Reading Solar and Climate Rectangular Data - Eric Sun

Verifying WEA Files

Assumptions: The structure of the wea files for the 5 locations were consistent, with each starting with the information:

place - latitude - longitude - time_zone - site_elevation - weather_data_file_units

The data is also a singular rectangular dataset, and the files all have the same starting and ending structure.

Method: I read the data using read.table() and skipped those 6 lines for each file, which gave me a data frame with the data we wanted to extract. In addition, I checked whether there were NAs in the columns, and if that was true the columns would be removed for further inspection. To ensure that all of the data was transferred in properly, I manually went through the data in a text editor to evaluate the first and last few lines and compared it to the head() and tail() of my data frame to check for similarity.

Another method that I used to verify is identifying the pattern in recording the data and checking the dimensions. An observation (row count) was recorded for every hour of each day of each month for a year, which should yield 365 x 24 = 8760 values (unless there were missing values in the data). To verify this, I checked the dimensions of all my wea files and they had 8760 rows.

```
# WEA function
open_wea <- function(filepath) {
  data = read.table(filepath, skip = 6)
  colnames(data) = c(
    "Month",
    "Day",
    "Standard Time",
    "Direct Normal Irradiance",
    "Diffuse Horizontal Radiance"
  )
  data = data[ , colSums(is.na(data))==0] # checking for columns w/ NAs
  return(data)

}
```

```
wea_tables = lapply(
  c(
    "USA_CA_Fairfield/fairfield.wea",
    "USA_CA_Napa/napa.wea",
    "USA_CA_UC_Davis/davis.wea",
    "USA_CA_Point/point_reyes.wea",
    "USA_CA_Marin/marin.wea"
  ),
  open_wea
)
```

```
lapply(wea_tables, head)
```

```
## [[1]]
##   Month Day Standard Time Direct Normal Irradiance Diffuse Horizontal Radiance
## 1     1   1             1                         0                           0
## 2     1   1             2                         0                           0
## 3     1   1             3                         0                           0
## 4     1   1             4                         0                           0
## 5     1   1             5                         0                           0
## 6     1   1             6                         0                           0
##
## [[2]]
##   Month Day Standard Time Direct Normal Irradiance Diffuse Horizontal Radiance
## 1     1   1             1                         0                           0
## 2     1   1             2                         0                           0
## 3     1   1             3                         0                           0
## 4     1   1             4                         0                           0
## 5     1   1             5                         0                           0
## 6     1   1             6                         0                           0
##
## [[3]]
##   Month Day Standard Time Direct Normal Irradiance Diffuse Horizontal Radiance
## 1     1   1             1                         0                           0
## 2     1   1             2                         0                           0
## 3     1   1             3                         0                           0
## 4     1   1             4                         0                           0
## 5     1   1             5                         0                           0
## 6     1   1             6                         0                           0
##
## [[4]]
##   Month Day Standard Time Direct Normal Irradiance Diffuse Horizontal Radiance
## 1     1   1             1                         0                           0
## 2     1   1             2                         0                           0
## 3     1   1             3                         0                           0
## 4     1   1             4                         0                           0
## 5     1   1             5                         0                           0
## 6     1   1             6                         0                           0
##
## [[5]]
##   Month Day Standard Time Direct Normal Irradiance Diffuse Horizontal Radiance
## 1     1   1             1                         0                           0
## 2     1   1             2                         0                           0
## 3     1   1             3                         0                           0
## 4     1   1             4                         0                           0
## 5     1   1             5                         0                           0
## 6     1   1             6                         0                           0

## [1] "The order of numbers in [[]] correspond to: Fairfield, Napa, Davis, Point, and Marin."
```

Verifying Pvsyst Files

Assumptions: Similar to the wea. files, the pvsyst files all started with the following strings of text:

TMY hourly data - Standard format for importing hourly data in PVsyst - Created from EnergyPlus Weather Converter version=2022.11.03 - WMO=720576Data Source=SRC-TMYx - Site,UC-Davis-University.AP - Country,USA - Data Source,SRC-TMYx WMO=720576 - Time step,Hour - Latitude,38.533 - Longitude,-121.783 - Altitude,21 - Time Zone,-8.00 - Year,Month,Day,Hour,Minute,GHI,DHI,DNI,Tamb,WindVel,WindDir - „„,W/m2,W/m2,W/m2,deg.C,m/sec,

The data is also rectangular, with each file having the same start and end structure.

Method: I did the same process for pvsyst as for wea., however I skipped the first 14 lines for the pvsyst files.

```
# PVSYST function
open_pvsyst <- function(filepath) {
  data = read.table(filepath, sep = ",", skip = 14)
  colnames(data) = c(
    'Year',
    'Month',
    'Day',
    'Hour',
    'Minute',
    'GHI',
    'DHI',
    'DNI',
    'Tamb',
    'WindVel',
    'WindDir'
  )
  data = data[ , colSums(is.na(data))==0]
  return(data)
}
```

```
pvsyst_tables = lapply(
  c(
    "USA_CA_Fairfield/fairfield.pvsyst",
    "USA_CA_Napa/napa.pvsyst",
    "USA_CA_UC_Davis/davis.pvsyst",
    "USA_CA_Point/point_reyes.pvsyst",
    "USA_CA_Marin/marin.pvsyst"
  ),
  open_pvsyst
)
```

```
lapply(pvsyst_tables, head)
```

```
## [[1]]
##   Year Month Day Hour Minute GHI DHI DNI Tamb WindVel WindDir
## 1 2059     1   1    1     30   0   0   0    5       4      90
## 2 2059     1   1    2     30   0   0   0    4       4      40
## 3 2059     1   1    3     30   0   0   0    3       5      70
## 4 2059     1   1    4     30   0   0   0    1       2      40
## 5 2059     1   1    5     30   0   0   0    1       3      80
## 6 2059     1   1    6     30   0   0   0    1       3      70
##
## [[2]]
##   Year Month Day Hour Minute GHI DHI DNI Tamb WindVel WindDir
## 1 2059     1   1    1     30   0   0   0    9       4     240
## 2 2059     1   1    2     30   0   0   0   10       3     220
## 3 2059     1   1    3     30   0   0   0    7       3      20
## 4 2059     1   1    4     30   0   0   0    4       3     330
## 5 2059     1   1    5     30   0   0   0    3       0     236
## 6 2059     1   1    6     30   0   0   0    3       2     300
##
## [[3]]
##   Year Month Day Hour Minute GHI DHI DNI Tamb WindVel WindDir
## 1 2059     1   1    1     30   0   0   0    4       5     350
## 2 2059     1   1    2     30   0   0   0    4       5     340
## 3 2059     1   1    3     30   0   0   0    4       6     340
## 4 2059     1   1    4     30   0   0   0    5       7     340
## 5 2059     1   1    5     30   0   0   0    5       6     340
## 6 2059     1   1    6     30   0   0   0    5       8     340
##
## [[4]]
##   Year Month Day Hour Minute GHI DHI DNI Tamb WindVel WindDir
## 1 2059     1   1    1     30   0   0   0   12       3     350
## 2 2059     1   1    2     30   0   0   0   12       3      10
## 3 2059     1   1    3     30   0   0   0   12       2     280
## 4 2059     1   1    4     30   0   0   0   13       3      40
## 5 2059     1   1    5     30   0   0   0   14       6      80
## 6 2059     1   1    6     30   0   0   0   14       2      30
##
## [[5]]
##   Year Month Day Hour Minute GHI DHI DNI Tamb WindVel WindDir
## 1 2059     1   1    1     30   0   0   0    7       0     316
## 2 2059     1   1    2     30   0   0   0    5       2     170
## 3 2059     1   1    3     30   0   0   0    4       0     349
## 4 2059     1   1    4     30   0   0   0    5       2     190
## 5 2059     1   1    5     30   0   0   0    2       0      80
## 6 2059     1   1    6     30   0   0   0    2       0      98
```

```
## [1] "The order of numbers in [[]] correspond to: Fairfield, Napa, Davis, Point, and Marin."
```

Verifying Stat files (MONTHLY)

Assumptions: The files are TSV (Tab Separated Values). Upon initial inspection, it was clear that the data tables were separated by equal distances of space. I verified this through the read.table() function. If the separator was anything but tabs, we would lose data. In addition, it is a rectangular dataset.

Method: I first read each of the monthly tables. I used the grep function that took in two arguments: title, readLines(filepath) to extract the index of the title in the file, and used that index as the starting line for the read.table function. For finding the endpoint of a table (nrows), I manually counted the number of lines in the table. This worked because of the consistency across monthly data for the 5 locations. After that, I used the assumptions that the structure of the tables were TSV. Then, I manually checked the contents of the generated tables with the tables in the text file. Every table generated two extra columns with NA values that were not a part of the original data, therefore I removed them.

Everything else after involved data manipulation based on the instructions. I switched the row and column names by transposing the data frame. At this point, I manually checked whether the new table I generated into R had the same values as the text file. There were no NAs, and the values were correct. Then, I converted the Day:Hour columns into POSIXct values and checked whether the dates matched the correct format with lapply(data, class). Finally, I constructed a loop and turned the other columns besides Day:Hour in the df to numeric, and also checked by using lapply(data, class).

And for one last check, I used str(df) at the end to verify that the columns are of the correct class. I verified my results by comparing the values of the table I generated vs the table in the text file, and used head()/tail().

```
# modified function that reads in multiple titles
open_monthly <- function(filepath, titles) {
  data_list <- list()
  for (title in titles) {
    unique_id <- grep(title, readLines(filepath))
    if (title == "Monthly Statistics for Dry Bulb temperatures") {
      data <-
        read.table(
          filepath,
          sep = "\t",
          header = TRUE,
          skip = unique_id,
          nrows = 12
        )
    } else if (title == "Monthly Statistics for Dew Point temperatures" |
               title == "Monthly Statistics for Wind Speed") {
      data <-
        read.table(
          filepath,
          sep = "\t",
          header = TRUE,
          skip = unique_id,
          nrows = 5
        )
    } else {
      data <-
        read.table(
          filepath,
          sep = "\t",
```

```
      header = TRUE,
      skip = unique_id,
      nrows = 16
    )
}

data = data[, colSums(is.na(data)) == 0]

data = t(data) # transpose

colnames(data) = data[1,] # after transposing the matrix, previous row names
# that are now column names are read in as the first line of data, and we want
# these to be the actual column names of our data

data = data[-1,] # get rid of the column names contained within our data frame

data = as.data.frame(data)

for (i in 1:ncol(data)) {
  colnames(data)[i] <-
    trimws(colnames(data)[i]) # trim whitespaces in column names
}

while (title != "Monthly Wind Direction") {
  # not applied to monthly bc no time data
  for (i in 1:nrow(data)) {
    day_hour <-
      strsplit(as.character(data[i, 2]), ":")[[1]] # Chat GPT assisted in
    # developing this code. Essentially the time data is being split by day and hours

    day <- as.numeric(day_hour[1])
    hour <- as.numeric(day_hour[2])

    data[i, "Datetime_Max"] <-
      as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"),
                 format = "%Y-%m-%d %H:%M:%S")
  }

  for (i in 1:nrow(data)) {
    day_hour <- strsplit(as.character(data[i, 4]), ":")[[1]]
    day <- as.numeric(day_hour[1])
    hour <- as.numeric(day_hour[2])

    data[i, "Datetime_Min"] <-
      as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"),
                 format = "%Y-%m-%d %H:%M:%S")
  }

  data[, 2] <-
    data$Datetime_Max # replace old column that contained time data
  # with new column Datetime_Max created above
  data[, 4] <- data$Datetime_Min # replace with Datetime_Min
  data <-
```

```
        subset(data, select = -c(Datetime_Max, Datetime_Min)) # delete the
      # new columns bc we already inputted the data into the original column indexes

      break
    }

    for (i in 1:ncol(data)) {
      # change every column into numeric aside for the
      # time columns
      if (colnames(data)[i] == "Day:Hour" |
          colnames(data)[i] == "Day:Hour.1") {
        next
      }
      data[, i] <- as.numeric(data[, i])
    }

    data_list[[title]] <-
      data.frame(data) # returns a list of dataframes, based
    # on the number of titles
  }
  return(data_list)
}
```

Davis Monthly File

I only included the head of the Davis file because if I included the other locations it would be too tedious to read.

```
davis_monthly = open_monthly(
  "USA_CA_UC_Davis/davis.stat",
  c(
    "Monthly Statistics for Dry Bulb temperatures",
    "Monthly Statistics for Dew Point temperatures",
    "Monthly Wind Direction",
    "Monthly Statistics for Wind Speed"
  )
)
```

```
lapply(davis_monthly, head)
```

```
## $`Monthly Statistics for Dry Bulb temperatures`
##     Maximum            Day.Hour Minimum          Day.Hour.1 Daily.Avg
## Jan    19.0 2023-01-27 16:00:00    -1.0 2023-01-02 07:00:00       9.7
## Feb    24.5 2023-02-24 16:00:00     1.0 2023-02-03 09:00:00      10.9
## Mar    25.6 2023-03-19 17:00:00     1.5 2023-03-11 07:00:00      12.4
## Apr    31.0 2023-04-30 15:00:00     3.0 2023-04-06 05:00:00      15.9
## May    35.0 2023-05-22 15:00:00     7.0 2023-05-14 06:00:00      19.5
## Jun    39.0 2023-06-28 16:00:00     9.0 2023-06-17 03:00:00      22.9
##     Daily.Range DayTime.Max DayTime.Min DayTime.Avg NightTime.Max NightTime.Min
## Jan         8.3        19.0        -1.0        10.9          15.0             0
## Feb        10.7        24.5         1.0        12.4          22.8             2
## Mar        11.1        25.6         1.5        14.2          22.7             2
## Apr        14.6        31.0         3.0        18.7          30.0             3
## May        15.6        35.0         9.0        22.9          34.0             7
## Jun        18.2        39.0        12.0        26.8          37.0             9
##     NightTime.Avg
## Jan           8.5
## Feb           9.4
## Mar          10.6
## Apr          13.1
## May          16.0
## Jun          18.9
##
## $`Monthly Statistics for Dew Point temperatures`
##     Maximum            Day.Hour Minimum          Day.Hour.1 Daily.Avg
## Jan    14.0 2023-01-20 13:00:00    -8.0 2023-01-01 07:00:00       6.7
## Feb    13.0 2023-02-08 12:00:00   -25.0 2023-02-23 19:00:00       2.6
## Mar    12.3 2023-03-21 19:00:00    -4.6 2023-03-14 07:00:00       5.4
## Apr    13.0 2023-04-20 16:00:00   -11.0 2023-04-14 17:00:00       4.7
## May    17.0 2023-05-03 20:00:00    -5.0 2023-05-17 16:00:00       8.1
## Jun    15.0 2023-06-06 12:00:00    -2.0 2023-06-19 20:00:00       9.3
##
## $`Monthly Wind Direction`
##     N..0. ENE..22.5. NE..45. ENE..67.5. E..90. ESE..112.5. SE..135. SSE..157.5.
## Jan    15          7       2          2      3           5        9          13
## Feb    25          6       4          3      1           2        1           3
```

```
## Mar     5          1       1          1      1          2        5          7
## Apr    15          3       0          0      0          1        2          4
## May     8          2       1          1      1          1        2          3
## Jun    10          3       1          1      0          1        2          6
##     S..180. SSW..202.5. SW..225. WSW..247.5. W..270. WNW..292.5. NW..315.
## Jan      11           5       3           3       2           2        6
## Feb       9          10       8           4       1           3        4
## Mar      13          16      17           6       3           4        6
## Apr      13          15      14           7       5           5        4
## May      20          21      13           5       6           3        3
## Jun      25          20      14           8       3           1        1
##     NNW..337.5.
## Jan          11
## Feb          16
## Mar          14
## Apr          12
## May          10
## Jun           4
##
## $`Monthly Statistics for Wind Speed`
##     Maximum            Day.Hour Minimum          Day.Hour.1 Daily.Avg
## Jan    14.4 2023-01-06 20:00:00     0.0 2023-01-01 21:00:00       2.4
## Feb    12.3 2023-02-15 13:00:00     0.0 2023-02-01 21:00:00       2.3
## Mar     6.2 2023-03-24 19:00:00     0.1 2023-03-16 13:00:00       2.4
## Apr    10.3 2023-04-15 09:00:00     0.0 2023-04-01 16:00:00       3.0
## May    10.3 2023-05-07 04:00:00     0.0 2023-05-02 00:00:00       3.1
## Jun     8.8 2023-06-12 11:00:00     0.0 2023-06-01 04:00:00       2.6
```

Verifying Stat files (HOURLY)

Assumptions: The hourly tables had the same structure as the monthly tables. The main difference between the two is that the average hours have the same number of rows for each table. The consistent structure made it easier to read in and validate the data.

Method: For verifying that the Max and Min Hours for each column of the data was true, I built a nested for loop that went through every row observation in the table with respect to an individual column index i and created a counter that would update when the maximum value in the ith column was equal to the Max/Min_Hour index that was stored in the Max_Min hour variables. This accounted for any potential duplicated max/min hour values as well. If there were multiple rows with the true max/min value in the respective column, the counter would still update if the variable contained one of those max/min values.

Omitting the Max_Hour and Min_Hour rows was simple, as they were always the last two rows in the table.

All to do after was to combine the month, hours, and temperature values (based on title) into a data frame. For verifying the results of the table, I compared it to the table in the text file.

```
# modified version of hourly_individual
open_hourly <- function(filepath, titles) {
  data_list <- list()

  # construct counters for correct max hours
  correct_max = 0
  incorrect_max = 0

  # counters for correct min hours
  correct_min = 0
  incorrect_min = 0

  for (title in titles) {
    unique_id = grep(title, readLines(filepath))
    data = read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 26
    )

    data = data[, colSums(is.na(data)) == 0]
    data = as.data.frame(data)
    colnames(data)[1] = "Hours"
    colnames(data) = trimws(colnames(data))
    data[, 1] = gsub("[[:space:]]", "", data[, 1])

    for (i in 2:ncol(data)) { # skip the first column (time)
      if (class(data[25, i]) == "character") { # checking if value we want to compare
        # (should be numerical) is not numerical for some reason
        data[25, i] <-
          as.numeric(gsub("[^0-9]", "", data[25, i])) # replaces character values
        # other than numeric from 0-9
      }
      if (class(data[26, i]) == "character") {
```

```
    data[26, i] <- as.numeric(gsub("[^0-9]", "", data[26, i]))
  }
}

temperatures = c()
max_val = 0
min_val = 0
max_index = 0
min_index = 0

for (i in 2:(ncol(data))) { # check every column besides for the first
  for (j in 1:(nrow(data))) { # checks every temp value per hour based on the ith column
    if (j == 25) { # the MAX Hour (variable we want to compare w/) lies on the 25th row
      max_index = data[j, i]
    } else if (j == 26) {
      min_index = data[j, i]
    } else {
      temperatures = c(temperatures, data[j, i]) # append all the temp values we
      # want to compare w/ in a vector
    }
  }
  max_val = max(temperatures) # retrieve max temp for the ith column
  min_val = min(temperatures) # retrieve min temp for the ith column

  if (data[max_index,i] == max_val) { #
    correct_max = correct_max + 1
  } else{
    incorrect_max = incorrect_max + 1
  }

  if (data[min_index,i] == min_val) {
    correct_min = correct_min + 1
  } else{
    incorrect_min = incorrect_min + 1
  }
  temperatures = c()
}

data = data[-c(25, 26),] # remove MAX and MIN rows

Hours = rep(data[, 1], 12) # repeat the hour column 12 times for each month = 288
# total rows
Months = c()
Temperature = c() # temperatures values based on title names (Dry, Dew, etc.)

for (i in 2:ncol(data)) { # extract temp values
  Temperature = c(Temperature, data[, i])
}

for (i in 2:ncol(data)) { # extract months
  Months = c(Months, rep(colnames(data)[i], 24))
}
```

```
    data = cbind(Hours, Months, Temperature)
    data_list[[title]] <- data.frame(data) # put each df into a list
  }

  print(
    paste0(
      "Out of the 60 maximum hours that were verified for the five hourly tables ",
      correct_max,
      " were correct."
    )
  )
  print(
    paste0(
      "Out of the 60 minimum hours that were verified for the five hourly tables ",
      correct_min,
      " were correct."
    )
  )

  return(data_list)
}
```

Davis Hourly File

```
davis_hourly = open_hourly(
  "~/Documents/UCD Classes/22-23/Spring Q23 Classes/STA 141B/HW1/USA_CA_UC_Davis/davis.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
```

```
## [1] "Out of the 60 maximum hours that were verified for the five hourly tables 60 were correct."
## [1] "Out of the 60 minimum hours that were verified for the five hourly tables 60 were correct."
```

```
lapply(davis_hourly, head)
```

```
## $`Average Hourly Statistics for Dry Bulb temperatures`
##       Hours Months Temperature
## 1 0:01-1:00    Jan         8.1
## 2 1:01-2:00    Jan         7.7
## 3 2:01-3:00    Jan         7.6
## 4 3:01-4:00    Jan         7.3
## 5 4:01-5:00    Jan         7.3
## 6 5:01-6:00    Jan         7.2
##
## $`Average Hourly Statistics for Dew Point temperatures`
##       Hours Months Temperature
## 1 0:01-1:00    Jan         6.3
## 2 1:01-2:00    Jan           6
## 3 2:01-3:00    Jan         5.9
## 4 3:01-4:00    Jan         5.7
## 5 4:01-5:00    Jan         5.6
## 6 5:01-6:00    Jan         5.6
##
## $`Average Hourly Relative Humidity`
##       Hours Months Temperature
## 1 0:01-1:00    Jan          89
## 2 1:01-2:00    Jan          90
## 3 2:01-3:00    Jan          89
## 4 3:01-4:00    Jan          91
## 5 4:01-5:00    Jan          90
## 6 5:01-6:00    Jan          91
##
## $`Average Hourly Statistics for Direct Normal Solar Radiation`
##       Hours Months Temperature
## 1 0:01-1:00    Jan           0
## 2 1:01-2:00    Jan           0
## 3 2:01-3:00    Jan           0
## 4 3:01-4:00    Jan           0
## 5 4:01-5:00    Jan           0
```

```
## 6 5:01-6:00    Jan           0
##
## $`Average Hourly Statistics for Wind Speed`
##       Hours Months Temperature
## 1 0:01-1:00    Jan           2
## 2 1:01-2:00    Jan         1.9
## 3 2:01-3:00    Jan           2
## 4 3:01-4:00    Jan         1.9
## 5 4:01-5:00    Jan         2.2
## 6 5:01-6:00    Jan         2.5
```

Verifying the Merged Hourly Data

I used the my open_hourly_individual() function within the function for merging to first extract the hour and month columns. I put these two columns into a new dataframe. I did this because all of the hourly tables have three columns, and two of these columns (hours, months) are identical across all tables. The only columns that are different are the 3rd column values, which are the values based on the titles that need to be passed to the function (i.e temp values, wind speed, solar radiation). To append the other column values, I looped through every table based on the title and extracted the third (last) column that stored the unique values, and column binded it to the new data frame. To check if the order of the values in the new data frame were correct, I checked all of the five hourly data tables to the corresponding month and hour using head() and tail().

```
# function for merging hourly data
merge_hourly_data <-
  function(filepath, titles) {
    # title is a vector that contains names of variables
    df = data.frame()

    hours_month = open_hourly_individual(filepath, titles[1]) # opens the hourly
    # data only for the first df
    hours_month = hours_month[,-3] # removes the last column (temp_vals) and keeps
    # the hour and month columns

    df = rbind(df, hours_month) # create a new df w/ only hour and months

    for (i in 1:length(titles)) {
      df = cbind(df, do.call(cbind, open_hourly_individual(filepath, titles[i])[3]))
      # column bind the 3rd column (temp_vals) of every df based on titles
    }

    colnames(df) = c(
      "Hours",
      "Months",
      "Dry_Bulb_Temp",
      "Dew_Point_Temp",
      "Relative_Humidity",
      "Direct_Normal_Solar_Radiation",
      "Wind_Speed"
    )

    hour_order <-
      c(
        "0:01-1:00",
        "1:01-2:00",
        "2:01-3:00",
        "3:01-4:00",
        "4:01-5:00",
        "5:01-6:00",
        "6:01-7:00",
        "7:01-8:00",
        "8:01-9:00",
        "9:01-10:00",
        "10:01-11:00",
        "11:01-12:00",
```

```
      "12:01-13:00",
      "13:01-14:00",
      "14:01-15:00",
      "15:01-16:00",
      "16:01-17:00",
      "17:01-18:00",
      "18:01-19:00",
      "19:01-20:00",
      "20:01-21:00",
      "21:01-22:00",
      "22:01-23:00",
      "23:01-24:00"
    )

  df$Hours <-
    factor(df$Hours, levels = hour_order) # set as factor, otherwise R doesn't
  # read in by order but by char

  month_order <-
    c("Jan",
      "Feb",
      "Mar",
      "Apr",
      "May",
      "Jun",
      "Jul",
      "Aug",
      "Sep",
      "Oct",
      "Nov",
      "Dec")

  df$Months <- factor(df$Months, levels = month_order)

  for (i in 3:ncol(df)) {
    df[, i] = as.numeric(df[, i])
  }

  df = as.data.frame(df)
  return(df)
}
```

Davis Merged Hourly Data

```
davis_hourly_data = merge_hourly_data(
  "USA_CA_UC_Davis/davis.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
```

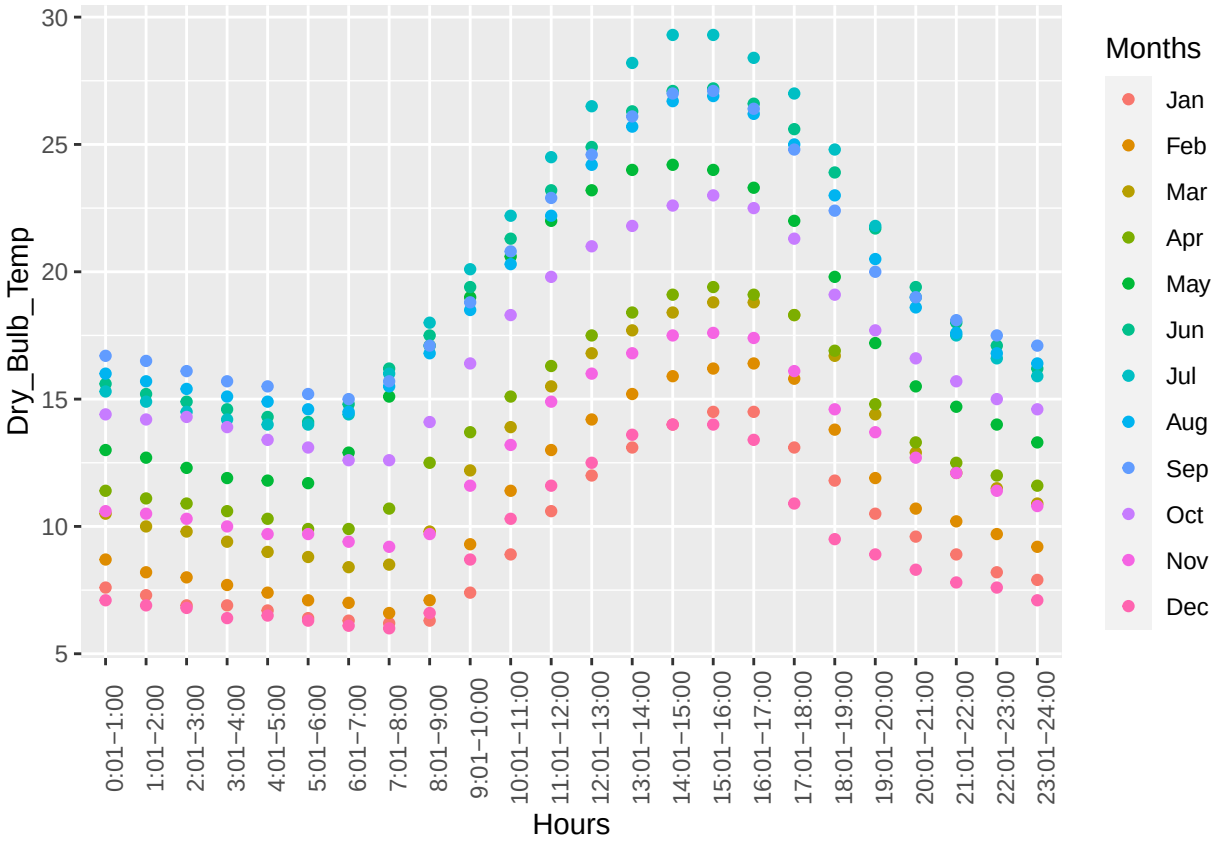
```
davis_hourly_data = head(davis_hourly_data)
davis_hourly_data
```

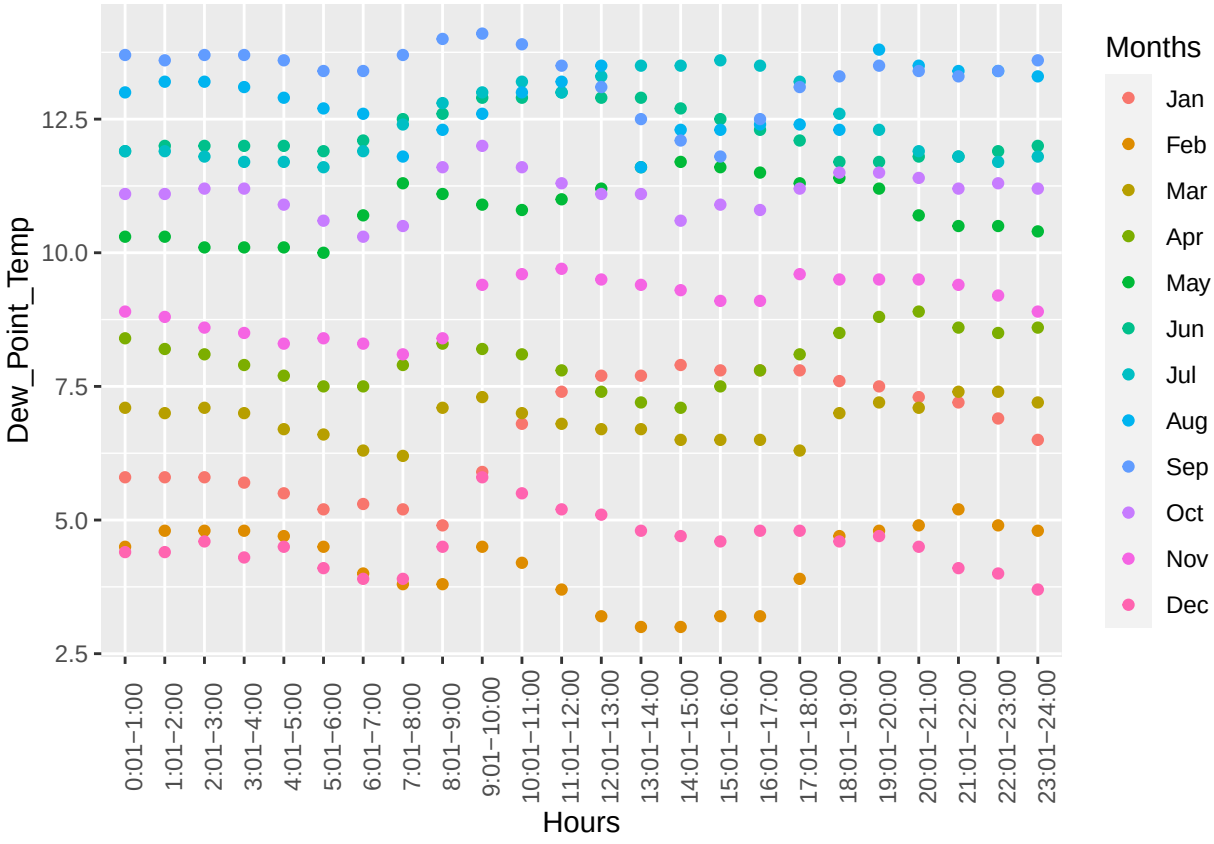
```
##       Hours Months Dry_Bulb_Temp Dew_Point_Temp Relative_Humidity
## 1 0:01-1:00    Jan           8.1            6.3                89
## 2 1:01-2:00    Jan           7.7            6.0                90
## 3 2:01-3:00    Jan           7.6            5.9                89
## 4 3:01-4:00    Jan           7.3            5.7                91
## 5 4:01-5:00    Jan           7.3            5.6                90
## 6 5:01-6:00    Jan           7.2            5.6                91
##   Direct_Normal_Solar_Radiation Wind_Speed
## 1                             0        2.0
## 2                             0        1.9
## 3                             0        2.0
## 4                             0        1.9
## 5                             0        2.2
## 6                             0        2.5
```

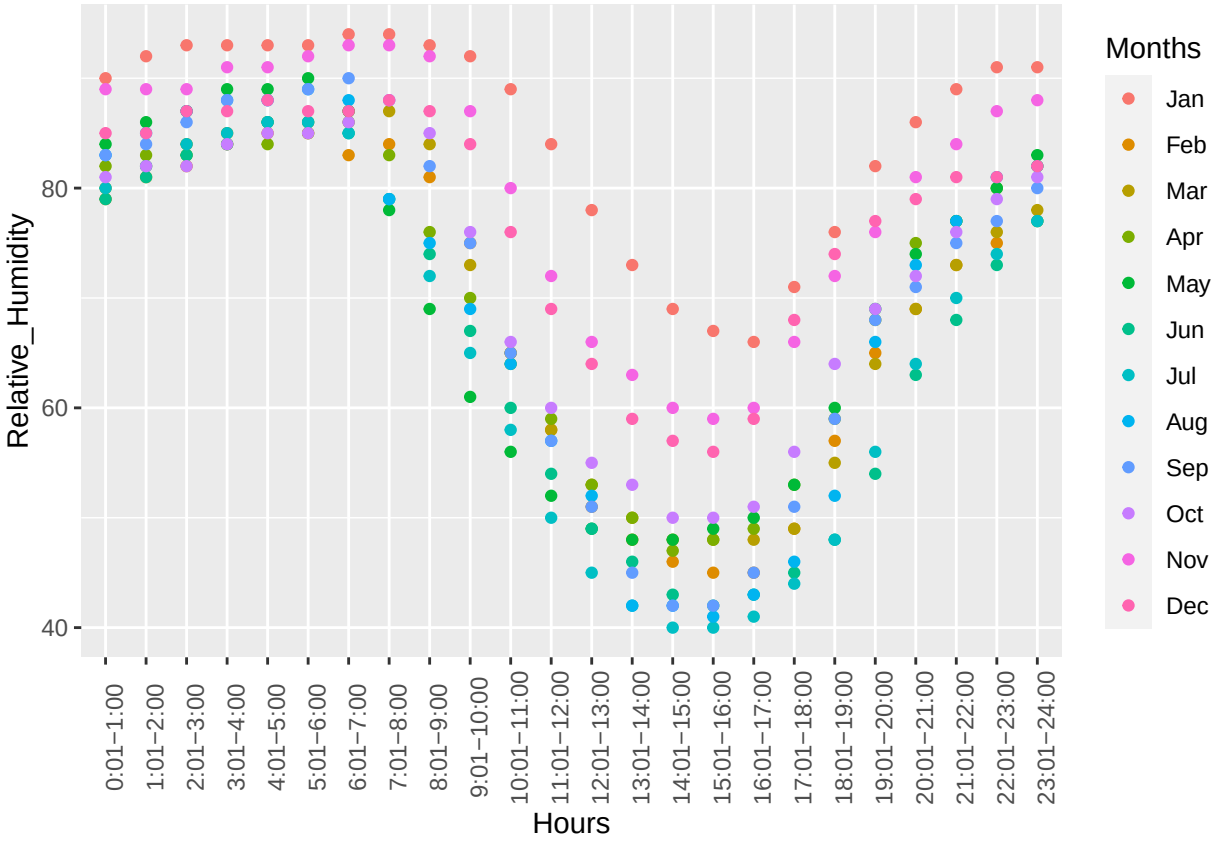
PLOTS OF HOURLY DATA (TEMPERATURES VS HOURS for EACH MONTH)

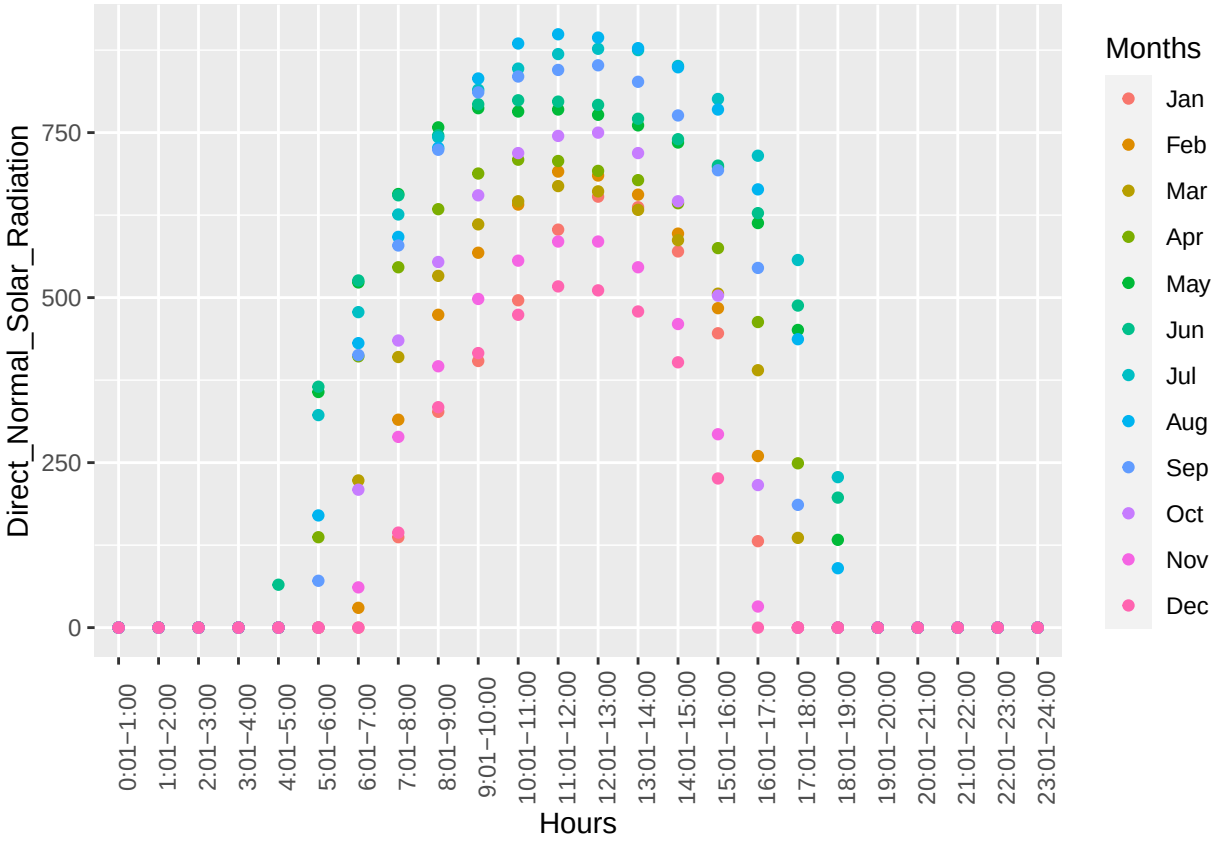
```r
# plotting function for hourly data vs temperatures for each location
library(ggplot2)
library(scales)

plot_hourly <- function(filename, titles) {
  hourly_data = merge_hourly_data(filename, titles)

  for (i in 1:length(titles)) {
    if (titles[i] == "Average Hourly Statistics for Dry Bulb temperatures") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Dry_Bulb_Temp, color = Months
          )) + theme(axis.text.x = element_text(angle = 90)) # flips the x-axis
        # names vertically for easier user read
      )
    } else if (titles[i] == "Average Hourly Statistics for Dew Point temperatures") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Dew_Point_Temp, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
      )
    } else if (titles[i] == "Average Hourly Relative Humidity") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Relative_Humidity, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
      )
    } else if (titles[i] == "Average Hourly Statistics for Direct Normal Solar Radiation") {
      print(
        ggplot() +
          geom_point(
            data = hourly_data,
            aes(x = Hours, y = Direct_Normal_Solar_Radiation, color = Months)
          ) + theme(axis.text.x = element_text(angle = 90))
      )
    } else {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Wind_Speed, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
      )
    }
  }
}
```

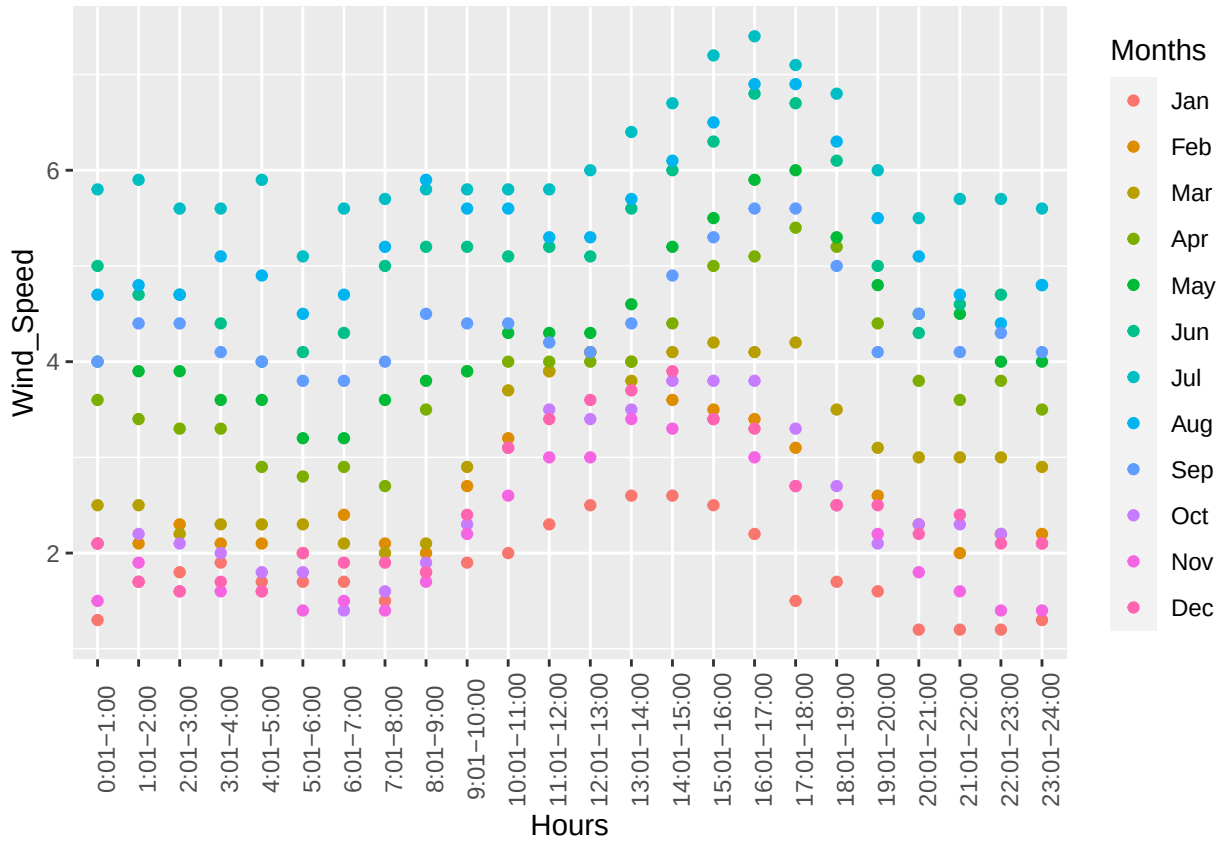
Fairfield Plots





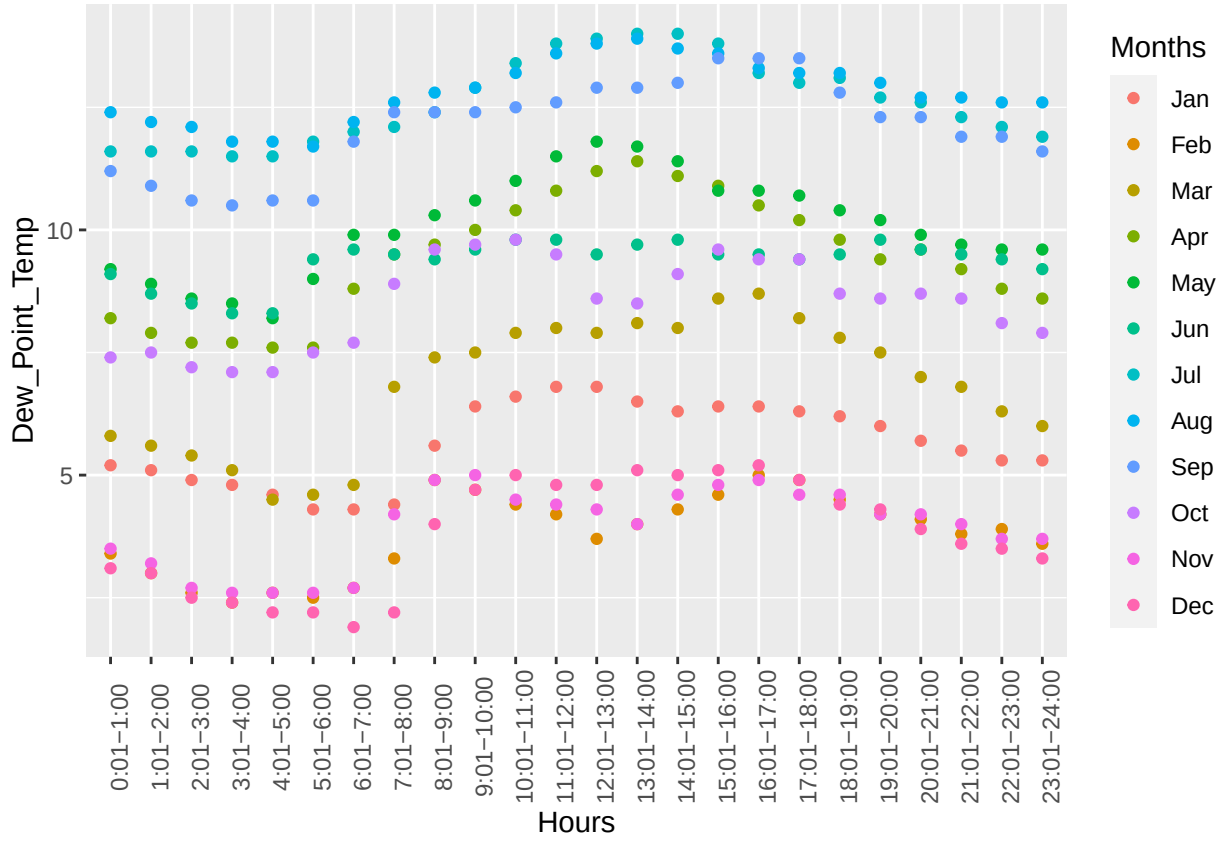

Relative_Humidity
80
60
40
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

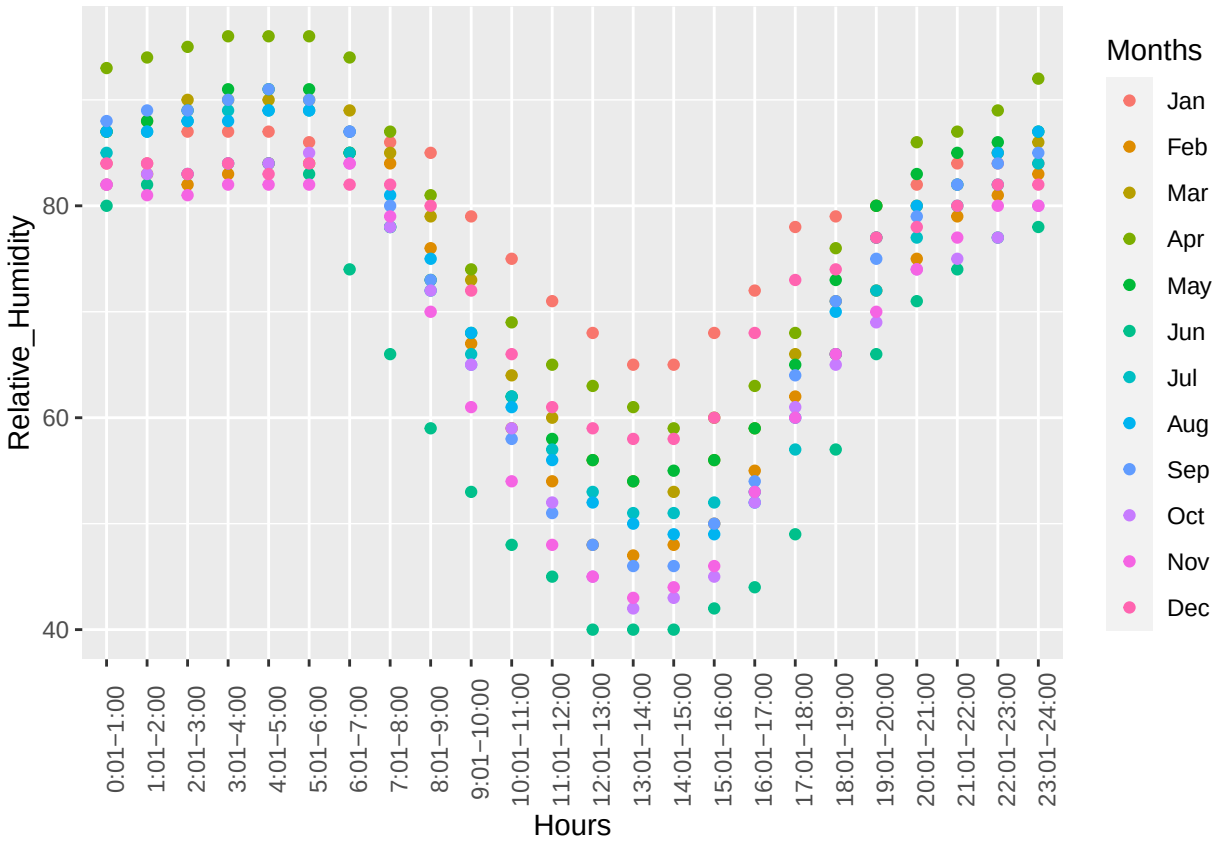

Direct_Normal_Solar_Radiation
750
500
250
0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

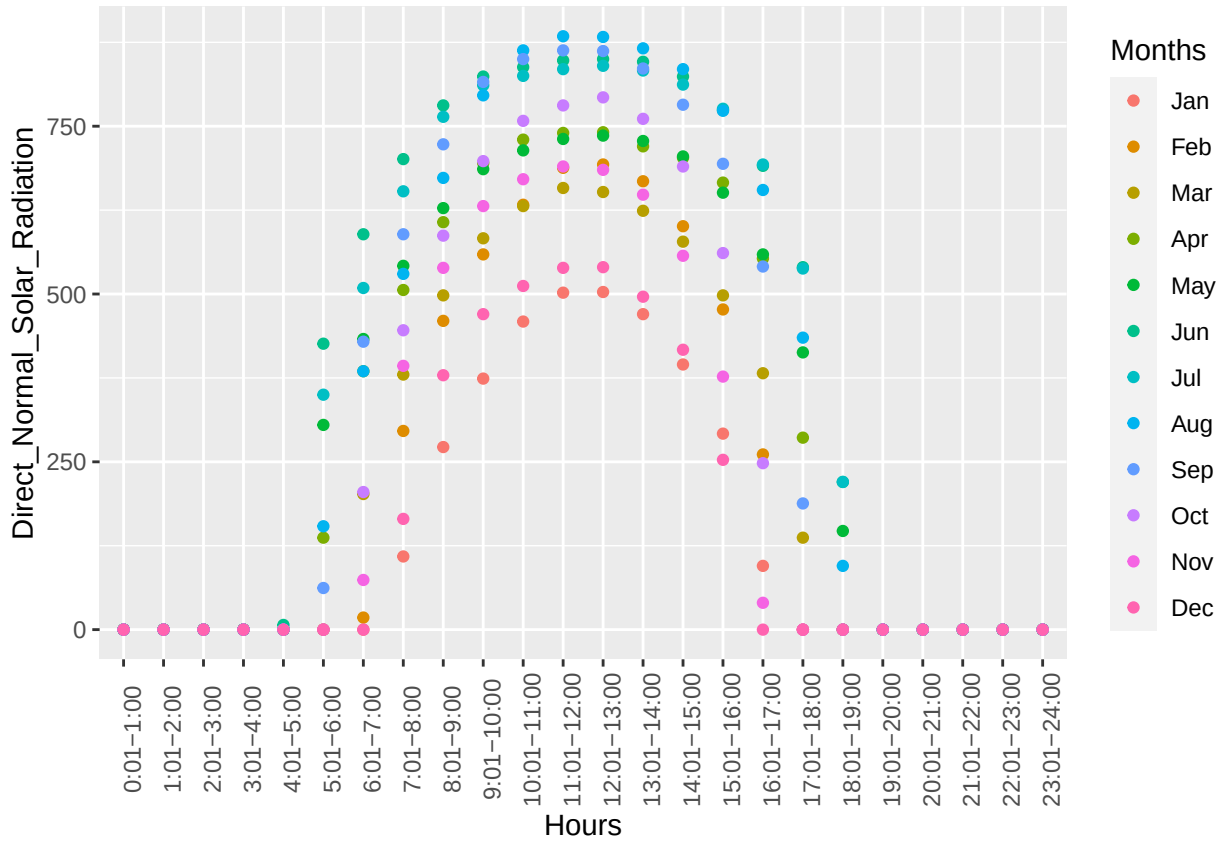

Wind_Speed
6
4
2
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

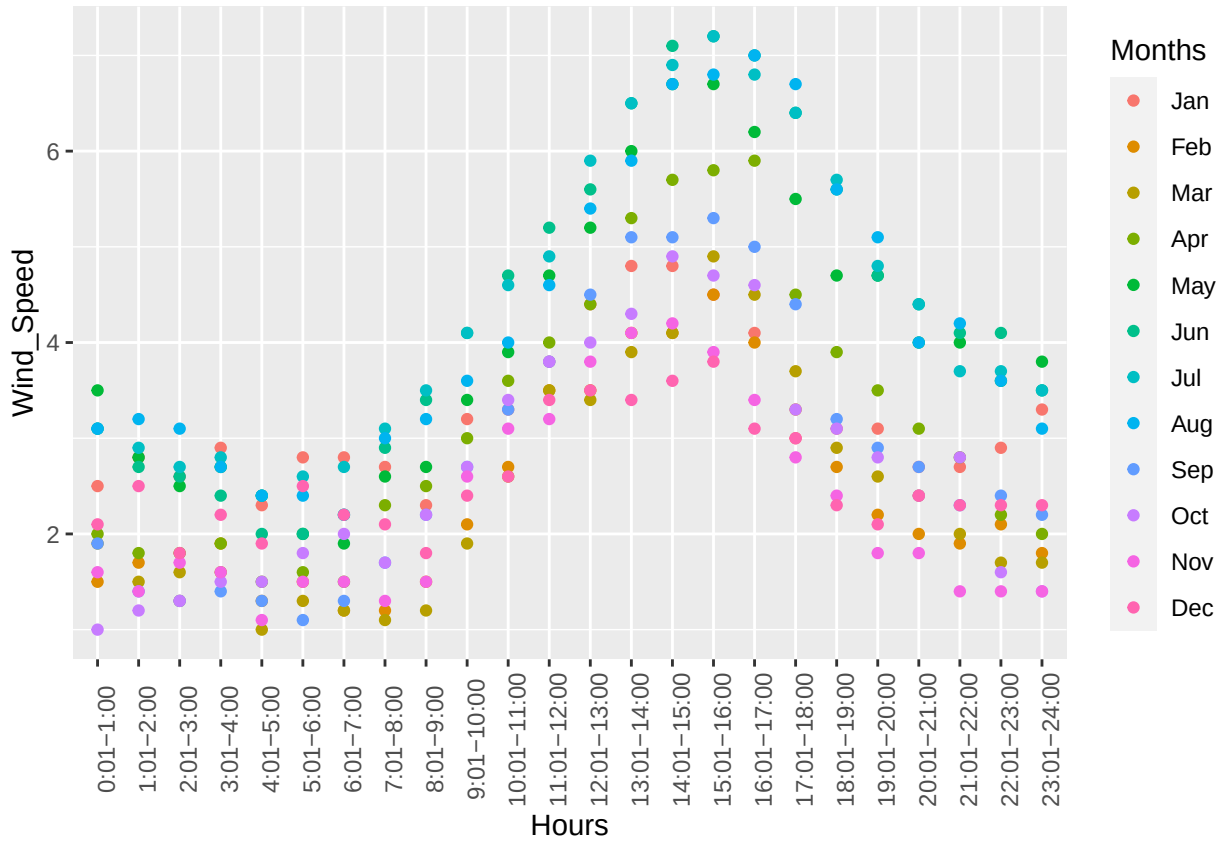
Napa Plots


Dry_Bulb_Temp
20
10
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

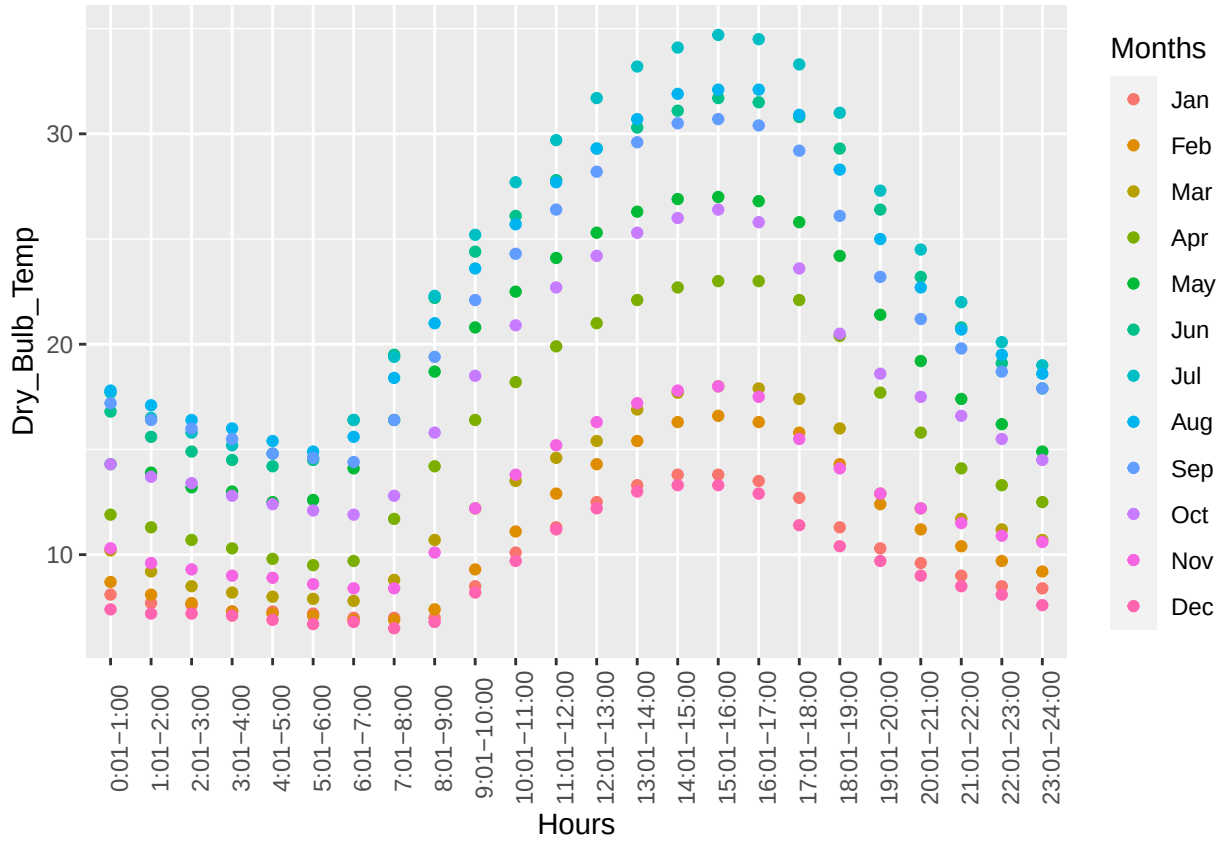

Dew_Point_Temp
10
5
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

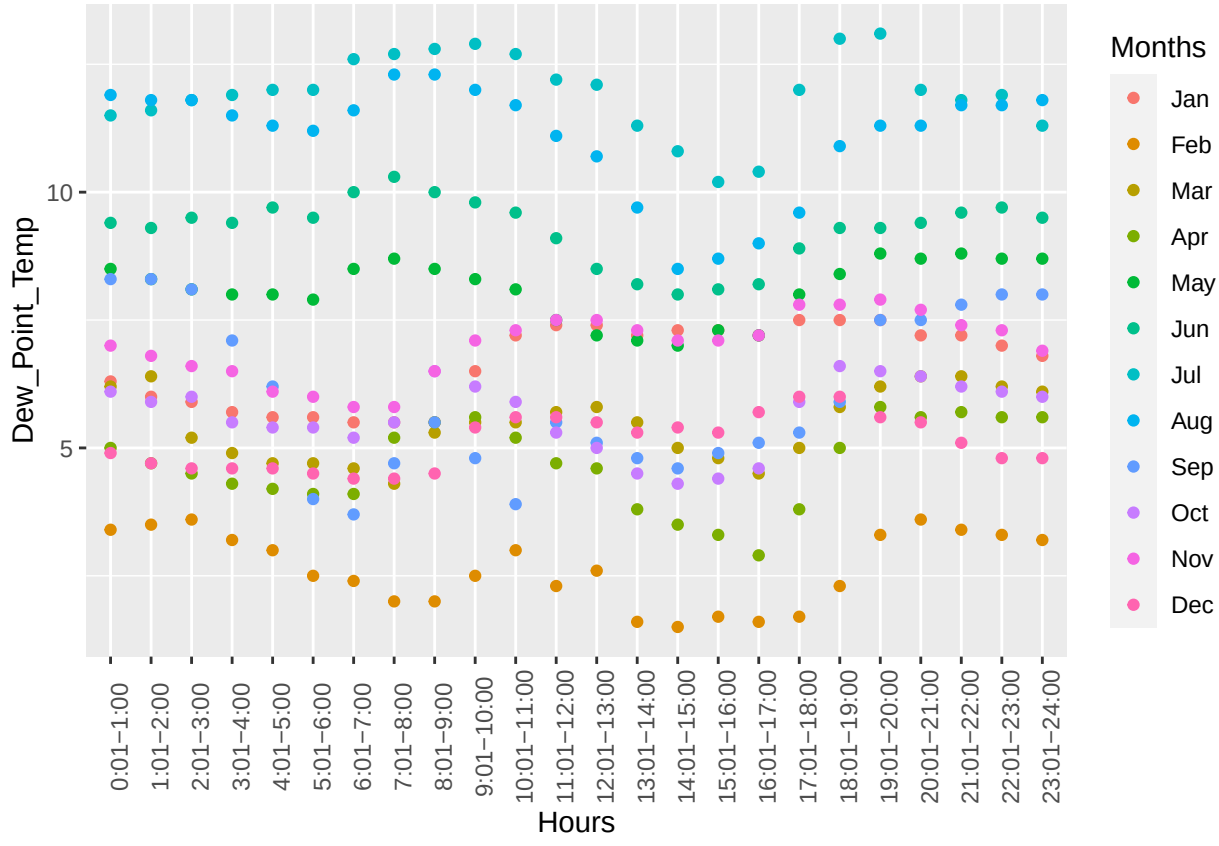

Relative_Humidity
80
60
40
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

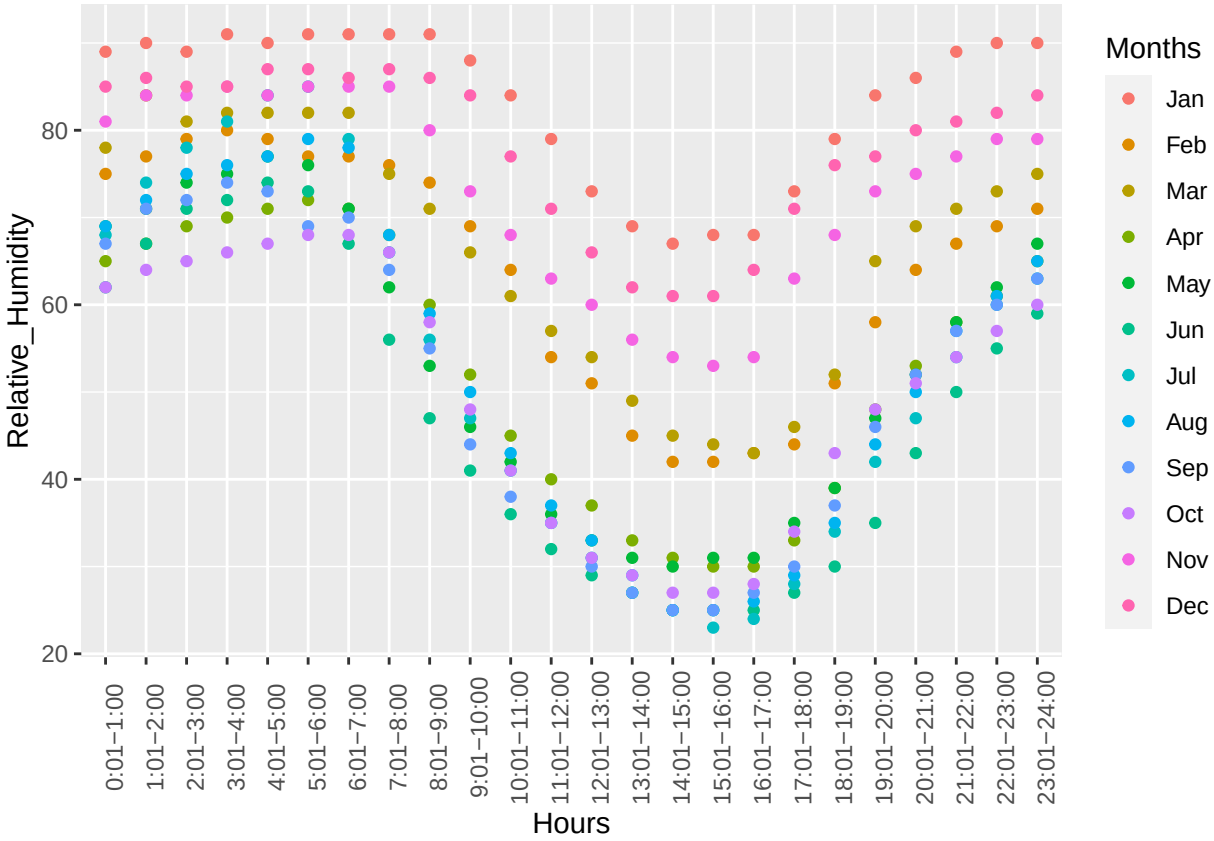

Direct_Normal_Solar_Radiation
750
500
250
0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

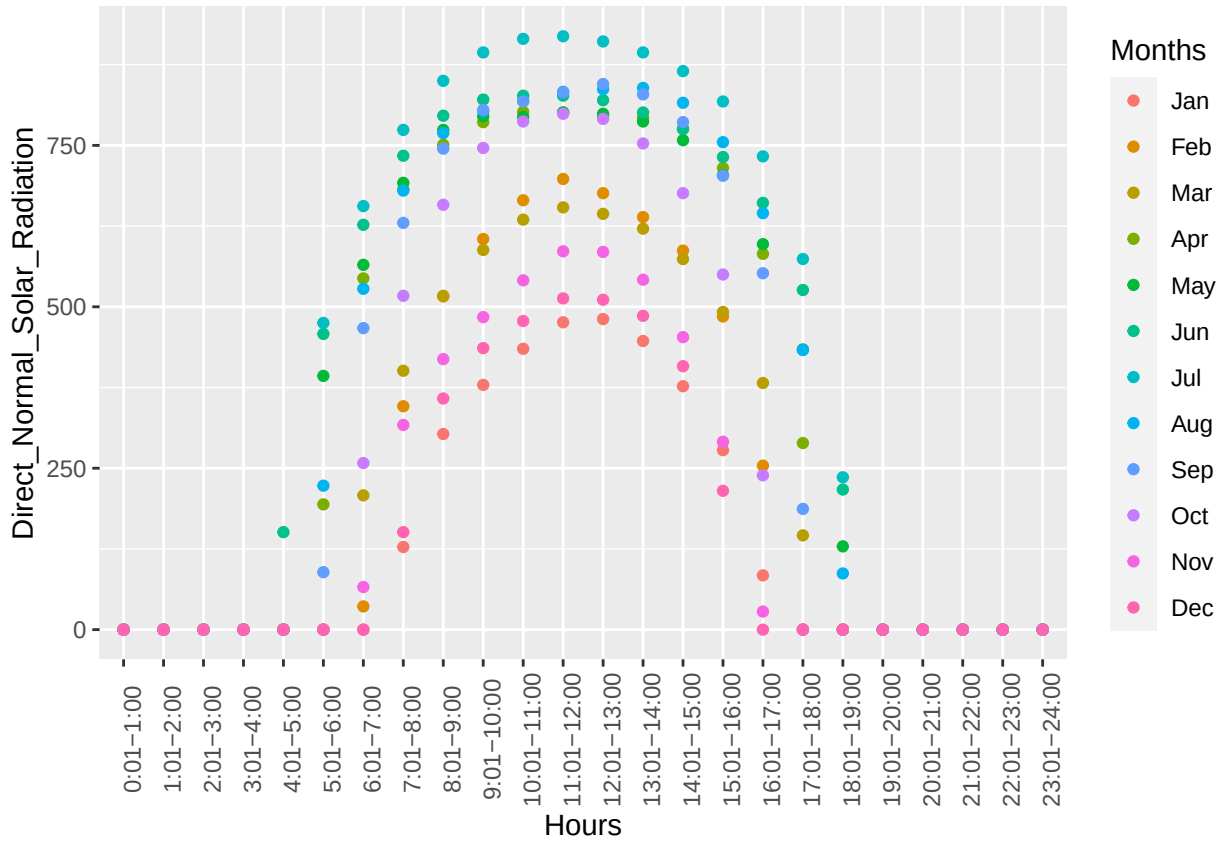

Wind_Speed
6
4
2
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

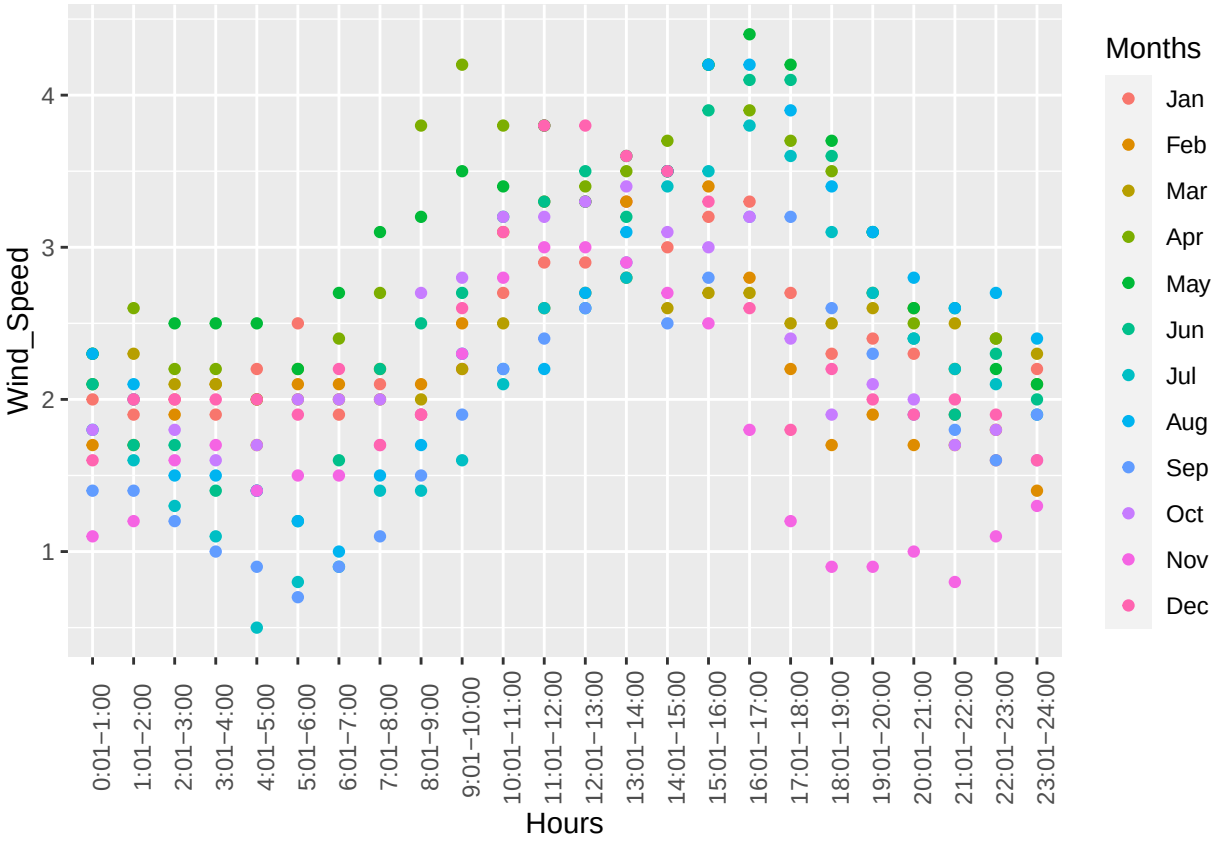
Davis Plots


Dry_Bulb_Temp
30
20
10
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

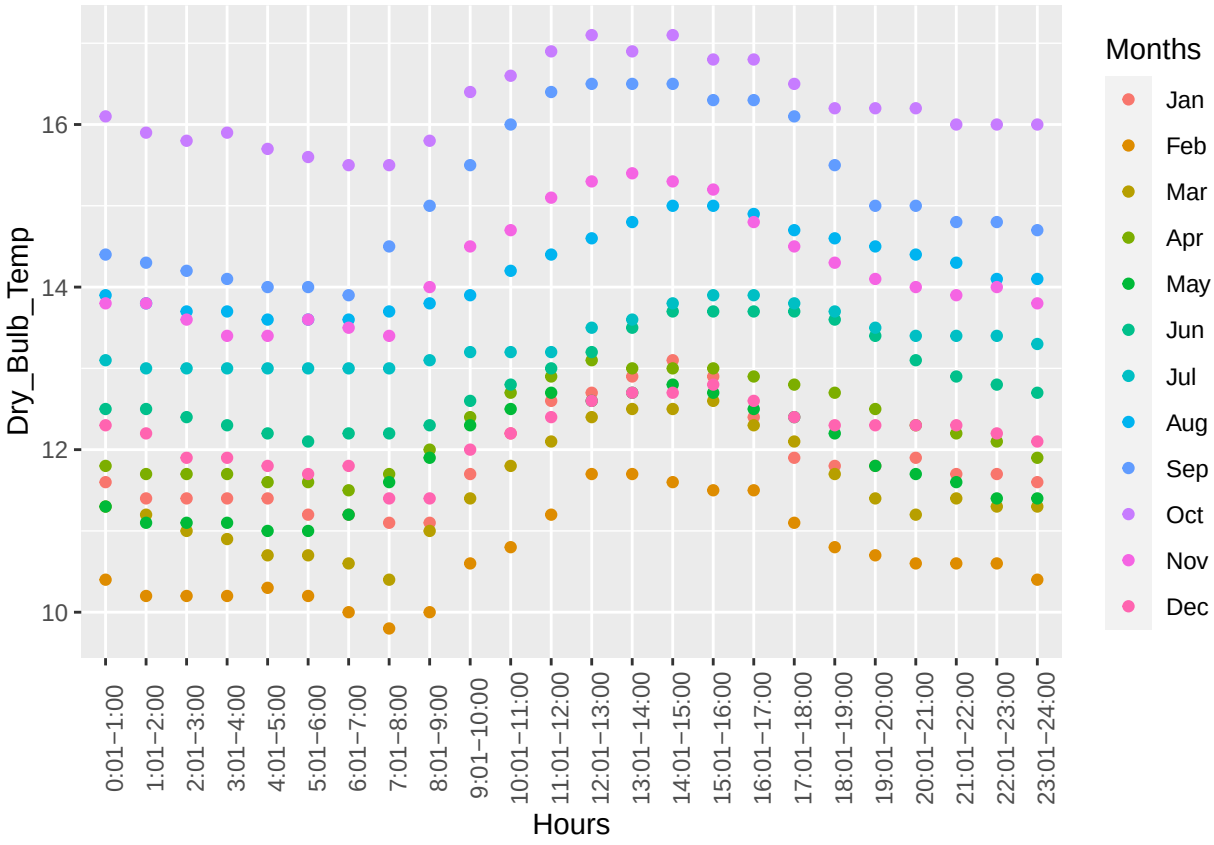

Dew_Point_Temp
10
5
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

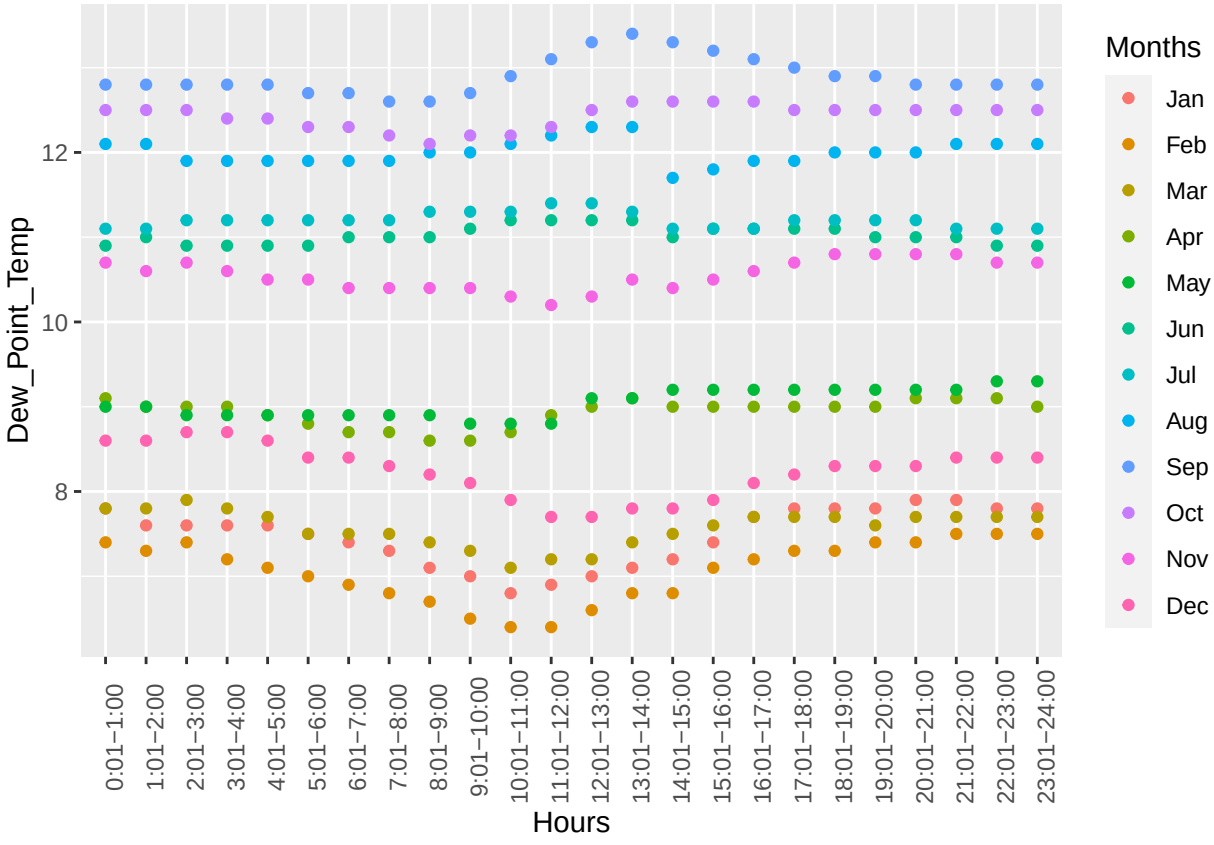

Relative_Humidity
80
60
40
20
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

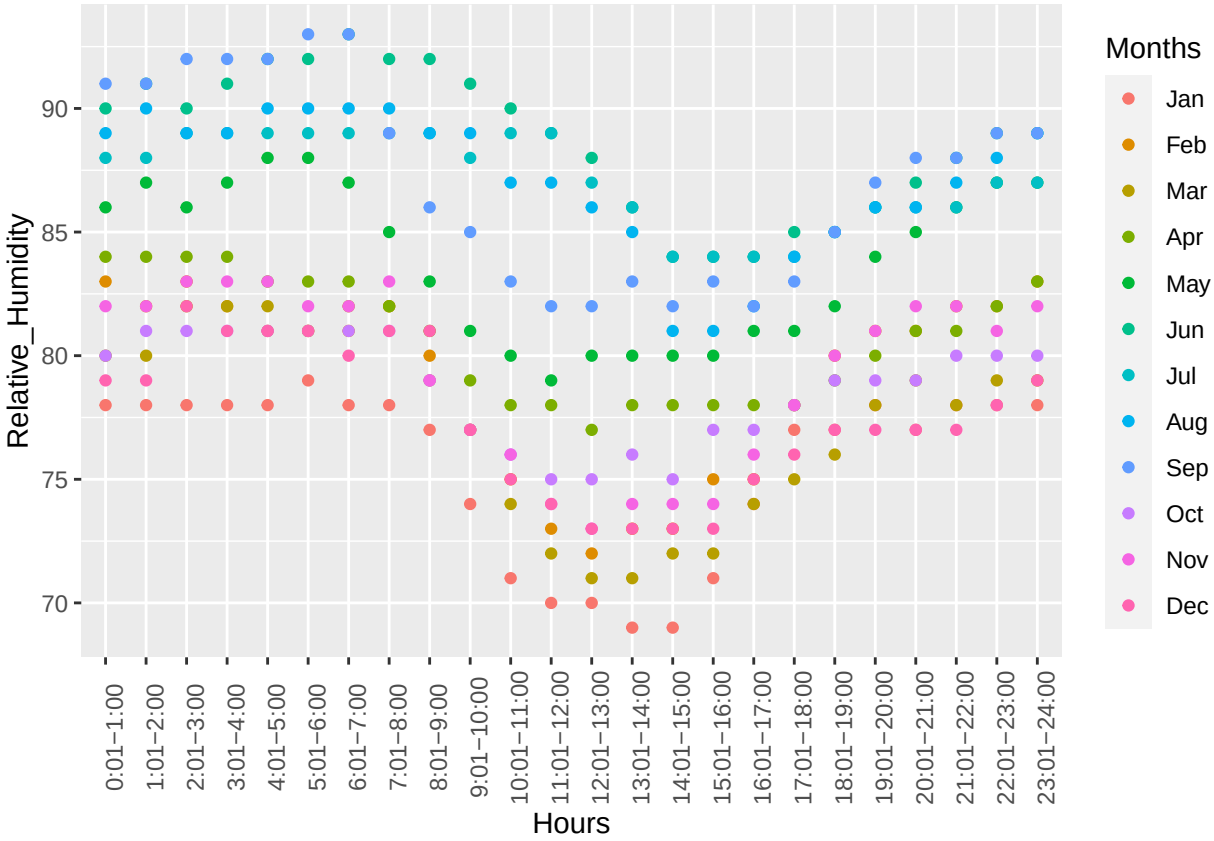

Direct_Normal_Solar_Radiation
750
500
250
0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

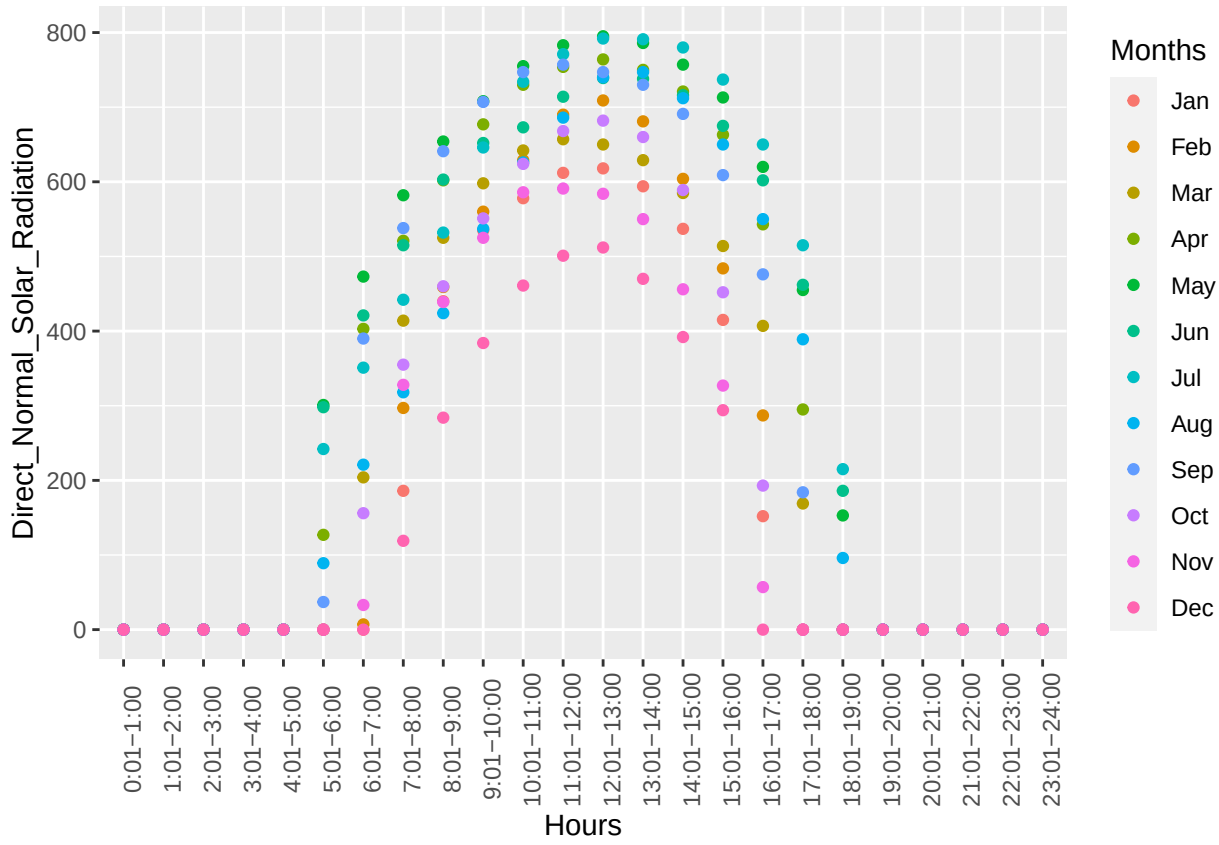

Wind_Speed
4
3
2
1
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

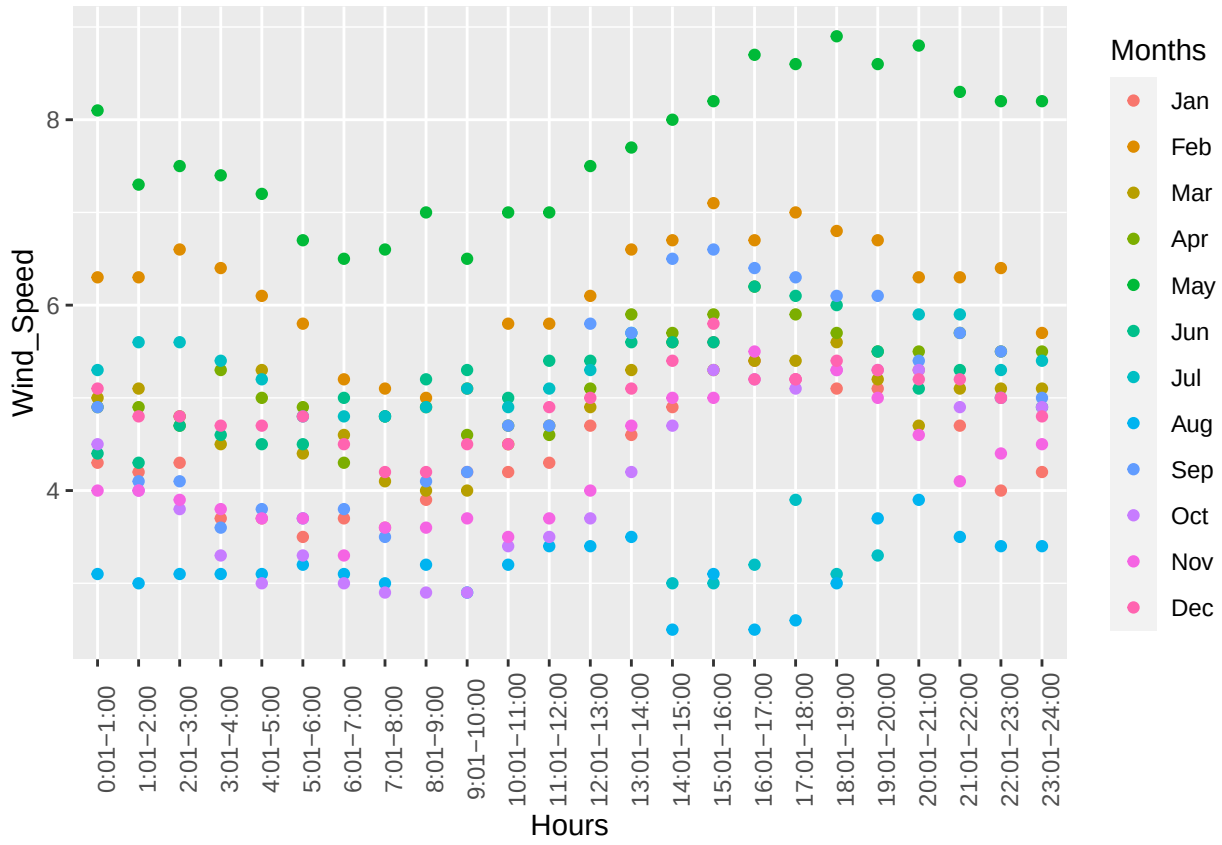
Point Reyes Plots



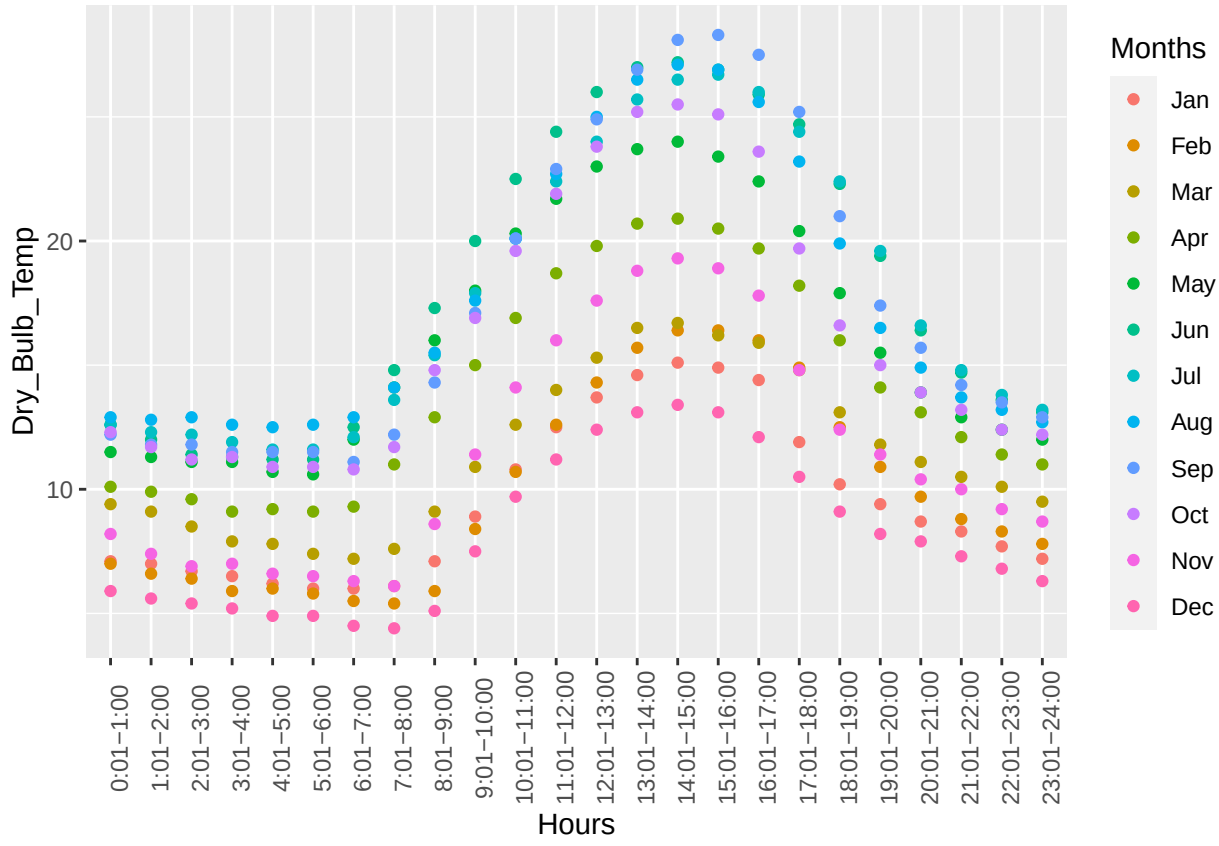


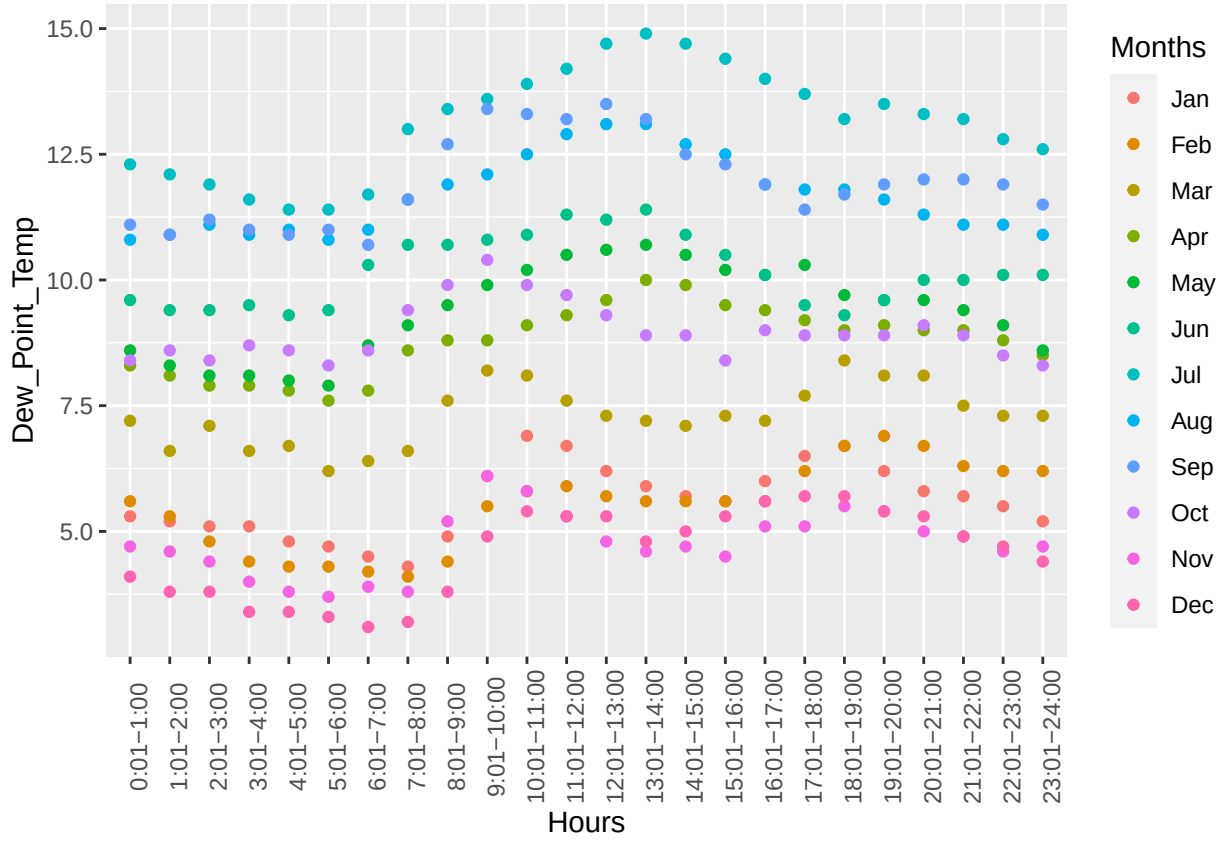

Relative_Humidity
90
85
80
75
70
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

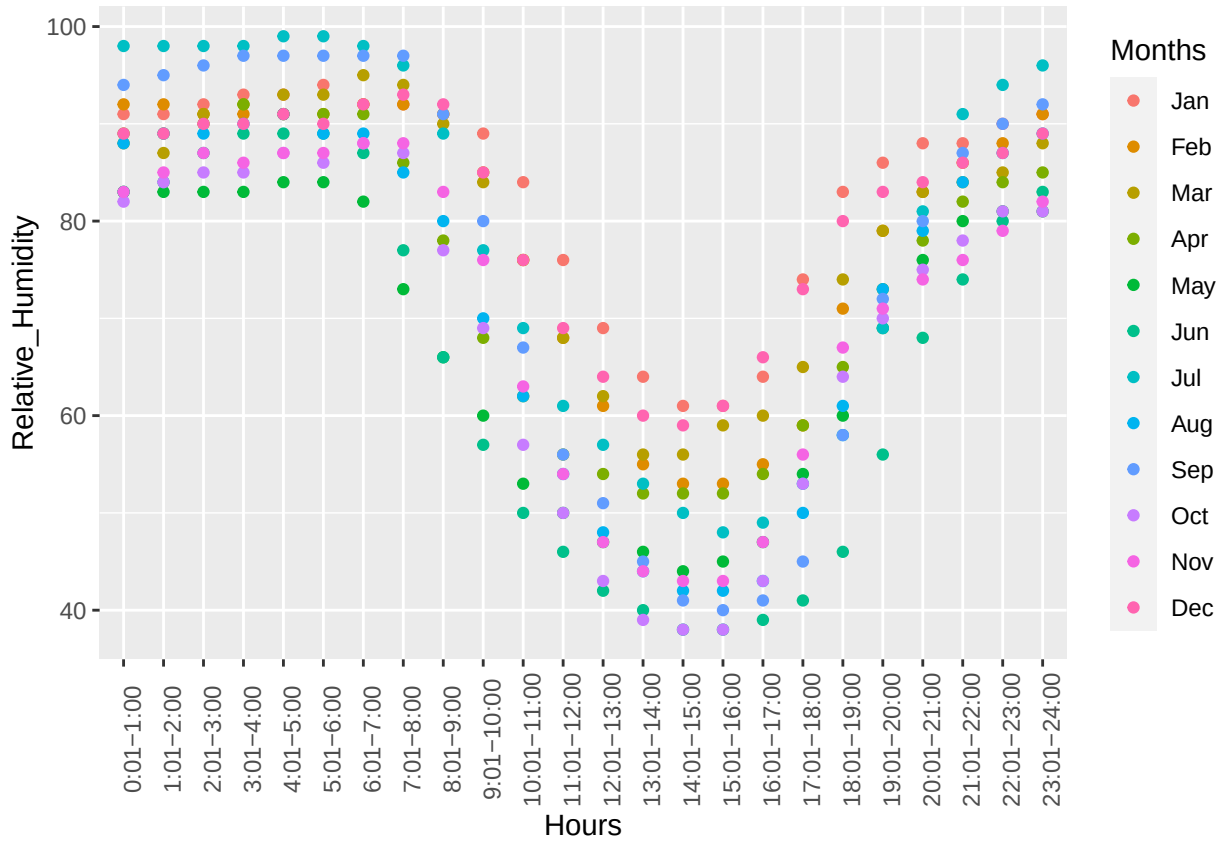

Direct_Normal_Solar_Radiation
800
600
400
200
0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

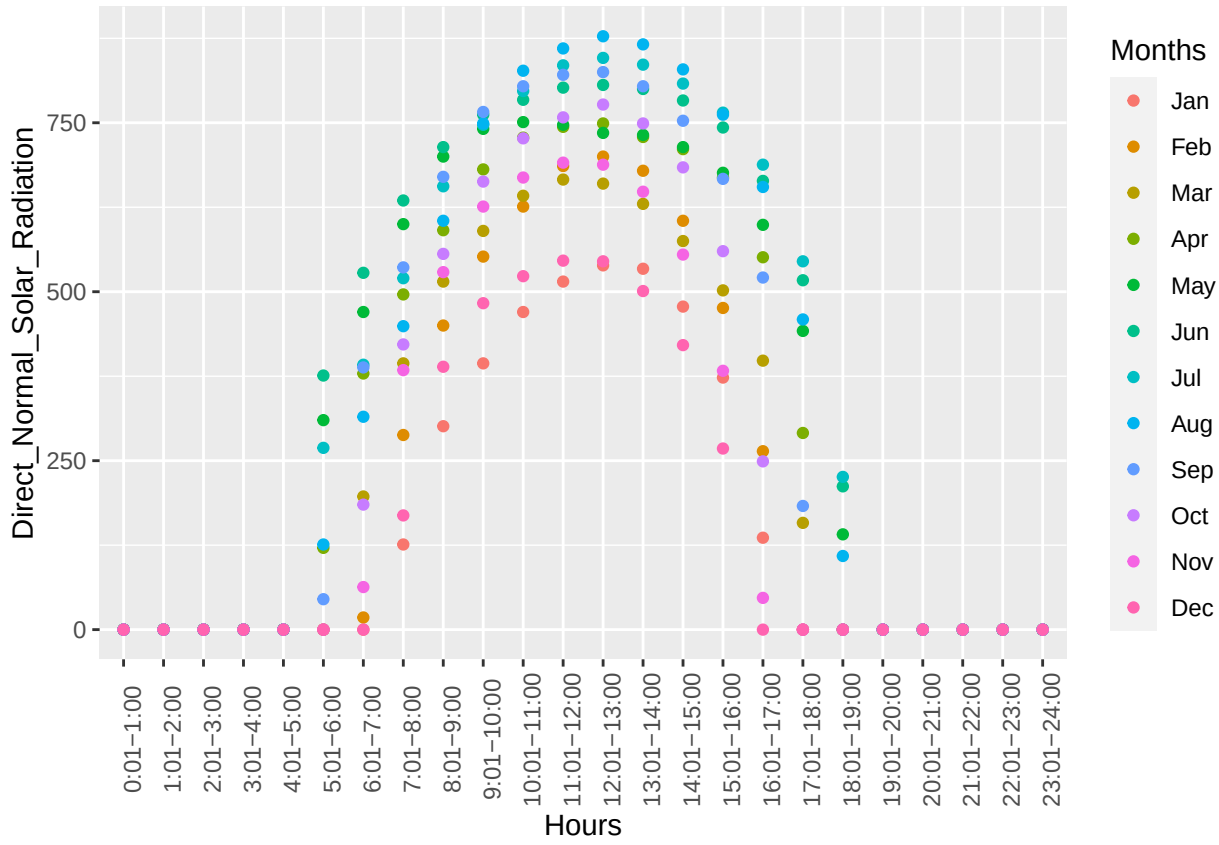

Wind_Speed
8
6
4
Hours
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec

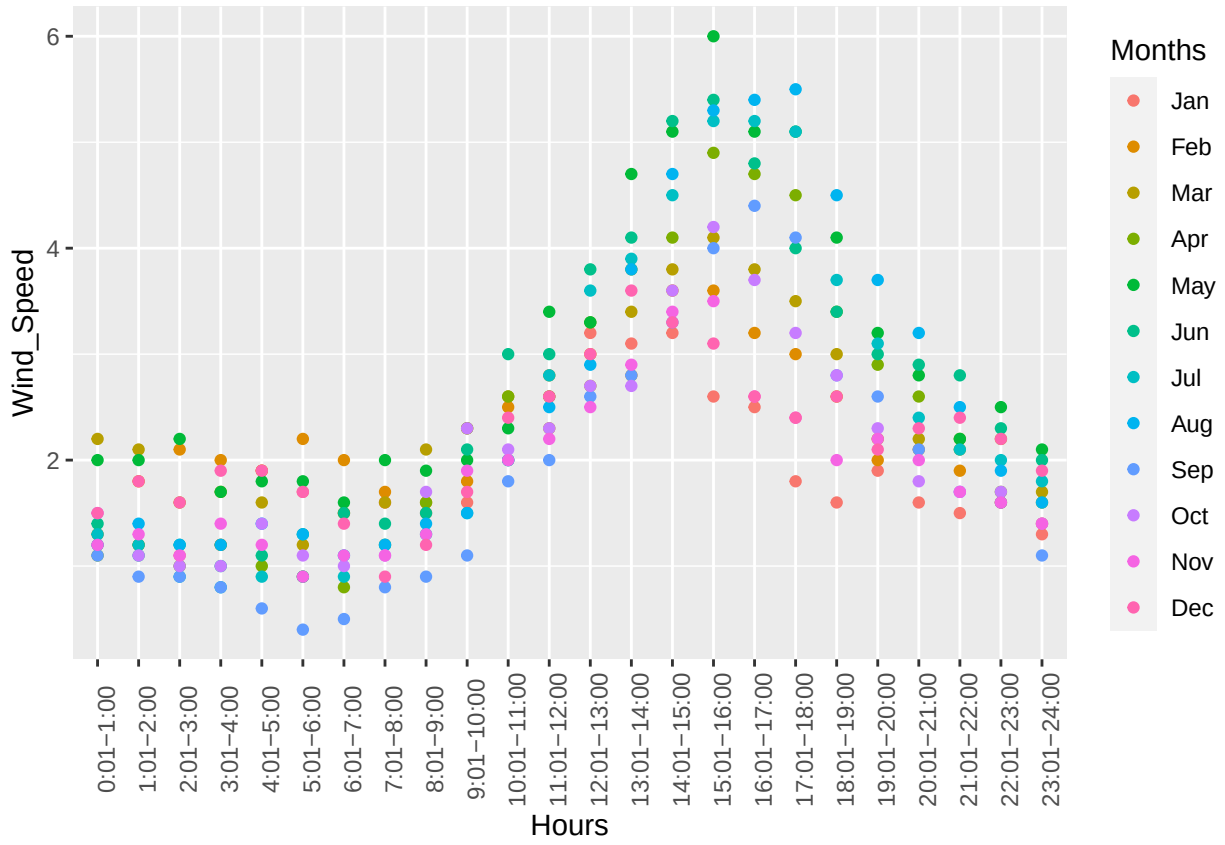
Marin Plots


Dry_Bulb_Temp
20
10
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Dew_Point_Temp
15.0
12.5
10.0
7.5
5.0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Relative_Humidity
100
80
60
40
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Direct_Normal_Solar_Radiation
750
500
250
0
0:01–1:00
1:01–2:00
2:01–3:00
3:01–4:00
4:01–5:00
5:01–6:00
6:01–7:00
7:01–8:00
8:01–9:00
9:01–10:00
10:01–11:00
11:01–12:00
12:01–13:00
13:01–14:00
14:01–15:00
15:01–16:00
16:01–17:00
17:01–18:00
18:01–19:00
19:01–20:00
20:01–21:00
21:01–22:00
22:01–23:00
23:01–24:00
Hours
Months
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec



Analysis of Plots

After verifying that the data is correct, I plotted the results for every location. Even without verifying the data (however I did because of practice), the visualizations show that the data does make sense. Because we are plotting Temperature vs Hours, it would be practical that the temperatures would be higher during the day/afternoon compared to night, and the data does follow that trend. In addition, we see that the months that correspond to summer (May - August) generally have higher placements (temperature values) compared to other months.

Code Appendix

```
knitr::opts_chunk$set(echo = TRUE)
options(width = 80)
# set cwd
setwd("~/Documents/UCD Classes/22-23/Spring Q23 Classes/STA 141B/HW1")

# Useful? - package -> codetools::findGlobals
# Notes from Discussion:
# Use is.NA to check for empty columns
# WEA function
open_wea <- function(filepath) {
  data = read.table(filepath, skip = 6)
  colnames(data) = c(
    "Month",
    "Day",
    "Standard Time",
    "Direct Normal Irradiance",
    "Diffuse Horizontal Radiance"
  )
  data = data[ , colSums(is.na(data))==0] # checking for columns w/ NAs
  return(data)

}
wea_tables = lapply(
  c(
    "USA_CA_Fairfield/fairfield.wea",
    "USA_CA_Napa/napa.wea",
    "USA_CA_UC_Davis/davis.wea",
    "USA_CA_Point/point_reyes.wea",
    "USA_CA_Marin/marin.wea"
  ),
  open_wea
)
lapply(wea_tables, head)
print("The order of numbers in [[]] correspond to: Fairfield, Napa, Davis, Point, and Marin.")
# PVSYST function
open_pvsyst <- function(filepath) {
  data = read.table(filepath, sep = ",", skip = 14)
  colnames(data) = c(
    'Year',
    'Month',
    'Day',
    'Hour',
    'Minute',
    'GHI',
    'DHI',
    'DNI',
    'Tamb',
    'WindVel',
    'WindDir'
  )
  data = data[ , colSums(is.na(data))==0]
```

```
  return(data)
}
pvsyst_tables = lapply(
  c(
    "USA_CA_Fairfield/fairfield.pvsyst",
    "USA_CA_Napa/napa.pvsyst",
    "USA_CA_UC_Davis/davis.pvsyst",
    "USA_CA_Point/point_reyes.pvsyst",
    "USA_CA_Marin/marin.pvsyst"
  ),
  open_pvsyst
)
lapply(pvsyst_tables, head)
print("The order of numbers in [[]] correspond to: Fairfield, Napa, Davis, Point, and Marin.")
open_monthly_individual <- function(filepath, title) {
  unique_id = grep(title, readLines(filepath)) # grabs line number (position) of title in text file
  if (title == "Monthly Statistics for Dry Bulb temperatures") {
    data = read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 12
    )
  } else if (title == "Monthly Statistics for Dew Point temperatures" |
             title[i] == "Monthly Statistics for Wind Speed") {
    data = read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 5
    )
  } else {
    data = read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 16
    )
  }

  data = data[ , colSums(is.na(data))==0]

  data = t(data) # transpose

  colnames(data) = data[1, ] # after transposing the matrix, previous row names that are now column nam

  data = data[-1, ] # get rid of the column names contained within our data frame

  data = as.data.frame(data)
```

```
  for (i in 1:ncol(data)) {
    colnames(data)[i] = trimws(colnames(data)[i]) # trim whitespaces for column names in df
  }


  while (title != "Monthly Wind Direction") {
    for (i in 1:nrow(data)) {
      day_hour <- strsplit(as.character(data[i, 2]), ":")[[1]] # CHATGPT Assistance
      day <- as.numeric(day_hour[1])
      hour <- as.numeric(day_hour[2])

      data[i, "Datetime_Max"] <-
        as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"), format = "%Y-%m-%d %H:%M:%S")
    }

    for (i in 1:nrow(data)) {
      day_hour <- strsplit(as.character(data[i, 4]), ":")[[1]]
      day <- as.numeric(day_hour[1])
      hour <- as.numeric(day_hour[2])

      data[i, "Datetime_Min"] <-
        as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"), format = "%Y-%m-%d %H:%M:%S")
    }

    data[, 2] = data$Datetime_Max
    data[, 4] = data$Datetime_Min
    data = subset(data, select = -c(Datetime_Max, Datetime_Min))

    for (i in 1:ncol(data)) {
      if (colnames(data)[i] == "Day:Hour" |
          colnames(data)[i] == "Day:Hour.1") {
        next
      }
      data[, i] = as.numeric(data[, i])
    }
    break
  }

  return(data)
}
# modified function that reads in multiple titles
open_monthly <- function(filepath, titles) {
  data_list <- list()
  for (title in titles) {
    unique_id <- grep(title, readLines(filepath))
    if (title == "Monthly Statistics for Dry Bulb temperatures") {
      data <-
        read.table(
          filepath,
          sep = "\t",
          header = TRUE,
          skip = unique_id,
          nrows = 12
```

```
    )
} else if (title == "Monthly Statistics for Dew Point temperatures" |
           title == "Monthly Statistics for Wind Speed") {
  data <-
    read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 5
    )
} else {
  data <-
    read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 16
    )
}

data = data[, colSums(is.na(data)) == 0]

data = t(data) # transpose

colnames(data) = data[1,] # after transposing the matrix, previous row names
# that are now column names are read in as the first line of data, and we want
# these to be the actual column names of our data

data = data[-1,] # get rid of the column names contained within our data frame

data = as.data.frame(data)

for (i in 1:ncol(data)) {
  colnames(data)[i] <-
    trimws(colnames(data)[i]) # trim whitespaces in column names
}

while (title != "Monthly Wind Direction") {
  # not applied to monthly bc no time data
  for (i in 1:nrow(data)) {
    day_hour <-
      strsplit(as.character(data[i, 2]), ":")[[1]] # Chat GPT assisted in
    # developing this code. Essentially the time data is being split by day and hours

    day <- as.numeric(day_hour[1])
    hour <- as.numeric(day_hour[2])

    data[i, "Datetime_Max"] <-
      as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"),
                 format = "%Y-%m-%d %H:%M:%S")
  }
```

```
      for (i in 1:nrow(data)) {
        day_hour <- strsplit(as.character(data[i, 4]), ":")[[1]]
        day <- as.numeric(day_hour[1])
        hour <- as.numeric(day_hour[2])

        data[i, "Datetime_Min"] <-
          as.POSIXct(paste0("2023-", i, "-", day, " ", hour, ":00:00"),
                     format = "%Y-%m-%d %H:%M:%S")
      }

      data[, 2] <-
        data$Datetime_Max # replace old column that contained time data
      # with new column Datetime_Max created above
      data[, 4] <- data$Datetime_Min # replace with Datetime_Min
      data <-
        subset(data, select = -c(Datetime_Max, Datetime_Min)) # delete the
      # new columns bc we already inputted the data into the original column indexes

      break
    }

    for (i in 1:ncol(data)) {
      # change every column into numeric aside for the
      # time columns
      if (colnames(data)[i] == "Day:Hour" |
          colnames(data)[i] == "Day:Hour.1") {
        next
      }
      data[, i] <- as.numeric(data[, i])
    }

    data_list[[title]] <-
      data.frame(data) # returns a list of dataframes, based
    # on the number of titles
  }
  return(data_list)
}
davis_monthly = open_monthly(
  "USA_CA_UC_Davis/davis.stat",
  c(
    "Monthly Statistics for Dry Bulb temperatures",
    "Monthly Statistics for Dew Point temperatures",
    "Monthly Wind Direction",
    "Monthly Statistics for Wind Speed"
  )
)
lapply(davis_monthly, head)
open_hourly_individual <- function(filepath, title) {
  unique_id = grep(title, readLines(filepath))
  data = read.table(
    filepath,
    sep = "\t",
    header = TRUE,
```

```
  skip = unique_id,
  nrows = 26
)

data = data[ , colSums(is.na(data))==0]
data = as.data.frame(data)
colnames(data)[1] = "Hours"
colnames(data) = trimws(colnames(data))

data[, 1] = gsub("[[:space:]]", "", data[, 1])

temperatures = c()
max_val = 0
min_val = 0
max_index = 0
min_index = 0

for (i in 2:ncol(data)) {
  if (class(data[25, i]) == "character") {
    data[25, i] <- as.numeric(gsub("[^0-9]", "", data[25, i]))
  }
  if (class(data[26, i]) == "character") {
    data[26, i] <- as.numeric(gsub("[^0-9]", "", data[26, i]))
  }

}

for (i in 2:(ncol(data))) {
  for (j in 1:(nrow(data))) {
    if (j == 25) {
      max_index = data[j, i]
    } else if (j == 26) {
      min_index = data[j, i]
    } else {
      temperatures = c(temperatures, data[j, i])
    }
  }
  max_val = max(temperatures)
  min_val = min(temperatures)

  # construct counters for correct max hours
  correct_max = 0
  incorrect_max = 0

  if (data[max_index, i] == max_val) {
    correct_max = correct_max + 1
  } else{
    incorrect_max = incorrect_max + 1
  }

  # counters for correct min hours
  correct_min = 0
  incorrect_min = 0
```

```
    if (data[min_index, i] == min_val) {
      correct_min = correct_min + 1
    } else{
      incorrect_in = incorrect_min + 1
    }
    temperatures = c()
  }

  data = data[-c(25, 26), ] # remove MAX and MIN rows

  Hours = rep(data[, 1], 12)
  Months = c()
  Temperature = c()

  for (i in 2:ncol(data)) {
    Temperature = c(Temperature, data[, i])
  }

  for (i in 2:ncol(data)) {
    Months = c(Months, rep(colnames(data)[i], 24))
  }

  data = cbind(Hours, Months, Temperature)
  data = as.data.frame(data)

  return(data)
}
# modified version of hourly_individual
open_hourly <- function(filepath, titles) {
  data_list <- list()

  # construct counters for correct max hours
  correct_max = 0
  incorrect_max = 0

  # counters for correct min hours
  correct_min = 0
  incorrect_min = 0

  for (title in titles) {
    unique_id = grep(title, readLines(filepath))
    data = read.table(
      filepath,
      sep = "\t",
      header = TRUE,
      skip = unique_id,
      nrows = 26
    )

    data = data[, colSums(is.na(data)) == 0]
    data = as.data.frame(data)
    colnames(data)[1] = "Hours"
    colnames(data) = trimws(colnames(data))
```

```
data[, 1] = gsub("[[:space:]]", "", data[, 1])

for (i in 2:ncol(data)) { # skip the first column (time)
  if (class(data[25, i]) == "character") { # checking if value we want to compare
    # (should be numerical) is not numerical for some reason
    data[25, i] <-
      as.numeric(gsub("[^0-9]", "", data[25, i])) # replaces character values
    # other than numeric from 0-9
  }
  if (class(data[26, i]) == "character") {
    data[26, i] <- as.numeric(gsub("[^0-9]", "", data[26, i]))
  }
}

temperatures = c()
max_val = 0
min_val = 0
max_index = 0
min_index = 0

for (i in 2:(ncol(data))) { # check every column besides for the first
  for (j in 1:(nrow(data))) { # checks every temp value per hour based on the ith column
    if (j == 25) { # the MAX Hour (variable we want to compare w/) lies on the 25th row
      max_index = data[j, i]
    } else if (j == 26) {
      min_index = data[j, i]
    } else {
      temperatures = c(temperatures, data[j, i]) # append all the temp values we
      # want to compare w/ in a vector
    }
  }
  max_val = max(temperatures) # retrieve max temp for the ith column
  min_val = min(temperatures) # retrieve min temp for the ith column

  if (data[max_index,i] == max_val) { #
    correct_max = correct_max + 1
  } else{
    incorrect_max = incorrect_max + 1
  }

  if (data[min_index,i] == min_val) {
    correct_min = correct_min + 1
  } else{
    incorrect_min = incorrect_min + 1
  }
  temperatures = c()
}

data = data[-c(25, 26),] # remove MAX and MIN rows

Hours = rep(data[, 1], 12) # repeat the hour column 12 times for each month = 288
# total rows
Months = c()
```

```
    Temperature = c() # temperatures values based on title names (Dry, Dew, etc.)

    for (i in 2:ncol(data)) { # extract temp values
      Temperature = c(Temperature, data[, i])
    }

    for (i in 2:ncol(data)) { # extract months
      Months = c(Months, rep(colnames(data)[i], 24))
    }

    data = cbind(Hours, Months, Temperature)
    data_list[[title]] <- data.frame(data) # put each df into a list
  }

  print(
    paste0(
      "Out of the 60 maximum hours that were verified for the five hourly tables ",
      correct_max,
      " were correct."
    )
  )
  print(
    paste0(
      "Out of the 60 minimum hours that were verified for the five hourly tables ",
      correct_min,
      " were correct."
    )
  )

  return(data_list)
}
davis_hourly = open_hourly(
  "~/Documents/UCD Classes/22-23/Spring Q23 Classes/STA 141B/HW1/USA_CA_UC_Davis/davis.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
lapply(davis_hourly, head)
# function for merging hourly data
merge_hourly_data <-
  function(filepath, titles) {
    # title is a vector that contains names of variables
    df = data.frame()

    hours_month = open_hourly_individual(filepath, titles[1]) # opens the hourly
    # data only for the first df
    hours_month = hours_month[,-3] # removes the last column (temp_vals) and keeps
    # the hour and month columns
```

```
df = rbind(df, hours_month) # create a new df w/ only hour and months

for (i in 1:length(titles)) {
  df = cbind(df, do.call(cbind, open_hourly_individual(filepath, titles[i])[3]))
  # column bind the 3rd column (temp_vals) of every df based on titles
}

colnames(df) = c(
  "Hours",
  "Months",
  "Dry_Bulb_Temp",
  "Dew_Point_Temp",
  "Relative_Humidity",
  "Direct_Normal_Solar_Radiation",
  "Wind_Speed"
)

hour_order <-
  c(
    "0:01-1:00",
    "1:01-2:00",
    "2:01-3:00",
    "3:01-4:00",
    "4:01-5:00",
    "5:01-6:00",
    "6:01-7:00",
    "7:01-8:00",
    "8:01-9:00",
    "9:01-10:00",
    "10:01-11:00",
    "11:01-12:00",
    "12:01-13:00",
    "13:01-14:00",
    "14:01-15:00",
    "15:01-16:00",
    "16:01-17:00",
    "17:01-18:00",
    "18:01-19:00",
    "19:01-20:00",
    "20:01-21:00",
    "21:01-22:00",
    "22:01-23:00",
    "23:01-24:00"
  )

df$Hours <-
  factor(df$Hours, levels = hour_order) # set as factor, otherwise R doesn't
# read in by order but by char

month_order <-
  c("Jan",
    "Feb",
    "Mar",
```

```
        "Apr",
        "May",
        "Jun",
        "Jul",
        "Aug",
        "Sep",
        "Oct",
        "Nov",
        "Dec")

    df$Months <- factor(df$Months, levels = month_order)

    for (i in 3:ncol(df)) {
      df[, i] = as.numeric(df[, i])
    }

    df = as.data.frame(df)
    return(df)
  }
davis_hourly_data = merge_hourly_data(
  "USA_CA_UC_Davis/davis.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
davis_hourly_data = head(davis_hourly_data)
davis_hourly_data
# plotting function for hourly data vs temperatures for each location
library(ggplot2)
library(scales)

plot_hourly <- function(filename, titles) {
  hourly_data = merge_hourly_data(filename, titles)

  for (i in 1:length(titles)) {
    if (titles[i] == "Average Hourly Statistics for Dry Bulb temperatures") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Dry_Bulb_Temp, color = Months
          )) + theme(axis.text.x = element_text(angle = 90)) # flips the x-axis
        # names vertically for easier user read
      )
    } else if (titles[i] == "Average Hourly Statistics for Dew Point temperatures") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Dew_Point_Temp, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
```

```
      )
    } else if (titles[i] == "Average Hourly Relative Humidity") {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Relative_Humidity, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
      )
    } else if (titles[i] == "Average Hourly Statistics for Direct Normal Solar Radiation") {
      print(
        ggplot() +
          geom_point(
            data = hourly_data,
            aes(x = Hours, y = Direct_Normal_Solar_Radiation, color = Months)
          ) + theme(axis.text.x = element_text(angle = 90))
      )
    } else {
      print(
        ggplot() +
          geom_point(data = hourly_data, aes(
            x = Hours, y = Wind_Speed, color = Months
          )) + theme(axis.text.x = element_text(angle = 90))
      )
    }
  }
}
fairfield_hourly_plots = plot_hourly(
  "USA_CA_Fairfield/fairfield.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
napa_hourly_plots = plot_hourly(
  "USA_CA_Napa/napa.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
davis_hourly_plots = plot_hourly(
  "USA_CA_UC_Davis/davis.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
```

```
    "Average Hourly Statistics for Wind Speed"
  )
)
point_reyes_hourly_plots = plot_hourly(
  "USA_CA_Point/point_reyes.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
marin_hourly_plots = plot_hourly(
  "USA_CA_Marin/marin.stat",
  c(
    "Average Hourly Statistics for Dry Bulb temperatures",
    "Average Hourly Statistics for Dew Point temperatures",
    "Average Hourly Relative Humidity",
    "Average Hourly Statistics for Direct Normal Solar Radiation",
    "Average Hourly Statistics for Wind Speed"
  )
)
```