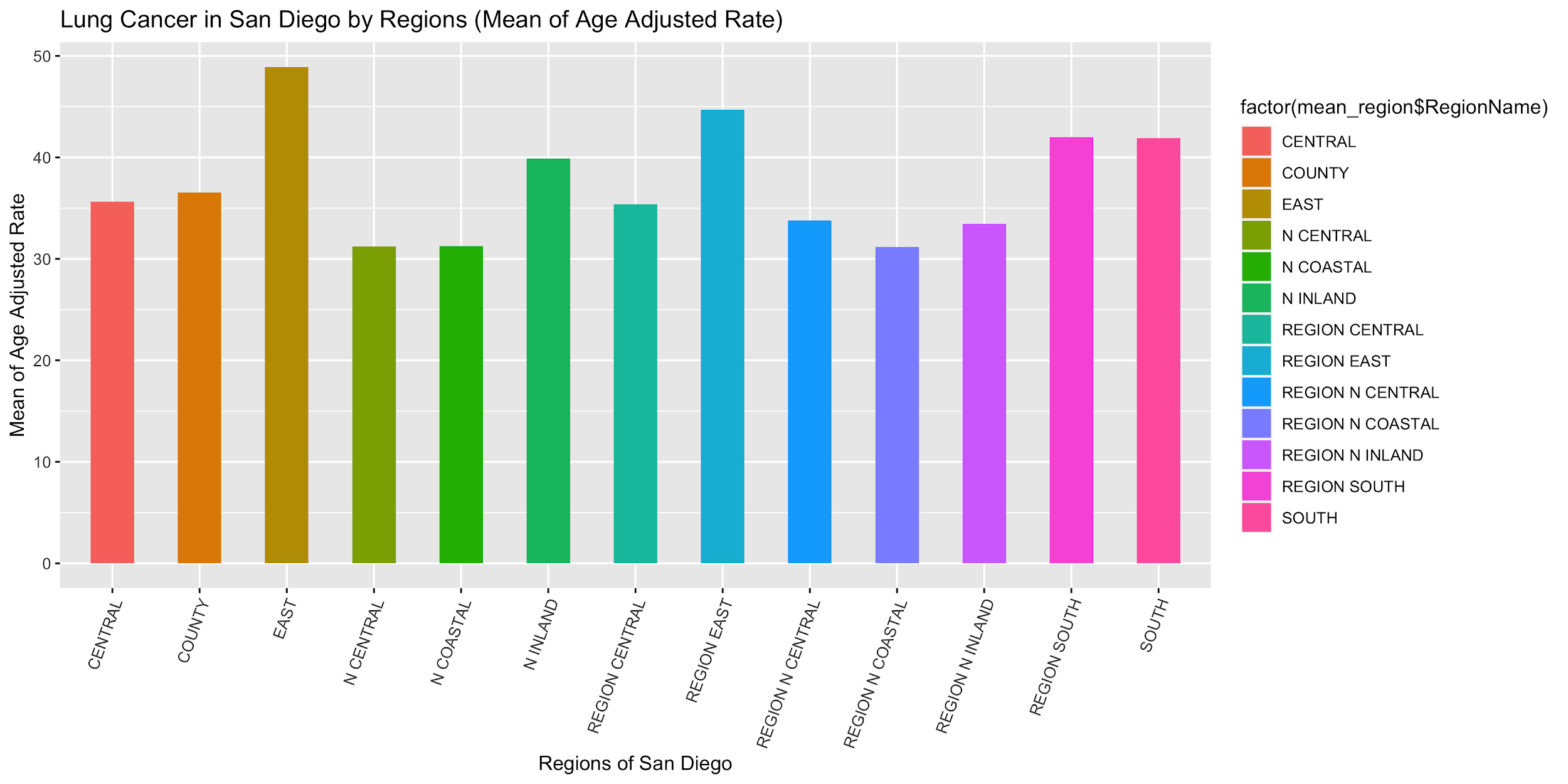
# Web Exercise2: Introduction of GitHub and Online Data Science Resources

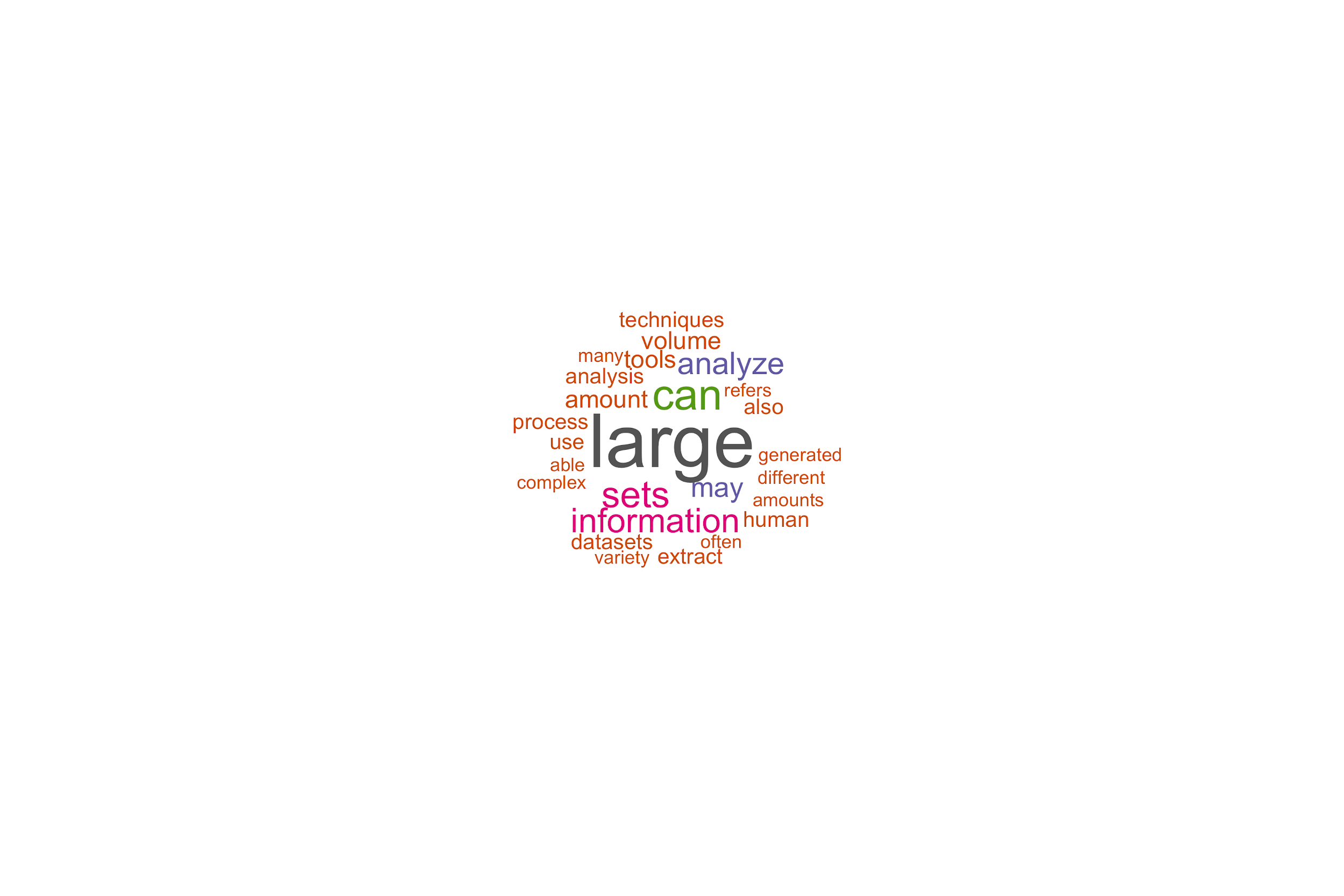
1. Attach the two Big Data Definition Word Cloud Images (texts from the students in the current semester).

Following images and R scripts are attached:

* **myFigure.png**



* **BigData-definition-V2.png**



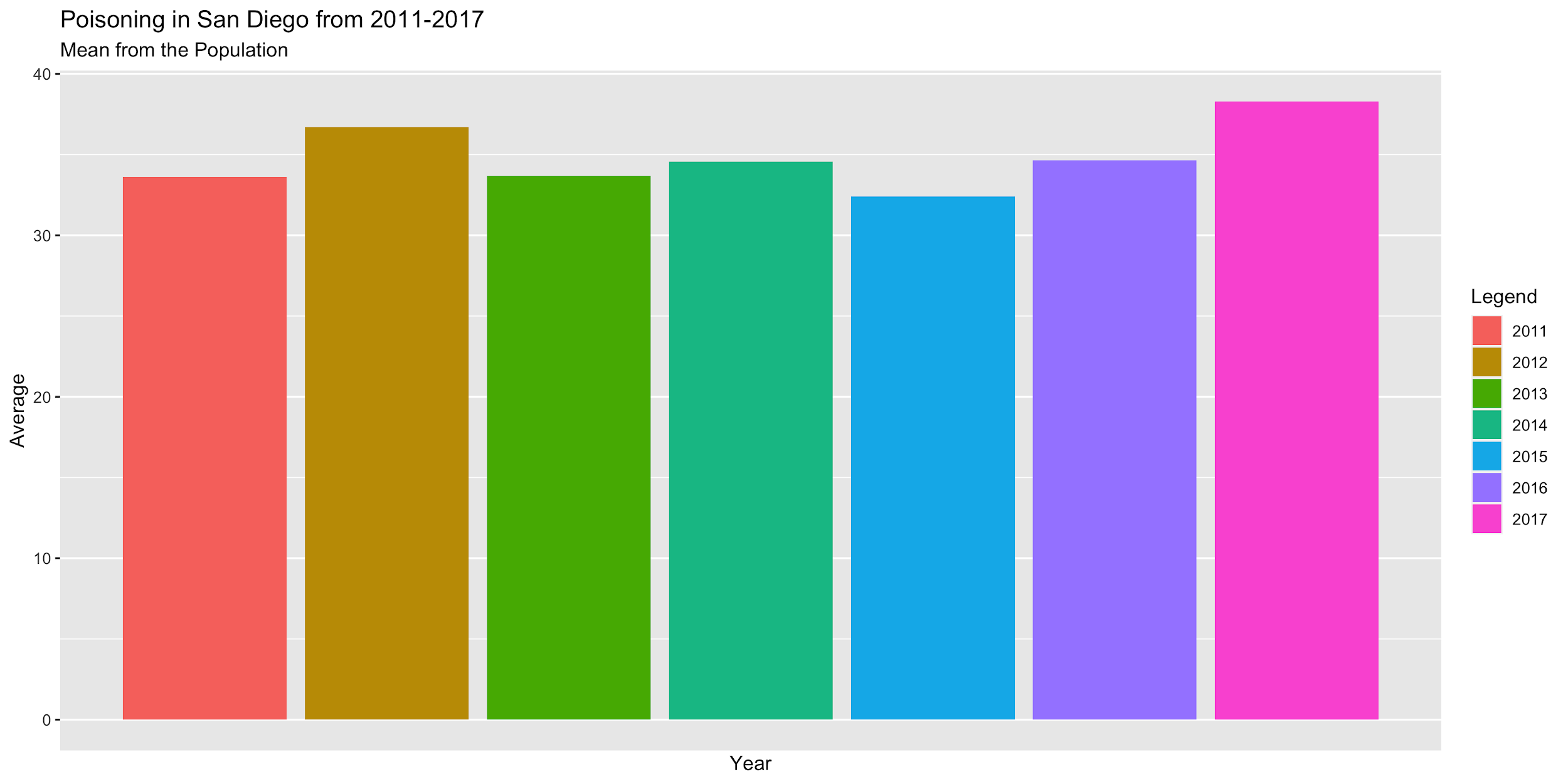
1. Go to the San Diego County Data Portal (https://data.sandiegocounty.gov/). Pick up another data types (other cancer data or other injuries data). Import the data into R and conduct some basic statistical analysis and draw some new visualization graphics (similar or different from the previous example). You will need to revise R-script that is used for previous Cancer analysis. Attach your new R-script and the new data file for creating the analysis and the visual graphics in the report.

**San Diego County Poisoning Data URL**: <https://data.sandiegocounty.gov/Health/Poisoning/ruqz-bg4m>

* Filtered the “OUTCOME” to “Death” and exported the file as **Poisoning\_Death.csv**.

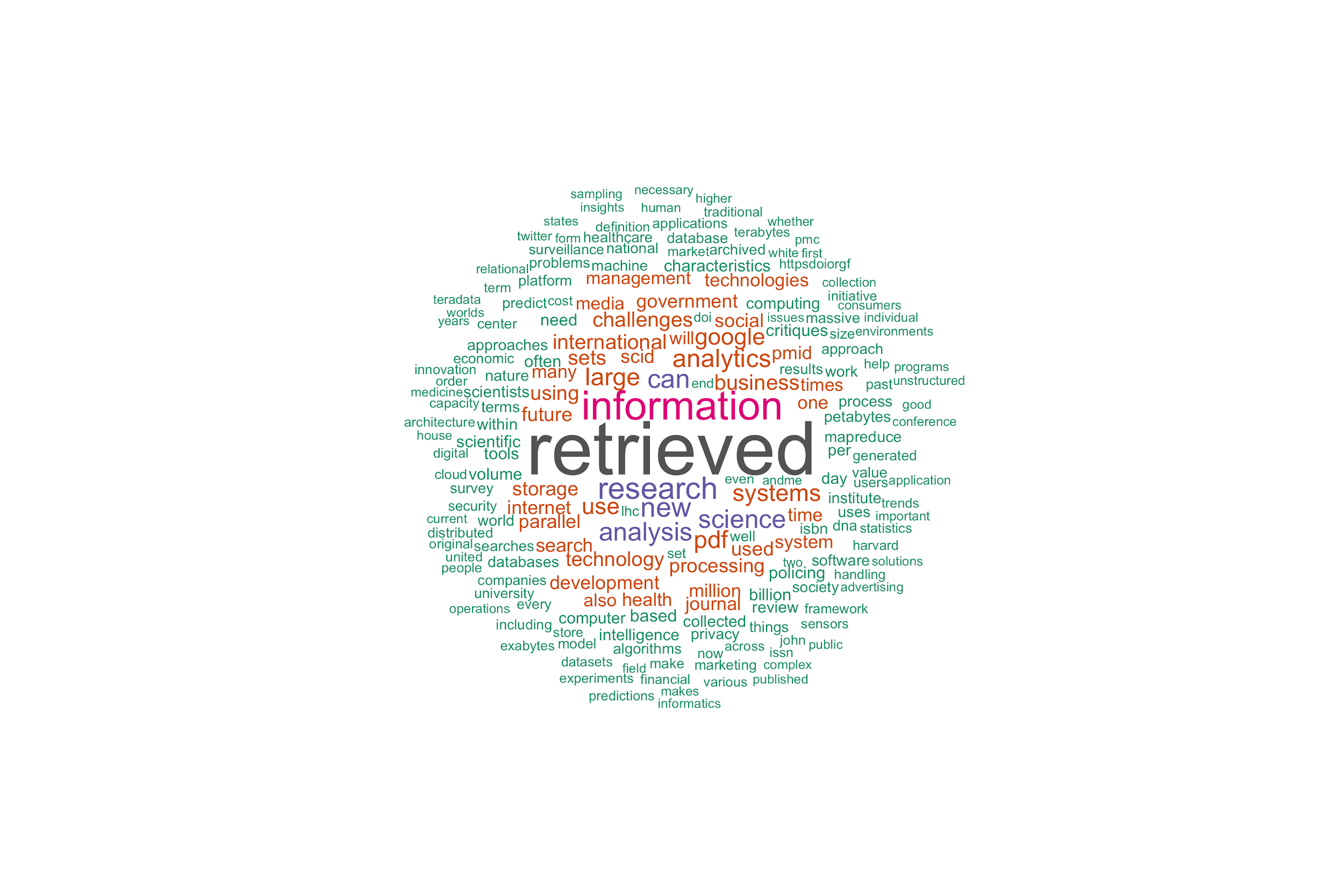
Following **poisoning.png** image and R script attached:

|  |
| --- |
| #### Load required libraries  library(ggplot2)  ### Read csv data into R dataframe  poison\_data <- read.csv("Poisoning\_Death.csv")  ### Generate some statistics  ## Aggregate region by names, then calculate the mean for three rates  mean\_year <- aggregate(poison\_data[, c("Total\_Male","Total\_MaleRate", "Total\_Female", "Total\_FemaleRate", "TotalRate", "Total")], by=list(Year = poison\_data$Year), FUN=mean, na.rm=TRUE)  print(mean\_year)  ### Make a visual plot ! aes: aesthetic mappings, geom\_bar: rectangle bars  poison\_plot <- ggplot(mean\_year, aes(x = Year, y = Total)) +  geom\_bar(stat="identity", aes(fill = factor(Year))) +  scale\_x\_continuous(breaks = NULL) +  labs(title = "Poisoning in San Diego from 2011-2017", subtitle = "Mean from the Population", y = "Average", x = 'Year', fill = 'Legend')  poison\_plot  ## Show the plot & save it to file  ggsave("poisoning.png", width = 12, height = 6) |



1. Select a Webpage or a group of text files, create a word cloud map with some selected “Stopwords”. In the report, indicate the text sources, the selected “stopwords”, and the output of WordCloud images.

* **Text Source**: Big Data (<https://en.wikipedia.org/wiki/Big_data>)
  + Exported the web page as a PDF and saved as a text file.
* **Selected Stopwords**:
  + References to big data: "big", "data", "bigdata"
  + References to Wikipedia: "httpsenwikipediaorgwikibigdata", "wikipedia", "wikimedia"
  + Symbols counted as words: "–"
  + Websites counted as words: "httpswwwworldcatorgissn", "httpswwwncbinlmnihgovpmcarticlespmc", "httpspubmedncbinlmnihgov", "httpsapisemanticscholarorgcorpusid"
  + Months: "january", "february", "march", "april", "may", "june", "july", "august", "september", "october", "november", "december"
* **Filter**: Removed rows with a word frequency less than 5 for more focus on words with a much higher frequency (50+ counts).
* **Image**: BigDataWiki.png



1. Introduce R-Shiny and its major functions (at least 200 words) using your own languages (don’t copy/paste the definition from the Internet). Find one on-line example of R-Shiny applications and provide the URL (Web address) and the application screen shot.

**R-Shiny** is a package from R Studio that creates interactive web applications using R. It has three components to guarantee that the input of the user automatically displays the output.

* **User Interface (UI)**: Sets up the visual design and layout of the web page (e.g. title name). Specifically, the UI will have a section where the user can input information and another section where the output will be displayed after an input is selected. For example, the input call can be as simple as the user defining the number of bins for a histogram. The output display is the histogram data reflecting the number of bins selected from the user.
* **Server**: What makes R-Shiny interactive is the reactive expressions defined in the server. It contains the R code to take in the input values from the UI and creates a plot based on the input. For example, after the user inputs eight bins, the server will take that input value and create a histogram plot with eight bins. That plot will be pushed to the UI for display.
* **Shiny App**: This function takes in the UI and server to create an R script called app.R. This creates the Shiny application. Going back to the histogram example, this web page can now be accessed by users to create histograms of different bin sizes.

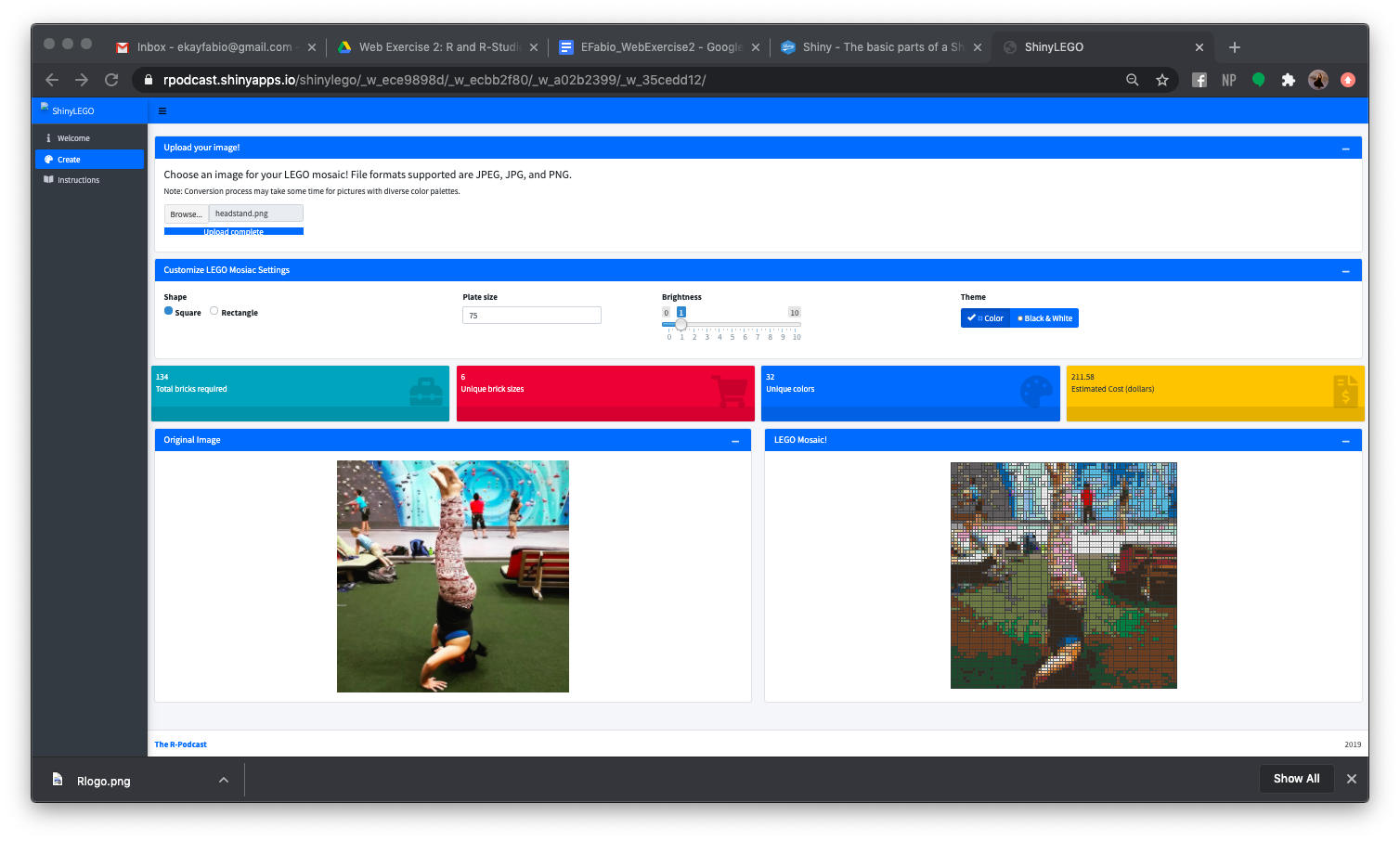
An online example of an R-Shiny application is called **ShinyLEGO** (<https://rpodcast.shinyapps.io/shinylego/_w_ece9898d/_w_ecbb2f80/_w_a02b2399/_w_35cedd12/>). It takes an image and creates a Lego mosaic of that image. You can also adjust the following:

* Shape (Square or Rectangle)
* Plate Size (Determines the number of Legos to use for more detail)
* Brightness (Numerical value from 1 to 10)
* Theme (Color or Black & White)

Other than displaying the Lego mosaic, it gives further information to help you create the mosaic:

* Total bricks required
* Unique brick sizes
* Unique colors
* Estimated Cost (dollars)

**ShinyLego Application Screenshot**:



1. There are many data science oriented R packages (libraries) available (for data manipulation, data visualization, or machine learning). Please identify THREE packages (libraries) in R related to data science and introduce each of them briefly (at least 100 words for each packages).

R packages related to data science:

* **dplyr**: dplyr is a useful package in data manipulation and more intuitive when managing tabular data compared to using Pandas for Python. Similar to the ggplot2 package being referred to as the “grammar of graphics”, dplyr is defined as the “grammar of data manipulation”. dplyr is also part of the tidyverse package collection specifically designed for data science. This package contains the following functions for manipulating data frames such as selecting columns, filtering rows, and sorting rows of a table. There are also functions to clean your dataset such as removing duplicate entries, changing data types (e.g. string to numeric), and string parsing.
* **DataExplorer**: This package has three important goals: exploratory data analysis (EDA), feature engineering, and data reporting. EDA is a process of analyzing raw data to gain a better understanding of the dataset and seeing what the data can tell us so a formal hypothesis testing can take place. Feature engineering takes it one step further in using your domain knowledge for EDA and creating predictive models and new features. You can use DataExplorer functions to summarize your dataset that gives you the number of columns, rows, memory usage, and more. You can also visually plot your dataset summary analytics and the amount of missing data. As for data reporting, distribution plots can be created similar to ggplot2, like histograms and bar charts.
* **mlr3**: mlr3 is a successor package to mlr. This package provides the basic foundation to machine learning, and provides R6 objects for the following classes: tasks (stores data and metadata), learners (train and predict functions), resampling (defines partitioning the tasks to train and train sets), and measures (calculating performance of one or more measures). R6 is a style of programming that can be referred to as classical object-oriented programming. R6 objects are mutable and thus have reference semantics. The mlr3 package is heavily focused on computation operations but has add-on packages that can provide additional functionality like data visualization. mlr3 also supports parallelization and error handling and encapsulation.