Downloading and processing NOAA hourly weather station data

Version *0.1-1*

Paul L. Delamater, Andrew O. Finley, and Chad Babcock

1 About NOAA climate data

The National Oceanic and Atmospheric Administration (NOAA) offers a variety of free meteorological/climatological information through their data portal, the National Climatic Data Center (NCDC). The data available for download can be found at the Online Climate Data Directory (OCDD). This page offers HTML and FTP pages to access an assortment of raw sensor data and derived data products. This tutorial will focus on the Integrated Surface Hourly data available for the United States. Because many human/environment studies are conducted over large geographical areas (and also over long periods of time), we will show how to automate the data collection and processing using `R`.

2 Obtaining NOAA weather station information

Weather station data can be accessed from the OCDD. If only a small amount of data are necessary, the "Web Page" link offers a menu driven system for querying the database. For queries covering a large spatial and/or temporal coverage, the data can be more efficiently accessed through the OCDD FTP site. For the Integrated Surface Hourly Data, FTP access is available here: `ftp://ftp.ncdc.noaa.gov/pub/data/noaa/`. Once at the FTP location, the file "ish-history.csv" can be downloaded manually. However, this tutorial includes `R` to automate this process. The history file contains a list of all stations that have collected or are currently collecting hourly data, along with their location and the dates of data collection. It will be used to create custom **wget** commands that will automatically download data for a specific geographic area over specific temporal bounds. First, we read the file into `R` and modify some of the content to a more user-friendly format. Then, we subset the data to include only stations located in the US. In this tutorial we will use the **rgdal**, **spdep**, **sp**, **fields**, and **MBA** libraries in `R`.

```
> file <- "ftp://ftp.ncdc.noaa.gov/pub/data/noaa/ish-history.csv"
> repeat {
+     try(download.file(file, "data/ish-history.csv",
+         quiet = TRUE))
```

```
+     if (file.info("data/ish-history.csv")$size >
+         0) {
+         break
+     }
+ }
> st <- read.csv("data/ish-history.csv")
> dim(st)
```

```
[1] 30538    12
```

```
> names(st)
```

```
 [1] "USAF"         "WBAN"         "STATION.NAME" "CTRY"
 [5] "FIPS"         "STATE"        "CALL"         "LAT"
 [9] "LON"          "ELEV..1M."    "BEGIN"        "END"
```

```
> names(st)[c(3, 10)] <- c("NAME", "ELEV")
> st <- st[, -5]
> st <- st[st$CTRY == "US", ]
```

After checking the file "ish-history.txt" (from the same FTP location as ish-history.csv), we can see that latitude and longitude are in decimal degrees (multiplied by 1000), elevation is in meters (multiplied by 10), and the begin and end dates are in YYYYMMDD format. Because the actual weather station data files are only available per station and year, we can discard the "MMDD" part of the dates (i.e., if a station only collected data for one day in a calendar year, it will have a data file associated with that year).

```
> st$LAT <- st$LAT/1000
> st$LON <- st$LON/1000
> st$ELEV <- st$ELEV/10
> st$BEGIN <- as.numeric(substr(st$BEGIN, 1, 4))
> st$END <- as.numeric(substr(st$END, 1, 4))
```

Unfortunately, the station data are often key parameters, which can lead to a variety of processing errors. For example, a quick visual inspection shows that there are many stations that do not have a state, begin date, or end date assigned to them and some of the latitude and longitude locations are likely incorrect. A number of solutions for this can be formulated, but for this tutorial, we will ignore records without a state listed (but will keep those without dates).

```
> dim(st)
```

```
[1] 7369   11
```

```
> sum(is.na(st$BEGIN))
```

```
[1] 894
```

```
> sum(is.na(st$END))
[1] 894
> sum(st$STATE == "")
[1] 234
```

3 Obtaining NOAA weather station data

Now that station data are in a format that can be queried by state abbreviation or by latitude/longitude and temporally by begin and end date, we can create a custom **wget** command in R. The **wget** call is a simple "request" for data passed through the internet. In this example, we will collect all the hourly data for stations in Michigan during the year 2005. First, subset the stations to those that match these parameters. For reasons of brevity in this example, we do not include the stations with "NA" values for begin and end date. However, including them in the **wget** call will not cause any errors. The **wget** command will simply not return data if the file does not exist.

```
> mi.list <- st[st$STATE == "MI" & (st$BEGIN <= 2005 &
+     st$END >= 2005 & !is.na(st$BEGIN)), ]
> dim(mi.list)
[1] 88 11
```

If the desired data were for multiple years (and including the "NA" data), the subset command would be,

```
> mi.list.2 <- st[st$STATE == "MI" & (st$BEGIN <= 2009 &
+     st$END >= 2002 | is.na(st$BEGIN)), ]
> dim(mi.list.2)
[1] 198  11
```

Next, we use a system call from within R to execute the wget call to the NOAA ftp site using information from the station file. The following example only downloads data for one year (approximate download time is 3-4 minutes). For multiple years, simply change the begin and end year in the "for" statement. Occasionally, NOAA's ftp site struggles to serve information, therefore this code creates a holder with the file name and status of the download. If the file is downloaded, the entry in the STATUS field is 0, and if the file is not downloaded or does not exist, 256.

```
> outputs <- as.data.frame(matrix(NA, dim(mi.list)[1],
+     2))
> names(outputs) <- c("FILE", "STATUS")
> for (y in 2005:2005) {
```

```
+     y.mi.list <- mi.list[mi.list$BEGIN <= y & mi.list$END >=
+         y, ]
+     for (s in 1:dim(y.mi.list)[1]) {
+         outputs[s, 1] <- paste(sprintf("%06d", y.mi.list[s,
+             1]), "-", sprintf("%05d", y.mi.list[s,
+             2]), "-", y, ".gz", sep = "")
+         wget <- paste("wget -P data/raw ftp://ftp.ncdc.noaa.gov/pub/data/noaa/",
+             y, "/", outputs[s, 1], sep = "")
+         outputs[s, 2] <- try(system(wget, intern = FALSE,
+             ignore.stderr = TRUE))
+     }
+ }
> head(outputs)
```

```
                 FILE STATUS
1 692304-99999-2005.gz    256
2 720113-99999-2005.gz      0
3 720198-99999-2005.gz      0
4 720275-99999-2005.gz      0
5 720284-99999-2005.gz      0
6 720321-99999-2005.gz      0
```

```
> sum(outputs$STATUS == 256)
```

```
[1] 3
```

```
> sum(outputs$STATUS == 0)
```

```
[1] 85
```

4 Importing NOAA weather station data

Once the raw data have been downloaded, the files need to be decompressed before importing. Again, we use `R`'s ability to make system commands (note: the decompressed, raw data files in this tutorial use roughly 500Mb of storage space).

```
> system("gunzip -r data/raw", intern = FALSE, ignore.stderr = TRUE)
```

The detailed data format information for the hourly files can be found here. The files are in fixed-width format with many columns of data. The following code will import each of the files, subset to the desired measures, convert units of selected measures, and write the data out as .csv files (approximate processing time is 8 minutes, based on 85 files). This code also writes the geographic coordinates of each station to a separate table which will be used to make a map of the stations.

```
> files <- list.files("data/raw")
> column.widths <- c(4, 6, 5, 4, 2, 2, 2, 2, 1, 6,
+     7, 5, 5, 5, 4, 3, 1, 1, 4, 1, 5, 1, 1, 1, 6,
+     1, 1, 1, 5, 1, 5, 1, 5, 1)
> stations <- as.data.frame(matrix(NA, length(files),
+     6))
> names(stations) <- c("USAFID", "WBAN", "YR", "LAT",
+     "LONG", "ELEV")
> for (i in 1:length(files)) {
+     data <- read.fwf(paste("data/raw/", files[i],
+         sep = ""), column.widths)
+     data <- data[, c(2:8, 10:11, 13, 16, 19, 29,
+         31, 33)]
+     names(data) <- c("USAFID", "WBAN", "YR", "M",
+         "D", "HR", "MIN", "LAT", "LONG", "ELEV",
+         "WIND.DIR", "WIND.SPD", "TEMP", "DEW.POINT",
+         "ATM.PRES")
+     data$LAT <- data$LAT/1000
+     data$LONG <- data$LONG/1000
+     data$WIND.SPD <- data$WIND.SPD/10
+     data$TEMP <- data$TEMP/10
+     data$DEW.POINT <- data$DEW.POINT/10
+     data$ATM.PRES <- data$ATM.PRES/10
+     write.csv(data, file = paste("data/csv/", files[i],
+         ".csv", sep = ""), row.names = FALSE)
+     stations[i, 1:3] <- data[1, 1:3]
+     stations[i, 4:6] <- data[1, 8:10]
+ }
> write.csv(stations, file = "data/stations.csv", row.names = FALSE)
> mi.state <- readOGR("data/gis/.", "michigan", verbose = FALSE)
> mi <- mi.state[mi.state$AREA > 0.01, ]
> plot(mi, xlab = "Degrees", ylab = "Degrees", axes = T)
> points(stations$LONG, stations$LAT, pch = 16, col = "red")
```

5 Processing NOAA weather station data

Now that data have been converted to a more accessible file format, they can be processed similar to other point-referenced data. In the next step, we will create yearly plots of temperature and wind speed at one station. However, take note that the file sizes of the .csv files are not consistent. This is largely due to a number of stations that collect data at 15 minute intervals as opposed to hourly (and sometimes due to stations going off-line). When processing the data, we must pay attention to which of the weather variables are collected and the temporal resolution of their collection.

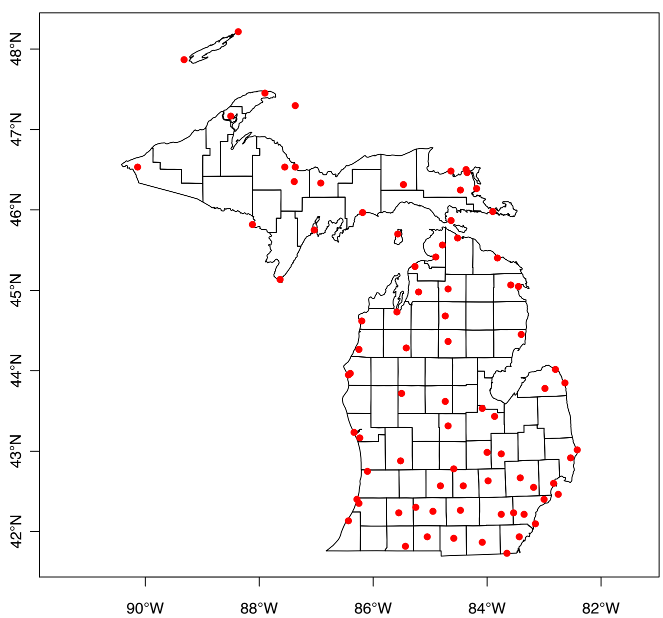

Figure 1: NOAA hourly station locations.

```
> files <- list.files("data/csv")
> st <- read.csv(file = paste("data/csv/", files[85],
+     sep = ""))
> head(st)
```

```
  USAFID  WBAN   YR M D HR MIN    LAT    LONG ELEV WIND.DIR
1 726355 99999 2005 1 1  0   0 42.133 -86.433  196      999
2 726355 99999 2005 1 1  0  53 42.133 -86.433  196      260
3 726355 99999 2005 1 1  1   0 42.133 -86.433  196      999
4 726355 99999 2005 1 1  1  53 42.133 -86.433  196      270
5 726355 99999 2005 1 1  2   0 42.133 -86.433  196      999
6 726355 99999 2005 1 1  2  53 42.133 -86.433  196      290
  WIND.SPD  TEMP DEW.POINT ATM.PRES
1    999.9 999.9     999.9   9999.9
2      4.6   6.0       0.0   1021.0
3    999.9 999.9     999.9   9999.9
4      4.1   4.0       0.0   1022.3
5    999.9 999.9     999.9   9999.9
6      6.2   4.0       0.0   1023.4
```

```
> dim(st)
```

```
[1] 20520    15
```

The first few entries in the table show us that this particular station collects data every 15 minutes. However, we can see many NoData flags in the table (999, 999.9, 9999.9). Because the first few entries of wind direction, dew point, and atmospheric pressure are all NoData, it might be useful test whether this station even collected information for those variables. Then, subset to only the necessary variables, set the NoData entries to "NA", and make plots of temperature and wind speed. Because neither temperature or wind speed was collected consistently at the 15 minute interval, we can might just focus on the hourly data (i.e., remove any values where minutes are equal to 15, 30, or 45).

```
> sum(st$TEMP == 999.9)
```

```
[1] 7987
```

```
> sum(st$WIND.SPD == 999.9)
```

```
[1] 7963
```

```
> sum(st$WIND.DIR == 999)
```

```
[1] 11080
```

```
> sum(st$DEW.POINT == 999.9)
```

```
[1] 8005
```

```
> sum(st$ATM.PRES == 9999.9)

[1] 11865

> st$WIND.DIR <- st$DEW.POINT <- st$ATM.PRES <- NULL
> st$TEMP[st$TEMP == 999.9] <- NA
> st$WIND.SPD[st$WIND.SPD == 999.9] <- NA
> st <- st[st$MIN == 0, ]
> dim(st)

[1] 8168   12

> 365 * 24

[1] 8760

> st <- st[order(st$M, st$D, st$HR), ]
> st$DATE <- as.Date(paste(st$YR, st$M, st$D, sep = "-"),
+     format = "%Y-%m-%d")
> d.mean <- aggregate(st$TEMP, list(DATE = format(st$DATE,
+     "%Y-%m-%d")), mean, na.rm = T)
> m.mean <- aggregate(st$TEMP, list(DATE = format(st$DATE,
+     "%Y-%m")), mean, na.rm = T)
> d.mean$DATE <- as.Date(d.mean$DATE)
> m.mean$DATE <- as.Date(paste(m.mean$DATE, "-15",
+     sep = ""))
> plot(st$DATE, st$TEMP, main = "Temperature readings",
+     ylab = "Temperature (Degrees C)", xlab = "Month",
+     col = "grey")
> points(d.mean$DATE, d.mean$x, col = "brown")
> lines(m.mean$DATE, m.mean$x, type = "b", pch = 16)
> legend("topleft", c("Hourly", "Daily mean", "Monthly mean"),
+     inset = 0.02, pch = c(1, 1, 16), col = c("grey",
+         "red", "black"))


> d.mean <- aggregate(st$WIND.SPD, list(DATE = format(st$DATE,
+     "%Y-%m-%d")), mean, na.rm = T)
> m.mean <- aggregate(st$WIND.SPD, list(DATE = format(st$DATE,
+     "%Y-%m")), mean, na.rm = T)
> d.mean$DATE <- as.Date(d.mean$DATE)
> m.mean$DATE <- as.Date(paste(m.mean$DATE, "-15",
+     sep = ""))
> plot(st$DATE, st$WIND.SPD, main = "Wind speed readings",
+     ylab = "Wind Speed (meters/second)", xlab = "Month",
+     col = "grey")
> points(d.mean$DATE, d.mean$x, col = "brown")
```

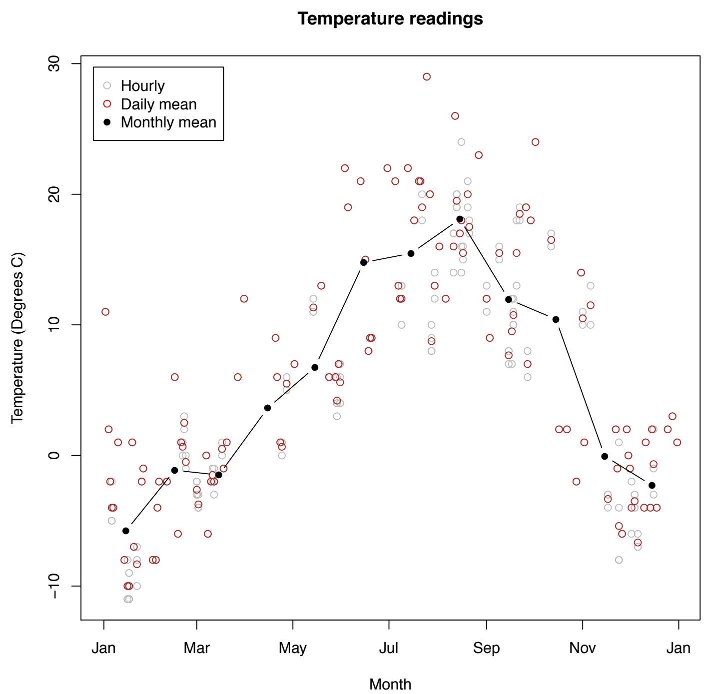

Figure 2: Hourly temperature measurements with daily, monthly means

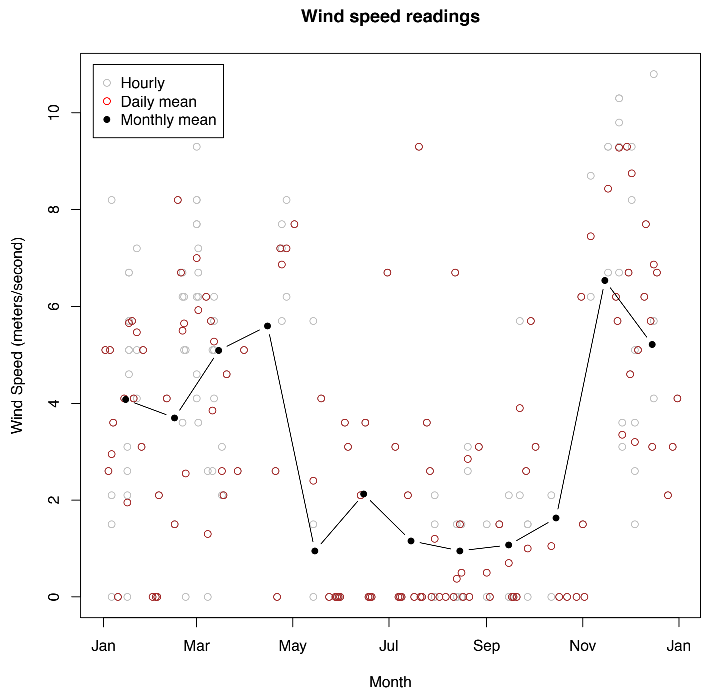

Figure 3: Hourly wind speed measurements with daily, monthly means

```
> lines(m.mean$DATE, m.mean$x, type = "b", pch = 16)
> legend("topleft", c("Hourly", "Daily mean", "Monthly mean"),
+     inset = 0.02, pch = c(1, 1, 16), col = c("grey",
+         "red", "black"))
```

6 Create daily data

In the final section, we will create temperature and wind speed surfaces for Michigan during the summer season of 2005. Before proceeding, the hourly data will be used to create a daily mean measurement at each station. The daily mean point measurements will then be used to create the surfaces for the entire state.

First, create a loop that opens all files and extracts only the hourly measurements taken at the top of the hour for summer dates (June 21 to September 21). Some stations have multiple sensors that measure weather variables. Therefore, the primary instrument must be identified. Also, because the measurements are not always taken exactly on the top of the hour, some observations have to be shifted by a few minutes (e.g., less than 10). Finally, if the station has a nearly complete set of hourly measurements (e.g., 18 or more) for one day, we calculate the daily mean.

```
> files <- list.files("data/csv")
> daily.data <- vector("list", length(files))
> dates <- seq(as.Date("2005-06-21"), as.Date("2005-09-21"),
+     1)
> for (i in 1:length(files)) {
+     daily.data[[i]]$DATE <- dates
+     daily.data[[i]]$TEMP <- daily.data[[i]]$WIND.SPD <- rep(NA,
+         length(dates))
+     daily.data[[i]]$USAFID <- daily.data[[i]]$WBAN <- rep(NA,
+         length(dates))
+     daily.data[[i]]$LAT <- daily.data[[i]]$LONG <- rep(NA,
+         length(dates))
+     daily.data[[i]] <- as.data.frame(daily.data[[i]])
+     st <- read.csv(paste("data/csv/", files[i], sep = ""))
+     st$WIND.DIR <- st$DEW.POINT <- st$ATM.PRES <- NULL
+     st$DATE <- as.Date(paste(st$YR, st$M, st$D, sep = "-"),
+         format = "%Y-%m-%d")
+     st <- st[st$DATE >= "2005-06-21" & st$DATE <=
+         "2005-09-21", ]
+     st$TEMP[st$TEMP == 999.9] <- NA
+     st$WIND.SPD[st$WIND.SPD == 999.9] <- NA
+     st <- st[!is.na(st$TEMP) | !is.na(st$WIND.SPD),
+         ]
+     u <- unique(st$LAT)
+     max.u <- vector("numeric", length(u))
+     for (z in 1:length(u)) {
+         max.u[z] <- sum(st$LAT == u[z])
+     }
+     pos <- which(max.u == max(max.u))
+     st <- st[st$LAT == u[pos], ]
+     if (length(unique(st$LONG)) > 1) {
+         u <- unique(st$LONG)
+         max.u <- vector("numeric", length(u))
+         for (z in 1:length(u)) {
+             max.u[z] <- sum(st$LONG == u[z])
+         }
+         pos <- which(max.u == max(max.u))
+         st <- st[st$LONG == u[pos], ]
+     }
+     if (sum(st$MIN == 0) < 2232) {
+         st$HR[st$MIN > 50] <- st$HR[st$MIN > 50] +
+             1
```

```
+         st$MIN[st$MIN > 50] <- 0
+         st$MIN[st$MIN < 10] <- 0
+         st$DATE[st$MIN == 24] <- st$DATE[st$MIN ==
+             24] + 1
+         st$YR <- as.numeric(format(st$DATE, "%Y"))
+         st$M <- as.numeric(format(st$DATE, "%m"))
+         st$D <- as.numeric(format(st$DATE, "%d"))
+     }
+     st <- st[st$MIN == 0, ]
+     daily.data[[i]]$USAFID <- rep(st$USAFID[1], length(dates))
+     daily.data[[i]]$WBAN <- rep(st$WBAN[1], length(dates))
+     daily.data[[i]]$LAT <- rep(st$LAT[1], length(dates))
+     daily.data[[i]]$LONG <- rep(st$LONG[1], length(dates))
+     for (j in dates) {
+         sub.st <- st[st$DATE == j, ]
+         if (dim(sub.st)[1] >= 18) {
+             daily.data[[i]]$TEMP[daily.data[[i]]$DATE ==
+                 j] <- mean(sub.st$TEMP)
+             daily.data[[i]]$WIND.SPD[daily.data[[i]]$DATE ==
+                 j] <- mean(sub.st$WIND.SPD)
+         }
+     }
+ }
```

Aggregate the data by the number of daily observations per variable and attach to the station table created earlier. Then, create maps that illustrate the number of observations per station for temperature and wind speed during the summer of 2005.

```
> stations$TEMP.OBS <- rep(NA, dim(stations)[1])
> stations$WIND.SPD.OBS <- stations$TEMP.OBS
> stations$ELEV <- NULL
> for (i in 1:length(daily.data)) {
+     n.temp <- sum(!is.na(daily.data[[i]]$TEMP))
+     n.wind <- sum(!is.na(daily.data[[i]]$WIND.SPD))
+     stations$TEMP.OBS[stations$USAFID == daily.data[[i]]$USAFID[1] &
+         stations$WBAN == daily.data[[i]]$WBAN[1]] <- n.temp
+     stations$WIND.SPD.OBS[stations$USAFID == daily.data[[i]]$USAFID[1] &
+         stations$WBAN == daily.data[[i]]$WBAN[1]] <- n.wind
+     stations$LAT[stations$USAFID == daily.data[[i]]$USAFID[1] &
+         stations$WBAN == daily.data[[i]]$WBAN[1]] <- daily.data[[i]]$LAT[1]
+     stations$LONG[stations$USAFID == daily.data[[i]]$USAFID[1] &
+         stations$WBAN == daily.data[[i]]$WBAN[1]] <- daily.data[[i]]$LONG[1]
+ }
> plot(mi, xlab = "Degrees", ylab = "Degrees", axes = T)
> symbols(stations$LONG, stations$LAT, circles = stations$TEMP.OBS,
+     inches = 0.1, fg = "black", bg = "green", add = T)
```

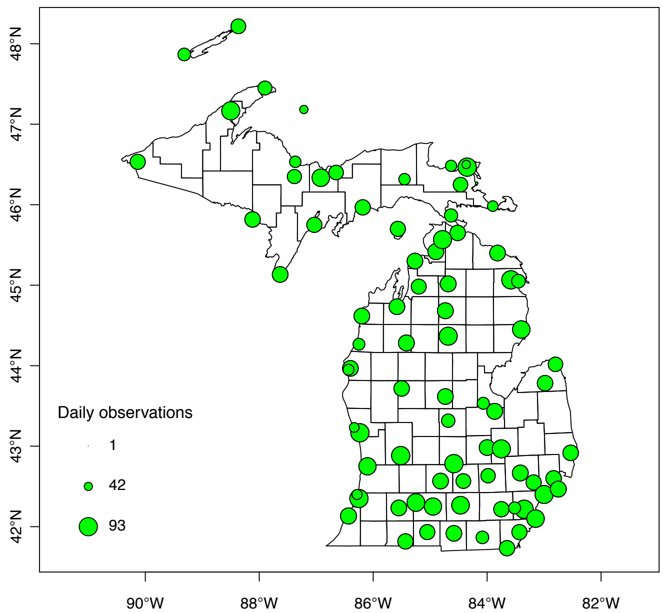

Figure 4: Number of daily temperature observations

```
> symbols(c(-91, -91, -91), c(43, 42.5, 42), circles = c(1,
+     42, 93), inches = 0.1, fg = "black", bg = "green",
+     add = T)
> text(c(-90.8, -90.8, -90.8), c(43, 42.5, 42), c("1",
+     "42", "93"), pos = 4)
> text(-91.7, 43.4, "Daily observations", pos = 4,
+     cex = 1.05)

> plot(mi, xlab = "Degrees", ylab = "Degrees", axes = T)
> symbols(stations$LONG, stations$LAT, circles = stations$WIND.SPD.OBS,
+     inches = 0.1, fg = "black", bg = "red", add = T)
> symbols(c(-91, -91, -91), c(43, 42.5, 42), circles = c(1,
+     42, 93), inches = 0.1, fg = "black", bg = "red",
+     add = T)
> text(c(-90.8, -90.8, -90.8), c(43, 42.5, 42), c("1",
+     "42", "93"), pos = 4)
> text(-91.7, 43.4, "Daily observations", pos = 4,
+     cex = 1.05)
```

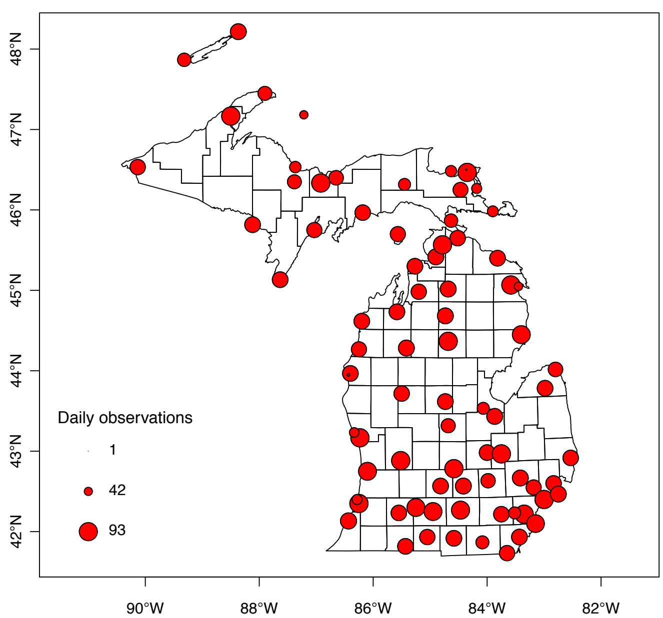

Figure 5: Number of daily wind speed observations

7 Create temperature and wind speed surfaces

Next, we use the daily point observations to interpolate surfaces for Michigan. In an effort to keep processing time low, we will use the **MBA** package and a course pixel resolution for our surfaces. Also note that the data are not in a "projected" coordinate system. Using unprojected spatial data to create surfaces (or any other spatial application) is not recommended outside of this tutorial. First, combine the list of dataframes containing all the daily observations for the year into one large dataframe. We then subset the readings for one day and create the surface. Finally, change the units from metric to English and plot the surfaces and point locations.

```
> daily.obs <- do.call(rbind, daily.data)
> t.obs <- daily.obs[daily.obs$DATE == dates[1] & !is.na(daily.obs$TEMP),
+     c("LONG", "LAT", "TEMP")]
> mba.bbox <- c(-91, -82, 41, 49)
> surf <- mba.surf(t.obs, 75, 75, extend = TRUE, sp = TRUE,
+     b.box = mba.bbox)$xyz.est
> surf@data <- surf@data * (!is.na(overlay(surf, mi)))
> surf$z[surf$z == 0] <- NA
> surf$z <- surf$z * (9/5) + 32
> image.plot(as.image.SpatialGridDataFrame(surf), asp = 1.25)
> title(main = "Temperature surface (degrees F)")
> plot(mi, add = TRUE)
> points(t.obs$LONG, t.obs$LAT, pch = 20)
```

```
> w.obs <- daily.obs[daily.obs$DATE == dates[1] & !is.na(daily.obs$WIND.SPD),
+     c("LONG", "LAT", "WIND.SPD")]
> surf.w <- mba.surf(w.obs, 75, 75, extend = TRUE,
+     sp = TRUE, b.box = mba.bbox)$xyz.est
> surf.w@data <- surf.w@data * (!is.na(overlay(surf.w,
+     mi)))
> surf.w$z[surf.w$z == 0] <- NA
> surf.w$z <- surf.w$z * 2.23693629
> image.plot(as.image.SpatialGridDataFrame(surf.w),
+     asp = 1.25)
> title(main = "Wind speed surface (miles per hour)")
> plot(mi, add = TRUE)
> points(w.obs$LONG, w.obs$LAT, pch = 20)
```

Next, create a loop that cycles through each day of summer and creates a surface using observations from that day. To help speed processing, the surface values will be stored in a dataframe object that can be later attached to the coordinates.

```
> t.surfs <- as.data.frame(matrix(NA, length(surf$z),
+     length(dates)))
```

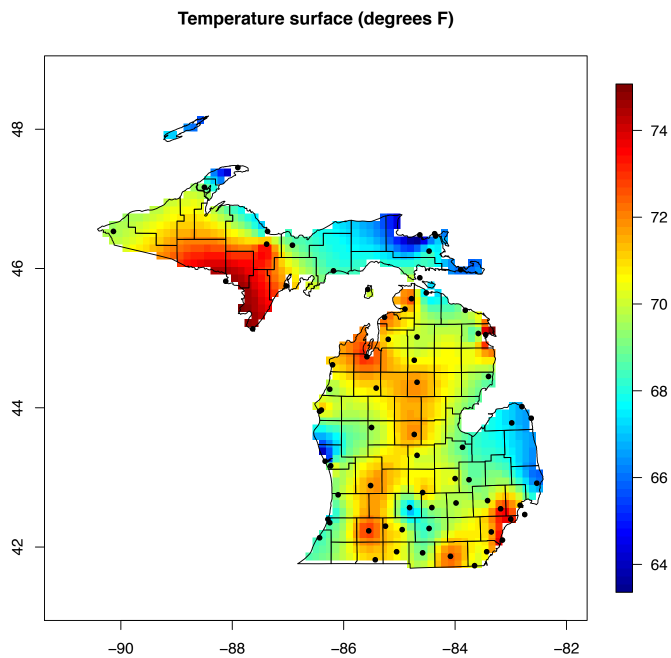

Figure 6: MBA temperature surface clipped to state bounds

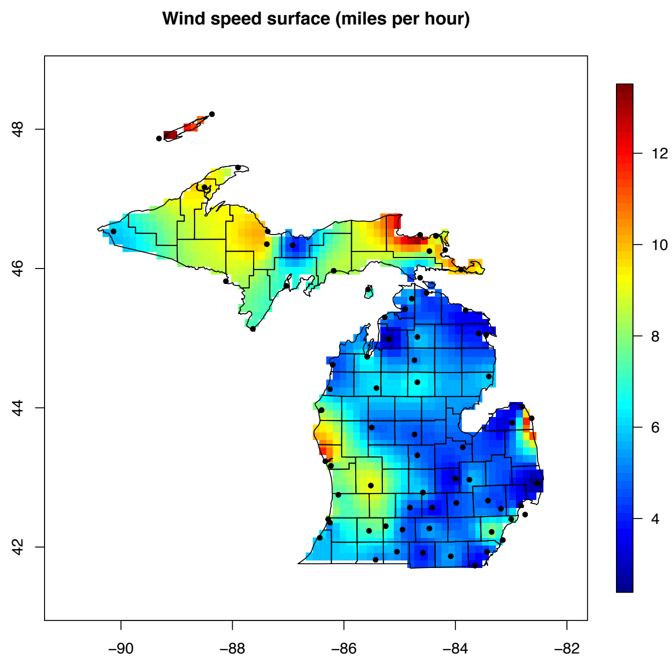

Figure 7: MBA temperature surface clipped to state bounds

```
> w.surfs <- t.surfs
> for (i in 1:length(dates)) {
+     t.obs <- daily.obs[daily.obs$DATE == dates[i] &
+         !is.na(daily.obs$TEMP), c("LONG", "LAT",
+         "TEMP")]
+     w.obs <- daily.obs[daily.obs$DATE == dates[i] &
+         !is.na(daily.obs$WIND.SPD), c("LONG", "LAT",
+         "WIND.SPD")]
+     surf <- mba.surf(t.obs, 75, 75, extend = TRUE,
+         sp = TRUE, b.box = mba.bbox)$xyz.est
+     surf@data <- surf@data * (!is.na(overlay(surf,
+         mi)))
+     surf$z[surf$z == 0] <- NA
+     t.surfs[, i] <- surf$z * (9/5) + 32
+     surf <- mba.surf(w.obs, 75, 75, extend = TRUE,
+         sp = TRUE, b.box = mba.bbox)$xyz.est
+     surf@data <- surf@data * (!is.na(overlay(surf,
+         mi)))
+     surf$z[surf$z == 0] <- NA
+     w.surfs[, i] <- surf$z * 2.23693629
+ }
```

Now, the dataframe objects can be manipulated and attached back to the coordinate locations. For example, we can calculate the mean and standard deviation of daily temperature and wind speed at each cell and map them.

```
> wind.sd <- apply(t.surfs, 1, mean)
> surf$z <- wind.sd
> par(mfrow = c(2, 2))
> image.plot(as.image.SpatialGridDataFrame(surf), asp = 1.25)
> title(main = "Mean daily temperature (deg F)")
> surf$z <- apply(t.surfs, 1, sd)
> image.plot(as.image.SpatialGridDataFrame(surf), asp = 1.25)
> title(main = "St.Dev. of daily temperature (deg F)")
> surf$z <- apply(w.surfs, 1, mean)
> image.plot(as.image.SpatialGridDataFrame(surf), asp = 1.25)
> title(main = "Mean daily wind speed (mph)")
> surf$z <- apply(w.surfs, 1, sd)
> image.plot(as.image.SpatialGridDataFrame(surf), asp = 1.25)
> title(main = "St.Dev. of daily wind speed (mph)")
```

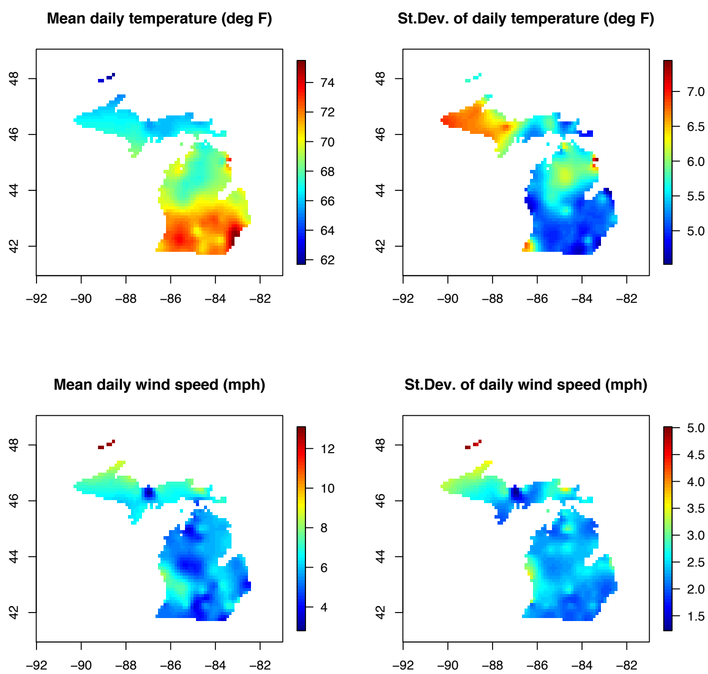

Figure 8: Mean and standard deviation of daily temperature and wind speed surfaces