**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/

## Heatmaps Treemaps and Alluvials

# So many ways to visualize data

*https://www.travelsavvy.agency/blog/what-airlines-fly-to-los-angeles*

# Use the dataset NYCFlights23 to explore 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(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✓ dplyr     1.1.4     ✓ readr     2.1.5
✓ forcats   1.0.0     ✓ stringr   1.5.1
✓ ggplot2   3.5.2     ✓ tibble    3.2.1
✓ lubridate 1.9.4     ✓ tidyr     1.3.1
✓ purrr     1.0.4
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to b
ecome errors
```

```
data(flights)
data(airlines)
```

# Create an initial scatterplot with loess smoother for distance to delays

Use “group_by” together with summarize functions

**Never use the function “na.omit”!!!!**

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

# Use group_by and summarise to create a summary table

The table includes counts for each destination, mean distance traveled, mean arrival delay, and mean departure delay

```
by_dest <- flights_nona |>
  group_by(dest) |>  # group all destinations
  summarise(count = n(),   # counts totals for each destination
            avg_dist = mean(distance), # calculates the mean distance traveled
            avg_arr_delay = mean(arr_delay),  # calculates the mean arrival delay
            avg_dep_delay = mean(dep_delay), # calculates the mean dep delay
            .groups = "drop") |>  # remove the grouping structure after summarizing
  arrange(avg_arr_delay) |>
  filter(avg_dist < 3000)
head(by_dest)
```

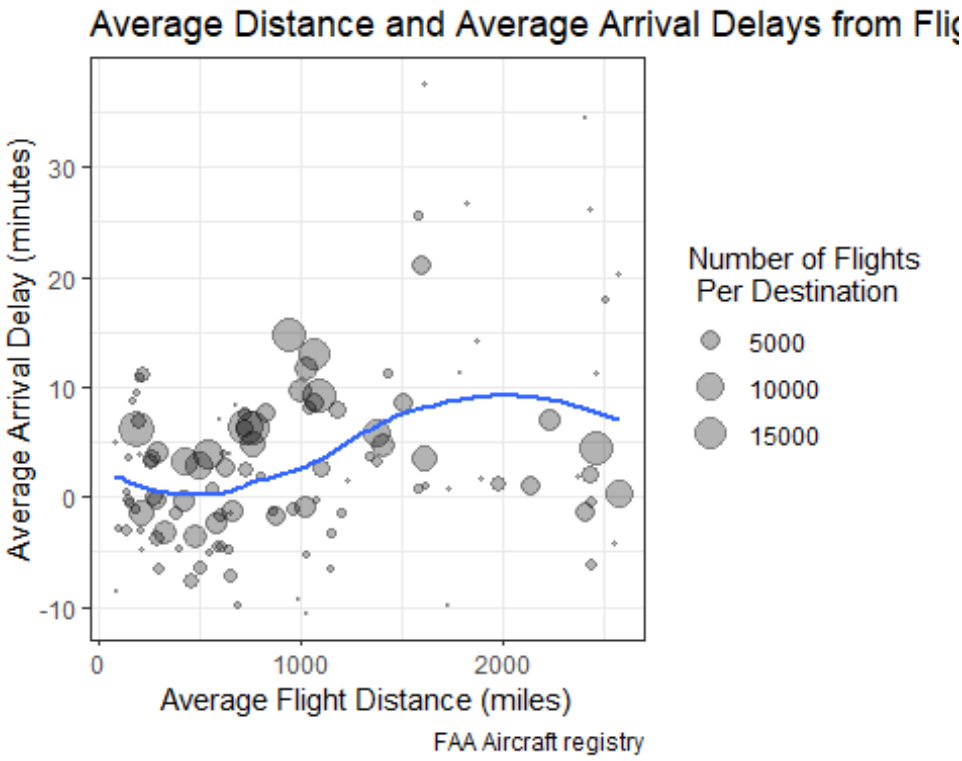
```
# A tibble: 6 × 5
  dest  count avg_dist avg_arr_delay avg_dep_delay
  <chr> <int>    <dbl>         <dbl>         <dbl>
1 PNS      71    1030         -10.6         -1.24
2 HHH     461     695.         -9.95         1.38
3 HDN      27    1728          -9.93         8.78
4 VPS     107     988          -9.41         2.62
5 AVP     140      93          -8.53        -0.957
6 GSO    2857     456.         -7.77         3.81
```

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

Show a scatterplot of distance versus

```
ggplot(by_dest, aes(avg_dist, avg_arr_delay)) +
  geom_point(aes(size = count), alpha = .3) +
  geom_smooth(se = FALSE) + # remove the error band
  scale_size_area() +
  theme_bw() +
  labs(x = "Average Flight Distance (miles)",
       y = "Average Arrival Delay (minutes)",
       size = "Number of Flights \n Per Destination",
       caption = "FAA Aircraft registry",
       title = "Average Distance and Average Arrival Delays from Flights from NY")
```

```
`geom_smooth()` using method = 'loess' and formula = 'y ~ x'
```




# Heatmaps

A heatmap is a 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)

# Heatmap of average departure delays, arrival delays, distance and flight times

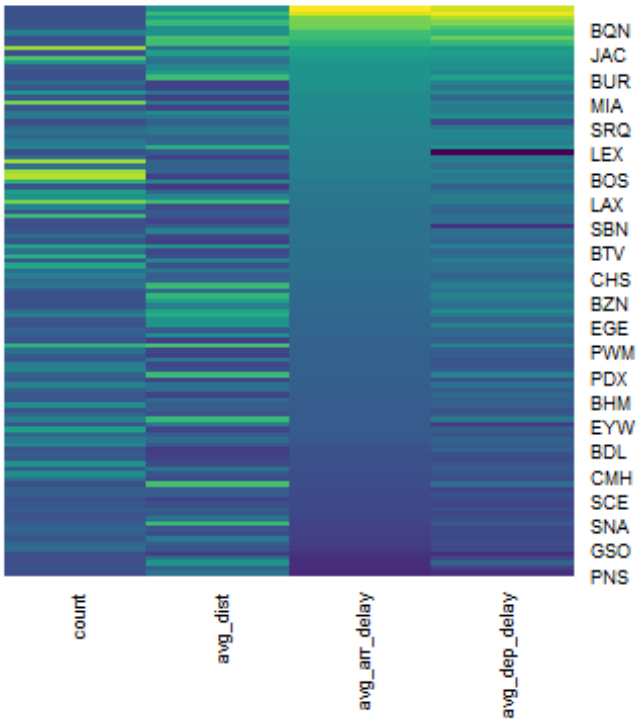
```
by_dest_matrix <- data.matrix(by_dest[, -1])  # drop dest from matrix so it won't show in heatmap
row.names(by_dest_matrix) <- by_dest$dest     # restore row names

library(viridis)
```

```
Loading required package: viridisLite
```

```
by_dest_heatmap <- heatmap(by_dest_matrix,
                           Rowv=NA,
                           Colv=NA,
                           col = viridis(250),
                           cexCol = .7,  # shrink x-axis label size
                           scale="column",
```

```
                    xlab = "",
                    ylab = "",
                    main = "")
```




# Which 6 destination airports have the highest average arrival delay from NYC?

PSE - Ponce Mercedita Airport, PR ABQ - Albuquerque, NM BQN - Rafael Hernández Airport, PR SJC - San José Mineta, CA MCO - Orlando International, FL FLL - Fort Lauderdale International, FL

# 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)

# Join the delay_punctuality dataset with the airlines dataset

Also remove “Inc.” or “Co.” from the Carrier Name

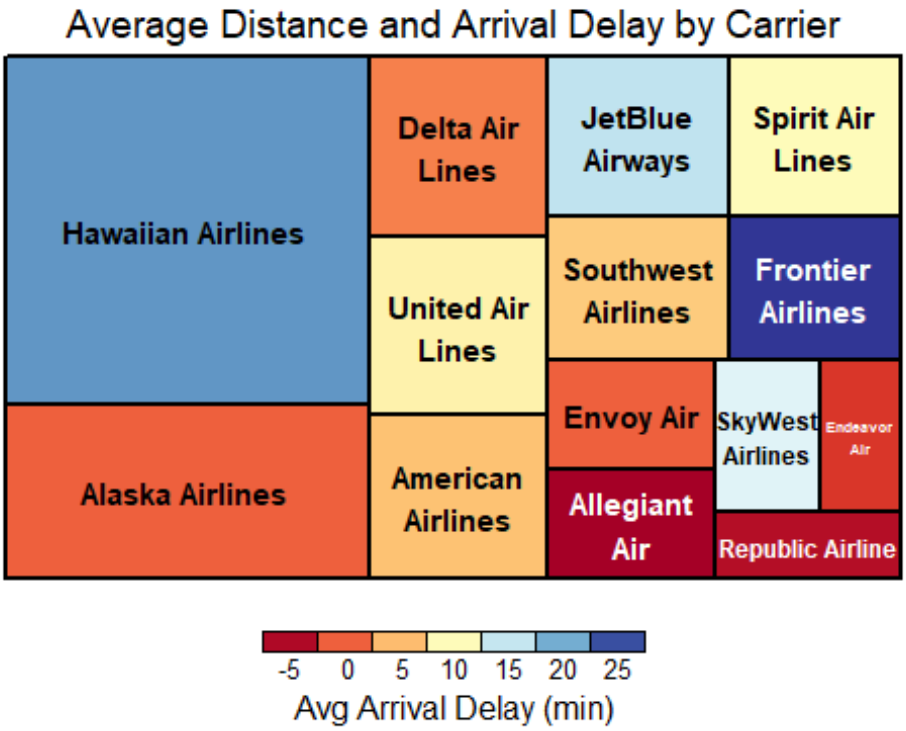
```
flights2 <- left_join(flights_nona, airlines, by = "carrier")
flights2$name <- gsub("Inc\\.|Co\\.", "", flights2$name)

# Convert months from numerical to abbreviated labels
flights3 <- flights2 |>
  group_by(name)|>
  summarise(avg_dist = mean(distance), # calculates the mean distance traveled
            avg_arr_delay = mean(arr_delay))  # calculates the mean arrival delay
#flights2$month_label <- month(flights2$month, label = TRUE, abbr = TRUE)
```

# Create a treemap for NYC FLights

- The index is a categorical variable - carrier
- The size of the box is by average distance
- The heatmap color is average arrival delay
- Notice how the treemap includes a legend for average arrival delay

```
library(RColorBrewer)
library(treemap)
treemap(flights3,
        index="name",
        vSize="avg_dist",
        vColor="avg_arr_delay",
        type="manual",
        palette="RdYlBu",  #Use RColorBrewer palette
        title = "Average Distance and Arrival Delay by Carrier",  # plot title
        title.legend = "Avg Arrival Delay (min)" )   # legend label
```




# Graph On-Time Performance using Departure Delay and Arrival Delay

Some of the most important data that is collected for reporting is to analyze key performance indicators (KPIs) and the subset that agencies look at the most is “On-Time Performance” (OTP) which is usually defined as arriving at the origin location within 15 minutes of the requested/scheduled pickup time. The following code will create a bidirectional bar graph that has both the departure delay percentage and arrival delay percentage for each carrier.

# Calculate the percentage of flights with less than 15 minutes delay (OTP)

```
delay_OTP <- flights2 |>
  group_by(name) |>
  summarize(Departure_Percentage = sum(dep_delay <= 15)
            / n() * 100,
            Arrival_Percentage = sum(arr_delay <= 15) / n() * 100)
```

# Create a bidirectional horizontal bar chart

```
ggplot(delay_OTP, aes(x = -Departure_Percentage, y = reorder(name, Departure_Percentage))
) +
  geom_text(aes(label = paste0(round(Departure_Percentage, 0), "%")),
            hjust = 1.1, size = 3.5) +  #departure % labels
  geom_bar(aes(fill = "Departure_Percentage"), stat = "identity", width = .75) +
```

```
  geom_bar(aes(x = Arrival_Percentage, fill = "Arrival_Percentage"),
           stat = "identity", width = .75) +
  geom_text(aes(x = Arrival_Percentage, label = paste0(round(Arrival_Percentage, 0), "%")
),
            hjust =-.1, size = 3.5) +  # arrival % labels

  labs(x = "Departures < On-Time Performance > Arrivals",
       y = "Carrier",
       title = "On-Time Performance of Airline Carriers \n (Percent of Flights < 15 Minut
es Delay)",
       caption = "Source: FAA") +

  scale_fill_manual(
    name = "Performance",
    breaks = c("Departure_Percentage", "Arrival_Percentage"),  # Specify the order of leg
end items
    values = c("Departure_Percentage" = "#8bd3c7", "Arrival_Percentage" = "#beb9db"),
    labels = c("Departure_Percentage" = "Departure", "Arrival_Percentage" = "Arrival")
  ) +

  scale_x_continuous(labels = abs, limits = c(-120, 120)) +  # Positive negative axis
  theme_minimal()
```

# 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

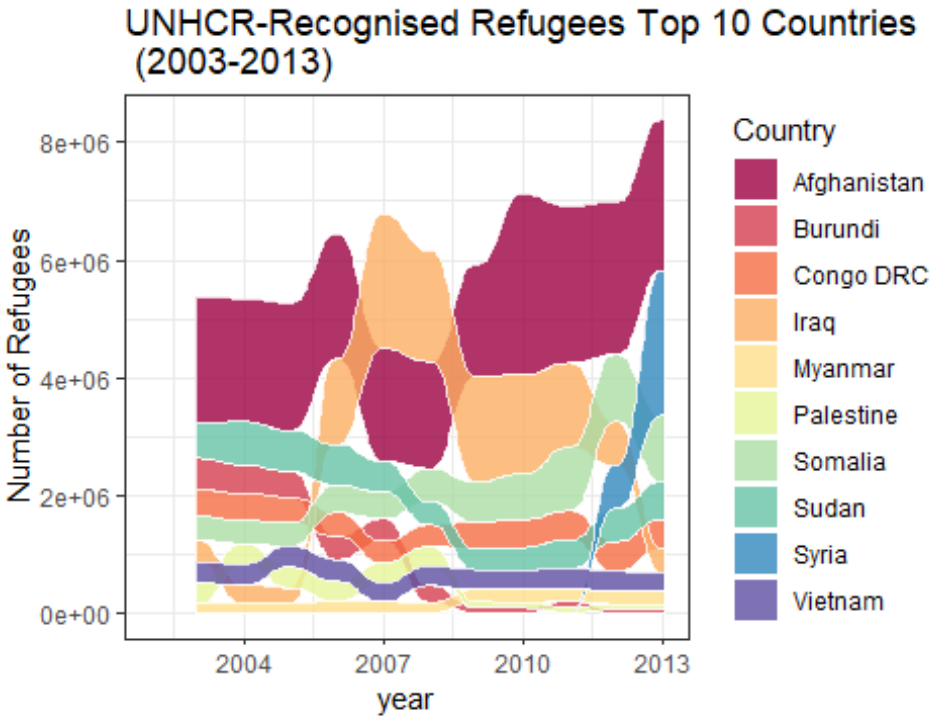
Top 10 most affected countries causing refugees from 2003-2013 Alluvials need the variables: *time-variable, value, category*

```
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
```

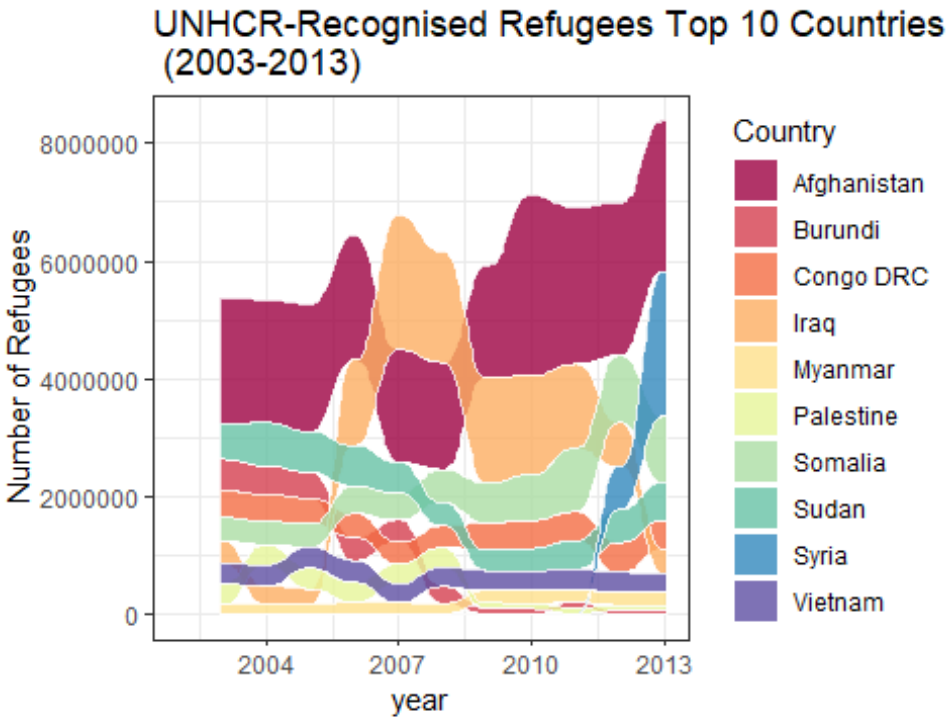



ce: United Nations High Commissioner for Refugees (UNHCR)

# 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
```




ce: United Nations High Commissioner for Refugees (UNHCR)

# Streamgraphs - 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)
```

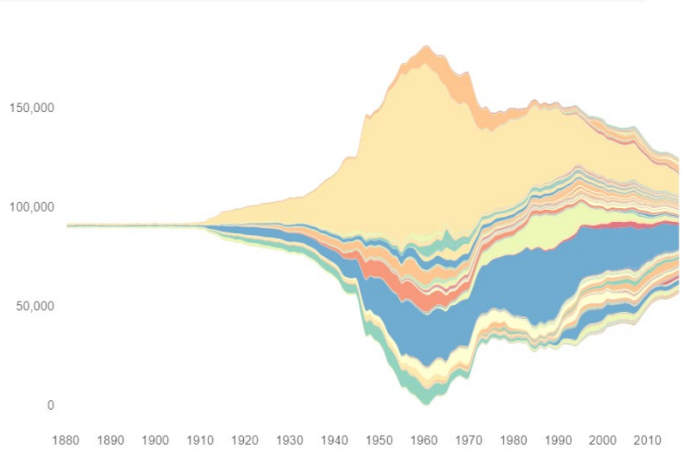
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
   1880    1951    1985    1975    2003    2017
```

# Babynames streamgraph

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

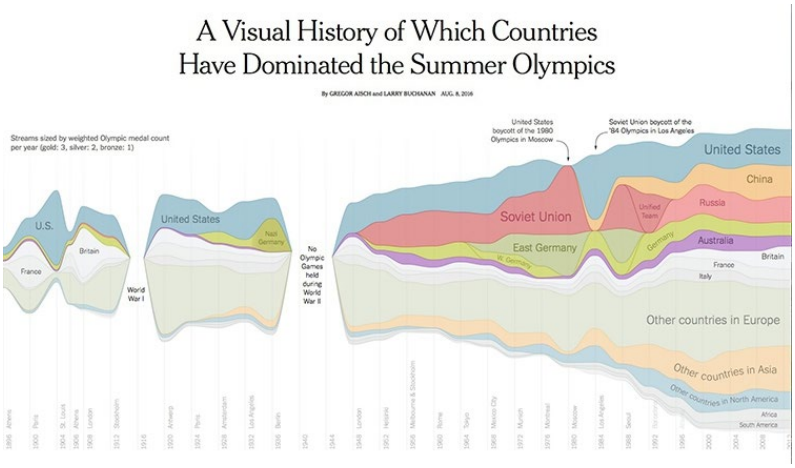
```
babynames |>
  filter(grepl("^Jo", name)) |>
  group_by(year, name) |>
  tally(wt=n) |>
  streamgraph("name", "n", "year")
```

```
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)
data(flights)
data(airlines)
```

   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*. If your visualization includes any 2-letter carriers or 3-letter airports, please convert them to their actual names.

Submit this assignment in the assignment dropbox **by 11:59 pm on Sunday, March 2nd.** ***You will present in class next week.***