# Assignment 1 — STA 141B

Filip Wilhelm Sjostrand

2023-04-17

## Introduction

My starting point for the task was to create functions that could easily take on more or less zip files or table names. Thus fundamentally, we need the path and name of our zip files. As we will see later, we also need the name of the table names among all of the possibilites in stat files.

```r
# Raw info ----------
source("/Users/filipsjostrand/Documents/UC Davis/Courses/STA 141B/data/assignment1/functions.R")

data_map <- "/Users/filipsjostrand/Documents/UC Davis/Courses/STA 141B/data/assignment1/"
zip_files <- c(
  "USA_CA_Fairfield-San.Francisco.Bay.Reserve.998011_TMYx.2007-2021.zip",
  "USA_CA_Marin.County.AP-Gnoss.Field.720406_TMYx.2007-2021.zip",
  "USA_CA_Napa.County.AP.724955_TMYx.2007-2021.zip",
  "USA_CA_Point.Reyes.Lighthouse.724959_TMYx.2007-2021.zip",
  "USA_CA_UC-Davis-University.AP.720576_TMYx.2007-2021.zip"
)

table_names <- c(
  "Monthly Statistics for Dry Bulb temperatures",
  "Monthly Statistics for Dew Point temperatures",
  "Average Hourly Statistics for Dry Bulb temperatures",
  "Average Hourly Statistics for Dew Point temperatures",
  "Average Hourly Relative Humidity",
  "Monthly Wind Direction",
  "Average Hourly Statistics for Direct Normal Solar Radiation",
  "Monthly Statistics for Wind Speed",
  "Average Hourly Statistics for Wind Speed"
)
```

The very first function we got is straight forward. It takes the path to the zip files, the name of as many zip files we want as arguments. Further, it also have an optional input to name the maps for the unzipped files—as standard it is the name of our locations. This function only assumes that all zip files are located in the same path, which is easily moderated with only 5 files.

```r
# Running algorithm ----------
paths <- unzipper(data_map, zip_files)
```

## Reading *wea* files

All functions designed to read files begin by using the `type_adder(type, path)`. The function simply take the path to the files, get the basename of the all the files, and append a designated file type. In the first reader below, it appends *wea*.

The `wea_reader()` function is designed to read *wea* files. Since such files are fairly tidy its rather straight forward. This function Assumes that all *wea* files start with 6 rows of meta data, which is to be skipped. Quick manual observation of the files in a text editor confirms that the function indeed only need to skip 6 rows.

Observing the summary tables below we can confirm that all of the date variables are not expressing any non plausible values (e.g. month having max value = 25), nor are the irradiance measures indicating extreme outliers that differs between the tables.

```
wea <- wea_reader(paths)

for (i in 1:length(wea)) {
  print(knitr::kable(summary(wea[[i]]), caption = names(wea)[i]))
  cat('\n\n\n\n')
}
```

```
##
##
## Table: Fairfield
##
## |   |    month       |     day       |     time      |direct_irradiance |diffuse_irradiance |
## |:--|:---------------|:--------------|:--------------|:-----------------|:------------------|
## |   |Min.   : 1.000  |Min.   : 1.00  |Min.   : 1.00  |Min.   :  0.0     |Min.   :  0.00     |
## |   |1st Qu.: 4.000  |1st Qu.: 8.00  |1st Qu.: 6.75  |1st Qu.:  0.0     |1st Qu.:  0.00     |
## |   |Median : 7.000  |Median :16.00  |Median :12.50  |Median :  0.0     |Median :  8.00     |
## |   |Mean   : 6.526  |Mean   :15.72  |Mean   :12.50  |Mean   :283.9     |Mean   : 48.92     |
## |   |3rd Qu.:10.000  |3rd Qu.:23.00  |3rd Qu.:18.25  |3rd Qu.:640.0     |3rd Qu.: 90.00     |
## |   |Max.   :12.000  |Max.   :31.00  |Max.   :24.00  |Max.   :998.0     |Max.   :433.00     |
##
##
##
##
##
##
## Table: Marin
##
## |   |    month       |     day       |     time      |direct_irradiance |diffuse_irradiance |
## |:--|:---------------|:--------------|:--------------|:-----------------|:------------------|
## |   |Min.   : 1.000  |Min.   : 1.00  |Min.   : 1.00  |Min.   :  0.0     |Min.   :  0.00     |
## |   |1st Qu.: 4.000  |1st Qu.: 8.00  |1st Qu.: 6.75  |1st Qu.:  0.0     |1st Qu.:  0.00     |
## |   |Median : 7.000  |Median :16.00  |Median :12.50  |Median :  0.0     |Median :  8.00     |
## |   |Mean   : 6.526  |Mean   :15.72  |Mean   :12.50  |Mean   :280.4     |Mean   : 50.31     |
## |   |3rd Qu.:10.000  |3rd Qu.:23.00  |3rd Qu.:18.25  |3rd Qu.:641.0     |3rd Qu.: 93.00     |
## |   |Max.   :12.000  |Max.   :31.00  |Max.   :24.00  |Max.   :988.0     |Max.   :452.00     |
##
##
##
##
```

```
##
##
## Table: Napa
##
## |   |    month      |     day      |     time     |direct_irradiance |diffuse_irradiance |
## |:--|:--------------|:-------------|:-------------|:-----------------|:------------------|
## |   |Min.   : 1.000 |Min.   : 1.00 |Min.   : 1.00 |Min.   :  0.0     |Min.   :  0.00     |
## |   |1st Qu.: 4.000 |1st Qu.: 8.00 |1st Qu.: 6.75 |1st Qu.:  0.0     |1st Qu.:  0.00     |
## |   |Median : 7.000 |Median :16.00 |Median :12.50 |Median :  0.0     |Median :  7.00     |
## |   |Mean   : 6.526 |Mean   :15.72 |Mean   :12.50 |Mean   :284.8     |Mean   : 47.18     |
## |   |3rd Qu.:10.000 |3rd Qu.:23.00 |3rd Qu.:18.25 |3rd Qu.:659.0     |3rd Qu.: 89.00     |
## |   |Max.   :12.000 |Max.   :31.00 |Max.   :24.00 |Max.   :994.0     |Max.   :414.00     |
##
##
##
##
##
##
## Table: Point-Reyes
##
## |   |    month      |     day      |     time     |direct_irradiance |diffuse_irradiance |
## |:--|:--------------|:-------------|:-------------|:-----------------|:------------------|
## |   |Min.   : 1.000 |Min.   : 1.00 |Min.   : 1.00 |Min.   :  0.0     |Min.   :  0.00     |
## |   |1st Qu.: 4.000 |1st Qu.: 8.00 |1st Qu.: 6.75 |1st Qu.:  0.0     |1st Qu.:  0.00     |
## |   |Median : 7.000 |Median :16.00 |Median :12.50 |Median :  0.0     |Median :  8.00     |
## |   |Mean   : 6.526 |Mean   :15.72 |Mean   :12.50 |Mean   :262.8     |Mean   : 55.39     |
## |   |3rd Qu.:10.000 |3rd Qu.:23.00 |3rd Qu.:18.25 |3rd Qu.:573.0     |3rd Qu.: 96.00     |
## |   |Max.   :12.000 |Max.   :31.00 |Max.   :24.00 |Max.   :991.0     |Max.   :428.00     |
##
##
##
##
##
##
## Table: UC-Davis
##
## |   |    month      |     day      |     time     |direct_irradiance |diffuse_irradiance |
## |:--|:--------------|:-------------|:-------------|:-----------------|:------------------|
## |   |Min.   : 1.000 |Min.   : 1.00 |Min.   : 1.00 |Min.   :  0.0     |Min.   :  0.00     |
## |   |1st Qu.: 4.000 |1st Qu.: 8.00 |1st Qu.: 6.75 |1st Qu.:  0.0     |1st Qu.:  0.00     |
## |   |Median : 7.000 |Median :16.00 |Median :12.50 |Median :  0.0     |Median :  9.00     |
## |   |Mean   : 6.526 |Mean   :15.72 |Mean   :12.50 |Mean   :293.9     |Mean   : 47.09     |
## |   |3rd Qu.:10.000 |3rd Qu.:23.00 |3rd Qu.:18.25 |3rd Qu.:676.0     |3rd Qu.: 88.00     |
## |   |Max.   :12.000 |Max.   :31.00 |Max.   :24.00 |Max.   :985.0     |Max.   :479.00     |
```

## Reading *pvsyst* files

The *pvsyst* files are fairly similar to the *wea* in the sense that they start with some meta data and then followed by a single table. We assume that all *pvsyst* files contain 12 rows of meta data, which was easily confirmed using the code below.

```r
all_pvsyst <- type_adder("pvsyst", paths$base_names)
lapply(all_pvsyst, read_lines, n_max=13)
```

The second step was to extract the column names. Since the row following the column names are the column units, we shall first append the two rows. From there one its straightforward reading the tables and setting the column names to fit the one's we extracted.

```r
pvsyst <- pvsyst_reader(paths)
```

We confirm the result similarly as the tables above, using summary tables. Non of the values appear to be extreme nor implausible. Only one table is printed to keep the report concise.

```r
knitr::kable(summary(pvsyst[[1]]), caption = names(pvsyst)[1])
```

Table 1: Fairfield

| Year | Month | Day | Hour | Minute | GHI W/m2 | DHI W/m2 | DNI W/m2 | Tamb deg.C | WindVel m/sec | WindDir /xb0 |
|---|---|---|---|---|---|---|---|---|---|---|
| Min. :2059 | Min. : 1.000 | Min. : 1.00 | Min. : 1.00 | Min. :30 | Min. : 0.0 | Min. : 0.00 | Min. : 0.0 | Min. :-1.0 | Min. : 0.000 | Min. : 0.0 |
| 1st Qu.:2059 | 1st Qu.: 4.000 | 1st Qu.: 8.00 | 1st Qu.: 6.75 | 1st Qu.:30 | 1st Qu.: 0.0 | 1st Qu.: 0.00 | 1st Qu.: 0.0 | 1st Qu.:11.0 | 1st Qu.: 2.000 | 1st Qu.:190.0 |
| Median :2059 | Median : 7.000 | Median :16.00 | Median :12.50 | Median :30 | Median : 16.0 | Median : 8.00 | Median : 0.0 | Median :15.0 | Median : 4.000 | Median :270.0 |
| Mean :2059 | Mean : 6.526 | Mean :15.72 | Mean :12.50 | Mean :30 | Mean : 217.8 | Mean : 48.92 | Mean :283.9 | Mean :15.3 | Mean : 3.796 | Mean :230.8 |
| 3rd Qu.:2059 | 3rd Qu.:10.000 | 3rd Qu.:23.00 | 3rd Qu.:18.25 | 3rd Qu.:30 | 3rd Qu.: 408.0 | 3rd Qu.: 90.00 | 3rd Qu.:640.0 | 3rd Qu.:19.0 | 3rd Qu.: 6.000 | 3rd Qu.:290.0 |
| Max. :2059 | Max. :12.000 | Max. :31.00 | Max. :24.00 | Max. :30 | Max. :1016.0 | Max. :433.00 | Max. :998.0 | Max. :38.0 | Max. :12.000 | Max. :360.0 |

## Reading *stat* files

The *stat* files are rather intricate. The files consist of a couple of rows of meta data, followed by several tables in the same file. The approach here is to first being able to isolate specified tables from all of the tables in a *stat* file. Further, each isolated file needs to be tidy. For the hourly data, we need to (1) control the max/min values and (2) combine all of those tables.

Following this procedure, the `stat_reader()` function first calls `type_adder()` followed by `read_lines()` to get the raw data of the tables. Then, `isolator()` is called by providing it the raw data and the names of the tables we wish to read. The basic approach of `isolator()` is that we assume all of the tables in a *stat* file either start with "Monthly" or "Average". By extracting where all of the tables start in the *stat* file we can simply start at the index of a desired table and read lines until we hit the next table in the file. By visually observing if our assumption hold we can confirm that it is technically not true that all tables start with those two terms—given that one of the table start with "Wind". However, the algorithm still works given our desired table names, since neither fall between the "Wind" table. I.e., the "Wind" table will be considered as an extension to the previous table and therefore not interfere with our isolating algorithm. Should be noted that if we decide to add a new desired table, we need to keep in mind if that table is the preceding, the actual, or the following table of "Wind"-table. Now, we have untidy, but isolated tables.

next in line, `data_frame_format()` is called. Since our data need to be transposed the function handles that. More specifically, It begin by adding a dummy column name to the column containing all the actual column names, thus, when we transpose we have the correct dimensions to our data. In order to do so, we must first shift each column name once to the right and insert the dummy name leftmost. Now, there are two approaches: for the hourly data we transpose, make the first row the column names, make the row names into its own column, and remove any rows that is completely NA's; for monthly data we do the same procedure but make sure each column has a unique name. The reason we have a `supressWatnings()` around our `stat_reader()` is that the rows that contains all NA's throws a warning, but since they immediately removed it can be ignored.

The following functions calls intends to make the hourly tables into a tidy format. First `max_min_correct()` is called to ensure we have no discrepancies between the rows containing that info and therefore can be omitted. The function basically finds all possible alternatives of a min or max time frame, take that frame as an integer (e.g., 00:01-01:00 becomes 1) and stores it for each month, it then compares the vector of alternatives with the value stored in the max/min row. If there is a discrepancy we are thrown an error message explaining in which table, for which month, if it is min or max, and what the difference is. If no error is thrown we can confirm that each table is correct.

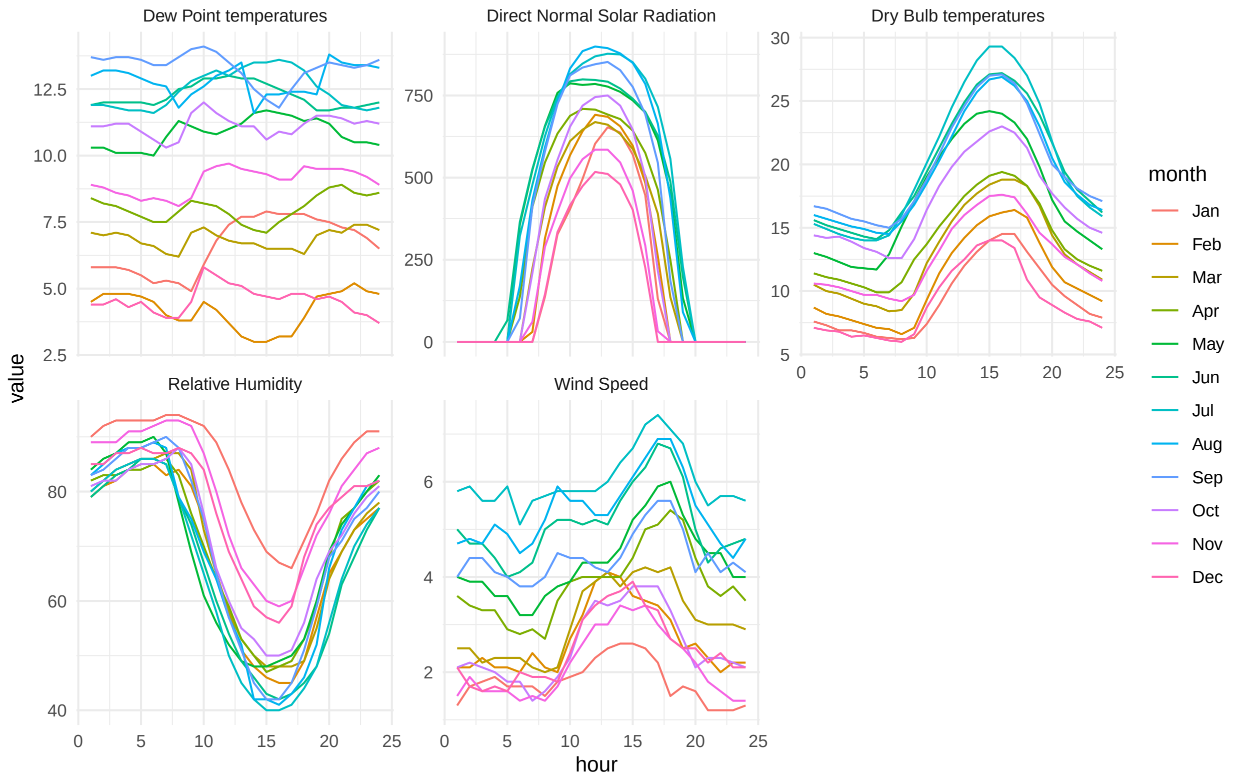
Further, the `tidy_hourly()` is called to make all of our hourly tables into a three column table, where the first column is the month, the second is the time-frame, and the third is the value. This makes it easier to merge the hourly tables later. Thus, when `compact_hourly()` is called we have the right format for using `left_joins()` to merge the tables. `tidy_monthly()` perform similar function as `tidy_hourly()`, with the exception that it makes sure all the columns have their correct data type. This finishes `stat_reader()` which returns a list of all stat tables.

```r
stat <- suppressWarnings(stat_reader(paths))
```

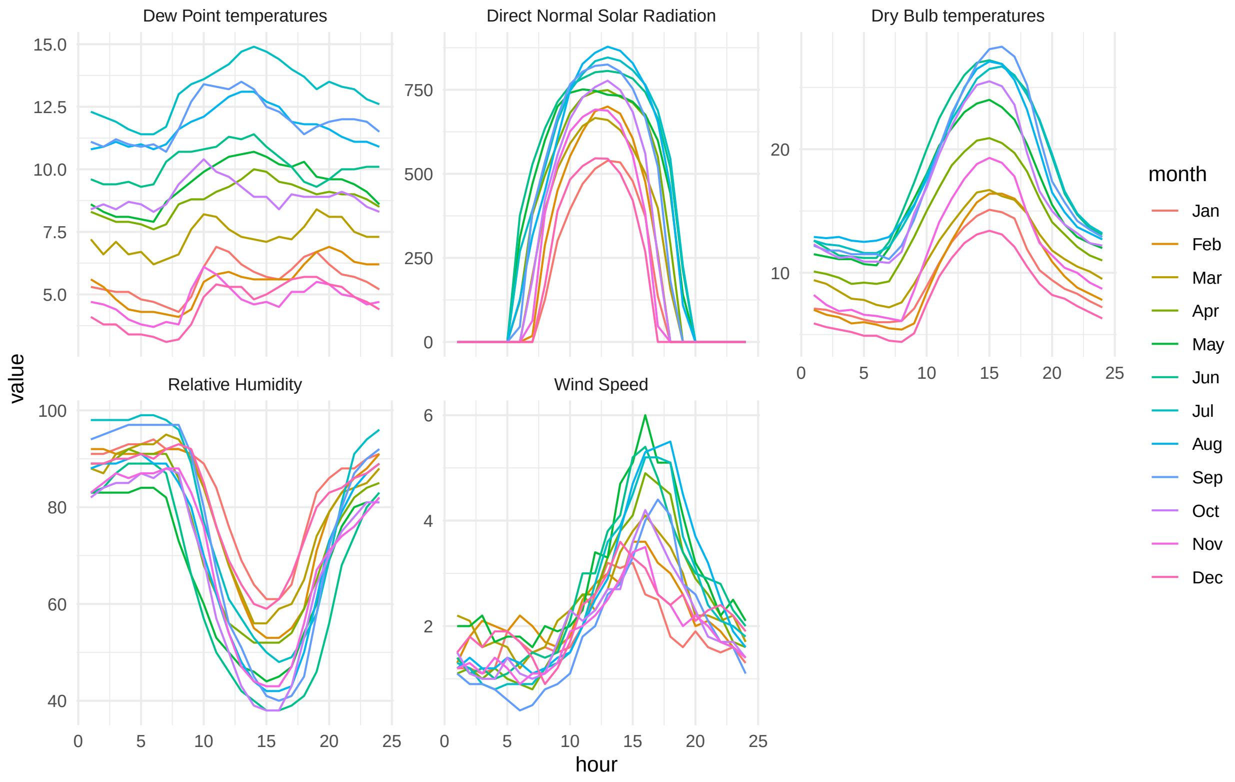
Last but not least, we have the `plotter()` function that plots the hourly data values of stat files. The function begin by pivoting longer such that we can facet more efficiently (given that we would accumulate a lot of plots quickly otherwise). Observing the plots they all tend to signal very similar patterns throughout variables, hours, and months. However, Point-Reyes seems to produce slightly different trends than the rest. Given that Point-Reyes is next to the shore it could contribute to slightly different readings.

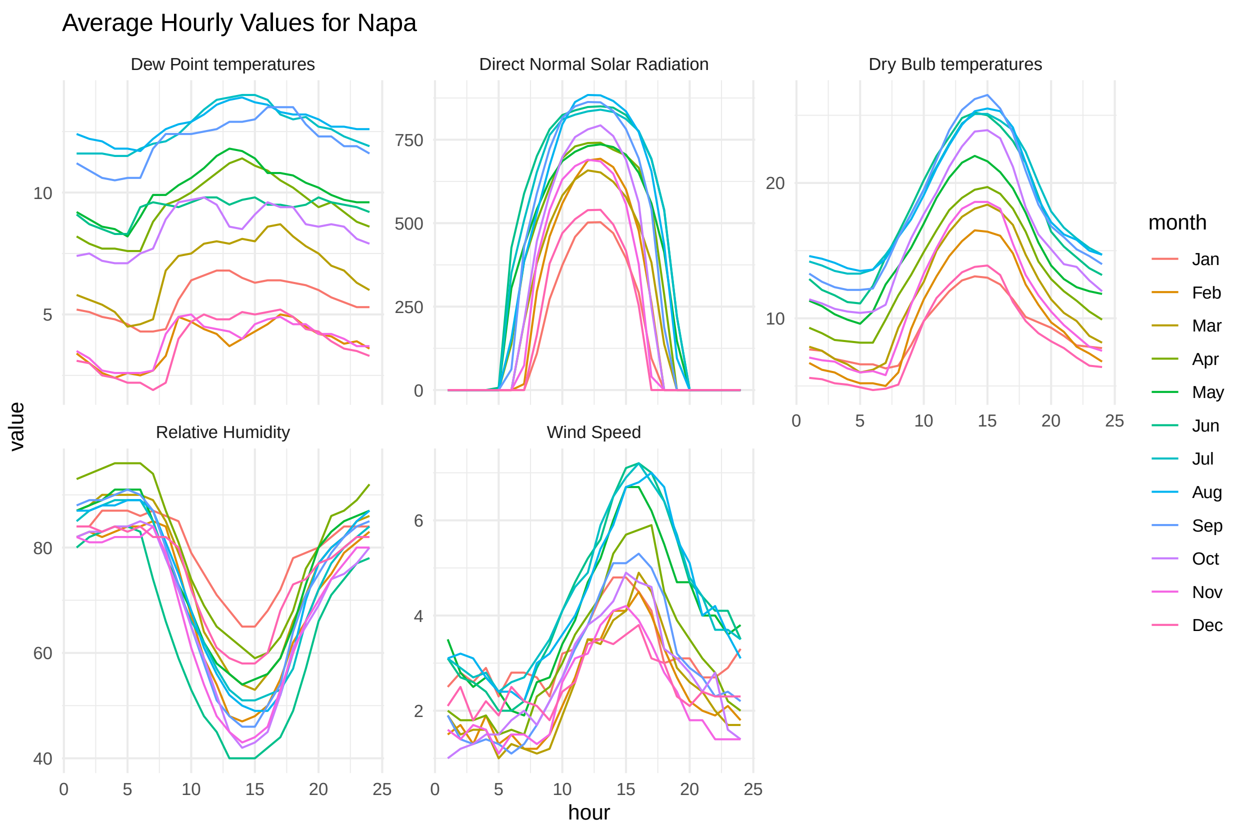
```r
plots <- list()
for (i in 1:length(stat)) {
  df <- stat[[i]][[5]]
  plot_name <- names(stat)[i]
  plots[[paths[[3]][i]]] <- plotter(df, plot_name)
}
```

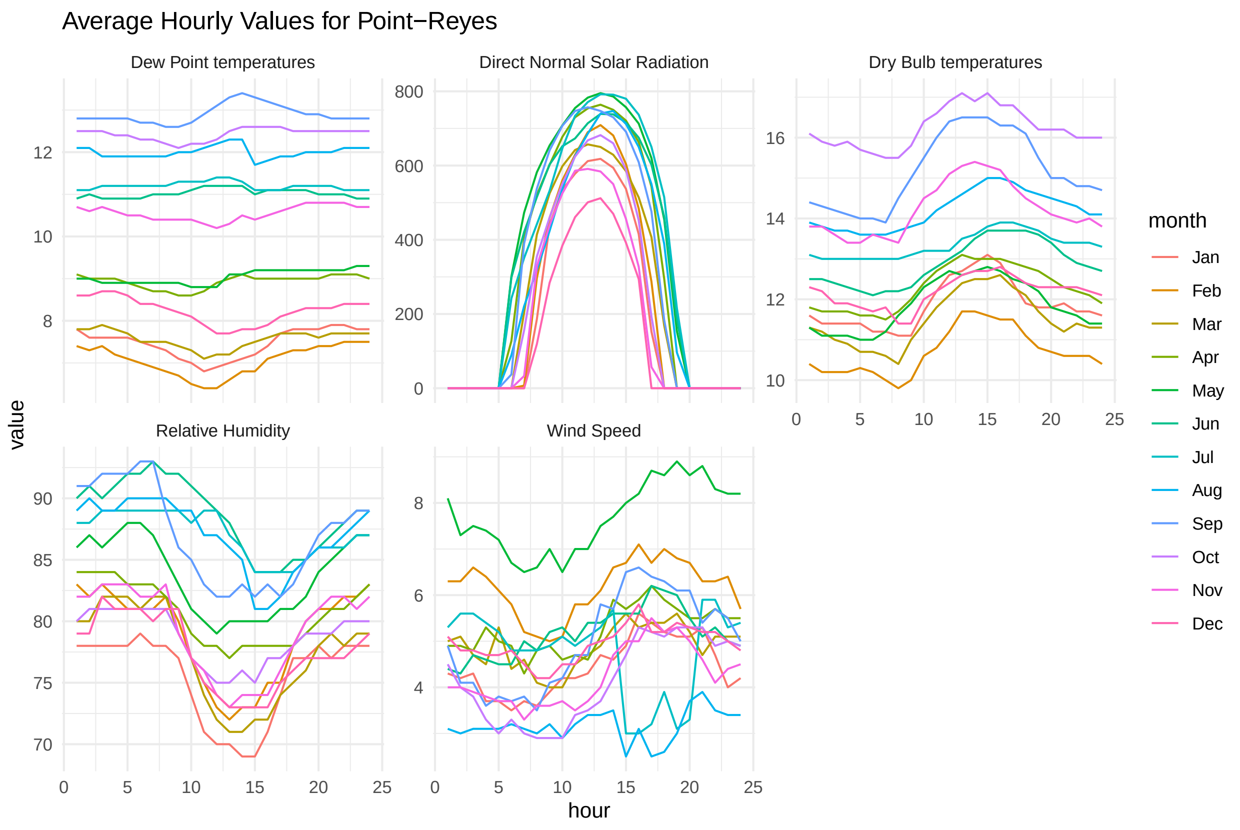
Average Hourly Values for Fairfield

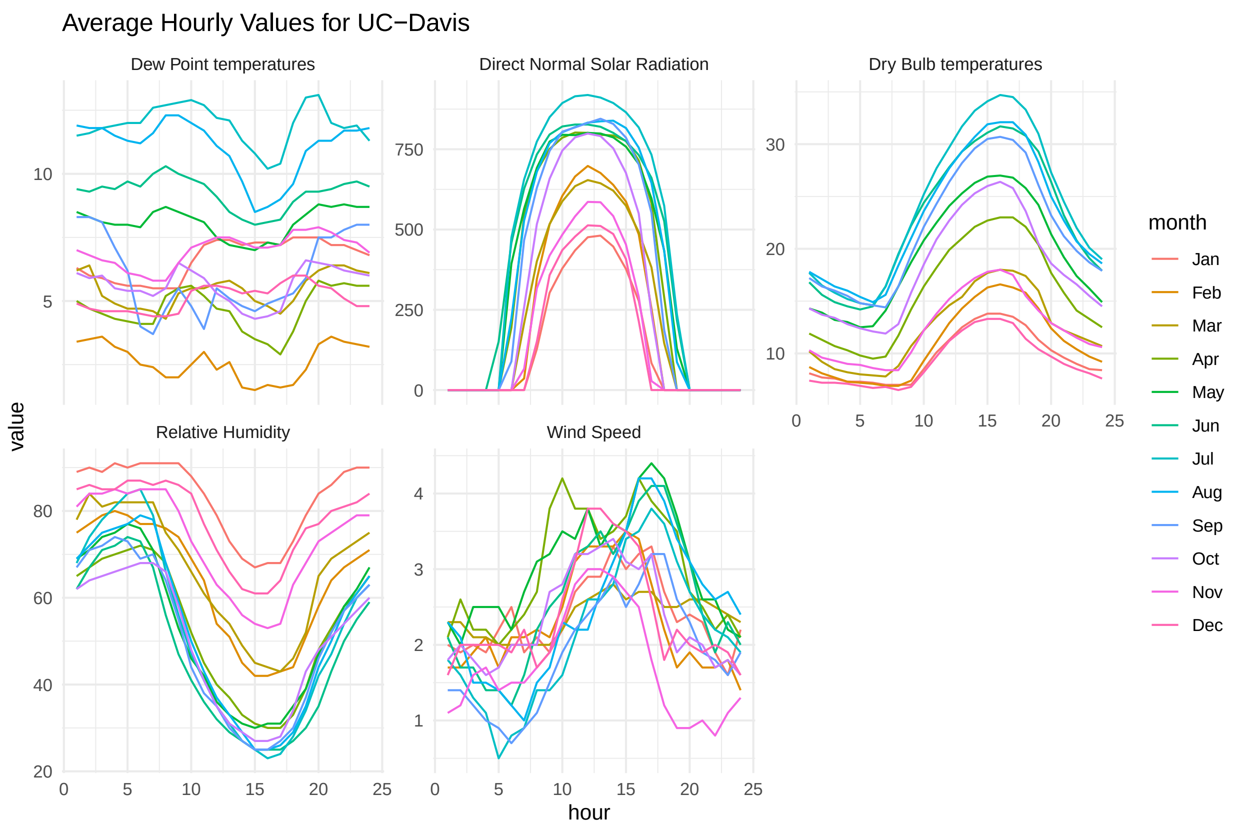


Average Hourly Values for Marin




Average Hourly Values for Napa
Dew Point temperatures
Direct Normal Solar Radiation
Dry Bulb temperatures
Relative Humidity
Wind Speed
value
hour
month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Average Hourly Values for Point−Reyes
Dew Point temperatures
Direct Normal Solar Radiation
Dry Bulb temperatures
Relative Humidity
Wind Speed
value
hour
month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec


Average Hourly Values for UC−Davis
Dew Point temperatures
Direct Normal Solar Radiation
Dry Bulb temperatures
Relative Humidity
Wind Speed
value
hour
month
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec