x.anomaly)
}
```

3.3 Apply Function to All Stations

So far, I have only run function for 1 station, but I suspect you can figure out how to run it for each one!

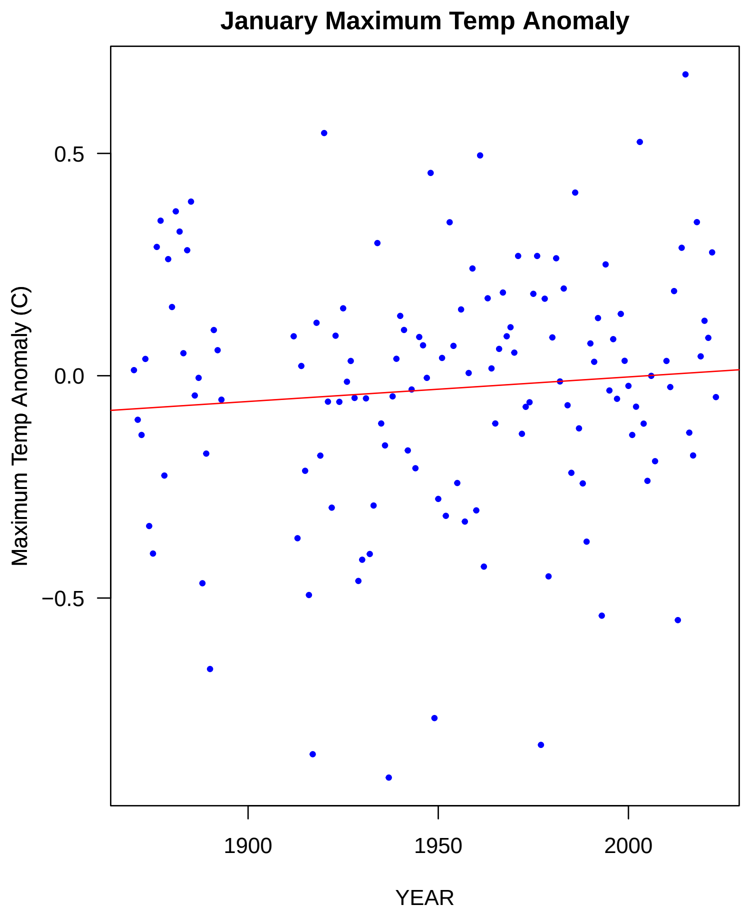
```
station1.TMAX = cleandataframe.fun(station1)
```

3.4 Plot Anomaly

Graphic has lots of issues. more next time! But here's a start.

```
options(scipen=5)
par(mar=c(4,6,2,5))

plot(ANOMALY ~ YEAR, data = subset(station1.TMAX, MONTH == 1),
     las=1, pch=19, col = "blue", cex=.5, #xlab = "Year",
     ylab = "Maximum Temp Anomaly (C)",
     main="January Maximum Temp Anomaly")
```

```
mtext("Maximum Temp Anomaly (C)", side = 2, line = 3)
temp.lm = lm(ANOMALY ~ YEAR, data = subset(station1.TMAX, MONTH == 1))
abline(coef(temp.lm), col = "red")
```



4 Next Steps

This is all we need to do so far. Next week, we'll look at different way to visualize the data!