STAT 479 HW3

Problems

```
#load some packages and libraries
library("dslabs")
library("dplyr")
library("tidyr")
library("ggplot2")
library("naniar")
library("UpSetR")
library("rpart")
library("simputation")
library("readr")
library("tsibble")
library("tsibbledata")
library("lubridate")
library("fpp2")
library("feasts")
library("stringr")
library("grid")
library("gridExtra")
library("ggmap")
library("ggrepel")
library("sf")
library("raster")
library("RStoolbox")
library("scico")
library("ggraph")
library("tidygraph")
library("networkD3")
library("igraph")
library("ggnetwork")
theme_set(theme_graph())
theme_set(theme_minimal())
theme_set(theme_bw())
```

Matching Autocorrelation Functions

The purpose of this problem is to build further intuition about auto-correlation. We'll simulate data with known structure and then see what happens to the associated autocorrelation functions.

a. The code below simulates a sinusoidal pattern over the course of 2020. Extend the code so that `date` and `y` are contained in a tsibble object.

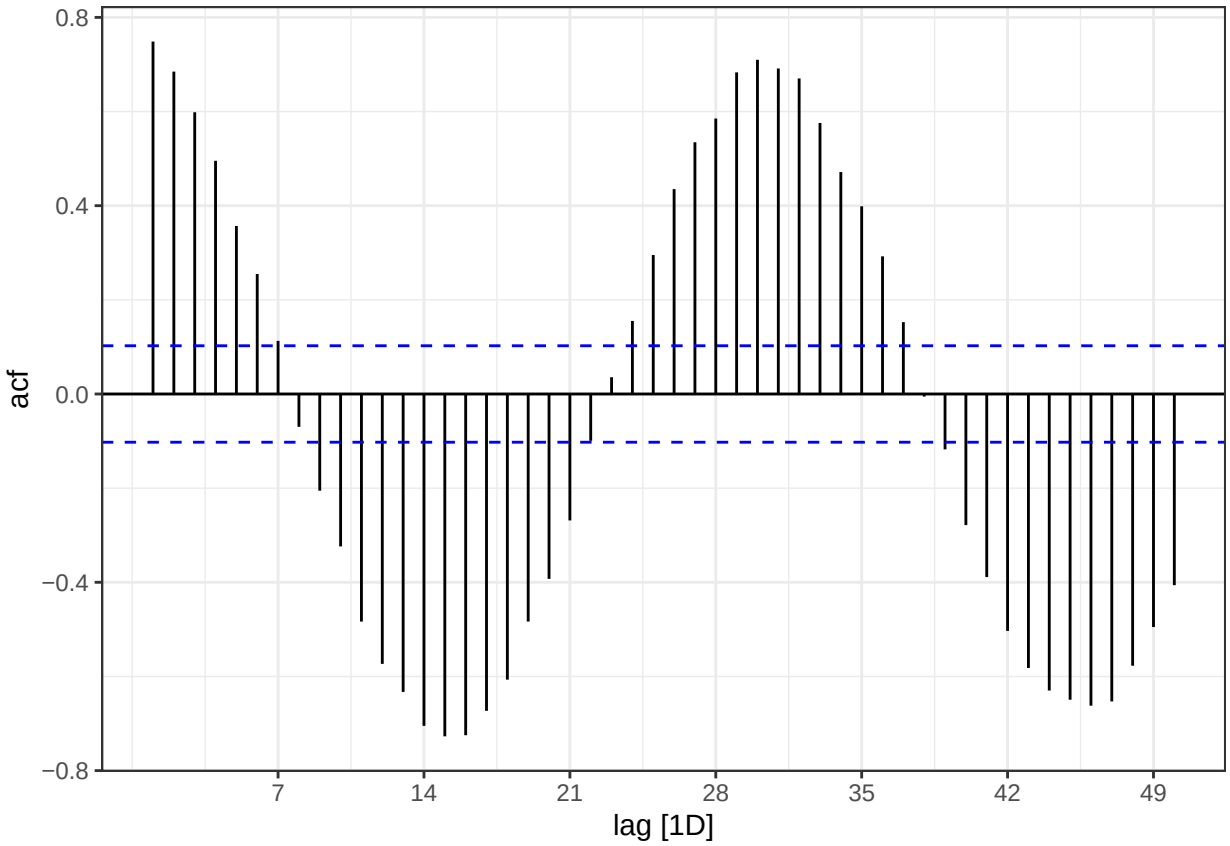
```
date <- seq(from = as_date("2020-01-01"), to = as_date("2020-12-31"), by = 1)
x <- seq(0, 12 * 2 * pi, length.out = length(date))
y <- sin(x) + rnorm(length(x), 0, .4)
data <- tsibble(day = date, value = y, index = day)
```

b. Using the tsibble object, calculate and visualize the induced autocorrelation function. Use a maximum lag of 50, and interpret the resulting plot.

```
acf_data <- ACF(data, lag_max = 50)
acf_data
```

```
## # A tsibble: 50 x 2 [1D]
##      lag     acf
##    <lag>   <dbl>
##  1    1D  0.749
##  2    2D  0.685
##  3    3D  0.598
##  4    4D  0.495
##  5    5D  0.357
##  6    6D  0.255
##  7    7D  0.113
##  8    8D -0.0697
##  9    9D -0.205
## 10   10D -0.324
## # ... with 40 more rows
```

```
autoplot(acf_data)
```



Interpretation:

By observing the plot above, we found that within 50 days, acf reaches a peak every 30 days. There will be around 15 days between a peak and a valley.

c. Write a function to simulate a version of the tsibble above, but with a linear trend from 0 to **z**, where **z**

is an argument to the function.

```
f <- function(z) {
  date <- seq(from = as_date("2020-01-01"), to = as_date("2020-12-31"), by = 1)
  x <- seq(0, 12 * 2 * pi, length.out = length(date))
  y <- sin(x) + rnorm(length(x), 0, .4) + z*x
  data <- tsibble(day = date, value = y, index = day)
  return(data)
}
```

d. Using the function from (c), generate 5 datasets with linear trends of varying magnitudes. Plot the associated autocorrelation functions and comment on the relationship between the strength of the trend and the shape of the function.

```
#generate 5 datasets
d1 <- f(1)
d2 <- f(5)
d3 <- f(50)
d4 <- f(100)
d5 <- f(500)
d1;d2;d3;d4;d5
```

```
## # A tsibble: 366 x 2 [1D]
##    day        value
##    <date>     <dbl>
##  1 2020-01-01 0.360
##  2 2020-01-02 0.365
##  3 2020-01-03 0.733
##  4 2020-01-04 1.21
##  5 2020-01-05 1.21
##  6 2020-01-06 2.09
##  7 2020-01-07 1.41
##  8 2020-01-08 2.20
##  9 2020-01-09 2.75
## 10 2020-01-10 2.43
## # ... with 356 more rows
```

```
## # A tsibble: 366 x 2 [1D]
##    day         value
##    <date>      <dbl>
##  1 2020-01-01  0.462
##  2 2020-01-02  1.12
##  3 2020-01-03  2.05
##  4 2020-01-04  3.54
##  5 2020-01-05  5.12
##  6 2020-01-06  5.71
##  7 2020-01-07  7.01
##  8 2020-01-08  8.49
##  9 2020-01-09  9.58
## 10 2020-01-10 10.1
## # ... with 356 more rows
```

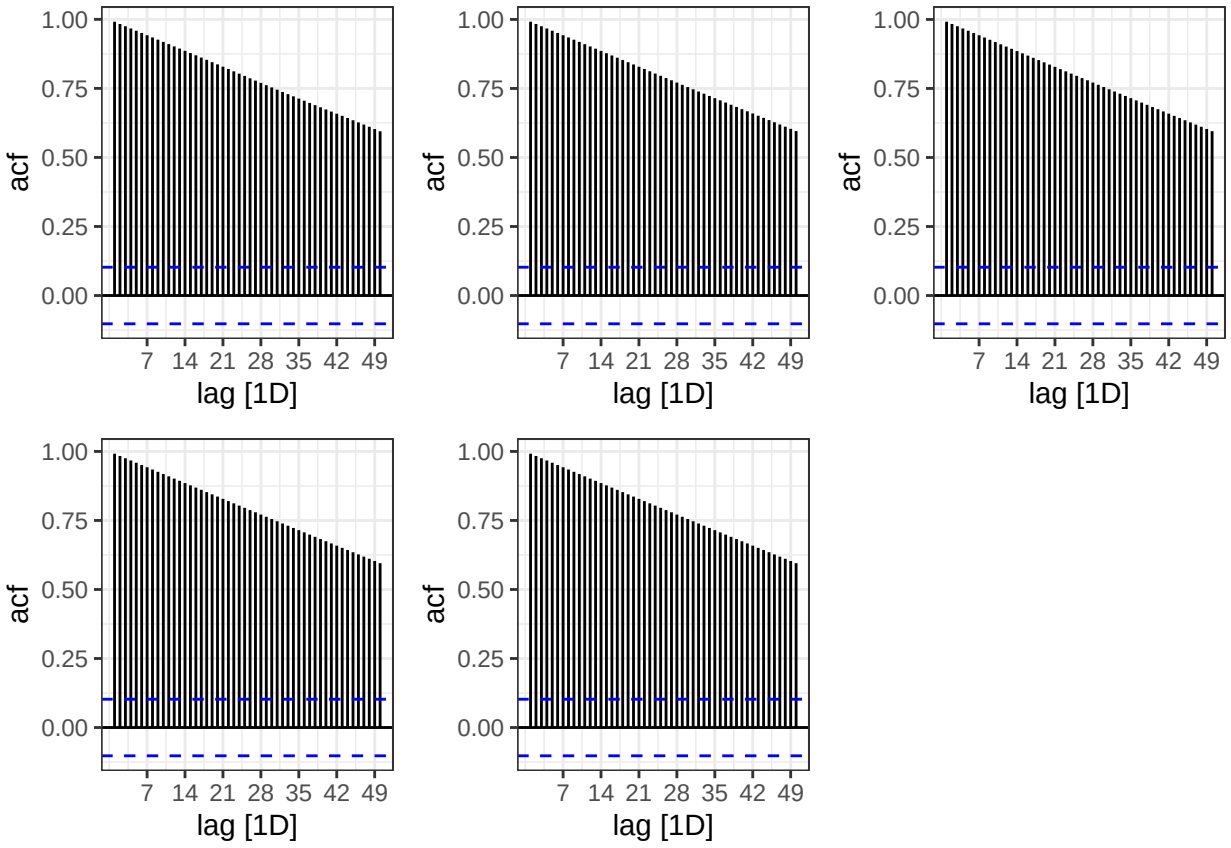
```
## # A tsibble: 366 x 2 [1D]
##    day         value
##    <date>      <dbl>
##  1 2020-01-01  0.682
##  2 2020-01-02 10.7
```

```
##  3 2020-01-03 21.4
##  4 2020-01-04 32.2
##  5 2020-01-05 42.5
##  6 2020-01-06 52.7
##  7 2020-01-07 62.8
##  8 2020-01-08 73.6
##  9 2020-01-09 83.8
## 10 2020-01-10 94.0
## # ... with 356 more rows
```

```
## # A tsibble: 366 x 2 [1D]
##    day          value
##    <date>       <dbl>
##  1 2020-01-01  -0.126
##  2 2020-01-02  20.4
##  3 2020-01-03  42.2
##  4 2020-01-04  62.9
##  5 2020-01-05  83.2
##  6 2020-01-06 103.
##  7 2020-01-07 125.
##  8 2020-01-08 146.
##  9 2020-01-09 166.
## 10 2020-01-10 187.
## # ... with 356 more rows
```

```
## # A tsibble: 366 x 2 [1D]
##    day          value
##    <date>       <dbl>
##  1 2020-01-01  -0.383
##  2 2020-01-02 104.
##  3 2020-01-03 207.
##  4 2020-01-04 310.
##  5 2020-01-05 413.
##  6 2020-01-06 517.
##  7 2020-01-07 621.
##  8 2020-01-08 724.
##  9 2020-01-09 827.
## 10 2020-01-10 930.
## # ... with 356 more rows
```

```
#plot
p1 <- ACF(d1, lag_max = 50) %>%
  autoplot()
p2 <- ACF(d2, lag_max = 50) %>%
  autoplot()
p3 <- ACF(d3, lag_max = 50) %>%
  autoplot()
p4 <- ACF(d4, lag_max = 50) %>%
  autoplot()
p5 <- ACF(d5, lag_max = 50) %>%
  autoplot()
grid.arrange(p1, p2, p3, p4, p5, nrow = 2)
```



Comment: We found the acf trend within 50 days is pretty similar.

Spotify Time Series

In this problem, we will study music streaming on Spotify in 2017. We'll start by looking at some characteristics of the most streamed song, and then will practice how to extract features from across the collection of most streamed songs.

a. Let's look at the most streamed song of 2017, which was "Shape of You." The dataset here contains the number of streams for this song across regions, for each day in which it was in the Spotify 100 most streamed songs for that region. Create a `tsibble` object from this dataset, keying by `region` and indexing by `date`.

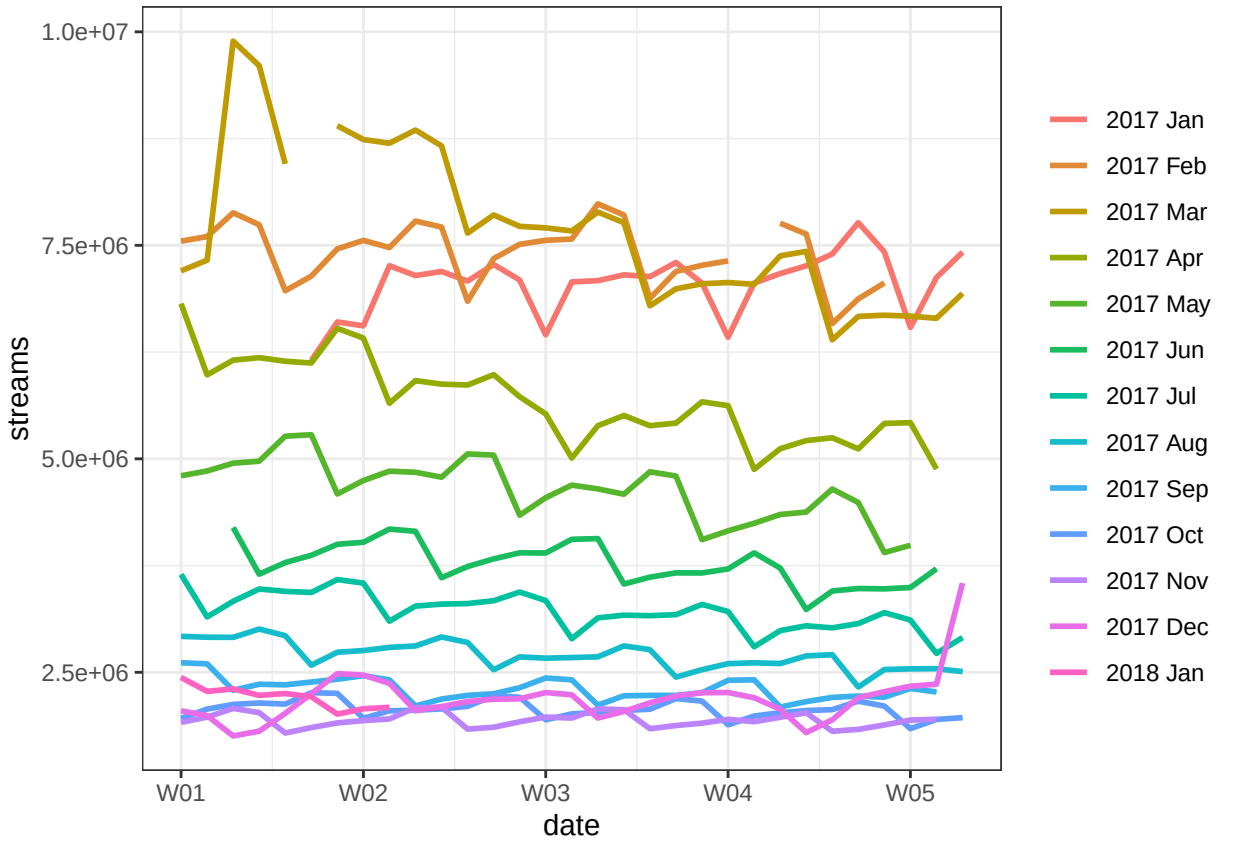
```
# read the csv
spotify <- read.csv("https://uwmadison.box.com/shared/static/hvplyr3jy6vbt7s80lqgfx81ai4hdl0q.csv")

#create a tsibble object
spotify_t <- spotify %>%
  mutate(date = as_date(date)) %>%
  as_tsibble(index = date, key = region)

spotify_t
```

```
## # A tsibble: 19,907 x 5 [1D]
## # Key:       region [54]
##    artist     track_name   region date       streams
##    <chr>      <chr>        <chr>  <date>       <int>
##  1 Ed Sheeran Shape of You ar     2017-01-06   57617
```

```
##  2 Ed Sheeran Shape of You ar     2017-01-07   51212
##  3 Ed Sheeran Shape of You ar     2017-01-08   47299
##  4 Ed Sheeran Shape of You ar     2017-01-09   58157
##  5 Ed Sheeran Shape of You ar     2017-01-10   56384
##  6 Ed Sheeran Shape of You ar     2017-01-11   59494
##  7 Ed Sheeran Shape of You ar     2017-01-12   59565
##  8 Ed Sheeran Shape of You ar     2017-01-13   64899
##  9 Ed Sheeran Shape of You ar     2017-01-14   71555
## 10 Ed Sheeran Shape of You ar     2017-01-15   62253
## # ... with 19,897 more rows
```

b. Filter to `region == "global"`, and make a `gg_season` plot by month. Comment on the what you see.

```
spotify_t %>%
  filter(region == "global") %>%
  gg_season(streams, period = "month") +
  geom_line(size = 1)
```

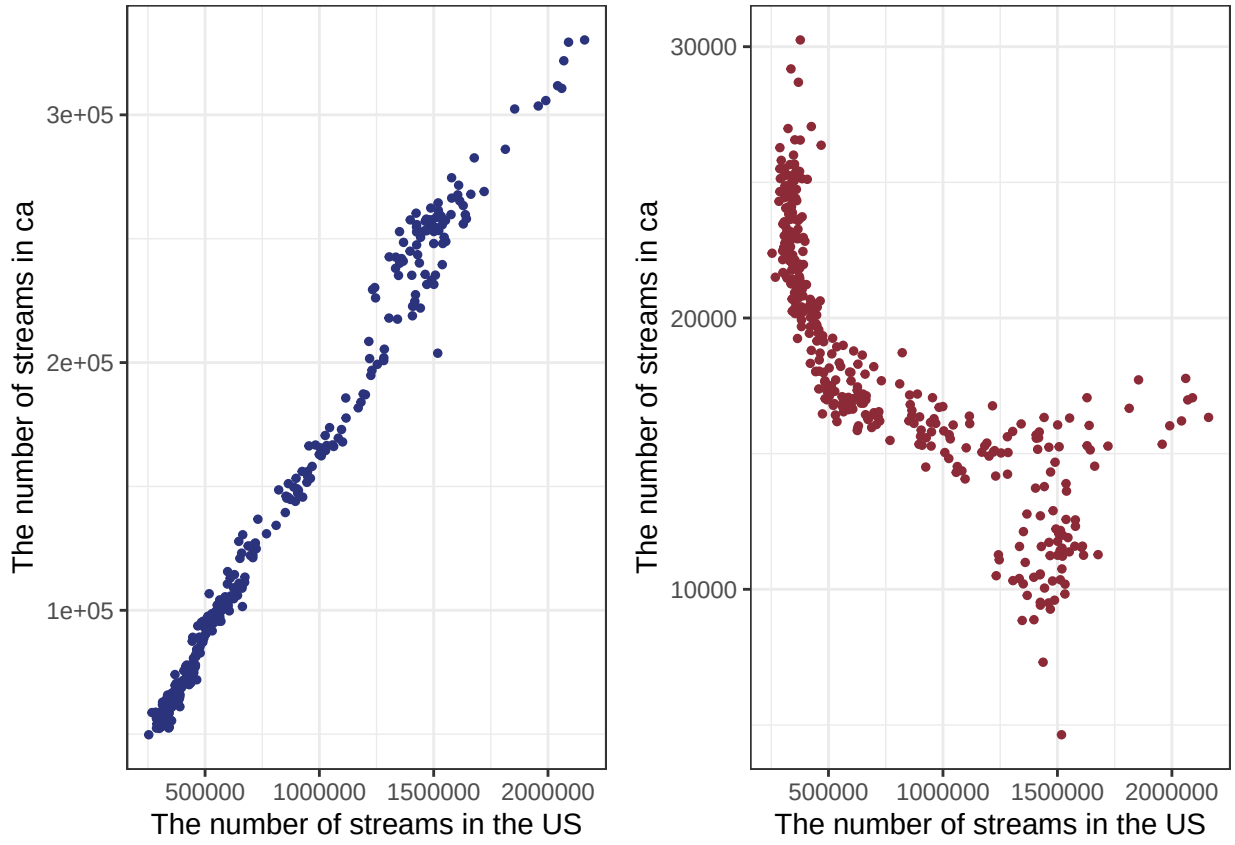


Comment:

1. In the first three months, 2017 Jan, 2017 Feb, and 2017 Mar, the streams are relatively high.
2. The streams decreased monthly.
3. From 2017 Oct, the streams tend to be stable, which is a little bit lower than 2500000.

c. Provide a scatterplot showing the relationship between the number of streams of this song in the US and in Canada. Do the same between the US and Japan. Briefly comment. **Hint**: Use `pivot_wider` to spread the time series for each region across columns of a reshaped dataset.

```r
# spread the dataset
spotify_spread <- spotify_t %>%
  pivot_wider(names_from = region, values_from = streams)

# make a plot showing the relationship between the number of streams in the US and in Canada
p1 <- ggplot(spotify_spread, aes(x = us, y = ca)) +
  geom_point(col = "#2b337c", size = 1) +
  labs(
    x = "The number of streams in the US",
    y = "The number of streams in ca"
  )

# make a plot showing the relationship between the number of streams in the US and in Japan
p2 <- ggplot(spotify_spread, aes(x = us, y = jp)) +
  geom_point(col = "#8C2B37", size = 1) +
  labs(
    x = "The number of streams in the US",
    y = "The number of streams in ca"
  )

grid.arrange(p1, p2, ncol = 2)
```



Comment:

From the first plots, we see there is a linear relationship between the number of streams in US and the number of streams in ca, which is positive. From the second plots, we see the graph is pretty similar to a parabola, which refers to a quadratic relationship between the number of streams in us and the number of

streams in jp.

d. The dataset here contains similar data, but for all songs that appeared in the Spotify 100 for at least 200 days in 2017. We have filtered to only the global totals. Read these data into a tibble, keyed by `artist:region` and extract features of the `streams` time series using the `features` function in the feasts library. It is normal to see a few errors reported by this function, it just means that some of the statistics could not be calculated.

```
# read the csv
spotify_all <- read.csv("https://uwmadison.box.com/shared/static/xj4vupjbicw6c8tbhuynw0pll6yh1w0d.csv")

#create a tsibble object
spotify_all_t <- spotify_all %>%
  mutate(date = as_date(date)) %>%
  as_tsibble(index = date, key = artist:region) %>%
  features(streams, feature_set(pkgs = "feasts"))
```

```
## Error in ar.burg.default(x, aic = aic, order.max = order.max, na.action = na.action,  :
##   'order.max' must be >= 1
## Error in ar.burg.default(x, aic = aic, order.max = order.max, na.action = na.action,  :
##   'order.max' must be >= 1
```

```
spotify_all_t
```

```
## # A tibble: 121 x 52
##    artist track_name region trend_strength seasonal_streng~ seasonal_peak_w~
##    <chr>  <chr>      <chr>           <dbl>            <dbl>            <dbl>
##  1 "A Bo~ "Drowning~ global          0.970            0.737                1
##  2 "AJR"  "Weak"     global          0.995            0.666                4
##  3 "Alan~ "Faded"    global          0.983            0.739                6
##  4 "Ales~ "How Far ~ global          0.997            0.652                4
##  5 "Alok" "Hear Me ~ global          0.995            0.774                5
##  6 "Anue~ "Sola (Re~ global          0.969            0.801                5
##  7 "Aria~ "Side To ~ global          0.997            0.797                6
##  8 "Axwe~ "More Tha~ global          0.987            0.727                3
##  9 "Big ~ "Bounce B~ global          0.991            0.551                6
## 10 "Brun~ "24K Magi~ global          0.998            0.768                6
## # ... with 111 more rows, and 46 more variables: seasonal_trough_week <dbl>,
## #   spikiness <dbl>, linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>,
## #   stl_e_acf10 <dbl>, acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>,
## #   diff1_acf10 <dbl>, diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>,
## #   pacf5 <dbl>, diff1_pacf5 <dbl>, diff2_pacf5 <dbl>, season_pacf <dbl>,
## #   zero_run_mean <dbl>, nonzero_squared_cv <dbl>, zero_start_prop <dbl>,
## #   zero_end_prop <dbl>, lambda_guerrero <dbl>, kpss_stat <dbl>,
## #   kpss_pvalue <dbl>, pp_stat <dbl>, pp_pvalue <dbl>, ndiffs <int>,
## #   nsdiffs <int>, bp_stat <dbl>, bp_pvalue <dbl>, lb_stat <dbl>,
## #   lb_pvalue <dbl>, var_tiled_var <dbl>, var_tiled_mean <dbl>,
## #   shift_level_max <dbl>, shift_level_index <dbl>, shift_var_max <dbl>,
## #   shift_var_index <dbl>, shift_kl_max <dbl>, shift_kl_index <dbl>,
## #   spectral_entropy <dbl>, n_crossing_points <int>, longest_flat_spot <int>,
## #   coef_hurst <dbl>, stat_arch_lm <dbl>, arch_lm <dbl>
```

e. Which tracks had the highest and lowest `trend_strength`'s? Visualize their streams over the course of the year.

```
# identify tracks had the highese and the lowest 'trend_strength'
spotify_all_t %>%
```

```
  arrange(trend_strength) %>%
  head(1)
```

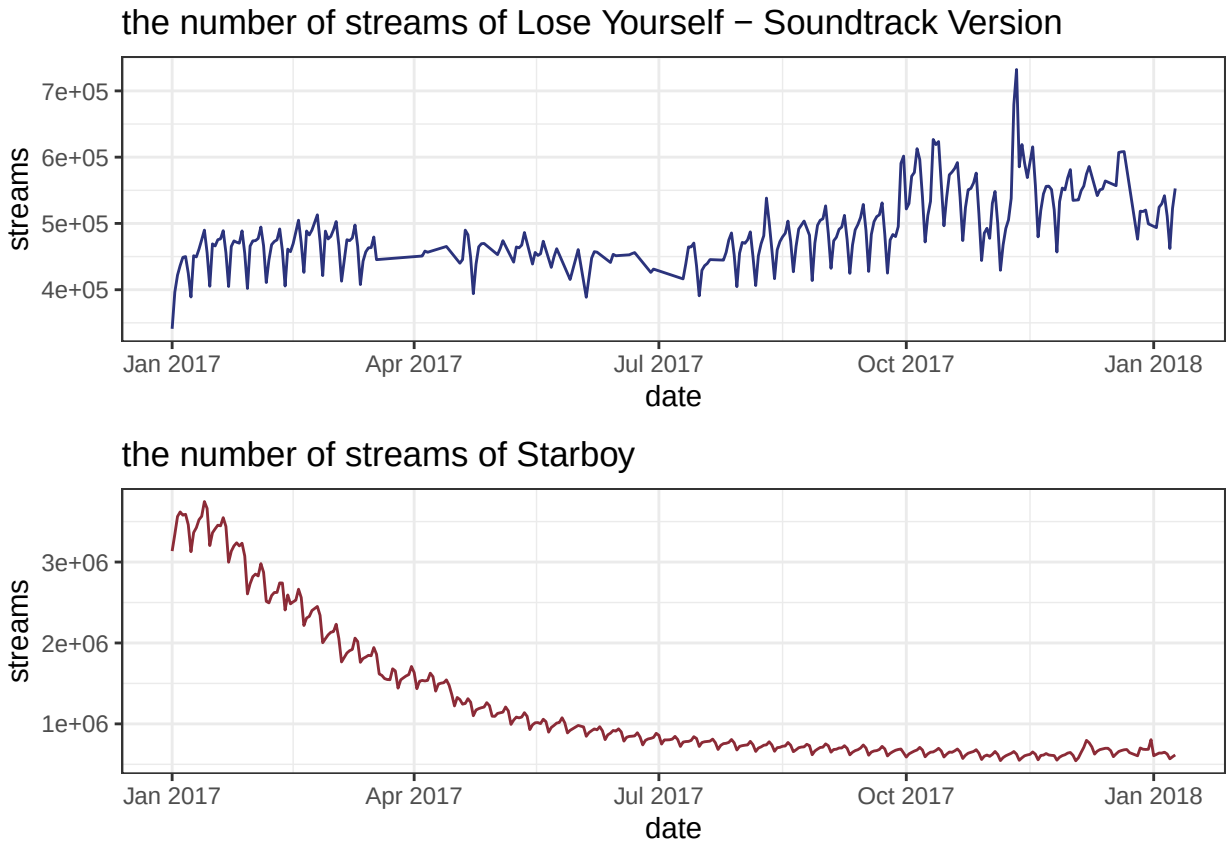
```
## # A tibble: 1 x 52
##   artist track_name region trend_strength seasonal_streng~ seasonal_peak_w~
##   <chr>  <chr>      <chr>           <dbl>            <dbl>            <dbl>
## 1 Eminem Lose Your~ global          0.842            0.572                1
## # ... with 46 more variables: seasonal_trough_week <dbl>, spikiness <dbl>,
## #   linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>,
## #   acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>, pacf5 <dbl>,
## #   diff1_pacf5 <dbl>, diff2_pacf5 <dbl>, season_pacf <dbl>,
## #   zero_run_mean <dbl>, nonzero_squared_cv <dbl>, zero_start_prop <dbl>,
## #   zero_end_prop <dbl>, lambda_guerrero <dbl>, kpss_stat <dbl>,
## #   kpss_pvalue <dbl>, pp_stat <dbl>, pp_pvalue <dbl>, ndiffs <int>,
## #   nsdiffs <int>, bp_stat <dbl>, bp_pvalue <dbl>, lb_stat <dbl>,
## #   lb_pvalue <dbl>, var_tiled_var <dbl>, var_tiled_mean <dbl>,
## #   shift_level_max <dbl>, shift_level_index <dbl>, shift_var_max <dbl>,
## #   shift_var_index <dbl>, shift_kl_max <dbl>, shift_kl_index <dbl>,
## #   spectral_entropy <dbl>, n_crossing_points <int>, longest_flat_spot <int>,
## #   coef_hurst <dbl>, stat_arch_lm <dbl>, arch_lm <dbl>
```

```
spotify_all_t %>%
  arrange(desc(trend_strength)) %>%
  head(1)
```

```
## # A tibble: 1 x 52
##   artist track_name region trend_strength seasonal_streng~ seasonal_peak_w~
##   <chr>  <chr>      <chr>           <dbl>            <dbl>            <dbl>
## 1 The W~ Starboy    global          0.998            0.806                6
## # ... with 46 more variables: seasonal_trough_week <dbl>, spikiness <dbl>,
## #   linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>,
## #   acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>, pacf5 <dbl>,
## #   diff1_pacf5 <dbl>, diff2_pacf5 <dbl>, season_pacf <dbl>,
## #   zero_run_mean <dbl>, nonzero_squared_cv <dbl>, zero_start_prop <dbl>,
## #   zero_end_prop <dbl>, lambda_guerrero <dbl>, kpss_stat <dbl>,
## #   kpss_pvalue <dbl>, pp_stat <dbl>, pp_pvalue <dbl>, ndiffs <int>,
## #   nsdiffs <int>, bp_stat <dbl>, bp_pvalue <dbl>, lb_stat <dbl>,
## #   lb_pvalue <dbl>, var_tiled_var <dbl>, var_tiled_mean <dbl>,
## #   shift_level_max <dbl>, shift_level_index <dbl>, shift_var_max <dbl>,
## #   shift_var_index <dbl>, shift_kl_max <dbl>, shift_kl_index <dbl>,
## #   spectral_entropy <dbl>, n_crossing_points <int>, longest_flat_spot <int>,
## #   coef_hurst <dbl>, stat_arch_lm <dbl>, arch_lm <dbl>
```

```
# we see the track 'Lose Yourself - Soundtrack Version' had the lowest trend_strength, which is 0.84188
# and the track 'Starboy' had the highest trend_strength, which is 0.9984568.

#visualization

#the lowest trend_strength
p1 <- spotify_all %>%
  mutate(date = as_date(date)) %>%
  filter(track_name == "Lose Yourself - Soundtrack Version") %>%
  ggplot(aes(date, streams))+
```

```
  geom_line(col = "#2b337c") +
  labs(
    title = "the number of streams of Lose Yourself - Soundtrack Version"
  )
#the highest trend_strength
p2 <- spotify_all %>%
  mutate(date = as_date(date)) %>%
  filter(track_name == "Starboy") %>%
  ggplot(aes(date, streams))+
  geom_line(col = "#8C2B37") +
  labs(
    title = "the number of streams of Starboy"
  )

grid.arrange(p1, p2, nrow = 2)
```



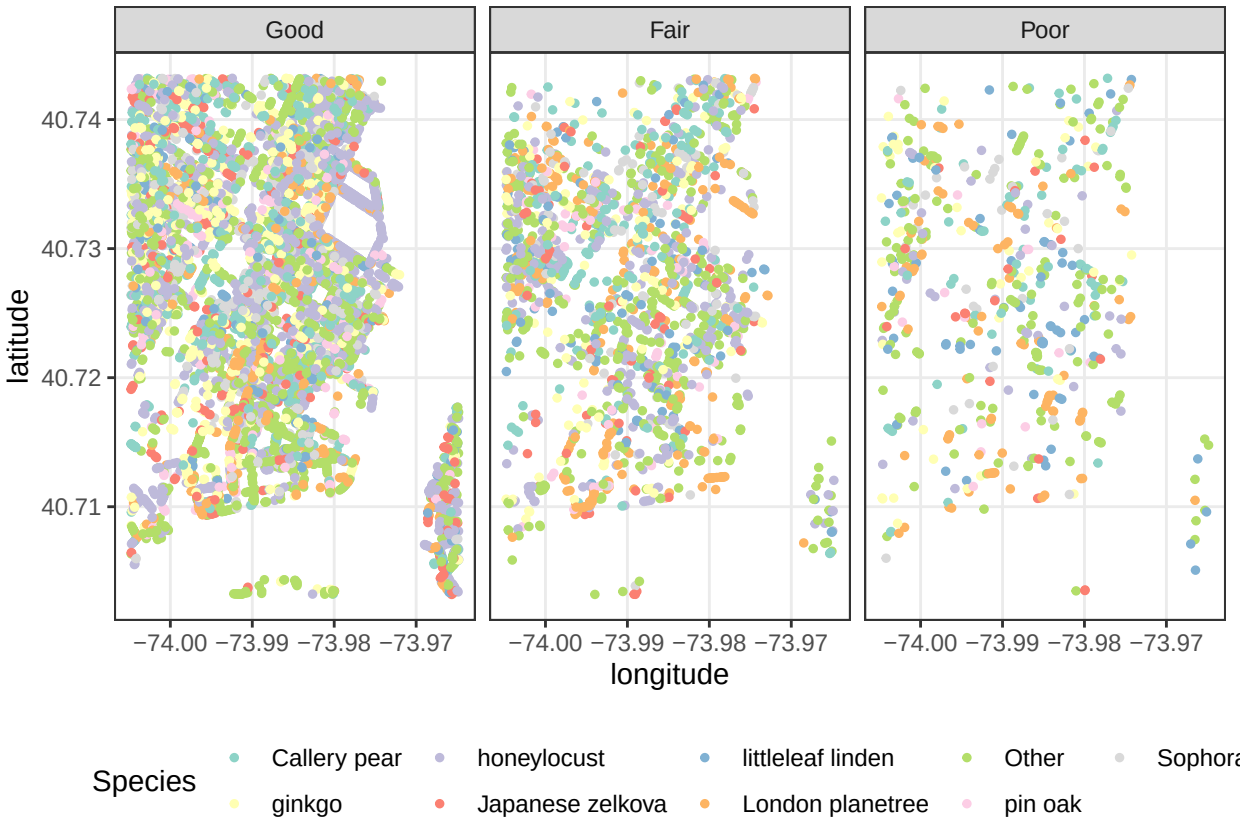
NYC Trees

In this problem, we'll use vector data to enrich a visualization of trees in New York City. In the process, we'll practice reading in and generating summaries of geospatial data.

a. The data at this link include a subset of data from the New York City Tree Census. Make a scatterplot of the locations of all trees in the data, coloring in by tree species group and faceting by health.

```
# read the data
trees <- read.csv("https://uwmadison.box.com/shared/static/t1mk6i4u5ks5bjxaw2c7soe2z8i75m2o.csv") %>%
  mutate(health = factor(health, level = c("Good", "Fair", "Poor")))
```

```
# make the plot
ggplot(trees, aes(longitude, latitude, col = species_group)) +
  geom_point(size = 1) +
  facet_wrap(~health) +
  scale_color_brewer(palette = "Set3") +
  labs(
    color = "Species"
  ) +
  theme(
    legend.position = "bottom",
    panel.grid.minor = element_blank()
  )
```



b. Suppose we wanted to relate these data to characteristics of the built environment. We have curated public data on roads and buildings within the same neighborhood. Read these data into `sf` objects using `read_sf`. For both datasets, report (i) the associated CRS and (ii) the geometry type (i.e., one of point, linestring, polygon, multipoint, multilinestring, multipolygon, geometry collection).

```
# Read the data
roads <- read_sf("https://uwmadison.box.com/shared/static/28y5003s1d0w9nqjnk9xme2n86xazuuj.geojson")
buildings <- read_sf("https://uwmadison.box.com/shared/static/qfmrp9srsoq0a7oj0e7xmgu5spojr33e.geojson")

# for roads
crs(roads)
```

```
## CRS arguments: +proj=longlat +datum=WGS84 +no_defs
```

```
st_geometry(roads)
```

```
## Geometry set for 775 features
## geometry type:  MULTILINESTRING
## dimension:      XY
## bbox:           xmin: -74.0048 ymin: 40.7032 xmax: -73.9648 ymax: 40.7432
## geographic CRS: WGS 84
## First 5 geometries:
```

```
# the geographic CRS is WGS 84 and the geometry type is multilinestring

# for buildings
crs(buildings)
```

```
## CRS arguments: +proj=longlat +datum=WGS84 +no_defs
```

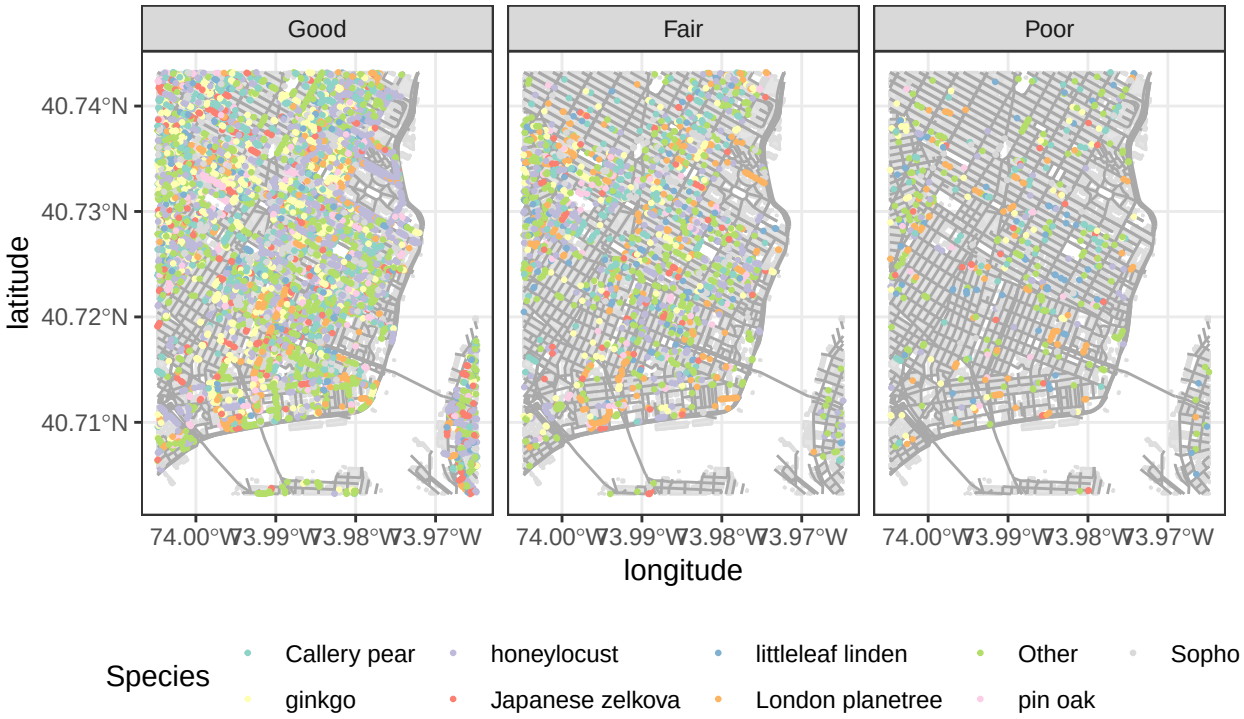
```
st_geometry(buildings)
```

```
## Geometry set for 2726 features
## geometry type:  MULTIPOLYGON
## dimension:      XY
## bbox:           xmin: -74.0048 ymin: 40.7032 xmax: -73.9648 ymax: 40.7432
## geographic CRS: WGS 84
## First 5 geometries:
```

```
# the geographic CRS is WGS 84 and the geometry type is multipolygon
```

c. Generate a version of the plot in (a) that has the roads and buildings in the background. An example result is given in Figure 1.

```
ggplot(trees, aes(longitude, latitude, col = species_group)) +
  geom_sf(data = buildings, col = "gray87", inherit.aes = F) +
  geom_sf(data = roads, col = "darkgray", inherit.aes = F) +
  geom_point(size = 0.5) +
  facet_wrap(~health) +
  scale_color_brewer(palette = "Set3") +
  labs(
    color = "Species"
  ) +
  theme(
    legend.position = "bottom",
    panel.grid.minor = element_blank()
  )
```



Himalayan Glaciers

In this problem, we'll apply the reading's discussion of raster data to understand a dataset containing Landsat 7 satellite imagery of a Himalayan glacier system.

a. Read the data into a `brick` object. What is the spatial extent of the file (that is, within what geographic coordinates do we have data)? How many layers of sensor measurements are available?

```
glacier <- brick("glaciers-small.tif")
glacier
```

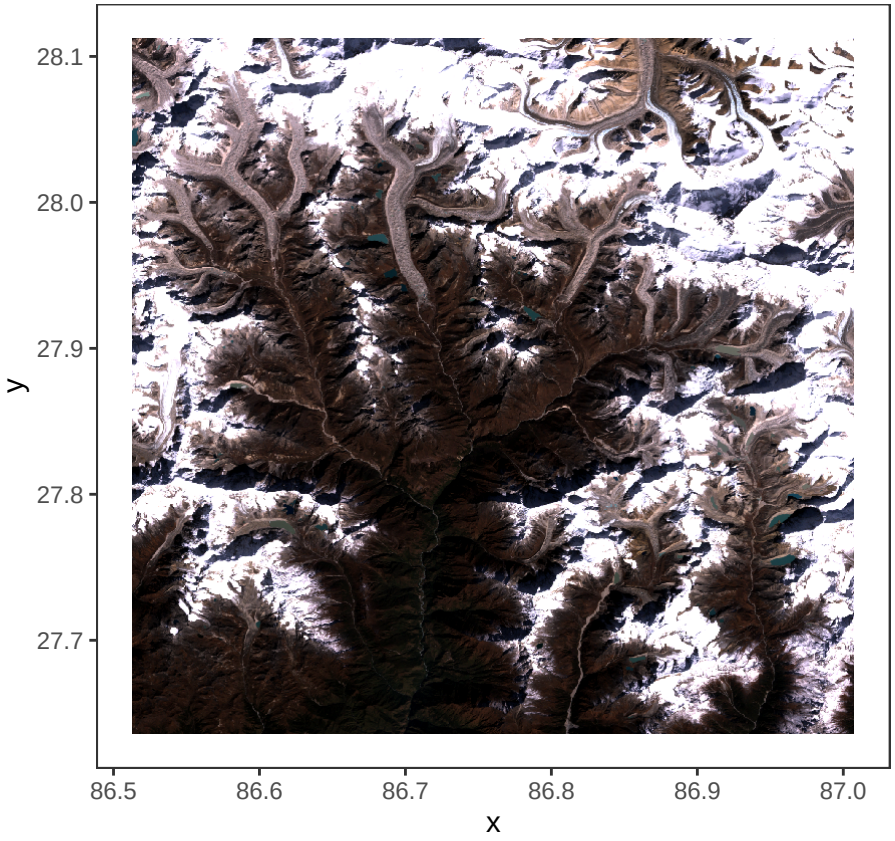
```
## class      : RasterBrick
## dimensions : 1645, 1708, 2809660, 15  (nrow, ncol, ncell, nlayers)
## resolution : 0.0002893326, 0.000289386  (x, y)
## extent     : 86.51314, 87.00732, 27.63608, 28.11212  (xmin, xmax, ymin, ymax)
## crs        : +proj=longlat +datum=WGS84 +no_defs
## source     : C:/Users/kiara/OneDrive/desktop/stat 479/hw3/glaciers-small.tif
## names      : B1, B2, B3, B4, B5, B6_VCID_1, B6_VCID_2, B7, B8, BQA, ndvi, ndsi, ndwi, elevation, slo
```

From above, we see the spacial extend is 86.51314 - 87.00732 for x and 27.63608 - 28.11212 for y. There are 15 layers of sensor measurements available.

b. Generate an RGB image of this area. In Landsat 7, the first three layers (B1, B2, and B3) provide the red, green, and blue channels.

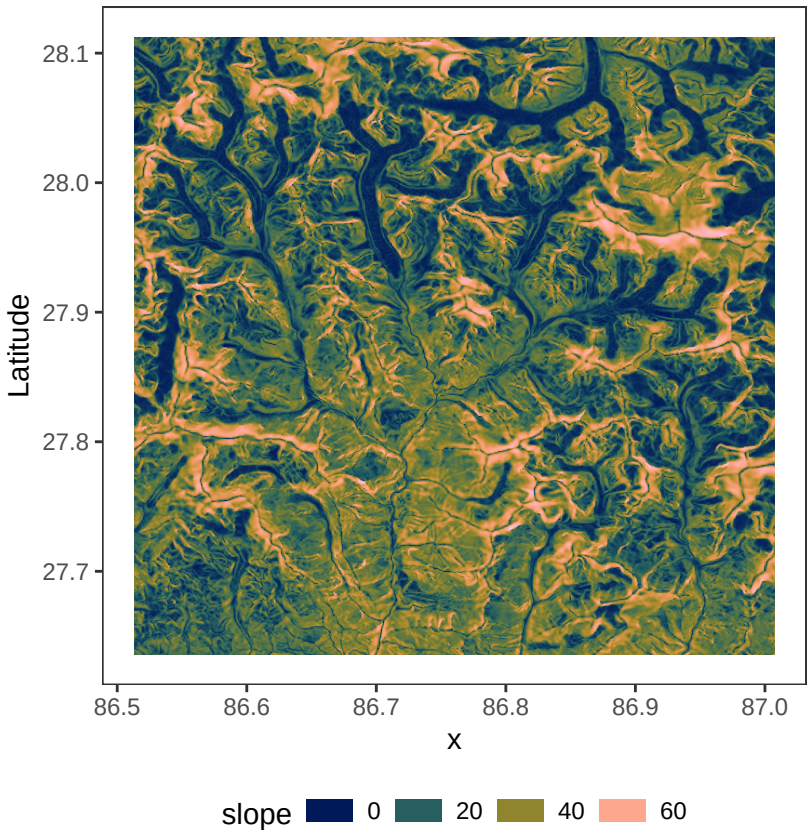
```
ggRGB(glacier, stretch = "lin")+
  theme(
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
```

```
  ) +
  coord_fixed(xlim = c(86.51314, 87.00732), ylim = c(27.63608, 28.11212))
```



c. Make a plot of the slopes associated with each pixel within this region. An example result is shown in Figure 2.

```
glacier %>%
  as.data.frame(xy = T) %>%
  ggplot(aes(x, y, fill = slope)) +
  geom_raster() +
  scale_fill_scico(palette = "batlow") +
  labs(
    X = "Longitude",
    y = "Latitude"
  ) +
  theme(
    legend.position = "bottom",
    panel.grid.major = element_blank(),
    panel.grid.minor = element_blank()
  ) +
  coord_fixed(xlim = c(86.51314, 87.00732), ylim = c(27.63608, 28.11212)) +
  guides(fill = guide_legend(keyheight = 0.5))
```



CalFresh Enrollment

In this problem, we will investigate spatial and temporal aspects of enrollment in CalFresh, a nutritional assistance program in California.

a. The code below reads in the CalFresh data. We've filtered out February 2019, since benefits were distributed differently in this month, leading to outliers for most counties. Extract features of the `calfresh` time series using the `features` function in the feasts library.

```
calfresh <- read_csv("https://uwmadison.box.com/shared/static/rduej9hsc4w3mdethxnx9ccv752f22yr.csv") %>%
  filter(date != "2019 Feb") %>%
  mutate(date = yearmonth(date)) %>%
  as_tsibble(key = county, index = date)

#extract features
calfresh_new <- calfresh %>%
  features(calfresh, feature_set(pkgs = "feasts"))

calfresh_new
```

```
## # A tibble: 58 x 40
##    county trend_strength seasonal_streng~ seasonal_peak_y~ seasonal_trough~
##    <chr>           <dbl>            <dbl>            <dbl>            <dbl>
##  1 Alame~          0.926            0.290                5                2
##  2 Alpine          0.946            0.247                3                9
##  3 Amador          0.943            0.265                5                2
##  4 Butte           0.748            0.216                5                2
##  5 Calav~          0.763            0.187                5               11
```

```
##  6 Colusa          0.953            0.537                5               11
##  7 Contr~          0.939            0.266                7                2
##  8 Del N~          0.941            0.154                5                2
##  9 El Do~          0.784            0.307                5               11
## 10 Fresno          0.766            0.191                5               11
## # ... with 48 more rows, and 35 more variables: spikiness <dbl>,
## #   linearity <dbl>, curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>,
## #   acf1 <dbl>, acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>,
## #   diff2_acf1 <dbl>, diff2_acf10 <dbl>, season_acf1 <dbl>,
## #   lambda_guerrero <dbl>, kpss_stat <dbl>, kpss_pvalue <dbl>, pp_stat <dbl>,
## #   pp_pvalue <dbl>, ndiffs <int>, nsdiffs <int>, bp_stat <dbl>,
## #   bp_pvalue <dbl>, lb_stat <dbl>, lb_pvalue <dbl>, var_tiled_var <dbl>,
## #   var_tiled_mean <dbl>, shift_level_max <dbl>, shift_level_index <dbl>,
## #   shift_var_max <dbl>, shift_var_index <dbl>, shift_kl_max <dbl>,
## #   shift_kl_index <dbl>, spectral_entropy <dbl>, n_crossing_points <int>,
## #   longest_flat_spot <int>, stat_arch_lm <dbl>
```

b. Visualize CalFresh enrollment over time for the counties with the highest and lowest `seasonal_strength_year`.

```
# identify counties had the highest and the lowest 'seasonal_strength_year'
calfresh_new %>%
  arrange(seasonal_strength_year) %>%
  head(1)
```

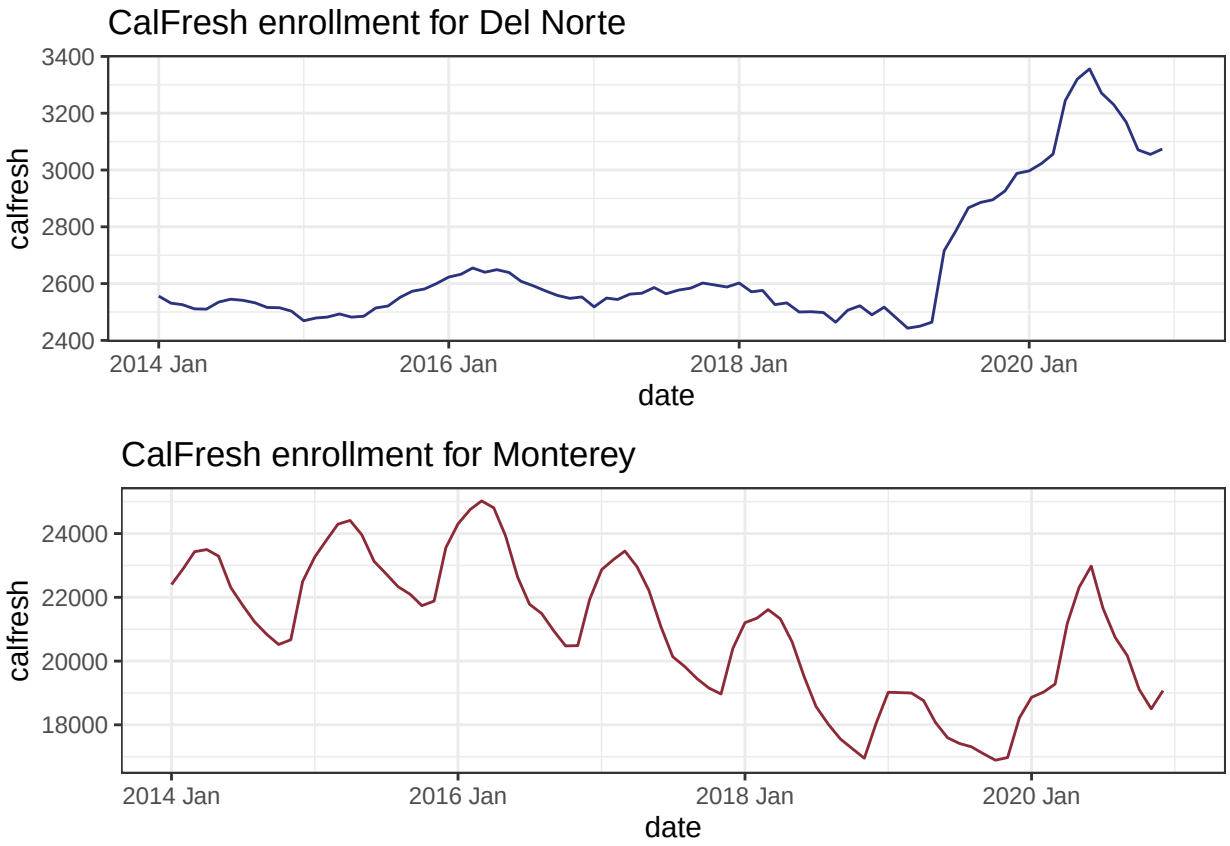
```
## # A tibble: 1 x 40
##   county trend_strength seasonal_streng~ seasonal_peak_y~ seasonal_trough~
##   <chr>           <dbl>            <dbl>            <dbl>            <dbl>
## 1 Del N~          0.941            0.154                5                2
## # ... with 35 more variables: spikiness <dbl>, linearity <dbl>,
## #   curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>, acf1 <dbl>,
## #   acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>, diff2_acf1 <dbl>,
## #   diff2_acf10 <dbl>, season_acf1 <dbl>, lambda_guerrero <dbl>,
## #   kpss_stat <dbl>, kpss_pvalue <dbl>, pp_stat <dbl>, pp_pvalue <dbl>,
## #   ndiffs <int>, nsdiffs <int>, bp_stat <dbl>, bp_pvalue <dbl>, lb_stat <dbl>,
## #   lb_pvalue <dbl>, var_tiled_var <dbl>, var_tiled_mean <dbl>,
## #   shift_level_max <dbl>, shift_level_index <dbl>, shift_var_max <dbl>,
## #   shift_var_index <dbl>, shift_kl_max <dbl>, shift_kl_index <dbl>,
## #   spectral_entropy <dbl>, n_crossing_points <int>, longest_flat_spot <int>,
## #   stat_arch_lm <dbl>
```

```
calfresh_new %>%
  arrange(desc(seasonal_strength_year)) %>%
  head(1)
```

```
## # A tibble: 1 x 40
##   county trend_strength seasonal_streng~ seasonal_peak_y~ seasonal_trough~
##   <chr>           <dbl>            <dbl>            <dbl>            <dbl>
## 1 Monte~          0.941            0.867                3               10
## # ... with 35 more variables: spikiness <dbl>, linearity <dbl>,
## #   curvature <dbl>, stl_e_acf1 <dbl>, stl_e_acf10 <dbl>, acf1 <dbl>,
## #   acf10 <dbl>, diff1_acf1 <dbl>, diff1_acf10 <dbl>, diff2_acf1 <dbl>,
## #   diff2_acf10 <dbl>, season_acf1 <dbl>, lambda_guerrero <dbl>,
## #   kpss_stat <dbl>, kpss_pvalue <dbl>, pp_stat <dbl>, pp_pvalue <dbl>,
## #   ndiffs <int>, nsdiffs <int>, bp_stat <dbl>, bp_pvalue <dbl>, lb_stat <dbl>,
## #   lb_pvalue <dbl>, var_tiled_var <dbl>, var_tiled_mean <dbl>,
## #   shift_level_max <dbl>, shift_level_index <dbl>, shift_var_max <dbl>,
```

```
## #   shift_var_index <dbl>, shift_kl_max <dbl>, shift_kl_index <dbl>,
## #   spectral_entropy <dbl>, n_crossing_points <int>, longest_flat_spot <int>,
## #   stat_arch_lm <dbl>
```

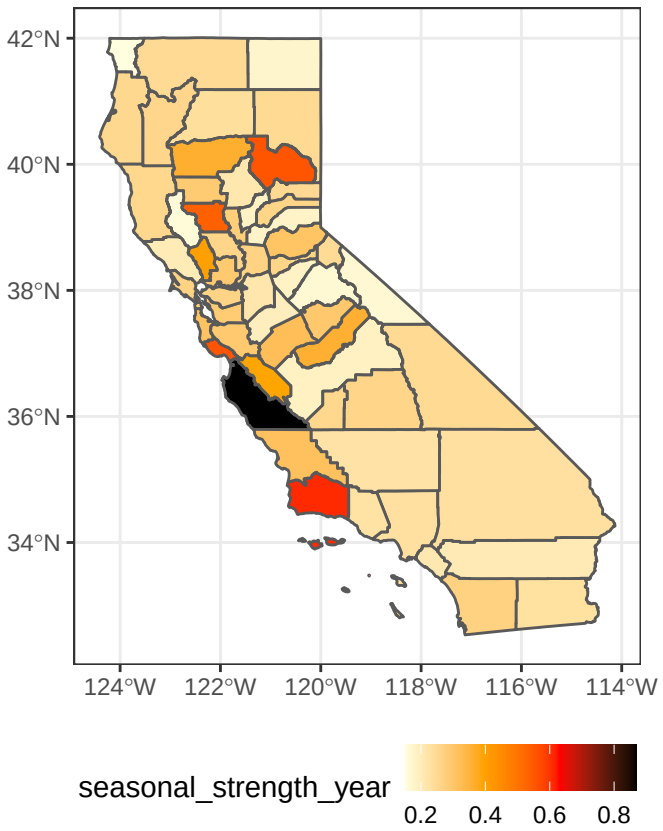
```r
# we see the county Del Norte has the lowest seasonal_strength_year and the county Monterey has the hig

#visualization

#the lowest seasonal_strength_year
p1 <- calfresh%>%
  filter(county == "Del Norte") %>%
  ggplot(aes(date, calfresh))+
  geom_line(col = "#2b337c") +
  labs(
    title = "CalFresh enrollment for Del Norte"
  )
#the highest seasonal_strength_year
p2 <- calfresh%>%
  filter(county == "Monterey") %>%
  ggplot(aes(date, calfresh))+
  geom_line(col = "#8C2B37") +
  labs(
    title = "CalFresh enrollment for Monterey"
  )
grid.arrange(p1, p2, nrow = 2)
```



c. The code below reads in a vector dataset demarcating the county boundaries in California. Join in

the features dataset from (a) with this these vector data. Use this to produce a map with each county shaded in by its `seasonal_strength_year`. An example result is shown in Figure 3.

```
counties <- read_sf("https://uwmadison.box.com/shared/static/gropucqxgqm82yhq13do1ws9k16dnxq7.geojson")

# join the two datasets
calfreshe_joined <- counties %>%
  left_join(calfresh_new)

#produce the map
ggplot(calfreshe_joined) +
  geom_sf(aes(fill = seasonal_strength_year)) +
  scale_fill_gradientn(colors = c("lightyellow", "orange", "red", "black")) +
  theme(
    legend.position = "bottom"
  )
```



d. Propose, but do not implement, a visualization of this dataset that makes use of dynamic queries. What questions would the visualization answer? What would be the structure of interaction, and how would the display update when the user provides a cue?

I think we can make use of selections and dynamic queries to visualize CalFresh enrollment for different counties. For example, if we want to see the situation in Monterey, we can select "Monterey", and a graph like part b) will be displayed. We can also make comparisons by having the user choose two or three counties, using interval selections. For example, the user can select several different counties, and by using cursor to select a time interval, he can observe the situation of different counties in the same period. This visualization can answer questions like "what are the differences between the situation in two different counties" or "how the calFresh enrollments in xx county change from 2018 to 2020".

Political Book Recommendations

In this problem, we'll study a network dataset of Amazon bestselling US Politics books. Books are linked by an edge if they appeared together in the recommendations ("customers who bought this book also bought these other books").

a. The code below reads in the edges and nodes associated with the network. The edges dataset only contains IDs of co-recommended books, while the nodes data includes attributes associated with each book. Use the edges dataset to create an igraph graph object, and use the `ggnetwork` function to construct a data.frame summarizing the layout of the network. Use a `layout_with_fr` layout, as in the reading.

```
edges <- read_csv("https://uwmadison.box.com/shared/static/54i59bfc5jhymnn3hsw8fyolujesalut.csv")
nodes <- read_csv("https://uwmadison.box.com/shared/static/u2x392i79jycubo5rhzryxjsvd1jjrdy.csv", col_ty

# create an igraph graph object
edge_graph <- graph_from_data_frame(edges, directed = F)

#construct a data frame
layout <- ggnetwork(edge_graph, layout = layout_with_fr(edge_graph))
```

b. The output from (a) does not include any attributes about the books, since this is only available in the `nodes` dataset, and we built the graph layout using only `edges`. Join in the node attribute data. **Hint**: Use `left_join`, but using the `by` argument to ensure that `name` in the output of `ggnetwork` and `Id` in the original `nodes` dataset are associated with one another.

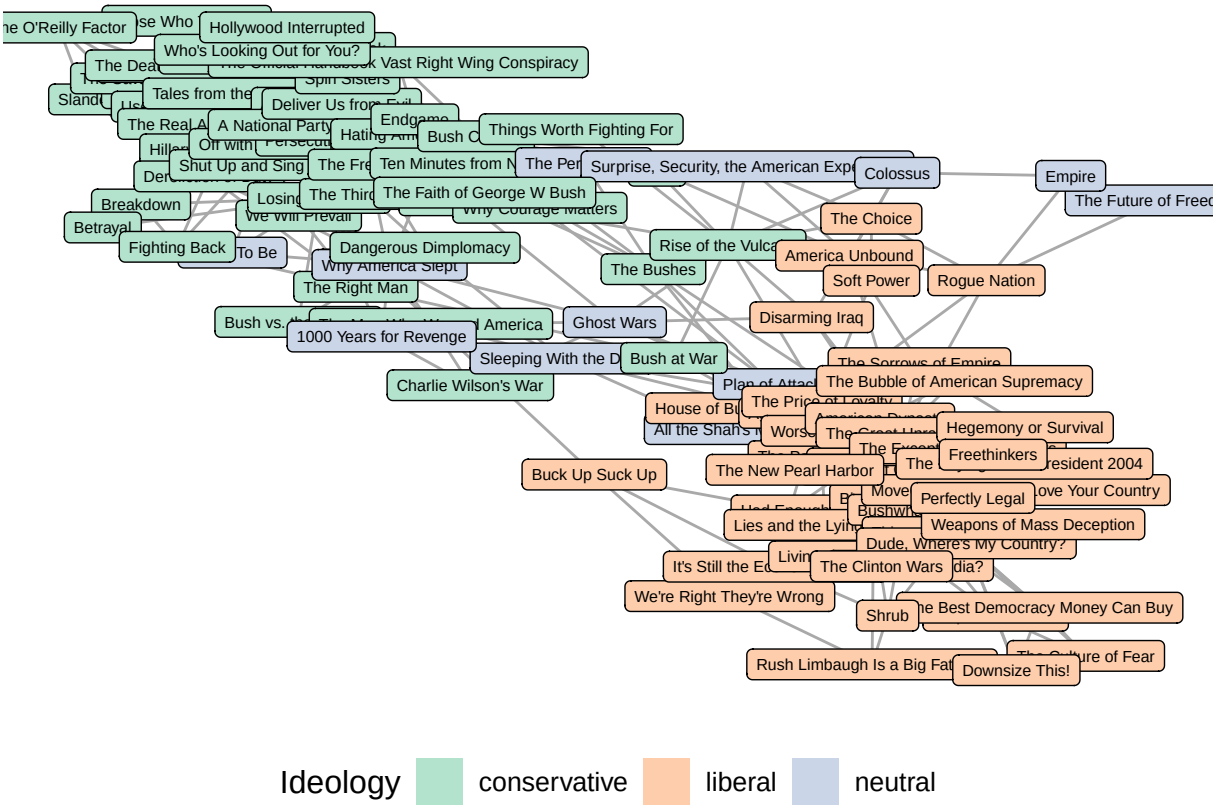
```
data <- layout %>%
  left_join(nodes, by = c("name" = "Id"))
head(data)
```

```
##            x         y name       xend      yend Weight     Label
## 1 0.02462282 0.8789803   38 0.08151345 0.9112350      1   Slander
## 2 0.02462282 0.8789803   38 0.00000000 0.9896486      1   Slander
## 3 0.06179676 0.8884891   37 0.21757844 0.8737573      1      Bias
## 4 0.06179676 0.8884891   37 0.02462282 0.8789803      1      Bias
## 5 0.06179676 0.8884891   37 0.00000000 0.9896486      1      Bias
## 6 0.07484940 0.7170824   15 0.04008092 0.6817166      1 Breakdown
##   political_ideology
## 1       conservative
## 2       conservative
## 3       conservative
## 4       conservative
## 5       conservative
## 6       conservative
```

c. Use the result from part (b) to visualize the network using. Include the book's title in the node label, and shade in the node according to political ideology. An example result is shown in Figure 4.

```
data %>%
  ggplot(aes(x, y, xend = xend, yend = yend)) +
  geom_edges(color = "darkgrey") +
  geom_nodes() +
  geom_nodelabel(aes(label = Label, fill = political_ideology), size = 2) +
  scale_fill_brewer(palette = "Pastel2") +
  theme_blank() +
  theme(
    legend.position = "bottom"
```

```
) +
guides(fill = guide_legend(title = "Ideology", override.aes = aes(label = "")))
```



Feedback

a. How much time did you spend on this homework? One day.
b. Which problem did you find most valuable? The Spotify problem, because I also use Spotify and I feel it is really interesting.