Study Guide
DATA 110
Saidi

Working with Continuous Variables with DS Labs and HighCharter

Disease and Democracy

This data attempts to illustrate a controversial theory suggesting that the emergence of democratic political systems has depended largely on nations having low rates of infectious disease, from the Global Infectious Diseases and Epidemiology Network and *Democratization: A Comparative Analysis of 170 Countries*

Load the required packages and dataset from our course datasets link

Load the required packages and dataset from our course datasets link

```
# load required packages
library(tidyverse)
# load disease and democracy data
setwd("C:/Users/rsaidi/Dropbox/Rachel/MontColl/Datasets/Datasets")
disease_democ <- read_csv("disease_democ.csv")
```

Change the font

R's basic fonts are fairly limited (run: names(postscriptFonts())) to view those available). Using extrafont in three easy steps The first step is to install extrafont, and then import the fonts from your system into the extrafont database:

```
# R's basic fonts are fairly limited. View those available by running this code
names(postscriptFonts())
```

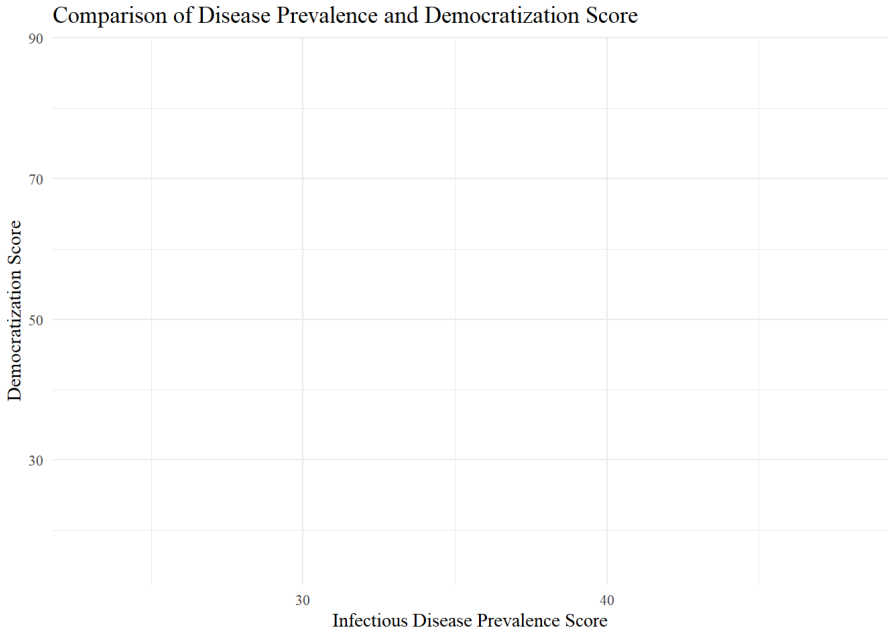
```
 [1] "serif"               "sans"                "mono"
 [4] "AvantGarde"          "Bookman"             "Courier"
 [7] "Helvetica"           "Helvetica-Narrow"    "NewCenturySchoolbook"
[10] "Palatino"            "Times"               "URWGothic"
[13] "URWBookman"          "NimbusMon"           "NimbusSan"
[16] "URWHelvetica"        "NimbusSanCond"       "CenturySch"
[19] "URWPalladio"         "NimbusRom"           "URWTimes"
[22] "URW2Helvetica"       "URW2HelveticaItalic" "URW2Times"
[25] "NimbusMonoPS"        "ArialMT"             "ComputerModern"
[28] "ComputerModernItalic" "Japan1"             "Japan1HeiMin"
[31] "Japan1GothicBBB"     "Japan1Ryumin"        "Korea1"
[34] "Korea1deb"           "CNS1"                "GB1"
```

```
# install.packages("extrafont")
library(extrafont)
```

Change the theme

The default gray theme of ggplot2 has a rather academic look. You can use one of the ggplot2 built-in themes, and then customize the fonts.

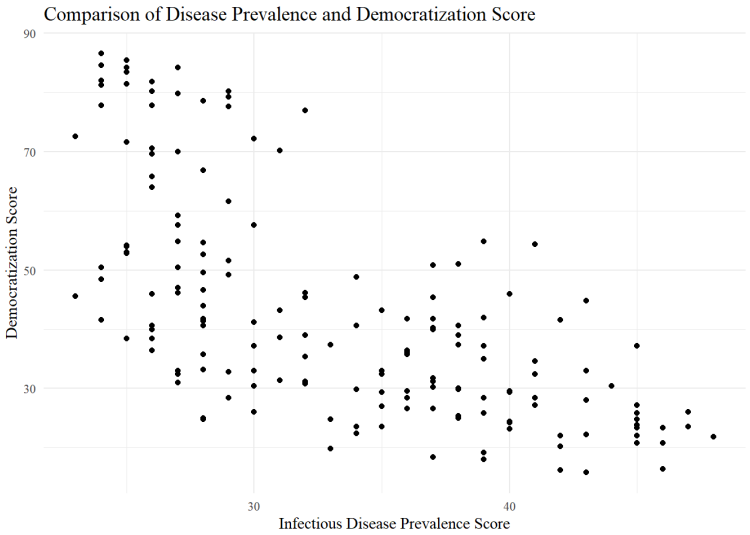
```
disease_democ_chart <-ggplot(disease_democ, aes(x = infect_rate, y = democ_score)) +
  labs(x="Infectious Disease Prevalence Score",
       y="Democratization Score",
       title="Comparison of Disease Prevalence and Democratization Score") +
  theme_minimal(base_family = "serif")
disease_democ_chart
```



Add a layer with points

This code will add a geom layer with points to the template:

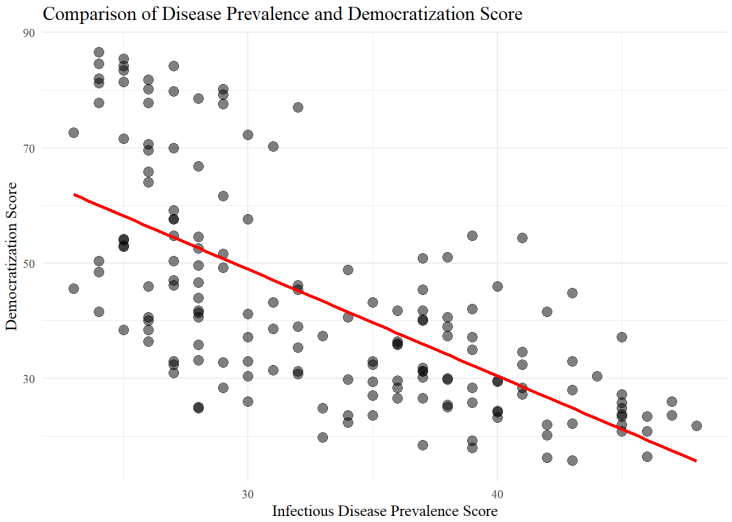
```
disease_democ_chart +
  geom_point()
```



Customize the two layers we've added to the chart

The following code modifies the two geom layers to change their appearance.

```
disease_democ_chart +
  geom_point(size = 3, alpha = 0.5) +
  geom_smooth(method = lm, se=FALSE, color = "red")
```

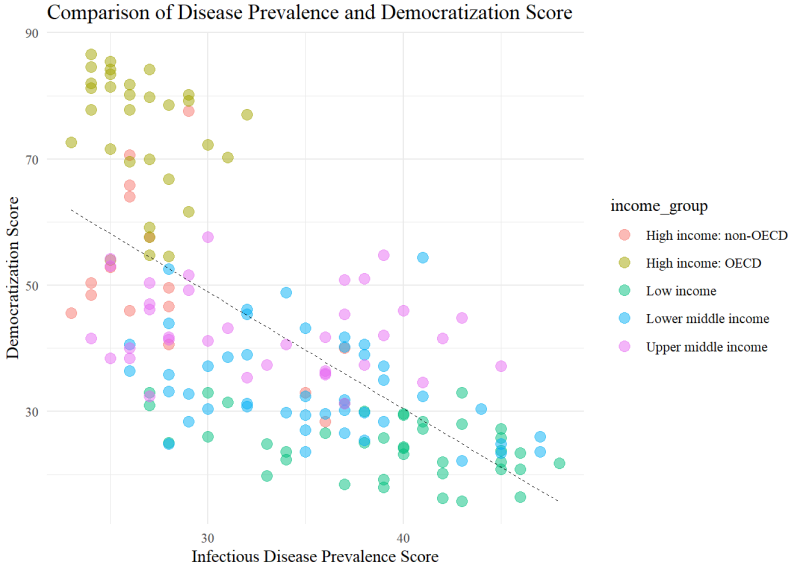


Customize again, coloring the points by income group

You can make a dashed line by: linetype = “dotdash”, or equivalently, lty = 2

You can make a dashed line by: linetype = “dotdash”, or equivalently, lty = 2

```
disease_democ_chart +
  geom_point(size = 3, alpha = 0.5, aes(color = income_group)) +
  geom_smooth(method = lm, se  =FALSE, color = "black", lty = 2, linewidth = 0.3)
```



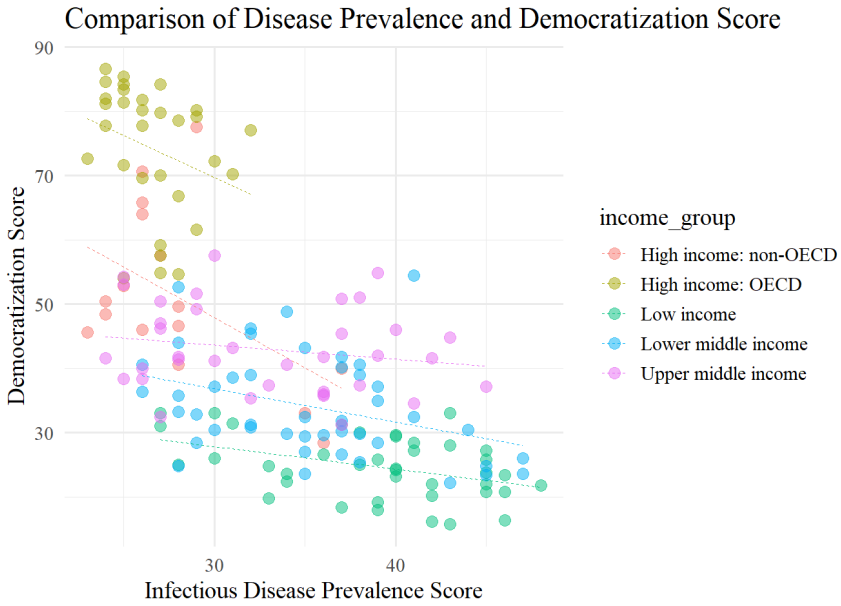
Color the entire chart by income group

Notice how the aes function colors the points by values in the data, rather than setting them to a single color. ggplot2 recognizes that income_group is a categorical variable, and uses its default qualitative color palette.

Now run this code, to see the different effect of setting the aes color mapping for the entire chart, rather than just one geom layer.

```
ggplot(disease_democ, aes(x = infect_rate, y = democ_score, color=income_group)) +
  labs(x="Infectious Disease Prevalence Score",
       y="Democratization Score",
       title="Comparison of Disease Prevalence and Democratization Score") +
```

```
  theme_minimal(base_size = 14, base_family = "serif") +
  geom_point(size = 3, alpha = 0.5) +
  geom_smooth(method=lm, se=FALSE, lty = 2, linewidth = 0.3)
```

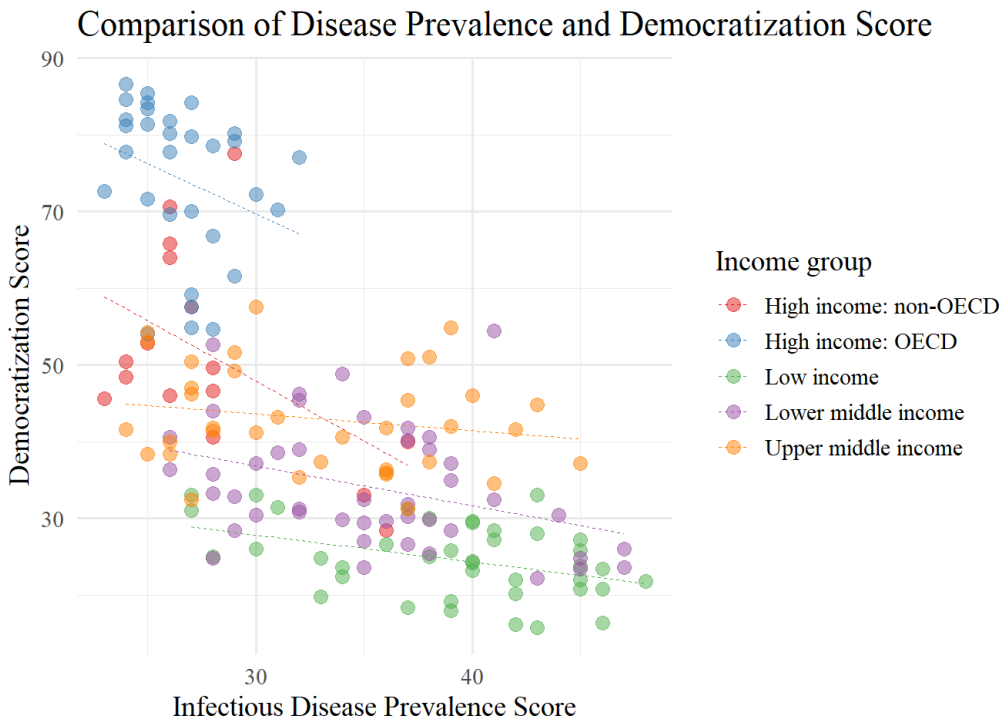


Because here we mapped the variable income group to color for the whole chart, and not just the points, it also affects the geom_smooth layer, so a separate trend line, colored the same as the points, is calculated for each income_group.

Set the axis ranges, and use a different color palette

You can apply ColorBrewer qualitative palettes by using the scale_color_brewer function. Add the text you want to appear as a legend title using name.

```
# set the axis ranges, change color palette
ggplot(disease_democ, aes(x = infect_rate, y = democ_score, color=income_group)) +
  labs(x="Infectious Disease Prevalence Score",
       y="Democratization Score",
       title="Comparison of Disease Prevalence and Democratization Score") +
  theme_minimal(base_size = 14, base_family = "serif") +
  geom_point(size = 3, alpha = 0.5) +
  geom_smooth(method=lm, se=FALSE, lty = 2, linewidth = 0.3)+
  scale_color_brewer(name="Income group", palette = "Set1")
```



Notice how slopes for each income group are different. What might that indicate?

DS Labs Datasets

Use the package DSLabs (Data Science Labs)

There are a number of datasets in this package to use to practice creating visualizations

```
# install.packages("dslabs")  # these are data science labs
library("dslabs")
```

```
Warning: package 'dslabs' was built under R version 4.2.2
```

```
data(package="dslabs")
list.files(system.file("script", package = "dslabs"))
```

```
 [1] "make-admissions.R"
 [2] "make-brca.R"
 [3] "make-brexit_polls.R"
 [4] "make-death_prob.R"
 [5] "make-divorce_margarine.R"
 [6] "make-gapminder-rdas.R"
 [7] "make-greenhouse_gases.R"
 [8] "make-historic_co2.R"
 [9] "make-mnist_27.R"
[10] "make-movielens.R"
[11] "make-murders-rda.R"
[12] "make-na_example-rda.R"
[13] "make-nyc_regents_scores.R"
[14] "make-olive.R"
[15] "make-outlier_example.R"
```

```
[16] "make-polls_2008.R"
[17] "make-polls_us_election_2016.R"
[18] "make-reported_heights-rda.R"
[19] "make-research_funding_rates.R"
[20] "make-stars.R"
[21] "make-temp_carbon.R"
[22] "make-tissue-gene-expression.R"
[23] "make-trump_tweets.R"
[24] "make-weekly_us_contagious_diseases.R"
[25] "save-gapminder-example-csv.R"
```

Note that the package dslabs also includes some of the scripts used to wrangle the data from their original source:

US murders

This dataset includes gun murder data for US states in 2010. I use this dataset to introduce the basics of R program.

```
data("murders")
library(tidyverse)
library(ggthemes)
```

```
Warning: package 'ggthemes' was built under R version 4.2.2
```

```
library(ggrepel)
```

```
Warning: package 'ggrepel' was built under R version 4.2.2
```

```
#view(murders)
write_csv(murders, "murders.csv", na="")
```

Work with the Murders Dataset

Calculate the average murder rate for the country

Once we determine the per million rate to be r, this line is defined by the formula: y=rx, with y and x our axes: total murders and population in millions respectively.

In the log-scale this line turns into: log(y)=log(r)+log(x). So in our plot it's a line with slope 1 and intercept log(r). To compute r, we use dplyr:

```
r <- murders |>
  summarize(rate = sum(total) /  sum(population) * 10^6) |>
  pull(rate)
```

Create a static graph for which each point is labeled

Use the data science theme. Plot the murders with the x-axis as population for each state per million, the y-axis as the total murders for each state.

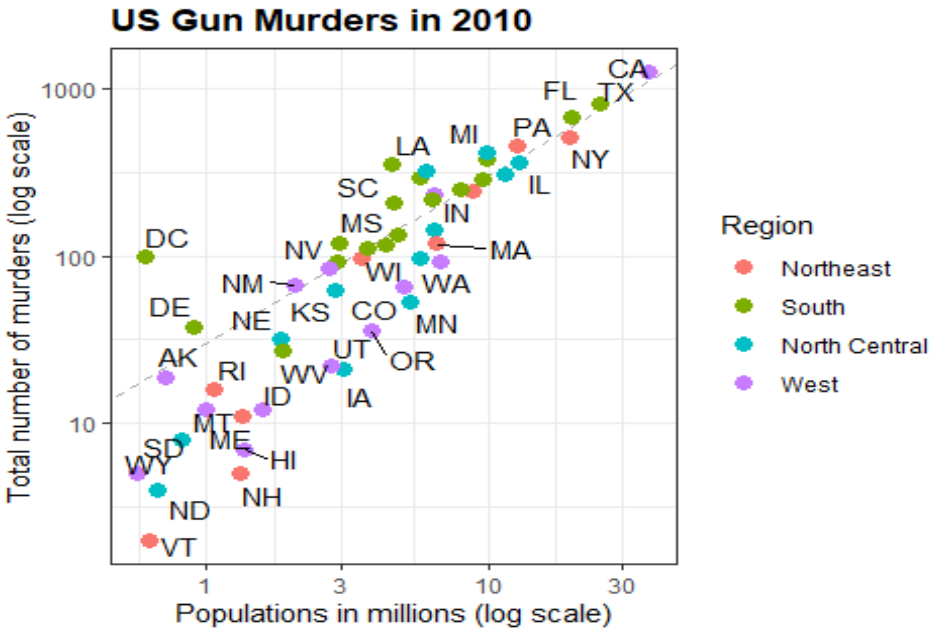
Color by region, add a linear regression line based on your calculation for r above, where we only need the intercept: geom_abline(intercept = log10(r))

Alternatively, use geom_smooth(method="lm", se=FALSE) to get the same results

Scale the x- and y-axes by a factor of log 10, add axes labels and a title.

You can use the command nudge_x argument, if you want to move the text slightly to the right or to the left:

```
ds_theme_set()
murders |> ggplot(aes(x = population/10^6, y = total, label = abb)) +
  geom_abline(intercept = log10(r), lty=2, col="darkgrey") +
  #geom_smooth(method = "lm", se = FALSE, lty=2, col="darkgrey") +
  #use geom_smooth(method = "lm", se = FALSE) instead of geom_abline with log10(r)
  geom_point(aes(color=region), size = 3) +
  geom_text_repel(nudge_x = 0.005) +
  scale_x_log10("Populations in millions (log scale)") +
  scale_y_log10("Total number of murders (log scale)") +
  ggtitle("US Gun Murders in 2010") +
  scale_color_discrete(name="Region")
```



Gapminder Dataset

This dataset includes health and income outcomes for 184 countries from 1960 to 2016. It also includes two character vectors, OECD and OPEC, with the names of OECD and OPEC countries from 2016.

Name the regions: The West, East Asia, Latin America, Sub-Saharan Africa, and Others

```
data("gapminder")

# create a list of "western" states to be used in the following code with case_when
west <- c("Western Europe","Northern Europe","Southern Europe",
          "Northern America","Australia and New Zealand")

gapminder1 <- gapminder |>
  mutate(group = case_when(
    region %in% west ~ "The West",
    region %in% c("Eastern Asia", "South-Eastern Asia") ~ "East Asia",
    region %in% c("Caribbean", "Central America", "South America") ~ "Latin America",
    continent == "Africa" & region != "Northern Africa" ~ "Sub-Saharan Africa",
    TRUE ~ "Others"))
```

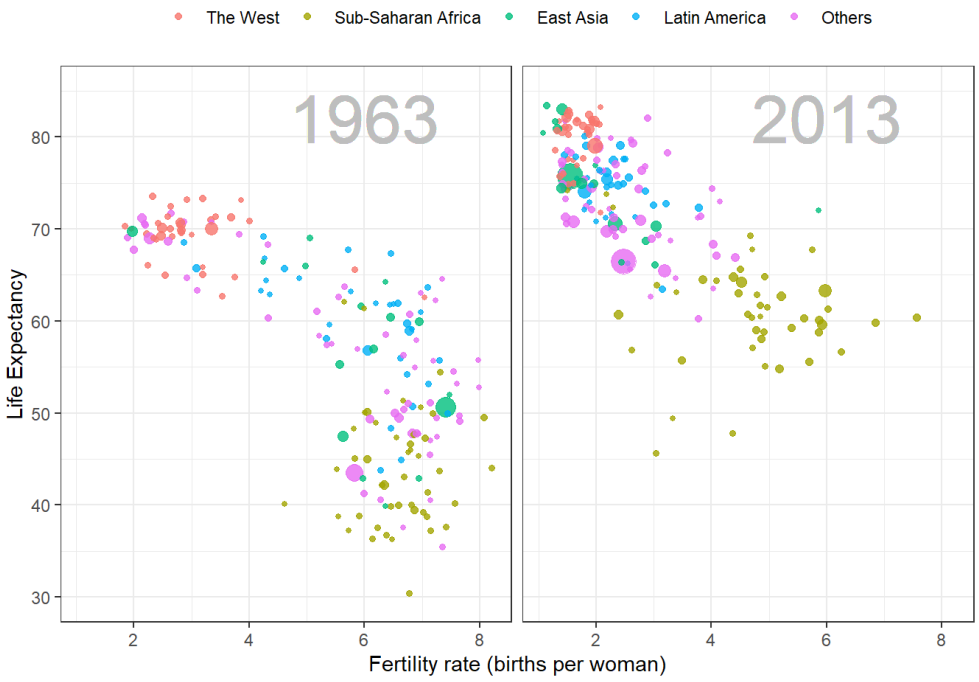
```
gapminder2 <- gapminder1 |>
  mutate(group = factor(group, levels = rev(c("Others", "Latin America", "East Asia","Sub-Saharan Africa", "The West"))))
```

1. Remove all na values from “group”, “fertility”, and “life_expectancy” using !is.na (works the same as na.rm = TRUE)

please do not ever use na.omit()

3. mutate the population to be a value per million
4. change the theme of the plot
5. Use the command: geom_text(aes(x=7, y=82, label=year), cex=12, color=“grey”) to label the two plots at the top right inside the plots (by their years).
6. Shift the legend to go across the top.

```
gapminder2 |>
  filter(year %in% c(1963, 2013) & !is.na(group) &
         !is.na(fertility) & !is.na(life_expectancy)) |>
  mutate(population_in_millions = population/10^6) |>
  ggplot( aes(x=fertility, y=life_expectancy, col = group, size = population_in_millions)) +
  geom_point(alpha = 0.8) +
  guides(size="none") +  # removes the legend
  theme(plot.title = element_blank(), legend.title = element_blank()) +
  coord_cartesian(ylim = c(30, 85)) +
  xlab("Fertility rate (births per woman)") +
  ylab("Life Expectancy") +
  geom_text(aes(x=6, y=82, label=year), cex=12, color="grey") +
  facet_grid(. ~ year) +
  theme(strip.background = element_blank(),
        strip.text.x = element_blank(),
        strip.text.y = element_blank(),
   legend.position = "top")
```



Contagious disease data for US states

The next dataset contains yearly counts for Hepatitis A, measles, mumps, pertussis, polio, rubella, and smallpox for US states. Original data courtesy of Tycho Project. Use it to show ways one can plot more than 2 dimensions.

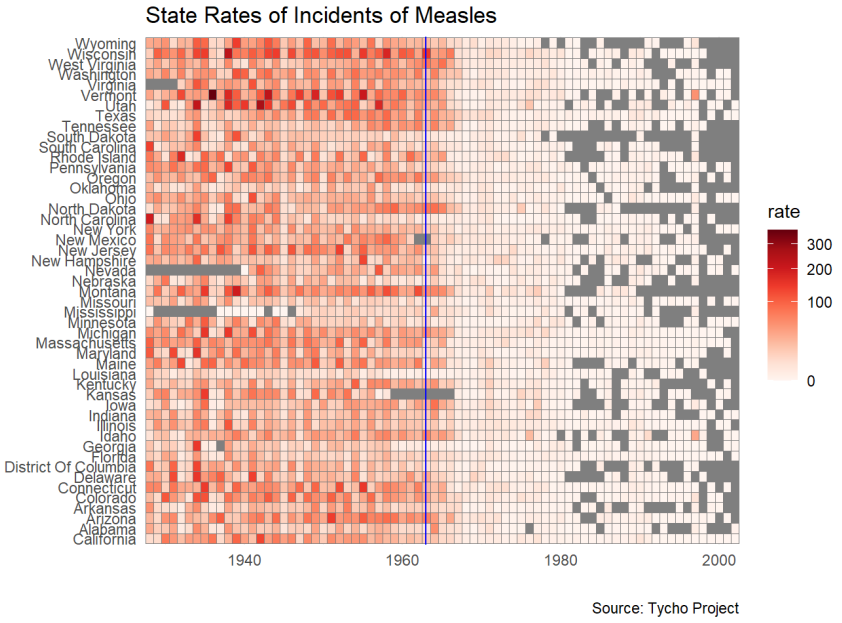
Focus on just measles 1. filter out Alaska and Hawaii

2. mutate the rate of measles by taking the count/(population*10,000)/(weeks_reporting/52)

$$\frac{\frac{count}{population} * 10{,}000}{\frac{weeks\_reporting}{52}}$$

3. draw a vertical line for 1963, which is when the measles vaccination was developed

```
# Focus only on the "Measles" disease
us_contagious_diseases |>
  filter(!state%in%c("Hawaii","Alaska") & disease ==  "Measles") |>
  mutate(rate = count / population * 10000 /(weeks_reporting/52)) |>
  mutate(state = reorder(state, rate)) |>
  ggplot(aes(year, state,  fill = rate)) +
  geom_tile(color = "grey50") +
  scale_x_continuous(expand=c(0,0)) +
  scale_fill_gradientn(colors = brewer.pal(9, "Reds"), trans = "sqrt") +
  geom_vline(xintercept=1963, col = "blue") +
  theme_minimal() +
  labs(title = "State Rates of Incidents of Measles",
       caption = "Source: Tycho Project",
       x = "",
       y = "")
```



Fivethirtyeight 2016 Poll Data

This data includes poll results from the US 2016 presidential elections aggregated from HuffPost Pollster, RealClearPolitics, polling firms and news reports. The dataset also includes election results (popular vote) and electoral college votes in results_us_election_2016. Use this dataset to explore inference.

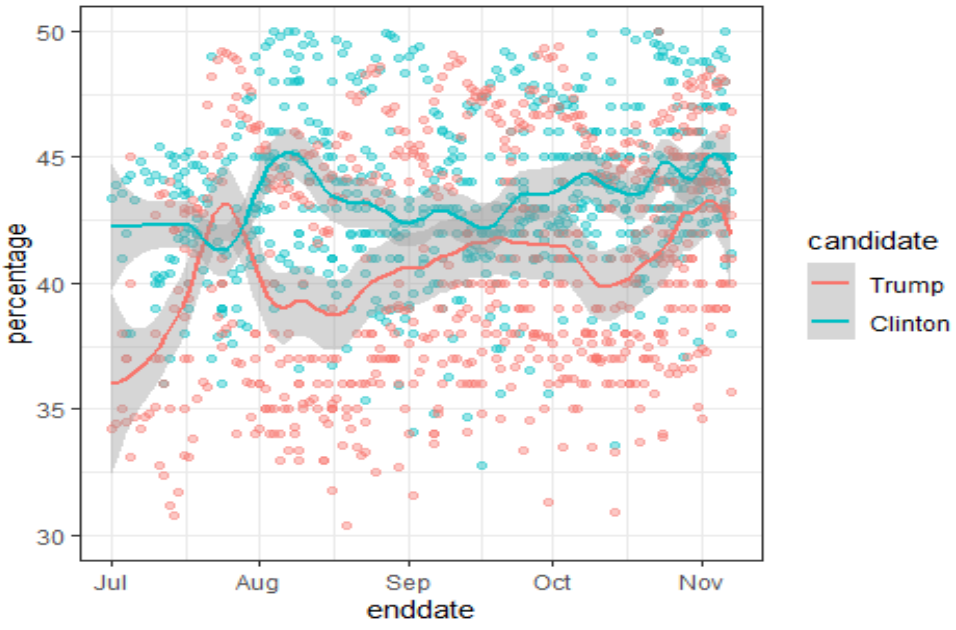
1. Focus on polls for Clinton and Trump after July 2016
2. Plot a scatterplot of the enddate to the percentage in the polls
3. Include a loess smoother regression

```
data(polls_us_election_2016)
polls_us_election_2016 |>
  filter(state == "U.S." & enddate>="2016-07-01") |>
  select(enddate, pollster, rawpoll_clinton, rawpoll_trump) |>
  rename(Clinton = rawpoll_clinton, Trump = rawpoll_trump) |>
  gather(candidate, percentage, -enddate, -pollster) |>
  mutate(candidate = factor(candidate, levels = c("Trump","Clinton")))|>
  group_by(pollster) |>
  filter(n()>=10) |>
  ungroup() |>
  ggplot(aes(enddate, percentage, color = candidate)) +
  geom_point(show.legend = FALSE, alpha=0.4)  +
  geom_smooth(method = "loess", span = 0.15) +
  scale_y_continuous(limits = c(30,50))
```

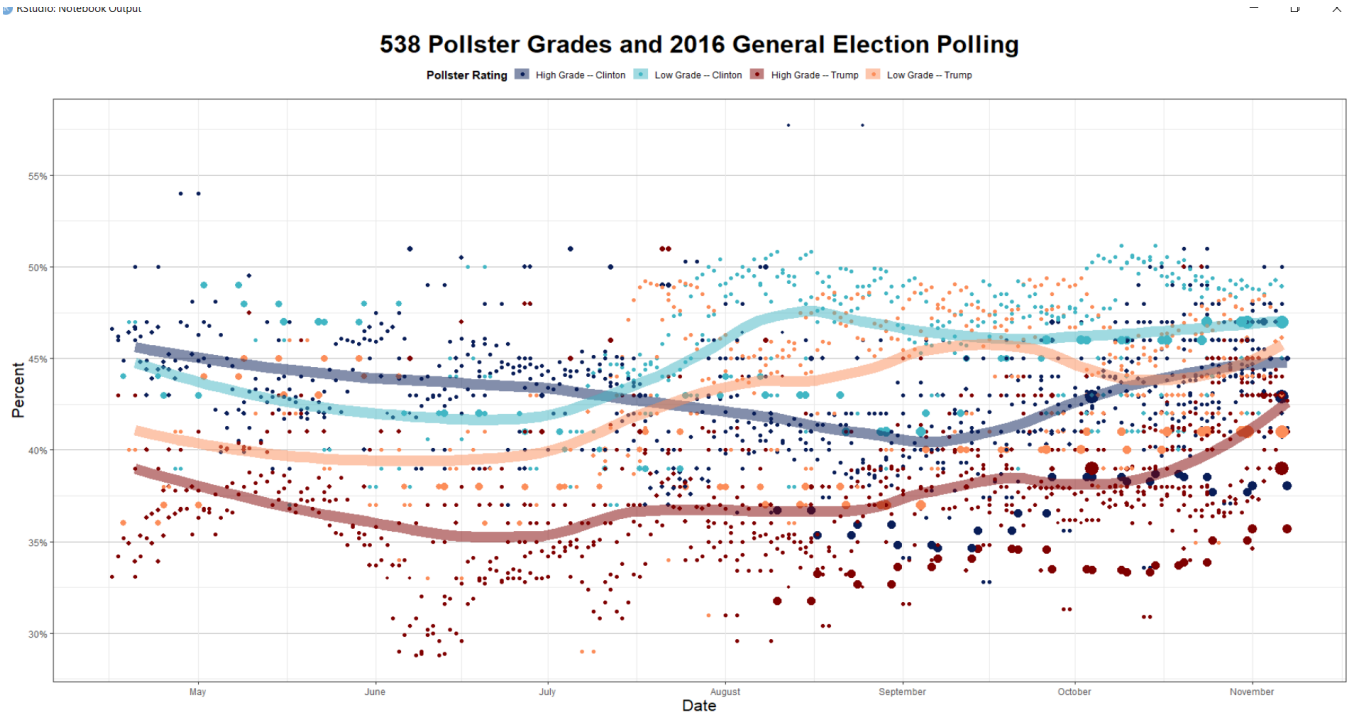
```
`geom_smooth()` using formula = 'y ~ x'
```

```
Warning: Removed 22 rows containing non-finite values (`stat_smooth()`).
```

```
Warning: Removed 22 rows containing missing values (`geom_point()`).
```



Here is an example of this polling dataset, done by a former student, Michael Metzler (code for his work is posted in Blackboard as a markdown file)



Working with HTML Widgets and Highcharter

Set your working directory to access your files

```
# load required package - scales
library(scales)
```

Make a range of simple charts using the highcharter package

Highcharter is a package within the htmlwidgets framework that connects R to the Highcharts and Highstock JavaScript visualization libraries. For more information, see https://github.com/jbkunst/highcharter/

Also check out this site: https://cran.r-project.org/web/packages/highcharter/vignettes/charting-data-frames.html

Install and load required packages

Now install and load highcharter, plus RColorBrewer, which will make it possible to use ColorBrewer color palettes.

Also load dplyr and readr for loading and processing data.

```
# install highcharter, RColorBrewer
# install.packages("highcharter","RColorBrewer")
```

```
# load required packages
library(highcharter)
library(RColorBrewer)
```

Load and process nations data

Load the nations data, and add a column showing GDP in trillions of dollars.

```
nations <- read_csv("nations.csv") |>
  mutate(gdp_tn = gdp_percap*population/1000000000000)
```

Make a version of the "China's rise" chart from your prior assignment

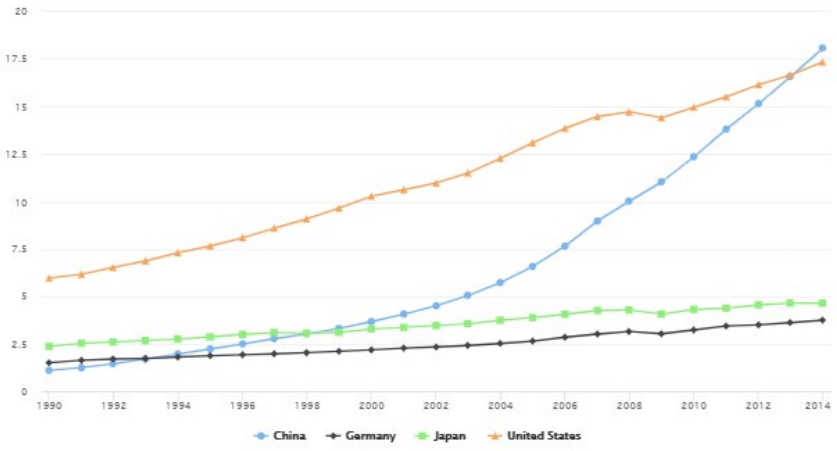
First, prepare the data using dplyr:

```
# prepare data
big4 <- nations |>
  filter(iso3c == %in% c("CHN","DEU", "JPN", "USA")) |>
  arrange(year)
```

The arrange step is important, because highcharter needs the data in order when drawing a time series - otherwise any line drawn through the data will follow the path of the data order, not the correct time order.

Now draw a basic chart with default settings:

```
# basic symbol-and-line chart, default settings
highchart() |>
  hc_add_series(data = big4,
                  type = "line", hcaes(x = year,
                  y = gdp_tn,
                  group = country))
```



In the code above, the function highchart() creates a chart.

Clicking on the legend items allows you to remove or add series from the chart.

Highcharter works by adding data "series" to a chart, and from R you can add the variables from a data frame all in one go using the hc_add_series function.

Inside this function we define the data frame to be used, with data, the type of chart, the variables to be mapped to the x and y axes, and the variable to group the data: here this draws a separate line for each country in the data.

Go to the github site provided above for the chart types available in Highcharts.

Now let's begin customizing the chart.

Use a ColorBrewer palette

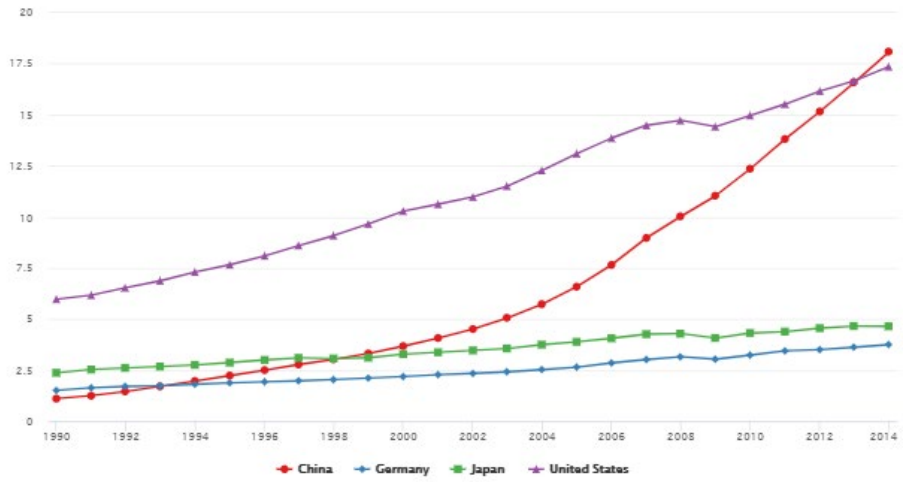
Using RColorBrewer, we can set a palette, and then use it in highcharter.

```
# define color palette

cols <- brewer.pal(4, "Set1")


highchart() |>

  hc_add_series(data = big4,
                    type = "line", hcaes(x = year,
                    y = gdp_tn,
                    group = country)) |>
                    hc_colors(cols)
```

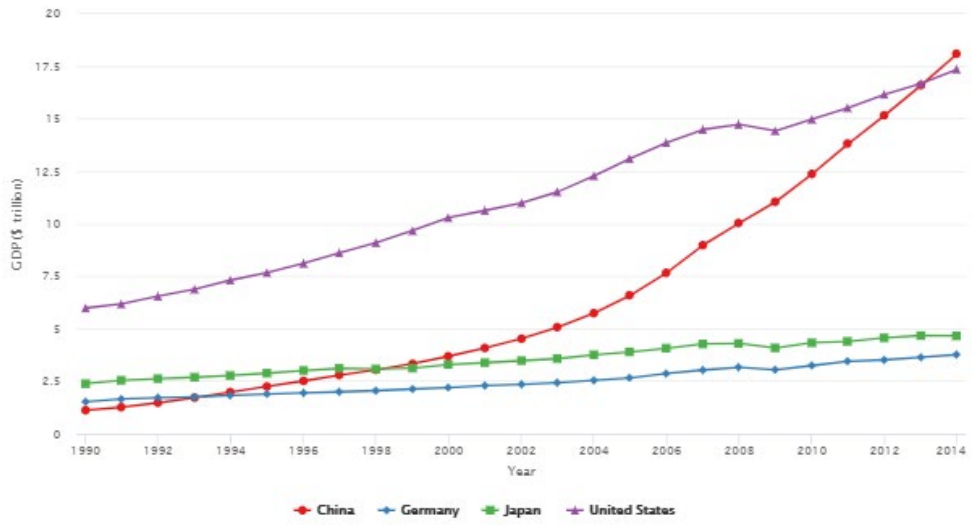


The first line of code sets a palette with four colors, using the "Set1" palette from ColorBrewer. This is then fed to the function hc_colors() to use those colors on the chart.

Add axis labels

```
highchart() |>
  hc_add_series(data = big4,
                    type = "line",
                    hcaes(x = year,
                    y = gdp_tn,
```

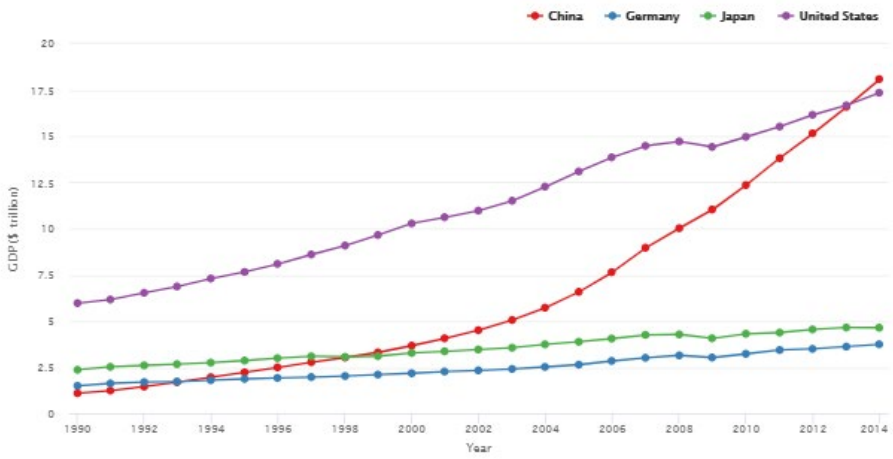
```
                    group = country)) |>
  hc_colors(cols) |>
  hc_xAxis(title = list(text="Year")) |>
  hc_yAxis(title = list(text="GDP ($ trillion)"))
```



Change the legend position

For this, we use the hc_legend function.

```
highchart() |>
  hc_add_series(data = big4,
                    type = "line",
                    hcaes(x = year,
                    y = gdp_tn,
                    group = country)) |>
  hc_colors(cols) |>
  hc_xAxis(title = list(text="Year")) |>
  hc_yAxis(title = list(text="GDP ($ trillion)")) |>
  hc_plotOptions(series = list(marker = list(symbol = "circle"))) |>
  hc_legend(align = "right",
            verticalAlign = "top")
```

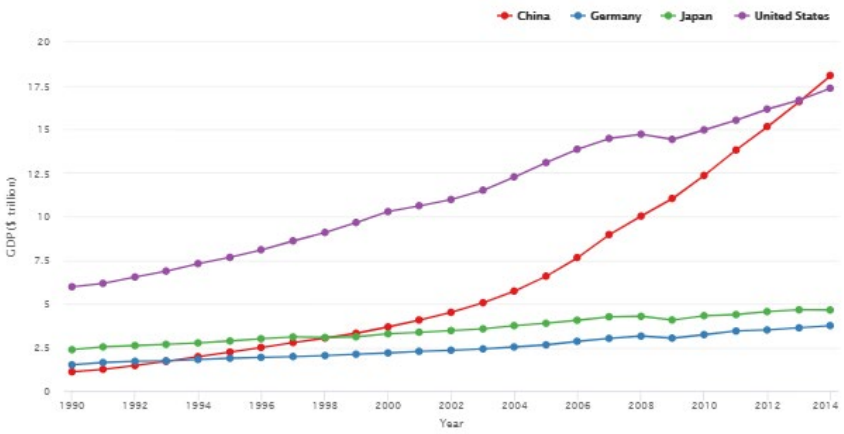


Customize the tooltips

By default we have a tooltip for each series, or line, and the numbers run to many decimal places.

We can change to one tooltip for each year with “shared = TRUE”, and round all the numbers to two decimal places with pointFormat = "{point.country}: {point.gdp_tn:.2f}

```
# customize the tooltips
big4_chart <- highchart() |>
  hc_add_series(data = big4,
                  type = "line",
                  hcaes(x = year,
                  y = gdp_tn,
                  group = country)) |>
  hc_colors(cols) |>
  hc_xAxis(title = list(text="Year")) |>
  hc_yAxis(title = list(text="GDP ($ trillion)")) |>
  hc_plotOptions(series = list(marker = list(symbol = "circle"))) |>
  hc_legend(align = "right",
            verticalAlign = "top") |>
  hc_tooltip(shared = TRUE,
             borderColor = "black",
             pointFormat = "{point.country}: {point.gdp_tn:.2f}<br>")
big4_chart
```



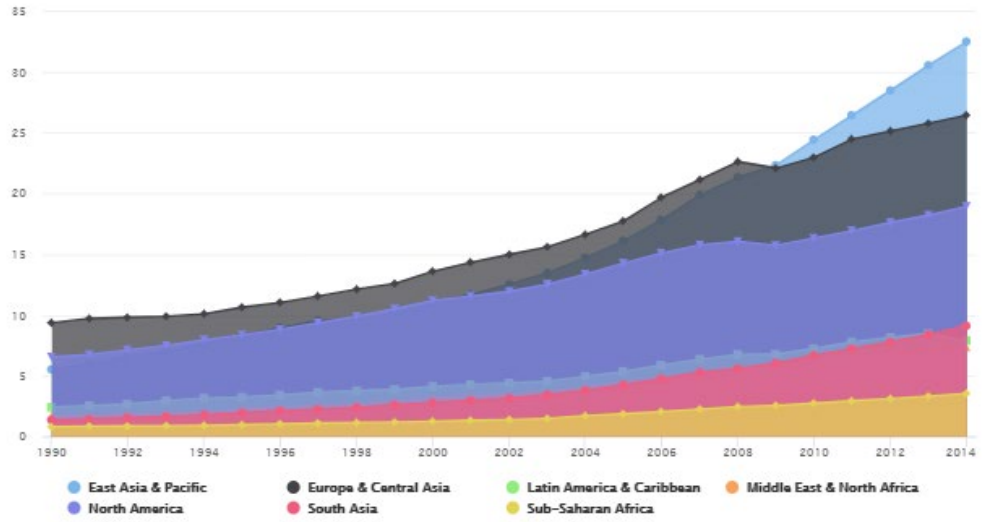
Prepare the data

First, prepare the data using dplyr.

```
# prepare data
regions <- nations |>
  group_by(year,region) |>
  summarize(gdp_tn = sum(gdp_tn, na.rm = TRUE)) |>
  arrange(year,region)
```

Make an area chart using default options

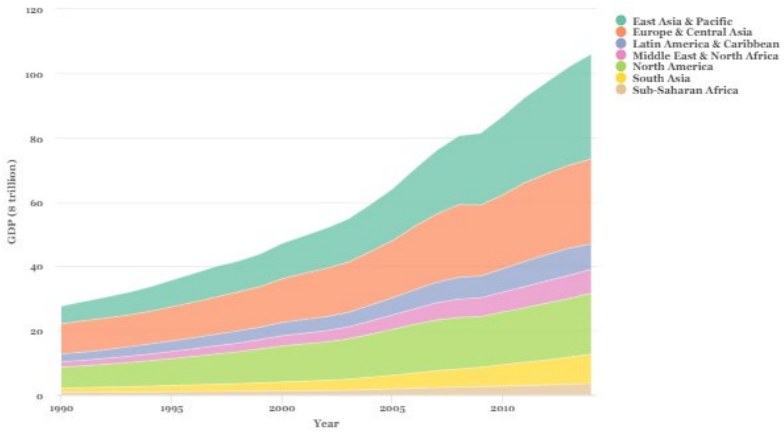
```
# basic area chart, default options
highchart () |>
  hc_add_series(data = regions,
                type = "area",
                hcaes(x = year,
                y = gdp_tn,
                group = region))
```



The following code customizes the chart in other ways. It uses the same ColorBrewer palette, with seven colors.

```
# set color palette
cols <- brewer.pal(7, "Set2")

# stacked area chart
highchart () |>
  hc_add_series(data = regions,
                  type = "area",
                  hcaes(x = year,
                  y = gdp_tn,
                  group = region)) |>
  hc_colors(cols) |>
  hc_chart(style = list(fontFamily = "Georgia",
                        fontWeight = "bold")) |>
  hc_plotOptions(series = list(stacking = "normal",
                               marker = list(enabled = FALSE,
                               states = list(hover = list(enabled = FALSE))),
                               lineWidth = 0.5,
                               lineColor = "white")) |>
  hc_xAxis(title = list(text="Year")) |>
  hc_yAxis(title = list(text="GDP ($ trillion)")) |>
  hc_legend(align = "right", verticalAlign = "top",
            layout = "vertical") |>
  hc_tooltip(enabled = FALSE)
```



We have already encountered the main functions used here. The key changes are in the hc_plotOptions() function:

stacking = “normal” creates the stacked chart. See what happens if you use stacking = “percent”.

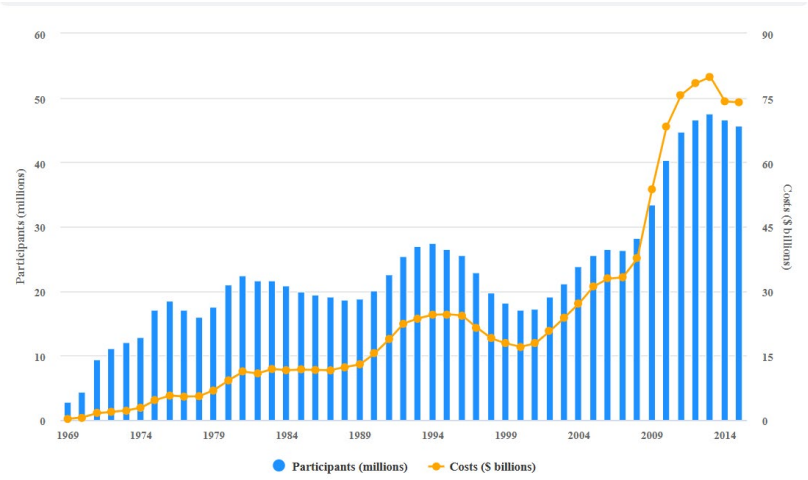
lineWidth and lineColor set the width and color for the lines under marker = list() the code states = list(hover = list(enabled = FALSE)) turns off the hovering effect for each marker on the chart, so that the markers no longer reappear when hovered or tapped.

In the hc_legend() function, layout = “vertical” changes the layout so that the legend items appear in a vertical column.

Food Stamps Data - Combine Two Types

```
cols <- c("dodgerblue","orange")          # set the color palette
food_stamps<- read_csv("food_stamps.csv")
highchart() |>
  hc_yAxis_multiples(
    list(title = list(text = "Participants (millions)")),
    list(title = list(text = "Costs ($ billions)"),
         opposite = TRUE)) |>
  hc_add_series(data = food_stamps$participants,
                name = "Participants (millions)",
                type = "column",
                yAxis = 0) |>
  hc_add_series(data = food_stamps$costs,
                name = "Costs ($ billions)",
                type = "line",
                yAxis = 1) |>
  hc_xAxis(categories = food_stamps$year,
           tickInterval = 5) |>
  hc_colors(cols) |>
  hc_chart(style = list(fontFamily = "AvantGarde",
```

```
                    fontWeight = "bold"))
```



Week 7 Assignment Using DSLabs Datasets

1. (Ungraded) Review all code provided in the notes for the plots in DS Labs and Highcharter

2. **(Worth up to 20 points)**

   Take any of the datasets included in "dslabs" package (not death_prb) and create a new multivariable graph. You may use any of the datasests including those used in the notes examples, as long as you create something dramatically different from my work about this data. You must include the following in your graph:

   - Comments describing all chunks of code
   - **Meaningful** labels for x- and y-axes
   - **Meaningful** title
   - A theme for the graph (you must change the default ggplot style)
   - Colors for a third variable, with a legend

   You can create a scatterplot, heatmap, or other plot appropriate for continuous variables. You may **not create a bar graph for categorical variables for this assignment.** Feel free to incorporate elements from prior weeks' materials, as well. You may want to try to include interactivity using highcharter.

   In your rendered markdown/quarto document, be sure you describe in a paragraph what dataset you have used and document how you have created your graph. Be sure to describe any interesting insights you found in your visualization. If you choose to use one of the datasets from the examples in the tutorial, be sure I can ***clearly understand what you have created that is meaningfully different.***

   This assignment is due **at 11:59 pm on Sunday, March 23rd. You will present your visualization during the following class.**