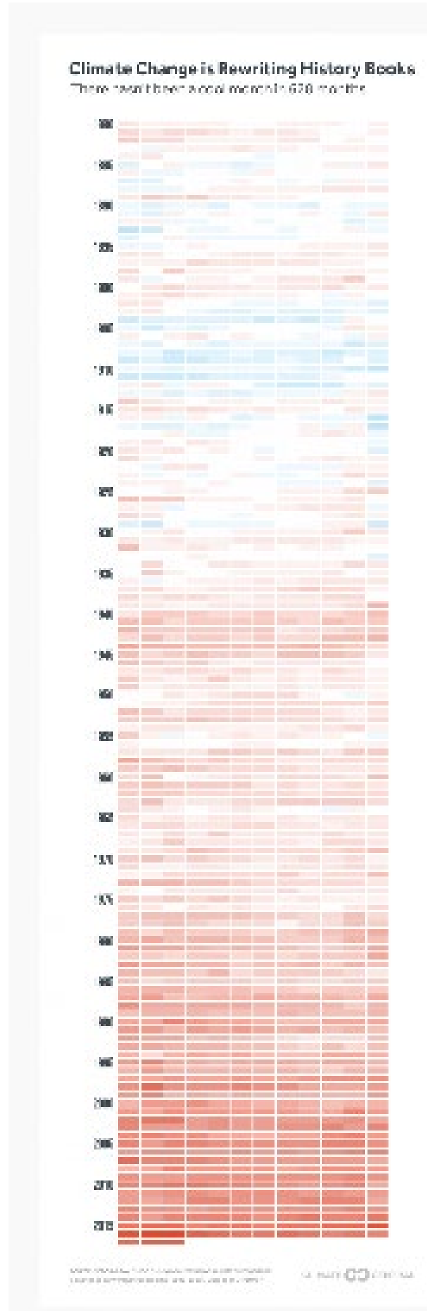
Study Guide Unit 5
DATA 110
Saidi

Treemaps, Heatmaps, Alluvials and Streamgraphs



Intro to Heatmaps

The following heatmap is one of my favorites created to show how climate change is affecting global temperatures. It tells a story of temperatures warming over a long period of time.

Climate Change is Rewriting the History Books – There hasn't been a cool month in 628 months (dated from 2015)

This week, you will learn how to make a heatmap in R from Flowing Data Tutorial

https://flowingdata.com/2010/01/21/how-to-make-a-heatmap-a-quick-and-easy-solution/

Heat Tree Stream Alluvial - Part 1

R Saidi

So many ways to visualize data

Load the packages and the data from flowingdata.com website

```
library(treemap)
library(tidyverse)
library(RColorBrewer)
library(webshot2)
```

Heatmaps

A heatmap is a literal way of visualizing a table of numbers, where you substitute the numbers with colored cells. There are two fundamentally different categories of heat maps: the cluster heat map and the spatial heat map. In a cluster heat map, magnitudes are laid out into a matrix of fixed cell size whose rows and columns are discrete categories, and the sorting of rows and columns is intentional. The size of the cell is arbitrary but large enough to be clearly visible. By contrast, the position of a magnitude in a spatial heat map is forced by the location of the magnitude in that space, and there is no notion of cells; the phenomenon is considered to vary continuously. (Wikipedia)

Load the nba data from Yau's website

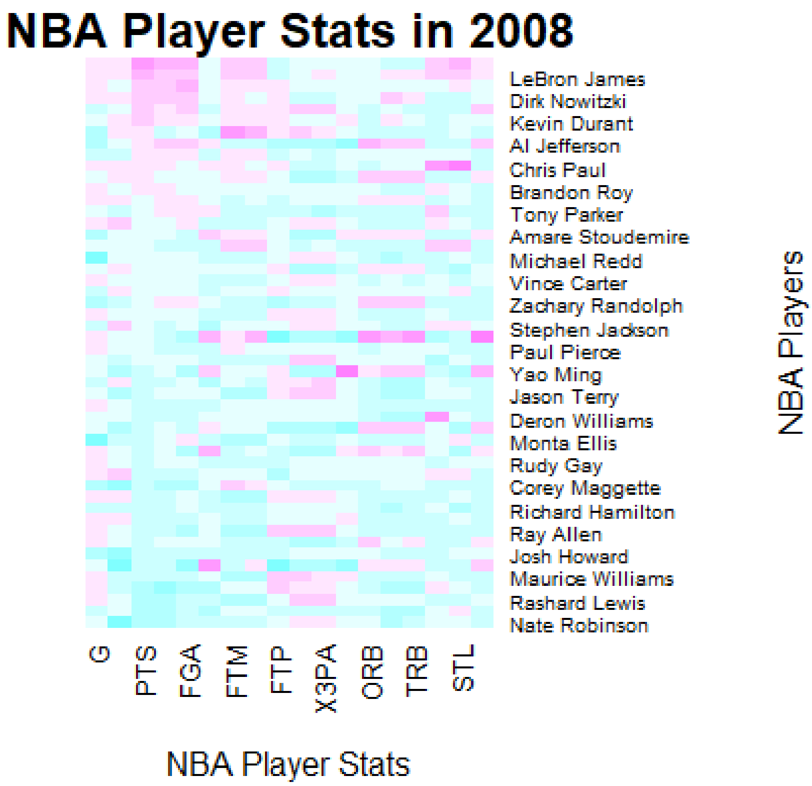
This data appears to contain data about 2008 NBA player stats.

```
nba <- read.csv("http://datasets.flowingdata.com/ppg2008.csv")
#apparently you have to use read.csv here instead of read_csv
head(nba)

            Name  G  MIN  PTS  FGM  FGA   FGP FTM FTA   FTP X3PM X3PA  X3PP ORB
1   Dwyane Wade  79 38.6 30.2 10.8 22.0 0.491 7.5 9.8 0.765  1.1  3.5 0.317 1.1
2  LeBron James  81 37.7 28.4  9.7 19.9 0.489 7.3 9.4 0.780  1.6  4.7 0.344 1.3
3   Kobe Bryant  82 36.2 26.8  9.8 20.9 0.467 5.9 6.9 0.856  1.4  4.1 0.351 1.1
4 Dirk Nowitzki  81 37.7 25.9  9.6 20.0 0.479 6.0 6.7 0.890  0.8  2.1 0.359 1.1
5 Danny Granger  67 36.2 25.8  8.5 19.1 0.447 6.0 6.9 0.878  2.7  6.7 0.404 0.7
6  Kevin Durant  74 39.0 25.3  8.9 18.8 0.476 6.1 7.1 0.863  1.3  3.1 0.422 1.0
  DRB TRB AST STL BLK  TO  PF
1 3.9 5.0 7.5 2.2 1.3 3.4 2.3
2 6.3 7.6 7.2 1.7 1.1 3.0 1.7
3 4.1 5.2 4.9 1.5 0.5 2.6 2.3
4 7.3 8.4 2.4 0.8 0.8 1.9 2.2
5 4.4 5.1 2.7 1.0 1.4 2.5 3.1
6 5.5 6.5 2.8 1.3 0.7 3.0 1.8
```

Create a cool-color heatmap

This older heatmap function requires the data to be formatted as a matrix using the data.matrix

```
nba <- nba[order(nba$PTS),]
row.names(nba) <- nba$Name
nba <- nba[,2:19]
nba_matrix <- data.matrix(nba)
nba_heatmap <- heatmap(nba_matrix,
                       Rowv=NA,
                       Colv=NA,
                       col = cm.colors(10),
                       scale="column",
                       margins=c(5,10),
                       xlab = "NBA Player Stats",
                       ylab = "NBA Players",
                       main = "NBA Player Stats in 2008")
```



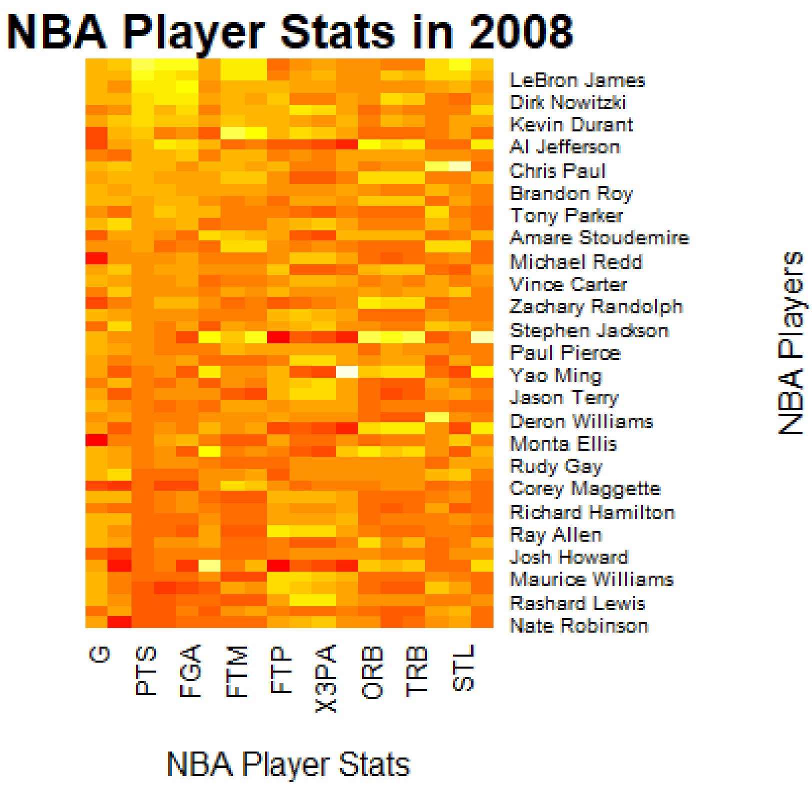
What did that plot show?

The basic layout of the heatmap relies on the parameters rows, columns and values. You can think of them like aesthetics in ggplot2::ggplot(), similar to something like aes(x = columns, y = rows, fill = values).

Change to warm color palette

```
nba_heatmap <- heatmap(nba_matrix,
                       Rowv=NA,
                       Colv=NA,
                       col = heat.colors(20),
                       scale="column",
                       margins=c(5,10),
                       xlab = "NBA Player Stats",
```

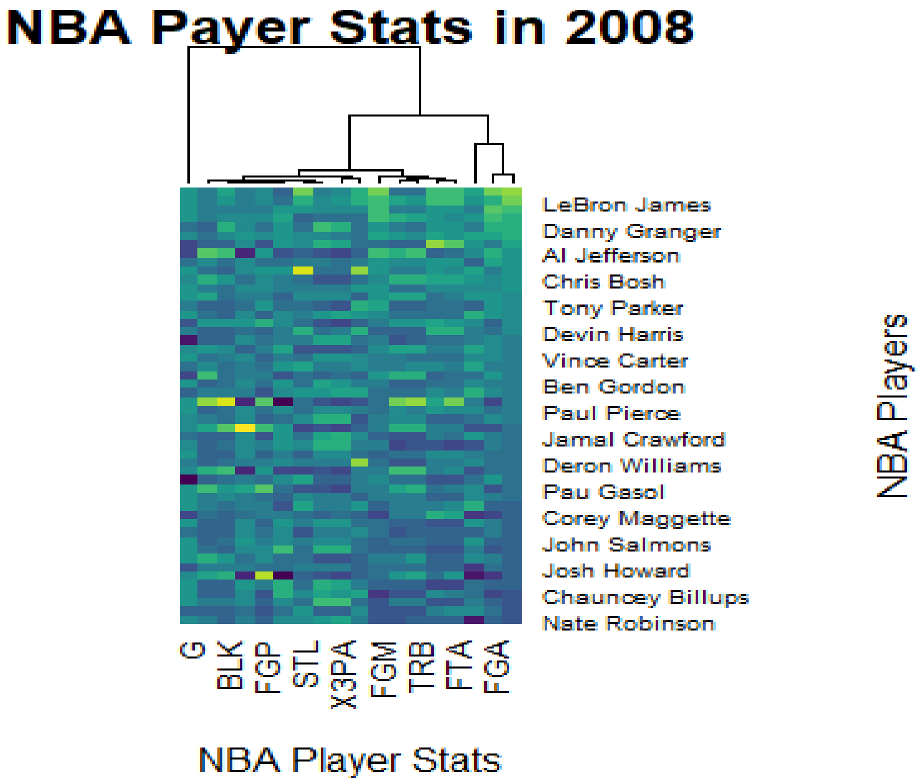
```
                     ylab = "NBA Players",
                     main = "NBA Player Stats in 2008")
```



Use the viridis color palette

For some reason the veridis colors from viridisLite package default to give dentrite clusering (the branches).

```
library(viridis)

Loading required package: viridisLite

# Loading required package: viridis
nba_heatmap <- heatmap(nba_matrix,
                     Rowv=NA,
                     col = viridis(20),
                     scale="column",
                     margins=c(5,10),
                     xlab = "NBA Player Stats",
                     ylab = "NBA Players",
                     keep.dendro = FALSE,
                     main = "NBA Payer Stats in 2008")
```



Treemaps

Treemaps display hierarchical (tree-structured) data as a set of nested rectangles. Each branch of the tree is given a rectangle, which is then tiled with smaller rectangles representing sub-branches. A leaf node's rectangle has an area proportional to a specified dimension of the data.[1] Often the leaf nodes are colored to show a separate dimension of the data.

When the color and size dimensions are correlated in some way with the tree structure, one can often easily see patterns that would be difficult to spot in other ways, such as whether a certain color is particularly relevant. A second advantage of treemaps is that, by construction, they make efficient use of space. As a result, they can legibly display thousands of items on the screen simultaneously.

The Downside to Treemaps

The downside of treemaps is that as the aspect ratio is optimized, the order of placement becomes less predictable. As the order becomes more stable, the aspect ratio is degraded. (Wikipedia)

Use Nathan Yau's dataset from the flowingdata website: http://datasets.flowingdata.com/post-data.txt You will need the package "treemap" and the package "RColorBrewer".

Create a treemap which explores categories of views

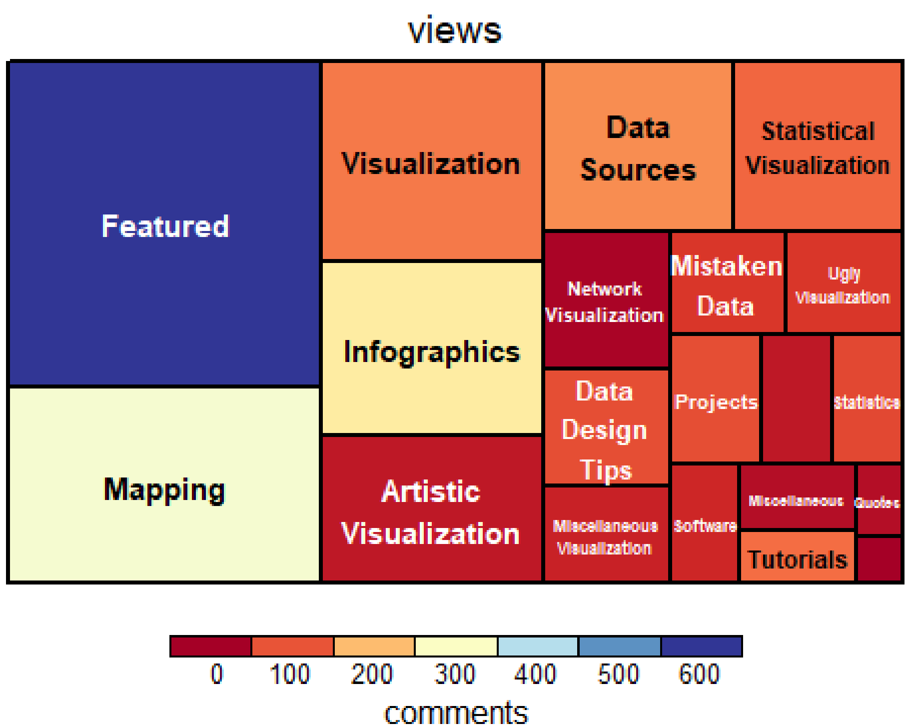
Load the data for creating a treemap from Nathan Yao's flowing data which explores number of views and comments for different categories of posts on his website.

```
flowingdata <- read.csv("http://datasets.flowingdata.com/post-data.txt")
# again, here use read.csv instead of read_csv
head(flowingdata)
```

```
    id  views comments              category
1 5019 148896       28 Artistic Visualization
2 1416  81374       26          Visualization
3 1416  81374       26               Featured
4 3485  80819       37               Featured
5 3485  80819       37                Mapping
6 3485  80819       37           Data Sources
```

Use RColorBrewer to change the palette to RdYlBu

```
treemap(flowingdata, index="category", vSize="views",
        vColor="comments", type="manual",
        # note: type = "manual" changes to red yellow blue
        palette="RdYlBu")
```



Notice the following:

- The index is a categorical variable - in this case, “category” of post
- The size of the box is by number of views of the post
- The heatmap color is by number of comments for the post
- Notice how the treemap includes a legend for number of comments *

A heatmap of World Happiness

Set your working directory and read in the happiness19.csv from the class google drive.

```
setwd("C:/Users/rsaidi/Dropbox/Rachel/MontColl/Datasets/Datasets")
happy19 <- read_csv("happiness2019.csv")
head(happy19)
```

```
# A tibble: 6 × 9
  `Overall rank` `Country or region` Score `GDP per capita` `Social support`
           <dbl> <chr>               <dbl>            <dbl>            <dbl>
1              1 Finland              7.77             1.34             1.59
2              2 Denmark              7.6              1.38             1.57
3              3 Norway               7.55             1.49             1.58
4              4 Iceland              7.49             1.38             1.62
5              5 Netherlands          7.49             1.40             1.52
6              6 Switzerland          7.48             1.45             1.53
# ℹ 4 more variables: `Healthy life expectancy` <dbl>,
#   `Freedom to make life choices` <dbl>, Generosity <dbl>,
#   `Perceptions of corruption` <dbl>
```

We can see that there are 156 countries ranked by their "happiness score" based on other measurements.

Clean the happiness dataset to work with it

first remove the first column for "overall_rank". Clean the remaining headers.

```
happy <- happy19 |>
  select(-`Overall rank`)
names(happy) <- tolower(names(happy))
names(happy) <- gsub(" ", "_", names(happy))
```

Because there are 156 countries, narrow the inclusion criteria to be for the top 20 scoring countries. Then do the same for the lowest 20.

```
happytop <- happy |>
  arrange(desc(score)) |>
  mutate(happy = "top") |> # add a column for use later
  head(20)
happytop
```

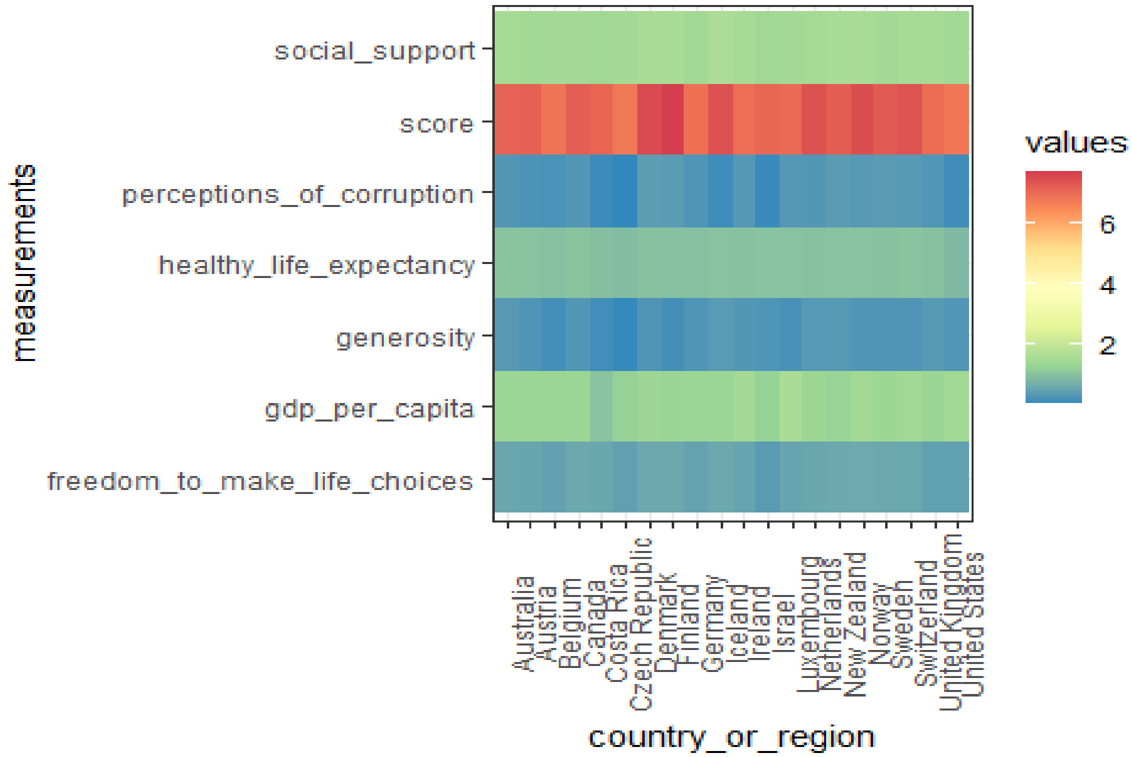
```
# A tibble: 20 × 9
```

```
happybottom <- happy |>
  arrange(score) |>
  mutate(happy = "bottom") |>
  head(20)
happybottom
```

```
# A tibble: 20 × 9
```

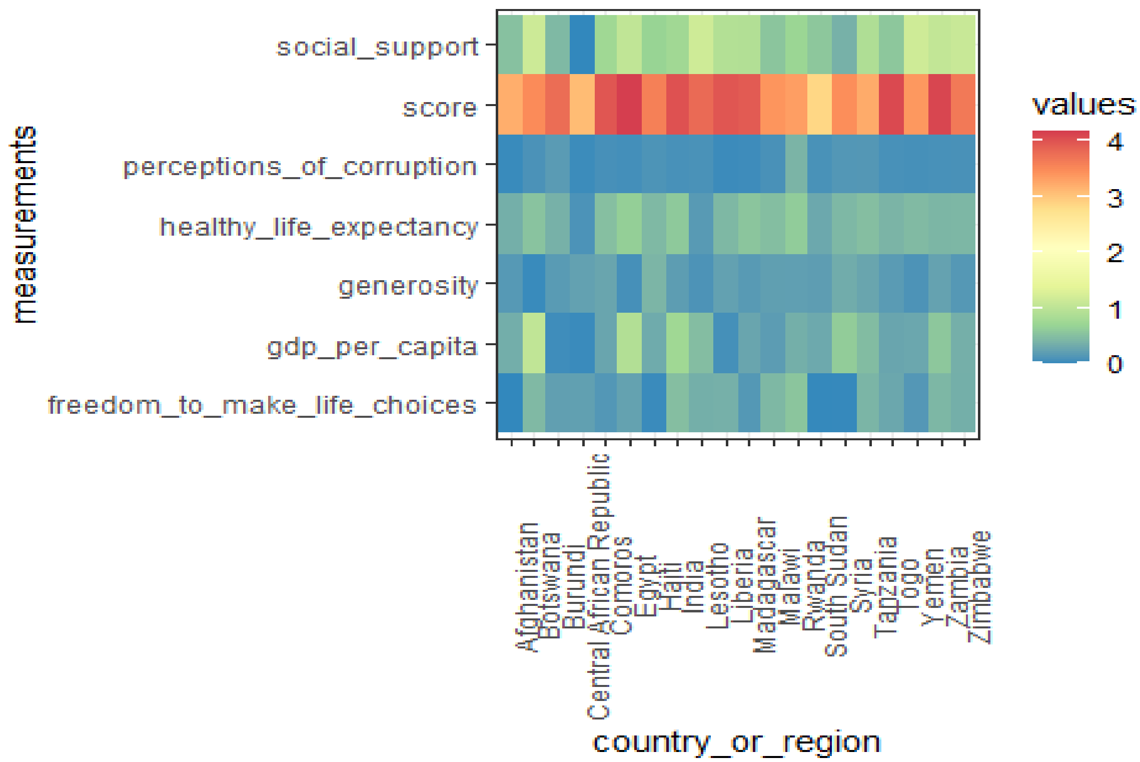
Then convert from wide to long format.

```
happy_longtop <- happytop |>
  pivot_longer(cols = 2:8,
               names_to = "measurements",
               values_to = "values")
happy_longtop

# A tibble: 140 × 4
   country_or_region happy measurements                 values
   <chr>             <chr> <chr>                         <dbl>
 1 Finland           top   score                         7.77
 2 Finland           top   gdp_per_capita                1.34
 3 Finland           top   social_support                1.59
 4 Finland           top   healthy_life_expectancy       0.986
 5 Finland           top   freedom_to_make_life_choices  0.596
 6 Finland           top   generosity                    0.153
 7 Finland           top   perceptions_of_corruption     0.393
 8 Denmark           top   score                         7.6
 9 Denmark           top   gdp_per_capita                1.38
10 Denmark           top   social_support                1.57
# ℹ 130 more rows

happy_longbottom <- happybottom |>
  pivot_longer(cols = 2:8,
               names_to = "measurements",
               values_to = "values")
happy_longbottom

# A tibble: 140 × 4
   country_or_region        happy  measurements                 values
   <chr>                    <chr>  <chr>                         <dbl>
 1 South Sudan              bottom score                         2.85
 2 South Sudan              bottom gdp_per_capita                0.306
 3 South Sudan              bottom social_support                0.575
 4 South Sudan              bottom healthy_life_expectancy       0.295
 5 South Sudan              bottom freedom_to_make_life_choices  0.01
 6 South Sudan              bottom generosity                    0.202
 7 South Sudan              bottom perceptions_of_corruption     0.091
 8 Central African Republic bottom score                         3.08
 9 Central African Republic bottom gdp_per_capita                0.026
10 Central African Republic bottom social_support                0
# ℹ 130 more rows

ggplot(data = happy_longtop, aes(x=country_or_region, y=measurements, fill = values)) +
  geom_tile()+
  scale_fill_distiller(palette="Spectral") +
  theme_bw()+
  theme(axis.text.x = element_text(angle = 90))
```



What do we notice a common patterns and different patterns at the top?

```
ggplot(data = happy_longbottom, aes(x=country_or_region, y=measurements, fill = values))
+
  geom_tile()+
  scale_fill_distiller(palette="Spectral") +
  theme_bw()+
  theme(axis.text.x = element_text(angle = 90))
```



What do we notice a common patterns and different patterns at the bottom?

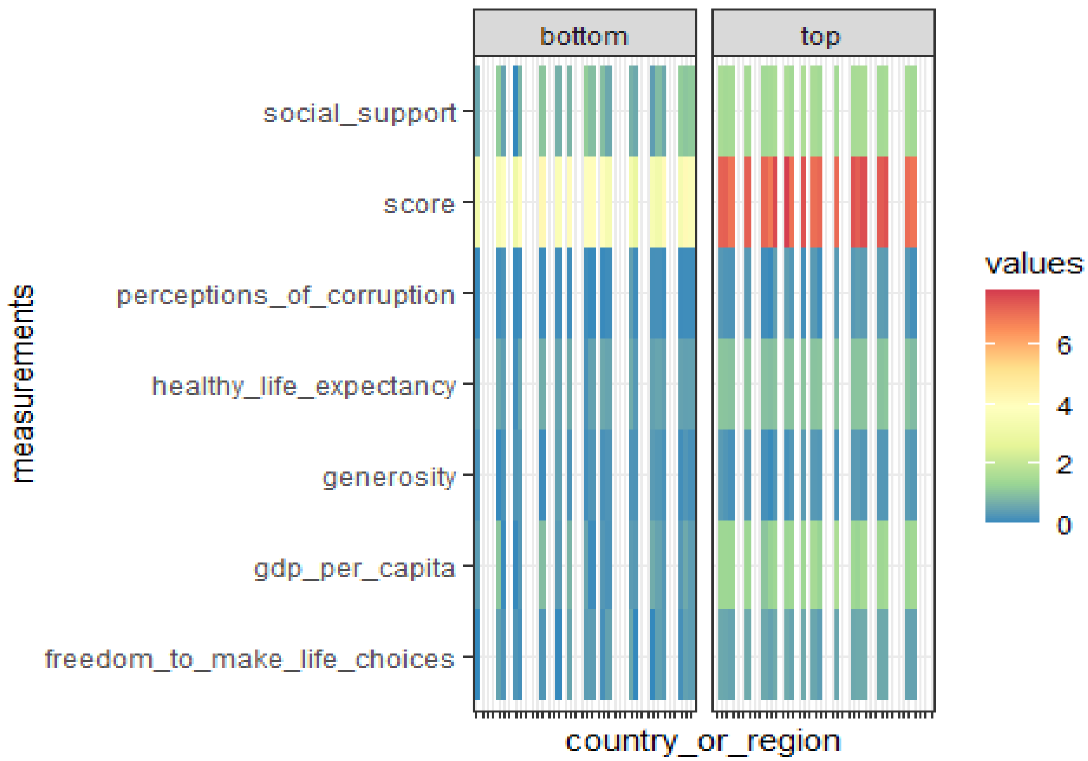
Put the top 20 and bottom 20 together to compare the two plots

```
newdf <- rbind(happytop, happybottom)
newdf_long <- newdf |>
    pivot_longer(cols = 2:8,
                 names_to = "measurements",
                 values_to = "values")
head(newdf_long)

# A tibble: 6 × 4
  country_or_region happy measurements                 values
  <chr>             <chr> <chr>                         <dbl>
1 Finland           top   score                         7.77
2 Finland           top   gdp_per_capita                1.34
3 Finland           top   social_support                1.59
4 Finland           top   healthy_life_expectancy       0.986
5 Finland           top   freedom_to_make_life_choices  0.596
6 Finland           top   generosity                    0.153
```

create a facet plot of the geom_tile

```
ggplot(data = newdf_long, aes(x=country_or_region, y=measurements, fill = values)) +
  geom_tile()+
  scale_fill_distiller(palette="Spectral") +
  facet_grid(~happy) +
  theme_bw()+
```

```
  theme(axis.text.x = element_blank())  # remove the countries to generally compare top a
nd bottom ranked countries
```



The facet plot places the top and bottom countries on the same scale, and now we can really make comparisons.

What do you notice now?

Alluvials

Load the alluvial package

Refugees is a prebuilt dataset in the alluvial package

If you want to save the prebuilt dataset to your folder, use the write_csv function

```
library(alluvial)
library(ggalluvial)
data(Refugees)
```

Show UNHCR-recognised refugees

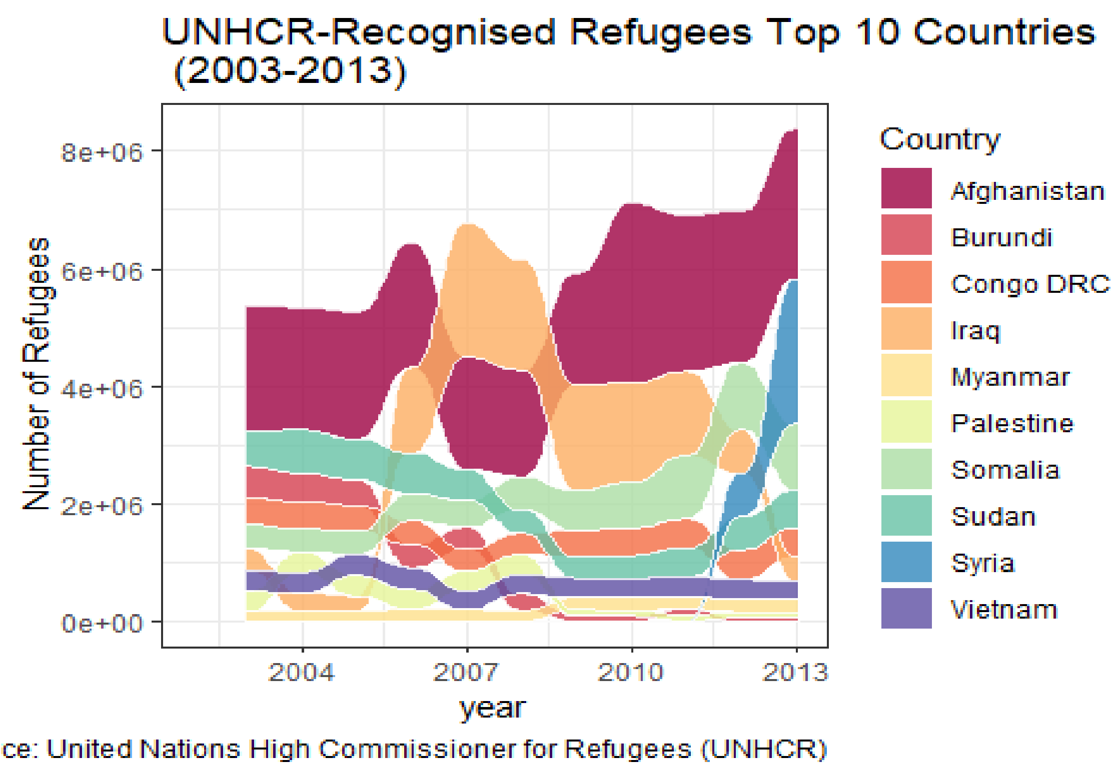
Top 10 most affected countries causing refugees from 2003-2013 Alluvials need the variables: *time-variable, value, category*

```
ggalluv <- Refugees |>
  ggplot(aes(x = year, y = refugees, alluvium = country)) +
  theme_bw() +
```

```
  geom_alluvium(aes(fill = country),
                color = "white",
                width = .1,
                alpha = .8,
                decreasing = FALSE) +
  scale_fill_brewer(palette = "Spectral") +
  # Spectral has enough colors for all countries listed
  scale_x_continuous(lim = c(2002, 2013)) +
  labs(title = "UNHCR-Recognised Refugees Top 10 Countries\n (2003-2013)",
         # \n breaks the long title
       y = "Number of Refugees",
       fill = "Country",
       caption = "Source: United Nations High Commissioner for Refugees (UNHCR)")
```

Plot the Alluvial

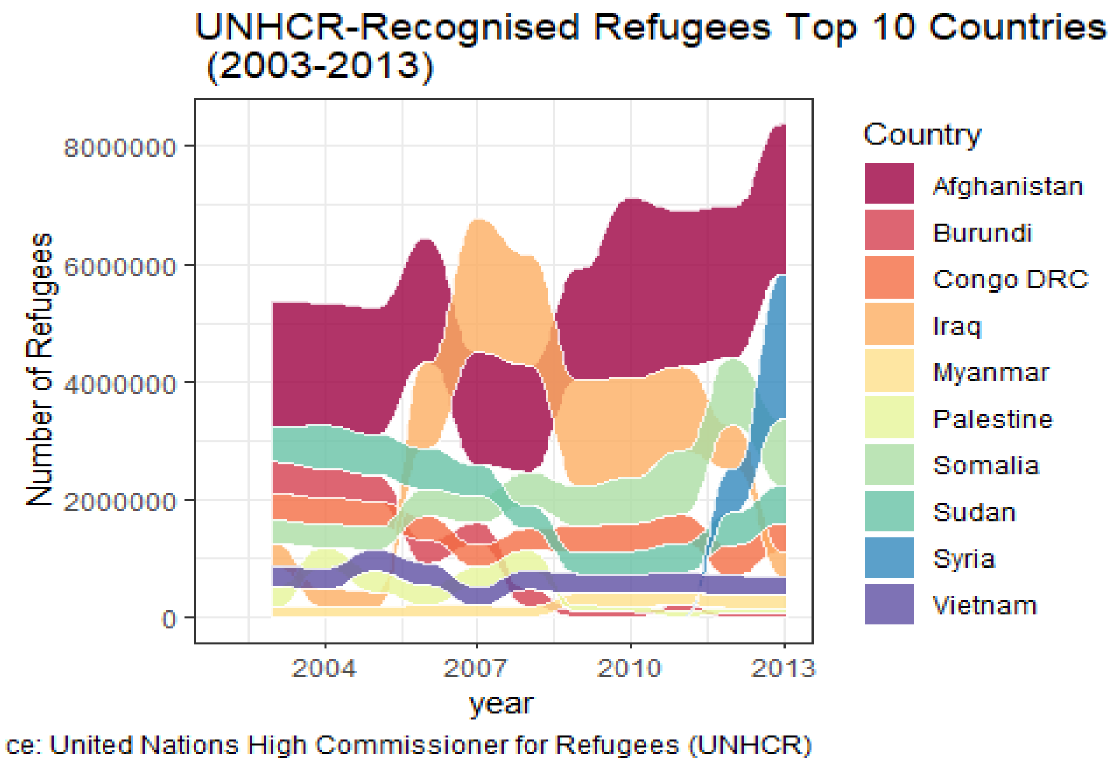
```
ggalluv
```



A final touch to fix the y-axis scale

Notice the y-values are in scientific notation. We can convert them to standard notation with options scipen function

```
options(scipen = 999)
ggalluv
```



Use the dataset NYCFlights23 to create a heatmap that explores Late Arrivals

Source: FAA Aircraft registry, https://www.faa.gov/licenses_certificates/aircraft_certification/aircraft_registry/releasable_aircraft_download/

```
#install.packages("nycflights23")
library(nycflights23)
library(RColorBrewer)
data(flights)
data(airlines)
```

Create an initial scatterplot with loess smoother for distance to delays

Use “group_by” together with summarise functions

Remove observations with NA values from distand and arr_delay variables - notice number of rows changed from 336,776 to 327,346

Never use the function “na.omit”!!!!

```
flights_nona <- flights |>
  filter(!is.na(distance) & !is.na(arr_delay))
# remove na's for distance and arr_delay
```

Use group_by and summarise to create a summary table

The table includes, counts for each tail number, mean distance traveled, and mean arrival delay

```
by_tailnum <- flights_nona |>
  group_by(tailnum) |>  # group all tailnumbers together
  summarise(count = n(),   # counts totals for each tailnumber
            dist = mean(distance), # calculates the mean distance traveled
            delay = mean(arr_delay)
            ) # calculates the mean arrival delay
head(by_tailnum)
```

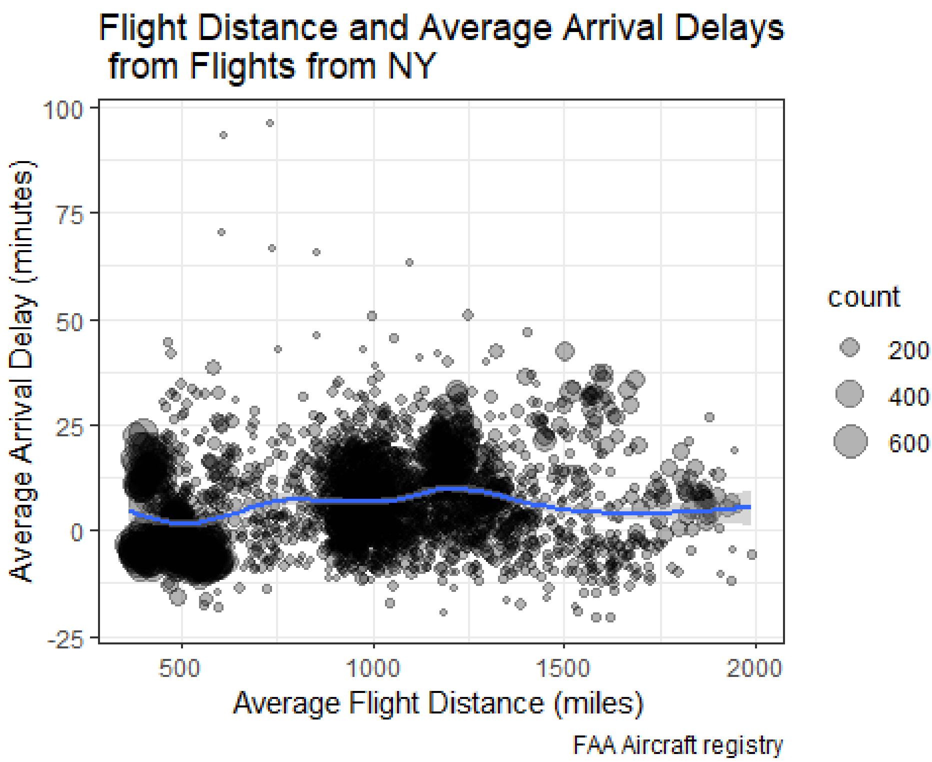
```
# A tibble: 6 × 4
  tailnum count  dist  delay
  <chr>   <int> <dbl>  <dbl>
1 190NV      29  597.  -9.24
2 191NV       6  583   -6.67
3 193NV      18  626. -13.8
4 195NV       2  587   -0.5
5 196NV       3  605  -11.3
6 202NV      17  597. -14.1
```

```
delay <- filter(by_tailnum, count > 20, dist < 2000)
# only include counts > 20 and distance < 2000 mi
```

Average delay is only slightly related to average distance flown by a plane.

```
ggplot(delay, aes(dist, delay)) +
  geom_point(aes(size = count), alpha = .3) +
  geom_smooth() +
  scale_size_area() +
  theme_bw() +
  labs(x = "Average Flight Distance (miles)",
       y = "Average Arrival Delay (minutes)",
       caption = "FAA Aircraft registry",
       title = "Flight Distance and Average Arrival Delays \n from Flights from NY")
```

```
`geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```



Heat Tree Stream Alluvial - Part 2

R Saidi

Streamgraphs

```
library(treemap)
library(tidyverse)
library(RColorBrewer)
library(webshot2)
```

Streamgraphs (unfortunately do not render to rpubs)

This type of visualisation is a variation of a stacked area graph, but instead of plotting values against a fixed, straight axis, a streamgraph has values displaced around a varying central baseline. Streamgraphs display the changes in data over time of different categories through the use of flowing, organic shapes that somewhat resemble a river-like stream. This makes streamgraphs aesthetically pleasing and more engaging to look at.

The size of each individual stream shape is proportional to the values in each category. The axis that a streamgraph flows parallel to is used for the timescale. Color can be used to either distinguish each category or to visualize each category's additional quantitative values through varying the color shade.

What are streamgraphs good for?

Streamgraphs are ideal for displaying high-volume datasets, in order to discover trends and patterns over time across a wide range of categories. For example, seasonal peaks and troughs in the stream shape can suggest a periodic pattern. A streamgraph could also be used to visualize the volatility for a large group of assets over a certain period of time.

The downside to a streamgraph is that they suffer from legibility issues, as they are often very cluttered. The categories with smaller values are often drowned out to make way for categories with much larger values, making it impossible to see all the data. Also, it's impossible to read the exact values visualized, as there is no axis to use as a reference.

Streamgraph code

The code for making streamgraphs has changed with new updates to R. You have to download and install Rtools40 from the link, https://cran.rstudio.com/bin/windows/Rtools/. and then used the code provided below.

Load devtools and libraries to create the following streamgraphs

install the package "devtools" also run the line: devtools::install_github("hrbrmstr/streamgraph") , then comment it out.

```
#install "devtools" (as a package)
devtools::install_github("hrbrmstr/streamgraph")
library(streamgraph)  # install "streamgraph" as a package
library(babynames)  # install "babynames"
data(babynames)
```

Now look at the babynames dataset

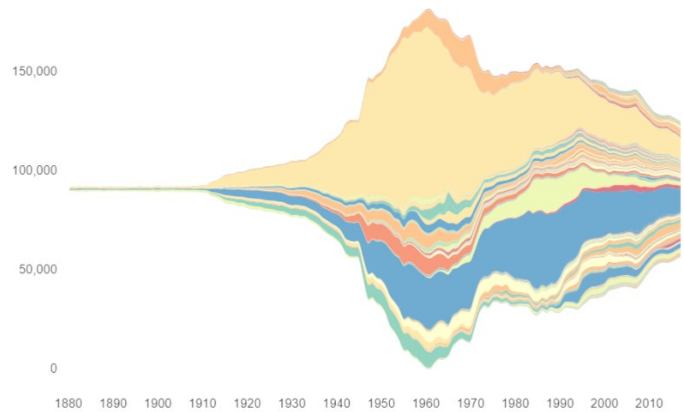
```
ncol(babynames)

[1] 5

head(babynames)

# A tibble: 6 × 5
   year sex   name          n   prop
  <dbl> <chr> <chr>     <int>  <dbl>
1  1880 F     Mary       7065 0.0724
2  1880 F     Anna       2604 0.0267
3  1880 F     Emma       2003 0.0205
4  1880 F     Elizabeth  1939 0.0199
5  1880 F     Minnie     1746 0.0179
6  1880 F     Margaret   1578 0.0162

summary(babynames$year)

   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1880    1951    1985    1975    2003    2017
```

Babynames streamgraph

Mouse over the colors and years to look at the pattern of various names

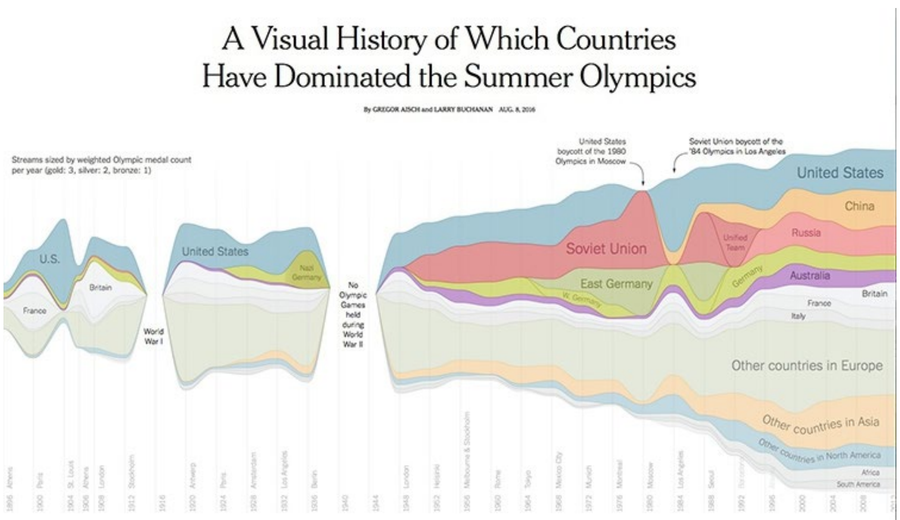
```
babynames |>
  filter(grepl("^Jo", name)) |>
  group_by(year, name) |>
  tally(wt=n) |>
  streamgraph("name", "n", "year")
```

```
babynames |>
  filter(grepl("^Da", name)) |>
  group_by(year, name) |>
  tally(wt=n) |>
  streamgraph("name", "n", "year")
```



Learn from the experts (a streamgraph)

Over the coming weeks and beyond, make a habit of looking for innovative graphics, especially those employing unusual chart forms that communicate the story from data in an effective way. Work out how they use visual cues to encode data. Here are a couple of examples from *The New York Times* to get you started. Follow the links from the source credits to explore the interactive versions:



(Source: *The New York Times*)

Your Assignment This Week 5

1. **(Ungraded)** Copy the Markdown code from these notes to explore how to create treemaps, heatmaps, streamgraphs, and alluvials. You can publish your RMD or Quarto file in Rpubs.

2. **(Worth up to 15 points)** NYC Flights Homework

   This week, you will create your first visualization on your own using the pre-built dataset, nycflights23. Load the libraries and view the "flights" dataset

```
library(tidyverse)
library(nycflights23)
```

   Now create one data visualization with this dataset. Your assignment is to create one plot to visualize one aspect of this dataset. The plot may be any type we have covered so far in this class (bargraphs, scatterplots, boxplots, histograms, treemaps, heatmaps, streamgraphs, or alluvials)

   Requirements for the plot:

   a. Include at least one dplyr command (filter, sort, summarize, group_by, select, mutate, ….)
   b. Include labels for the x- and y-axes
   c. Include a title and caption for the data source
   d. Your plot must incorporate at least 2 colors
   e. Include a legend that indicates what the colors represent
   f. Write a brief paragraph that describes the visualization you have created and at least one aspect of the plot that you would like to highlight.

Start early so that if you do have trouble, you can email me with questions.

Submit this assignment in the assignment dropbox **by 11:59 pm on _____________.** ***You will present in class next week.***