Guide 2: Cleaning and Pre-Processing Weather Station Data

Marc Los Huertos

February 3, 2024 (ver. 0.4)

1 Introduction

1.1 Goals

The goal of this guide is to provide a step-by-step process for cleaning and pre-processing weather station data. The data is from the Global Historical Climatology Network (GHCN) Daily dataset. The data is available from the National Oceanic and Atmospheric Administration (NOAA) and is available from the National Centers for Environmental Information (NCEI) at https://www.ncdc.noaa.gov/data-access/land-based-station-data/land-based-datasets/global-historical-climatol

1.2 Background

What is GHCN-Daily?Links to an external site.

1.3 Approach

We need to address several things that might get in the way of our analysis:

1. read station data into R
2. date format (convert to POSIXct, adding month and year as varibles)
3. evaluate missing data
4. units
5. outliers
6. Other stuff??

2 Cleaning and Pre-Processing Functions

```
## Error in MonthlyAnomalies.fun(station1.monthly, station1.normals):
object 'station1.monthly' not found
```

2.1 Before Starting the Process

Before you begin, make sure you have the stations to read into R. Got to the "Files" tab in RStudio and make sure you see the station csv files in your data folder. If not, please Slack Marc and mentors, so we can troubleshoot the issue!

2.2 R Code with Custom Functions

Run the Guide2.R code and the functions will be loaded into your environment automatically.

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 Checking dataframe objects

Read all csv files into R The If the data are not in the R environment, you can read them in using the following but you have the uncompressed files, you can read them in using the following function:

```
## function (datafolder)
## {
##     StationList <- list.files(datafolder, full.names = TRUE,
##         pattern = "\\.csv$")
##     colnames <- c("ID", "DATE", "ELEMENT", "VALUE", "M-FLAG",
##         "Q-FLAG", "S-FLAG", "OBS-TIME")
##     for (i in 1:length(StationList)) {
##         assign(paste0("station", i), read.csv(StationList[i],
##             header = TRUE, col.names = colnames), envir = parent.frame())
##     }
## }
```

Example of how to use the function:

```
datafolder = "/home/mwl04747/RTricks/04_Regional_Climate_Trends/Data/SP24"
ReadStations.fun(datafolder)
```

Check the dataframes in the environment You can do this by running the following code:

```
ls()
```

```
##  [1] "ConvertUnits.fun"     "coverage.fun"         "datafolder"
##  [4] "fixdates.fun"         "MonthlyAnomalies.fun" "MonthlyNormals.fun"
##  [7] "MonthlyValues.fun"    "QAQC.fun"             "ReadStations.fun"
## [10] "station1"             "station2"             "station3"
## [13] "station4"             "station5"
```

You should see weather station objects.

Check the structure of the dataframes Using str(), make sure the datasets look right!

For example:

```
str(station2)
```

2.5 Clean Data

Next we'll "clean" the dataset by fixing the date format and preparing it for the analysis stages.

Function to Fix Dates

```
## function (station)
## {
##     station$Ymd = as.Date(as.character(station$DATE), format = "%Y%m%d")
##     station$MONTH = as.numeric(format(station$Ymd, "%m"))
##     station$YEAR = as.numeric(format(station$Ymd, "%Y"))
##     return(station)
## }
```

Example of how to use the function (Note: the new dataframe name change):

```
station1a <- fixdates.fun(station1)
```

Evaluation Data Coverage We need to know how much data we have for each station. This is important for the next steps in the process.

```
## function (station, element = "TMAX")
## {
##     Dates.all = data.frame(Ymd = seq.Date(from = min(station$Ymd),
##         to = max(station$Ymd), by = "day"))
##     station.full = merge(Dates.all, station, all = TRUE)
##     station.coverage = sum(!is.na(station.full$VALUE[station.full$ELEMENT ==
```

```
##         element]))/length(station.full$VALUE[station.full$ELEMENT ==
##         element]) * 100
##     return(round(station.coverage, 2))
## }
```

Example of how to use the function:

```
coverage.fun(station1a)
```

In general, we want something like 95% coverage for the period of record. If you don't have that, please let us know and we'll help you get additional stations with better coverage!

Function to Convert Units According the the NOAA website, the csv.gz files have five core elements include the following units, which we will convert:

PRCP = Precipitation (tenths of mm) → mm

SNOW = Snowfall (mm) → cm

SNWD = Snow depth (mm) → cm

TMAX = Maximum temperature (tenths of degrees C) → degrees C

TMIN = Minimum temperature (tenths of degrees C) → degrees C

```
## function (station)
## {
##     station$VALUE = station$VALUE/10
##     return(station)
## }
```

Example of how to use the function (Note: the new dataframe name change):

```
station1b <- ConvertUnits.fun(station1a)
```

Checking for Outliers NOAA website conducts a very rudimentary data quality check, but every year, we find stations with wacky numbers. Hopefully this function will find them. But if you do have some, let Marc and mentors know so we can figure out how to address them!

Here are some stuff we'll look for:

Extreme Values Plot values with time, is the scale crazy with just a few observations at the extreme?

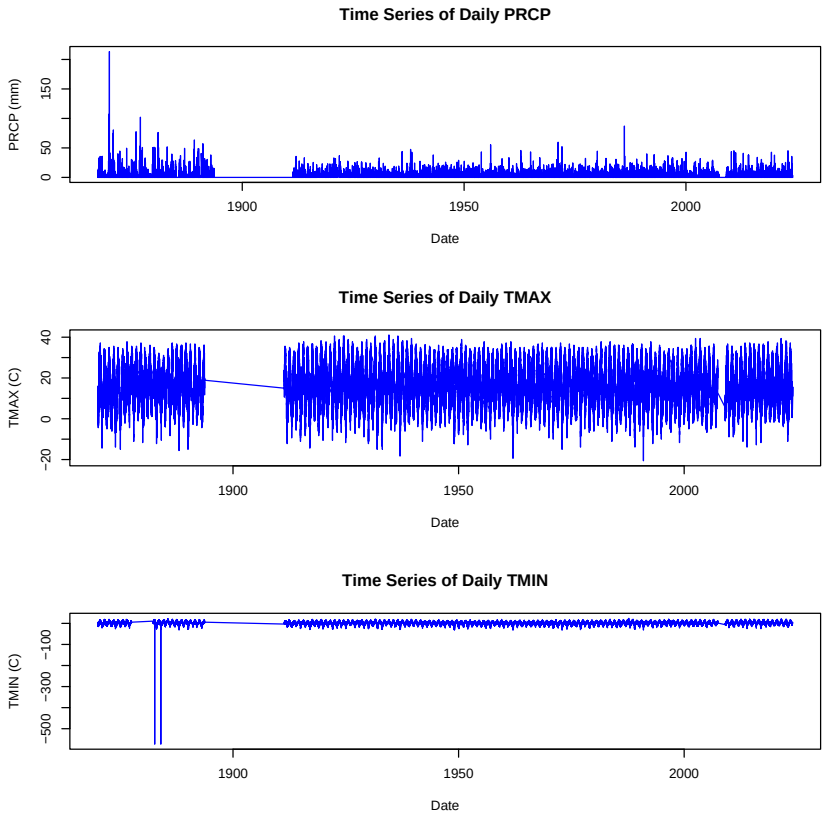
Sudden Shift ???

Note QA/QC Flags TBD

```
## function (station)
## {
##     par(mfrow = c(3, 1))
##     plot(VALUE ~ Ymd, data = subset(station, subset = ELEMENT ==
##         "PRCP"), type = "l", col = "blue", main = "Time Series of Daily PRCP",
##         xlab = "Date", ylab = "PRCP (mm)")
##     plot(VALUE ~ Ymd, data = subset(station, subset = ELEMENT ==
##         "TMAX"), type = "l", col = "blue", main = "Time Series of Daily TMAX",
##         xlab = "Date", ylab = "TMAX (C)")
##     plot(VALUE ~ Ymd, data = subset(station, subset = ELEMENT ==
##         "TMIN"), type = "l", col = "blue", main = "Time Series of Daily TMIN",
##         xlab = "Date", ylab = "TMIN (C)")
##     station = subset(station, Q.FLAG != "")
##     station = subset(station, M.FLAG != "")
##     station = subset(station, S.FLAG != "")
##     return(station)
## }
```

Example of how to use the function:

```
QAQC.fun(station1b)
```



If you have any hints of data problems, let's sort them out together!

Function to Create Monthly Values and Normals For TMAX and TMIN, we want monthly means, for rainfall, we'll want monthly totals.[1]

Here's the function for Monthly Values:

```
## function (x = station1)
## {
##     x.TMAX.monthly = aggregate(VALUE ~ MONTH + YEAR, data = subset(x,
##         ELEMENT == "TMAX"), mean)
##     names(x.TMAX.monthly) <- c("MONTH", "YEAR", "TMAX")
##     x.TMIN.monthly = aggregate(VALUE ~ MONTH + YEAR, data = subset(x,
##         ELEMENT == "TMIN"), mean)
##     names(x.TMIN.monthly) <- c("MONTH", "YEAR", "TMIN")
##     x.PRCP.monthly = aggregate(VALUE ~ MONTH + YEAR, data = subset(x,
```

[1] Brody/Evyln: We need code to exclue months with missing data, these might not be represenative of the month if missing, especically for PRCP!

```
##         ELEMENT == "PRCP"), sum)
##     names(x.PRCP.monthly) <- c("MONTH", "YEAR", "PRCP")
##     return(list(x.TMAX.monthly, x.TMIN.monthly, x.PRCP.monthly))
## }
```

Here's the function for Monthly Normals:

```
## function (x = station1b)
## {
##     x.normals = subset(x, Ymd >= "1961-01-01" & Ymd <= "1990-12-31")
##     x.TMAX.normals.monthly = aggregate(VALUE ~ MONTH, data = subset(x.normals,
##         ELEMENT == "TMAX"), mean)
##     names(x.TMAX.normals.monthly) <- c("MONTH", "NORMALS")
##     x.TMIN.normals.monthly = aggregate(VALUE ~ MONTH, data = subset(x.normals,
##         ELEMENT == "TMIN"), mean)
##     names(x.TMIN.normals.monthly) <- c("MONTH", "NORMALS")
##     x.PRCP.normals.month.year = aggregate(VALUE ~ MONTH + YEAR,
##         data = subset(x.normals, ELEMENT == "PRCP"), sum)
##     x.PRCP.normals.monthly = aggregate(VALUE ~ MONTH, data = subset(x.PRCP.normals.m
##         mean)
##     names(x.PRCP.normals.monthly) <- c("MONTH", "NORMALS")
##     return(list(x.TMAX.normals.monthly, x.TMIN.normals.monthly,
##         x.PRCP.normals.monthly))
## }
```

Example of how to use the functions:

```
station1.monthly <- MonthlyValues.fun(station1b)
station1.normals <- MonthlyNormals.fun(station1b)
```

Function to Create Anomalies Here's the function:

```
## function (station.monthly, station.normals)
## {
##     for (i in seq_along(station.monthly)) {
##         TMAX <- merge(station.monthly[[1]], station.normals[[1]],
##             by = "MONTH")
##         TMAX$TMAX.a = TMAX$TMAX - TMAX$NORMALS
##         TMAX$Ymd = as.Date(paste(TMAX$YEAR, TMAX$MONTH, "01",
##             sep = "-"))
##         TMIN <- merge(station.monthly[[2]], station.normals[[2]],
##             by = "MONTH")
##         TMIN$TMIN.a = TMIN$TMIN - TMIN$NORMALS
##         TMIN$Ymd = as.Date(paste(TMIN$YEAR, TMIN$MONTH, "01",
```

```
##             sep = "-"))
##         PRCP <- merge(station.monthly[[3]], station.normals[[3]],
##             by = "MONTH")
##         PRCP$PRCP.a = PRCP$PRCP - PRCP$NORMALS
##         PRCP$Ymd = as.Date(paste(PRCP$YEAR, PRCP$MONTH, "01",
##             sep = "-"))
##         return(list(TMAX, TMIN, PRCP))
##     }
## }
## <bytecode: 0x1599e88>
```

Example of how to use the function:

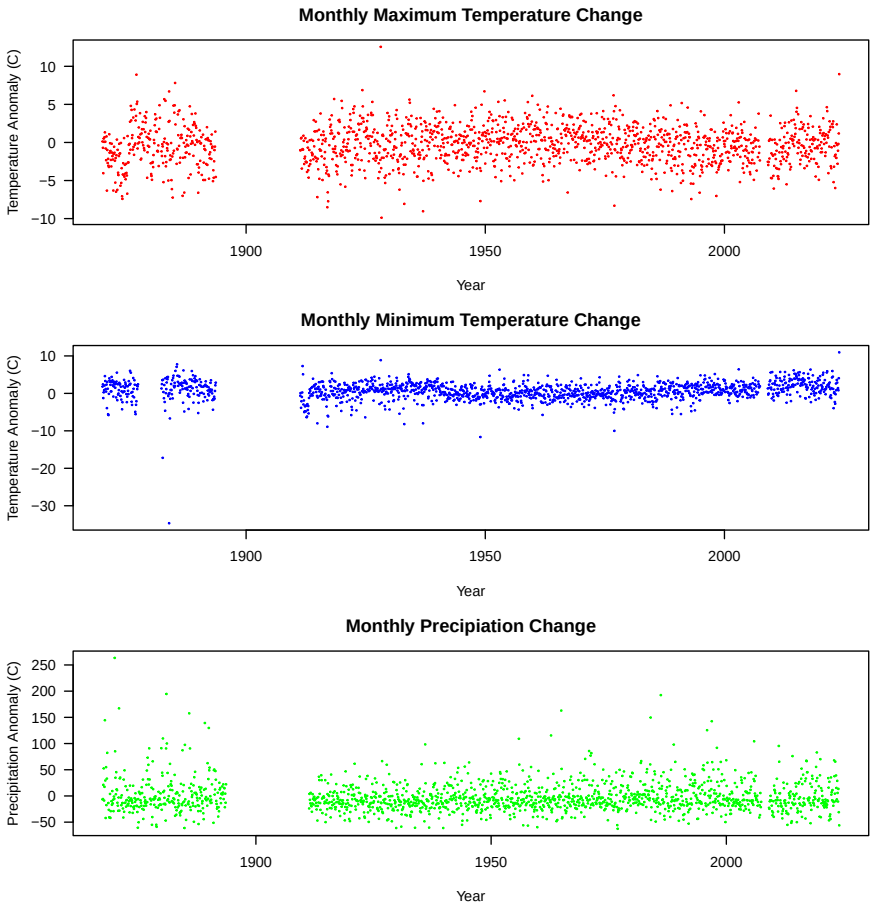
```
station1.anomolies <- MonthlyAnomalies.fun(station1.monthly, station1.normals)
```

2.6 Checking on the Results

Getting into the data is a bit tricky. The datasets is a list of dataframes. Each dataframe is a different variable, where 1 is TMAX, 2 is TMIN, and 3 is PRCP.

The get access to each you use the following code:

- `station1.anomolies[[1]]` for TMAX
- `station1.anomolies[[2]]` for TMIN
- `station1.anomolies[[3]]` for PRCP


Monthly Maximum Temperature Change
Temperature Anomaly (C)
10
5
0
−5
−10
1900
1950
2000
Year
Monthly Minimum Temperature Change
Temperature Anomaly (C)
10
0
−10
−20
−30
1900
1950
2000
Year
Monthly Precipiation Change
Precipitation Anomaly (C)
250
200
150
100
50
0
−50
1900
1950
2000
Year

3 Explaining the Code, TBD!

3.1 QA/QC: Troubleshooting

I will be getting all the guides working before working on this! But if there are errors with the custom function, this is where workarounds will be described! Please Slack me and mentors if you have any problems!

3.2 Work Arounds

TBD!

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

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)

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
```

```
x=station1
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.TMAX.monthly = aggregate(VALUE ~ MONTH + YEAR,
```

```
            data = subset(x, ELEMENT == "TMAX"), mean)
  names(x.TMAX.monthly) <- c("MONTH", "YEAR", "TMAX")
  x.TMIN.monthly = aggregate(VALUE ~ MONTH + YEAR,
            data = subset(x, ELEMENT == "TMIN"), mean)
  names(x.TMIN.monthly) <- c("MONTH", "YEAR", "TMIN")
  x.PRCP.monthly = aggregate(VALUE ~ MONTH + YEAR,
            data = subset(x, ELEMENT == "PRCP"), sum)
  names(x.PRCP.monthly) <- c("MONTH", "YEAR", "PRCP")

  x.normals = subset(x, Ymd >= "1961-01-01" & Ymd <= "1990-12-31")
  x.TMAX.normals.monthly = aggregate(VALUE ~ MONTH,
            data = subset(x.normals, ELEMENT == "TMAX"), mean)
  names(x.TMAX.normals.monthly) <- c("MONTH", "NORMALS")
  x.TMIN.normals.monthly = aggregate(VALUE ~ MONTH,
            data = subset(x.normals, ELEMENT == "TMIN"), mean)
  names(x.TMIN.normals.monthly) <- c("MONTH", "NORMALS")
  x.PRCP.normals.monthly = aggregate(VALUE ~ MONTH,
            data = subset(x.normals, ELEMENT == "PRCP"), sum)
  names(x.PRCP.normals.monthly) <- c("MONTH", "NORMALS")


  x.TMAX.anomaly = merge(x.TMAX.monthly, x.TMAX.normals.monthly, by = "MONTH")
  x.TMAX.anomaly$TMAX.anomaly = x.TMAX.anomaly$TMAX - x.TMAX.anomaly$NORMALS

  x.TMIN.anomaly = merge(x.TMIN.monthly, x.TMIN.normals.monthly, by = "MONTH")
  x.TMIN.anomaly$TMIN.anomaly = x.TMIN.anomaly$TMIN - x.TMIN.anomaly$NORMALS

  x.PRCP.anomaly = merge(x.PRCP.monthly, x.PRCP.normals.monthly, by = "MONTH")
  x.PRCP.anomaly$PRCP.anomaly = x.PRCP.anomaly$PRCP - x.PRCP.anomaly$NORMALS

  TEMP <- merge(x.TMAX.anomaly, x.TMIN.anomaly, by = c("MONTH", "YEAR") )
  x.anomaly <- merge(TEMP, x.PRCP.anomaly, by = c("MONTH", "YEAR"))[,c(1:3, 5:6, 8:9, 11)]
  library(lubridate)
  #x.anomaly£Ym1 = as.Date(paste(x.anomaly£YEAR, x.anomaly£MONTH), format="%Y %m")
  x.anomaly$Ym1 =  lubridate::myd(paste(x.anomaly$MONTH, x.anomaly$YEAR, "1"))
  str(x.anomaly)
  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.clean= 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 QA/QC

4.1 Missing Data

```
# determine percent missing in station1
station1.TMAX.coverage = sum(!is.na(station1$VALUE[station1$ELEMENT=="TMAX"]))/length(static

# function to determine percent missing
coverage.fun <- function(station, element){
  Dates.all = data.frame(Ymd=seq.Date(from=min(station$Ymd), to=max(station$Ymd), by="day"))
  station.full = merge(Dates.all, station, all = TRUE)
  station.coverage = sum(!is.na(station.full$VALUE[station.full$ELEMENT==element]))/
    length(station.full$VALUE[station.full$ELEMENT==element])*100
  return(round(station.coverage,2))
}

coverage.data(station1, "TMAX")
coverage.data(station2, "TMAX")

Date.full = data.frame(Ymd=seq.Date(from=min(station1$Ymd), to=max(station1$Ymd), by="day"))
str(Date.full)

station1.full = merge(Date.full, station1, all = TRUE)
coverage.fun(station1, "TMAX")
coverage.fun(station2, "TMAX")
```

5 Next Steps

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

I'll save all the station data into csv files, then use them in the next guide to clean, process, and visualize data.

```
write.csv(station1, file = paste0(here::here("04_Regional_Climate_Trends", "Data", "SP24", '
write.csv(station2, file = paste0(here::here("04_Regional_Climate_Trends", "Data", "SP24", '
write.csv(station3, file = paste0(here::here("04_Regional_Climate_Trends", "Data", "SP24", '
write.csv(station4, file = paste0(here::here("04_Regional_Climate_Trends", "Data", "SP24", '
write.csv(station5, file = paste0(here::here("04_Regional_Climate_Trends", "Data", "SP24", '
```