Ag Carpentry - Weather and Soil Data

Brittani

2019-10-29

Motivating Questions:

- What are the common file types in agricultural data?
- What publicly available datasets exist for my field?

Objectives with Agricultural Data Types

- Describe the format of public weather and soil datasets
- Import weather data from the internet, eg. daymetr
- Access to elevation and ssurgo data with higher resolution
- Derive topography data from elevation data

Keypoints:

- sf is prefereable for data analysis; it is easier to access the dataframe
- Projecting your data in utm is necessary for many of the geometric operations you perform (e.g. making trial grids and splitting plots into subplot data)
- Compare different data formats, such as gpkg, shp(cpg,dbf,prj,sbn,sbx),geojson,tif

Daymet Weather Data

The Oak Ridge National Laboratory produces a datset called Daymet which contains predicted weather observations on a one meter grid. These data come from weather station climate observations in a climate model for prediction and include variables such as precipitation, snow water equivalent, temperature, day length, solar radians, and vapor pressure.

There is a package in r `daymetr` that downloads the daymet weather data within the R environment. For a > single point, you can use the command `download_daymet()`. If you want to download the data for a set of points, there is also the command `download_daymet_batch()` which takes an argument a .csv of the points in lat/long.

We will use the mean latitude and longitude values from the bounding box as our point for the weather data. This should be a point near the middle of the field.

```
boundary <- read_sf("data/boundary.gpkg")
lon <- cent_long(boundary)
lat <- cent_lat(boundary)
```

{: .callout}

We also call the site `Field1`, but this will be the name of a specific field if you use it in the future. We can choose the start and end years. If the data is not available for the year you request, an error will be reported. We choose 2000 to 2018 for this example; later we will use the historical data for comparison. The final option `internal = TRUE` means that the daymet data is brought into the R environment rather than saved in your working directory.

```
weather <- download_daymet(site = "Field1", lat = lat, lon = lon, start = 2000, end = 2018, internal = 
```

```
## Downloading DAYMET data for: Field1 at 41.7430279883331/-81.9757922217347 latitude/longitude !
```

```
## Done !
```

The object `weather` is a list of 7 objects, the last of is the data.

Exercise 1: Explore the weather data

1. Grab this object and save it as `weather_data`.
2. How is the date reported?
3. What other variables exist?
4. What are the units for the different variables? *Remember:* Sometimes you need to use a search engine to understand what objects are created from a specific R function.

Exercise 1 Solutions

```
weather_data <- weather$data
summary(weather_data)
```

```
##       year           yday        dayl..s.      prcp..mm.day.
##  Min.   :2000   Min.   :  1   Min.   :32141   Min.   :  0.000
##  1st Qu.:2004   1st Qu.: 92   1st Qu.:35942   1st Qu.:  0.000
##  Median :2009   Median :183   Median :43200   Median :  0.000
##  Mean   :2009   Mean   :183   Mean   :43198   Mean   :  3.239
##  3rd Qu.:2014   3rd Qu.:274   3rd Qu.:50458   3rd Qu.:  3.000
##  Max.   :2018   Max.   :365   Max.   :54259   Max.   :108.000
##   srad..W.m.2.    swe..kg.m.2.      tmax..deg.c.     tmin..deg.c.
##  Min.   : 44.8   Min.   :  0.000   Min.   :-15.50   Min.   :-25.000
##  1st Qu.:204.8   1st Qu.:  0.000   1st Qu.:  6.00   1st Qu.: -1.000
##  Median :294.4   Median :  0.000   Median : 16.00   Median :  6.500
##  Mean   :297.0   Mean   :  5.659   Mean   : 14.85   Mean   :  6.415
##  3rd Qu.:390.4   3rd Qu.:  0.000   3rd Qu.: 24.50   3rd Qu.: 15.000
##  Max.   :585.6   Max.   :188.000   Max.   : 34.50   Max.   : 26.000
##     vp..Pa.
##  Min.   :  80
##  1st Qu.: 560
##  Median : 960
##  Mean   :1142
##  3rd Qu.:1680
##  Max.   :3360
```

The date is reported as the year and day of the year. Other variables include day length, precipitation, solar radiation, snow water equivalent, maximum temperature, minimum temperature, and vapor pressure. The units for the variables are given after the variable name. For example, day length is in seconds and solar radiation is in watts per square meter. While precipitation and temperature have intuitive names, vapor pressure and snow water equivalent are not so apparent. Use the `datmetr` vignette to understand the meaning of these variables.

Unit Conversions

Publicly available data are usually given in metric units as we saw in the weather data above. We may want to have these data in imperial units as these are the units we often are comparing in the United States; additionally, you may know the value of crop requirements and threshholds in imperial units rather than metric units.

The package `measurements` in R converts observations from one unit to another. The command `conv_unit()` converts the column from one stated unit to another unit. To see the possible units for a specific kind of measure, look at the `conv_unit_options` for the specific measure you are converting (e.g. length, area, weight, etc.).

We can convert the daily precipitation from milimeters to inches using the function `mm_to_in()` which uses `conv_unit()`. The following lines converts `prcp..mm.day.` to inches and creates a new column called `prec`.

```
weather_data$prec <- mm_to_in(weather_data$prcp..mm.day.)
```

Exercise 2: Unit Conversions

1. Convert the two temperature variables into fahrenheit from celsius using the function `c_to_f()` with the names `tmax` and `tmin`.

Execrise 2 Solutions

```
weather_data$tmax <- c_to_f(weather_data$tmax..deg.c.)
weather_data$tmin <- c_to_f(weather_data$tmin..deg.c.)
head(weather_data$tmax)
```

```
## [1] 49.1 58.1 54.5 53.6 34.7 38.3
```

Precipitation Graph

Perhaps you want to see what the weather this year was like compared to the average historic weather for the same area. We will make a graph showing the total monthly precipitation from 2018 compared to the average precipitation from the years 2000 to 2017. This is a common way to look at seasonal rainfall and allows us to look at the rainfall during the critical months of July and August.

Dates in Dataframes

There is a class within R for dates. Once a column is of the `date` class, we can subset or order the dataset by time. `as.Date()` converts a column to a data, but here if we try the command `weather_data$date <- as.Date(weather_data$yday)`, we will receive an error saying an origin must be supplied.

The function can see that the date is in days after some starting time or origin. The name `yday` means this is the day of the year, so the origin should be the last day of the previous year. There are multiple years in our dataframe, so the origin should change for each year. This is accomplished in the function as.Date.daymetr(). To see this code and understand how it works go to the functions script in github.

```
weather_data$date <- as.Date.daymetr(weather_data)
head(weather_data$date)
```

```
## [1] "2000-01-01" "2000-01-02" "2000-01-03" "2000-01-04" "2000-01-05"
## [6] "2000-01-06"
```

Currently, there is no month variable in our dataframe. There is a package called `lubridate` that can facilitate easy transformations of dates in R. We use the command `month()` to add a variable called `month` to the dataframe. The option `label = TRUE` creates a string with the month name instead of a number.

```
weather_data$month <- lubridate::month(weather_data$date, label = TRUE)
```

Now, we need to sum the daily precipitation for each year and month combination. There is a package called `dplyr` that helps with many kinds of data manipulation. A popular task is to perform an action over a group, like taking the sum of something. To specify the grouping variables, you use `group_by()` then add the additional command `summarise()` which defines the action. For this exercise we wrote functions that use `dplyr` to make the task simpler for you `sumprec_by_monthyear()` and `avgprec_by_month()`.

2. Create a monthly precipitation for each year called `prec_month` using the `dplyr` commands.

```
by_month_year <- sumprec_by_monthyear(weather_data)
```

We now have a dataframe with the rainfall for each month of each year. We will look at the first rows.

```r
head(by_month_year)
```

```
## # A tibble: 6 x 3
## # Groups:   month [1]
##   month  year prec_month
##   <ord> <dbl>      <dbl>
## 1 Jan    2000       1.97
## 2 Jan    2001       1.18
## 3 Jan    2002       2.52
## 4 Jan    2003       1.10
## 5 Jan    2004       2.76
## 6 Jan    2005       4.06
```

Now we want to separate the 2018 data from the rest of the years before we take the average monthly precipitation. We will take the subset of `by_month_year` that is in 2018.

```r
monthprec_2018 <- subset(by_month_year, year==2018)
```

Now find the subset of `by_month_year` that is not in 2018, and we will take the average of the precipitation for each month.

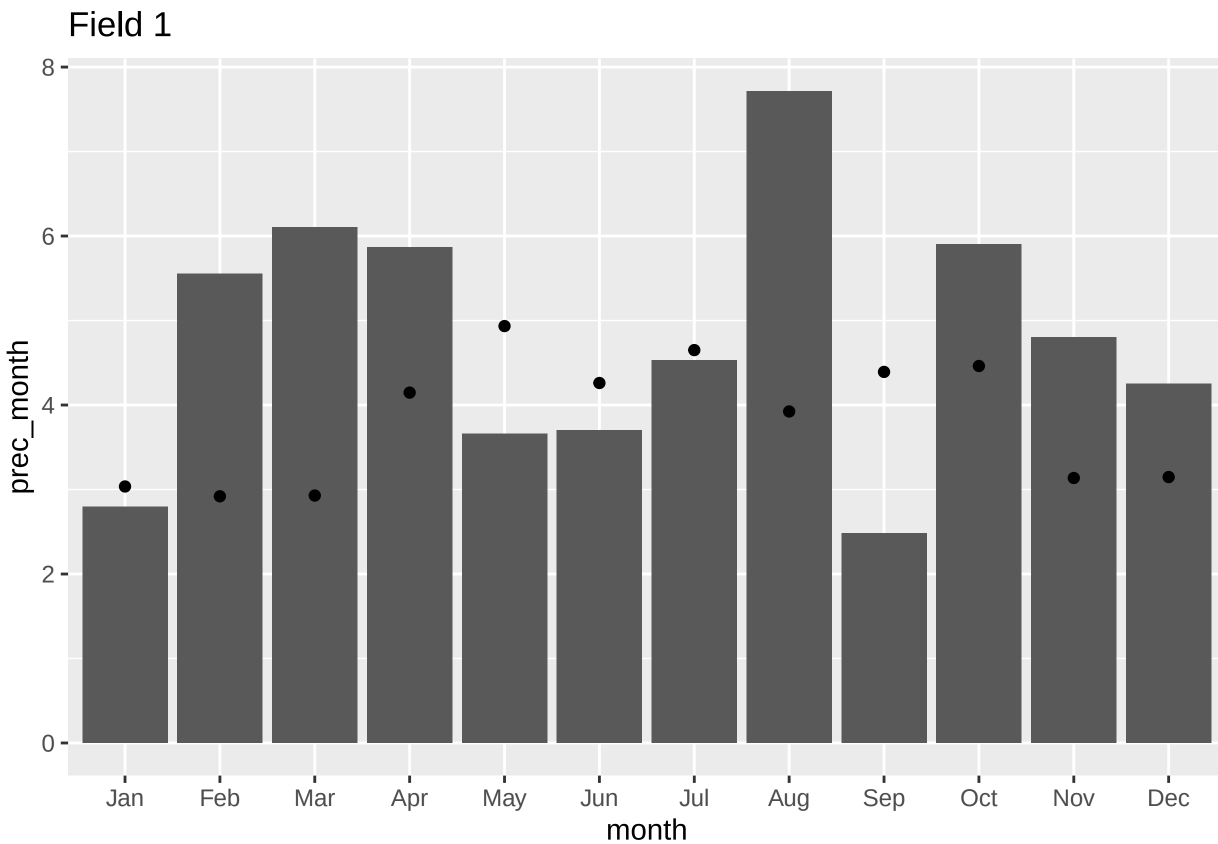
```r
monthprec_hist <- avgprec_by_month(subset(by_month_year, year != 2018))
```

We now have two separate dataframes `by_month_2018` and `by_month` with the rainfall for each month. We can use the common variable `month` to merge them into one dataframe with the average monthly rainfall and the 2018 monthly rainfall using the `merge()` function.

```r
prec_plot <- merge(monthprec_hist, monthprec_2018, by = "month")
```

We will now use `ggplot` to create a graph with a bar representing the monthly precipitation in 2018 and a point with the average rainfall from 2000 to 2017. In the function `geom_bar()` `stat = identity` creates a bar graph where the height of the bar is the value of the variable rather than the count of the observations, the common use of a bar chart.

```r
monthly_prec <- ggplot(prec_plot) +
  geom_bar(aes(x = month, y = prec_month), stat = 'identity')
monthly_prec + geom_point(aes(month, prec_avg), show.legend = TRUE) + ggtitle("Field 1")
```



The most notable feature of the weather graph is the below average rainfall in July, the most critical growing period for corn. To understand whether this affected yield on the field, we woud also need to look at historic yield. But on your field, you will know those historic average and be able to have a pretty clear idea of weather impacted the average yield in a growing season.

Another possible graph you could make with these data is on the accumulated GDD each month.