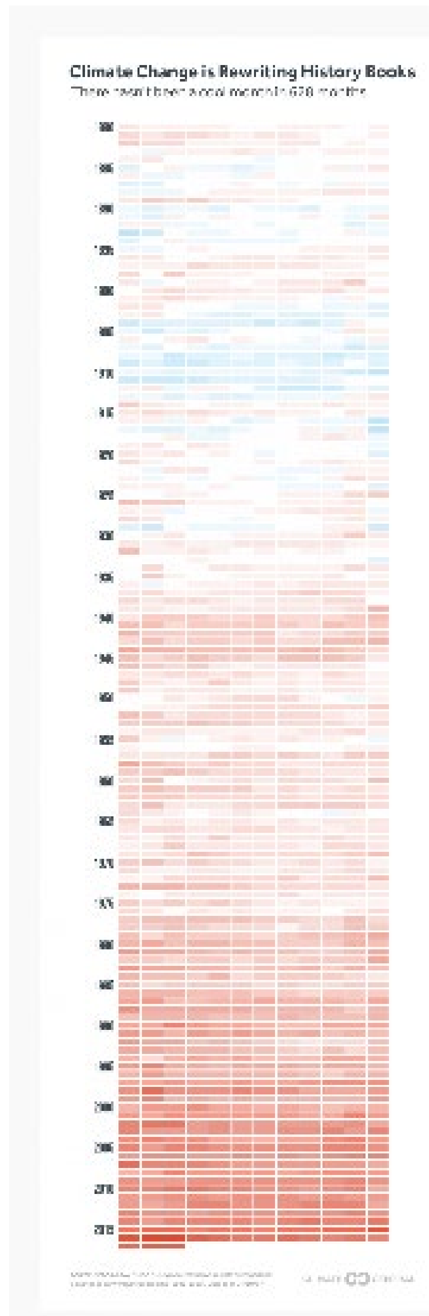
Study Guide Unit 5
DATA 110
Saidi

Treemaps, Heatmaps, Streamgraphs, and Alluvials

- Intro about heatmaps
- Heatmaps
- Treemaps
- Intro to NYCFlights13 dataset
- Use group_by and summarize to create a table
- Use kable (from knitr) to create a nicely formatted table
- Streampgraphs
- Alluvials
- Week 5 Homework Assignment

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/

Heatmaps, Treemaps, Streamgraphs and Alluvials

Rachel Saidi

2/18/21

So many ways to visualize data

Load the packages and the data from flowingdata.com website

The data is a csv file that compares number of views, number of comments to various categories of Yau's visualization creations

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

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.

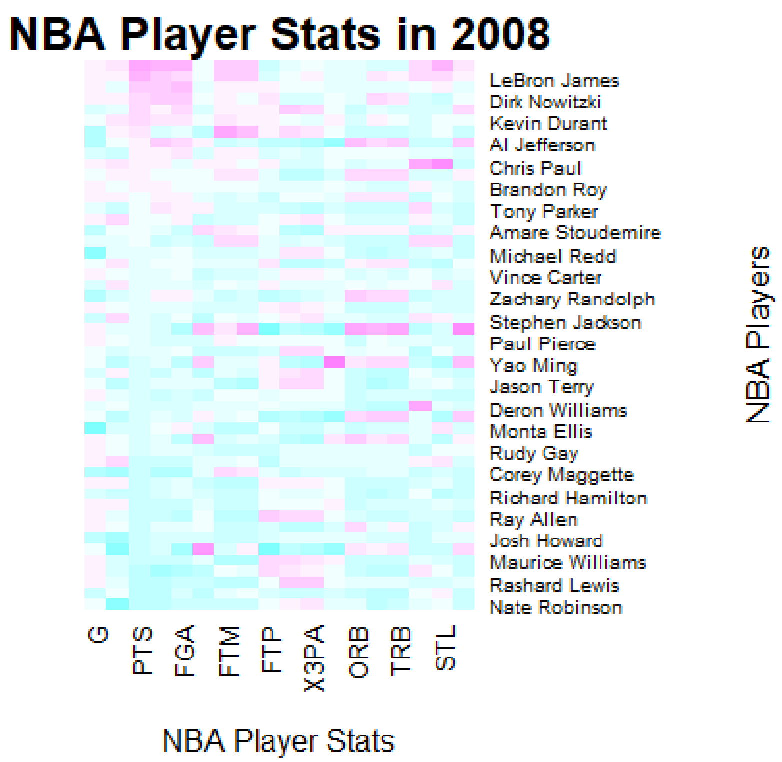
```
nba <- read.csv("http://datasets.flowingdata.com/ppg2008.csv")
#apparently you have to use read.csv here instead of read_csv
head(nba)
```

```
            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(20),
                        scale="column",
                        margins=c(5,10),
                        xlab = "NBA Player Stats",
                        ylab = "NBA Players",
                        main = "NBA Player Stats in 2008")
```

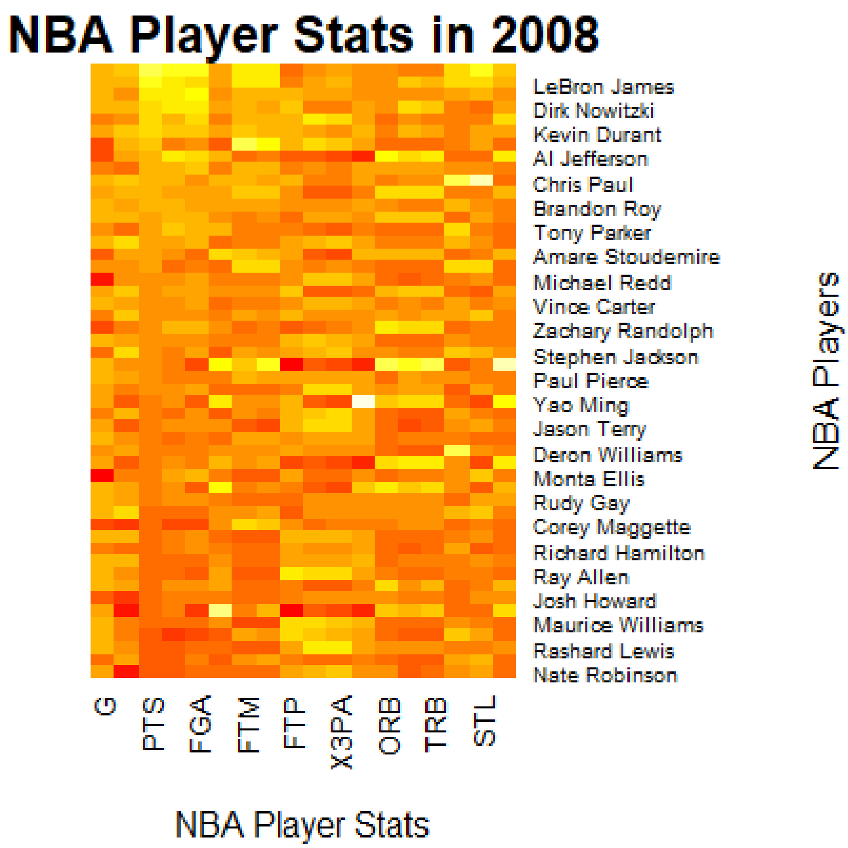


Improve/update the heatmap

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)

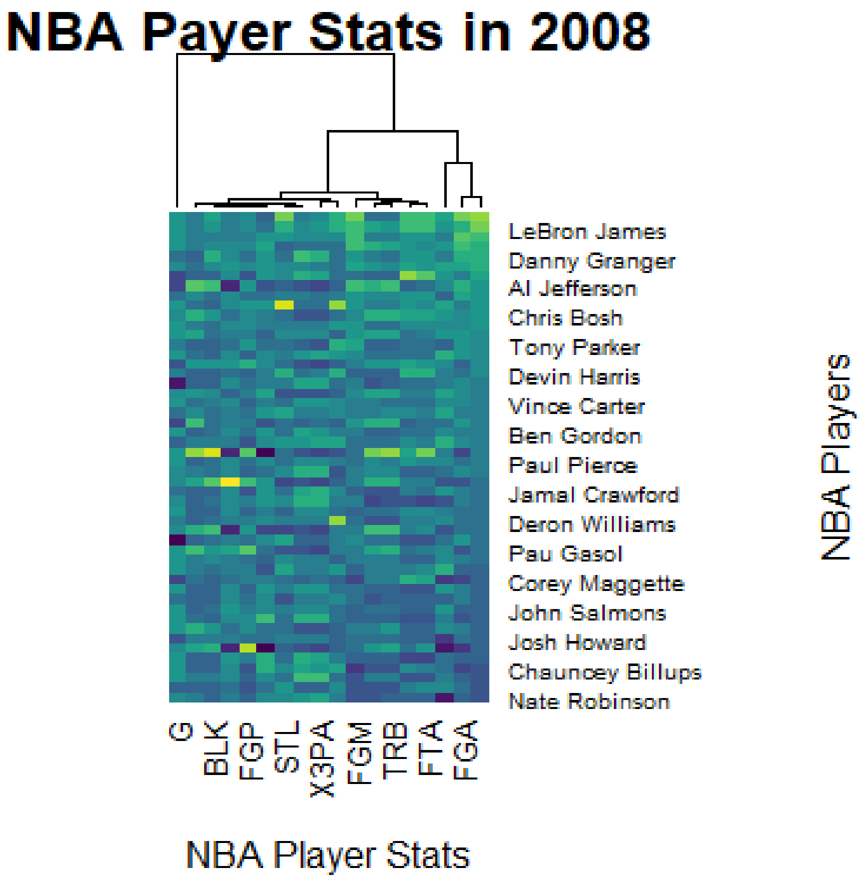
```
library(viridis)

Loading required package: viridisLite

# Loading required package: viridisLite

nba_heatmap <- heatmap(nba_matrix,
                       Rowv=NA,
                       col = viridis(20),
                       scale="column",
```

```
                   margins=c(5,10),
                   xlab = "NBA Player Stats",
                   ylab = "NBA Players",
                   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

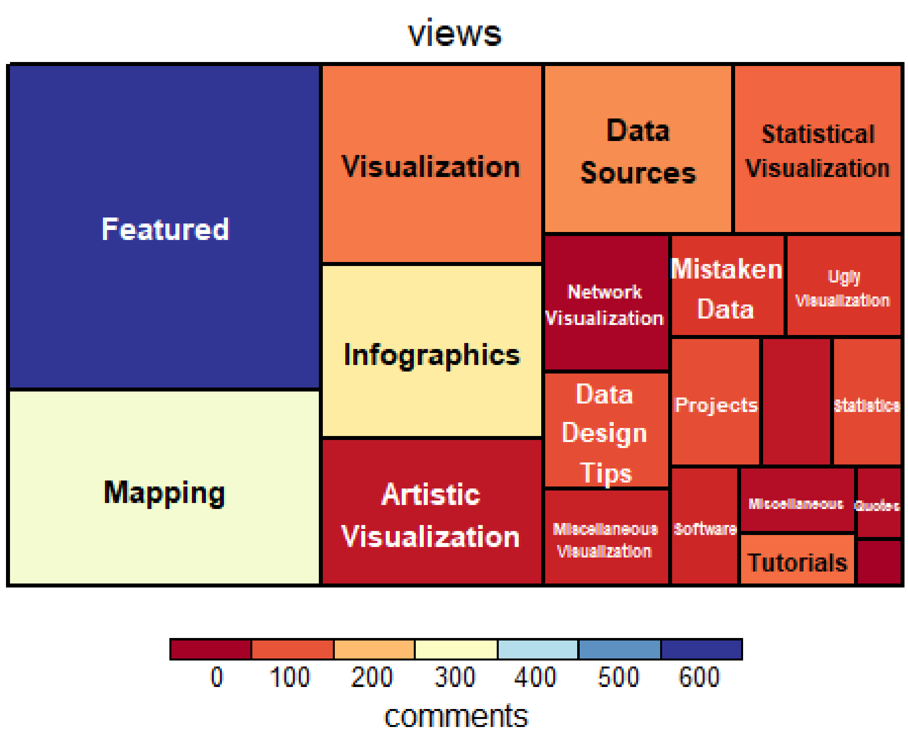
Load the data for creating a treemap from Nathan Yao's flowing data which explores number of views and comments for different categories of posts on his website.

```
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 *

Use the dataset NYCFlights13 to create a heatmap that explores Late Arrivals

```
#install.packages("nycflights13")
library(nycflights13)
library(RColorBrewer)
data(flights)
```

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

```
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

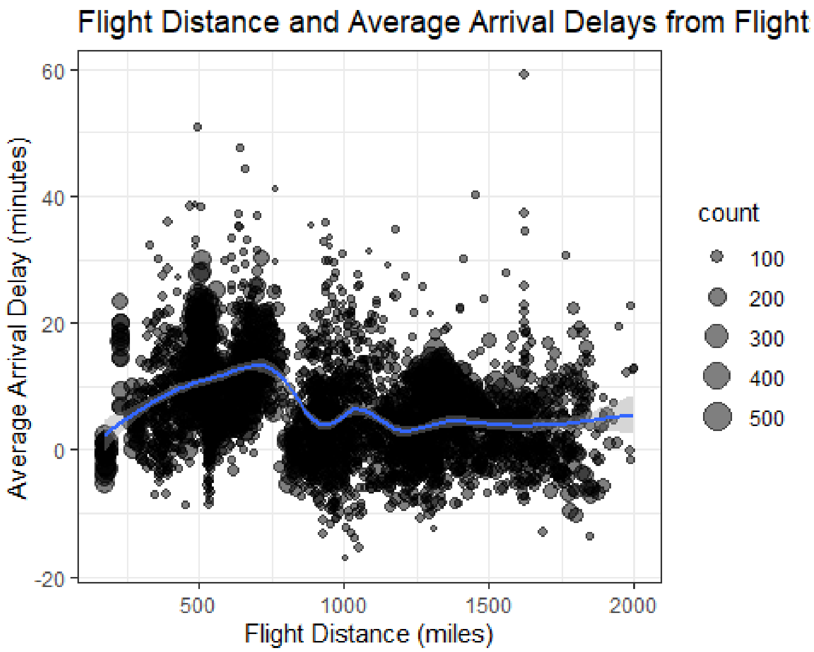
The table includes, counts for each tail number, mean distance traveled, and mean arrival delay

```
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
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 = 1/2) +
  geom_smooth() +
  scale_size_area() +
  theme_bw() +
  labs(x = "Flight Distance (miles)",
       y = "Average Arrival Delay (minutes)",
       title = "Flight Distance and Average Arrival Delays from Flights from NY")
```

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



Late Arrivals Affect the Usage Cost of Airports

This was modified from Raul Miranda's work Create a dataframe that is composed of summary statistics

```
delays <- flights_nona |> # create a delays dataframe by:
  group_by (dest) |>       # grouping by point of destination
  summarize (count = n(),
             # creating variables: number of flights to each destination,
             dist = mean (distance),
             # the mean distance flown to each destination,
             delay = mean (arr_delay),
             # the mean delay of arrival to each destination,
             delaycost = mean(count*delay/dist))
             # delay cost index defined as:
             # [(number of flights)*delay/distance] for a destination
delays <- arrange(delays, desc(delaycost))
# sort the rows by delay cost
head(delays)
```

```
# A tibble: 6 × 5
  dest  count  dist delay delaycost
  <chr> <int> <dbl> <dbl>     <dbl>
1 DCA    9111  211.  9.07      391.
2 IAD    5383  225. 13.9       332.
3 ATL   16837  757. 11.3       251.
4 BOS   15022  191.  2.91      230.
5 CLT   13674  538.  7.36      187.
6 RDU    7770  427. 10.1       183.
```

This shows Reagan National (DCA) and Dulles with the highest delay costs

Here is another way to display all destinations in the table using the knitr package with the function, kable

```
#install.packages("knitr")
library(knitr)
kable(delays[1:10,], # only show first 10 rows
      caption = "Table of Mean Distance, Mean Arrival Delay,
      and Highest Delay Costs",
      digits = 2)  # round values to 2 decimal places
```

Table of Mean Distance, Mean Arrival Delay, and Highest Delay Costs

dest	count	dist	delay	delaycost
DCA	9111	211.08	9.07	391.36
IAD	5383	224.74	13.86	332.08
ATL	16837	757.14	11.30	251.29
BOS	15022	190.74	2.91	229.53
CLT	13674	538.01	7.36	187.07
RDU	7770	426.73	10.05	183.04
RIC	2346	281.27	20.11	167.74
PHL	1541	94.34	10.13	165.42
BUF	4570	296.87	8.95	137.71
ORD	16566	729.02	5.88	133.54

Now get the top 100 delay costs to create a heatmap of those flights.

```
top100 <- delays |> # select the 100 largest delay costs
  head(100) |>
  arrange(delaycost) # sort ascending - heatmap displays descending costs
row.names(top100) <- top100$dest

Warning: Setting row names on a tibble is deprecated.

# rename the rows according to destination airport codes
```

In order to make a heatmap, convert the dataframe to matrix form

```
delays_mat <- data.matrix(top100)
# convert delays dataframe to a matrix  (required by heatmap)
delays_mat2 <- delays_mat[,2:5]
# remove the redundant column of destination airport codes
```

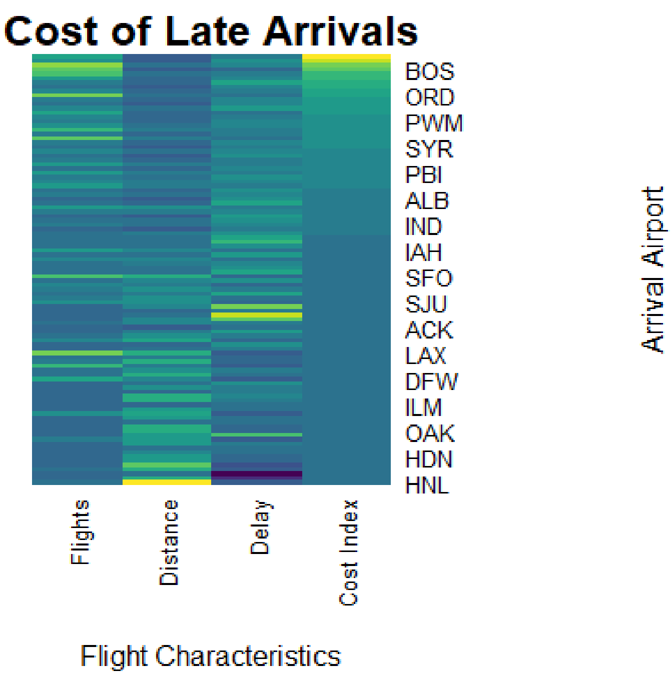
Create a heatmap using colorBrewer

color set, margins=c(7,10) for aspect ratio, titles of graph, x and y labels,font size of x and y labels, and set up a RowSideColors bar

Flights Plot

```
heatmap(delays_mat2,
        Rowv = NA, Colv = NA,
        col= viridis(25),
        s=0.6,
        v=1,
        scale="column",
        margins=c(7,10),
        main = "Cost of Late Arrivals",
        xlab = "Flight Characteristics",
        ylab="Arrival Airport",
        labCol = c("Flights","Distance","Delay","Cost Index"),
        cexCol=1,
        cexRow =1)
```

```
layout: widths =  0.05 4 , heights =  0.25 4 ; lmat=
     [,1] [,2]
[1,]    0    3
[2,]    2    1
```



What did this heatmap show?

“Cost index” is defined as a measure of how arrival delays impact the cost of flying into each airport and is calculated as number of flights * mean delay / mean flight distance. For airlines it is a measure of how much the cost to fly to an airport increases due to frequent delays of arrival. Cost index is inversely proportional to distance because delays affect short flights more than long flights and because the profit per seat increases with distance due to the larger and more efficient planes used for longer distances.

The variance in delays across airports is mainly due to (a) airline traffic congestion relative to the airport size; and (b)regional climate and weather events. It is not strongly dependent upon airline carrier or tailnumber.

Therefore, airports such as ORD and BOS have high cost index because they are highly congested and are frequently delayed due to weather. Airports like IAD, PHL, DTW, etc., are very congested despite their large size and also show high cost index. Smaller airports such as HDN, SNA, HNL, LEX, etc., have null to slightly negative cost index because they are not congested and keep flights on time.

Streamgraphs

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

Now look at the babynames dataset

```
ncol(babynames)
```

```
[1] 5
```

```r
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
```

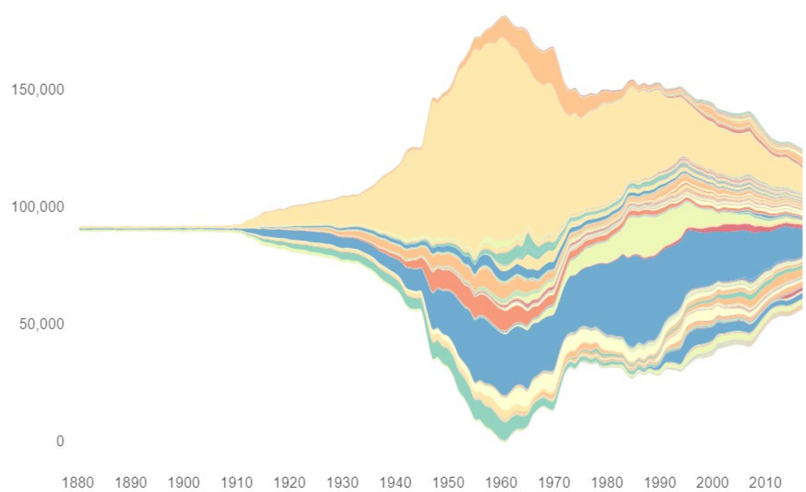
```r
str(babynames)
```

```
tibble [1,924,665 × 5] (S3: tbl_df/tbl/data.frame)
 $ year: num [1:1924665] 1880 1880 1880 1880 1880 1880 1880 1880 1880 1880 ...
 $ sex : chr [1:1924665] "F" "F" "F" "F" ...
 $ name: chr [1:1924665] "Mary" "Anna" "Emma" "Elizabeth" ...
 $ n   : int [1:1924665] 7065 2604 2003 1939 1746 1578 1472 1414 1320 1288 ...
 $ prop: num [1:1924665] 0.0724 0.0267 0.0205 0.0199 0.0179 ...
```

Babynames streamgraph

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

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



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)
```

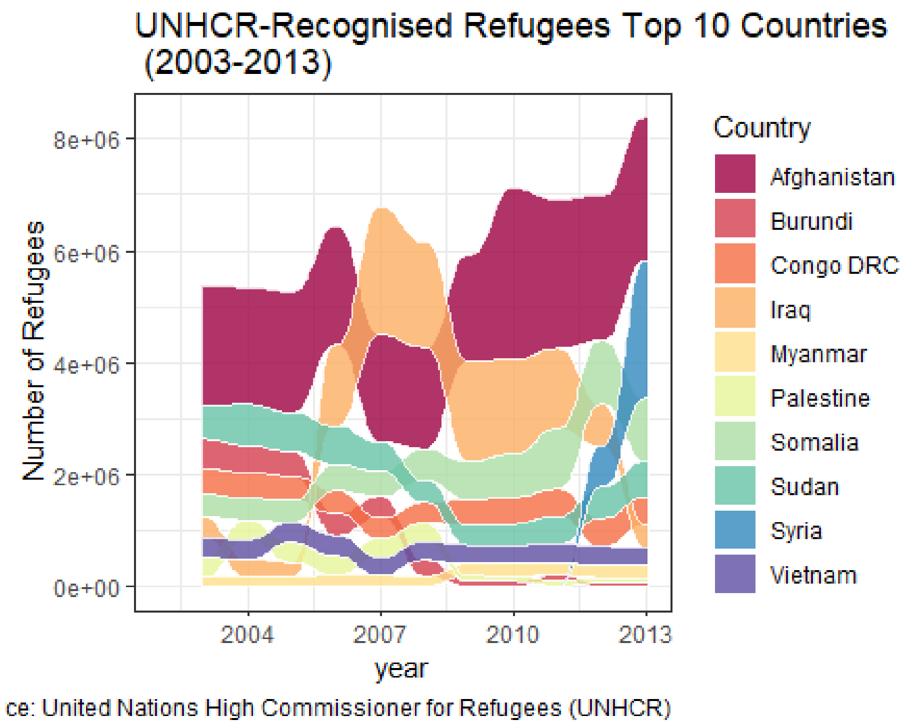
Show UNHCR-recognised refugees

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

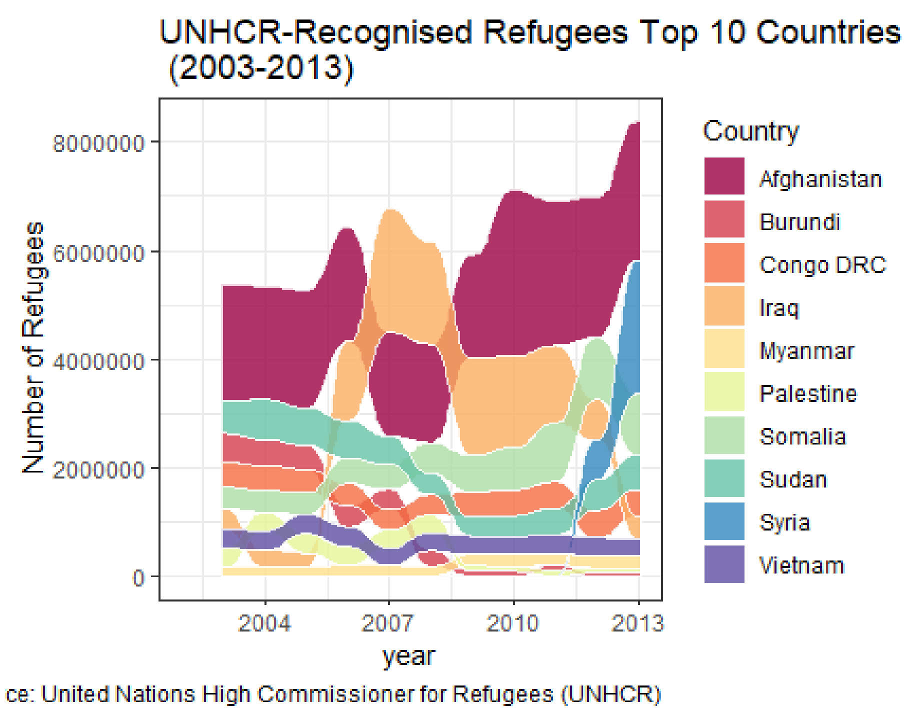
```
ggalluv
```



A final touch to fix the y-axis scale

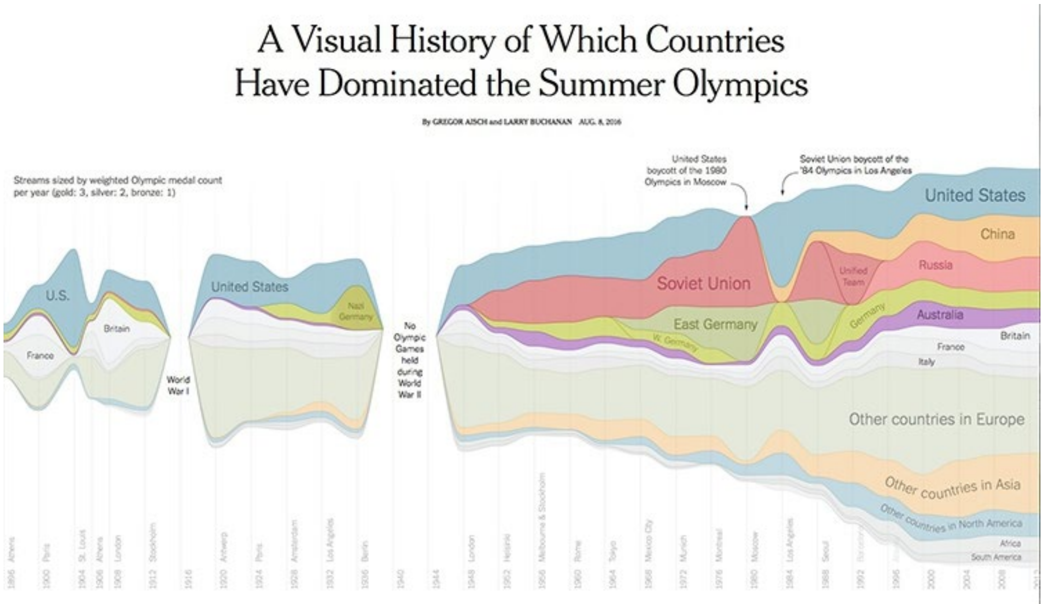
Notice the y-values are in scientific notation. We can convert them to standard notation with options scipen function

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



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, `nycflights13`. Load the libraries and view the "flights" dataset

```
library(tidyverse)
library(nycflights13)
```

   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
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 Tuesday, ___.** You will present your visualization in class on Wednesday.