INF 550 Section 2.15 - NEON Coding Lab Part 2

Natasha Wesely

2022-09-25

2.15 NEON Coding Lab Part 2

Question 1

Use the answers that you've provided above to select a single NEON site.

JORN

Question 2

Use the answers that you've provided above to select 3 NEON data products from either the TOS, TIS or ARS (AOP) collection methods. Summarize each product with its NEON identifier, along with a summary.

1) Precipitation DP1.00006.001

The precipitation data product measures precipitation through time. It breaks out the precipitation measurements into three kinds of observations: primary precip is measured using a weighing gauge, secondary precip is measured using a tipping bucket at the very top of the tower, and ground level precip is measured using a tipping bucket as well but at the ground level.

2) H2O Concentration - DP1.00035

This data product measures the water concentration in the air around the top of the NEON tower. This is important for estimating water vapor exchange and ET (evapotranspiration).

3) CO2 Concentration - DP1.00034

The CO2 concentration data product observes the concentration of CO2 in the air around the top of the NEON tower. This is important for estimating carbon exchange between the atmosphere and ecosystem.

Question 3

Using the NEON Ulitites package or the API pull in those data along with metadata.

```
sitesOfInterest <- c("JORN")

# precip data data product ID
dpid1 <- as.character('DP1.00006.001')

# download the precip data
precipDat <- loadByProduct(dpID=dpid1,
                     site = sitesOfInterest,
                     package = "basic",
                     check.size = FALSE,
                     token=NEON_TOKEN)
```

```
## Finding available files
##   |                                                                              |
```

```
# save(precipDat, file = "precipDat.Rdata")
# load("precipDat.Rdata")

# eddy covar data product ID
dpid2 = as.character("DP4.00200.001")

# Download eddy covariance data
zipsByProduct(dpID=dpid2,
              site = sitesOfInterest,
              startdate = "2021-06",
              enddate = "2021-08",
              package = "basic",
              check.size = FALSE,
              savepath = "./",
              token=NEON_TOKEN)
```

```
## Finding available files
##   |                                                                              |
##
## Downloading files totaling approximately 245.146425 MB
```

```
## .//filesToStack00200 already exists. Download will proceed, but check for duplicate files.
## Downloading 3 files
##   |                                                                              |
## 3 files successfully downloaded to .//filesToStack00200
```

```
flux <- stackEddy(filepath = "./filesToStack00200",
                   level = "dp04")
```

```
## Extracting data
##   |                                                                              |
## Stacking data tables by month
##   |                                                                              |
## Joining data variables
##   |                                                                              |
```

```
# save(flux, file = "flux.Rdata")
# load("flux.Rdata")
```

Question 4

Organize your data into data.frames and produce summaries for each of your data:

```
# unlist all data frames
list2env(precipDat ,.GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

```
# summary of precip data (30 min)
str(THRPRE_30min)
```

```
## 'data.frame':    360192 obs. of  12 variables:
##  $ domainID          : chr  "D14" "D14" "D14" "D14" ...
##  $ siteID            : chr  "JORN" "JORN" "JORN" "JORN" ...
##  $ horizontalPosition: chr  "001" "001" "001" "001" ...
##  $ verticalPosition  : chr  "000" "000" "000" "000" ...
##  $ startDateTime     : POSIXct, format: "2017-07-13 00:00:00" "2017-07-13 00:30:00" ...
##  $ endDateTime       : POSIXct, format: "2017-07-13 00:30:00" "2017-07-13 01:00:00" ...
##  $ TFPrecipBulk      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TFPrecipExpUncert : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TFPrecipRangeQF   : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ TFPrecipSciRvwQF  : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ publicationDate   : chr  "20211210T162452Z" "20211210T162452Z" "20211210T162452Z" "20211210T16245
##  $ release           : chr  "RELEASE-2022" "RELEASE-2022" "RELEASE-2022" "RELEASE-2022" ...
```

```
# unlist all data frames
list2env(flux ,.GlobalEnv)
```

```
## <environment: R_GlobalEnv>
```

```r
# summary of eddy covar data
str(JORN)
```

```
## 'data.frame':    4416 obs. of  34 variables:
##  $ timeBgn                       : POSIXct, format: "2021-06-01 00:00:00" "2021-06-01 00:30:00" ...
##  $ timeEnd                       : POSIXct, format: "2021-06-01 00:29:59" "2021-06-01 00:59:59" ...
##  $ data.fluxCo2.nsae.flux        : num  -0.0838 -0.0691 -0.1529 -0.0945 0.0888 ...
##  $ data.fluxCo2.stor.flux        : num  0.03661 -0.038266 0.000255 -0.044372 -0.018265 ...
##  $ data.fluxCo2.turb.flux        : num  -0.1204 -0.0308 -0.1532 -0.0501 0.1071 ...
##  $ data.fluxH2o.nsae.flux        : num  -4.37 -2.14 -6.91 -7.93 8.04 ...
##  $ data.fluxH2o.stor.flux        : num  -3.36 -1.13 3.86 -3.79 1.57 ...
##  $ data.fluxH2o.turb.flux        : num  -1.01 -1 -10.78 -4.14 6.46 ...
##  $ data.fluxMome.turb.veloFric   : num  0.666 0.514 0.52 0.528 0.415 ...
##  $ data.fluxTemp.nsae.flux       : num  83.08 36.91 22.83 -2.85 -13.92 ...
##  $ data.fluxTemp.stor.flux       : num  -6.652 -0.663 -3.102 -3.025 -4.469 ...
##  $ data.fluxTemp.turb.flux       : num  89.736 37.575 25.936 0.174 -9.452 ...
##  $ data.foot.stat.angZaxsErth    : num  19.9 25 44.3 58.4 57.2 ...
##  $ data.foot.stat.distReso       : num  8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 ...
##  $ data.foot.stat.veloYaxsHorSd  : num  1.504 0.954 1.597 0.839 0.658 ...
##  $ data.foot.stat.veloZaxsHorSd  : num  0.846 0.733 0.716 0.492 0.429 ...
##  $ data.foot.stat.veloFric       : num  0.701 0.555 0.617 0.582 0.452 ...
##  $ data.foot.stat.distZaxsMeasDisp: num  8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 8.34 ...
##  $ data.foot.stat.distZaxsRgh    : num  0.0332 0.0477 0.2951 0.3618 0.5609 ...
##  $ data.foot.stat.distZaxsAbl    : num  1000 1000 1000 1000 1000 1000 1000 1000 1000 1000 ...
##  $ data.foot.stat.distXaxs90     : num  367 325 259 317 259 ...
##  $ data.foot.stat.distXaxsMax    : num  142 125 108 125 108 ...
##  $ data.foot.stat.distYaxs90     : num  16.7 16.7 33.4 33.4 33.4 ...
##  $ qfqm.fluxCo2.nsae.qfFinl      : num  1 0 0 0 0 0 0 1 1 1 ...
##  $ qfqm.fluxCo2.stor.qfFinl      : num  1 0 0 0 0 0 0 1 1 1 ...
##  $ qfqm.fluxCo2.turb.qfFinl      : num  1 0 0 0 0 0 0 0 1 1 ...
##  $ qfqm.fluxH2o.nsae.qfFinl      : num  1 0 0 1 1 0 0 1 1 1 ...
##  $ qfqm.fluxH2o.stor.qfFinl      : num  1 0 0 0 0 0 0 1 1 1 ...
##  $ qfqm.fluxH2o.turb.qfFinl      : num  1 0 0 1 1 0 0 0 1 1 ...
##  $ qfqm.fluxMome.turb.qfFinl     : num  0 0 0 0 0 0 0 0 1 1 ...
##  $ qfqm.fluxTemp.nsae.qfFinl     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ qfqm.fluxTemp.stor.qfFinl     : num  1 1 1 1 1 1 1 1 1 1 ...
##  $ qfqm.fluxTemp.turb.qfFinl     : num  0 0 0 1 0 0 0 0 1 1 ...
##  $ qfqm.foot.turb.qfFinl         : num  0 0 0 0 0 0 0 0 1 1 ...
```

Question 5

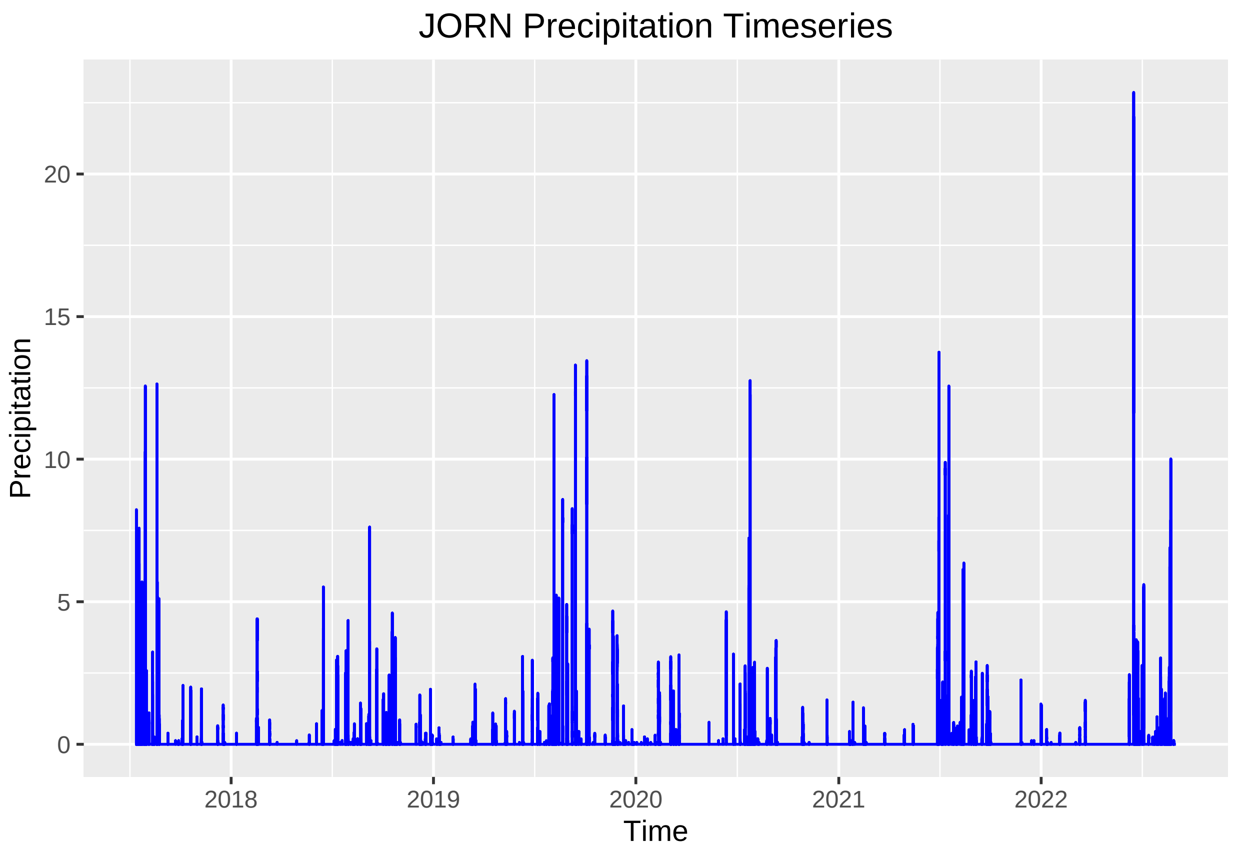
Filter and format your data based on metadata and quality flags:

```r
precip.df = THRPRE_30min %>%
  select(endDateTime, TFPrecipBulk) %>%
  # get rid of duplicate rows
  distinct() %>%
  # get rid of any rows that have an NA
  na.omit()

#flux H2O turbulent data
H2Oconc.df = JORN %>%
  select(timeEnd, data.fluxH2o.turb.flux) %>%
  # get rid of duplicate rows
  distinct() %>%
  # get rid of any rows that have an NA
  na.omit()

CO2conc.df = JORN %>%
  select(timeEnd, data.fluxCo2.turb.flux) %>%
  # get rid of duplicate rows
  distinct() %>%
  # get rid of any rows that have an NA
  na.omit()

# luckily for me, the date columns in all of these dataframes are correctly reading as date objects.
```

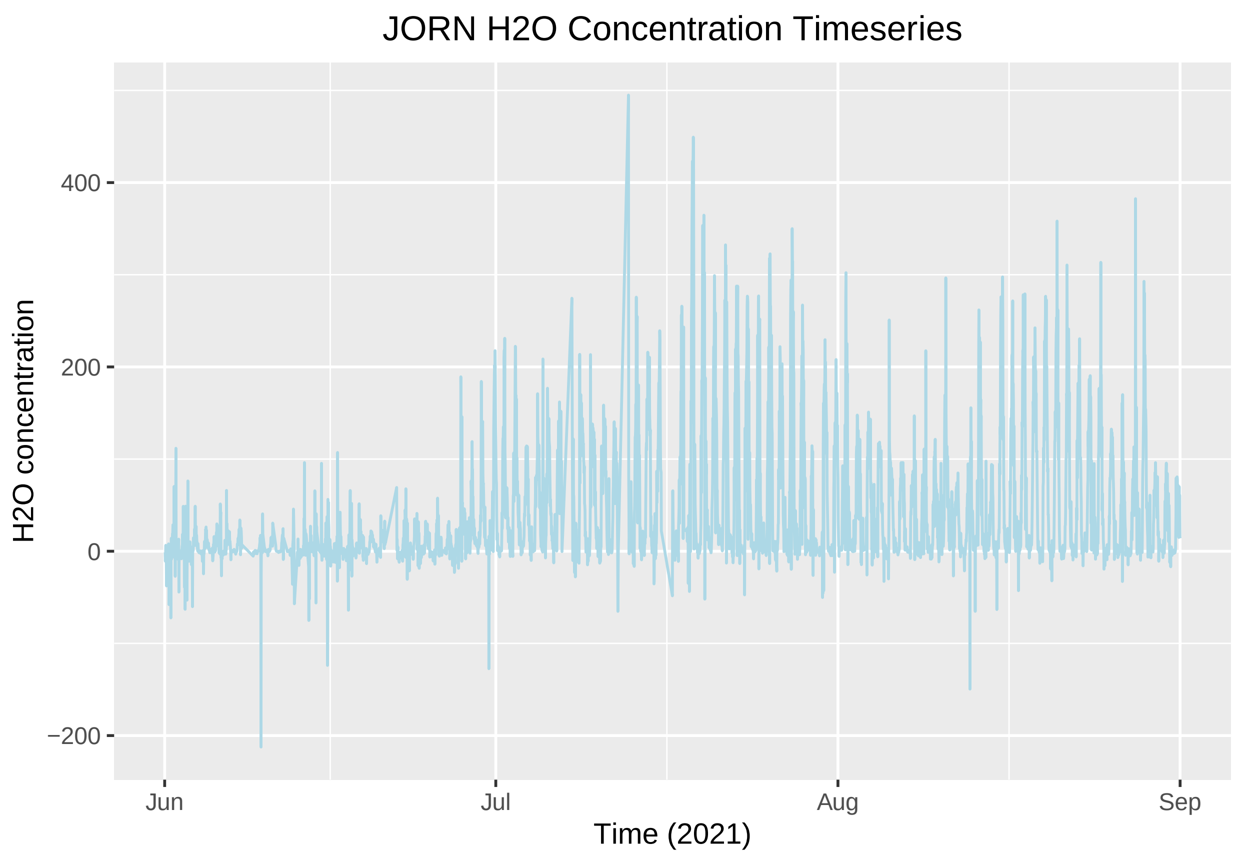
Question 6

Create minimum of 1 plot per data type (minimum of 3 plots total). These will vary based on that data that you've chosen.

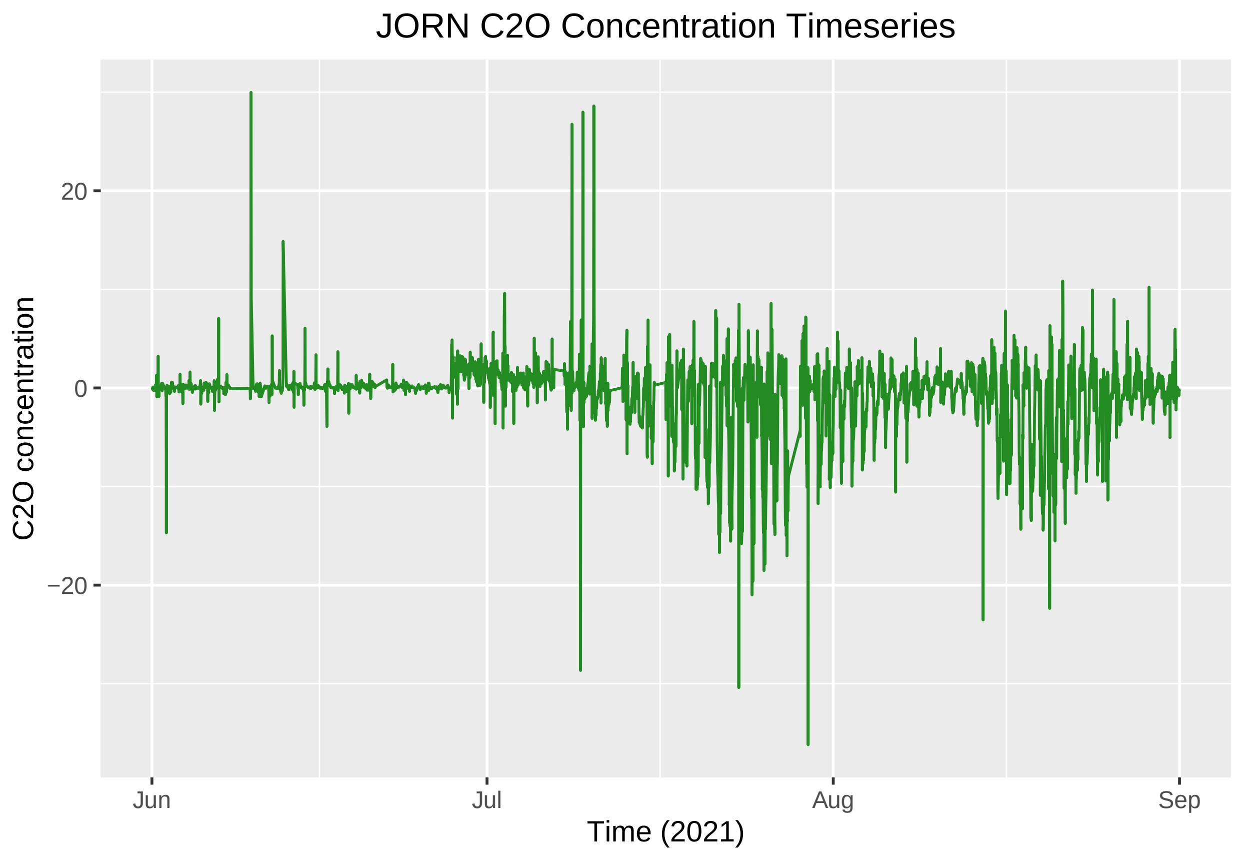
```
# precip timeseries
precip.df %>% ggplot() +
  geom_line(aes(x = endDateTime, y = TFPrecipBulk), color = "blue") +
  labs(title = "JORN Precipitation Timeseries", x = "Time", y = "Precipitation") +
  theme(plot.title = element_text(hjust = .5))
```



```
# H2O concentration timeseries
H2Oconc.df %>% ggplot() +
  geom_line(aes(x = timeEnd, y = data.fluxH2o.turb.flux), color = "lightblue") +
  labs(title = "JORN H2O Concentration Timeseries", x = "Time (2021)", y = "H2O concentration") +
  theme(plot.title = element_text(hjust = .5))
```



```
# C2O concentration timeseries
CO2conc.df %>% ggplot() +
  geom_line(aes(x = timeEnd, y = data.fluxCo2.turb.flux), color = "forestgreen") +
  labs(title = "JORN C2O Concentration Timeseries", x = "Time (2021)", y = "C2O concentration") +
  theme(plot.title = element_text(hjust = .5))
```



Question 7

What is the temporal frequency of observations in the data you decided was of interest? The precipitation data is 30 minute data. The H2O concentration data that I am using in the graph above is about daily (however this is an aggregated version of the data). The CO2 concentration data I am using above is about daily as well.

How do the data align to answer a central question?

These data align to inform me about what ecological processes were happening to what magnitude through out the last few year. This will help me inform my process-based model of the major ecosystem processes over the course of annual cycles.

What challenges did you run into when investigating these data?

There is just a lot of information that gets downloaded along with the actual data itself. I found this overwhelming and a lot to wade through to get to the actual data. I also struggled to get the eddy covariance data to download since it's a bundled data product (aka it's a huge file). But constraining the date range (instead of downloading all data ever) helped alleviate this problem. Another challenge is gaps in the data, which can easily disrupt an analysis.

How will you address these challenges and document your code? One to two paragraphs

To address the issue of overwhelming metadata, I will simply have to spend time going over each of the included metadata files. As I download and go over more NEON products, I'm sure the documentations will become more logical to me and I will be more easily able to find the information I want. To document this in my code, it would be a good idea to write out comments in my code about what each of the variables/acronyms/etc mean and where in the metadata I found that information in case I (or someone else) want to know where my information came from later on.

To address the issue of downloading huge files, I could either break up the download into multiple chunks (perhaps my time intervals) or try to download all the data all at once on a more powerful computer than my laptop. I could even use a super computer (like Monsoon) to download all the data at once for me. Whether in the future I choose to download the data in chunks or download the data all at once on a super computer, I will make clear comments in my code about why I am doing what and how I am accomplishing the overall data download needed to replicate my work.