STA 141B Solar

Robert Baranic

2023-04-17

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(plyr)
```

```
## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(janitor)
```

```
##
## Attaching package: 'janitor'

## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

```
library(tidyr)
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(stringr)
library(ggplot2)
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

First, I listed all of the zip files in the working directory so that I could unzip them all at once to access all the files needed.

```
zip = list.files(pattern = "\\.zip$") #https://stackoverflow.com/questions/4876813/using-r-to-list-all-
#list only files with .zip extension so we can unzip
```

```
lapply(zip, unzip) #unzip all zip files in directory
```

```
## [[1]]
## [1] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.clm"
## [2] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.ddy"
## [3] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.epw"
## [4] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.pvsyst"
## [5] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.rain"
## [6] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.stat"
## [7] "./USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.wea"
##
## [[2]]
## [1] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.clm"
## [2] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.ddy"
## [3] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.epw"
## [4] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.pvsyst"
## [5] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.rain"
## [6] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.stat"
## [7] "./USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.wea"
##
## [[3]]
## [1] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.clm"
## [2] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.ddy"
```

```
## [3] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.epw"
## [4] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.pvsyst"
## [5] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.rain"
## [6] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.stat"
## [7] "./USA_CA_Napa.County.AP.724955_TMYx.2007-2021.wea"
##
## [[4]]
## [1] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.clm"
## [2] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.ddy"
## [3] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.epw"
## [4] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.pvsyst"
## [5] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.rain"
## [6] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.stat"
## [7] "./USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.wea"
##
## [[5]]
## [1] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.clm"
## [2] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.ddy"
## [3] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.epw"
## [4] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.pvsyst"
## [5] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.rain"
## [6] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.stat"
## [7] "./USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.wea"
```

When all the files are unzipped, we can access all the files of a given type so we can apply our functions all at once. Here, we are working on the .wea files first.

```
w = list.files(pattern = "\\.wea$")
```

```
w = lapply(w, readLines)
```

By readinng the head of the .wea files, we find a pattern immediately: the data starts at line 7, there are nno headers, and the are separated by white space.

```
readLines("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.wea", n=10)
```

```
##  [1] "place Fairfield-San.Francisco.Bay.Reserve_USA"
##  [2] "latitude 38.20"
##  [3] "longitude 122.03"
##  [4] "time_zone 120"
##  [5] "site_elevation 4.0"
##  [6] "weather_data_file_units 1"
##  [7] "1 1 1.000 0 0"
##  [8] "1 1 2.000 0 0"
##  [9] "1 1 3.000 0 0"
## [10] "1 1 4.000 0 0"
```

```
wea = readLines("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.wea")
wea2 = wea[-1:-6]
```

To verify the data, we first check the basic characteristics: it is a dataframe with 5 variables and there are no NA values in the data. Upon visual inspection, the entries appear to be the same. A quick look at frequency

tables for columns 1 and 2 show that the data is likely observations taken 24 times a day (once per hour), everyday. This is consistent with the frequencies observed.

```
wea = read.table(textConnection(wea2))
class(wea)
```

```
## [1] "data.frame"
```

```
dim(wea)
```

```
## [1] 8760    5
```

```
any(is.na(wea))
```

```
## [1] FALSE
```

```
table(wea$V2)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288
##  21  22  23  24  25  26  27  28  29  30  31
## 288 288 288 288 288 288 288 288 264 264 168
```

```
table(wea$V1)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12
## 744 672 744 720 744 720 744 744 720 744 720 744
```

Next, we perform the same process on the .pvsyst files in order to apply our read functions across all the data files at once.

```
p = list.files(pattern = "\\.pvsyst$")
```

From looking at the head of the file, we can see the data starts at line 15 and our header is on line 13, so we extract line 13 and save it as a vector so we can use them in our read.table function. Class and dim report the same result as the .wea file which suggests it is also collected hourly. The table of day numbers also line up with the .wea file. There are also no NA values.

```
lines = readLines("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.pvsyst")
head(lines, n = 20)
```

```
##  [1] "#TMY hourly data"
##  [2] "#Standard format for importing hourly data in PVsyst"
##  [3] "#Created from EnergyPlus Weather Converter version=2022.11.03"
##  [4] "#WMO=998011Data Source=SRC-TMYx"
##  [5] "#Site,Fairfield-San.Francisco.Bay.Reserve"
##  [6] "#Country,USA"
```

```
##  [7] "#Data Source,SRC-TMYx WMO=998011"
##  [8] "#Time step,Hour"
##  [9] "#Latitude,38.200"
## [10] "#Longitude,-122.026"
## [11] "#Altitude,4"
## [12] "#Time Zone,-8.00"
## [13] "Year,Month,Day,Hour,Minute,GHI,DHI,DNI,Tamb,WindVel,WindDir"
## [14] ",,,,,W/m2,W/m2,W/m2,deg.C,m/sec,\xb0"
## [15] "2059,1,1,1,30,0,0,0,5.000,4.00,90"
## [16] "2059,1,1,2,30,0,0,0,4.000,4.00,40"
## [17] "2059,1,1,3,30,0,0,0,3.000,5.00,70"
## [18] "2059,1,1,4,30,0,0,0,1.000,2.00,40"
## [19] "2059,1,1,5,30,0,0,0,1.000,3.00,80"
## [20] "2059,1,1,6,30,0,0,0,1.000,3.00,70"
```

```
lines[13]
```

```
## [1] "Year,Month,Day,Hour,Minute,GHI,DHI,DNI,Tamb,WindVel,WindDir"
```

```
strsplit(lines[13], split = ",")
```

```
## [[1]]
##  [1] "Year"    "Month"   "Day"     "Hour"    "Minute"  "GHI"     "DHI"
##  [8] "DNI"     "Tamb"    "WindVel" "WindDir"
```

```
names = as.vector(strsplit(lines[13], split = ","))
data = lines[15:length(lines)]
pv = read.table(textConnection(data), sep = ",", col.names = names[[1]])
class(pv)
```

```
## [1] "data.frame"
```

```
dim(pv)
```

```
## [1] 8760   11
```

```
table(pv$Day)
```

```
##
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
## 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288 288
##  21  22  23  24  25  26  27  28  29  30  31
## 288 288 288 288 288 288 288 288 264 264 168
```

```
any(is.na(pv))
```

```
## [1] FALSE
```

.stat:

```
list.files(pattern = "\\.stat$")
```

```
## [1] "USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.stat"
## [2] "USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.stat"
## [3] "USA_CA_Napa.County.AP.724955_TMYx.2007-2021.stat"
## [4] "USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.stat"
## [5] "USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.stat"
```

```
ll = readLines("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.stat")
```

To make it simpler, I broke down the functions to read in the monthly data and the hourly data separately, but the processes are similar to begin with. #Monthly: First, we must find the start of each table. This is done with regular expressions, matching the start with the exact title, putting it into a vector. We verify this along the way by subsetting the lines with the vector and it outputs only the titles of the tables. We can also clearly see that the line numbers where they start are not the same and are some distance between each other.

```
monthStart = c(grep("Monthly Statistics for Dry Bulb temperatures", ll, useBytes = T),
               grep("Monthly Statistics for Dew Point temperatures", ll, useBytes = T),
               grep("Monthly Wind Direction", ll, useBytes = T),
               grep("Monthly Statistics for Wind Speed", ll, useBytes = T))
monthStart
```

```
## [1] 138 172 411 430
```

```
ll[monthStart]
```

```
## [1] " - Monthly Statistics for Dry Bulb temperatures [C]"
## [2] " - Monthly Statistics for Dew Point temperatures [C]"
## [3] " - Monthly Wind Direction {Interval 11.25 deg from displayed deg) [%]"
## [4] " - Monthly Statistics for Wind Speed [m/s]"
```

To find the end of the tables, we did another regular expression to match exactly to a few strings. By looking through the file, I noticed that each table was followed by either ” - Monthly” or ” - Average” which would be the start to another table, or it listed maximum and minimum values, so we also looked for ” - Maximum”. We passed this through a for loop since this was easier for me to visualize with the amount of conditional operations. We find the smallest line number in which any of those strings is greater than the line number for the start of the table, signifying the end of that table. By printing out the lines, I manually verified that these were the ends of the table.

```
monthEnd = c()
for (i in 1 : length(monthStart)) {
  monthEnd = c(monthEnd, min(grep("- Monthly|- Average|- Maximum", ll, useBytes = T)[grep("- Monthly|- A
  print(monthEnd)
}
```

```
## [1] 162
## [1] 162 182
## [1] 162 182 430
## [1] 162 182 430 440
```

```
ll[monthEnd]
```

```
## [1] "   - Maximum Dry Bulb temperature of  37.8C on Jun  8"
## [2] "   - Maximum Dew Point temperature of  18.3C on Aug 15"
## [3] " - Monthly Statistics for Wind Speed [m/s]"
## [4] "   - Maximum Wind Speed of  12.4 m/s on Dec 31"
```

Now we can read in the data by using our start and end indices, separating at tabs. However, the start and end of lines also use tabs, so this will create two blank columns. We use remove_empty from janitor package to clear these columns since they contain no information. I manually verified the data from this process and observed some patterns. All the tables were in a similar format except for the monthly wind statistics, this will be addressed when the function is created.

```
monthlyTables = ll[(monthStart[1] + 1): (monthEnd[1] -1)]
monthlyTables = monthlyTables[monthlyTables != ""]
monthlyTables
```

```
##  [1] " \t           \tJan\tFeb\tMar\tApr\tMay\tJun\tJul\tAug\tSep\tOct\tNov\tDec\t"
##  [2] " \tMaximum \t 24.4\t 24.7\t 24.1\t 31.4\t 33.4\t 37.8\t 36.9\t 33.9\t 35.3\t 31.3\t 26.0\t 17.3
##  [3] " \t Day:Hour\t31:16\t23:17\t13:18\t22:18\t21:16\t 8:17\t10:15\t18:16\t10:16\t17:16\t 7:17\t25:1
##  [4] " \tMinimum \t -0.8\t  0.9\t  2.6\t  3.1\t  6.8\t 11.6\t 11.7\t 11.4\t 12.7\t  6.0\t  2.9\t -0.1
##  [5] " \t Day:Hour\t 2:07\t27:07\t 7:06\t 6:07\t13:06\t19:06\t16:05\t 5:07\t 7:07\t28:08\t12:08\t29:0
##  [6] " \tDaily Avg\t9.4  \t10.9 \t13.0 \t14.0 \t17.3 \t19.8 \t20.3 \t19.5 \t19.8 \t17.0 \t12.7 \t9.0
##  [7] " \tDaily Range\t9.5  \t10.8 \t11.3 \t10.5 \t13.3 \t13.6 \t15.8 \t12.8 \t12.5 \t11.3 \t9.5  \t10
##  [8] " \tDayTime Max\t24.4 \t24.7 \t24.1 \t31.4 \t33.4 \t37.8 \t36.9 \t33.9 \t35.3 \t31.3 \t26.0 \t17
##  [9] " \tDayTime Min\t-0.8 \t0.9  \t3.1  \t3.1  \t7.4  \t12.7 \t12.3 \t11.4 \t12.7 \t6.0  \t2.9  \t-0
## [10] " \tDayTime Avg\t10.6 \t12.3 \t14.7 \t15.8 \t20.6 \t22.5 \t23.7 \t21.9 \t22.2 \t18.8 \t14.1 \t10
## [11] " \tNightTime Max\t17.8 \t21.0 \t23.1 \t30.3 \t28.3 \t34.6 \t34.2 \t31.1 \t29.5 \t25.4 \t22.7 \t
## [12] " \tNightTime Min\t-0.3 \t1.4  \t2.6  \t3.3  \t6.8  \t11.6 \t11.7 \t11.6 \t12.8 \t6.5  \t3.0  \t
## [13] " \tNightTime Avg\t8.2  \t9.4  \t11.3 \t12.1 \t14.0 \t17.1 \t16.9 \t17.0 \t17.5 \t15.2 \t11.3 \t
```

```
df = read.table(textConnection(monthlyTables), header = T, sep = "\t")
df = remove_empty(df)
```

```
## value for "which" not specified, defaulting to c("rows", "cols")
```

```
df
```

```
##              X.1   Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct
## 1       Maximum   24.4  24.7  24.1  31.4  33.4  37.8  36.9  33.9  35.3  31.3
## 2       Day:Hour 31:16 23:17 13:18 22:18 21:16  8:17 10:15 18:16 10:16 17:16
## 3       Minimum   -0.8   0.9   2.6   3.1   6.8  11.6  11.7  11.4  12.7   6.0
## 4       Day:Hour  2:07 27:07  7:06  6:07 13:06 19:06 16:05  5:07  7:07 28:08
## 5      Daily Avg 9.4   10.9  13.0  14.0  17.3  19.8  20.3  19.5  19.8  17.0
## 6    Daily Range 9.5   10.8  11.3  10.5  13.3  13.6  15.8  12.8  12.5  11.3
## 7    DayTime Max 24.4  24.7  24.1  31.4  33.4  37.8  36.9  33.9  35.3  31.3
## 8    DayTime Min -0.8  0.9   3.1   3.1   7.4   12.7  12.3  11.4  12.7  6.0
## 9    DayTime Avg 10.6  12.3  14.7  15.8  20.6  22.5  23.7  21.9  22.2  18.8
## 10 NightTime Max 17.8  21.0  23.1  30.3  28.3  34.6  34.2  31.1  29.5  25.4
## 11 NightTime Min -0.3  1.4   2.6   3.3   6.8   11.6  11.7  11.6  12.8  6.5
## 12 NightTime Avg 8.2   9.4   11.3  12.1  14.0  17.1  16.9  17.0  17.5  15.2
```

```
##      Nov   Dec
## 1   26.0  17.3
## 2   7:17 25:16
## 3    2.9  -0.2
## 4  12:08 29:08
## 5  12.7  9.0
## 6  9.5   10.0
## 7  26.0  17.3
## 8  2.9   -0.2
## 9  14.1  10.6
## 10 22.7  13.1
## 11 3.0   0.7
## 12 11.3  7.4
```

We must then transpose all the tables. The old row names are used as the column names, so we can now remove them from the data. I updated the names of column 2 and 4 so we do not have duplicate names, reducing errors while writing functions. Time was then converted to POSIXct for all the tables except for the wind table which did not contain any time information. A conditional if statement was implemented in the final function to avoid performing these operations on this table. Time was converted with the functions below and all the columns except for the time columns were converted to numerics.

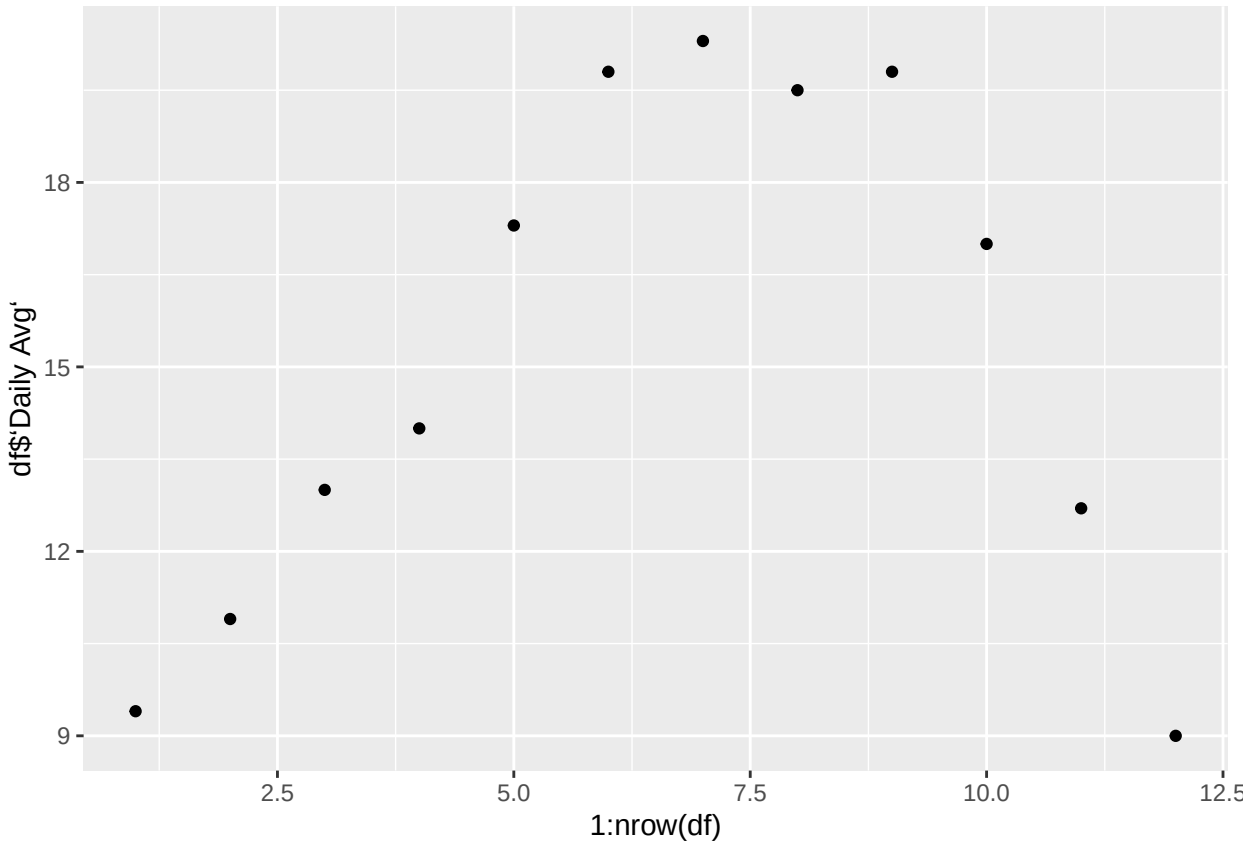
```
df = as.data.frame(t(df))
colnames(df) <- as.character(df[1, ])
df = df[-1, ]

names(df)[2] <- "Maximum Time"
names(df)[4] <- "Minimum Time"

dateMax = sprintf("%s/%s/%s/%s", 2023, rownames(df), sapply(strsplit(df$`Maximum Time`, split = ":"), "

dateMaxPos = as.POSIXct(strptime(dateMax, "%Y/%b/%d/%H"))

dateMin = sprintf("%s/%s/%s/%s", 2023, rownames(df), sapply(strsplit(df$`Minimum Time`, split = ":"), "

dateMinPos = as.POSIXct(strptime(dateMin, "%Y/%b/%d/%H"))

df$`Maximum Time` <- dateMaxPos
df$`Minimum Time` <- dateMinPos

df[, c(-2, -4)] <- sapply(df[, c(-2, -4)], as.numeric)
df
```

```
##     Maximum         Maximum Time Minimum         Minimum Time Daily Avg
## Jan    24.4 2023-01-31 16:00:00    -0.8 2023-01-02 07:00:00       9.4
## Feb    24.7 2023-02-23 17:00:00     0.9 2023-02-27 07:00:00      10.9
## Mar    24.1 2023-03-13 18:00:00     2.6 2023-03-07 06:00:00      13.0
## Apr    31.4 2023-04-22 18:00:00     3.1 2023-04-06 07:00:00      14.0
## May    33.4 2023-05-21 16:00:00     6.8 2023-05-13 06:00:00      17.3
## Jun    37.8 2023-06-08 17:00:00    11.6 2023-06-19 06:00:00      19.8
## Jul    36.9 2023-07-10 15:00:00    11.7 2023-07-16 05:00:00      20.3
## Aug    33.9 2023-08-18 16:00:00    11.4 2023-08-05 07:00:00      19.5
## Sep    35.3 2023-09-10 16:00:00    12.7 2023-09-07 07:00:00      19.8
## Oct    31.3 2023-10-17 16:00:00     6.0 2023-10-28 08:00:00      17.0
## Nov    26.0 2023-11-07 17:00:00     2.9 2023-11-12 08:00:00      12.7
```

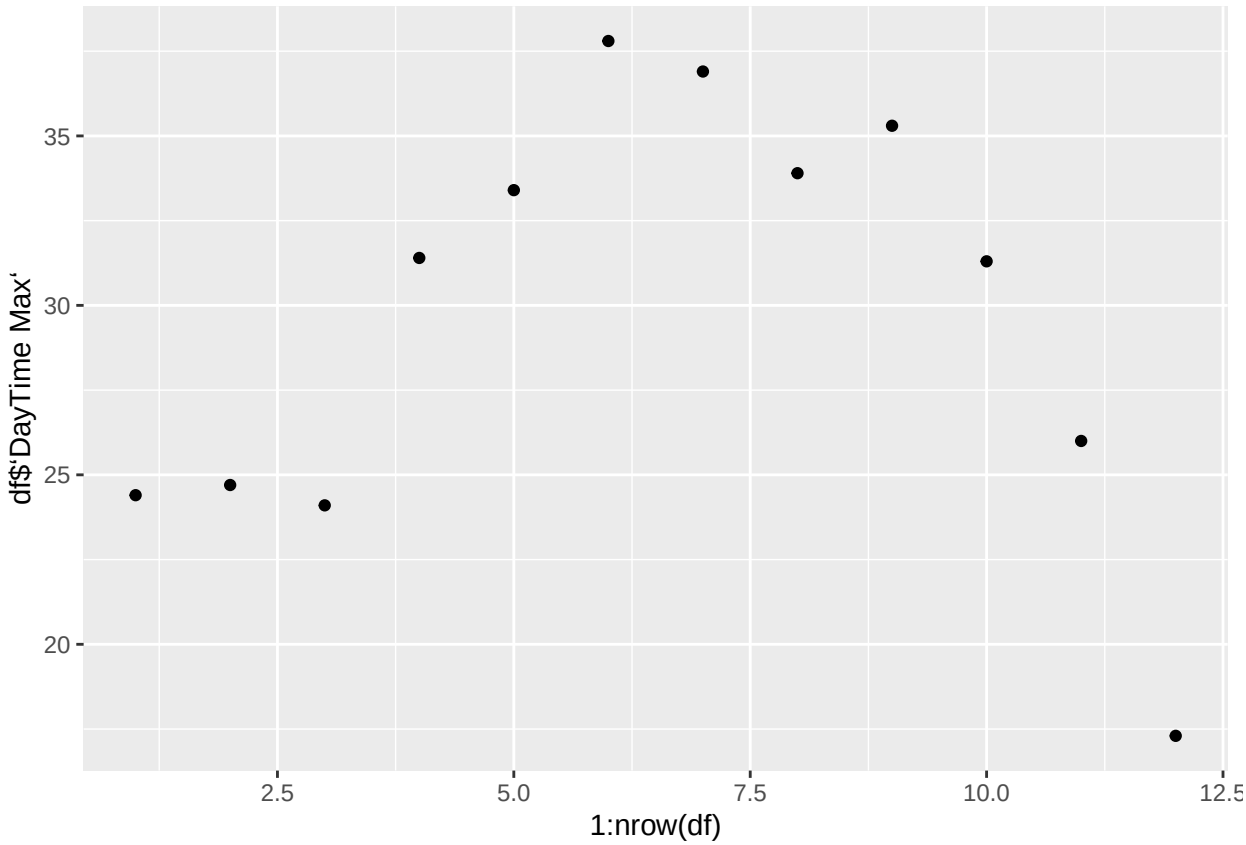
```
## Dec     17.3 2023-12-25 16:00:00     -0.2 2023-12-29 08:00:00       9.0
##     Daily Range DayTime Max DayTime Min DayTime Avg NightTime Max NightTime Min
## Jan         9.5        24.4        -0.8        10.6          17.8          -0.3
## Feb        10.8        24.7         0.9        12.3          21.0           1.4
## Mar        11.3        24.1         3.1        14.7          23.1           2.6
## Apr        10.5        31.4         3.1        15.8          30.3           3.3
## May        13.3        33.4         7.4        20.6          28.3           6.8
## Jun        13.6        37.8        12.7        22.5          34.6          11.6
## Jul        15.8        36.9        12.3        23.7          34.2          11.7
## Aug        12.8        33.9        11.4        21.9          31.1          11.6
## Sep        12.5        35.3        12.7        22.2          29.5          12.8
## Oct        11.3        31.3         6.0        18.8          25.4           6.5
## Nov         9.5        26.0         2.9        14.1          22.7           3.0
## Dec        10.0        17.3        -0.2        10.6          13.1           0.7
##     NightTime Avg
## Jan           8.2
## Feb           9.4
## Mar          11.3
## Apr          12.1
## May          14.0
## Jun          17.1
## Jul          16.9
## Aug          17.0
## Sep          17.5
## Oct          15.2
## Nov          11.3
## Dec           7.4
```

We can verify the data with some simple plots. These are consistent with what we would believe. Temperatures are highest in the summer months and lowest in the winter months.

```
ggplot(df) %>% +
  geom_point(aes(x = 1:nrow(df), y = df$`Daily Avg`))
```




```
ggplot(df) %>% +
  geom_point(aes(x = 1:nrow(df), y = df$`DayTime Max`))
```



#Hourly: Lastly, for the hourly data, we have the same process to begin with. Find the start of the tables with the exact titles, then find the end with the same for loop. Verified in the same manner, I manually checked that the ending of the tables was correct.

```
hourStart = c(grep("Average Hourly Statistics for Dry Bulb temperatures", ll, useBytes = T),
              grep("Average Hourly Statistics for Dew Point temperatures", ll, useBytes = T),
              grep("Average Hourly Relative Humidity", ll, useBytes = T),
              grep("Average Hourly Statistics for Direct Normal Solar Radiation", ll, useBytes = T),
              grep("Average Hourly Statistics for Wind Speed", ll, useBytes = T))
hourStart
```

```
## [1] 185 214 257 555 443
```

```
statName = ll[hourStart]
statName = str_replace_all(statName, c(" - Average Hourly Statistics for" = "", " - Average Hourly" = "
```

```
hourEnd = c()
for (i in 1 : length(hourStart)) {
  hourEnd = c(hourEnd, min(grep("- Monthly|- Average|- Maximum", ll, useBytes = T)[grep("- Monthly|- Av
  print(hourEnd)
}
```

```
## [1] 214
## [1] 214 243
## [1] 214 243 286
## [1] 214 243 286 584
## [1] 214 243 286 584 472
```

Here we read in the data the same way too. It is separated by tabs at the beginning and end, so we use the same method to remove the empty columns.

```
hourlyTables = ll[(hourStart[3] + 1): (hourEnd[3] - 1)]
hourlyTables = hourlyTables[hourlyTables != ""]
df2 = read.table(textConnection(hourlyTables), header = T, sep = "\t")
df2 = remove_empty(df2)
```

```
## value for "which" not specified, defaulting to c("rows", "cols")
```

```
df2
```

```
##            X.1 Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1   0:01- 1:00  90  79  80  82  84  79  80  83  83  81  89  85
## 2   1:01- 2:00  92  81  82  83  86  81  82  85  84  82  89  85
## 3   2:01- 3:00  93  82  84  83  87  83  84  87  86  82  89  87
## 4   3:01- 4:00  93  84  85  84  89  84  85  88  88  84  91  87
## 5   4:01- 5:00  93  85  86  84  89  86  86  88  88  85  91  88
## 6   5:01- 6:00  93  85  86  85  90  86  86  89  89  85  92  87
## 7   6:01- 7:00  94  83  87  86  87  85  85  88  90  86  93  87
## 8   7:01- 8:00  94  84  87  83  78  79  79  79  88  88  93  88
## 9   8:01- 9:00  93  81  84  76  69  74  72  75  82  85  92  87
## 10  9:01-10:00  92  75  73  70  61  67  65  69  75  76  87  84
## 11 10:01-11:00  89  65  65  64  56  60  58  64  65  66  80  76
## 12 11:01-12:00  84  58  58  59  52  54  50  57  57  60  72  69
## 13 12:01-13:00  78  51  53  53  49  49  45  52  51  55  66  64
## 14 13:01-14:00  73  48  50  50  48  46  42  42  45  53  63  59
## 15 14:01-15:00  69  46  48  47  48  43  40  42  42  50  60  57
## 16 15:01-16:00  67  45  48  48  49  42  40  41  42  50  59  56
## 17 16:01-17:00  66  45  48  49  50  43  41  43  45  51  60  59
## 18 17:01-18:00  71  49  49  53  53  45  44  46  51  56  66  68
## 19 18:01-19:00  76  57  55  59  60  48  48  52  59  64  72  74
## 20 19:01-20:00  82  65  64  68  69  54  56  66  68  69  76  77
## 21 20:01-21:00  86  69  69  75  74  63  64  73  71  72  81  79
## 22 21:01-22:00  89  73  73  77  77  68  70  77  75  76  84  81
## 23 22:01-23:00  91  75  76  80  80  73  74  81  77  79  87  81
## 24 23:01-24:00  91  77  78  82  83  77  77  82  80  81  88  82
## 25  Max Hour     7   5   7   7   6   6   5   6   7   8   7   5
## 26  Min Hour    17  17  16  15  14  16  15  16  16  16  16  16
```

```
df2 = df2[-25:-26, ]
df2
```

```
##            X.1 Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 1   0:01- 1:00  90  79  80  82  84  79  80  83  83  81  89  85
## 2   1:01- 2:00  92  81  82  83  86  81  82  85  84  82  89  85
## 3   2:01- 3:00  93  82  84  83  87  83  84  87  86  82  89  87
## 4   3:01- 4:00  93  84  85  84  89  84  85  88  88  84  91  87
## 5   4:01- 5:00  93  85  86  84  89  86  86  88  88  85  91  88
## 6   5:01- 6:00  93  85  86  85  90  86  86  89  89  85  92  87
## 7   6:01- 7:00  94  83  87  86  87  85  85  88  90  86  93  87
## 8   7:01- 8:00  94  84  87  83  78  79  79  79  88  88  93  88
```

```
## 9   8:01- 9:00  93  81  84  76  69  74  72  75  82  85  92  87
## 10  9:01-10:00  92  75  73  70  61  67  65  69  75  76  87  84
## 11 10:01-11:00  89  65  65  64  56  60  58  64  65  66  80  76
## 12 11:01-12:00  84  58  58  59  52  54  50  57  57  60  72  69
## 13 12:01-13:00  78  51  53  53  49  49  45  52  51  55  66  64
## 14 13:01-14:00  73  48  50  50  48  46  42  42  45  53  63  59
## 15 14:01-15:00  69  46  48  47  48  43  40  42  42  50  60  57
## 16 15:01-16:00  67  45  48  48  49  42  40  41  42  50  59  56
## 17 16:01-17:00  66  45  48  49  50  43  41  43  45  51  60  59
## 18 17:01-18:00  71  49  49  53  53  45  44  46  51  56  66  68
## 19 18:01-19:00  76  57  55  59  60  48  48  52  59  64  72  74
## 20 19:01-20:00  82  65  64  68  69  54  56  66  68  69  76  77
## 21 20:01-21:00  86  69  69  75  74  63  64  73  71  72  81  79
## 22 21:01-22:00  89  73  73  77  77  68  70  77  75  76  84  81
## 23 22:01-23:00  91  75  76  80  80  73  74  81  77  79  87  81
## 24 23:01-24:00  91  77  78  82  83  77  77  82  80  81  88  82
```

We use pivot_longer in the tidyr package to mutate the data into the form that we want. However, we run into a problem where the data is combined such that all the like hours are grouped, but this results in our data not being in chronological order, which is what we would likely desire. The way I solved this was by setting the months to be factor levels and ordering them based on factor level. This will descend the data starting with Jan, Feb, etc. We then overwrite the time column to be 0:23 instead of the time intervals in order to verify data easier. The data is verified at the end in the plots where all the tables are combined.

```
df2$X.1 <- 0:23
df3 = df2
df3 <- df3 %>%
  mutate_all(as.numeric) %>%
  pivot_longer(cols = !X.1)

df3$name <- factor(df3$name, levels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep",
df3 = df3[order(as.factor(df3$name)), ]
colnames(df3) <- c("Time", "Month", statName[3])

df3
```

```
## # A tibble: 288 x 3
##     Time Month ‘ Relative Humidity [%]‘
##    <dbl> <fct>                    <dbl>
##  1     0 Jan                         90
##  2     1 Jan                         92
##  3     2 Jan                         93
##  4     3 Jan                         93
##  5     4 Jan                         93
##  6     5 Jan                         93
##  7     6 Jan                         94
##  8     7 Jan                         94
##  9     8 Jan                         93
## 10     9 Jan                         92
## # ... with 278 more rows
```

#Functions:

##.wea: This is the simplest function. We just input the file name and it reads the lines, skipping the first 6. This is simply the combining of the steps used to read in the .wea file.

```
readWea = function(file) {
  lines = readLines(file)
  wea = lines[-1:-6]
  con = textConnection(wea)
  read.table(con)
}
```

##.pvsyst: This is also a very simple function that combines the two steps of reading the .pvsyst file. First we save the names of the column, then we read in the data with the column names.

```
readPvsyst = function(file) {
  lines = readLines(file)
  names = as.vector(strsplit(lines[13], split = ","))
  data = lines[15:length(lines)]
  read.table(textConnection(data), sep = ",", col.names = names[[1]])
}

pv = lapply(p, readPvsyst)
```

##.stat: For the .stat file, I broke it up into the monthly data and the hourly data to avoid having an even more complicated set of functions. ##Monthly: I broke the monthly function into two separate functions. One to read the actual data into the dataframe given the start and ends of the table, and the second one use mapply to read in all the tables at once given the file name.

For the first function, it is the same process that was described above except we now have an if conditional statement that will avoid trying to convert any times if there is not a column called "Maximum" in the original table. This will prevent the wind table from going through the time converting processes, but the other tables will still go through it. Otherwise it is the same and works for all the files.

```
readMonthlyBetween = function(lines, monthStart, monthEnd) {
  monthlyTables = lines[(monthStart + 1): (monthEnd - 1)]
  monthlyTables = monthlyTables[monthlyTables != ""]

  df = read.table(textConnection(monthlyTables), header = T, sep = "\t")
  df = remove_empty(df)

  df = as.data.frame(t(df))
  colnames(df) <- as.character(df[1, ])
  df = df[-1, ]
  if (sum(((as.vector(colnames(df))) == "Maximum ")) >= 1 ){
      names(df)[2] <- "Maximum Time"
      names(df)[4] <- "Minimum Time"
      dateMax = sprintf("%s/%s/%s/%s", 2023, rownames(df), sapply(strsplit(df$`Maximum Time`, split = "

      dateMaxPos = as.POSIXct(strptime(dateMax, "%Y/%b/%d/%H"))

      dateMin = sprintf("%s/%s/%s/%s", 2023, rownames(df), sapply(strsplit(df$`Minimum Time`, split = "

      dateMinPos = as.POSIXct(strptime(dateMin, "%Y/%b/%d/%H"))

      df$`Maximum Time` <- dateMaxPos
      df$`Minimum Time` <- dateMinPos
```

```
      df[, c(-2, -4)] <- sapply(df[, c(-2, -4)], as.numeric)
  }
  else {
    df <- as.data.frame(sapply(df, as.numeric))
  }
  df
}

readStatMonthly = function(file) {
  lines = readLines(file)
  monthStart = c(grep("Monthly Statistics for Dry Bulb temperatures", lines, useBytes = T),
                 grep("Monthly Statistics for Dew Point temperatures", lines, useBytes = T),
                 grep("Monthly Wind Direction", lines, useBytes = T),
                 grep("Monthly Statistics for Wind Speed", lines, useBytes = T))
  monthEnd = c()
  for (i in 1 : length(monthStart)) {
    monthEnd = c(monthEnd, min(grep("- Monthly|- Average|- Maximum", lines, useBytes = T)[grep("- Monthl
  }
  mapply(readMonthlyBetween, monthStart, monthEnd, MoreArgs = list(lines = lines))
}

w = readStatMonthly("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.stat")
```

```
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
```

```
w = readMonthlyBetween(ll, monthStart = monthStart, monthEnd = monthEnd)
```

```
## Warning in (monthStart + 1):(monthEnd - 1): numerical expression has 4 elements:
## only the first used

## Warning in (monthStart + 1):(monthEnd - 1): numerical expression has 4 elements:
## only the first used

## value for "which" not specified, defaulting to c("rows", "cols")
```

##Hourly: For the hourly data, it is split into two functions as well, doing similar things in each. The first function reads in all the tables given the start and ending point as well as the name of the stat so we can name the column. This is necessary so that in the second function, we can correctly name the column of the stat when we place all the data into one dataframe. This process is verified by the graphs at the end.

```
readHourlyBetween = function(lines, hourStart, hourEnd, statName){

  hourlyTables = lines[(hourStart + 1): (hourEnd - 1)]
  hourlyTables = hourlyTables[hourlyTables != ""]
  hourlyDF = read.table(textConnection(hourlyTables), header = T, sep = "\t")
  hourlyDF = remove_empty(hourlyDF)
  hourlyDF = hourlyDF[-25:-26, ]
```

```
  hourlyDF$X.1 <- 0:23
  hourlyDF2 = hourlyDF
  hourlyDF2 <- hourlyDF2 %>%
    mutate_all(as.numeric) %>%
    pivot_longer(cols = !X.1)

  hourlyDF2$name <- factor(hourlyDF2$name, levels = c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
  hourlyDF2 = hourlyDF2[order(as.factor(hourlyDF2$name)), ]
  colnames(hourlyDF2) <- c("Time", "Month", statName)

  hourlyDF2
}

readStatHourly = function(file) {
  lines = readLines(file)
  hourStart = c(grep("Average Hourly Statistics for Dry Bulb temperatures", lines, useBytes = T),
                grep("Average Hourly Statistics for Dew Point temperatures", lines, useBytes = T),
                grep("Average Hourly Relative Humidity", lines, useBytes = T),
                grep("Average Hourly Statistics for Direct Normal Solar Radiation", lines, useBytes = T
                grep("Average Hourly Statistics for Wind Speed", lines, useBytes = T))

  hourEnd = c()
  for (i in 1 : length(hourStart)) {
    hourEnd = c(hourEnd, min(grep("- Monthly|- Average|- Maximum", lines, useBytes = T)[grep("- Monthly
  }

  statName = lines[hourStart]
  statName = str_replace_all(statName, c(" - Average Hourly Statistics for" = "", " - Average Hourly" =

  list = mapply(readHourlyBetween, hourStart, hourEnd, statName, MoreArgs = list(lines = lines))

  totalHourly = data.frame(list[1, 1])
  totalHourly$Month = list[2, 1]$Month
  for (i in 1:length(statName)) {
    totalHourly[i+2] = list[3, i]
  }

  colnames(totalHourly) = c("Time", "Month", statName[1:length(statName)])
  totalHourly
}

w = readStatHourly("USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.stat")
```

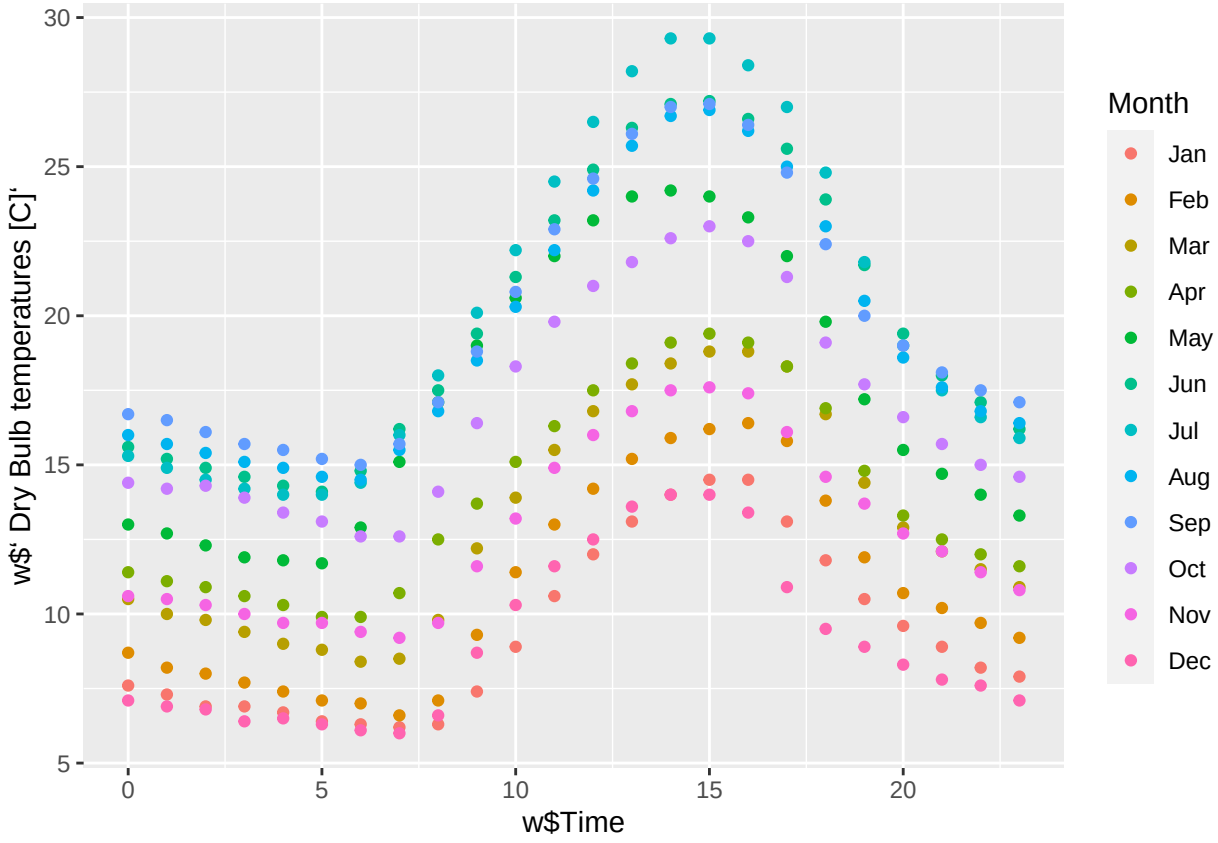
```
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
## value for "which" not specified, defaulting to c("rows", "cols")
```

#Plots Here we plot all the statistics against time and colored by month. All follow what we would intuitively assume about there respective statistics, giving us some verification of the data.

```
w %>% ggplot() +
  geom_point(aes(y = w$` Dry Bulb temperatures [C]`, x = w$Time, color = Month))
```

```
## Warning: Use of ‘w$Time‘ is discouraged.
## i Use ‘Time‘ instead.
```

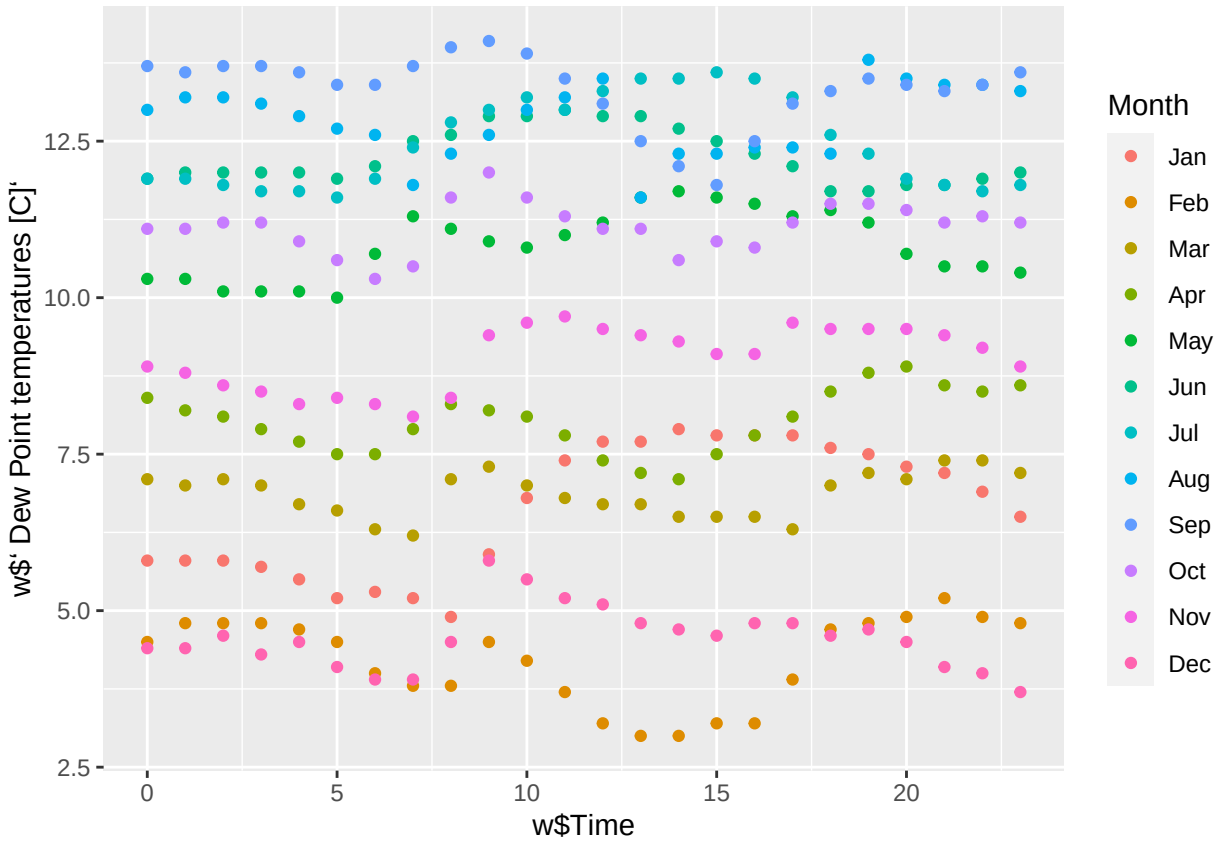
```
## Warning: Use of ‘‘ w$‘ Dry Bulb temperatures [C]‘ ‘‘ is discouraged.
## i Use ‘ Dry Bulb temperatures [C]‘ instead.
```




```
w %>% ggplot() +
  geom_point(aes(y = w$` Dew Point temperatures [C]`, x = w$Time, color = Month))
```

```
## Warning: Use of ‘w$Time‘ is discouraged.
## i Use ‘Time‘ instead.
```

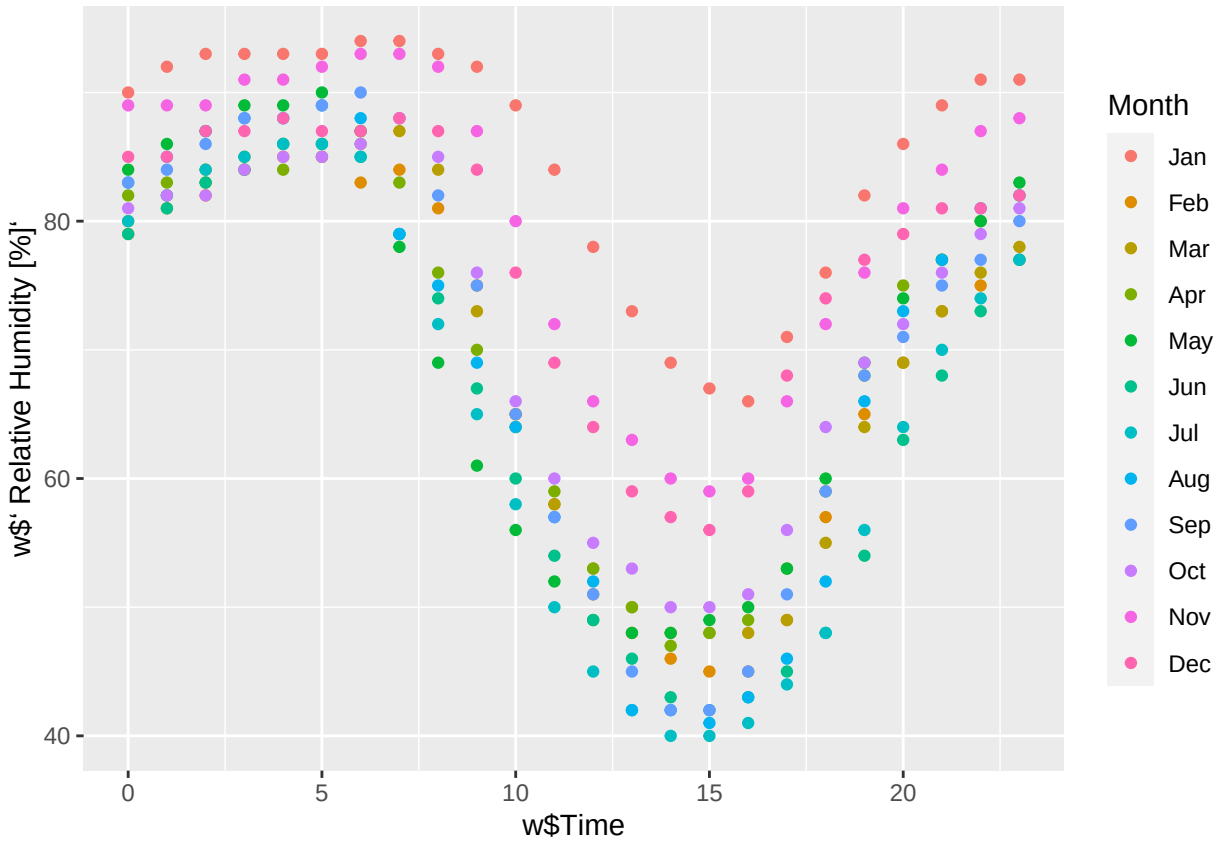
```
## Warning: Use of ‘‘ w$‘ Dew Point temperatures [C]‘ ‘‘ is discouraged.
## i Use ‘ Dew Point temperatures [C]‘ instead.
```




```
w %>% ggplot() +
  geom_point(aes(y = w$` Relative Humidity [%]`, x = w$Time, color = Month))
```

```
## Warning: Use of ‘w$Time‘ is discouraged.
## i Use ‘Time‘ instead.
```

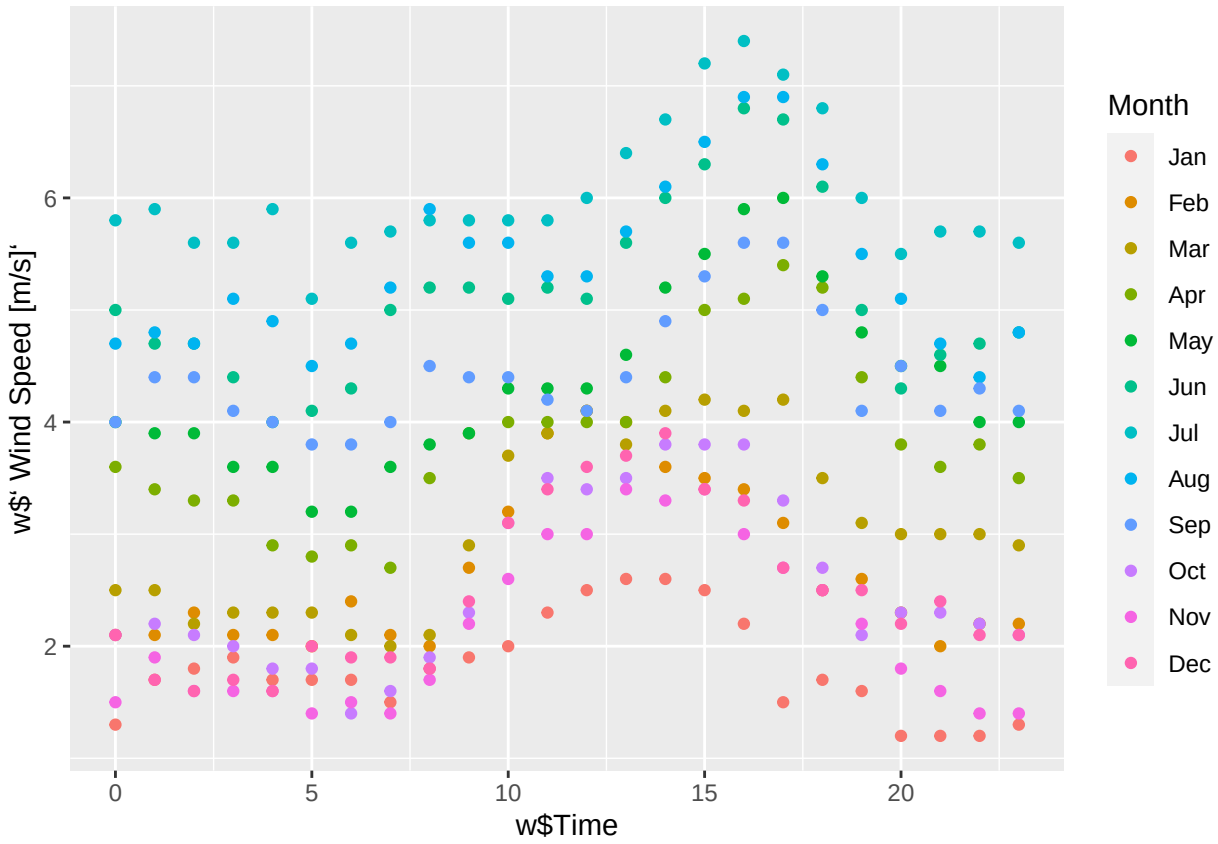
```
## Warning: Use of ‘‘ w$‘ Relative Humidity [%]‘ ‘‘ is discouraged.
## i Use ‘ Relative Humidity [%]‘ instead.
```




```
w %>% ggplot() +
  geom_point(aes(y = w$` Wind Speed [m/s]`, x = w$Time, color = Month))
```

```
## Warning: Use of ‘w$Time‘ is discouraged.
## i Use ‘Time‘ instead.
```

```
## Warning: Use of ‘‘ w$‘ Wind Speed [m/s]‘ ‘‘ is discouraged.
## i Use ‘ Wind Speed [m/s]‘ instead.
```


w$‘ Wind Speed [m/s]‘
6
4
2
0
5
10
15
20
w$Time
Month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec