[redacted]

[redacted]

by

Scott Greenberg

Anumber: [redacted]

Date: [redacted]

[redacted]

[redacted]

[redacted]

[redacted]

(i) [redacted]

Summarize each of the four selected videos on about two pages. Start with the authors, title, and year of a video.

One page should be a textual description that summarizes the main features of the video. What is the underlying data, which visualization techniques have been used, etc.? Note that the web page contains a short description for most of the videos. You can build on this description, but this description is not enough to obtain full points for this question and it has to be extended considerably.

The other page should contain figures that show two or three screenshots taken from the video (different than the ones from the web page). Your screenshots should show some of the most relevant visualization techniques from the video. If a technique from the video has been introduced in Stat Viz I or II, mention this in your accompanying text. You will be surprised to see how many of the early techniques shown in these videos still are in use these days.

Overall, the answer to this question should occupy about 8 pages in your final HW solutions.

Answer:

Real-time Rotation by Jih-Jie Chang (1970)

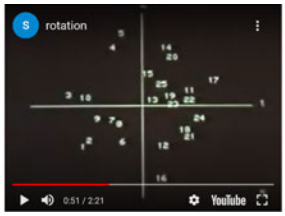
The video visualizes the operations of a computer program called ROTATE.

ROTATE does orthogonal rotations (90 degree rotations) of three-dimenstional combinations in a higher dimensional space.

This specific video visualizes how ROTATE operates searching for a five-by-five lattice design in two dimensions embedded in a five-dimenstional space.

We can see in Rotate.Fig1-4 the starting position, and after each rotation of the combination of the first (shown in center far right of figures), second (shown in far top center of the figures), and fifth (shown in the bottom right of the figures) dimension.

(a) Rotation.Fig1:Start



(b) Rotation.Fig2:Xaxis Rotation

(c) Rotation.Fig3:Yaxis Rotation

(d) Rotation.Fig4:Zaxis Rotation

The visualization of rotation is reminiscent of 3d plots used early in the semester, except there is no clearly defined 3rd axis in this visualization.

Integrated Documents by G.Sawitzki (1994, 1999).

Using a paper on diagnostic plots as example. Paper is Diagnostic plos for One-Dimensional Data by G, Sawitzki.

Video starts by showing various diagnostic plots. Gives example of empirical distribution function for KS statistic as related functional. Narrator then supposes that an expirimentalist would like to see how his data looks with that diagnostic plot.

Shows that in an integrated document, mark your data then dragging and dropping the data into the plot to create the diagnostic plot of the data. We can see this in ID.Fig1.

One can also view the default data making the plot and highlight a subset to see only the diagnostic plot of that subset

The narrator then shows that one can then drag multiple plots over to document and have them draw from random clock. Can use this to show that for bimodal data, a box and whisker does not show bimodality but a shorts plot reveals it easily. We can see this in ID.fig2

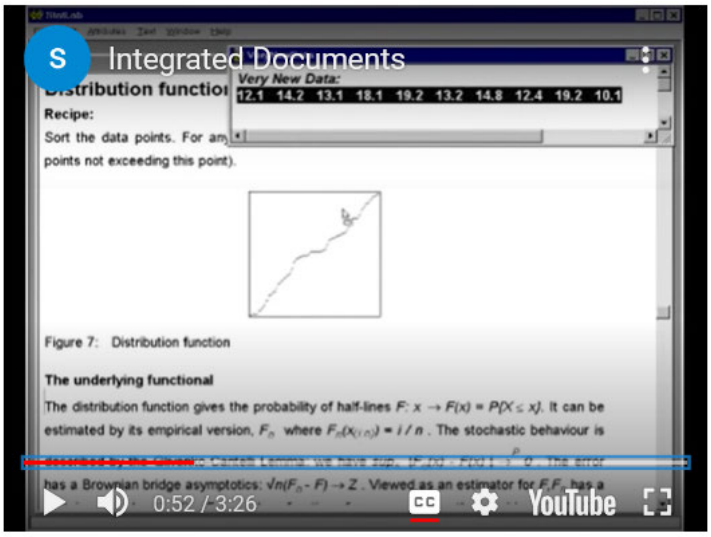


Figure 2: ID.Fig1: Data Drag n Drop

ID.Figure1 shows the ability to drag and drop data. The closest thing this seems similar to is interactive plots we dealt with this semester. In addition, below we have a figure of several diagnostic plots (that is a little chaotic). I find that how this data is presented is similar to extended scatterplot matricies, except that we are dealing with 1 dimensional data.

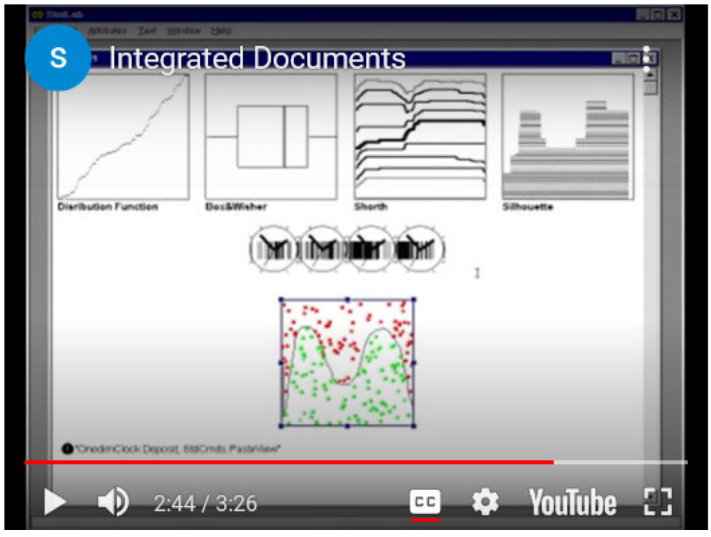


Figure 3: ID.Fig2: Bimodality

Ozone in the Northeast by Richard A. Becker, William S. Cleveland, Beat Kleiner & Jack L. Warner. Video credited data collection from 1974. Video created in 1978 with the additional help of Tracy Warner, and Robert Cleveland (With Tracy Warner being the daughter of Jack L. Warner and Robert Cleveland being the son of William S. Cleveland.

The data was collected at a variety of measuring sites in the northeast on July 2nd 1974.

The video shows the change in ozone concentrations at different locations of New England over the course of about 20 hours, starting at 6AM on July 2nd 1974, then seemingly repeats the same movie at a faster speed. There is a clock at the bottom right that shows the time of day. The diameter of each circle show the ozone concentrations; and, the diameter of the clock on the bottom right was the [then] current EPA standard of the time 0.08 ppm (parts per million).

This video detailed the important discovery that photochemical air pollution, previously thought to only travel 5 to 10 miles, actually traved hundreds of miles with the aid of wind. We can see this in Ozone Fig1-6, as we see that the air pollution travels all the way to New Hampshire from New York City in the course of a day.

(a) Ozone.Fig1:7AM

(b) Ozone.Fig2:10AM

(c) Ozone.Fig3:12PM

(a) Ozone.Fig4:3PM

(b) Ozone.Fig5:7PM

(c) Ozone.Fig6 :12AM

In the above figures, we see something similar to a comparative micromap, which is something we did in Stat Visualization 2. There is a map as a background image, similar to choropleth maps. The data is represented by size of a circle rather that colors (what we have done traditionally), as regions may not have been suitable as they would be of vastly different sizes.

Image of a Thunderstorm by Anne Freeny and John Gabbe at AT&T Bell Laboratories (1966 - video made in 1968).

The 4 minute 10 second video is a time series representation of a thunderstorm, more specifically the rate of rainfall of a thunderstorm, in Holmdel,(?) New Jersey that occured in July 1966. The city is partitioned into 96 squares (since each rain gauge is approximately 0.8 miles apart), where the whiteness of a square corresponds to a larger rate of rainfall (The whitest squares correspond to 200mm/hr). The map is drawn so that the top of the video is north and the squares are put against a black background to simulate greyscale. The 4 minute 10 second segment of video shown depicts the measurement of the 20 minutes of rainfall. Each gauge recorded a measurement every 10 seconds.

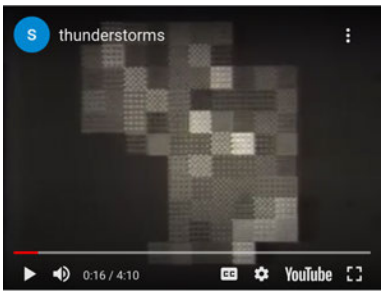
The shading was achieved by having each square be a 8x8 grid and plotting one of two characters (of different color) at either 8, 16, 32, 48, or all 64 of the gridpoints of the square. This was done to simulate shading while still only using a small amount of computational time.

Occasionally a square turns black and this corresponds to a gauge that temporarily stopped transmitting data; and, the top right of a square flashing white is caused by an error in the graphing software. We can see both of these in Thunderstorms.Fig1.

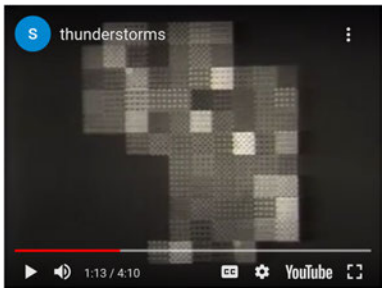
The goal of the video was to model the intensity and movement of the rainstorm.It is shown that over time there is a movement of white squares to the bottom right corner, and this represents the heaviest parts of the storm moving in a southeast direction. It is also shown that the storm 'dies down', as the number of white squares decreases. It was stated that a static display of the data as twodimensional set of timeseries was a much more useful guide to finding interesting pieces of data, since variability in space and time was so large that patterns of the movement of heavy rain were very hard to detect by eye.

Figure 6: Thunderstorms.Fig1: Above we have a screenshot of the video that depicts both types of errors mentioned earlier.

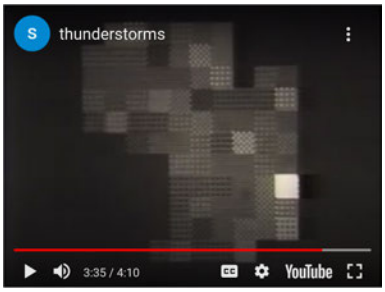
In Thunderstorms.Figure1 we see the errors that occur. AIt is interesting since there are errors in the data (represented by a black square), and errors in the software (represented by a tiny white square in the top left of a square). Even today we deal with errors in data or even in the software. The images below are reminiscent of comparative micromaps.However, a [choropleth map] as the background image was not used, different than how we have done things in Stat Visualization 2.



(a) Thunderstorm.Fig2:Early



(b) Thunderstorm.Fig3:Mid



(c) Thunderstorm.Fig4:Late

(ii) **Shiny App for Bad Color Graphic from HW 3** (25 Points):
Build a shiny app that allows a user to toggle between the original graphic with the bad colors and a graphic with improved colors. These should be your graphics / improvements from HW 3.

Moreover, add a second selection mechanism that let's a user check for various colorblind appearances of these graphics, i.e., the original and the improved one. This should be based on the *dichromat* R package and the three color–blindness types deutan, protan, and tritan. Overall, your app should be able to show six differently colored graphics: two sets of color (original and improved) $\times$ three types of color–blindness. Include screenshots that show your shiny app and the resulting graphics for all of these six settings.

Adding a toggle button that allows you to add/choose different symbols or sizes (or what else you may have done in HW 3 to further improve your original graph) is optional.

When you develop such an app, follow the principle from class: Start with the most basic app that just shows your original bad graph (the default). Then add one control option after the other.

Include only the final version of your R code. Make sure that you change to `eval=FALSE` for this question when you compile your Rnw document.

Answer:

```
library(shiny)
runApp("shinyhw4q2")
```

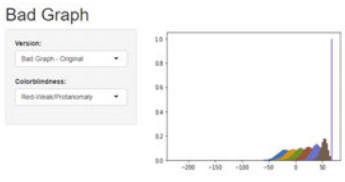
```
library(shiny)

shinyUI(pageWithSidebar(
      headerPanel("Bad Graph"),
      sidebarPanel(
             selectInput("version", "Version:",
             list("Bad Graph - Original" = "o",
                  "Bad Graph - Improved" = "cbs")),
             selectInput("type", "Colorblindness:",
             list("None" = "badgraph",
                  "Red-Weak/Protanomaly" = "rwp",
                  "Green-Weak/Deuteranomaly" = "gwd",
                  "Blue-Weak/Tritanomaly" =  "bwt",
                  "Blue Cone Monochromacy" = "bcm",
                  "Monochromacy/Achromatopsia" = "mca",
                  "Red-Blind/Protanopia" = "rbp",
                  "Green-Blind/Deuteranopia" = "gbd",
                  "Blue-Blind/Tritanopia" = "bbt"))
             ),
      mainPanel(
          imageOutput("Plot")
               )
))
```

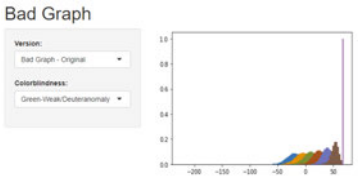
Figure 8: shiny UI.R

```
library(shiny)
library(png)
setwd("C:/Users/Scott/Downloads/statvis2/")
shinyServer(function(input, output) {
         file <- reactive({
                     paste0(input$version, input$type, "hw3.png")
         })
         output$Plot <- renderImage({
                              list(src = file(),
            contentType = "image/png",
            width = 400, height = 300)
         }, deleteFile = FALSE)
})
```

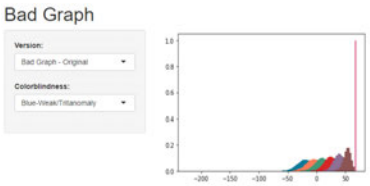
Figure 9: shiny Server.R



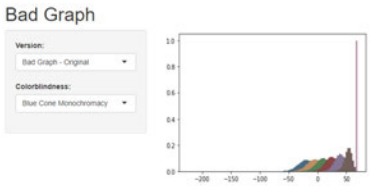
(a) Red-Weak/Protanomaly



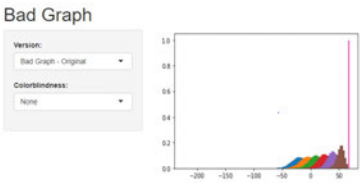
(b) Green-Weak/Deuteranomaly



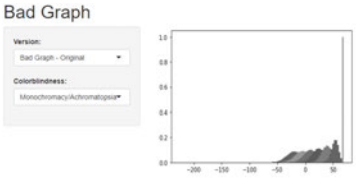
(c) Blue-Weak/Tritanomaly



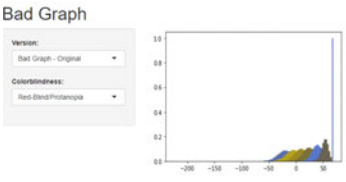
(a) Blue Cone Monochromacy



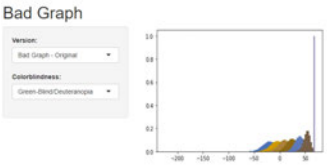
(b) Original Image



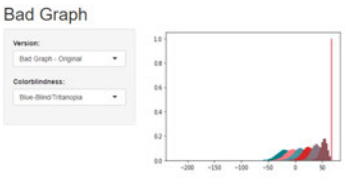
(c) Monochromacy/Achromatopsia



(a) Red-Blind/Protanopia



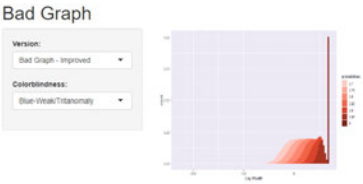
(b) Green-Blind/Deuteranopia



(c) Blue-Blind/Tritanopia

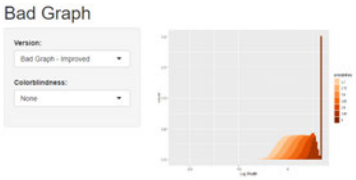
(a) Red-Weak/Protanomaly

(b) Green-Weak/Deuteranomaly

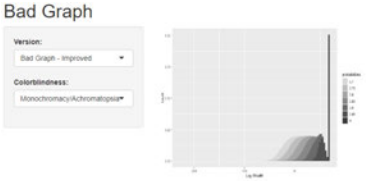


(c) Blue-Weak/Tritanomaly

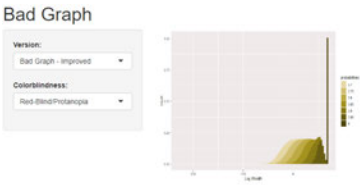
(a) Blue Cone Monochromacy



(b) Original Image



(c) Monochromacy/Achromatopsia



(a) Red-Blind/Protanopia

(b) Green-Blind/Deuteranopia

(c) Blue-Blind/Tritanopia

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

[redacted]

(iv) **Overlay of [redacted] from HW 2 on Street Map or Terrain Map** (25 Points):

Create an overlay of your previously created map for the [redacted] on a street map or terrain map. Choose any R package that works for you. If you work with *ggmap*, it may be easiest to ignore most of your work from HW 2 and reconsider how to use *ggplot* aesthetics (hint: the winning intervals) for a far more elegant solution. The trick will be to include a factor level for the unused interval for Jones. If you work with *RgoogleMaps*, you should be able to reuse most of your R code from HW 2. The Illinois example from the *Choropleth Maps Overlays* section of our lecture notes may be helpful for this question.

Do not forget to include a meaningful legend for all 10 intervals, although only 9 are used in this map. If you wonder why this is important, think of small multiples where you use the same colors for several other states as well. While the Democratic candidate did not win a single county with a percentage of 40% to 50% of the votes in Alabama, this may be the case in some counties in other states. Thus, we should not omit this interval/color from the Alabama legend.

Figure 16 shows an overlay on a Google map from Spring 2018 when the Google map server could still be freely accessed without a Google API key. Thus, your final figure should be similar, but you won't be able to produce exactly the same figure (unless you obtain a Google API key).

Answer:

```
library(httr)

## Warning:  package 'httr' was built under R version 4.0.4

library(XML)

## Warning:  package 'XML' was built under R version 4.0.3

library(RgoogleMaps)

## Warning:  package 'RgoogleMaps' was built under R version 4.0.5

library(PBSmapping)

## Warning:  package 'PBSmapping' was built under R version 4.0.5
##
## ------------------------------------------------------------
## PBS Mapping 2.73.0 -- Copyright (C) 2003-2021 Fisheries and Oceans Canada
##
## PBS Mapping comes with ABSOLUTELY NO WARRANTY;
## for details see the file COPYING.
## This is free software, and you are welcome to redistribute
## it under certain conditions, as outlined in the above file.
##
## A complete user guide 'PBSmapping-UG.pdf' is located at
## C:/Users/Scott/Documents/R/win-library/4.0/PBSmapping/doc/PBSmapping-UG.pdf
##
## Packaged on 2021-01-12
## Pacific Biological Station, Nanaimo
##
## All available PBS packages can be found at
## https://github.com/pbs-software
##
## To see demos, type '.PBSfigs()'.
## ------------------------------------------------------------

library(maps)

## Warning:  package 'maps' was built under R version 4.0.4

library(mapdata)

## Warning:  package 'mapdata' was built under R version 4.0.4

library(maptools)
```

```
## Warning:  package 'maptools' was built under R version 4.0.4

## Loading required package:  sp

## Warning:  package 'sp' was built under R version 4.0.4

## Checking rgeos availability:  FALSE
##  Note:  when rgeos is not available, polygon geometry computations in maptools depend on gpclib,
##  which has a restricted licence.  It is disabled by default;
##  to enable gpclib, type gpclibPermit()

library(RColorBrewer)

## Warning:  package 'RColorBrewer' was built under R version 4.0.3

library(data.table)

## Warning:  package 'data.table' was built under R version 4.0.3

page <- GET(paste0("https://www.nytimes.com/elections/results/alabam",
                   "a-senate-special-election-roy-moore-doug-jones"))

# extract the nodes related to the tables
pagehtml <- htmlParse(page)
nodes <- getNodeSet(pagehtml, "//table")

# extract the election table from the page
etable <- readHTMLTable(nodes[[2]])

colnames(etable) <- gsub(" |\n|[-]|[.]", "", colnames(etable))
etable$JonesNum <- as.integer(gsub(",", "", etable$Jones))
etable$MooreNum <- as.integer(gsub(",", "", etable$Moore))
etable$WriteInsNum <- as.integer(gsub(",", "", etable$WriteIns))

maxs <- as.vector(unlist(lapply(data.frame(t(etable[, 6:8])), max)))
percents <- (maxs/rowSums(etable[, 6:8]))*100
winner <- max.col(etable[, 6:8])
na.df <- data.frame(matrix(nrow = 67, ncol = 2))
na.df[winner == 1, 1] <- percents[winner == 1]
na.df[winner == 2, 2] <- percents[winner == 2]
names(na.df) <- c("JonesWinPct", "MooreWinPct")
p.table <- cbind(etable, na.df)



pdf("hw04_alabama_answer.pdf")
a.breaks <- c(49, 57, 65, 73, 81, 89)
a.break.class.d <- cut(p.table[, 9], a.breaks)
a.break.class.r <- cut(p.table[, 10], a.breaks)
d.col <- brewer.pal(10, "RdBu")[c(6, 7, 8, 9, 10)][a.break.class.d]
r.col <- brewer.pal(10, "RdBu")[c(5, 4, 3, 2, 1)][a.break.class.r]
a.col <-fcoalesce(d.col, r.col)
map.a.col <- a.col[order(gsub("^St", "Saint", etable[, 1]))]
a.counties <- map("county", "Alabama", fill = TRUE, border = "White",
                  col = map.a.col, plot = FALSE)
```

```
apiKey <- scan("bingmapkey.txt", what = "")
AL.IDs <- sub("^alabama,", "", a.counties$names)
AL_sp <- map2SpatialPolygons(a.counties, AL.IDs, CRS("+proj=longlat"))
bb <- qbbox(lon = coordinates(AL_sp)[, 1],
            lat = coordinates(AL_sp)[, 2])
MyALMap <- GetBingMap(mapArea = c(bb$latR[1], bb$lonR[1], bb$latR[2],
                                  bb$lonR[2]), maptype = "Road",
                      apiKey = apiKey, destfile = "AL_Bing.jpg")
AL.col <- sub("$", "66", map.a.col)
PlotPolysOnStaticMap(MyALMap, AL_sp, lwd = .5, col = AL.col,
                     add = FALSE)

title(main = paste0("\n Winning percents of counties in Alabama's ",
                    "2017 Special Senate Election"))
legend("left", legend = levels(a.break.class.d), title.adj = 0.12,
       title = " (D) Jones' Win% Interval", cex = 0.95, box.lty = 0,
       box.lwd = 0,
       fill = brewer.pal(10, "RdBu")[c(6, 7, 8, 9, 10)])
legend("bottomleft", legend = levels(a.break.class.r), box.lty = 0,
       box.lwd = 0,
       title.adj = 0.12, title = " (R) Moore's Win% Interval",
       cex = 0.95, fill = brewer.pal(10, "RdBu")[c(5, 4, 3, 2, 1)])
dev.off()


## pdf
##   2
```

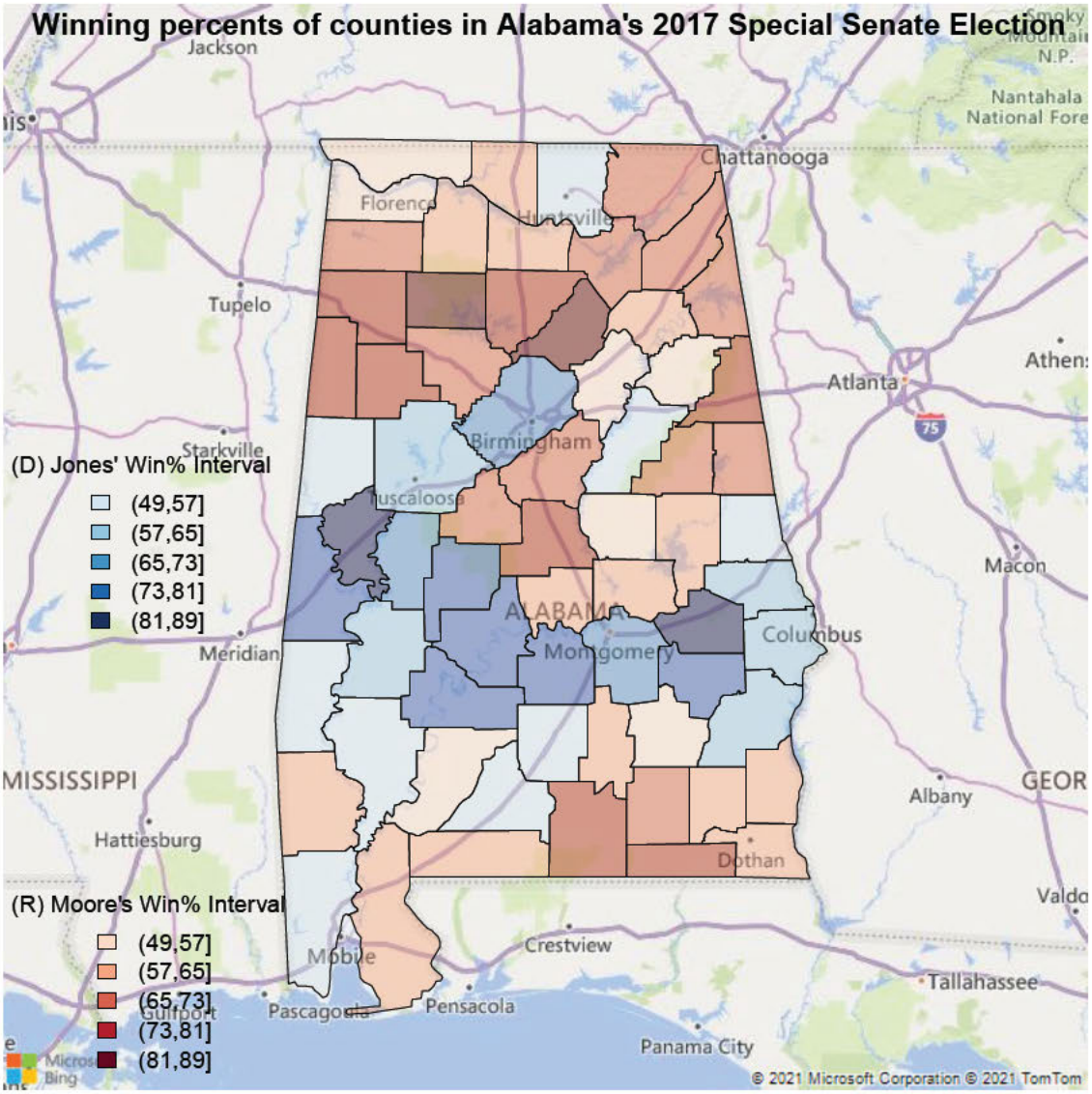


Figure 16: County–by–county Outcome of the 2017 Alabama Special Election in the Doug Jones (Dem) vs. Roy Moore (Rep) Senate Race. The map shows the winning percentage (in %) in each of the 67 Alabama counties.

General Instructions

(i) Create a single pdf document, using R Markdown, Sweave, or knitr. When you take this course at the 6000–level, you have to use LaTeX in combination with Sweave or knitr. You only have to submit this one document to Canvas.

(ii) Include a title page that contains your name, your A–number, the number of the assignment, the submission date, and any other relevant information.

(iii) Start your answers to each main question on a new page (continuing with the next part of a question on the same page is fine). Clearly label each question and question part. Your answer to question (i) should start on page 2!

(iv) Show your R code and resulting graph(s) [if any] for each question part!

(v) Before you submit your homework, check that you follow all recommendations from Google's R Style Guide (see `http://web.stanford.edu/class/cs109l/unrestricted/resources/google-style.html`). Moreover, make sure that your R code is consistent, i.e., that you use the same type of assignments and the same type of quotes throughout your entire homework.

(vi) Give credit to external sources, such as stackoverflow or help pages. Be specific and include the full URL where you found the help (or from which help page you got the information). Consider R code from such sources as "legacy code or third–party code" that does not have to be adjusted to Google's R Style (even though it would be nice, in particular if you only used a brief code segment).

(vii) **Not following the general instructions outlined above will result in point deductions!**

(viii) For general questions related to this homework, please use the corresponding discussion board in Canvas! I will try to reply as quickly as possible. Moreover, if one of you knows an answer, please post it. It is fine to refer to web pages and R commands, but do not provide the exact R command with all required arguments or which of the suggestions from a stackoverflow web page eventually worked for you! This will be the task for each individual student!

(ix) Submit your single pdf file via Canvas by the submission deadline. Late submissions will result in point deductions as outlined on the syllabus.

1 References

No references used