W271 Assignment 6

```
rm(list = ls())
library(tidyverse)
library(patchwork)

library(lubridate)

library(tsibble)
library(feasts)
install.packages('forecast')
library(forecast)

library(sandwich)
library(lmtest)

library(nycflights13)
install.packages('blsR')
library(blsR)
```

```
theme_set(theme_minimal())
```

Plot Flights and Weather Data

To start with this homework, you will be using the same data that Jeffrey uses in the lecture – US flights data. The data comes from the packages `nycflights13`.

Question Goals

Our goal with the tasks in this question are to try to familiarize yourself with some of the key programming concepts related to time series data – setting time indexes and key variables, grouping and indexing on those variables, and producing descriptive plots of data that is stored in a time series form.

Question 1 - Flights to nice places

In the package declarations, we have loaded the `nycflights13` package. This provides three objects that we are going to use:

1. `flights`;
2. `airports`; and,
3. `weather`.

You can investigate these objects more by issuing a `?` before them, to access their documentation.

(1 point) Create Data

As stored, both `flights` and `weather` are stored a "plain" data frames. To begin, cast the `flights` dataset into a time series dataset, a `tsibble`.

- Use the combination of year, month, day, hour, and minute to produce the time index. Call this newly mutated variable `time_index`. There is very good handling of dates inside of the `lubridate` package. There is a nice one-page cheetsheet that Rstudio makes available. For this task you might be looking for `lubridate::make_datetime`.
- Although it may not generally be true, for this work, also assume that you can uniquely identify a flight by the carrier and the flight number, so you can use these two pieces of information to define the `key`. We need to define a key because in some cases there are more than one flight that leave at the same time – this is because the granularity of our time measure is at the minute and it is possible for two planes to leave within the same minute.

```
head(flights)
```

```
## # A tibble: 6 x 19
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 11 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

```
flights <- flights %>% mutate(time_index = lubridate::make_datetime(year=year, month=month, day=day, hou

flights_tsibble <- flights %>% as_tsibble(index=time_index, key=c(carrier, flight))
head(flights_tsibble)
```

```
## # A tsibble: 6 x 20 [1m] <UTC>
## # Key:       carrier, flight [1]
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013    11     3     1531           1540        -9     1653           1725
## 2  2013    11     4     1539           1540        -1     1712           1725
## 3  2013    11     5     1548           1540         8     1708           1725
## 4  2013    11     6     1535           1540        -5     1657           1725
## 5  2013    11     7     1549           1540         9     1733           1725
## 6  2013    11     8     1539           1540        -1     1706           1725
## # i 12 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, time_index <dttm>
```

(1 point) Flights Per Month

Using `ggplot`, create a plot of the number of flights per month. What, if anything, do you note about the total volume of flights throughout the year? (Don't worry if the plot doesn't tell something interesting about the data. This data is pretty... boring.)

```
tmp_flights <- flights
tmp_flights$unique_flight <- paste(tmp_flights$carrier, tmp_flights$flight)
head(tmp_flights)
```

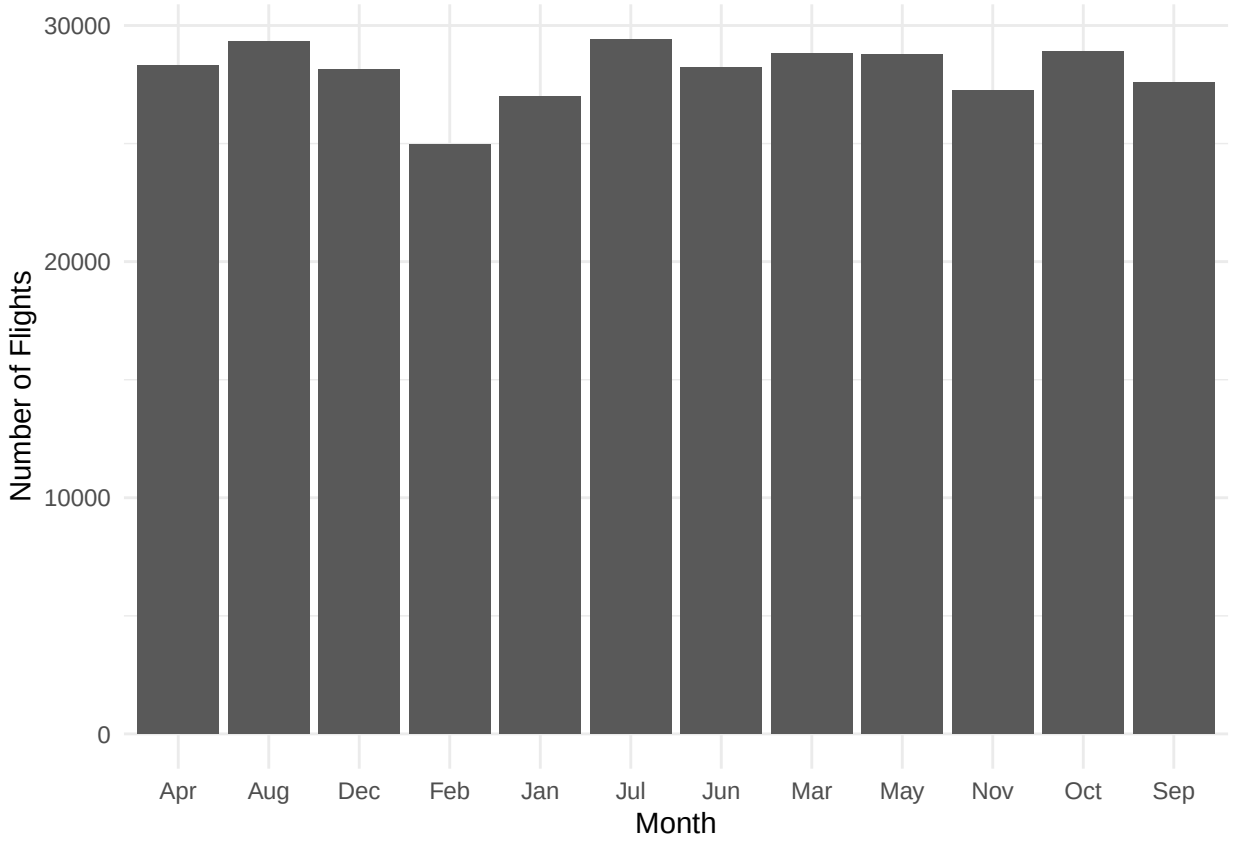
```
## # A tibble: 6 x 21
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
```

```
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013     1     1      517            515         2      830            819
## 2  2013     1     1      533            529         4      850            830
## 3  2013     1     1      542            540         2      923            850
## 4  2013     1     1      544            545        -1     1004           1022
## 5  2013     1     1      554            600        -6      812            837
## 6  2013     1     1      554            558        -4      740            728
## # i 13 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, time_index <dttm>,
## #   unique_flight <chr>
```

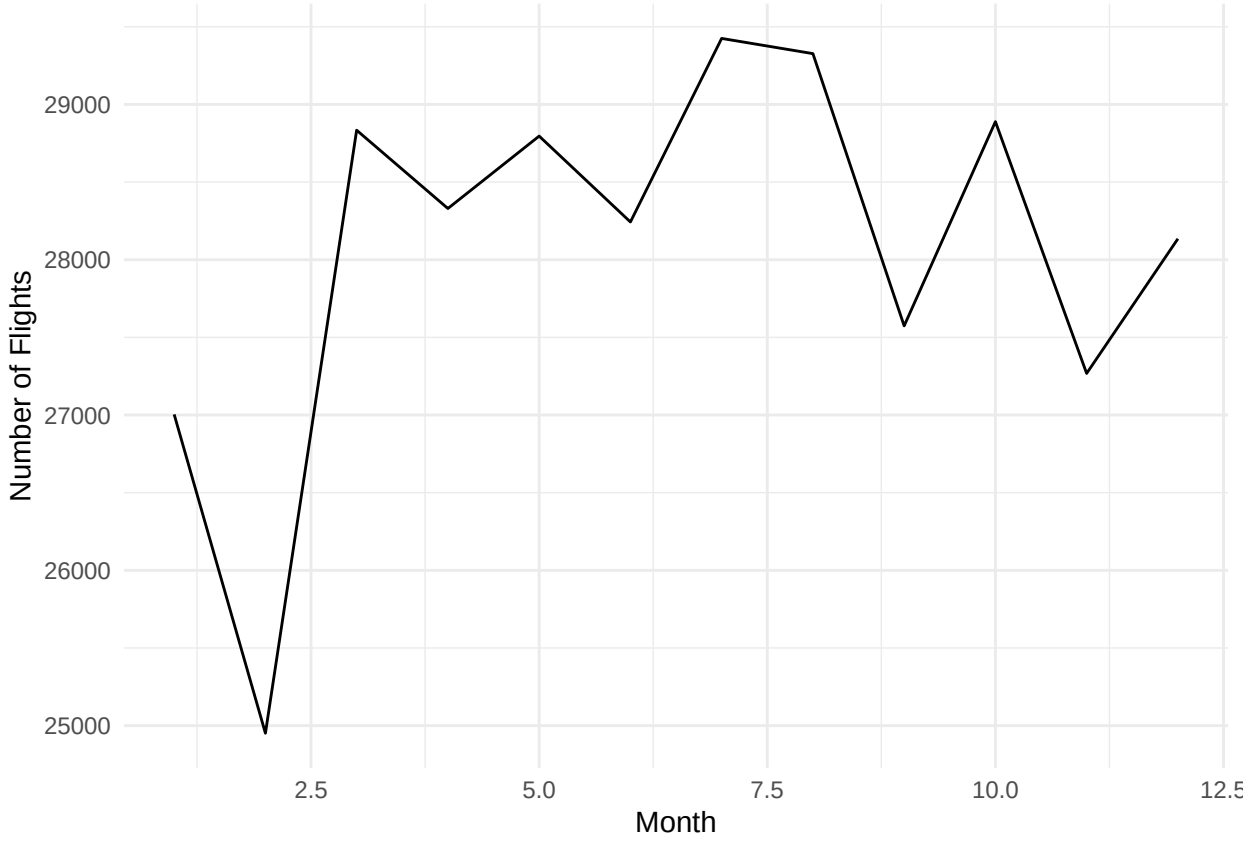
```
tmp_flights <- tmp_flights %>% mutate(month_name=month.abb[month])
month_with_flights <- tmp_flights %>% select(month_name, unique_flight)
head(month_with_flights)
```

```
## # A tibble: 6 x 2
##   month_name unique_flight
##   <chr>      <chr>
## 1 Jan        UA 1545
## 2 Jan        UA 1714
## 3 Jan        AA 1141
## 4 Jan        B6 725
## 5 Jan        DL 461
## 6 Jan        UA 1696
```

```
month_with_flights %>% ggplot(aes(x=month_name)) + geom_bar() + xlab("Month") + ylab("Number of Flights"
```



```
month_flights_df <- tmp_flights %>% select(month, unique_flight) %>% group_by(month) %>% summarise(num_
month_flights_df %>% ggplot(aes(x=month, y = num_flights)) + geom_line() + xlab("Month") + ylab("Number
```



Based on the plot, the number of flights peaks in July and August. February appears to be the month with the lowest number of flights. Unlike the Winter and Fall months, the Spring and Summer months see comparatively high number of flights. Overall, the number of flights remain between 20000 and 30000 throughout the year.

(1 point) The Tropics

Is there a difference in flights to tropical destinations throughout the year? Use the following concept of a tropical destination:

> A tropical destination is one who is "in the tropics" – that is, they are located between the Tropic of Cancer and the Tropic of Capricorn.

1. Using the `airports` dataset, create a new variable, `is_tropical` that notes whether the destination airport is in a tropical latitude.
2. Join this airports data onto the flights data.
3. Produce a plot that shows the volume of flights to tropical and non-tropical destinations, counted as a monthly total, throughout the year.

a. First, try to do this using a `group_by` call that groups on `month` and `is_tropical`. Why does this not work? What is happening when grouping by `month` while also having a time index?
b. Instead, you will need to look into `tsibble::index_by` and combine this with a `lubridate` "extractor" to pull the time object that you want out of the `time_index` variable that you created.
c. To produce the plot, `group_by(is_tropical)`, and `index_by` the month that you extract from your `time_index`. (This is a bit of a strange part of the `geom_*` API, but this might be a useful place to use the `geom_step` geometry to highlight changes in this series.)

4. Comment on what you see in the flights to the tropics, compared to flight to non-tropical destinations.

```
head(airports)
```

```
## # A tibble: 6 x 8
##   faa   name                            lat   lon   alt    tz dst   tzone
##   <chr> <chr>                         <dbl> <dbl> <dbl> <dbl> <chr> <chr>
## 1 04G   Lansdowne Airport              41.1 -80.6  1044    -5 A     America/Ne~
## 2 06A   Moton Field Municipal Airport  32.5 -85.7   264    -6 A     America/Ch~
## 3 06C   Schaumburg Regional            42.0 -88.1   801    -6 A     America/Ch~
## 4 06N   Randall Airport                41.4 -74.4   523    -5 A     America/Ne~
## 5 09J   Jekyll Island Airport          31.1 -81.4    11    -5 A     America/Ne~
## 6 0A9   Elizabethton Municipal Airport 36.4 -82.2  1593    -5 A     America/Ne~
```

```
airports <- airports %>% mutate(is_tropical=case_when(lat >= -23.5 & lat <= 23.5 ~ "Yes",
                                      TRUE ~ "No"))

flights_tsibble <- inner_join(flights_tsibble, airports, by=c("dest"="faa"))
head(flights_tsibble)
```

```
## # A tsibble: 6 x 28 [1m] <UTC>
## # Key:       carrier, flight [1]
##    year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time
##   <int> <int> <int>    <int>          <int>     <dbl>    <int>          <int>
## 1  2013    11     3     1531           1540        -9     1653           1725
## 2  2013    11     4     1539           1540        -1     1712           1725
## 3  2013    11     5     1548           1540         8     1708           1725
## 4  2013    11     6     1535           1540        -5     1657           1725
## 5  2013    11     7     1549           1540         9     1733           1725
## 6  2013    11     8     1539           1540        -1     1706           1725
## # i 20 more variables: arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>, time_index <dttm>, name <chr>,
## #   lat <dbl>, lon <dbl>, alt <dbl>, tz <dbl>, dst <chr>, tzone <chr>,
## #   is_tropical <chr>
```

The tropics are roughly between the latitudes 23.5 degrees North (Tropic of Cancer) and 23.5 degrees South (Tropic of Capricorn).

Grouping by month and is_tropical only does not account for the time_index. Since time_index includes hours and minute, the group by procedure will be done for each recorded minute of each hour of each day of each month for the year.

```
#flights_tsibble  %>%
#    group_by(month, is_tropical)  %>%
#   summarise(total_flights = n())
```

Below is the successful index-by operation.

```
flights_tropical_grouped <- flights_tsibble %>% group_by(is_tropical) %>% index_by(Month=yearmonth(tim
flights_tropical_grouped
```

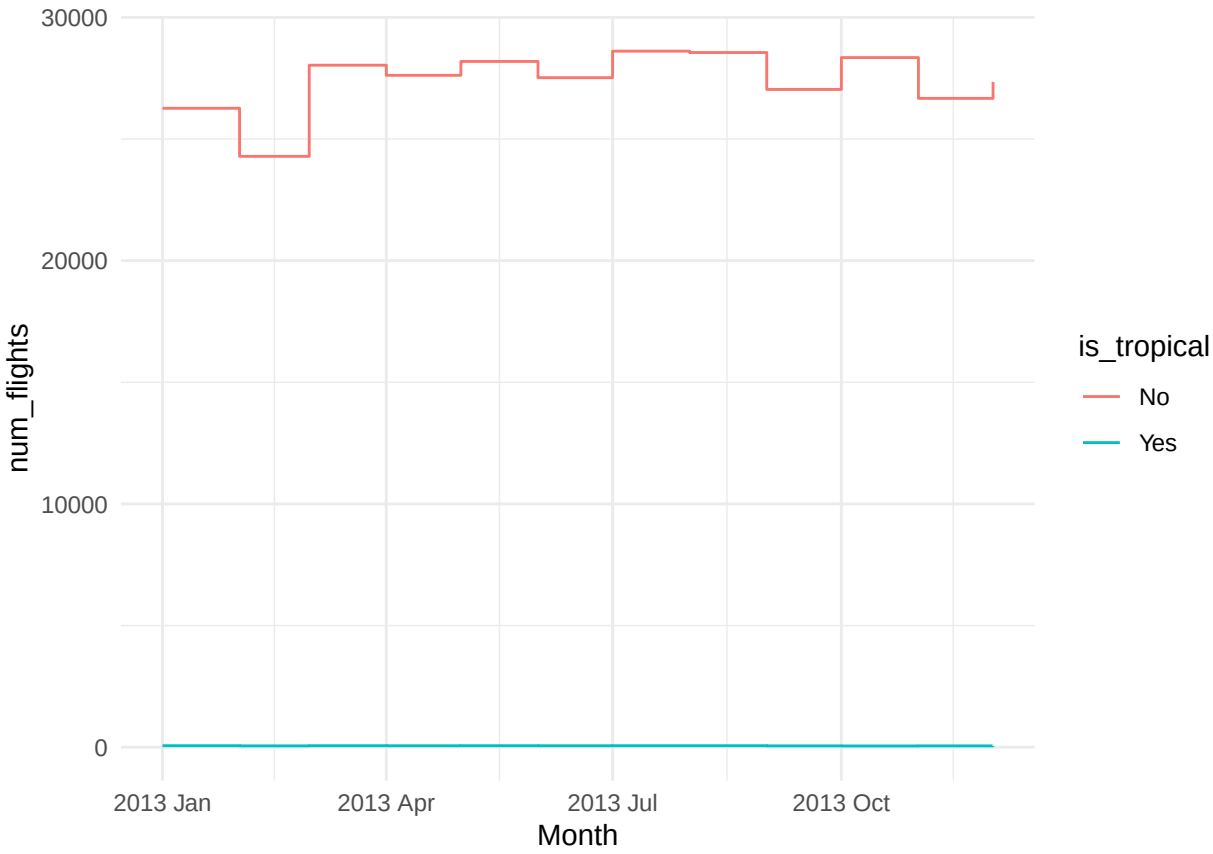
```
## # A tsibble: 24 x 3 [1M]
## # Key:       is_tropical [2]
##    is_tropical    Month num_flights
##    <chr>          <mth>       <int>
##  1 No          2013 Jan       26262
##  2 No          2013 Feb       24287
```

```
##  3 No          2013 Mar       28032
##  4 No          2013 Apr       27616
##  5 No          2013 May       28188
##  6 No          2013 Jun       27520
##  7 No          2013 Jul       28611
##  8 No          2013 Aug       28554
##  9 No          2013 Sep       27036
## 10 No          2013 Oct       28347
## # i 14 more rows
```

```
flights_tropical_grouped$is_tropical
```

```
##  [1] "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"  "No"
## [13] "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes" "Yes"
```

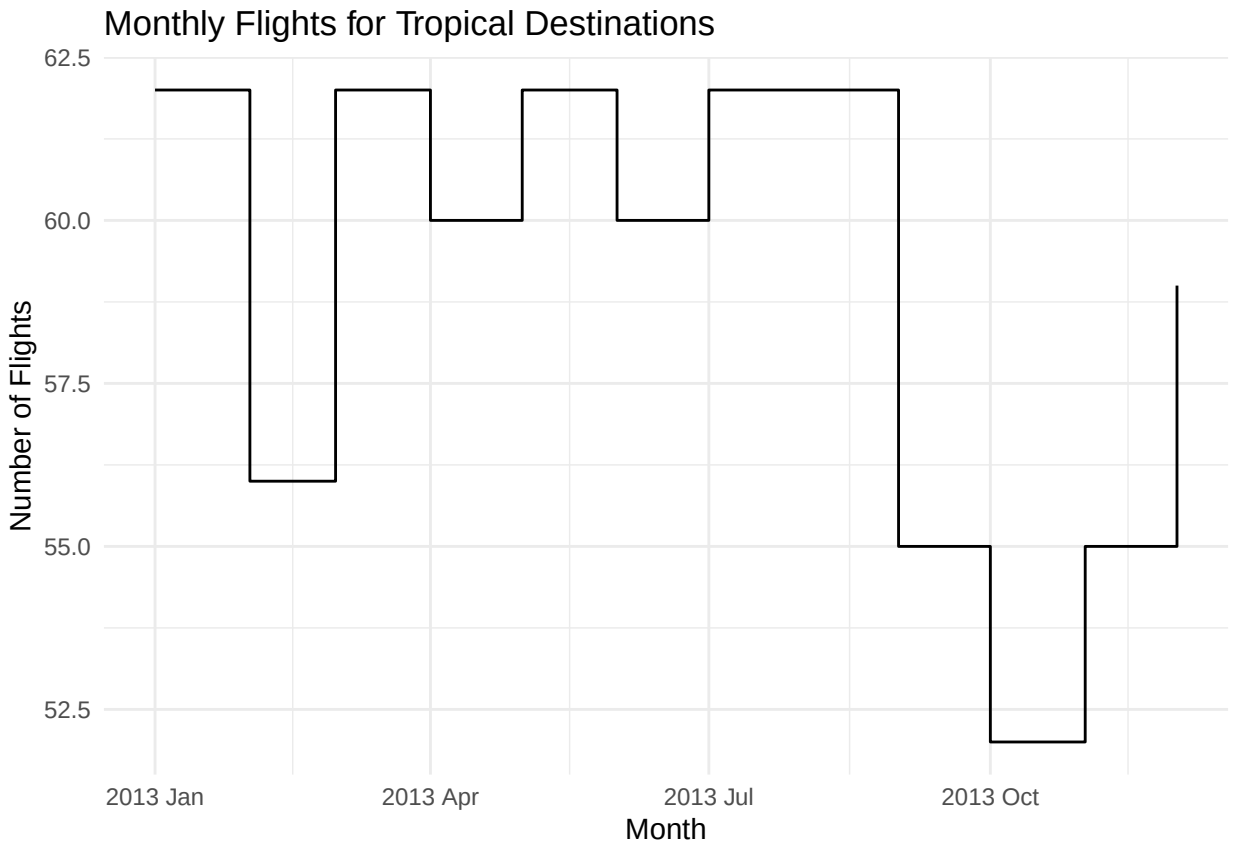
```
flights_tropical_grouped %>% ggplot() + geom_step(mapping=aes(x=Month, y=num_flights, color=is_tropical
```



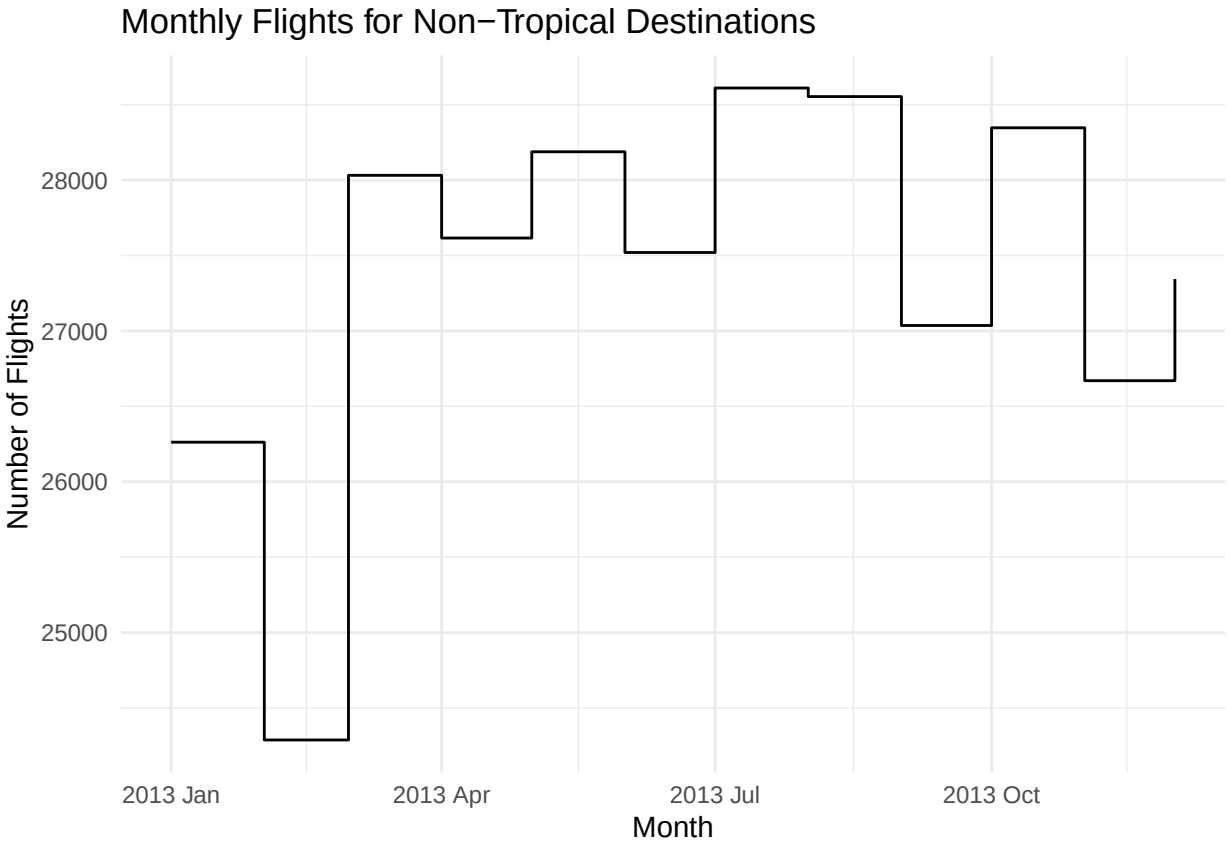
The number of flights to the tropics is much lower than the number of flights not to the tropics throughout the year. This is consistent with the low number of flights to the tropics seen in the flights_tropical_grouped tsibble.

Additional plots of flight counts for tropical destinations and non-tropical destinations are included below for clarity.

```
flights_tropical_grouped %>% filter(is_tropical == "Yes") %>% ggplot() + geom_step(mapping=aes(x=Month,
```




```
flights_tropical_grouped %>% filter(is_tropical == "No") %>% ggplot() + geom_step(mapping=aes(x=Month, y
```



The plots above confirm that there are much more flights to non-tropical destinations than there are to tropical destinations.

Question 2 - Weather at New York Airports

Our goal in this question is to ask you to re-apply what you know about producing time series objects to very similarly structurd data.

(1 point) Create a time series of weather

Turn your attention to the weather data that is provided in the `nycflights13::weather` dataset. Produce a `tsibble` that uses time as a time index, and `origin` as a key for this data. You will notice that there are three origins, "EWR", "JFK" and "LGA".

(Hint: We anticipate that you are going to see the following error on the first time that you try to convert this data frame:

```
Error in `validate_tsibble()`:
A valid tsibble must have distinct rows identified by key and index.
Please use `duplicates()` to check the duplicated rows.
Run `rlang::last_error()` to see where the error occurred.
```

This is a *very* helpful error, with a helpful error message. If you see this error message, we suggest doing as the message suggests, and look into the `duplicates()` function to determine what the issue is. Once you have found the issue, (1) document the issue; (2) propose a solution that seems reasonable; and, (3) implement your proposed solution and keep it moving to answer this question.

```
weather %>%
   mutate(time_index = make_datetime(year, month, day, hour)) %>%
```

```
   duplicates(key = origin, index = time_index) %>%
   glimpse()
```

```
## Rows: 6
## Columns: 16
## $ origin     <chr> "EWR", "EWR", "JFK", "JFK", "LGA", "LGA"
## $ year       <int> 2013, 2013, 2013, 2013, 2013, 2013
## $ month      <int> 11, 11, 11, 11, 11, 11
## $ day        <int> 3, 3, 3, 3, 3, 3
## $ hour       <int> 1, 1, 1, 1, 1, 1
## $ temp       <dbl> 51.98, 50.00, 53.96, 51.98, 55.04, 53.96
## $ dewp       <dbl> 39.02, 39.02, 37.94, 37.94, 39.02, 39.92
## $ humid      <dbl> 61.15, 65.80, 54.51, 58.62, 54.67, 58.89
## $ wind_dir   <dbl> 310, 290, 320, 310, 330, 310
## $ wind_speed <dbl> 6.90468, 5.75390, 9.20624, 6.90468, 9.20624, 8.05546
## $ wind_gust  <dbl> NA, NA, NA, NA, NA, NA
## $ precip     <dbl> 0, 0, 0, 0, 0, 0
## $ pressure   <dbl> 1009.8, 1010.5, 1009.8, 1010.5, 1009.3, 1010.2
## $ visib      <dbl> 10, 10, 10, 10, 10, 10
## $ time_hour  <dttm> 2013-11-03 01:00:00, 2013-11-03 01:00:00, 2013-11-03 01:00:~
## $ time_index <dttm> 2013-11-03 01:00:00, 2013-11-03 01:00:00, 2013-11-03 01:00:~
```

1.) The key origin does not uniquely identify a record, as there are multiple rows with the origin EWR. 2.) Since the weather dataset records hourly meterological data for LGA, JFK, and EWR, the combination of origin and time_hour would create a unique key. This is valid since meterological data would not be recorded more than once for each hour for each origin. 3.) The solution can be verified below.

```
weather %>%
   mutate(time_index = make_datetime(year, month, day, hour)) %>% duplicates(index = time_index, key = c
```

```
## # A tibble: 0 x 16
## # i 16 variables: origin <chr>, year <int>, month <int>, day <int>, hour <int>,
## #   temp <dbl>, dewp <dbl>, humid <dbl>, wind_dir <dbl>, wind_speed <dbl>,
## #   wind_gust <dbl>, precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>, time_index <dttm>
```

We see that there are no duplicates when we combine origin and time_hour.

```
weather <- weather %>%
   mutate(time_index = make_datetime(year, month, day, hour)) %>% as_tsibble(index=time_index, key=c(ori
head(weather)
```

```
## # A tsibble: 6 x 16 [1h] <UTC>
## # Key:       origin, time_hour [6]
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed wind_gust
##   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>     <dbl>
## 1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4         NA
## 2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06        NA
## 3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5         NA
## 4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7         NA
## 5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7         NA
## 6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5         NA
## # i 5 more variables: precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>, time_index <dttm>
```

```
summary(weather)
```

```
##     origin               year          month             day
##  Length:26115       Min.   :2013   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00
##  Mode  :character   Median :2013   Median : 7.000   Median :16.00
##                     Mean   :2013   Mean   : 6.504   Mean   :15.68
##                     3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00
##                     Max.   :2013   Max.   :12.000   Max.   :31.00
##
##       hour            temp            dewp           humid
##  Min.   : 0.00   Min.   : 10.94   Min.   :-9.94   Min.   : 12.74
##  1st Qu.: 6.00   1st Qu.: 39.92   1st Qu.:26.06   1st Qu.: 47.05
##  Median :11.00   Median : 55.40   Median :42.08   Median : 61.79
##  Mean   :11.49   Mean   : 55.26   Mean   :41.44   Mean   : 62.53
##  3rd Qu.:17.00   3rd Qu.: 69.98   3rd Qu.:57.92   3rd Qu.: 78.79
##  Max.   :23.00   Max.   :100.04   Max.   :78.08   Max.   :100.00
##                  NA's   :1        NA's   :1       NA's   :1
##     wind_dir       wind_speed        wind_gust         precip
##  Min.   :  0.0   Min.   :   0.000   Min.   :16.11   Min.   :0.000000
##  1st Qu.:120.0   1st Qu.:   6.905   1st Qu.:20.71   1st Qu.:0.000000
##  Median :220.0   Median :  10.357   Median :24.17   Median :0.000000
##  Mean   :199.8   Mean   :  10.518   Mean   :25.49   Mean   :0.004469
##  3rd Qu.:290.0   3rd Qu.:  13.809   3rd Qu.:28.77   3rd Qu.:0.000000
##  Max.   :360.0   Max.   :1048.361   Max.   :66.75   Max.   :1.210000
##  NA's   :460     NA's   :4          NA's   :20778
##     pressure          visib          time_hour
##  Min.   : 983.8   Min.   : 0.000   Min.   :2013-01-01 01:00:00.0
##  1st Qu.:1012.9   1st Qu.:10.000   1st Qu.:2013-04-01 21:30:00.0
##  Median :1017.6   Median :10.000   Median :2013-07-01 14:00:00.0
##  Mean   :1017.9   Mean   : 9.255   Mean   :2013-07-01 18:26:37.7
##  3rd Qu.:1023.0   3rd Qu.:10.000   3rd Qu.:2013-09-30 13:00:00.0
##  Max.   :1042.1   Max.   :10.000   Max.   :2013-12-30 18:00:00.0
##  NA's   :2729
##    time_index
##  Min.   :2013-01-01 01:00:00.00
##  1st Qu.:2013-04-01 21:30:00.00
##  Median :2013-07-01 14:00:00.00
##  Mean   :2013-07-01 18:05:51.25
##  3rd Qu.:2013-09-30 13:00:00.00
##  Max.   :2013-12-30 18:00:00.00
##
```

We see that more than half of the wind gust column records have missing NA values. We can impute missing values with the mean value of each column for each origin.

```
new_weather <- weather %>% group_by(origin) %>% mutate(temp=replace(temp, is.na(temp), mean(temp, na.rm
summary(new_weather)
```

```
##     origin               year          month             day
##  Length:26115       Min.   :2013   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00
##  Mode  :character   Median :2013   Median : 7.000   Median :16.00
##                     Mean   :2013   Mean   : 6.504   Mean   :15.68
##                     3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00
##                     Max.   :2013   Max.   :12.000   Max.   :31.00
##
```

```
##       hour             temp             dewp           humid
##  Min.   : 0.00   Min.   : 10.94   Min.   :-9.94   Min.   : 12.74
##  1st Qu.: 6.00   1st Qu.: 39.92   1st Qu.:26.06   1st Qu.: 47.05
##  Median :11.00   Median : 55.40   Median :42.08   Median : 61.79
##  Mean   :11.49   Mean   : 55.26   Mean   :41.44   Mean   : 62.53
##  3rd Qu.:17.00   3rd Qu.: 69.98   3rd Qu.:57.92   3rd Qu.: 78.79
##  Max.   :23.00   Max.   :100.04   Max.   :78.08   Max.   :100.00
##                                   NA's   :1       NA's   :1
##     wind_dir       wind_speed       wind_gust         precip
##  Min.   :  0.0   Min.   :   0.000   Min.   :16.11   Min.   :0.000000
##  1st Qu.:120.0   1st Qu.:   6.905   1st Qu.:20.71   1st Qu.:0.000000
##  Median :220.0   Median :  10.357   Median :24.17   Median :0.000000
##  Mean   :199.8   Mean   :  10.518   Mean   :25.49   Mean   :0.004469
##  3rd Qu.:290.0   3rd Qu.:  13.809   3rd Qu.:28.77   3rd Qu.:0.000000
##  Max.   :360.0   Max.   :1048.361   Max.   :66.75   Max.   :1.210000
##  NA's   :460     NA's   :4          NA's   :20778
##     pressure          visib          time_hour
##  Min.   : 983.8   Min.   : 0.000   Min.   :2013-01-01 01:00:00.0
##  1st Qu.:1012.9   1st Qu.:10.000   1st Qu.:2013-04-01 21:30:00.0
##  Median :1017.6   Median :10.000   Median :2013-07-01 14:00:00.0
##  Mean   :1017.9   Mean   : 9.255   Mean   :2013-07-01 18:26:37.7
##  3rd Qu.:1023.0   3rd Qu.:10.000   3rd Qu.:2013-09-30 13:00:00.0
##  Max.   :1042.1   Max.   :10.000   Max.   :2013-12-30 18:00:00.0
##  NA's   :2729
##    time_index
##  Min.   :2013-01-01 01:00:00.00
##  1st Qu.:2013-04-01 21:30:00.00
##  Median :2013-07-01 14:00:00.00
##  Mean   :2013-07-01 18:05:51.25
##  3rd Qu.:2013-09-30 13:00:00.00
##  Max.   :2013-12-30 18:00:00.00
##
```

```
dim(new_weather)
```

```
## [1] 26115    16
```

```
new_weather <- new_weather %>% group_by(origin) %>% mutate(temp = replace(temp, is.na(temp), mean(temp,
new_weather <- new_weather %>% group_by(origin) %>% mutate(dewp = replace(dewp, is.na(dewp), mean(dewp,
new_weather <- new_weather %>% group_by(origin) %>% mutate(humid = replace(humid, is.na(humid), mean(hu
new_weather <- new_weather %>% group_by(origin) %>% mutate(wind_dir = replace(wind_dir, is.na(wind_dir)
new_weather <- new_weather %>% group_by(origin) %>% mutate(wind_speed = replace(wind_speed, is.na(wind_s
new_weather <- new_weather %>% group_by(origin) %>% mutate(wind_gust = replace(wind_gust, is.na(wind_gus
new_weather <- new_weather %>% group_by(origin) %>% mutate(pressure = replace(pressure, is.na(pressure)
head(new_weather)
```

```
## # A tsibble: 6 x 16 [1h] <UTC>
## # Key:       origin, time_hour [6]
## # Groups:    origin [1]
##   origin  year month   day  hour  temp  dewp humid wind_dir wind_speed wind_gust
##   <chr>  <int> <int> <int> <int> <dbl> <dbl> <dbl>    <dbl>      <dbl>     <dbl>
## 1 EWR     2013     1     1     1  39.0  26.1  59.4      270      10.4       24.1
## 2 EWR     2013     1     1     2  39.0  27.0  61.6      250       8.06      24.1
## 3 EWR     2013     1     1     3  39.0  28.0  64.4      240      11.5       24.1
## 4 EWR     2013     1     1     4  39.9  28.0  62.2      250      12.7       24.1
```

```
## 5 EWR     2013     1     1     5  39.0  28.0  64.4      260      12.7       24.1
## 6 EWR     2013     1     1     6  37.9  28.0  67.2      240      11.5       24.1
## # i 5 more variables: precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>, time_index <dttm>
```

```
summary(new_weather)
```

```
##     origin               year          month             day
##  Length:26115       Min.   :2013   Min.   : 1.000   Min.   : 1.00
##  Class :character   1st Qu.:2013   1st Qu.: 4.000   1st Qu.: 8.00
##  Mode  :character   Median :2013   Median : 7.000   Median :16.00
##                     Mean   :2013   Mean   : 6.504   Mean   :15.68
##                     3rd Qu.:2013   3rd Qu.: 9.000   3rd Qu.:23.00
##                     Max.   :2013   Max.   :12.000   Max.   :31.00
##       hour            temp            dewp           humid
##  Min.   : 0.00   Min.   : 10.94   Min.   :-9.94   Min.   : 12.74
##  1st Qu.: 6.00   1st Qu.: 39.92   1st Qu.:26.06   1st Qu.: 47.05
##  Median :11.00   Median : 55.40   Median :42.08   Median : 61.79
##  Mean   :11.49   Mean   : 55.26   Mean   :41.44   Mean   : 62.53
##  3rd Qu.:17.00   3rd Qu.: 69.98   3rd Qu.:57.92   3rd Qu.: 78.79
##  Max.   :23.00   Max.   :100.04   Max.   :78.08   Max.   :100.00
##     wind_dir       wind_speed        wind_gust         precip
##  Min.   :  0.0   Min.   :   0.000   Min.   :16.11   Min.   :0.000000
##  1st Qu.:120.0   1st Qu.:   6.905   1st Qu.:24.14   1st Qu.:0.000000
##  Median :220.0   Median :  10.357   Median :25.14   Median :0.000000
##  Mean   :199.7   Mean   :  10.518   Mean   :25.61   Mean   :0.004469
##  3rd Qu.:290.0   3rd Qu.:  13.809   3rd Qu.:27.56   3rd Qu.:0.000000
##  Max.   :360.0   Max.   :1048.361   Max.   :66.75   Max.   :1.210000
##     pressure          visib          time_hour
##  Min.   : 983.8   Min.   : 0.000   Min.   :2013-01-01 01:00:00.0
##  1st Qu.:1013.5   1st Qu.:10.000   1st Qu.:2013-04-01 21:30:00.0
##  Median :1017.8   Median :10.000   Median :2013-07-01 14:00:00.0
##  Mean   :1017.9   Mean   : 9.255   Mean   :2013-07-01 18:26:37.7
##  3rd Qu.:1022.3   3rd Qu.:10.000   3rd Qu.:2013-09-30 13:00:00.0
##  Max.   :1042.1   Max.   :10.000   Max.   :2013-12-30 18:00:00.0
##    time_index
##  Min.   :2013-01-01 01:00:00.00
##  1st Qu.:2013-04-01 21:30:00.00
##  Median :2013-07-01 14:00:00.00
##  Mean   :2013-07-01 18:05:51.25
##  3rd Qu.:2013-09-30 13:00:00.00
##  Max.   :2013-12-30 18:00:00.00
```

As shown in the summary above, there are no missing values.

(4 points) Plot temperature

With this weather data, produce the following figure of the temperature every hour, for each of the origins.

This figure contains five separate plots:

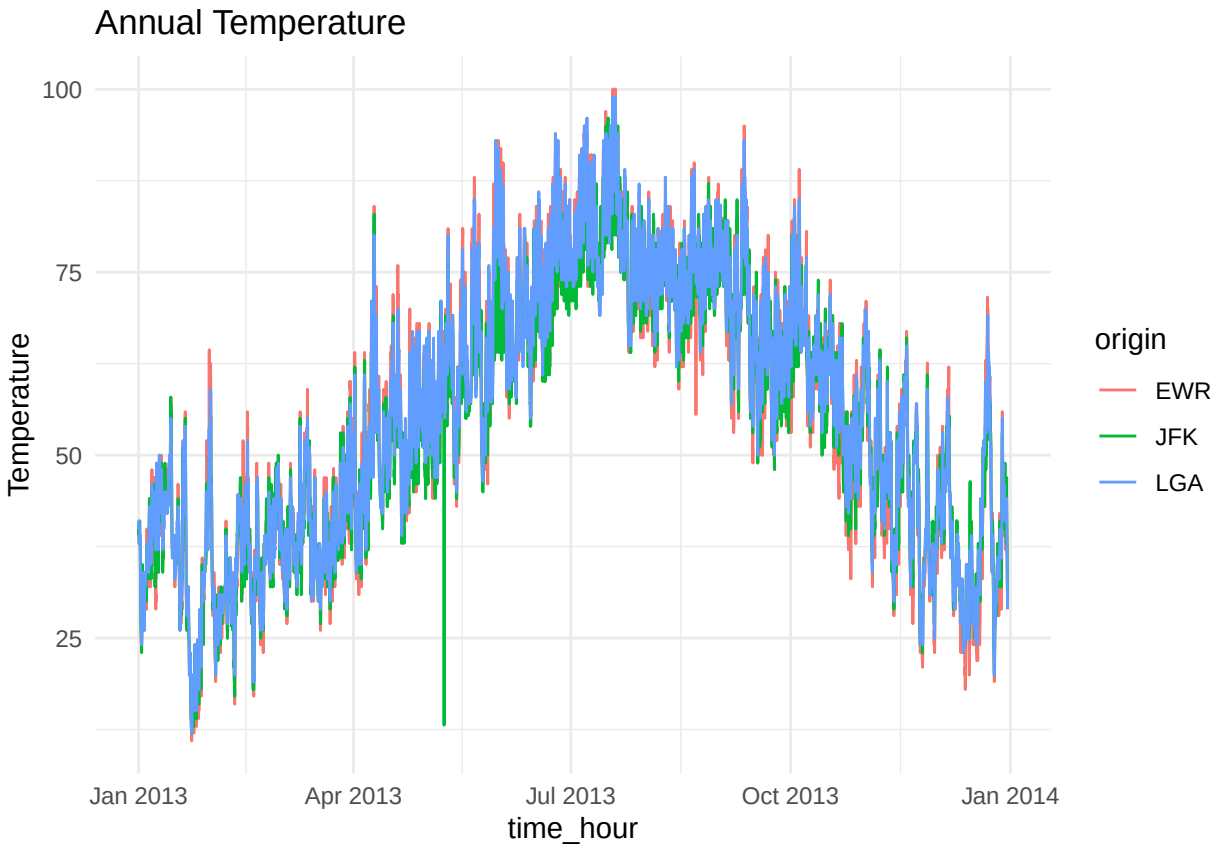
- One that shows the entire year's temperature data;
- Two that show the month of January and July; and,
- Two that show the first week of January and July.

You might think of these plots as "zooming in" on the time series to show more detail.

In your workflow, first create each of the plots. Then, use the `patchwork` package to compose each of these plots into a single figure.

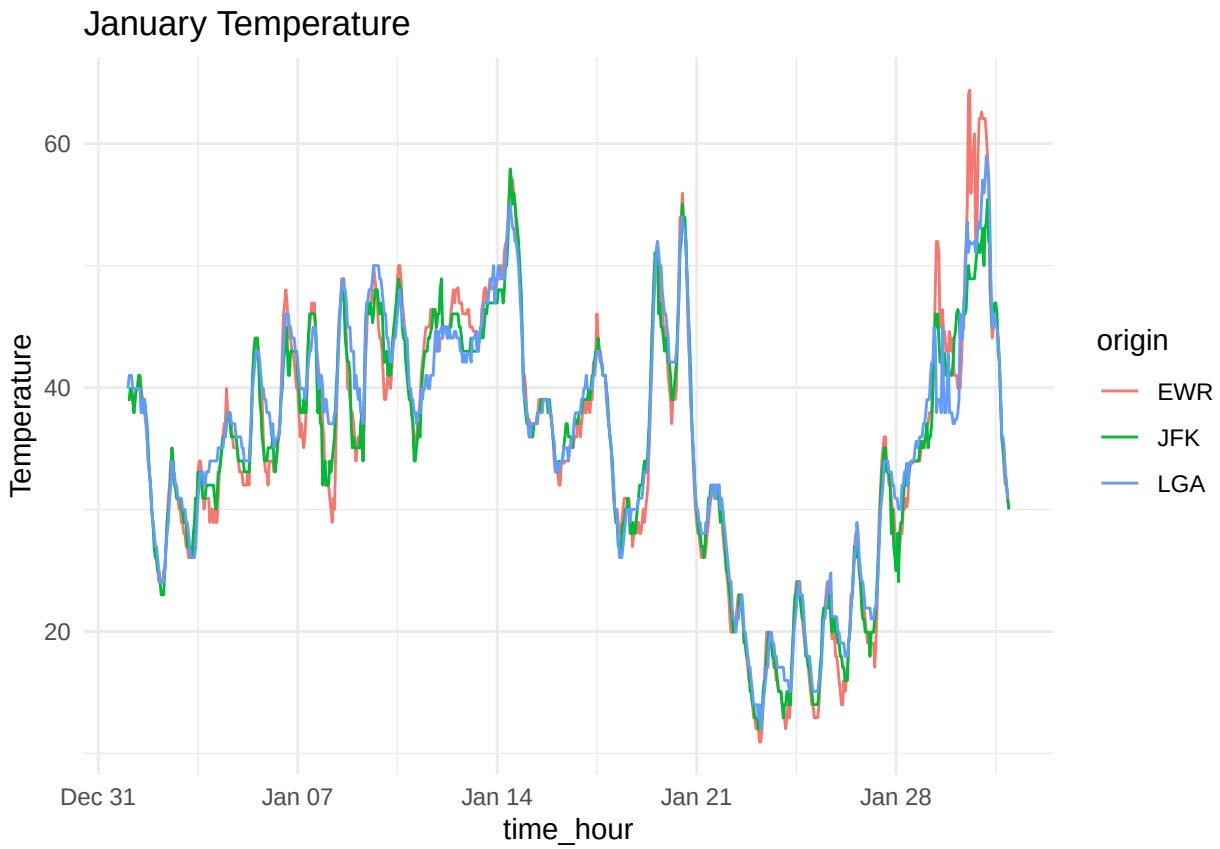
After you produce this figure, comment on what you notice at each of these scales and the figure overall.

```
yearly_plot <- new_weather %>% ggplot(aes(x=time_hour, y = temp, color=origin)) + geom_line() + ylab("T
yearly_plot
```

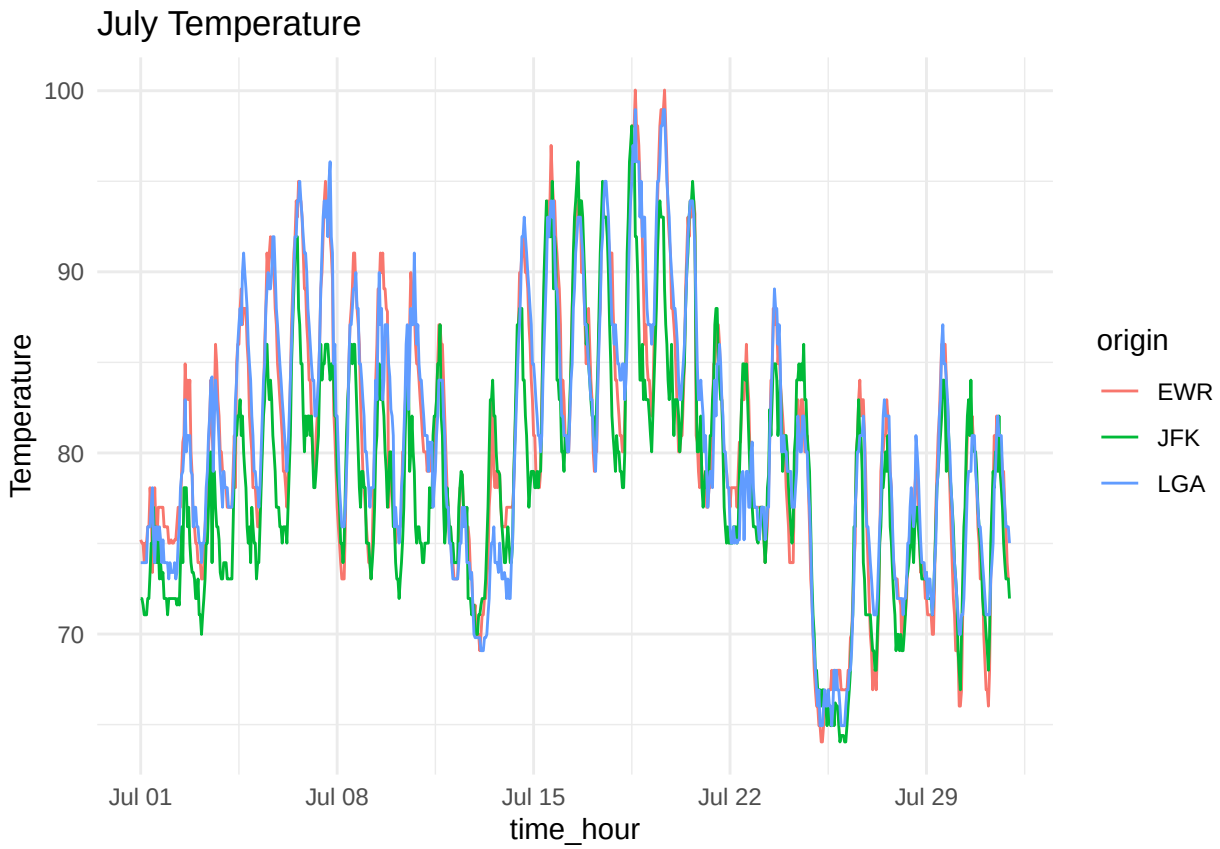


For each origin EWR, JFK, and LGA, the overall trends are similar across the entire year of 2013, with significant oscillations. Since the average temperature starts around 35 in January but peaks to around 80 in July before decreasing back to around 40 by the start of 2014, the time series of temperature for each of the origins are not mean stationary. Moreover, the lumps in oscillations at certain intervals of the year indicate that the variance of the temperature is not constant.

```
january_plot <- new_weather %>% filter(month == 1) %>% ggplot(aes(x=time_hour, y = temp, color=origin))
january_plot
```

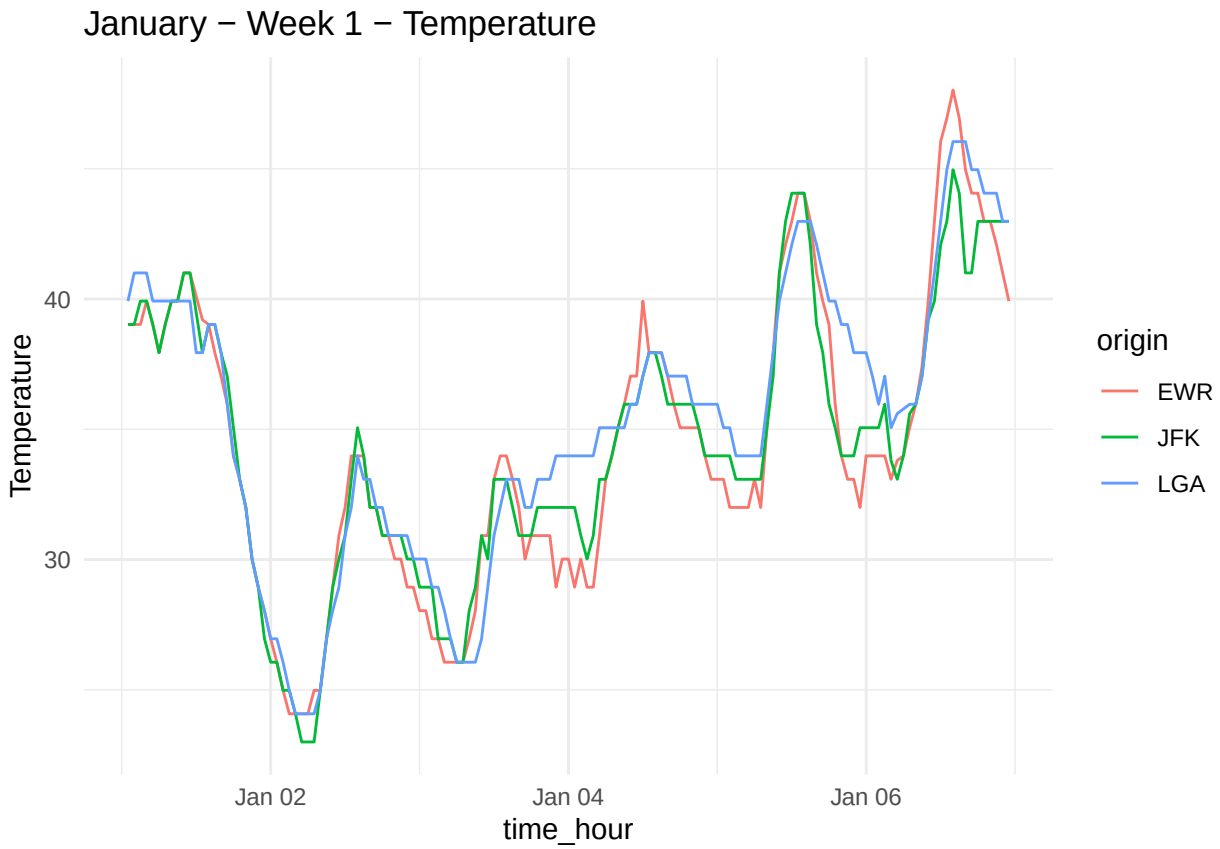



```
july_plot <- new_weather %>% filter(month == 7) %>% ggplot(aes(x=time_hour, y = temp, color=origin)) + g
july_plot
```



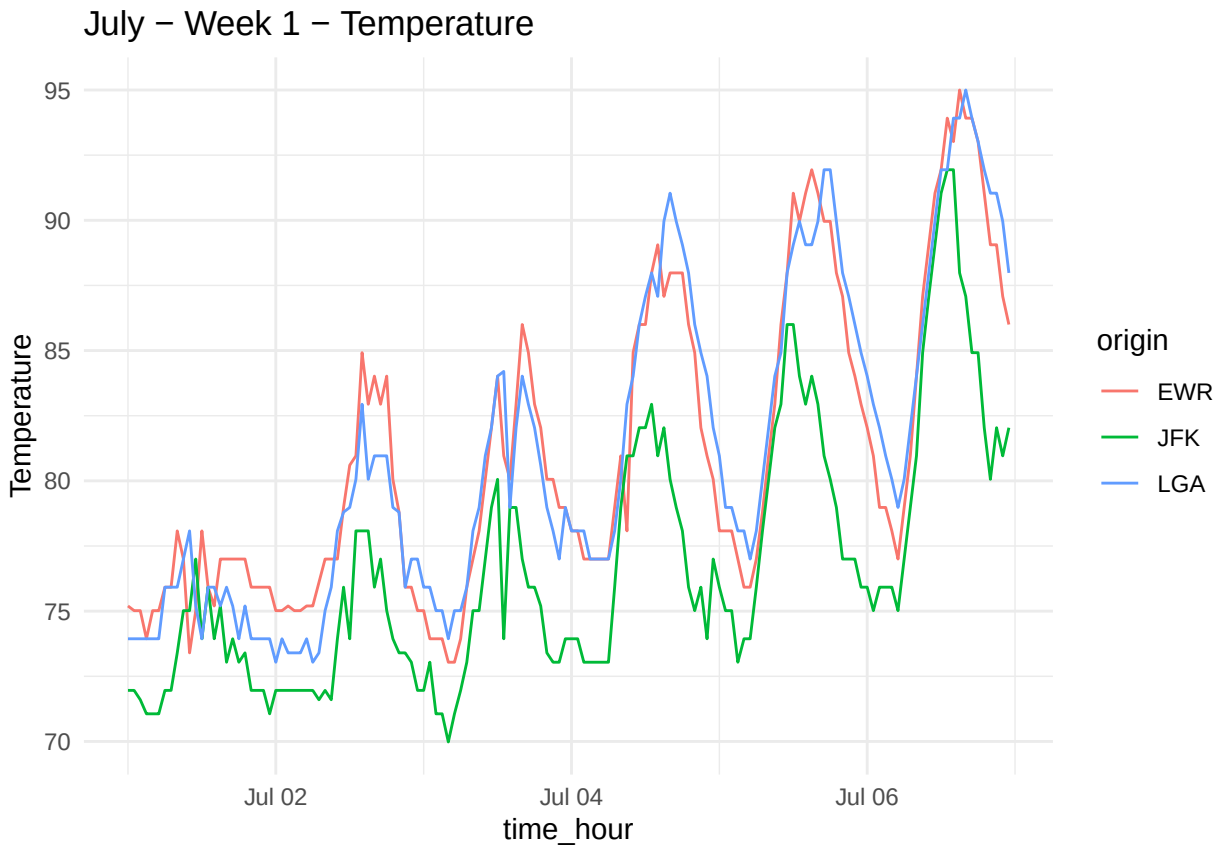
The July temperature plot shows a non-stationary time-series for all origins since the average temperature is not consistent throughout the month. Additionally, the temperature does not vary by the same amount throughout July, as some oscillations are visibly larger than others.

```
january_first_week <- new_weather %>% filter(month == 1) %>% filter(day >= 1 & day < 7) %>% ggplot(aes(
january_first_week
```



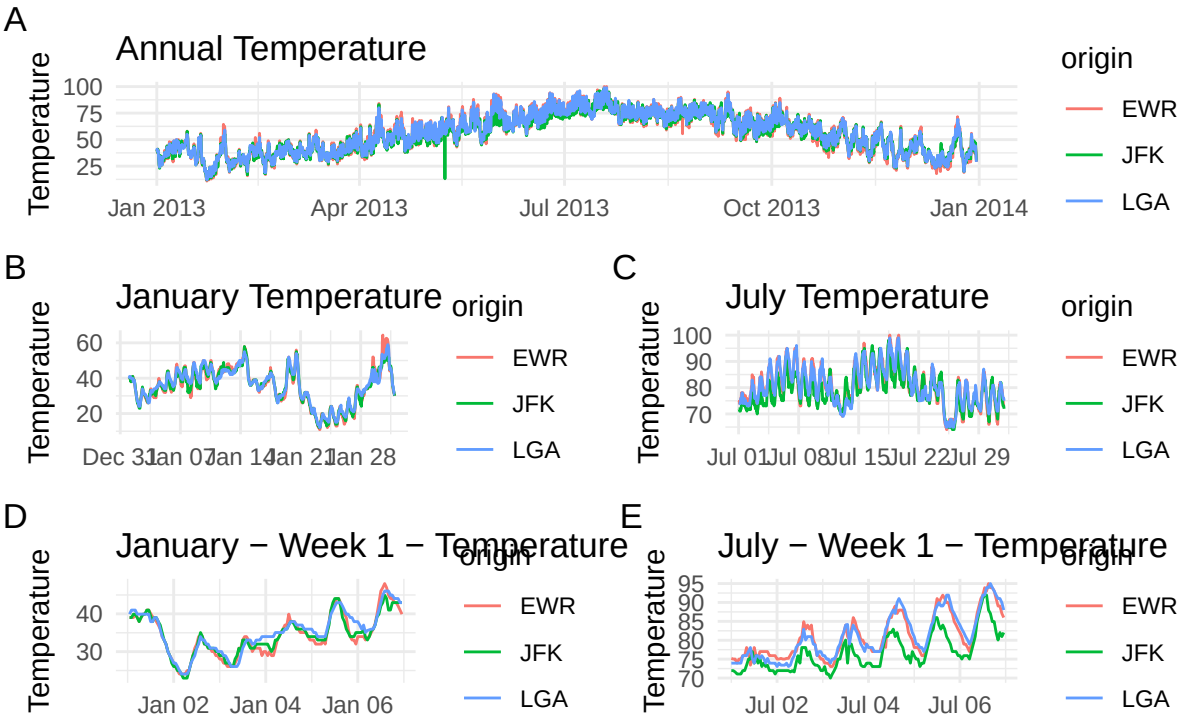
The above plot shows a non-stationary time series, as there is an upwards trend in the temperature from Jaunary 3 onwards.

```
july_first_week <- new_weather %>% filter(month == 7) %>% filter(day >= 1 & day < 7) %>% ggplot(aes(x=t
july_first_week
```



The plot above shows a non-stationary time series since the mean temperature throughout the first week of July trends upward.

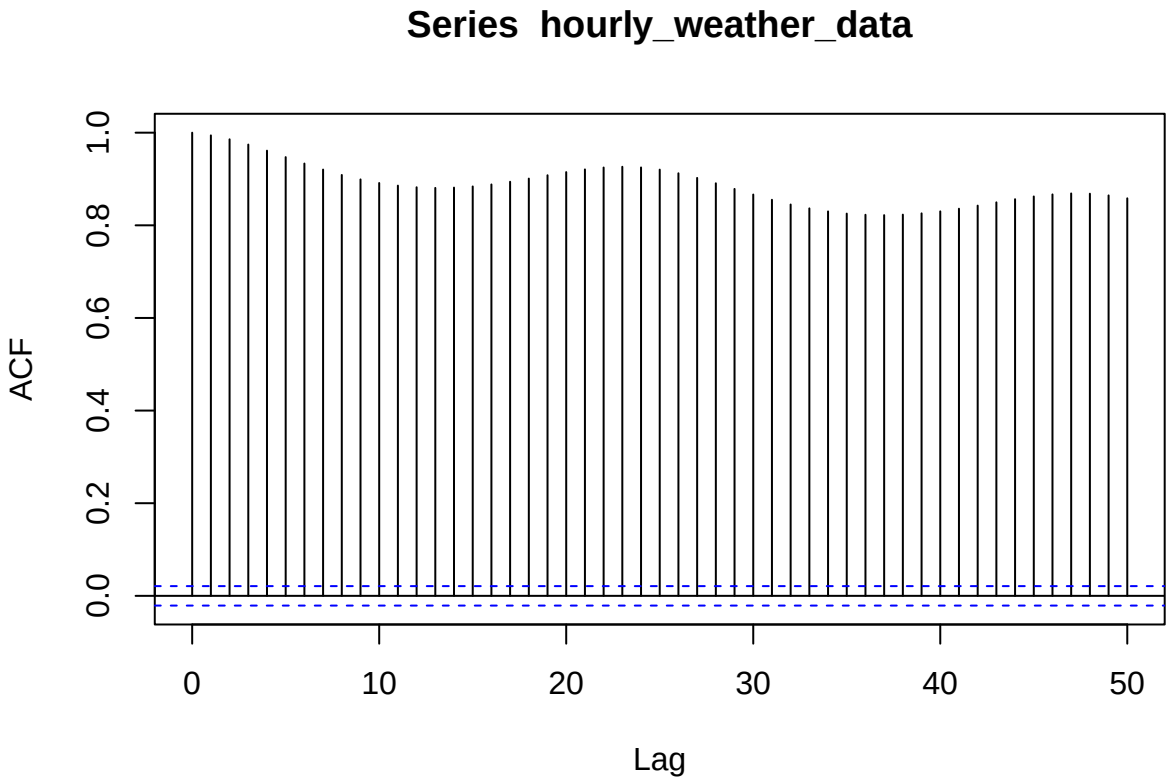
```
library(patchwork)
## Uncomment

 yearly_plot /
   (january_plot | july_plot) /
   (january_first_week | july_first_week) +
   plot_annotation(
     title    = 'Temperature at New York City Airports',
     subtitle = 'Many Different Views',
     tag_levels = 'A') &
   labs(x = NULL, y = 'Temperature')
```



(1 point) Hourly ACF

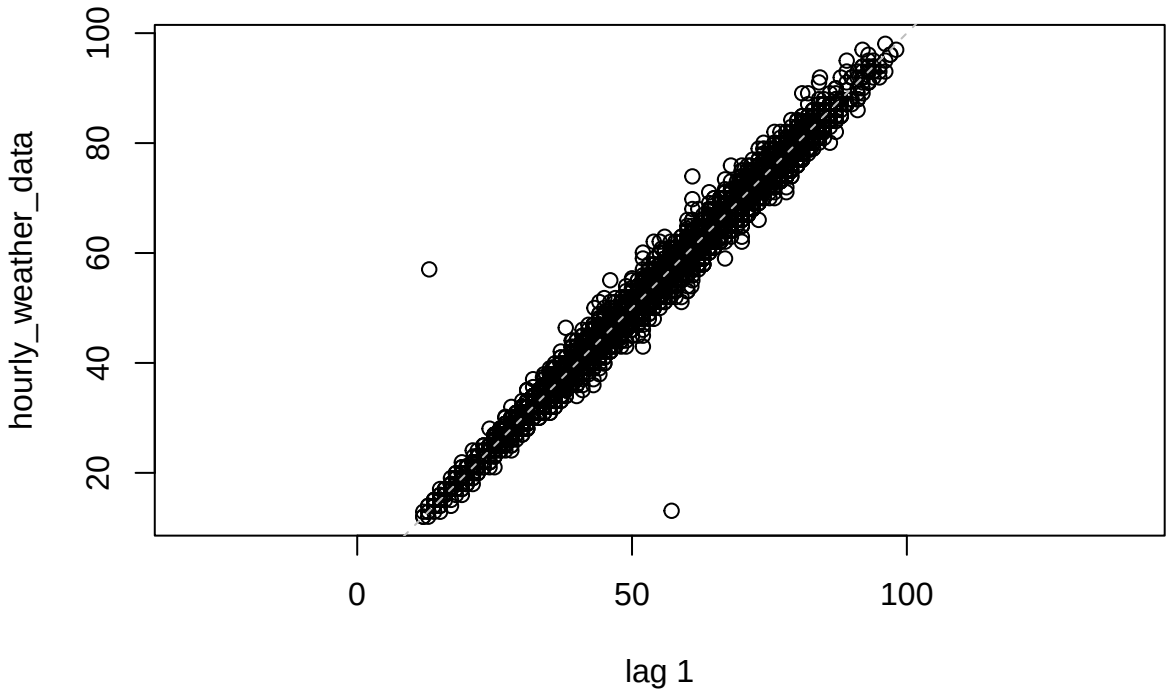
At the hourly level, produce an ACF and a lag plot at JFK. What do you learn from these plots? (Note that you can suppress all the coloring in the `gg_lag` call if you pass an additional argument, `color = 1`.)

```
hourly_weather_data <- (new_weather %>% filter(origin == 'JFK'))$temp

hourly_acf <- acf(hourly_weather_data, lag.max = 50)
```



We can see from the above ACF plot that there are high positive correlations between current observations and their lags. All of the correlations fall outside the blue dotted region, indicating that we have evidence against the null hypothesis that the correlation at all lags are 0 at the 5% level.

```
hourly_lag <- lag.plot(hourly_weather_data)
```

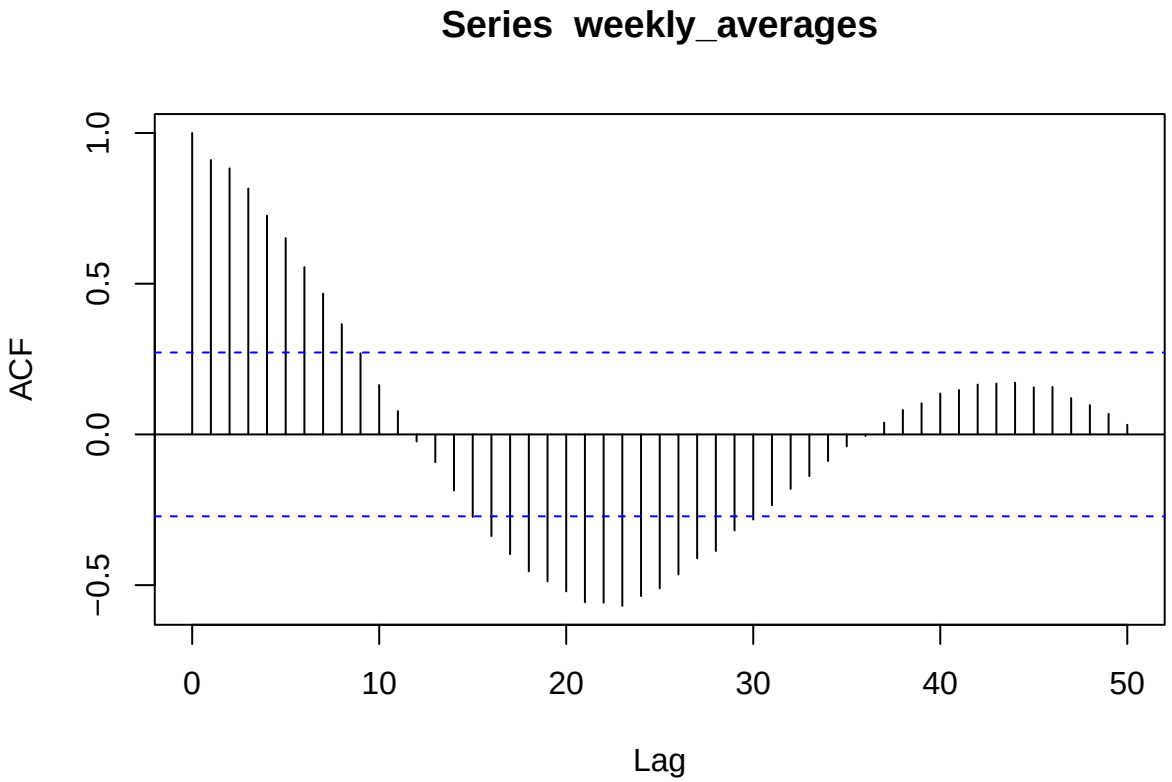


From the plot above, we can spot high correlation between hourly temperature and the first lag.

(1 point) Weekly ACF

At the weekly level, produce an ACF and a lag plot of the weekly average temperature at JFK. What do you learn from these plots?

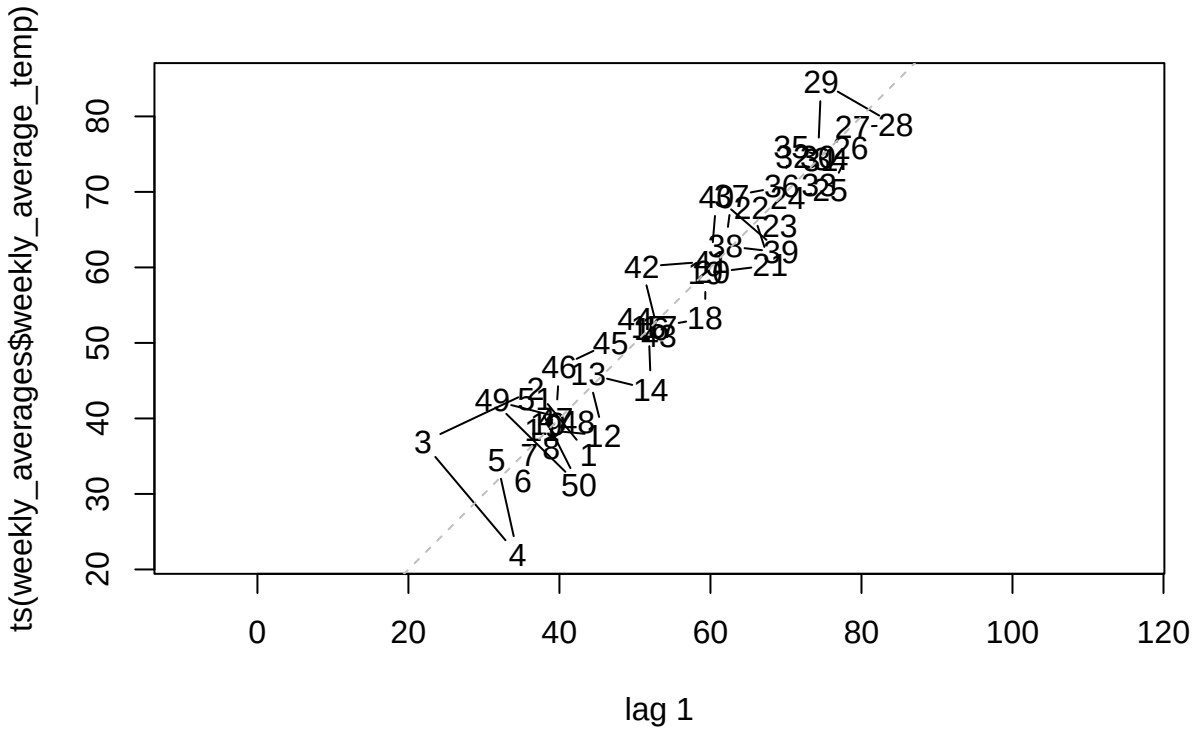
```
temp_weather <- new_weather
temp_weather$week <- ceiling(temp_weather$day / 7)
weekly_averages <- temp_weather %>% filter(origin == 'JFK') %>% index_by(week=week(time_index)) %>% summ

weekly_acf <- acf(weekly_averages, lag.max = 50)
```




```
## make the plot
```

We see both positive and negative correlations with the lags. Positive correlations exist with the first several lags, and negative correlations occur between lags 15 and 30. The correlations with the rest of the lags are within the dotted blue region, indicating that we do not have strong evidence to reject the null hypothesis that the correlations with those lags are 0.

```
weekly_lag <- lag.plot(ts(weekly_averages$weekly_average_temp))
```

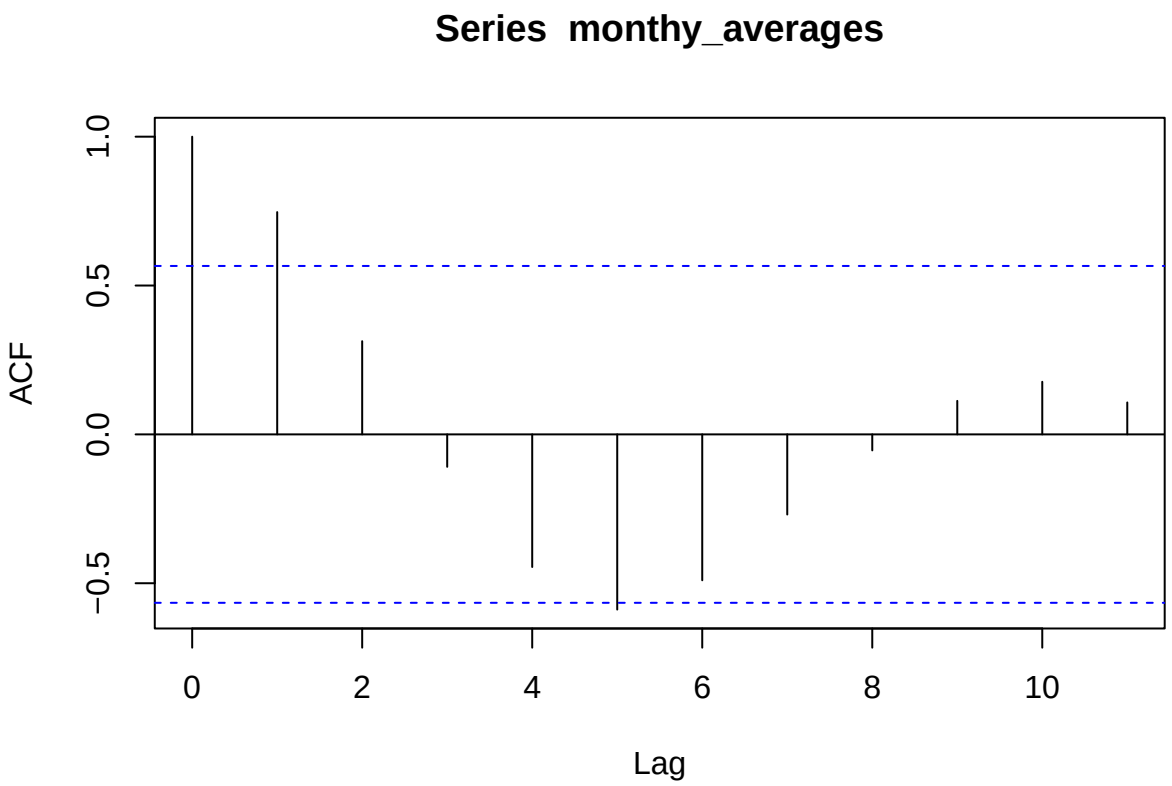


From the plot above, we can spot high correlation between weekly average temperature and the first lag.

(1 point) Monthly ACF

At the monthly level, produce an ACF plot of the monthly average temperature at JFK. What do you learn from these plots?

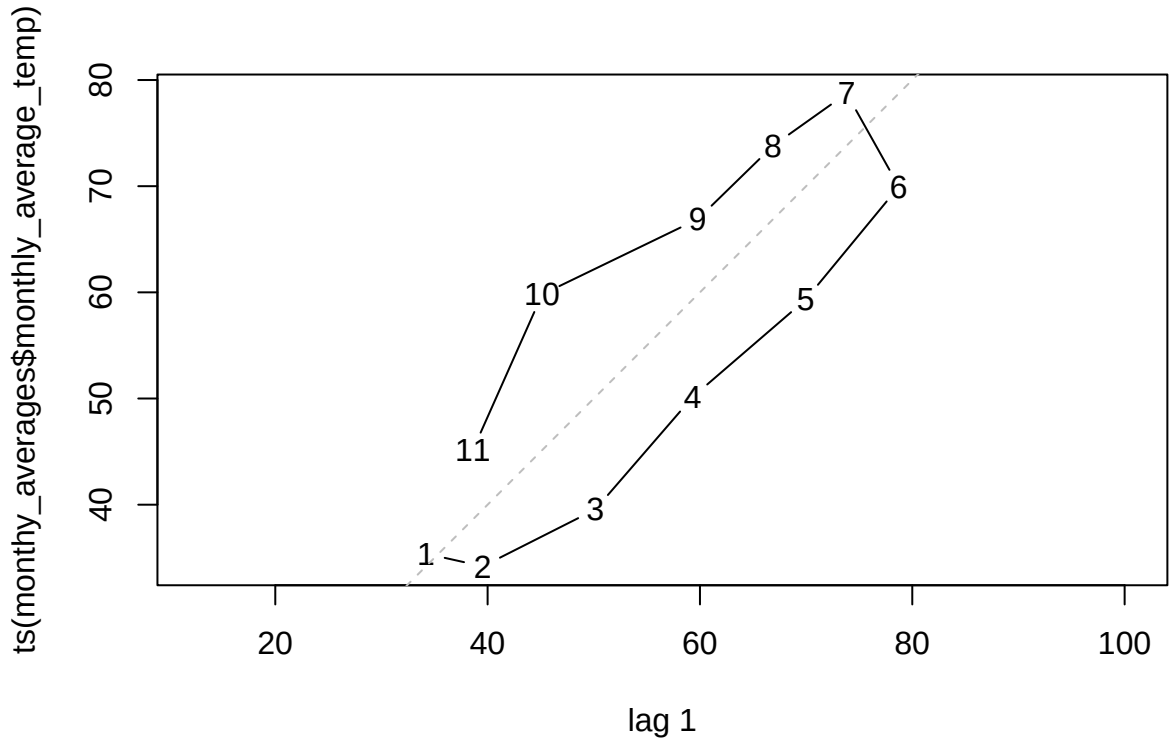
```
monthy_averages <- temp_weather %>% filter(origin == 'JFK') %>% index_by(month=month(time_index)) %>% su

monthly_acf <- acf(monthy_averages, lag.max = 20)
```




```
## make the plot
```

We can see that the correlations with the lags reach 0 before the second lag, indicating that the only correlations are those that exist between the current observation, the observation itself, and the first lag.

```
monthly_lag <- lag.plot(ts(monthy_averages$monthly_average_temp))
```



From the plot above, we can spot high correlation between monthly average temperature and the first lag.

Question 3 - Evaluate Time Series Objects

In this section, we are asking you to use the plotting tools that you have learned in the course to evaluate a time series of "unknown" origin. This week, we will simply be describing what we see in the time series; in future weeks we will also be conducting tests to evaluate whether these are stationary and the order of the time series.

For each time series that you evaluate, provide enough understanding of the series using plots and summaries that a collaborator would agree with your assessment, but do not use more than one printed page per dataset.

This will assuredly mean that not *every* plot or diagnostic that you produce initially will make it to what you present. Edit with intent – what you show your audience should move forward your assessment.

To begin, load the data set constructor, which is stored in `./dataset_generator/`. Within this folder, there is a file named `make_datasets.R`.

- By issuing the `source()` call, you will bring this function into the global namespace, and so can then execute the function by issuing `make_datasets()`.
- We have elected to use one (clumsy) idiom in this function – we have elected to assign objects that are generated *within* the function scope out into the global scope. If you look into `make_datasets()`, which is possible by issuing the function name without the parentheses, you will see that we are assigning using `<<-`. This reads in a similar way to the standard assignment, `<-`, but works globally. We have elected to do this so that all the time series objects that you create are available to you (and to us as graders) at the top, global-level of your session.
- While you *could* try to reverse engineer the randomization that we've built in this function to figure out which generator is associated with which data – please don't. Or, at least, don't until you've finished your diagnostics.

This function will make five data sets and store them in the global environment. They will be named, rather creatively:

- dataset_1
- dataset_2
- dataset_3
- dataset_4
- dataset_5

Your task is to use the plots and concepts covered in lecture and in *Forecasting, Principles and Practices* to describe the series that you see. Are any of these series white noise series? Do any of these series show trends or seasonal patterns?

For each series, using the `patchwork` library to layout your plots, produce a figure that:

- Shows the time series in the first plot;
- Shows the relevant diagnostic plots in one or two more plots within the same figure.

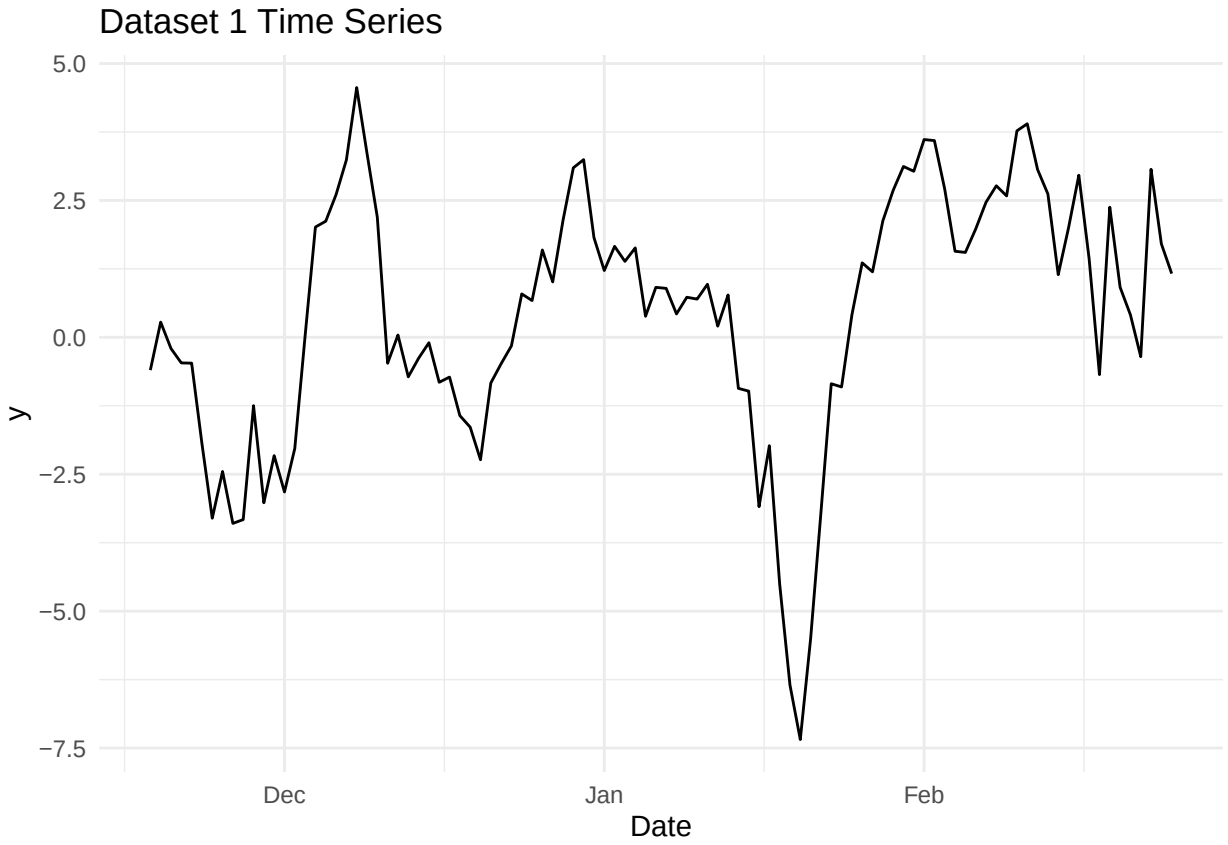
The additional plots could be lag plots, autocorrelation plots, partial autocorrelation plots, or whatever you think makes it the most clear for an interested audience who is as familiar as you with time series analysis come to an understanding of the series.

Along with each plot, include descriptive text (at least several sentences, but not more than a paragraph or two) that describes what, if anything you observe in the series. This is you chance to state what you see, so that your audience can (a) be informed of your interpretation; and (b) come to their own interpretation.

Your analysis and description of each dataset should fit onto a single PDF page.

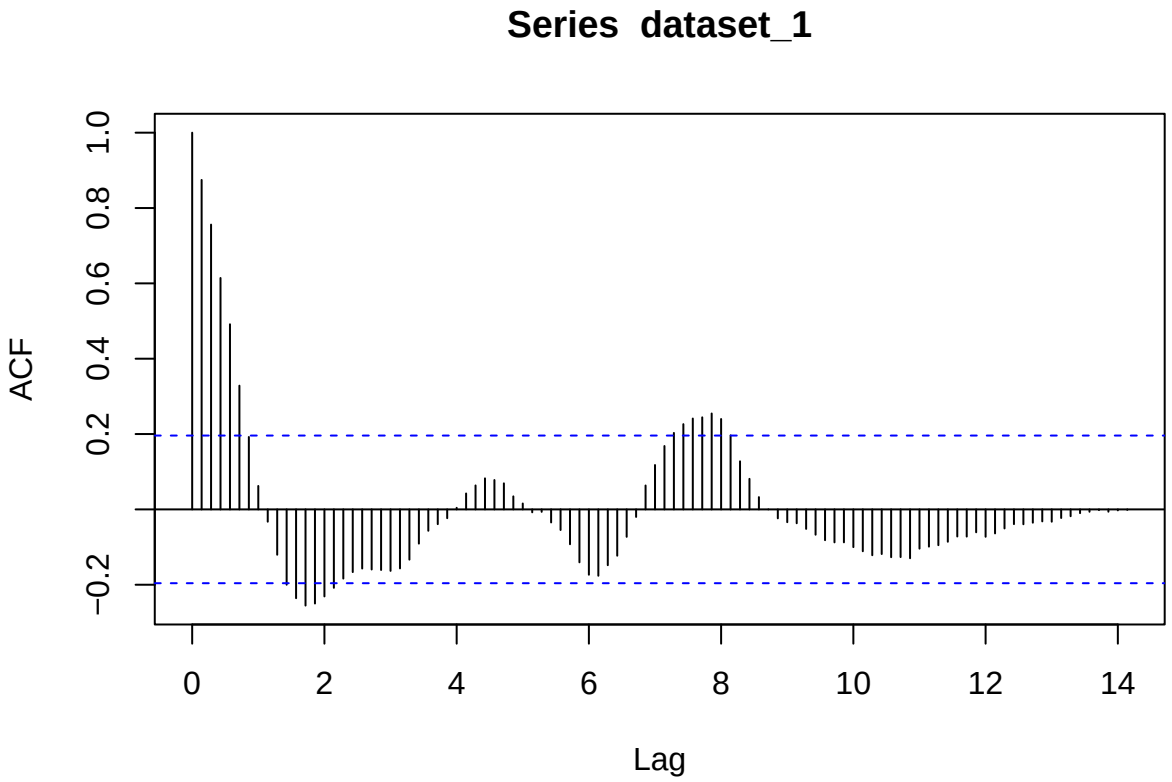
(1 point) Dataset One

```
dataset_1 %>% ggplot(aes(x=date, y = y)) + geom_line() + xlab("Date") + ggtitle("Dataset 1 Time Series")
```



The time series is not stationary since it does not have a constant mean. The lack of stationarity rules out the time series being white noise. A downwards trend occurs between December and January, and an upwards trend occurs between January and the end of February. There does not seem to be any seasonality.

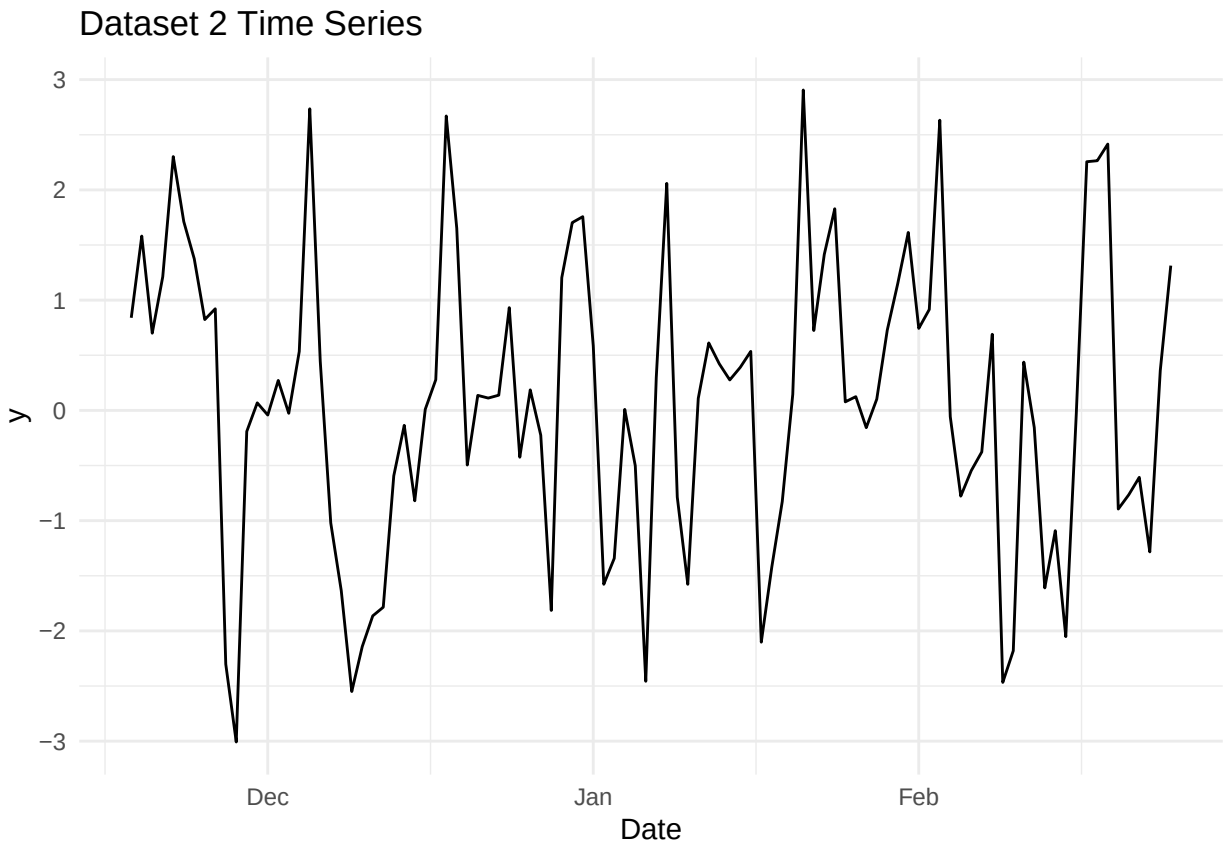
```
dataset_1_acf <- acf(dataset_1, lag.max = 100)
```



We can see from the ACF figure above that positive and negative correlations with the lags occur. However, for lags beyond 8, the correlations between the current observed values and the lags shrinks to 0.

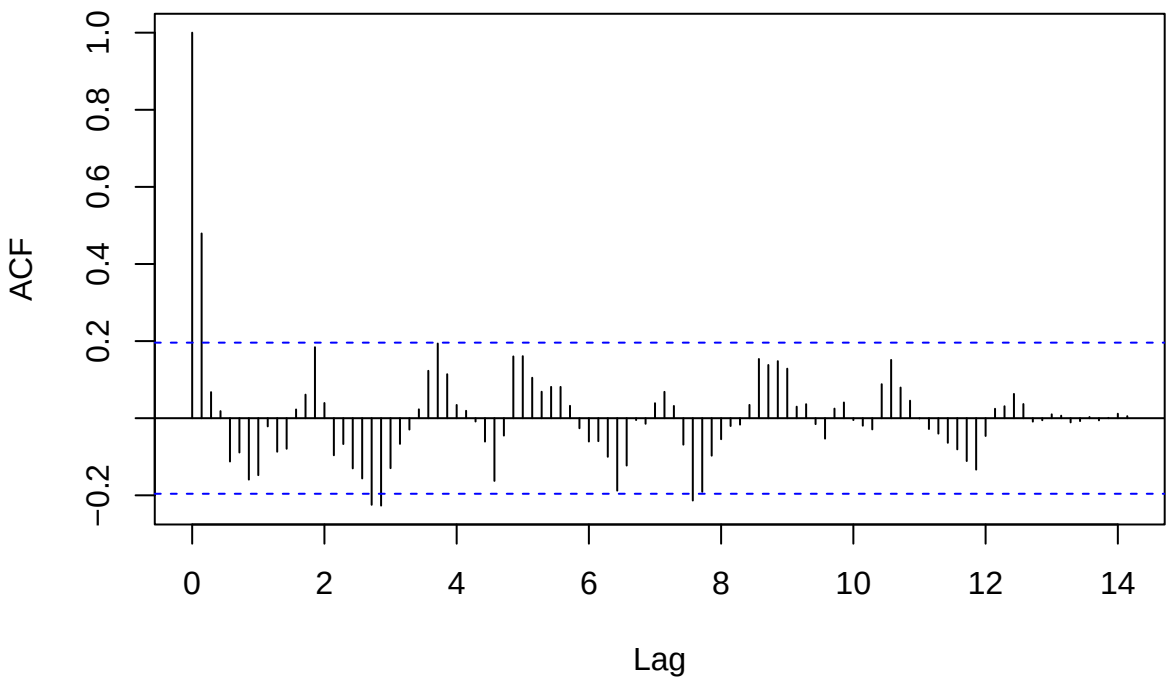
(1 point) Dataset Two

```
dataset_2 %>% ggplot(aes(x=date, y = y)) + geom_line() + xlab("Date") + ggtitle("Dataset 2 Time Series")
```



The time series above has not visible upward or downwards trends. Hence, it can be concluded that the mean hovers around 0. Since the mean is 0 and the variance more or less stays the same throughout, the time series can be considered white noise. This also means that the time series is stationary.

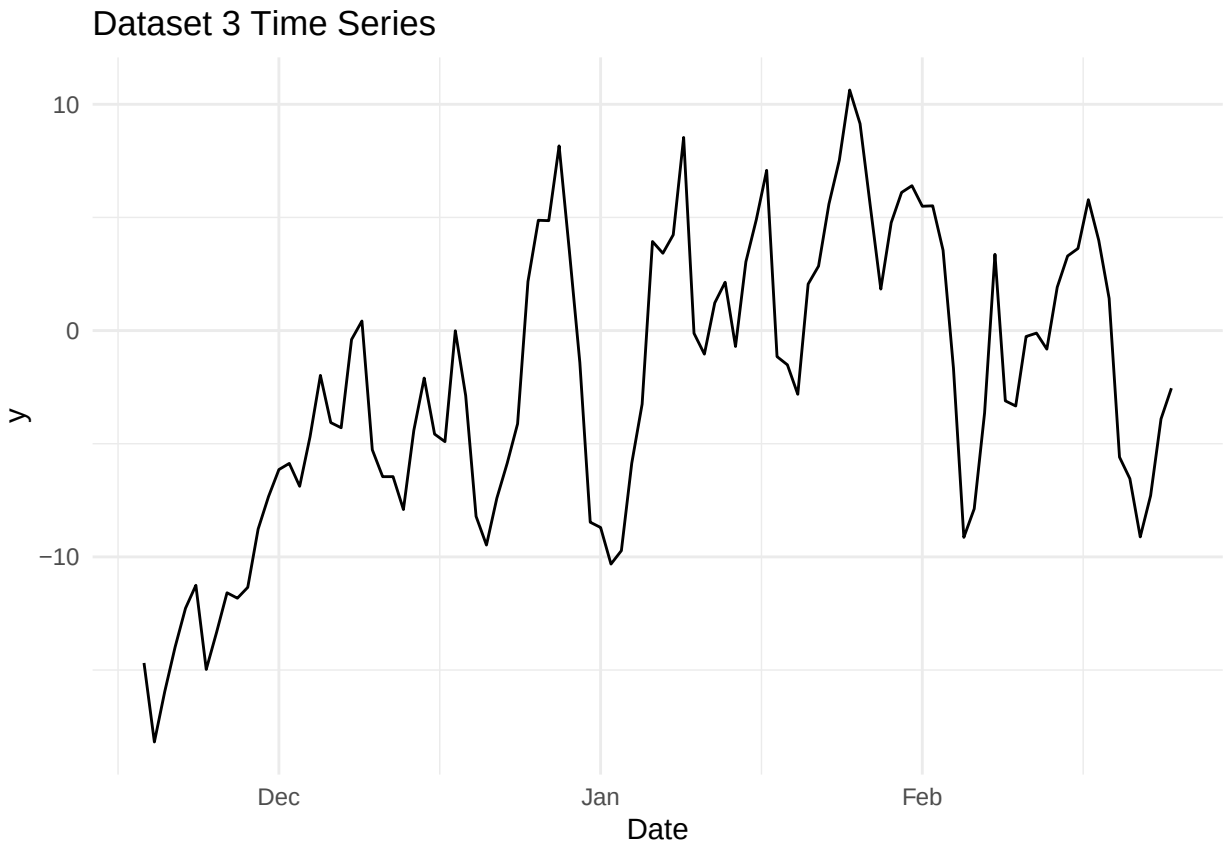
```
dataset_2_acf <- acf(dataset_2, lag.max = 100)
```



The lack of correlations with much of the lags is consistent with the series being close to white noise.

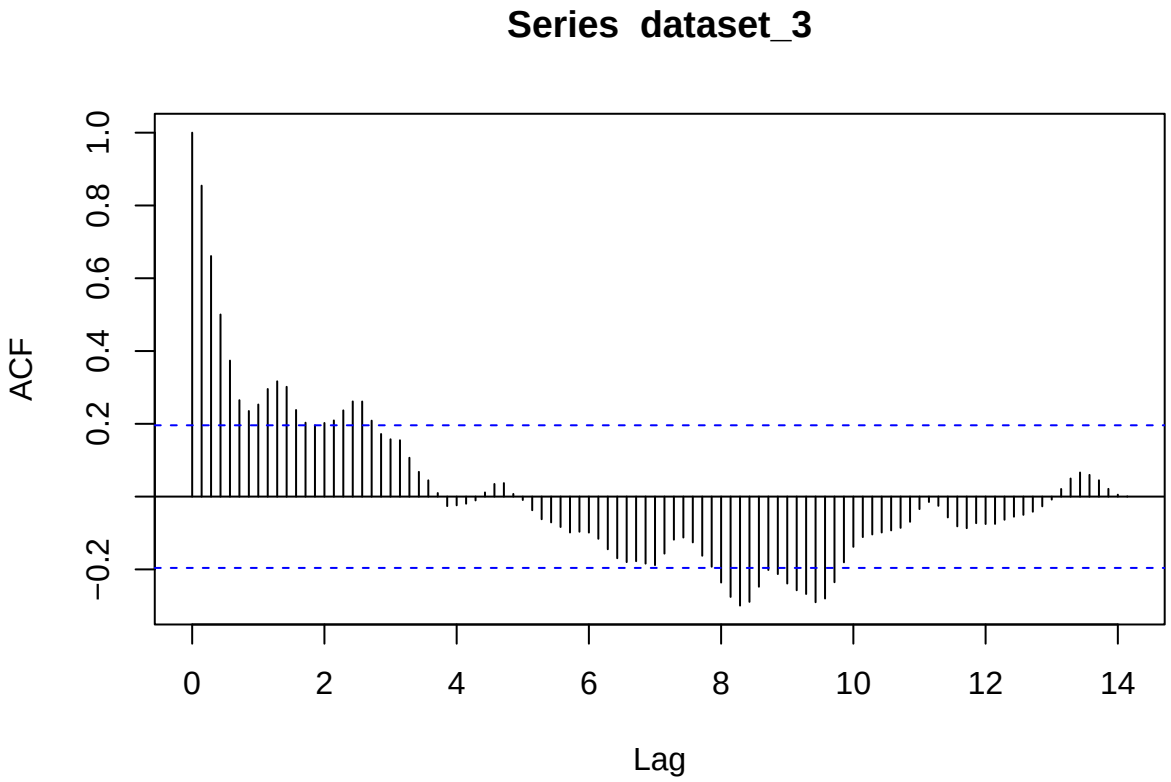
(1 point) Dataset Three

```
dataset_3 %>% ggplot(aes(x=date, y = y)) + geom_line() + xlab("Date") + ggtitle("Dataset 3 Time Series"
```



The time series above is not stationary since the mean is influcencd by large and small fluctuations throughout the plot. Since there are small peaks and larger peaks that alternate with each other between the beginning of December and the end of February, the time series has seasonality.

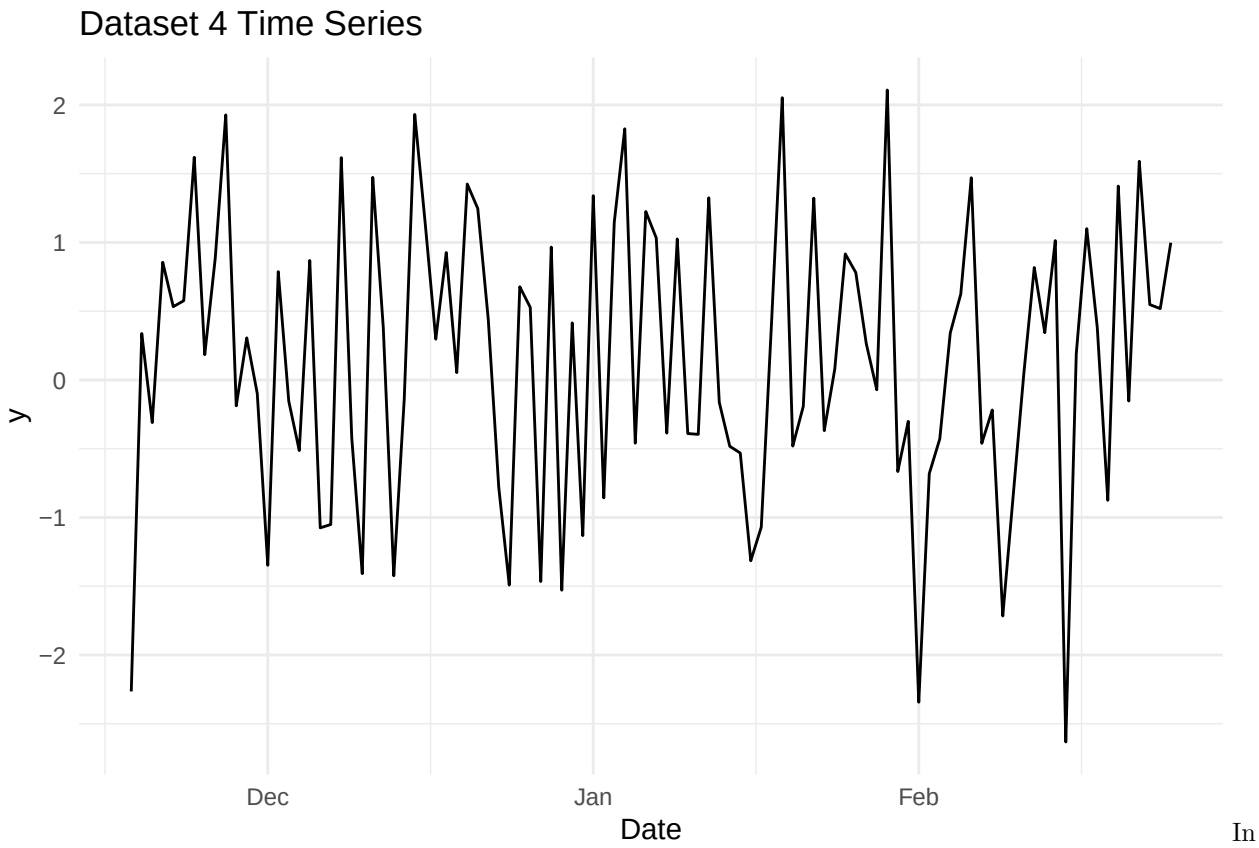
```
dataset_3_acf <- acf(dataset_3, lag.max = 100)
```



The ACF plot shows positive correlations with the first several lags and little to no correlations with the later lags.

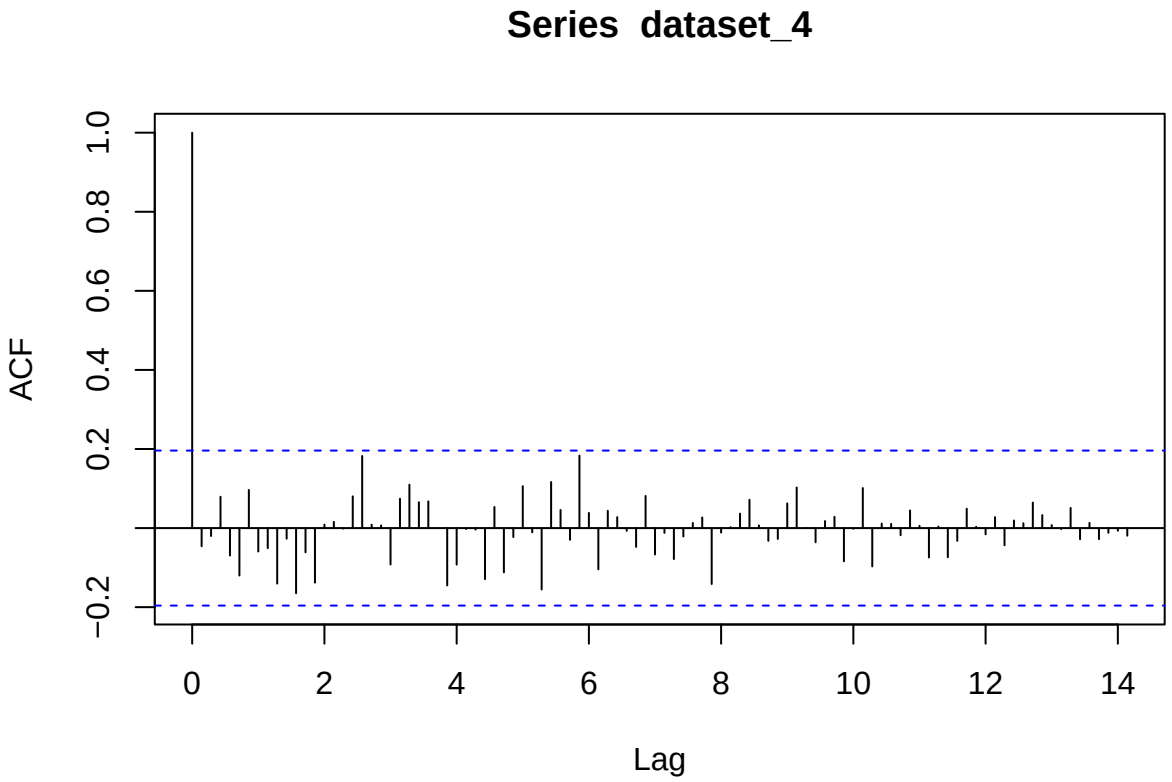
(1 point) Dataset Four

```
dataset_4 %>% ggplot(aes(x=date, y = y)) + geom_line() + xlab("Date") + ggtitle("Dataset 4 Time Series"
```



In the time series above, the mean hovers around 0 with no visible trend. However, the fluctuations increase in magnitude toward the middle of January and onwards. Additionally, there appears to be large dips following large peaks that occur at approximately regular intervals. Hence, there's seasonality.

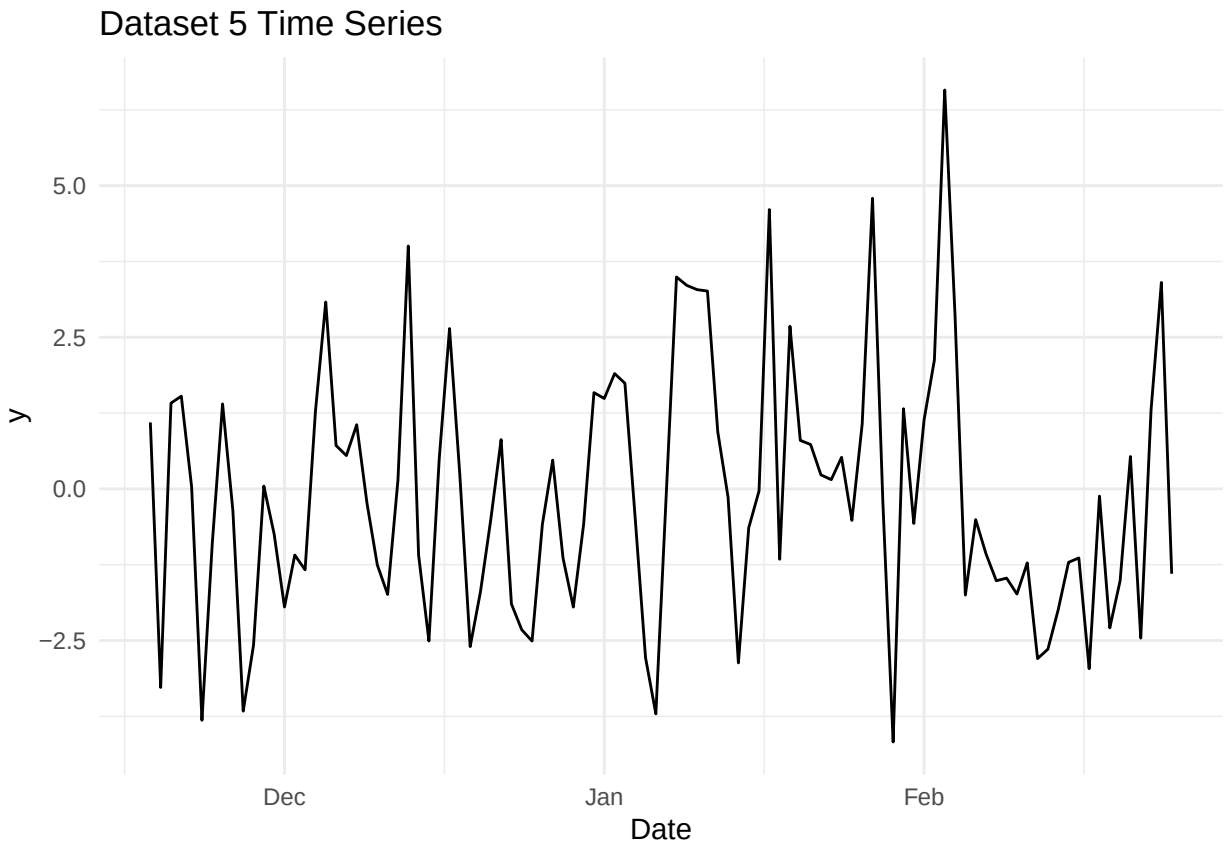
```
dataset_4_acf <- acf(dataset_4, lag.max = 100)
```



Since the ACF plot shows the correlations with the lags falling underneath the blue dotted region, there is no evidence that there are correlations with the lags.

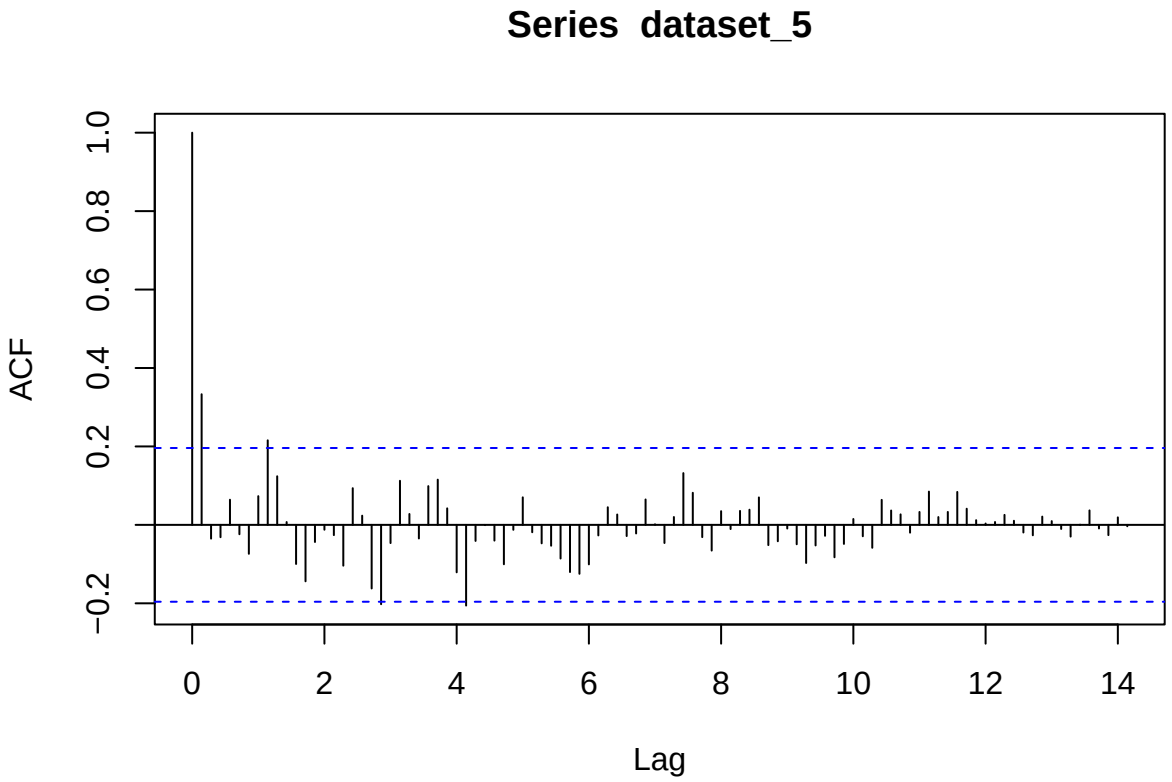
(1 point) Dataset Five

```
dataset_5 %>% ggplot(aes(x=date, y = y)) + geom_line() + xlab("Date") + ggtitle("Dataset 5 Time Series")
```



The plot above shows a non-stationary time series, as the mean fluctuates from December to February. Upon closer examination of the time series, the mean varies at a regular seasonal pattern. Despite this seasonality of the mean varying up and down, there is no general upwards or downwards trend.

```
dataset_5_acf <- acf(dataset_5, lag.max = 100)
```



The ACF plot shows that little to no correlations exist between the series observations and their lags.

Question 4 - BLS Data

This is the last exercise for this assignment. Here, we're going to do the same work that you have done twice before, but against "live" data that comes from the United States' Bureau of Labor Statistics.

Recall that in the lecture, Jeffrey identifies the unemployment rate as an example of a time series. You can get to this data from the public web-site. To do so, head here:

- www.bls.gov > Data Tools > BLS Popular Series
- Then check the box for `Unemployment Rate (Seasonally Adjusted)` and `Retreive Data`. Take note when you check the `Unemployment Rate (Seasonally Adjusted)`, what is the series number that is associated with this?

What do you see when you get to the next page? A rectangular data series that has months on the columns, years on the rows, and values as the internals to the cells? :facepalm:

- Does this meet the requirements of tidy data, or time series tidy data?
- If you were to build an analytic pipeline against data that you accessed in this way, what would be the process to update your analysis when the next edition of data is released? Would it require a manual download, then cleaning, then movement into your analysis? Could this be problematic?

This motivates the idea of using the BLS' data API. The data API provides consistently formatted JSON objects that can be converted to data of an arbitrary (that is, useful to us) formatting. Because the data is being provided in a JSON object, there is some work to coerce it to be useful, but we'll find that there are so many people who are doing this same coercion that there are ready-made wrappers that will help us to do this work.

As an example, you can view how these JSON objects are formatted by navigating to an API endpoint in your browser. Here is the endpoint for the national unemployment: [link].

Let's pull unemployment from the BLS data API.

1. Register for an API key with the BLS. You can register for this from the BLS' "Getting Started" page. They will then send you an API key to the email that you affiliate.
2. Find the series that we want to access. Frankly, this is part of accessing this API that is the most surprisingly difficult – the BLS does not publish a list of the data series. From their Data Retrieval Tools page there are links to popular series, a table lookup, and a Data Finder. Elsewhere they provide pages that describe how series IDs are formatted, but finding series still requires considerable meta-knowledge.

For this assignment, consider the following three series:

1. Total unemployment: `LNS14000000`
2. Male unemployment: `LNS14000001`
3. Female unemployment: `LNS14000002`

Our goal is to analyze these three series for the last 20 years.

To articulate the BLS API, we have found the `blsR` library to be the most effective (at the time that we wrote the assignment in 2022). Here are links to get you read into the package. Rather than providing you with a *full* walk-through for how to use this package to manipulate the BLS data API, instead a learning goal is for you to read these documents and come to an understanding of how the package works.

- CRAN Homepage
- GitHub
- Vignette (Called, incorrectly a README on the CRAN page)

(2 points) Form a successful query and tidy of data

Your task is to create an object called `unemployment` that is a `tsibble` class, that contains the overall unemployment rate, as well as the unemployment rate for male and female people.

Your target dataframe should have the following shape but extend to the current time period.

```
year month time_index name    value
   <int> <int>      <mth> <chr>   <dbl>
 1  2000     1   2000 Jan overall   4
 2  2000     1   2000 Jan male      3.9
 3  2000     1   2000 Jan female    4.1
 4  2000     2   2000 Feb overall   4.1
 5  2000     2   2000 Feb male      4.1
 6  2000     2   2000 Feb female    4.1
 7  2000     3   2000 Mar overall   4
 8  2000     3   2000 Mar male      3.8
 9  2000     3   2000 Mar female    4.3
10  2000     4   2000 Apr overall   3.8
```

```
install.packages("blsR")
```

```
## Installing package into '/usr/local/lib/R/site-library'
## (as 'lib' is unspecified)
```

```
library(blsR)

overall_employment <- get_series_table('LNS14000000', start_year = 2000, end_year = 2000)
male_employment <- get_series_table('LNS14000001', start_year = 2000, end_year = 2000)
female_employment <-get_series_table('LNS14000002', start_year = 2000, end_year = 2000)
```

```
temp_overall_employment <- overall_employment %>% mutate(time_index = yearmonth(paste(year, periodName)))
temp_overall_employment
```

```
## # A tibble: 12 x 5
```

```
##     year period periodName value time_index
##    <int> <chr>  <chr>      <dbl>      <mth>
##  1  2000 M12    December     3.9   2000 Dec
##  2  2000 M11    November     3.9   2000 Nov
##  3  2000 M10    October      3.9   2000 Oct
##  4  2000 M09    September    3.9   2000 Sep
##  5  2000 M08    August       4.1   2000 Aug
##  6  2000 M07    July         4     2000 Jul
##  7  2000 M06    June         4     2000 Jun
##  8  2000 M05    May          4     2000 May
##  9  2000 M04    April        3.8   2000 Apr
## 10  2000 M03    March        4     2000 Mar
## 11  2000 M02    February     4.1   2000 Feb
## 12  2000 M01    January      4     2000 Jan
```

```
temp_overall_employment$month <- as.integer(factor(temp_overall_employment$periodName, levels = month.na
temp_overall_employment
```

```
## # A tibble: 12 x 6
##     year period periodName value time_index month
##    <int> <chr>  <chr>      <dbl>      <mth> <int>
##  1  2000 M12    December     3.9   2000 Dec    12
##  2  2000 M11    November     3.9   2000 Nov    11
##  3  2000 M10    October      3.9   2000 Oct    10
##  4  2000 M09    September    3.9   2000 Sep     9
##  5  2000 M08    August       4.1   2000 Aug     8
##  6  2000 M07    July         4     2000 Jul     7
##  7  2000 M06    June         4     2000 Jun     6
##  8  2000 M05    May          4     2000 May     5
##  9  2000 M04    April        3.8   2000 Apr     4
## 10  2000 M03    March        4     2000 Mar     3
## 11  2000 M02    February     4.1   2000 Feb     2
## 12  2000 M01    January      4     2000 Jan     1
```

```
temp_overall_employment$name <- rep("overall", times = 12)
temp_overall_employment <- temp_overall_employment %>% select(year, month, time_index, name, value)
temp_overall_employment
```

```
## # A tibble: 12 x 5
##     year month time_index name    value
##    <int> <int>      <mth> <chr>   <dbl>
##  1  2000    12   2000 Dec overall   3.9
##  2  2000    11   2000 Nov overall   3.9
##  3  2000    10   2000 Oct overall   3.9
##  4  2000     9   2000 Sep overall   3.9
##  5  2000     8   2000 Aug overall   4.1
##  6  2000     7   2000 Jul overall   4
##  7  2000     6   2000 Jun overall   4
##  8  2000     5   2000 May overall   4
##  9  2000     4   2000 Apr overall   3.8
## 10  2000     3   2000 Mar overall   4
## 11  2000     2   2000 Feb overall   4.1
## 12  2000     1   2000 Jan overall   4
```

```
temp_male_employment <- male_employment %>% mutate(time_index = yearmonth(paste(year, periodName)))
temp_male_employment
```

```
## # A tibble: 12 x 5
##     year period periodName value time_index
##    <int> <chr>  <chr>      <dbl>      <mth>
##  1  2000 M12    December     4     2000 Dec
##  2  2000 M11    November     3.9   2000 Nov
##  3  2000 M10    October      3.9   2000 Oct
##  4  2000 M09    September    3.9   2000 Sep
##  5  2000 M08    August       3.9   2000 Aug
##  6  2000 M07    July         3.9   2000 Jul
##  7  2000 M06    June         3.8   2000 Jun
##  8  2000 M05    May          3.9   2000 May
##  9  2000 M04    April        3.7   2000 Apr
## 10  2000 M03    March        3.8   2000 Mar
## 11  2000 M02    February     4.1   2000 Feb
## 12  2000 M01    January      3.9   2000 Jan
```

```
temp_male_employment$month <- as.integer(factor(temp_male_employment$periodName, levels = month.name))
temp_male_employment
```

```
## # A tibble: 12 x 6
##     year period periodName value time_index month
##    <int> <chr>  <chr>      <dbl>      <mth> <int>
##  1  2000 M12    December     4     2000 Dec    12
##  2  2000 M11    November     3.9   2000 Nov    11
##  3  2000 M10    October      3.9   2000 Oct    10
##  4  2000 M09    September    3.9   2000 Sep     9
##  5  2000 M08    August       3.9   2000 Aug     8
##  6  2000 M07    July         3.9   2000 Jul     7
##  7  2000 M06    June         3.8   2000 Jun     6
##  8  2000 M05    May          3.9   2000 May     5
##  9  2000 M04    April        3.7   2000 Apr     4
## 10  2000 M03    March        3.8   2000 Mar     3
## 11  2000 M02    February     4.1   2000 Feb     2
## 12  2000 M01    January      3.9   2000 Jan     1
```

```
temp_male_employment$name <- rep("male", times = 12)
temp_male_employment <- temp_male_employment %>% select(year, month, time_index, name, value)
temp_male_employment
```

```
## # A tibble: 12 x 5
##     year month time_index name  value
##    <int> <int>      <mth> <chr> <dbl>
##  1  2000    12   2000 Dec male    4
##  2  2000    11   2000 Nov male    3.9
##  3  2000    10   2000 Oct male    3.9
##  4  2000     9   2000 Sep male    3.9
##  5  2000     8   2000 Aug male    3.9
##  6  2000     7   2000 Jul male    3.9
##  7  2000     6   2000 Jun male    3.8
##  8  2000     5   2000 May male    3.9
##  9  2000     4   2000 Apr male    3.7
## 10  2000     3   2000 Mar male    3.8
## 11  2000     2   2000 Feb male    4.1
## 12  2000     1   2000 Jan male    3.9
```

```
temp_female_employment <- female_employment %>% mutate(time_index = yearmonth(paste(year, periodName)))
temp_female_employment
```

```
## # A tibble: 12 x 5
##     year period periodName value time_index
##    <int> <chr>  <chr>      <dbl>      <mth>
##  1  2000 M12    December     3.8   2000 Dec
##  2  2000 M11    November     4     2000 Nov
##  3  2000 M10    October      3.9   2000 Oct
##  4  2000 M09    September    4     2000 Sep
##  5  2000 M08    August       4.3   2000 Aug
##  6  2000 M07    July         4.2   2000 Jul
##  7  2000 M06    June         4.1   2000 Jun
##  8  2000 M05    May          4.2   2000 May
##  9  2000 M04    April        4     2000 Apr
## 10  2000 M03    March        4.3   2000 Mar
## 11  2000 M02    February     4.1   2000 Feb
## 12  2000 M01    January      4.1   2000 Jan
```

```
temp_female_employment$month <- as.integer(factor(temp_female_employment$periodName, levels = month.name
temp_female_employment
```

```
## # A tibble: 12 x 6
##     year period periodName value time_index month
##    <int> <chr>  <chr>      <dbl>      <mth> <int>
##  1  2000 M12    December     3.8   2000 Dec    12
##  2  2000 M11    November     4     2000 Nov    11
##  3  2000 M10    October      3.9   2000 Oct    10
##  4  2000 M09    September    4     2000 Sep     9
##  5  2000 M08    August       4.3   2000 Aug     8
##  6  2000 M07    July         4.2   2000 Jul     7
##  7  2000 M06    June         4.1   2000 Jun     6
##  8  2000 M05    May          4.2   2000 May     5
##  9  2000 M04    April        4     2000 Apr     4
## 10  2000 M03    March        4.3   2000 Mar     3
## 11  2000 M02    February     4.1   2000 Feb     2
## 12  2000 M01    January      4.1   2000 Jan     1
```

```
temp_female_employment$name <- rep("female", times = 12)
temp_female_employment <- temp_female_employment %>% select(year, month, time_index, name, value)
temp_female_employment
```

```
## # A tibble: 12 x 5
##     year month time_index name   value
##    <int> <int>      <mth> <chr>  <dbl>
##  1  2000    12   2000 Dec female   3.8
##  2  2000    11   2000 Nov female   4
##  3  2000    10   2000 Oct female   3.9
##  4  2000     9   2000 Sep female   4
##  5  2000     8   2000 Aug female   4.3
##  6  2000     7   2000 Jul female   4.2
##  7  2000     6   2000 Jun female   4.1
##  8  2000     5   2000 May female   4.2
##  9  2000     4   2000 Apr female   4
## 10  2000     3   2000 Mar female   4.3
```

```
## 11  2000     2   2000 Feb female   4.1
## 12  2000     1   2000 Jan female   4.1
```

```
unemployment <- rbind(temp_overall_employment, temp_male_employment, temp_female_employment) %>% arrang
unemployment <- unemployment %>% as_tsibble(index=time_index, key = c(year, month, name))
```

```
head(unemployment)
```

```
## # A tsibble: 6 x 5 [1M]
## # Key:       year, month, name [6]
##    year month time_index name    value
##   <int> <int>      <mth> <chr>   <dbl>
## 1  2000     1   2000 Jan female    4.1
## 2  2000     1   2000 Jan male      3.9
## 3  2000     1   2000 Jan overall   4
## 4  2000     2   2000 Feb female    4.1
## 5  2000     2   2000 Feb male      4.1
## 6  2000     2   2000 Feb overall   4.1
```

(1 point) Plot the Unemployment Rate

Once you have queried the data and have it successfully stored in an appropriate object, produce a plot that shows the unemployment rate on the y-axis, time on the x-axis, and each of the groups (overall, male, and female) as a different colored line.

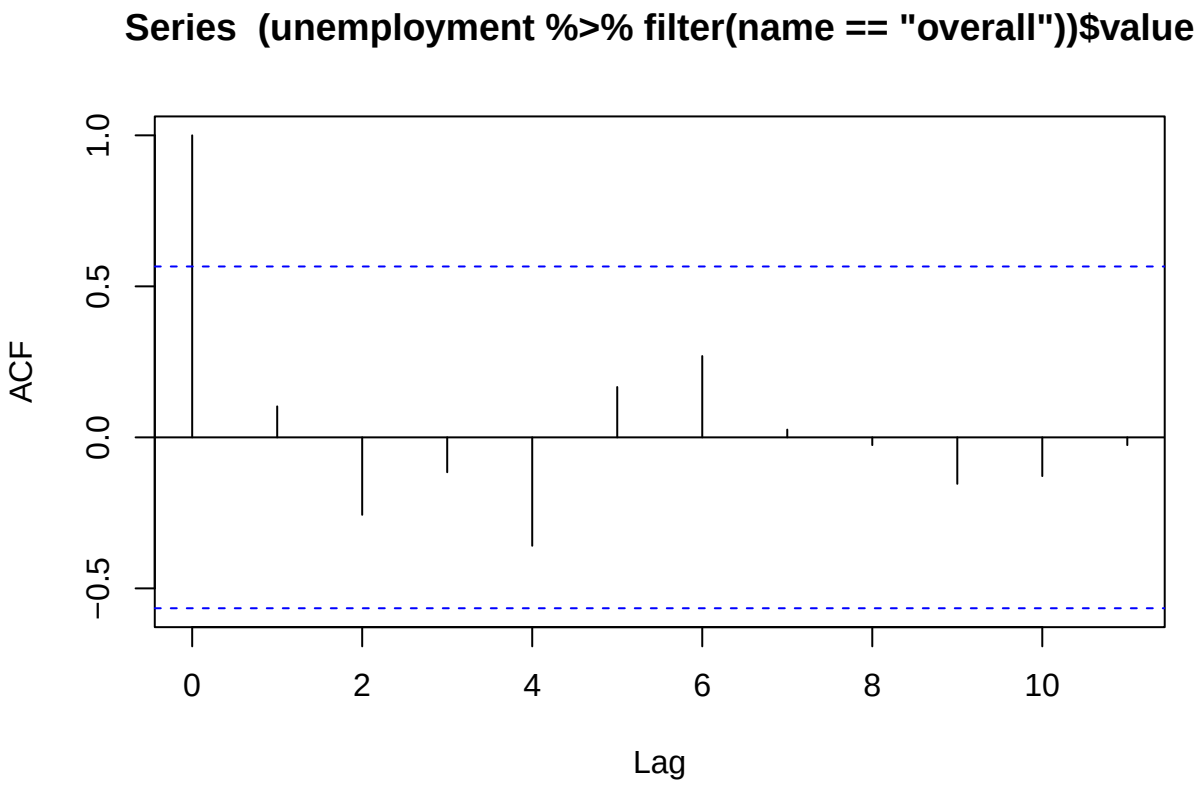
```
unemployment_plot <- unemployment %>% ggplot(aes(x=time_index, y = value, color=name)) + geom_line() + 
unemployment_plot
```

(1 point) Plot the ACF and Lags

This should feel familiar by now: Produce the ACF and lag plot of the `overall` unemployment series. What do you observe?

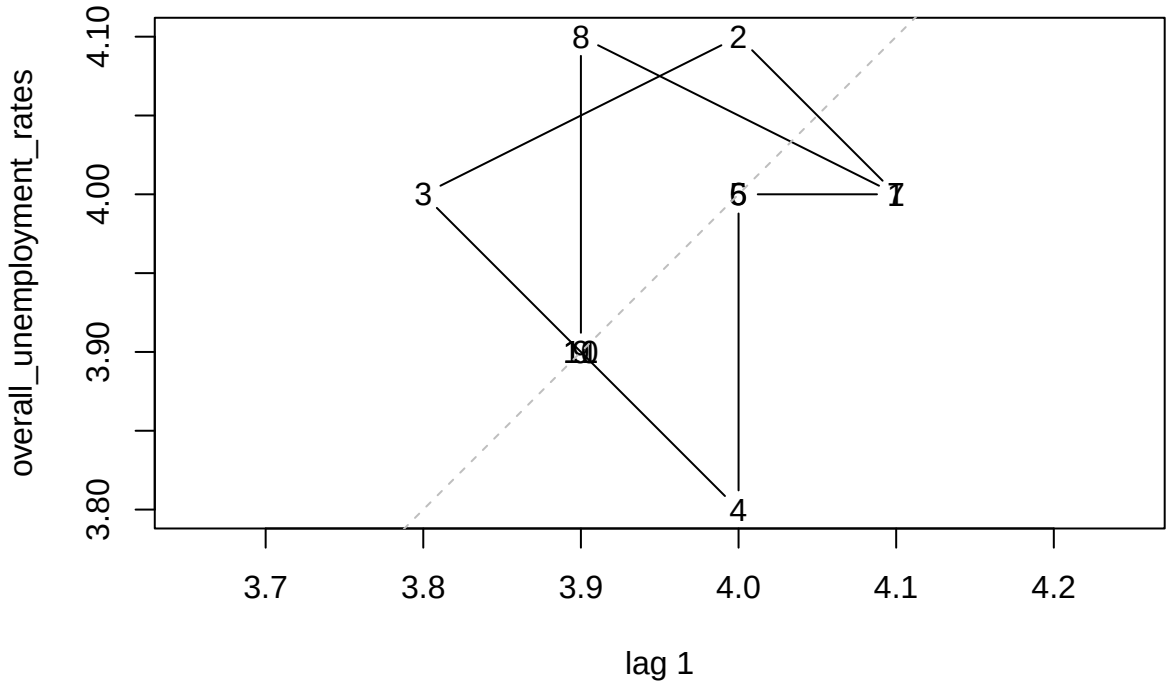
Based on the plot above, the unemployment rate among females is generally higher than that of males throughout all of the year 2000. However, starting in the mid-summer and continuing into the fall months, the female unemployment rate trends downward. The highest unemployment rate among males in the year 2000 occurs between late-January and early February, and the lowest unemployment rate among males in the year 2000 occurs near the start of April.

```
acf((unemployment %>% filter(name == "overall"))$value, lag.max = 30)
```



Based on the ACF plot above, there is no correlation between the series observations and their lags, indicating that there is no observed correlation between the unemployment rate at one point in time and a previous point in time within the time series.

```
overall_unemployment_rates <- (unemployment %>% filter(name == "overall"))$value
lag.plot(overall_unemployment_rates)
```



From the lag plot above, there is no linear pattern that indicates a positive or negative correlation with the overall unemployment rate and the first lag.