ESPM 174A - Lab 7 & Homework Assignment 4

Daniela Rodriguez-Chavez

October 27th, 2023

Instructions

In this lab and associated homework assignment, we will advance the final project. We will get the data ready, we will specify a model to fit your own data, we will fit the model, and we will interpret the results. This lab is meant to build on Homework Assignment 2 (HW2), and get you closer to the final model or set of models that you will be using for your project.

Most of you will be fitting MAR or MARSS models. If so, answer questions 1-5 below. Please submit the completed lab on bCourses as a knitted R Markdown (html or pdf) by Friday, Oct 20th before midnight. This submission will be the 4th and last Homework Assignment (HW4).

Questions

Question 1

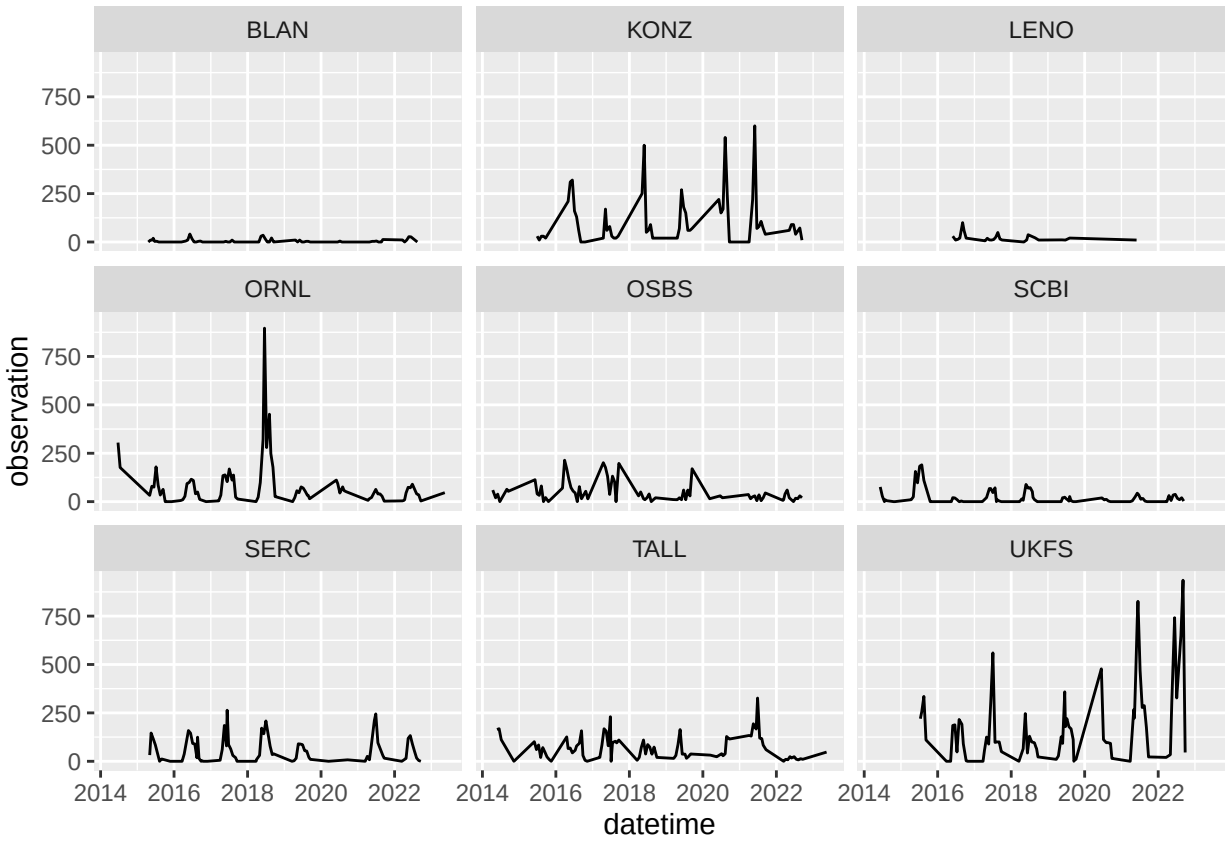
In your individual project, what is (are) your variate(s), also known as response(s)? Create the object (or recover it from HW2), and name it 'dat'. If you have many variates, you do not need to analyze the whole dataset in this lab/HW4. However, the closer the resemblance of this data set to the one you will end up analyzing, the better. E.g., if your question is at the community level, then include several species; if you would like to compare a particular physical variable across different sites, then include several sites. If you have multivariate responses, name rows appropriately so that you can keep track of each state. Do you need to further clean and/or transform these data for analysis (e.g., log-transform, z-score)? If so, do it below (and name this new object 'transformed_dat'). Remember time needs to go over columns (use tidyr's 'pivot_wider' if necessary), and you need a 'matrix' object–you can check that using the function 'class()' [1 point]

```
# packages
library(tidyverse)
library(neon4cast) # remotes::install_github("eco4cast/neon4cast")
library(neonUtilities)
library(BiocManager)
library(lubridate)
```

```
tick_data_raw <- read_csv("https://data.ecoforecast.org/neon4cast-targets/ticks/ticks-targets.csv.gz")
```

```
## Rows: 622 Columns: 5
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): site_id, variable, iso_week
## dbl  (1): observation
## date (1): datetime
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

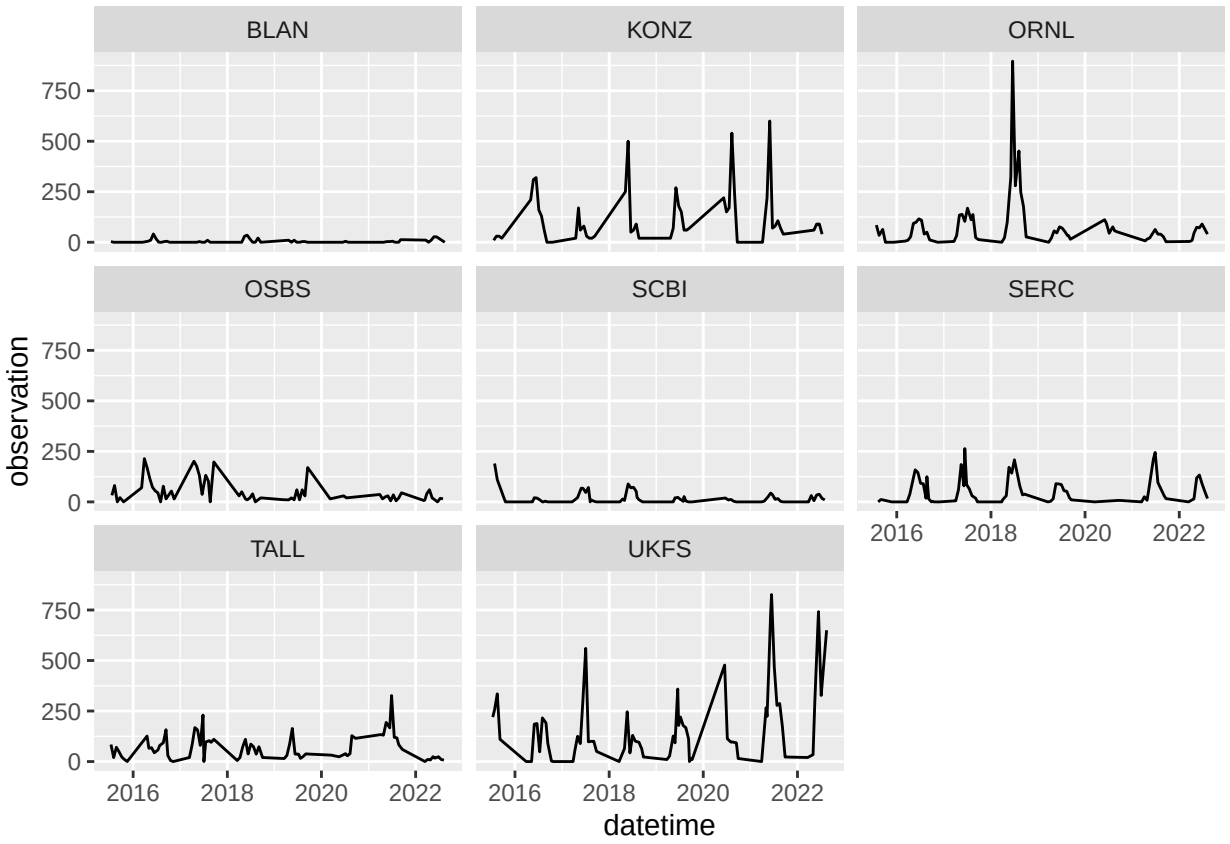
```
tick_data <- tick_data_raw # saving it
# remind ourselves of what the data looks like
tick_data |> ggplot(aes(datetime, observation)) + geom_line() + facet_wrap(~site_id)
```




```
# we want to standardize the start and end date for each site
# visually, we see that LENO might be cutting off too much data so we remove it
tick_data  <- tick_data |> filter(site_id != "LENO")
# now we check the min and max for each site
for (i in unique(tick_data$site_id)) {
  SITE_DATA <- tick_data |> filter(site_id == i)
  print(paste('For site', i, ', the starting date is', min(SITE_DATA$datetime),
              'and the last date recorded being', max(SITE_DATA$datetime)))
}
```

```
## [1] "For site BLAN , the starting date is 2015-04-20 and the last date recorded being 2022-08-15"
## [1] "For site KONZ , the starting date is 2015-06-29 and the last date recorded being 2022-09-12"
## [1] "For site ORNL , the starting date is 2014-06-23 and the last date recorded being 2023-05-15"
## [1] "For site OSBS , the starting date is 2014-04-14 and the last date recorded being 2022-09-12"
## [1] "For site SCBI , the starting date is 2014-06-09 and the last date recorded being 2022-09-12"
## [1] "For site SERC , the starting date is 2015-05-04 and the last date recorded being 2022-09-19"
## [1] "For site TALL , the starting date is 2014-05-26 and the last date recorded being 2023-05-15"
## [1] "For site UKFS , the starting date is 2015-07-13 and the last date recorded being 2022-09-26"
```

```
# standardizing the timeframe among the relevant sites
tick_data <- tick_data |> filter(datetime >= "2015-07-13" & datetime <= "2022-08-15") |>
  select(-variable, -iso_week)
tick_data |>  ggplot(aes(datetime, observation)) + geom_line() + facet_wrap(~site_id)
```




```
# create a sequence of weeks to have standardized weeks
datetime <- seq(min(tick_data$datetime), max(tick_data$datetime), "week")
# create a df where this is a column
EXPANDED_DATA <- data.frame(datetime)

# creating a function that will create a df of each site for all the weeks
# then also grouping by month
fill_data <- function(SITE_NAME) {
  # getting the general df
  SITE_DATA <- left_join(EXPANDED_DATA, subset(tick_data, site_id == SITE_NAME))
  # grouping by just month and year
  SITE_DATA['datetime'] <- format(SITE_DATA$datetime, format = "%Y-%m")
  # getting the monthly average
  SITE_DATA <-SITE_DATA %>% group_by(datetime) %>%
    summarise(obs_month_avg = mean(observation, na.rm=TRUE)) %>% ungroup()
  # recording the site name
  SITE_DATA["site_id"] <- SITE_NAME
  # returning the data
  SITE_DATA
}

# all the different sites
BLAN <- fill_data('BLAN')
```

```
## Joining with `by = join_by(datetime)`
```

```
KONZ <- fill_data('KONZ')
```

```
## Joining with `by = join_by(datetime)`
```

```
ORNL <- fill_data('ORNL')
```

```
## Joining with `by = join_by(datetime)`
```

```
OSBS <- fill_data('OSBS')
```

```
## Joining with `by = join_by(datetime)`
```

```
SCBI <- fill_data('SCBI')
```

```
## Joining with `by = join_by(datetime)`
```

```
SERC <- fill_data('SERC')
```

```
## Joining with `by = join_by(datetime)`
```

```
TALL <- fill_data('TALL')
```

```
## Joining with `by = join_by(datetime)`
```

```
UKFS <- fill_data('UKFS')
```

```
## Joining with `by = join_by(datetime)`
```

```
# joining all the sites together into one big dataframe
dat <- bind_rows(list(BLAN, KONZ, ORNL, OSBS, SCBI, SERC, TALL, UKFS))
# reorder by time
dat$datetime <- parse_date_time(dat$datetime, "Ym")
dat$datetime <- as.Date(dat$datetime, format = "%Y-%m")
dat <- dat[order(dat$datetime), ]; dat
```

```
## # A tibble: 688 x 3
##    datetime   obs_month_avg site_id
##    <date>             <dbl> <chr>
##  1 2015-07-01          3.66 BLAN
##  2 2015-07-01         10    KONZ
##  3 2015-07-01         85    ORNL
##  4 2015-07-01         32.4  OSBS
##  5 2015-07-01        190    SCBI
##  6 2015-07-01        NaN    SERC
##  7 2015-07-01         84.3  TALL
##  8 2015-07-01        238.   UKFS
##  9 2015-08-01          0    BLAN
## 10 2015-08-01         30    KONZ
## # i 678 more rows
```

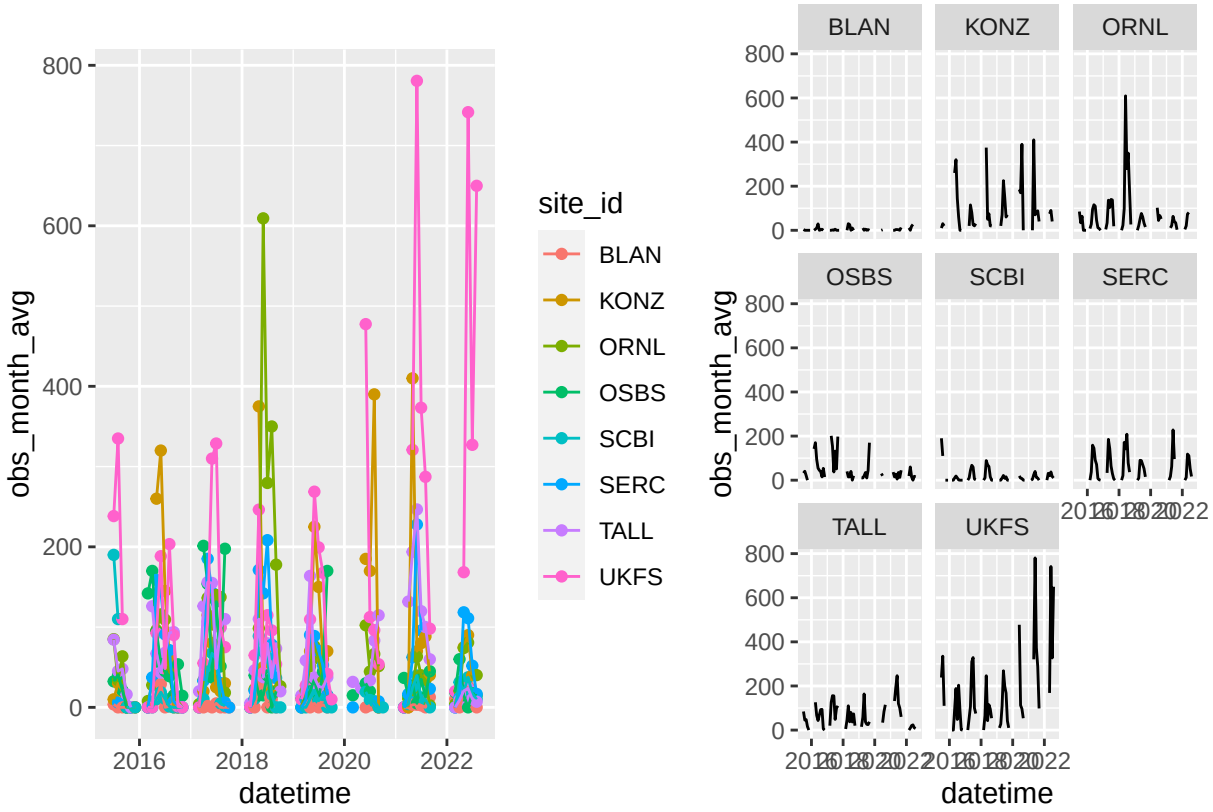
```
library(patchwork)
p1 <- ggplot(data = dat, aes(x = datetime, y = obs_month_avg, group = site_id)) +
  geom_point(aes(color = site_id)) + geom_line(aes(color=site_id))

p2 <- dat |> ggplot(aes(datetime, obs_month_avg)) + geom_line() + facet_wrap(~site_id)

p1 + p2
```

```
## Warning: Removed 298 rows containing missing values (`geom_point()`).
```

```
## Warning: Removed 2 rows containing missing values (`geom_line()`).
```




```
# time needs to go over columns (use tidyr's 'pivot_wider' if necessary)
dat <- pivot_wider(dat, names_from = datetime, values_from = obs_month_avg)
# turning it into a matrix
dat <- as.matrix(dat); dat
```

```
##      site_id 2015-07-01   2015-08-01   2015-09-01  2015-10-01 2015-11-01
## [1,] "BLAN"  "  3.661327" "  0.000000" "  0.00000" " 0.00000" "  0"
## [2,] "KONZ"  " 10.000000" " 30.000000" " 20.00000" NA         NA
## [3,] "ORNL"  " 85.000000" " 34.000000" " 63.60248" " 0.00000" "  0"
## [4,] "OSBS"  " 32.432432" " 40.506329" " 21.13208" " 0.00000" NA
## [5,] "SCBI"  "190.000000" "110.000000" NA          " 0.00000" "  0"
## [6,] "SERC"  NA           "  5.962733" NA          NA         "  0"
## [7,] "TALL"  " 84.313725" " 45.145457" " 47.94007" "16.18827" "  0"
## [8,] "UKFS"  "238.316832" "335.000000" "110.00000" NA         NA
##      2015-12-01 2016-01-01 2016-02-01 2016-03-01   2016-04-01   2016-05-01
## [1,] "  0"      NA         NA         "  0.000000" "  4.819277" "  9.876543"
## [2,] NA         NA         NA         NA           NA           "260.000000"
## [3,] "  0"      NA         NA         "  7.895174" " 27.722772" " 96.322169"
## [4,] NA         NA         NA         "141.990881" "170.000000" " 94.369679"
## [5,] "  0"      NA         NA         "  0.000000" "  0.000000" " 10.272873"
## [6,] NA         NA         NA         "  0.000000" " 37.288136" "158.730159"
## [7,] NA         NA         NA         NA           "125.984252" " 67.034349"
## [8,] NA         NA         NA         "  0.000000" "  0.000000" " 92.500000"
##      2016-06-01  2016-07-01  2016-08-01   2016-09-01  2016-10-01  2016-11-01
## [1,] " 29.04161" "  0.00000" "  1.659751" " 4.863222" " 0.000000" NA
## [2,] "320.00000" "145.00000" " 60.000000" " 0.000000" " 0.000000" NA
## [3,] "115.82213" "109.38776" " 45.216346" "11.463664" " 7.738815" " 0.00000"
```

```
## [4,] " 54.65839" " 45.00000" " 38.787879" "15.000000" "53.658537" "14.41441"
## [5,] " 18.91892" " 10.68447" "  1.139601" " 0.000000" " 0.000000" " 0.00000"
## [6,] "145.26316" " 90.95553" " 71.327913" " 7.397622" " 1.663202" " 0.00000"
## [7,] " 43.33333" " 68.00053" " 92.903226" "93.844340" " 5.000000" " 0.00000"
## [8,] "188.38710" " 48.17204" "203.451351" "90.000000" " 1.250000" " 0.00000"
##      2016-12-01 2017-01-01 2017-02-01 2017-03-01 2017-04-01  2017-05-01
## [1,] NA         NA         NA         "  0"      "  0.00000" "  1.702128"
## [2,] NA         NA         NA         NA         " 20.00000" "115.000000"
## [3,] NA         NA         NA         "  4"      " 33.66337" "136.380260"
## [4,] NA         NA         NA         NA         "201.22699" "153.869169"
## [5,] NA         NA         NA         "  0"      " 10.50903" " 44.472177"
## [6,] NA         NA         NA         NA         " 33.01644" "185.142857"
## [7,] NA         NA         NA         " 20"      "126.10526" "156.000000"
## [8,] NA         NA         NA         "  0"      " 55.00000" "106.944444"
##      2017-06-01  2017-07-01   2017-08-01   2017-09-01   2017-10-01 2017-11-01
## [1,] "  0.00000" "  4.958678" "  0.000000" "  0.000000" "  0"      NA
## [2,] " 80.00000" " 25.000000" " 20.000000" " 30.000000" NA         NA
## [3,] "103.22581" "139.986355" "137.331954" " 18.298842" NA         NA
## [4,] " 37.20930" "131.764706" " 51.063830" "197.647059" NA         NA
## [5,] " 67.55556" " 59.106529" "  4.968944" "  1.663202" NA         NA
## [6,] "143.09529" " 64.165589" " 30.311231" "  6.161746" "  0"      NA
## [7,] "155.00000" " 48.333333" " 99.166667" "110.000000" NA         NA
## [8,] "310.00000" "328.750000" "100.000000" " 75.000000" NA         NA
##      2017-12-01 2018-01-01 2018-02-01 2018-03-01 2018-04-01 2018-05-01
## [1,] NA         NA         NA         NA         " 0.00000" " 30.00000"
## [2,] NA         NA         NA         NA         NA         "375.00000"
## [3,] NA         NA         NA         "  0"      "22.74112" " 98.56263"
## [4,] NA         NA         NA         NA         "39.90683" " 15.00000"
## [5,] NA         NA         NA         "  0"      "10.64395" " 88.88889"
## [6,] NA         NA         NA         "  0"      "20.64777" "171.21019"
## [7,] NA         NA         NA         "  5"      "46.28866" "110.00000"
## [8,] NA         NA         NA         "  0"      "64.89860" "246.33431"
##      2018-06-01  2018-07-01  2018-08-01  2018-09-01  2018-10-01 2018-11-01
## [1,] " 25.83333" "  0.00000" " 10.38961" "  0.00000" NA         NA
## [2,] " 50.00000" " 75.00000" " 20.00000" " 20.00000" NA         NA
## [3,] "609.33056" "279.59596" "349.98544" "177.77778" "26.39175" NA
## [4,] " 14.96894" " 39.75155" "  0.00000" " 20.00000" NA         NA
## [5,] " 69.67742" " 62.12181" " 16.14414" "  0.00000" " 0.00000" NA
## [6,] "142.08494" "208.31025" " 78.09762" " 36.55792" NA         NA
## [7,] " 37.50000" " 79.82019" " 36.66667" " 73.33333" "20.00000" NA
## [8,] " 42.66667" "114.84395" " 95.85253" " 53.78205" NA         NA
##      2018-12-01 2019-01-01 2019-02-01 2019-03-01 2019-04-01  2019-05-01
## [1,] NA         NA         NA         NA         "10.191083" "  0.00000"
## [2,] NA         NA         NA         NA         "20.000000" " 70.00000"
## [3,] NA         NA         NA         " 0.00000" "21.281741" " 51.21428"
## [4,] NA         NA         NA         "10.81081" "10.000000" " 20.00000"
## [5,] NA         NA         NA         " 0.00000" " 0.000000" " 10.19108"
## [6,] NA         NA         NA         " 0.00000" " 9.064068" " 90.22556"
## [7,] NA         NA         NA         "14.90683" "58.457875" "163.77953"
## [8,] NA         NA         NA         "10.00000" "27.500000" "109.56457"
##      2019-06-01  2019-07-01   2019-08-01   2019-09-01  2019-10-01 2019-11-01
## [1,] "  5.00000" "  0.000000" "  3.330301" "  0.00000" NA         NA
## [2,] "225.00000" "150.000000" " 60.000000" " 70.00000" NA         NA
## [3,] " 76.26943" " 61.209048" " 32.601816" " 14.83007" NA         NA
```

```
## [4,] " 35.00000" " 10.000000" " 44.723926" "170.00000" NA         NA
## [5,] " 21.44343" "  8.257078" " 15.423966" "  0.00000" "  0"      NA
## [6,] " 88.88889" " 70.446317" " 52.658228" " 15.12658" NA         NA
## [7,] " 37.05263" " 26.075269" " 24.774194" " 37.53280" NA         NA
## [8,] "268.75242" "199.371002" "167.619048" " 41.77778" " 10"      NA
##      2019-12-01 2020-01-01 2020-02-01 2020-03-01 2020-04-01 2020-05-01
## [1,] NA         NA         NA         NA         NA         NA
## [2,] NA         NA         NA         NA         NA         NA
## [3,] NA         NA         NA         NA         NA         NA
## [4,] NA         NA         NA         "15.00000" NA         NA
## [5,] NA         NA         NA         NA         NA         NA
## [6,] NA         NA         NA         " 0.00000" NA         NA
## [7,] NA         NA         NA         "31.71806" NA         "23.41463"
## [8,] NA         NA         NA         NA         NA         NA
##      2020-06-01  2020-07-01   2020-08-01   2020-09-01  2020-10-01 2020-11-01
## [1,] "  0.00000" "  2.424242" NA           "  0.00000" NA         NA
## [2,] "185.00000" "170.000000" "390.000000" "  0.00000" NA         NA
## [3,] "102.32225" " 44.582043" " 66.487722" " 51.34576" NA         NA
## [4,] " 30.00000" " 20.000000" NA           NA          NA         NA
## [5,] " 19.32367" "  9.155937" "  7.688853" "  0.00000" "  0"      NA
## [6,] NA          NA           NA           "  7.50000" NA         NA
## [7,] NA          " 33.728673" " 83.340659" "114.78261" NA         NA
## [8,] "477.50000" "112.500000" " 96.253968" " 53.75000" NA         NA
##      2020-12-01 2021-01-01 2021-02-01 2021-03-01 2021-04-01   2021-05-01
## [1,] NA         NA         NA         " 0.00000" "  0.000000" "  3.305785"
## [2,] NA         NA         NA         NA         "  0.000000" "410.000000"
## [3,] NA         NA         NA         NA         " 11.002310" " 21.421215"
## [4,] NA         NA         NA         "36.66667" " 15.000000" " 27.500000"
## [5,] NA         NA         NA         " 0.00000" "  4.030227" " 25.035562"
## [6,] NA         NA         NA         " 0.00000" " 16.681992" " 62.402496"
## [7,] NA         NA         NA         NA         "131.806616" "193.710692"
## [8,] NA         NA         NA         " 0.00000" NA           "320.833333"
##      2021-06-01   2021-07-01   2021-08-01  2021-09-01  2021-10-01 2021-11-01
## [1,] "  3.238866" "  2.507837" "  0.00000" "12.749004" NA         NA
## [2,] " 70.000000" " 80.000000" " 88.33333" "40.000000" NA         NA
## [3,] " 63.201663" " 39.958377" " 34.09590" " 2.503912" NA         NA
## [4,] "  5.000000" " 35.000000" " 12.50000" "45.000000" NA         NA
## [5,] " 39.698491" " 12.379110" "  9.54236" " 0.000000" NA         NA
## [6,] "227.882565" " 95.760599" NA          "23.153551" NA         NA
## [7,] "246.598639" "119.800333" "100.08113" "59.911894" NA         NA
## [8,] "780.634328" "373.333333" "287.24409" "98.071705" NA         NA
##      2021-12-01 2022-01-01 2022-02-01 2022-03-01  2022-04-01  2022-05-01
## [1,] NA         NA         NA         "10.596026" " 0.000000" " 18.87247"
## [2,] NA         NA         NA         NA          NA          " 75.00000"
## [3,] NA         NA         NA         " 3.975155" "20.493315" " 73.94958"
## [4,] NA         NA         NA         "18.852459" "60.000000" " 15.00000"
## [5,] NA         NA         NA         " 0.000000" "31.372549" " 20.15427"
## [6,] NA         NA         NA         " 0.000000" "12.372747" "118.44417"
## [7,] NA         NA         NA         " 0.000000" " 9.047133" " 20.53558"
## [8,] NA         NA         NA         "20.000000" NA          "168.40589"
##      2022-06-01  2022-07-01  2022-08-01
## [1,] " 26.39175" NA          "  0.000000"
## [2,] " 90.00000" " 40.00000" NA
## [3,] " 80.68465" NA          " 40.209424"
```

```
## [4,] "  0.00000" " 17.77778" " 15.000000"
## [5,] " 37.47748" " 17.11230" " 10.000000"
## [6,] "111.05087" " 51.90409" " 16.701461"
## [7,] " 23.67865" " 10.38961" "  7.048458"
## [8,] "741.58730" "327.09984" "649.846154"
```

Answer: My variate is the average amount of ticks that were recorded in a month. I have 8 sites in total, though due to the constraints on the covariates, they will go down to three for this assignment.

Question 2

What is (are) your covariate(s), aka driver(s), if any? Z-score them and make sure they have no missing data (MARSS does not allow NA's in covariate data). You can name them 'trasnformed_covar'. Remember time needs to go over columns (use tidyr's 'pivot_wider' if necessary), and you need a 'matrix' object–you can check that using the function 'class()' [1 point]

```
# getting my covars
# just looking at air temperature
neon <- arrow::s3_bucket("neon4cast-targets/neon",
                  endpoint_override = "data.ecoforecast.org",
                  anonymous = TRUE)
temp <- arrow::open_dataset(neon$path("wss_daily_temp-basic-DP4.00001.001"))
temp <- temp |> dplyr::filter(siteID %in% dat[,1]) |>
  dplyr::collect()
# cleaning up the data
temp <- temp |> select(date, wssTempTripleMean, siteID) |>
  rename(site_id = siteID) |>
  rename(mean_temp = wssTempTripleMean)
temp
```

```
## # A tibble: 27,746 x 3
##    date       mean_temp site_id
##    <date>         <dbl> <chr>
##  1 2022-06-01      17.3 KONZ
##  2 2022-06-02      16.7 KONZ
##  3 2022-06-03      19.8 KONZ
##  4 2022-06-04      18.7 KONZ
##  5 2022-06-05      20.4 KONZ
##  6 2022-06-06      21.9 KONZ
##  7 2022-06-07      22.7 KONZ
##  8 2022-06-08      20.6 KONZ
##  9 2022-06-09      20.6 KONZ
## 10 2022-06-10      21.7 KONZ
## # i 27,736 more rows
```

```
# getting the dates because of NaNs we have to shorten the dates
temp <- temp |> filter(between(date, as.Date('2016-01-01'), as.Date('2016-12-31')))  |>
  rename(datetime = date)
# averaging it by month for each site
temp['datetime'] <- format(temp$datetime, format = "%Y-%m")
# getting the monthly average
temp <- temp %>% group_by(datetime, site_id) %>%
  summarise(temp_month_avg = mean(mean_temp, na.rm=TRUE)) %>% ungroup()
```

```
## `summarise()` has grouped output by 'datetime'. You can override using the
## `.groups` argument.
```

```r
# getting rid of the sites that have NaNs
temp |> filter(is.na(temp_month_avg))
```

```
## # A tibble: 6 x 3
##   datetime site_id temp_month_avg
##   <chr>    <chr>            <dbl>
## 1 2016-04  KONZ               NaN
## 2 2016-05  KONZ               NaN
## 3 2016-06  KONZ               NaN
## 4 2016-07  KONZ               NaN
## 5 2016-11  SCBI               NaN
## 6 2016-12  SCBI               NaN
```

```r
temp <- temp |> filter(site_id != "SCBI") |>
  filter(site_id != 'KONZ')
temp
```

```
## # A tibble: 36 x 3
##    datetime site_id temp_month_avg
##    <chr>    <chr>            <dbl>
##  1 2016-01  ORNL              3.63
##  2 2016-01  OSBS             10.5
##  3 2016-01  TALL              5.38
##  4 2016-02  ORNL              8.73
##  5 2016-02  OSBS             22.3
##  6 2016-02  TALL              9.66
##  7 2016-03  ORNL             13.3
##  8 2016-03  OSBS             21.4
##  9 2016-03  TALL             15.5
## 10 2016-04  ORNL             14.0
## # i 26 more rows
```

```r
# z-scoring it
temp_ORNL <- temp |> filter(site_id == "ORNL")
temp_OSBS <- temp |> filter(site_id == "OSBS")
temp_TALL <- temp |> filter(site_id == "TALL")

temp_ORNL$zscore_month_avg <- (temp_ORNL$temp_month_avg -
                                 mean(temp_ORNL$temp_month_avg))/sd(temp_ORNL$temp_month_avg)
temp_OSBS$zscore_month_avg <- (temp_OSBS$temp_month_avg -
                                 mean(temp_OSBS$temp_month_avg))/sd(temp_OSBS$temp_month_avg)
temp_TALL$zscore_month_avg <- (temp_TALL$temp_month_avg -
                                 mean(temp_TALL$temp_month_avg))/sd(temp_TALL$temp_month_avg)

# joining all the sites together into one big dataframe
covars <- bind_rows(list(temp_ORNL, temp_OSBS, temp_TALL))
covars <- covars |> select(-temp_month_avg)
covars
```

```
## # A tibble: 36 x 3
##    datetime site_id zscore_month_avg
##    <chr>    <chr>              <dbl>
##  1 2016-01  ORNL              -1.60
##  2 2016-02  ORNL              -0.964
##  3 2016-03  ORNL              -0.398
```

```
##  4 2016-04  ORNL              -0.309
##  5 2016-05  ORNL               0.545
##  6 2016-06  ORNL               0.997
##  7 2016-07  ORNL               1.16
##  8 2016-08  ORNL               1.17
##  9 2016-09  ORNL               0.968
## 10 2016-10  ORNL               0.321
## # i 26 more rows
```

```
# time needs to go over columns (use tidyr's 'pivot_wider' if necessary)
transformed_covar <- pivot_wider(covars, names_from = datetime, values_from = zscore_month_avg)
# for the purposes of this homework, just seeing if I only have an average across all site temperatures
transformed_covar <- transformed_covar |> select(-site_id) |> colMeans()
transformed_covar <- as.data.frame(transformed_covar)
# turning it into a matrix
vec <- transformed_covar$transformed_covar
transformed_covar <- matrix(vec,1,12)
transformed_covar
```

```
##         [,1]       [,2]       [,3]       [,4]      [,5]      [,6]     [,7]
## [1,] -1.8689 -0.7083703 -0.3259188 -0.1801618 0.4546908 0.9970689 1.147851
##         [,8]      [,9]     [,10]      [,11]     [,12]
## [1,] 1.07565 0.9066464 0.2624277 -0.5223358 -1.238648
```

Answer: My covariates are just temperature. I need to figure out how to tie different temperature time series to different sites, but for right now I just averaged across the three sites. I only have three sites and I significantly shortened the time series due to NaNs in the temperature data. Looking for other weather data will be important moving forward so I can have a longer time series.

Question 3

Is each observation supposed to be modeled as a different state, or do you have 'replicate' observations, i.e. more than one observation being funneled into a state (via the Z matrix)? What are the dimensions of your Y's (observations x time steps) and X's (states x time steps)? Build the Z matrix you need, or specify it using a shortcut (e.g., Z = "identity"). [1 point]

```
# your code here
# because we only have three regions, we use a unique Z matrix
Z = factor(c("site1", "site2", "site3"))
```

Answer: Because I only have three sites that I want to compare between, with one observation per time step in each site, my Z matrix will consist of each site being a separate state.

Question 4

Specify the rest of your MAR/MARSS parameters using a model list, like we have been doing so far: R (based on the number of observations), and U, B, C, Q (based on the number of states). If you would like to fit MAR instead of MARSS, then set R to "zero". Remember what we learned over the past few weeks, e.g. if you want to focus on the B matrix (e.g. species interactions) then it is best to fix U (long-term trend) to zero, which you can do after demeaning the variate data. If you are building custom matrices, remember that R and Q need to be symmetrical, B does not need to be. Also, R, Q, and B need to be square; all other matrices may be rectangular. If you have covariate data, assign it here as well to the model list ("c"). If you plan on comparing models, it is best practice to start with a simple model structure (e.g., Q = "diagonal and equal" instead of "unconstrained"), and make it progressively more complex. [1 point]

```
# when I ran it with it being "equal", i got that Q went to 0,
# so I am now making the model into a MAR model
R = "zero"
```

```
# because we are assuming that each site has its own independent processes,
# there is no covariance in the process error
Q = "diagonal and unequal"

# there is no density dependence in this situation
B = "identity"

# because we did not transform our count data,
# we assume that there is a trend happening in U that
# we did not account for
U = "unequal"

# covariate data
# we set C to unequal because we want to see if it affects each site differently
C = "unequal"
c = transformed_covar
```

Question 5

Fit the model. If you get errors or warnings, first check that the model specification is right (e.g., number of dimensions of each matrix). If dimensions are right, but the model does not converge, increase number of iterations using the agument 'maxit'. If it still does not converge, check if the model you are fitting does not make sense given the data (e.g. perhaps you are fitting a stationary model to a non-stationary process), and re-specify the model accordingly, in step 5. If you are fitting MARSS and one of the variances goes to zero (Q or R), try fitting a MAR model instead. If errors persist, check the MARSS User Guide: https://cran.r-project.org/web/packages/MARSS/vignettes/UserGuide.pdf ("Appendix A - Warnings and errors", page 309). Once it does work: bootstrap the model(s). What do you obtain? Is it what you expected? What are the next steps to complete analyses for your final project? [1 point]

```
# getting from 2016-01 to 2017-06
dat <- dat[,-(2:7)]
dat <- dat[,1:13]
# getting only the sites that match the transformed covars
dat <- as.tibble(dat)
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```
dat <- dat |> filter(site_id %in% c("ORNL", "OSBS", "TALL")) |>
  select(-site_id)
# turning into numeric vals
dat <- as.matrix(dat)
dat <- apply(dat, 2, as.numeric)
dat
```

```
##      2016-01-01 2016-02-01 2016-03-01 2016-04-01 2016-05-01 2016-06-01
## [1,]         NA         NA   7.895174   27.72277   96.32217  115.82213
## [2,]         NA         NA 141.990881  170.00000   94.36968   54.65839
## [3,]         NA         NA         NA  125.98425   67.03435   43.33333
##      2016-07-01 2016-08-01 2016-09-01 2016-10-01 2016-11-01 2016-12-01
## [1,]  109.38776   45.21635   11.46366   7.738815    0.00000         NA
## [2,]   45.00000   38.78788   15.00000  53.658537   14.41441         NA
```

```
## [3,]   68.00053   92.90323   93.84434   5.000000    0.00000         NA
```

```
library(MARSS)
mod0 = list() # This will be the list of parameters, each is a matrix

# X equation (process model)
mod0$B = B
mod0$U = U
mod0$Q = Q
mod0$C = C
mod0$c = c

# Y equation (observation model)
mod0$Z = Z
mod0$R = R

# Fit the MARSS model
mod0.fit = MARSS(dat, model=mod0)
```

```
## Success! abstol and log-log tests passed at 55 iterations.
## Alert: conv.test.slope.tol is 0.5.
## Test with smaller values (<0.1) to ensure convergence.
##
## MARSS fit is
## Estimation method: kem
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 55 iterations.
## Log-likelihood: -129.7006
## AIC: 283.4012   AICc: 307.4012
##
##                 Estimate
## U.site1             8.97
## U.site2           -11.77
## U.site3           -33.74
## Q.(site1,site1)  1091.30
## Q.(site2,site2)  1067.46
## Q.(site3,site3)  1141.72
## x0.site1          -74.85
## x0.site2          153.84
## x0.site3          338.60
## C.site1           -19.28
## C.site2            -8.03
## C.site3            26.48
## Initial states (x0) defined at t=0
##
## Standard errors have not been calculated.
## Use MARSSparamCIs to compute CIs and bias estimates.
```

```
# Bootstrap best model
myMARSSobject<-MARSSparamCIs(mod0.fit)
myMARSSobject
```

```
##
## MARSS fit is
## Estimation method: kem
```

```
## Convergence test: conv.test.slope.tol = 0.5, abstol = 0.001
## Estimation converged in 55 iterations.
## Log-likelihood: -129.7006
## AIC: 283.4012   AICc: 307.4012
##
##                   ML.Est Std.Err low.CI  up.CI
## U.site1             8.97    20.0  -58.5   19.9
## U.site2           -11.77    19.8  -46.8   30.7
## U.site3           -33.74    22.9  -18.5   71.4
## Q.(site1,site1) 1091.30   514.4   83.0 2099.6
## Q.(site2,site2) 1067.46   503.2   81.2 2053.7
## Q.(site3,site3) 1141.72   570.9   22.9 2260.6
## x0.site1          -74.85   111.6 -293.5  143.8
## x0.site2          153.84   110.3  -62.4  370.1
## x0.site3          338.60   153.0   38.8  638.4
## C.site1           -19.28    20.0  -58.5   19.9
## C.site2            -8.03    19.8  -46.8   30.7
## C.site3            26.48    22.9  -18.5   71.4
## Initial states (x0) defined at t=0
##
## CIs calculated at alpha = 0.05 via method=hessian
```

Answer: After bootstrapping the model, we see that both the trend and the influence of temperature on each of the sites is actually not that significant, since 0 is present within all of the confidence intervals. It is a little bit about what I expected because since I only have a year of data this time around, it doesn't make sense that there would be enough time to form a trend.

I definitely have a lot to work on, since I want to have 7 sites and a much longer time span. Things to focus on are figuring out how to tie specific covariates to specific sites, as well as get weather data for each site that is the length that I need it to be. I might also add humidity/precipitation as a second covariate.