Process Book: Health Insurance Coverage in the United States

Grace Kong, Iris Chen & Fang Liu (Group P)

Overview and Motivation

One thing that was discussed a lot during the last president election is Obamacare. The Patient Protection and Affordable Care Act (ACA), also commonly known as Obamacare, was a major and wide-sweeping healthcare reform initiative enacted by President Obama. The ACA was signed into law by Obama on March 23, 2010, and changes were gradually implemented over the next few years, though the majority of the provisions went into effect only on January 1, 2014. Trump criticized Obamacare a lot. So will the ACA survive into 2017? Does it depend on the results of the 2016 election? Whether ACA is as terrible as trump said?

To answer those questions, we are interested in finding out how health insurance coverage in the USA differs from state to state, its relationship to indicators of health performance, and whether it can be predicted by political factors. More specifically, we want to see the impact of Affordable care act on the insurance coverage rate.

Quantitatively, we want see the change over time and the difference between states. We want to know in which states did insurance coverage expand under Obamacare? Which states observed the greatest decline in their uninsured rate? How does insurance coverage differ from state to state? We will also control some variables like income, mortality rate and rates of major diseases.

And then, we want to see whether higher insurance rate really benefit people. And how does health insurance coverage relate to participation uptake in preventative measures and treatment options (such as vaccinations, mammograms)?

And finally, we want to know what people think about Obamacare. To realize this, we will do some text analysis and see how do the sentiments towards Obamacare differ from state to state, and how does it relate to voting patterns? If feasible, we plan to analyze sentiments from twitter data, whether Obamacare is mentioned alongside positive or negative words.

We will make use of ggplot2, spatial data techniques, and text mining techniques in our visualization project.

Data

We used these data sources for our project:

1. General state-level data (population, age distribution and income distribution). United States Census Bureau: https://www.census.gov/data.html The United States Census bureau provides dataset that include information like population, average income and age distribution.
2. Insurance coverage and health indicators datasets State data on the Affordable Care Act (ACA) (from US National Library of Medicine): https://aspe.hhs.gov/compilation-state-data-affordable-care-act This state-level dataset includes coverage rates of the ACA, growth and expansion status of the ACA, employer vs. individual market coverage, and Medicaid / Medicare numbers. (filename: states_aca_general)
3. State scorecard on various health indicators (from the Commonwealth Fund): http://datacenter.commonwealthfund.org/#ind=1/sc=1 This state-level dataset has collated information about the health performance of the states along various dimensions. It includes insurance coverage rates, participation rates in prevention and treatment activities, and some measures of mortality. It also provides some differentiating information by race and income, allowing us to explore issues of equity.
4. Health Insurance Marketplace Data (from US Department of Health and Human Services): https://www.kaggle.com/hhs/health-insurance-marketplace This dataset contains information on health and dental plans offered through the US Health Insurance Marketplace.
5. Twitter API data & New York Times API data For the text analysis, we use twitter API data and New York Times API data. For the twitter API, we will try to extract text data on individuals' thoughts and opinions on Obamacare, capturing positive and negative sentiments. And we downloaded the New York Times articles from LexisNexis and then use R to transform them into corpus that can be analysis with R.

Downloading Twitter API data

The more challenging part is the API data. At first we planned to download all tweets that mentioned ACA from 2008 to 2016. However, we realized that will be too much data and will be hard to clean up and analysis. So we decided to download the New York Times articles and do the long term text analysis based on the New York Times articles. For twitter API, we only focus on the most recent 1000 tweets and try to have a sense of people's opinion about Affordable care act after the election.

Code for downloading tweets

Get some tweets from Twitter to analyze and visualize Set up Twitter API: Selecting data including Obamacare, ACA, Affordable Care Act, and #ACA (n=1000)

```
library(httr)
# library(oauth)
library(ROAuth)
library(twitteR)
library(RCurl)
library(RJSONIO)
library(stringr)
# secretkey myapp <- oauth_app('twitter', key = 'liLn6XJFenGjtvWFwi5LnDS1M',
# secret = 'dsCBm9Kyaeu9GMKlM9xwKl7eKmDn6qsjP31LQtwMGkF6OQdLh6') #Get OAuth
# credentials twitter_token <- oauth1.0_token(oauth_endpoints('twitter'),
# myapp) Declare Twitter API Credentials
api_key <- "liLn6XJFenGjtvWFwi5LnDS1M"  # From dev.twitter.com
api_secret <- "dsCBm9Kyaeu9GMKlM9xwKl7eKmDn6qsjP31LQtwMGkF6OQdLh6"  # From dev.twitter.com
token <- "772176811455381505-PYuNAEqhHFc02r83WS9Y5dnsZciIY5v"  # From dev.twitter.com token_secret <- '
# Create Twitter Connection
library("base64enc")
```

```r
setup_twitter_oauth(api_key, api_secret, token, token_secret)
# Run Twitter Search. Format is searchTwitter('Search Terms', n=100,
# lang='en', geocode='lat,lng', also accepts since and until).
tweets <- searchTwitter("Obamacare OR ACA OR 'Affordable Care Act' OR #ACA",
    n = 1000, lang = "en", since = "2014-08-20")
# Transform tweets list into a data frame tweets.df <- twListToDF(tweets)
# head(tweets.df,3)
```

Data cleaning

Plots of states trends

We first cleaned our data and unified the name of some variables as the data sets have different sources. We had to perform substantial data cleaning, including transforming reshaping one of our datatsets to a wide format from its followingoriginal state. This was pretty challenging maneuver to figure out in terms of the data wrangling.

Original states health indicator data

Commonwealth Fund Scorecard on State Health System Performance, 2017 Edition

Citation: D. C. Radley, D. McCarthy, and S. L. Hayes, *Aiming Higher: Results from a Scorecard on State Health System Performance, 2017 Edition*, The Commonwe

State	Scorecard Dimension	Performance Measure	Data Year	Rate	Rank[1]
Alabama	Access & Affordabilty	a100: Access Dimension Summary	2017		34
Alabama	Access & Affordabilty	a100: Access Dimension Summary	Baseline		27
Alabama	Access & Affordabilty	a001: Adults ages 19–64 uninsured	2015	16	37
Alabama	Access & Affordabilty	a001: Adults ages 19–64 uninsured	2013	20	30
Alabama	Access & Affordabilty	a002: Children ages 0–18 uninsured	2015	3	3
Alabama	Access & Affordabilty	a002: Children ages 0–18 uninsured	2013	5	6
Alabama	Access & Affordabilty	a004: Adults who went without care because of cost in past year	2015	17	48
Alabama	Access & Affordabilty	a004: Adults who went without care because of cost in past year	2013	16	31
Alabama	Access & Affordabilty	a005: Individuals under age 65 with high out-of-pocket medical costs relative to their annu	2014-15	17	41
Alabama	Access & Affordabilty	a006: At-risk adults without a routine doctor visit in past two years	2015	12	19
Alabama	Access & Affordabilty	a006: At-risk adults without a routine doctor visit in past two years	2013	12	13
Alabama	Access & Affordabilty	a003: Adults without a dental visit in past year	2014	18	41
Alabama	Access & Affordabilty	a003: Adults without a dental visit in past year	2012	18	41
Alabama	Prevention & Treatment	q100: Prevention and Treatment Dimension Summary	2017		42
Alabama	Prevention & Treatment	q100: Prevention and Treatment Dimension Summary	Baseline		36
Alabama	Prevention & Treatment	q002: Adults with a usual source of care	2015	79	24
Alabama	Prevention & Treatment	q002: Adults with a usual source of care	2013	78	21
Alabama	Prevention & Treatment	q001a: Adults with age- and gender-appropriate cancer screenings	2014	67	28
Alabama	Prevention & Treatment	q001a: Adults with age- and gender-appropriate cancer screenings	2012	68	24
Alabama	Prevention & Treatment	q001b: Adults with age-appropriate vaccines	2015	38	30
Alabama	Prevention & Treatment	q001b: Adults with age-appropriate vaccines	2013	38	24
Alabama	Prevention & Treatment	q003: Children with a medical home	2011/12	54	38
Alabama	Prevention & Treatment	q004: Children with a medical and dental preventive care visit in the past year	2011/12	70	18
Alabama	Prevention & Treatment	q005: Children with emotional, behavioral, or developmental problems who received neec	2011/12	54	42
Alabama	Prevention & Treatment	q006: Children ages 19–35 months who received all recommended doses of seven key vac	2015	71	31
Alabama	Prevention & Treatment	q006: Children ages 19–35 months who received all recommended doses of seven key vac	2013	77	6
Alabama	Prevention & Treatment	q014: Medicare beneficiaries who received at least one drug that should be avoided in the	2014	18	48
Alabama	Prevention & Treatment	q014: Medicare beneficiaries who received at least one drug that should be avoided in the	2012	24	50
Alabama	Prevention & Treatment	q015: Medicare beneficiaries with dementia, hip/pelvic fracture, or chronic renal failure w	2014	23	50
Alabama	Prevention & Treatment	q015: Medicare beneficiaries with dementia, hip/pelvic fracture, or chronic renal failure w	2012	28	51
Alabama	Prevention & Treatment	q016: Medicare fee-for-service patients whose health provider always listens, explains, sh	2014	74	38
Alabama	Prevention & Treatment	q016: Medicare fee-for-service patients whose health provider always listens, explains, sh	2013	74	40
Alabama	Prevention & Treatment	q009: Risk-adjusted 30-day mortality among Medicare beneficiaries hospitalized for heart	07/2012 - 06/2015	15	38
Alabama	Prevention & Treatment	q009: Risk-adjusted 30-day mortality among Medicare beneficiaries hospitalized for heart	07/2010 - 06/2013	13.7	44
Alabama	Prevention & Treatment	q022: Central line-associated bloodstream infections (CLABSI), Standardized Infection Rati	2014	0.71	48
Alabama	Prevention & Treatment	q022: Central line-associated bloodstream infections (CLABSI), Standardized Infection Rati	2013	0.67	42
Alabama	Prevention & Treatment	q010: Hospitalized patients given information about what to do during their recovery at hc	2015	86	35
Alabama	Prevention & Treatment	q010: Hospitalized patients given information about what to do during their recovery at hc	2013	85	32

Figure 1:

We relabelled our variables extensively from the original excel sheets they came in, and came up with variable conversion lists such as the following. This helped us to get a bird-eye view of our available health insurance-related variables to work with. We were fortunate to find that we had much information in health and financial outcomes. Too much, in fact, so we tried to distill the variables down, highlighting the useful ones.

Code - data cleaning

Our full data cleaning code is below.

Data cleaning - states health trends

```
rm(list = ls())

library(leaflet)
library(maps)
library(rgdal)
```

```
## Loading required package: sp

## rgdal: version: 1.2-5, (SVN revision 648)
##  Geospatial Data Abstraction Library extensions to R successfully loaded
##  Loaded GDAL runtime: GDAL 2.1.2, released 2016/10/24
##  Path to GDAL shared files: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/rgdal/gdal
##  Loaded PROJ.4 runtime: Rel. 4.9.1, 04 March 2015, [PJ_VERSION: 491]
##  Path to PROJ.4 shared files: /Library/Frameworks/R.framework/Versions/3.3/Resources/library/rgdal/pr
##  Linking to sp version: 1.2-4
```

```
library(DT)
library(ggplot2)
library(ggthemes)
library(plotly)
```

```
##
## Attaching package: 'plotly'

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(magrittr)
library(readxl)
library(plyr)
```

```
##
## Attaching package: 'plyr'

## The following objects are masked from 'package:plotly':
##
##     arrange, mutate, rename, summarise

## The following object is masked from 'package:maps':
##
##     ozone
```

```
library(dplyr)
```

```
##
```

```
## Attaching package: 'dplyr'

## The following objects are masked from 'package:plyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
```

```
##
## Attaching package: 'tidyr'

## The following object is masked from 'package:magrittr':
##
##     extract
```

```r
library(readr)
library(stringi)
library(RColorBrewer)
library(countrycode)
require(gridExtra)
```

```
## Loading required package: gridExtra

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:dplyr':
##
##     combine
```

```r
## IMPORT DATA

# General ACA information
aca_general_raw <- read_excel("Data/states_aca_general.xlsx", sheet = 2, col_names = FALSE)

# States health outcomes
health_ind_raw <- read_excel("Data/states_health_ind.xlsx", sheet = 4, col_names = FALSE)

# Insurance coverage data Long format
insurance_long <- read_excel("Data/insurance-clean.xlsx", sheet = 1, col_names = TRUE)
# Wide format
insurance <- read_excel("Data/insurance-clean.xlsx", sheet = 2, col_names = TRUE)

# Demographic information
demographics1 <- read.csv("Data/Population_Age_Income.csv", header = TRUE)
# Provided with Assignment 4 (want the yearly population data)
demographics2 <- read_excel("Data/PopulationEstimates.xls")

# Election Results
```

```r
election <- read_excel("Data/US_Presidential_Results_by_State_1828-2016.xlsx",
    sheet = 2, col_names = FALSE)
```

```r
## CLEAN DATA

# List of states abbreviations
state_abbreviations <- c("AL", "AK", "AZ", "AR", "CA", "CO", "CT", "DE", "FL",
    "GA", "HI", "ID", "IL", "IN", "IA", "KS", "KY", "LA", "ME", "MD", "MA",
    "MI", "MN", "MS", "MO", "MT", "NE", "NV", "NH", "NJ", "NM", "NY", "NC",
    "ND", "OH", "OK", "OR", "PA", "RI", "SC", "SD", "TN", "TX", "UT", "VT",
    "VA", "WA", "WV", "WI", "WY")


## CLEAN INSURANCE RATES DATA (WIDE)

# Drop unnecessary columns
insurance <- insurance[1:50, -1]

# Rename variables
colnames(insurance) <- c("state_abb", "state", "uninsured_num_2008", "uninsured_pct_2008",
    "uninsured_num_2009", "uninsured_pct_2009", "uninsured_num_2010", "uninsured_pct_2010",
    "uninsured_num_2011", "uninsured_pct_2011", "uninsured_num_2012", "uninsured_pct_2012",
    "uninsured_num_2013", "uninsured_pct_2013", "uninsured_num_2014", "uninsured_pct_2014",
    "uninsured_num_2015", "uninsured_pct_2015")

# Create variables for insurance rates
insurance$insured_pct_2008 <- 100 - insurance$uninsured_pct_2008
insurance$insured_pct_2009 <- 100 - insurance$uninsured_pct_2009
insurance$insured_pct_2010 <- 100 - insurance$uninsured_pct_2010
insurance$insured_pct_2011 <- 100 - insurance$uninsured_pct_2011
insurance$insured_pct_2012 <- 100 - insurance$uninsured_pct_2012
insurance$insured_pct_2013 <- 100 - insurance$uninsured_pct_2013
insurance$insured_pct_2014 <- 100 - insurance$uninsured_pct_2014
insurance$insured_pct_2015 <- 100 - insurance$uninsured_pct_2015

# Reorder variables
insurance <- insurance[, c("state_abb", "state", "uninsured_num_2008", "uninsured_pct_2008",
    "insured_pct_2008", "uninsured_num_2009", "uninsured_pct_2009", "insured_pct_2009",
    "uninsured_num_2010", "uninsured_pct_2010", "insured_pct_2010", "uninsured_num_2011",
    "uninsured_pct_2011", "insured_pct_2011", "uninsured_num_2012", "uninsured_pct_2012",
    "insured_pct_2012", "uninsured_num_2013", "uninsured_pct_2013", "insured_pct_2013",
    "uninsured_num_2014", "uninsured_pct_2014", "insured_pct_2014", "uninsured_num_2015",
    "uninsured_pct_2015", "insured_pct_2015")]


## CLEAN INSURANCE RATES DATA (LONG)

# Rename variables
colnames(insurance_long) <- c("year", "state", "population", "uninsured_num",
    "uninsured_pct", "insured_num", "insured_pct")

# Add state abbreviations First create a look-up table (based on wide data)
state_abb_lookup <- insurance[, c("state", "state_abb")]
# Then do the matching
```

```
insurance_long$state_abb <- state_abb_lookup$state_abb[match(insurance_long$state,
    state_abb_lookup$state)]
# Reorder variable
insurance_long <- insurance_long[, c(1:2, ncol(insurance_long), 3:(ncol(insurance_long) -
    1))]

# Order observations (by state, then by year)
insurance_long <- insurance_long[order(insurance_long$state, insurance_long$year),
    ]


## CLEAN ACA GENERAL DATA

# Keep only relevant rows and columns
aca_general <- aca_general_raw[6:57, 1:73]

# Label variables for ACA general data
names(aca_general) <- c("state", "unins_all_pct_10", "unins_all_pct_15", "unins_all_decr_pct",
    "unins_all_decr", "cov_emp", "cov_parents_plan", "lifetime_lim_preACA_tot",
    "lifetime_lim_preACA_child", "lifetime_lim_preACA_adultM", "lifetime_lim_preACA_adultF",
    "cov_private_tot", "cov_private_child", "cov_private_adultM", "cov_private_adultF",
    "premium_emp_avg_growth_pct_00to10", "premium_emp_avg_growth_pct_10to15",
    "premium_emp_savings_15", "premium_emp_savings_16", "MLR_rebate_beneficiaries_12",
    "MLR_rebate_amt_12", "MLR_rebate_beneficiaries_13", "MLR_rebate_amt_13",
    "MLR_rebate_beneficiaries_14", "MLR_rebate_amt_14", "MLR_rebate_beneficiaries_15",
    "MLR_rebate_amt_15", "MLR_rebate_amt_12to15", "medicaid_enroll_13", "medicaid_enroll_16",
    "medicaid_enroll_child_13", "medicaid_enroll_incr_13to16", "medicaid_full_duals",
    "medicaid_partial_duals", "medicaid_full_or_partial_duals", "state_has_expanded",
    "insurance_incr_medicaid", "cholest_scr_incr_medicaid", "mammogram_incr_medicaid",
    "papsmear_incr_medicaid", "clinic_care_incr_medicaid", "all_care_incr_medicaid",
    "phycisian_visit_ann_incr_medicaid", "depression_decr_medicaid", "good_health_incr_medicaid",
    "deaths_ann_decr_medicaid", "catastrophic_oop_ann_decr_medicaid", "indebted_ppl_decr_medicaid",
    "fed_spending_net_incr_inMil", "uncompensated_care_decr_inMil", "mental_substance_elig_medicaid_sha
    "mental_substance_elig_medicaid", "preexisting_condition_09", "cov_mkt_plan_16",
    "cov_mkt_16", "receive_tax_credit", "avg_tax_credit", "receive_cost_sharing",
    "avg_num_mkt_plans_avail_17", "cov_mkt_under75D_pct", "cov_mkt_under100D_pct",
    "cov_offmkt_elig_tax_credit", "rate_review_funds_to_state", "HIECP_grant_award_to_state",
    "medicare_enroll_16", "medicare_benef_donuthole", "medicare_benef_donuthole_savings_tot",
    "medicare_benef_donuthole_savings_avg", "medicare_partB_free_prev_services",
    "medicare_partB_free_prev_services_share", "medicare_incr_readmit_rate",
    "medicare_avoided_readmit", "accountable_care_org_num")

# Trim white space on state variable
aca_general$state <- stri_trim_both(aca_general$state)

# Change format of variables Apart from T/F variable for 'state has
# expanded'
aca_general$state_has_expanded[aca_general$state_has_expanded == "yes"] <- TRUE
aca_general$state_has_expanded[aca_general$state_has_expanded == "no"] <- FALSE
# Convert others to numeric form
for (j in c(2:35, 37:73)) {
    aca_general[[j]] <- as.numeric(aca_general[[j]])
}
```

```r
# Save USA data separately
aca_general_USA <- aca_general[aca_general$state == "United States", ]

# Drop USA and DC data
aca_general <- aca_general[!(aca_general$state %in% c("United States", "District of Columbia")),
    ]

# Add state abbreviations
aca_general$state_abb <- state_abbreviations
aca_general <- aca_general[, c(1, ncol(aca_general), 2:(ncol(aca_general) -
    1))]

# Drop raw data
rm(aca_general_raw)


## CLEAN HEALTH INDICATOR DATA

# Keep only relevant rows and columns
health_ind <- health_ind_raw[4:6703, c(1, 3:6)]

# Label variables
names(health_ind) <- c("state", "measure", "year", "rate", "rank")

# Simplify date (years) - only keep last year if range was given
health_ind$year[health_ind$year == "07/2009 - 06/2012"] <- "2012"
health_ind$year[health_ind$year == "07/2010 - 06/2013"] <- "2013"
health_ind$year[health_ind$year == "07/2012 - 06/2015"] <- "2015"
health_ind$year[health_ind$year == "10/2013-9/2014"] <- "2014"
health_ind$year[health_ind$year == "2008-09"] <- "2009"
health_ind$year[health_ind$year == "2010-11"] <- "2011"
health_ind$year[health_ind$year == "2011-12"] <- "2012"
health_ind$year[health_ind$year == "2011/12"] <- "2012"
health_ind$year[health_ind$year == "2012-13"] <- "2013"
health_ind$year[health_ind$year == "2013-14"] <- "2014"
health_ind$year[health_ind$year == "2013(Q2-Q4)"] <- "2013"
health_ind$year[health_ind$year == "2014-15"] <- "2015"
health_ind$year[health_ind$year == "2015(Q2-Q4)"] <- "2015"

# Export to excel to manually label variable names
unique_measures <- unique(as.factor(health_ind$measure))
# Copied in from excel sheet
unique_measures_relabelled <- c("a.summary_access", "a.unins_adult", "a.unins_child",
    "a.no_care_bc_cost_adult", "a.high_OOP_relative_under65", "a.at_risk_no_routine_doc_adult",
    "a.no_dental_adult", "q.summary_prev_treat", "q.with_usual_care_adult",
    "q.with_cancer_screening_adult", "q.with_vaccines_adult", "q.with_medical_home_child",
    "q.with_prev_medical_dental_child", "q.with_mental_healthcare_child", "q.with_vaccines_infant",
    "q.drug_should_avoid_medicare", "q.drug_should_avoid_3conditions_medicare",
    "q.good_health_provider_medicare", "q.mortality_4conditions_medicare", "q.CLABSI_infection_ratio",
    "q.info_recovery_hospitalized", "q.good_hospital_staff_hospitalized", "q.improve_mobility_homehealth
    "q.improved_wounds_homehealth", "q.sores_NHres", "q.antipsychotic_med_NHres",
    "u.summary_avoidable_hosp_cost", "u.hosp_asthma_child", "u.hosp_ambulatory_65to74yrs",
    "u.hosp_ambulatory_above75yrs", "u.30day_hosp_readmit_medicare", "u.30day_hosp_readmit_NHres",
```

```
    "u.hosp_6mos_NHres", "u.hosp_medicare_homehealth", "u.avoidable_ER_medicare",
    "u.tot_reimb_employer_ins", "u.tot_reimb_medicare", "h.summary_healthy_lives",
    "h.deaths_amenable", "h.yrs_lost_potential_life_before75", "h.deaths_breast_cancer_F",
    "h.deaths_colorectal_cancer", "h.deaths_suicide", "h.deaths_infant_mortality",
    "h.poor_health_adult", "h.smoke_adult", "h.obese_adult", "h.obese_child",
    "h.poor_dental_adult", "u.premium_emp_private", "u.premium_emp_private_unadj",
    "u.reimb_medicare_unadj", "u.deaths_amenable_black", "u.deaths_amenable_white",
    NA, "q.with_prev_screening_above50yrs", NA)

# Replace variable names with abbreviated version
for (i in 1:57) {
    health_ind$measure <- replace(health_ind$measure, health_ind$measure ==
        unique_measures[i], unique_measures_relabelled[i])
}

# Remove the NA's
health_ind <- health_ind[!is.na(health_ind$measure), ]

# Remove duplicates By state, measure.year
health_ind <- health_ind[!duplicated(health_ind[, 1:3]), ]

# Interact measure and year
health_ind$measure_year <- interaction(health_ind$measure, health_ind$year)

# Convert to character variable
health_ind$measure_year <- as.character(health_ind$measure_year)

# Pre-reshape: Store existing data as 'long'
health_ind_long <- health_ind

# Reshape wide
health_ind <- as.data.frame(cbind(health_ind$state, health_ind$measure_year,
    health_ind$rate))
# Drop rank data for now
names(health_ind) <- c("state", "measure_year", "rate")
health_ind <- reshape(health_ind, idvar = "state", timevar = "measure_year",
    direction = "wide")

# Drop 'rate' from the name
names(health_ind)[2:ncol(health_ind)] <- substr(names(health_ind), 6, nchar(names(health_ind)))[2:ncol(

# Drop variables who have all entries as NA Due to 'unbalanced panel' in
# reshaping First create a vector to record if variable is all NA
variable_all_NA <- vector(mode = "logical", length = ncol(health_ind))
for (j in 1:ncol(health_ind)) {
    if (sum(is.na(health_ind[[j]])) == 52) {
        variable_all_NA[j] <- TRUE
    }
}
# Only keep the variables that are NOT all NA
health_ind <- health_ind[, !variable_all_NA]

# Change format of variables To character variable
```

```r
health_ind$state <- as.character(health_ind$state)
# To numeric variable
for (j in c(2:164)) {
    health_ind[[j]] <- as.character(health_ind[[j]])
    health_ind[[j]] <- as.numeric(health_ind[[j]])
}

# Save USA data separately
health_ind_USA <- health_ind[health_ind$state == "United States", ]

# Drop USA and DC data
health_ind <- health_ind[!(health_ind$state %in% c("United States", "District of Columbia")),
    ]

# Label state abbreviations
health_ind$state_abb <- state_abbreviations
health_ind <- health_ind[, c(1, ncol(health_ind), 2:(ncol(health_ind) - 1))]

# Drop raw data
rm(health_ind_raw)


## CLEAN DEMOGRAPHIC (1) DATA

# Drop unnecessary row and column
demographics1 <- demographics1[-1, -1]

# Rename variables
colnames(demographics1) <- c("state", "income", "population", "ppl_age0to4",
    "ppl_age5to9", "ppl_age10to14", "ppl_age15to19", "ppl_age20to24", "ppl_age25to29",
    "ppl_age30to34", "ppl_age35to39", "ppl_age40to44", "ppl_age45to49", "ppl_age50to54",
    "ppl_age55to59", "ppl_age60to64", "ppl_age65to69", "ppl_age70to74", "ppl_age75to79",
    "ppl_age80to84", "ppl_age85plus")

# Drop observations DC and PR
demographics1 <- demographics1[!demographics1$state == "District of Columbia" &
    !demographics1$state == "Puerto Rico", ]

# Label state abbreviations
demographics1$state_abb <- state_abbreviations
demographics1 <- demographics1[, c(1, ncol(demographics1), 2:(ncol(demographics1) -
    1))]

# Change format of variables To character variable
demographics1$state <- as.character(demographics1$state)
# To numeric variable
for (j in c(3:ncol(demographics1))) {
    demographics1[[j]] <- as.character(demographics1[[j]])
    demographics1[[j]] <- as.numeric(demographics1[[j]])
}

# BEA regions
demographics1$BEA_region <- ""
```

```r
demographics1$BEA_region[demographics1$state_abb %in% c("ME", "NH", "VT", "MA",
    "CT", "RI")] <- "New England"
demographics1$BEA_region[demographics1$state_abb %in% c("NY", "NJ", "PA", "MD",
    "DE")] <- "Mideast"
demographics1$BEA_region[demographics1$state_abb %in% c("WI", "IL", "MI", "IN",
    "OH")] <- "Great Lakes"
demographics1$BEA_region[demographics1$state_abb %in% c("WV", "KY", "VA", "TN",
    "NC", "SC", "AR", "LA", "MS", "AL", "GA", "FL")] <- "Southeast"
demographics1$BEA_region[demographics1$state_abb %in% c("ND", "SD", "NE", "KS",
    "MN", "IA", "MO")] <- "Plains"
demographics1$BEA_region[demographics1$state_abb %in% c("MT", "ID", "WY", "UT",
    "CO")] <- "Rocky Mountains"
demographics1$BEA_region[demographics1$state_abb %in% c("AZ", "NM", "TX", "OK")] <- "Southwest"


demographics1$BEA_region[demographics1$state_abb %in% c("WA", "OR", "CA", "NV",
    "AK", "HI")] <- "Far West"

# Tag income level (category)
demographics1$income_level <- ""
demographics1$income_level[demographics1$income >= 60000] <- "High"
demographics1$income_level[demographics1$income >= 50000 & demographics1$income <
    60000] <- "Upper Middle"
demographics1$income_level[demographics1$income >= 45000 & demographics1$income <
    50000] <- "Lower Middle"
demographics1$income_level[demographics1$income < 45000] <- "Low"
# Convert to factor variable
demographics1$income_level <- factor(demographics1$income_level, levels = c("Low",
    "Lower Middle", "Upper Middle", "High"))


## CLEAN DEMOGRAPHIC (2) DATA

colnames(demographics2) <- demographics2[2, ]

# Drop first few rows (non-observations)
demographics2 <- demographics2[-(1:2), ]

# Drop county-level data, keep state-level data only
demographics2 <- demographics2[demographics2$Area_Name == "United States" |
    demographics2$State != lag(demographics2$State), ]

# Keep only columns of population from 2010 to 2015
demographics2 <- demographics2[, c("State", "Area_Name", "POP_ESTIMATE_2010",
    "POP_ESTIMATE_2011", "POP_ESTIMATE_2012", "POP_ESTIMATE_2013", "POP_ESTIMATE_2014",
    "POP_ESTIMATE_2015")]

# Change format of variables To character variable
for (j in c(3:8)) {
    demographics2[[j]] <- as.numeric(demographics2[[j]])
}

# Rename variables
```

```r
colnames(demographics2) <- c("state_abb", "state", "population_2010", "population_2011",
    "population_2012", "population_2013", "population_2014", "population_2015")


## CLEAN ELECTION DATA

# Keep only 2012 and 2008 data, and data for the 50 states
election <- election[4:53, c("X0", "X5", "X6", "X9", "X10")]

# Rename variables
colnames(election) <- c("state", "Dem_pct_2012", "Rep_pct_2012", "Dem_pct_2008",
    "Rep_pct_2008")

# Add state abbreviations Match with look-up table
election$state_abb <- state_abb_lookup$state_abb[match(election$state, state_abb_lookup$state)]

# Change format of variables To numeric variable
for (j in c(4:6)) {
    election[[j]] <- as.numeric(election[[j]])
}

# Calculate winning party for each state
election$party_2012 <- ifelse(election$Dem_pct_2012 > election$Rep_pct_2012,
    "Democratic", "Republican")
election$party_2008 <- ifelse(election$Dem_pct_2008 > election$Rep_pct_2008,
    "Democratic", "Republican")

# Reorder variables
election <- election[, c("state", "state_abb", "Dem_pct_2012", "Rep_pct_2012",
    "party_2012", "Dem_pct_2008", "Rep_pct_2008", "party_2008")]
```

Data cleaning - NY Times articles

```r
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```r
library(tm.plugin.lexisnexis)
library(readxl)
library(gtools)  # for smartbind
library(dplyr)  # for data_frame
library(lubridate)  # for date formatting
```

```
##
## Attaching package: 'lubridate'
```

```
## The following object is masked from 'package:plyr':
##
##     here
```

```
## The following object is masked from 'package:base':
##
##     date
```

```r
library(stringr)
library(tools)  # Title case
library(quanteda)
```

```
## quanteda version 0.9.9.24

## Using 3 of 4 cores for parallel computing

##
## Attaching package: 'quanteda'

## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, stopwords

## The following object is masked from 'package:NLP':
##
##     ngrams

## The following object is masked from 'package:readr':
##
##     tokenize

## The following object is masked from 'package:utils':
##
##     View

## The following object is masked from 'package:base':
##
##     sample
```

```r
# library(ggplot2)
library(quanteda)
library(stringr)
library(tm)
library(qdap)
```

```
## Loading required package: qdapDictionaries

## Loading required package: qdapRegex

##
## Attaching package: 'qdapRegex'

## The following objects are masked from 'package:dplyr':
##
##     escape, explain

## The following object is masked from 'package:ggplot2':
##
##     %+%

## Loading required package: qdapTools

##
## Attaching package: 'qdapTools'

## The following object is masked from 'package:dplyr':
```

```
##
##     id
## The following object is masked from 'package:plyr':
##
##     id
##
## Attaching package: 'qdap'
## The following objects are masked from 'package:quanteda':
##
##     as.DocumentTermMatrix, as.wfm, ngrams, weight
## The following object is masked from 'package:stringr':
##
##     %>%
## The following objects are masked from 'package:tm':
##
##     as.DocumentTermMatrix, as.TermDocumentMatrix
## The following object is masked from 'package:NLP':
##
##     ngrams
## The following object is masked from 'package:tidyr':
##
##     %>%
## The following object is masked from 'package:dplyr':
##
##     %>%
## The following object is masked from 'package:magrittr':
##
##     %>%
## The following object is masked from 'package:plotly':
##
##     %>%
## The following object is masked from 'package:DT':
##
##     %>%
## The following object is masked from 'package:leaflet':
##
##     %>%
## The following object is masked from 'package:base':
##
##     Filter
```

```r
library(SnowballC)
library(dplyr)
library(tidytext)
library(wordcloud)

# Combine CSV and HTML Files
data <- read_excel("LexisNexis/NYTimes_Metadata.xlsx")
```

```r
colnames(data) <- tolower(colnames(data))

# Correct data
data$date <- substr(parse_date_time(data$date, c("mdy")), 1, 10)
data$author <- toTitleCase(tolower(data$byline))
data$byline <- NULL

## Get Text files
source1 <- LexisNexisSource("LexisNexis/The_New_York1.html")
source2 <- LexisNexisSource("LexisNexis/The_New_York2.html")

corpus1 <- Corpus(source1, readerControl = list(language = NA))
corpus2 <- Corpus(source2, readerControl = list(language = NA))

corpus <- c(corpus1, corpus2)

# Convert to quanteda corpus
corpus <- quanteda::corpus(corpus)

## Add Metadata Check: match(data$headline, corpus$documents$heading)
corpus$documents$datetimestamp <- substring(corpus$documents$datetimestamp,
    1, 4)
corpus$documents$date <- corpus$documents$datetimestamp
corpus$documents$description <- corpus$documents$id

# options(width = 200) kwic(corpus, 'Trump')
save(corpus, file = "nytimes.rda")
```

Data cleaning - Twitter API

```r
counts = table(tweets.df$screenName)
barplot(counts)
# Let's do something hacky: Limit the data set to show only folk who tweeted
# twice or more in the sample cc=subset(counts,counts>1)
barplot(cc, las = 2, cex.names = 0.3)
```

Design Evolution and Planning Process

We started from drawing the graphs on paper and whiteboard sessions. As we already have a general idea of our datasets, we were able to create some graph that can visualize information from different datasets. We divided our project into three sections.

The following was our original plan of the plots, though it had changed significantly since then.

The first section focus on the general insurance coverage in the US.

Graph 1: The first graph will show the general domestic information and also an insurance coverage rate.

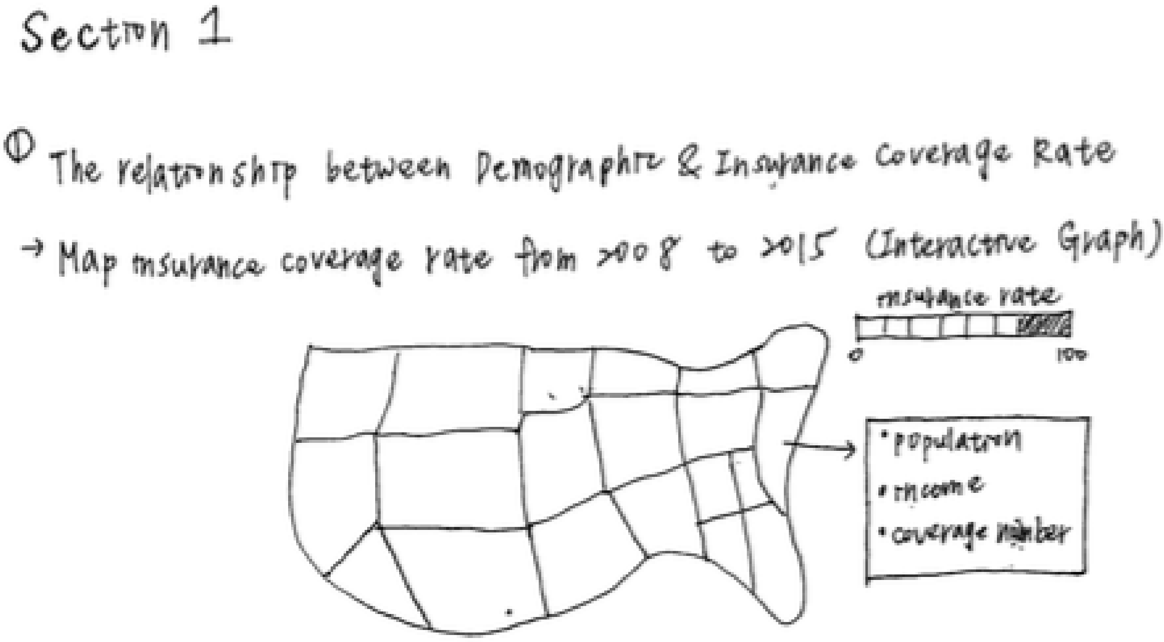


Figure 2:

Graph 2 and 3 focus on insurance rate. Graph 2 will show the trend of uninsured rate while graph 3 is a bar graph that will show the source of insurance by region

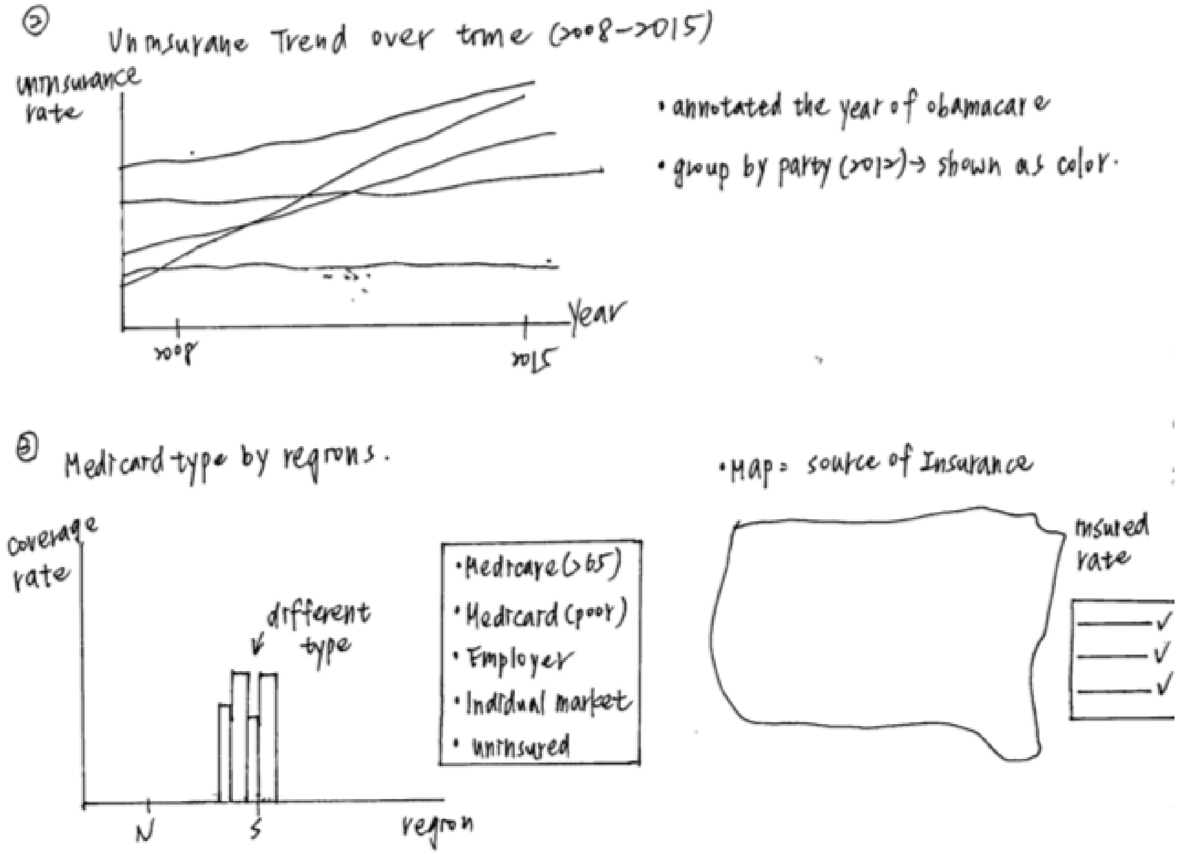


Figure 3:

Section 2 tries to see the insurance coverage in depth. We want to control some variables and see their relationship with insurance coverage rate.

The graph 4 is supposed to show the relationship between mortality rate and insurance coverage rate. We want to control some omitted variable like income, and expansion of Obamacare,

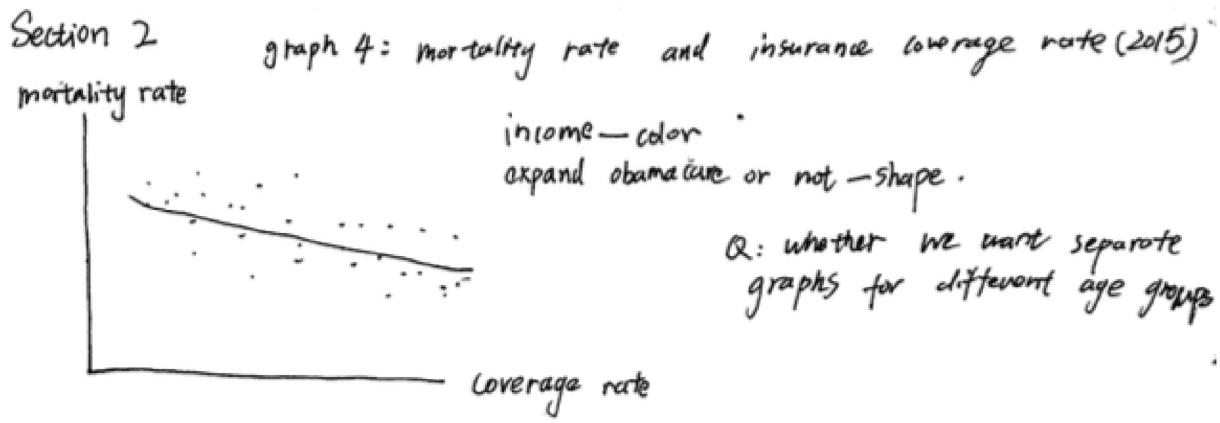


Figure 4:

Graph 5 will try to evaluate the relationship between the access of healthcare and insurance coverage. We divided health services into several categories based on our dataset like vaccine, dental and so on.

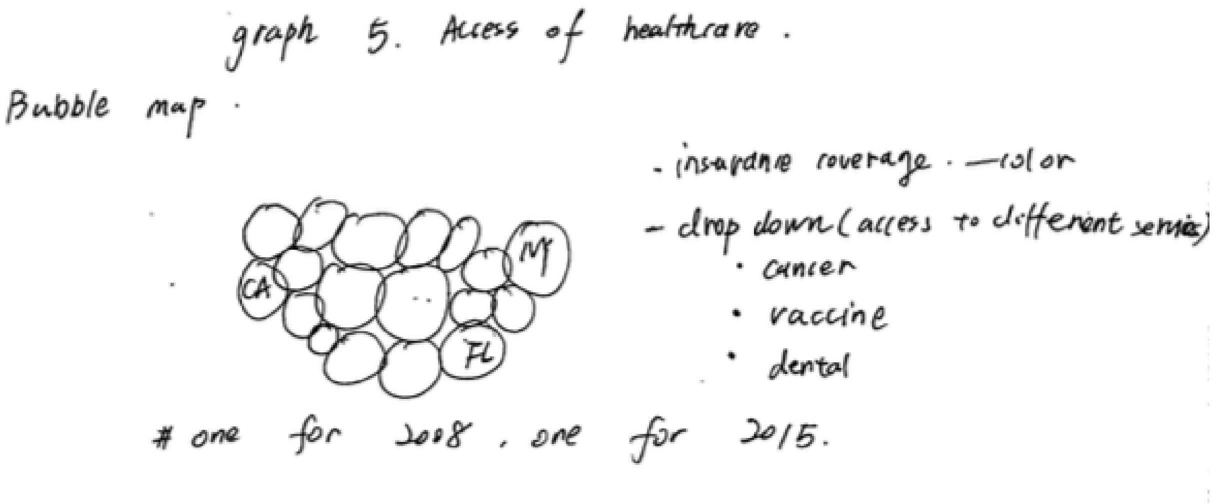


Figure 5:

The section 3 is the text analysis. We planned to first have a look of the change of numbers of tweets mentioned ACA and then sentiment analysis of tweets mentioned ACA over years.

Code for data exploration and refinement of plots

Shiny App

Figure 1

We attempted to create a simply Shiny App and it was pretty challenging! We took quite a while to figure out reactivity when it came to updating the map. Our shiny app allowed users to use a slider to control the year, and have the US map of uninsured rates shift in response.

Shiny Code - Data cleaning

```
# Import insurance data
insurance_shiny <- read.csv("shiny_insurance.csv")
# insurance_shiny <- read.csv('/Users/gracekongyx/Documents/*6_Data
# Science/QMSS G4063 Data Visualization/Final Project/R
# code/Health_Insurance_Shiny/shiny_insurance.csv') insurance_shiny <-
# read.csv('/Users/gracekongyx/Documents/*6_Data Science/QMSS G4063 Data
# Visualization/Final Project/Data/shiny_insurance.csv')

## DATA PROCESSING

# Save list of state abbreviations
state_abb_lookup <- insurance_shiny[, c("state", "state_abb")]

# Obtain shape files of US states
map_states = map("state", fill = TRUE, plot = FALSE)

# Standardize name format with rest of assignment Function to conver to
# proper case
properCase <- function(x) {
    s <- strsplit(x, " ")[[1]]
    paste(toupper(substring(s, 1, 1)), substring(s, 2), sep = "", collapse = " ")
}
# Convert to proper case
map_states$state <- sapply(map_states$names, properCase)
# Rename the main component of states with multiple parts in the map
map_states$state[map_states$state == "Massachusetts:main"] <- "Massachusetts"
map_states$state[map_states$state == "Michigan:south"] <- "Michigan"
map_states$state[map_states$state == "New York:main"] <- "New York"
map_states$state[map_states$state == "North Carolina:main"] <- "North Carolina"
map_states$state[map_states$state == "Virginia:main"] <- "Virginia"
map_states$state[map_states$state == "Washington:main"] <- "Washington"
# Match with state abbreviations
map_states$state_abb <- state_abb_lookup$state_abb[match(map_states$state, state_abb_lookup$state)]
# Then strip off all the extra location information for states with multiple
# parts (leaving just the state name)
for (i in 1:length(map_states$state)) {
    map_states$state[i] <- unlist(strsplit(map_states$state[i], ":"))[1]
}

# Import key indicators data, matched by state Remember to later match
# information based on state, not state abbreviation (due to how we labelled
# above)
map_states_ins <- map_states
map_states_ins$uninsured_pct_2008 <- insurance_shiny$uninsured_pct_2008[match(map_states$state,
```

```
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2009 <- insurance_shiny$uninsured_pct_2009[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2010 <- insurance_shiny$uninsured_pct_2010[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2011 <- insurance_shiny$uninsured_pct_2011[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2012 <- insurance_shiny$uninsured_pct_2012[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2013 <- insurance_shiny$uninsured_pct_2013[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2014 <- insurance_shiny$uninsured_pct_2014[match(map_states$state,
    insurance_shiny$state)]
map_states_ins$uninsured_pct_2015 <- insurance_shiny$uninsured_pct_2015[match(map_states$state,
    insurance_shiny$state)]

# Get coordinates of state centers
states_centers <- state.center
state.center <- cbind(states_centers, state_abb_lookup)
```

Shiny Code - Actual

```
## THE SHINY APP

# User interface
ui <- fluidPage(titlePanel("Uninsured Rate in the United States from 2008-2015"),
    sidebarLayout(sidebarPanel(sliderInput(inputId = "map_year", label = "Year",
        value = 2008, min = 2008, max = 2015, sep = "", round = TRUE)), mainPanel(leafletOutput("uninsu

# Server
server <- function(input, output) {
    variable_name <- reactive({
        paste("uninsured_pct_", as.character(input$map_year), sep = "")
    })
    # variable <- reactive({map_states_ins[[paste('uninsured_pct_',
    # input$map_year, sep = '')]]})
    output$uninsured_map <- renderLeaflet({
        map <- (leaflet(map_states_ins) %>% setView(lat = 39.8282, lng = -96,
            zoom = 3.5))
        map
    })
    observe({
        classes <- 6
        pal <- colorNumeric(palette = "Oranges", domain = c(0:30), n = classes)
        map <- leafletProxy("uninsured_map") %>% clearShapes() %>% clearControls() %>%
            clearMarkers() %>% addPolygons(data = map_states_ins, fillColor = ~pal(map_states_ins[[varia
            smoothFactor = 0.5, fillOpacity = 0.6, color = "#333333", weight = 1,
            popup = paste("<b>State: </b>", map_states$state, "<br/>", "<b>Uninsured Rate: </b>",
                map_states_ins[[variable_name()]], "%")) %>% addLabelOnlyMarkers(data = filter(state.cen
            state_abb != "AK" & state_abb != "HI"), lng = ~x, lat = ~y, label = ~state_abb,
            labelOptions = labelOptions(textsize = "9px", noHide = T, direction = "top",
                textOnly = T)) %>% addLegend(pal = pal, values = c(0:30), bins = classes,
            position = "bottomright", title = paste("Uninsured", "<br/>", "Rate ",
                "(", as.character(input$map_year), ")", "<br/>", "(%)", sep = ""),
```

```r
             opacity = 0.7)
    })
}

# Run the application
shinyApp(ui = ui, server = server)
```

One of our challenges with Shiny was slightly silly, but we took a while to figure out the scale to use for map. At first, taking direction from the class code, we set the scale / legend of each map to be defined by the variable being plotted (i.e. uninsured rate of a certain year). However, the colors of resulting maps would then show the relative difference between the states, and did not have the effect of showing the decreasing trend across the board.

So we realized we needed the same scale across the maps. We started out choosing the first insurance data year as the anchor but ran into problems of out-of-range values for the other years (see Texas below):

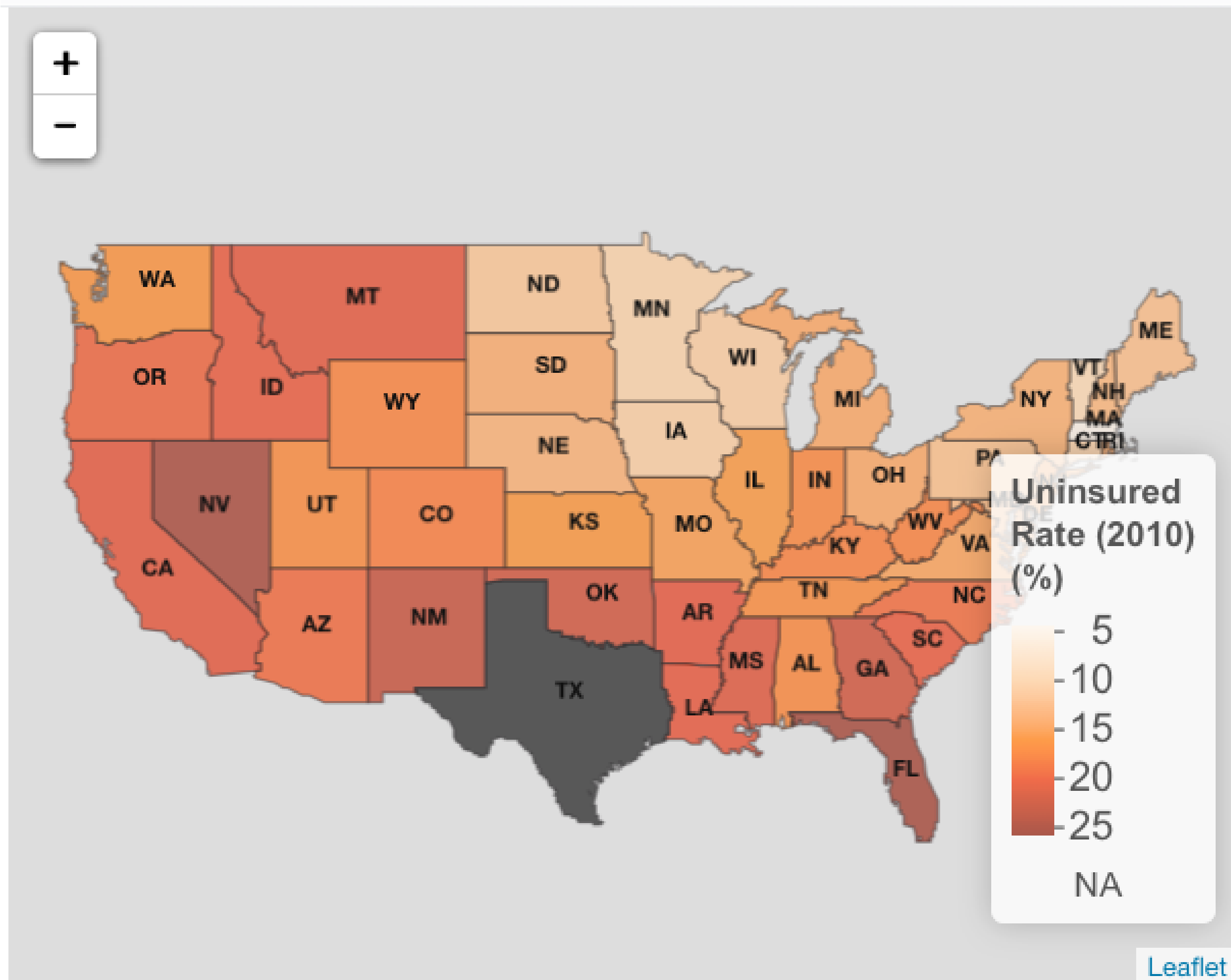


Figure 6:

Then we realized we could just manually input a numerical scale, one that encompassed all the possible values (between 0 and 30 percent). Our final app thus looked like this.

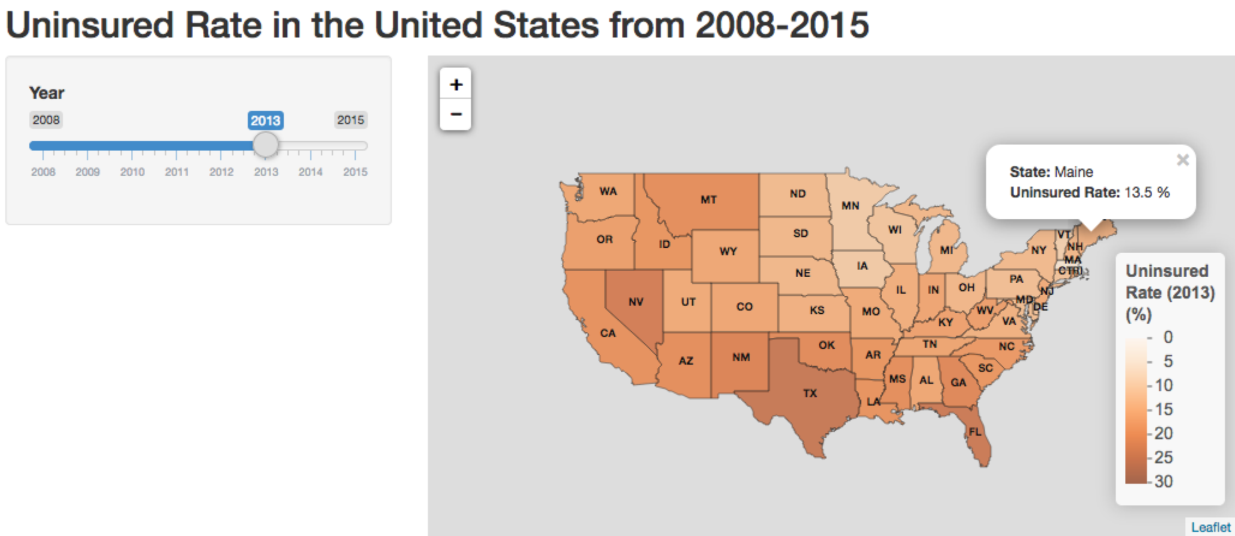


Figure 7:

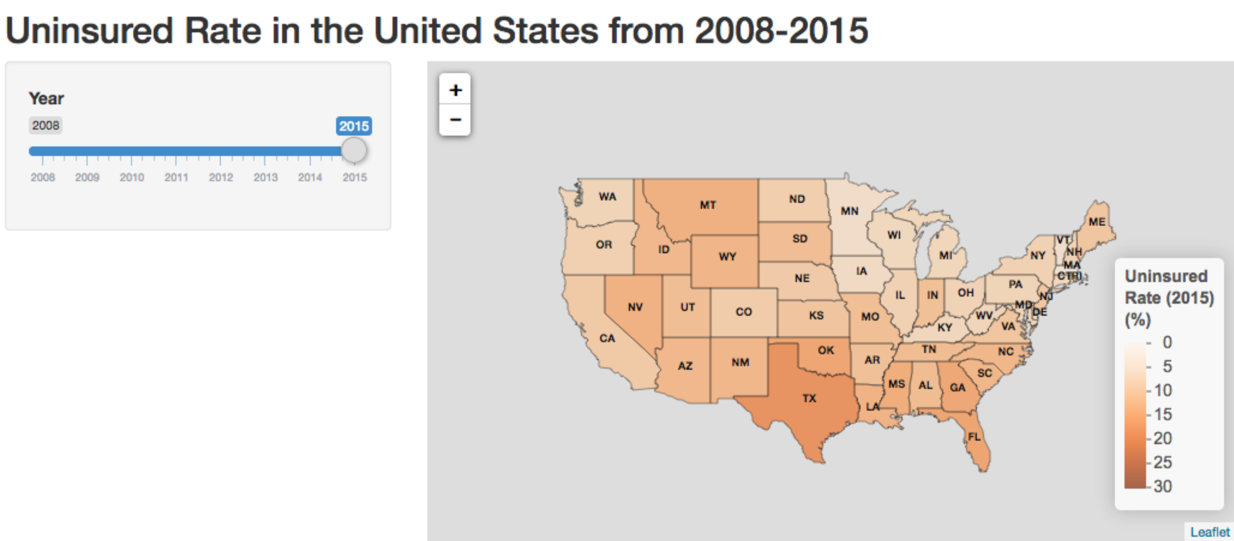


Figure 8:

Link to Shiny App: https://gracekongyx.shinyapps.io/health_insurance_shiny/

States trends

```
## SOME DATA PREPARATION

# Look-up information into insurance and insurance_long, on: Party (2012)
insurance_long$party_2012 <- election$party_2012[match(insurance_long$state,
    election$state)]
insurance$party_2012 <- election$party_2012[match(insurance$state, election$state)]
# BEA region
insurance_long$BEA_region <- demographics1$BEA_region[match(insurance_long$state,
    demographics1$state)]
insurance$BEA_region <- demographics1$BEA_region[match(insurance$state, demographics1$state)]
# Income level
insurance_long$income_level <- demographics1$income_level[match(insurance_long$state,
    demographics1$state)]
insurance$income_level <- demographics1$income_level[match(insurance$state,
    demographics1$state)]

# Also find nation-wide trend for insurance
insurance_long_US <- data.frame(year = c(2008, 2009, 2010, 2011, 2012, 2013,
    2014, 2015), uninsured_pct = c(16.8, 17.5, 18.2, 17.2, 16.9, 16.6, 13.3,
    10.5))
```

Figure 2A

Final figure 2A. This was very close to our original plan. For the interactivity part, we decided to group the legend in terms of region instead of just list all the states, as it would take very long to 'unclick' 49 states to get to the state you want.

This plot is interactive on the website.

```
## PLOT 2A
plot2a <- (ggplot(insurance_long, aes(year, uninsured_pct)) + geom_line(aes(color = BEA_region,
    group = state, label = uninsured_num)) + scale_x_continuous(breaks = c(2008,
    2009, 2010, 2011, 2012, 2013, 2014, 2015)) + scale_y_continuous(breaks = c(5,
    10, 15, 20, 25, 30)) + labs(x = "Year", y = "% Uninsured", color = "Region") +
    geom_line(data = insurance_long_US, lwd = 0.8) + geom_vline(xintercept = 2010,
    linetype = 2, color = "gray30") + geom_vline(xintercept = 2014, linetype = 2,
    color = "gray30") + annotate("text", x = 2009.4, y = 25, label = "Obama \nsigns ACA",
    color = "black", fontface = 2, size = 3) + annotate("text", x = 2013.4,
    y = 24.5, label = "Most ACA \nprovisions \ntake effect", color = "black",
    fontface = 2, size = 3) + annotate("text", x = 2014.2, y = 11, label = "USA \nAverage",
    color = "black", fontface = 2, size = 3) + ggtitle("Change in Percentage Uninsured by State, \nfrom
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0,
    size = 12)))
ggplotly(plot2a, dynamicTicks = FALSE, tooltip = c("state", "year", "uninsured_pct",
    "uninsured_num"))
```

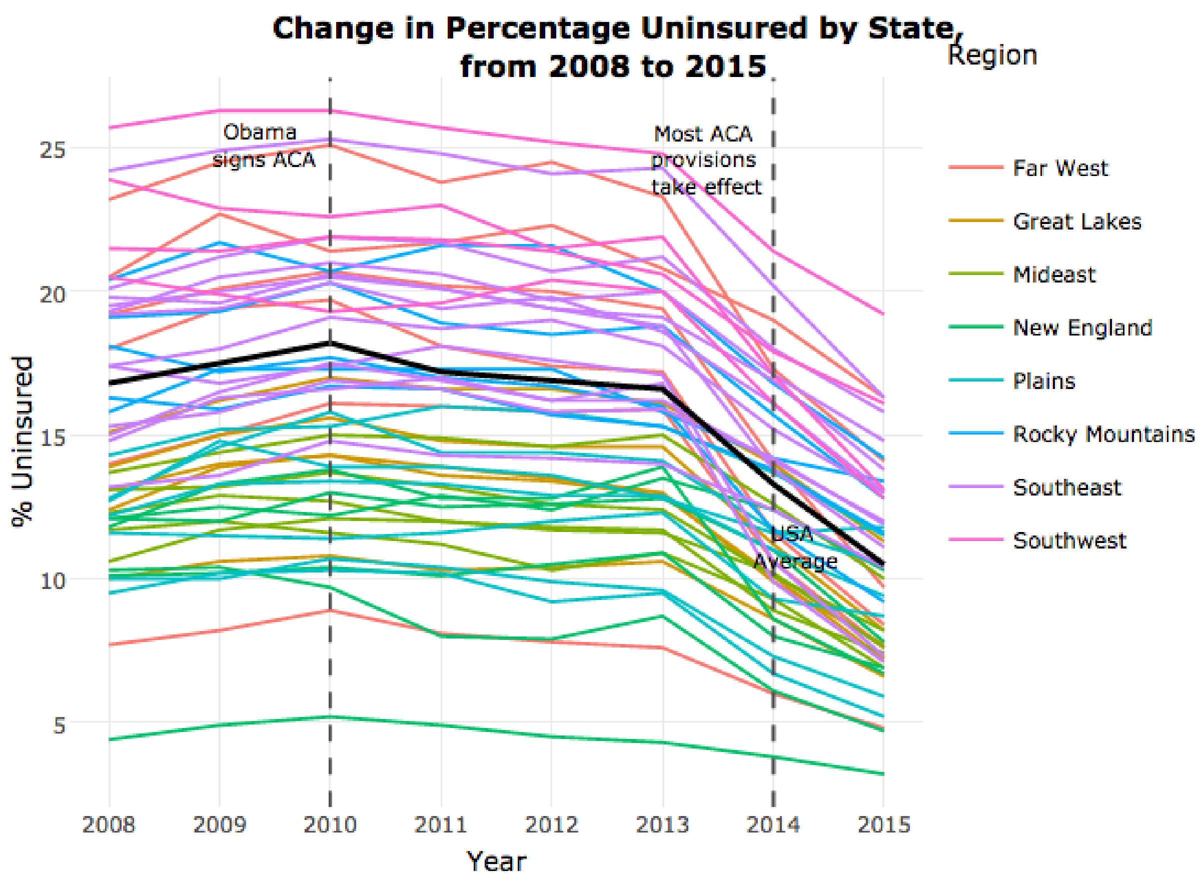
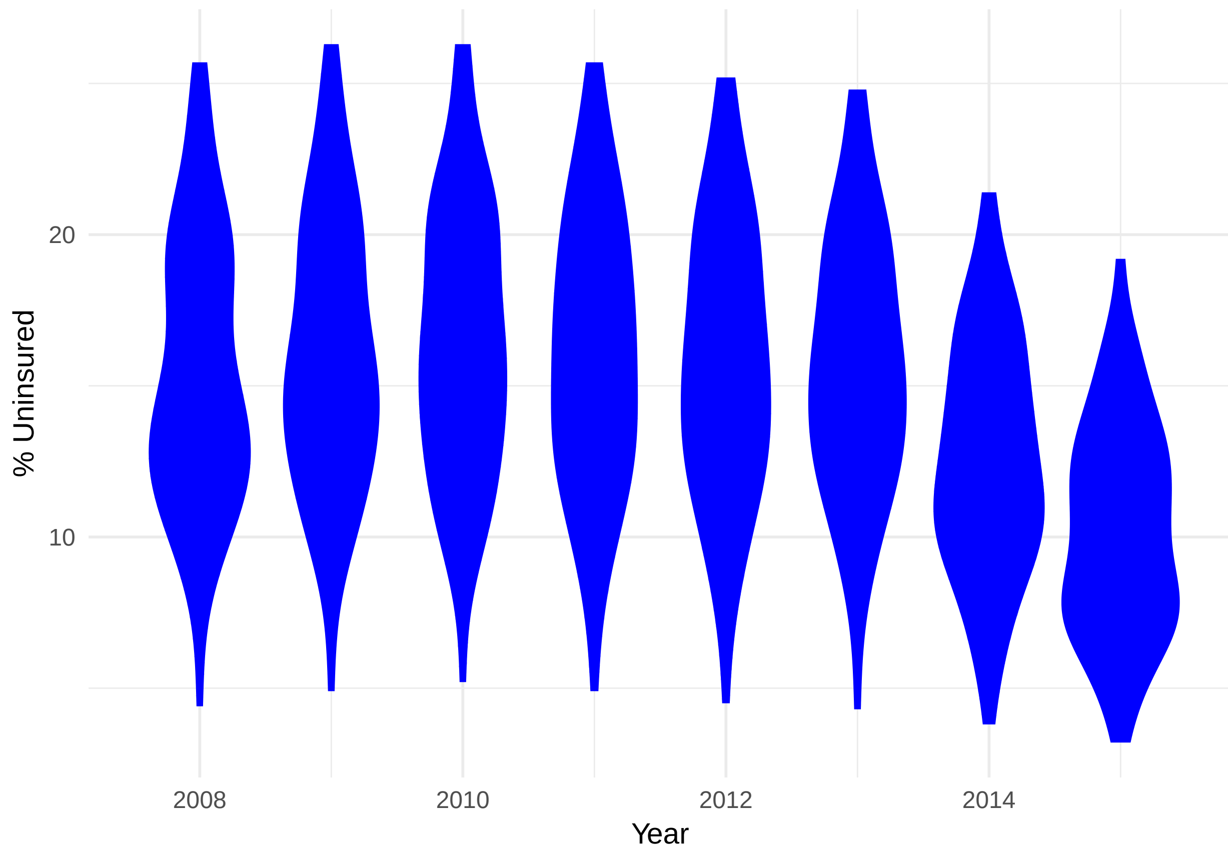


Figure 9:

Figure 2B

We realised that there was less spread in the uninsured percentage data points as the years passed. Therefore, figure 2B is a good contrast to figure 2A. While figure 2A grouped information to each state (through the years), figure 2B aggregates the information by year and directly shows the reduction in spread.

```
## PROCESS PLOT
(ggplot(insurance_long, aes(year, uninsured_pct)) + geom_violin(scale = "area",
    aes(group = year), fill = "blue", color = NA) + labs(x = "Year", y = "% Uninsured",
    color = "Region") + theme_minimal() + theme(plot.title = element_text(face = "bold",
    hjust = 0, size = 12)))
```



Final figure 2B:

```
## PLOT 2B
plot2b <- (ggplot(insurance_long, aes(year, uninsured_pct)) + geom_violin(scale = "area",
    aes(group = year), fill = "dark blue", color = NA) + scale_x_continuous(breaks = c(2008,
    2009, 2010, 2011, 2012, 2013, 2014, 2015)) + scale_y_continuous(breaks = c(5,
    10, 15, 20, 25, 30)) + labs(x = "Year", y = "% Uninsured") + ggtitle("Narrowing of Uninsured Rates A
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.5,
    size = 12)))
plot2b
```

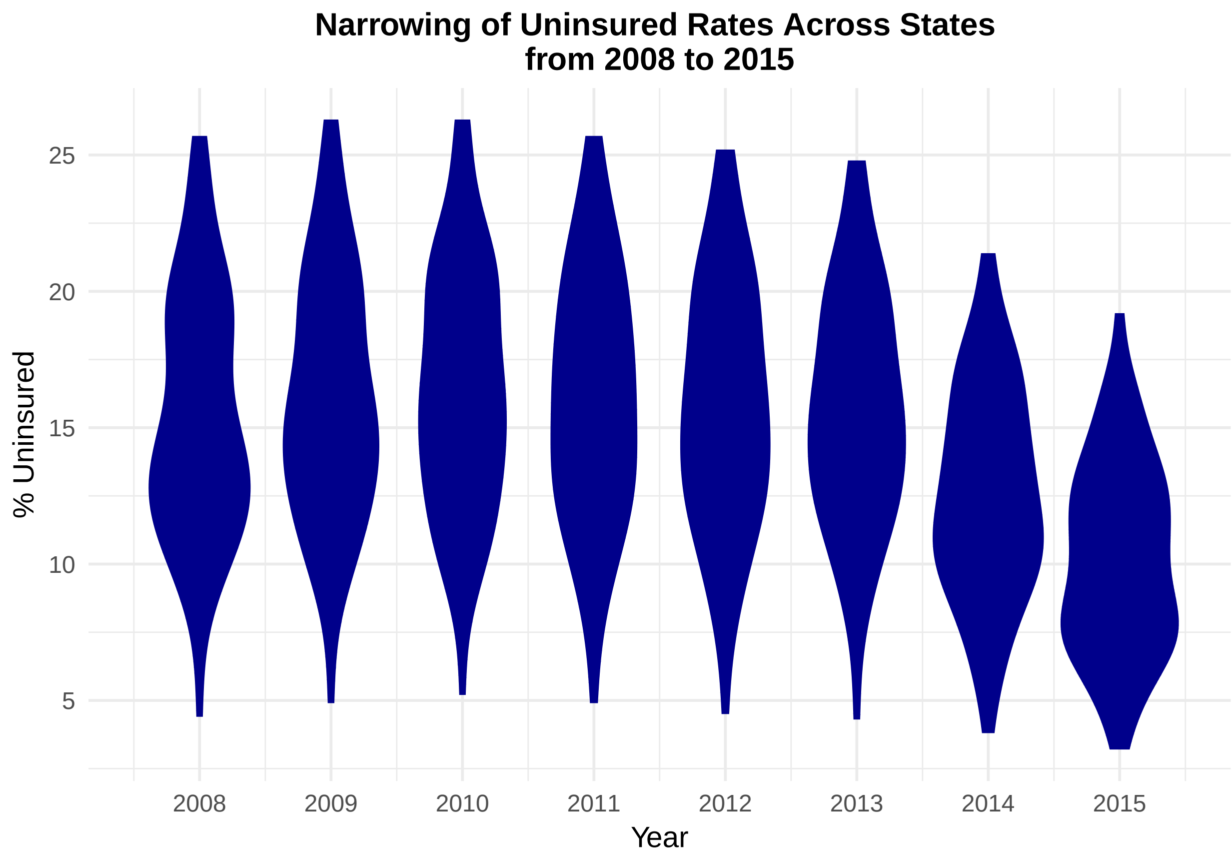


Figure 3

Final figure 3. This was close to our original plan as well.

This plot is interactive on the website.

```
## PLOT 3
plot3 <- (ggplot(insurance_long, aes(year, uninsured_pct)) + geom_line(aes(color = party_2012,
    group = state, label = uninsured_num)) + scale_x_continuous(breaks = c(2008,
    2009, 2010, 2011, 2012, 2013, 2014, 2015)) + scale_y_continuous(breaks = c(5,
    10, 15, 20, 25, 30)) + scale_color_manual(values = c("steel blue", "firebrick1")) +
    labs(x = "Year", y = "% Uninsured", color = "Party \n(2012 Winner)") + geom_line(data = insurance_l
    lwd = 1) + geom_vline(xintercept = 2010, linetype = 2, color = "gray30") +
    geom_vline(xintercept = 2014, linetype = 2, color = "gray30") + annotate("text",
    x = 2009.4, y = 25, label = "Obama \nsigns ACA", color = "black", fontface = 2,
    size = 3) + annotate("text", x = 2013.4, y = 24.5, label = "Most ACA \nprovisions \ntake effect",
    color = "black", fontface = 2, size = 3) + annotate("text", x = 2014.2,
    y = 11, label = "USA \nAverage", color = "black", fontface = 2, size = 3) +
    ggtitle("Change in Percentage Uninsured by Party, \nfrom 2008 to 2015") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0,
    size = 12)))
ggplotly(plot3, dynamicTicks = FALSE, tooltip = c("state", "year", "uninsured_pct",
    "uninsured_num"))
```

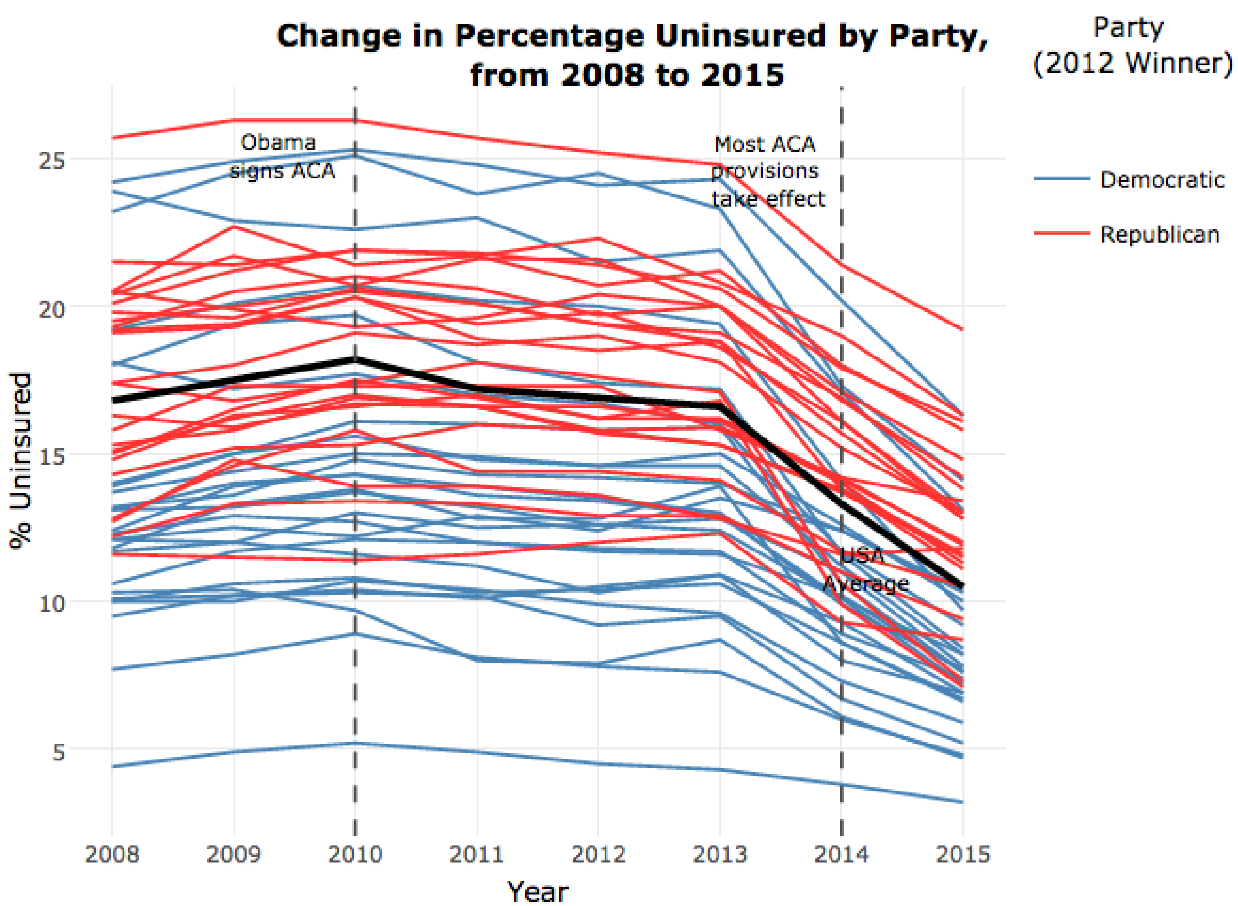
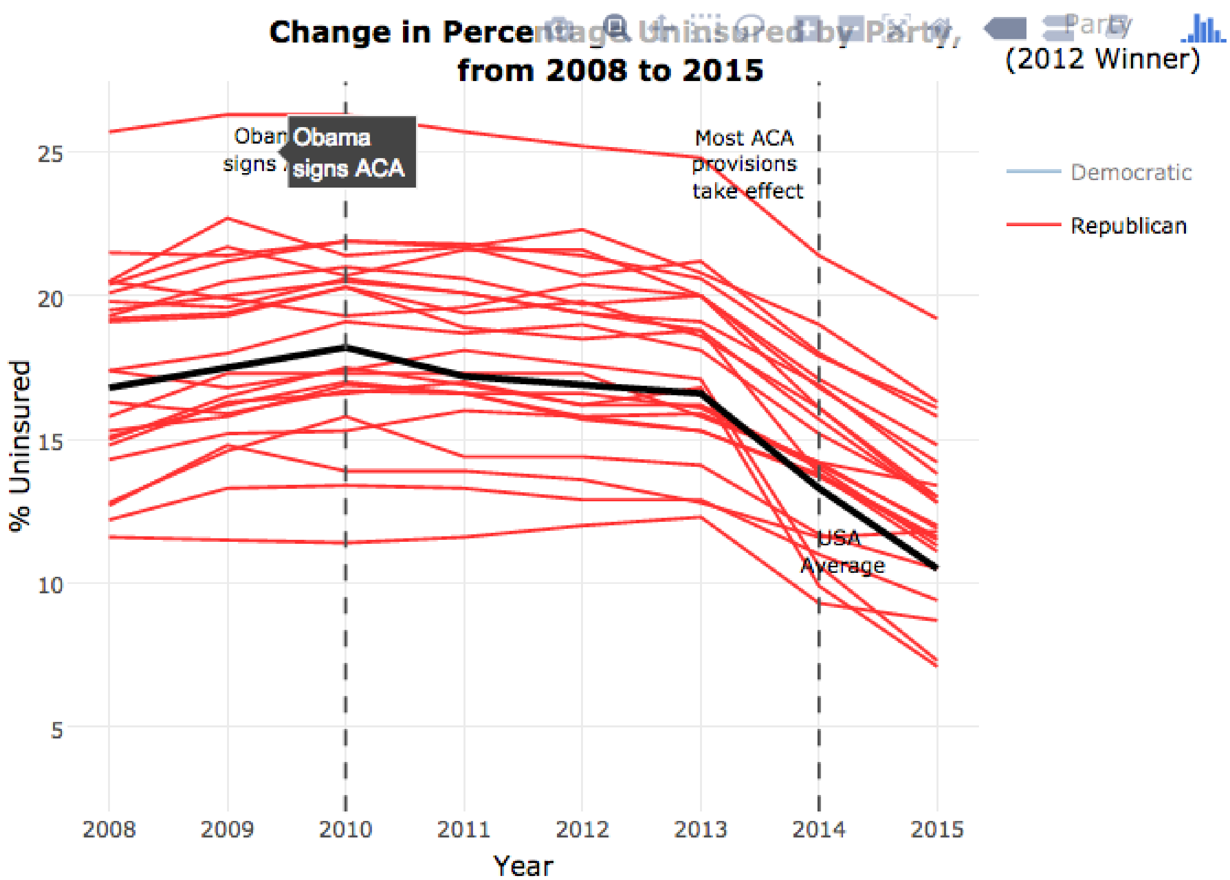


Figure 10:



Change in Percentage Uninsured by Party,
from 2008 to 2015
Party
(2012 Winner)
Democratic
Republican
Obama
signs ACA
Most ACA
provisions
take effect
USA
Average
% Uninsured
25
20
15
10
5
2008
2009
2010
2011
2012
2013
2014
2015
Year

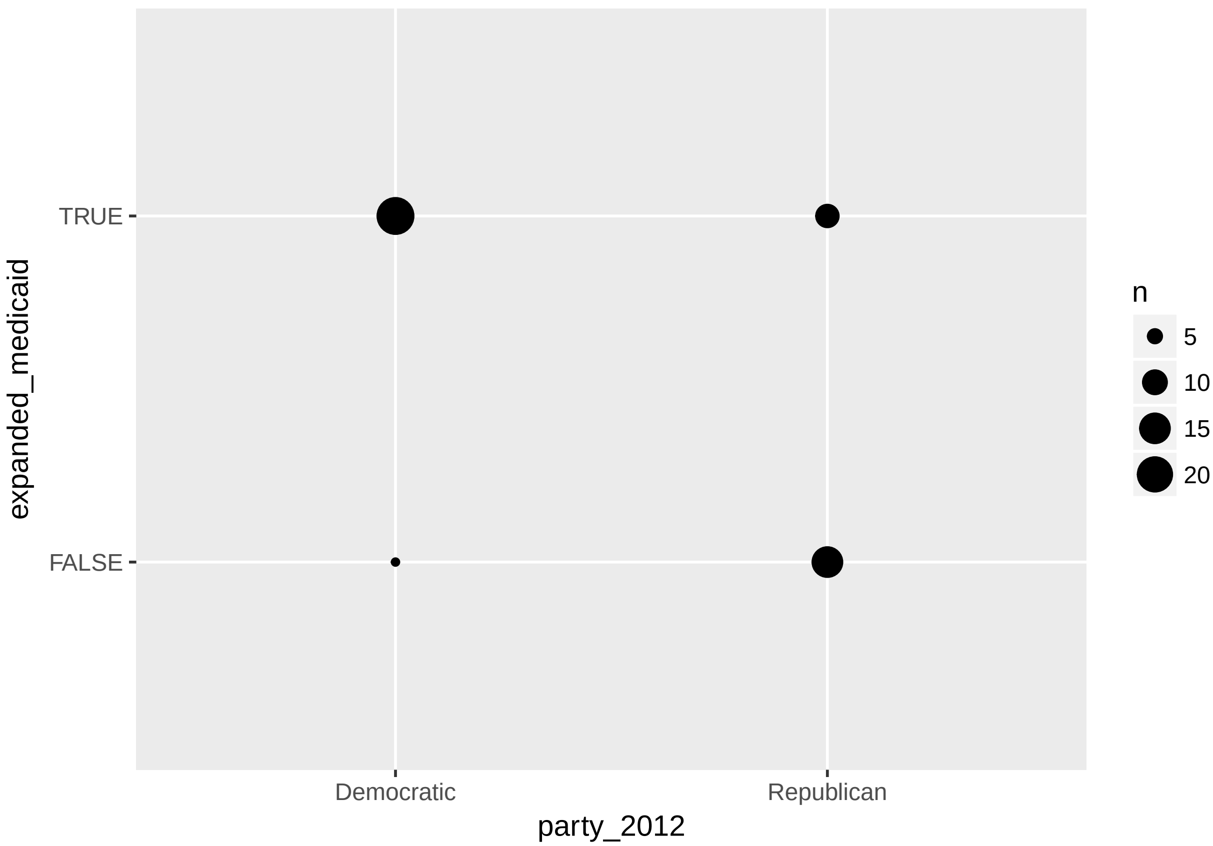
Figure 11:

Figure 4

We decided to add an additional plot correlating medicaid expansion with state party, as we hypothesized that one of the reasons for party differences was their differential adoption of medicaid expansion. Thus this plot was important for confirming our hypothesis.

We tried a geom count at first since it was 2 discrete variables. But we felt that the graph looked too sparese and not aesthetically appealing.

```
# Merge in Medicaid expansion data
insurance$expanded_medicaid <- aca_general$state_has_expanded[match(insurance$state,
    aca_general$state)]

# PROCESS PLOT
(ggplot(insurance) + geom_count(aes(party_2012, expanded_medicaid)))
```



Final figure 4:

```r
# Merge in Medicaid expansion data
insurance$expanded_medicaid <- aca_general$state_has_expanded[match(insurance$state,
    aca_general$state)]

plot4 <- (ggplot(insurance, aes(label = state_abb)) + geom_bar(aes(x = party_2012,
    fill = expanded_medicaid), position = "dodge") + scale_fill_manual(values = brewer.pal(n = 2,
    name = "Dark2")[c(2, 1)]) + labs(x = "Party (2012 Winner)", y = "Number of States",
    fill = "Expanded \nMedicaid") + ggtitle("Medicaid Expansion and Party of State") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.5,
    size = 12)))
plot4
```

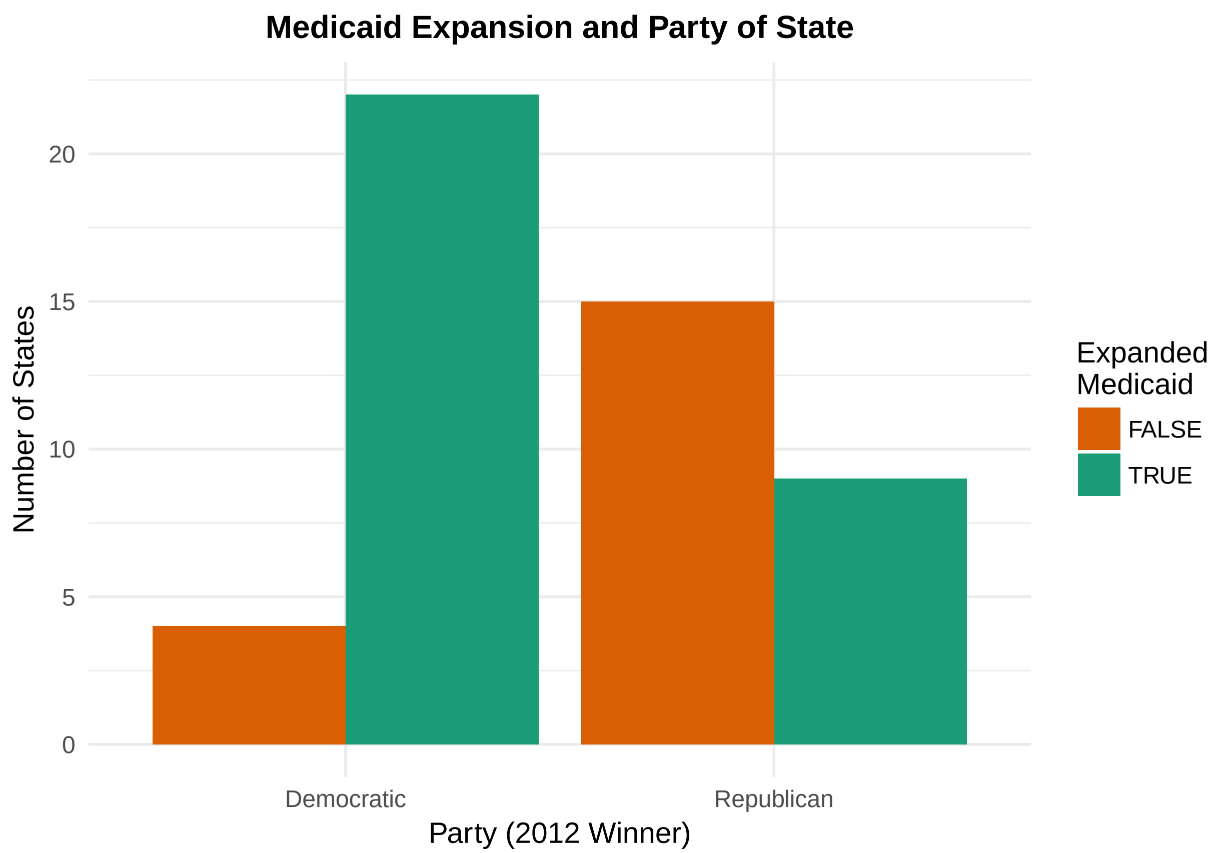


Figure 5

In response to student comments that we should present information comparing the states in the decrease in uninsured rates, and not just on the static rates at a point in time, we created another plot that ranks states on the gains in insured rate from 2010 to 2015. This also helped us gain insight as to whether the republican states are catching up or not (answer: to some extent, as they are in the middle of the group).

Final figure 5. This plot is interactive on the website.

```
insurance$unins_pct_decr_10_15 <- insurance$uninsured_pct_2010 - insurance$uninsured_pct_2015
insurance$unins_num_decr_10_15 <- insurance$uninsured_num_2010 - insurance$uninsured_num_2015

## PLOT 5
plot5 <- (ggplot(insurance, aes(x = reorder(state, unins_pct_decr_10_15), y = unins_pct_decr_10_15,
    fill = party_2012, label = unins_num_decr_10_15)) + geom_col() + coord_flip() +
    labs(x = "", y = "Decrease in % Uninsured", fill = "Party \n(2012 Winner)") +
    scale_fill_manual(values = c("steel blue", "firebrick1")) + ggtitle("Decrease in in Percentage Unins
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0,
    size = 12), axis.text.y = element_text(size = 6.5)))
plot5
```

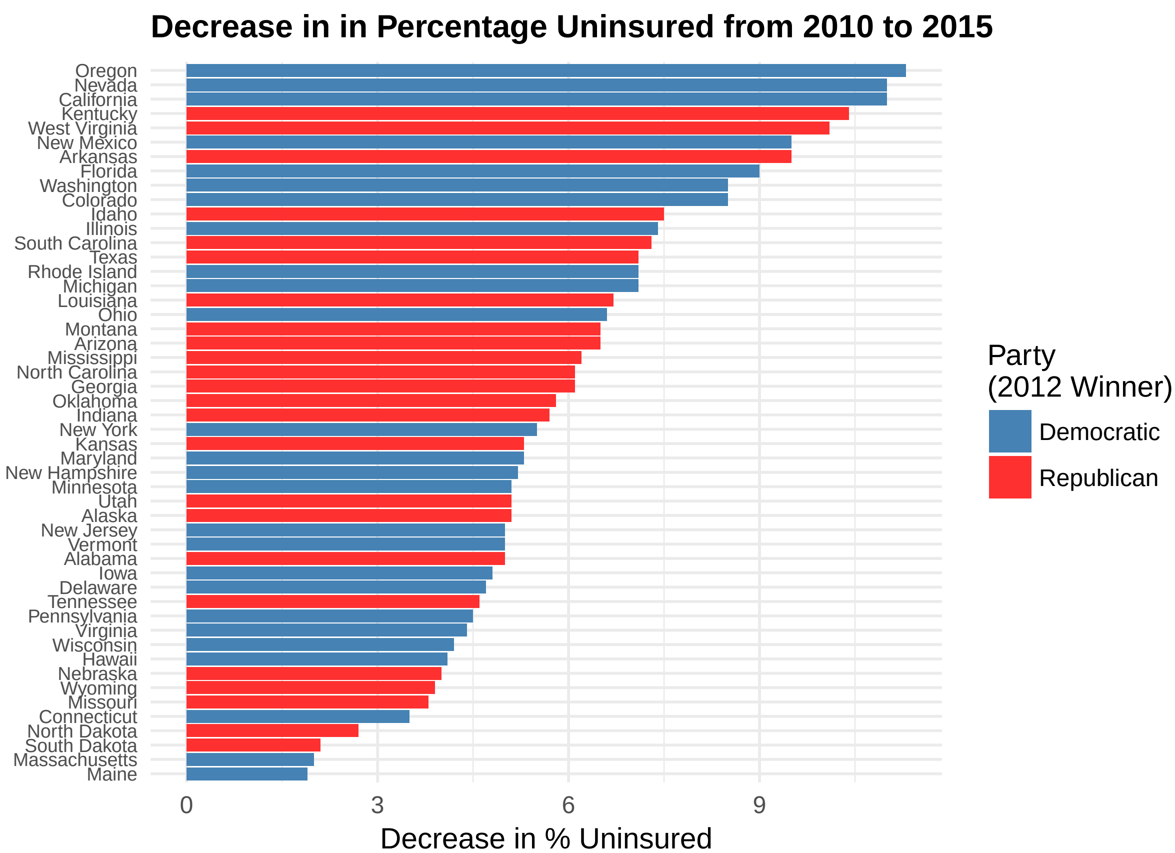


Figure 6A

We wanted to present a snapshot of how states are doing in 2015, compared across different dimensions.

Final figure 6A:

```
## PLOT 6A

# Subset relevant variables
insurance_2010_2015 <- insurance[, c("state", "insured_pct_2010", "unins_pct_decr_10_15",
    "insured_pct_2015", "party_2012", "BEA_region", "income_level")]
names(insurance_2010_2015)[names(insurance_2010_2015) == "unins_pct_decr_10_15"] <- "ins_pct_incr_10_15

# Boxplots by party
plot6a1 <- (ggplot(insurance_2010_2015, aes(x = reorder(party_2012, insured_pct_2015),
    y = insured_pct_2015)) + geom_boxplot(aes(fill = party_2012, label = insured_pct_2015),
    color = "gray30") + coord_flip() + scale_fill_manual(values = c("steel blue",
    "firebrick")) + ggtitle("Democratic vs. Republican States") + theme_minimal() +
    theme(plot.title = element_text(face = "bold", hjust = 0.37, size = 12),
        legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank()))

# Boxplots by BEA region
plot6a2 <- (ggplot(insurance_2010_2015, aes(x = reorder(BEA_region, insured_pct_2015),
    y = insured_pct_2015)) + geom_boxplot(aes(fill = BEA_region, label = insured_pct_2015),
    color = "gray30") + coord_flip() + scale_fill_manual(values = brewer.pal(n = 8,
    name = "Dark2")) + ggtitle("By Bureau of Economic Analysis (BEA) Region") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.24,
    size = 12), legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank(

# Boxplots by income level
plot6a3 <- (ggplot(insurance_2010_2015, aes(x = income_level, y = insured_pct_2015)) +
    geom_boxplot(aes(fill = income_level, label = insured_pct_2015), color = "gray30") +
    coord_flip() + scale_fill_manual(values = brewer.pal(n = 4, name = "Dark2")) +
    labs(x = "", y = "% Insured (2015)") + ggtitle("By State's Income Level") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.39,
    size = 12), legend.position = "none", axis.title.y = element_blank()))

# Combine plots vertically
grid.arrange(plot6a1, plot6a2, plot6a3, nrow = 3, top = "2015 Insurance Rates at a Glance")
```

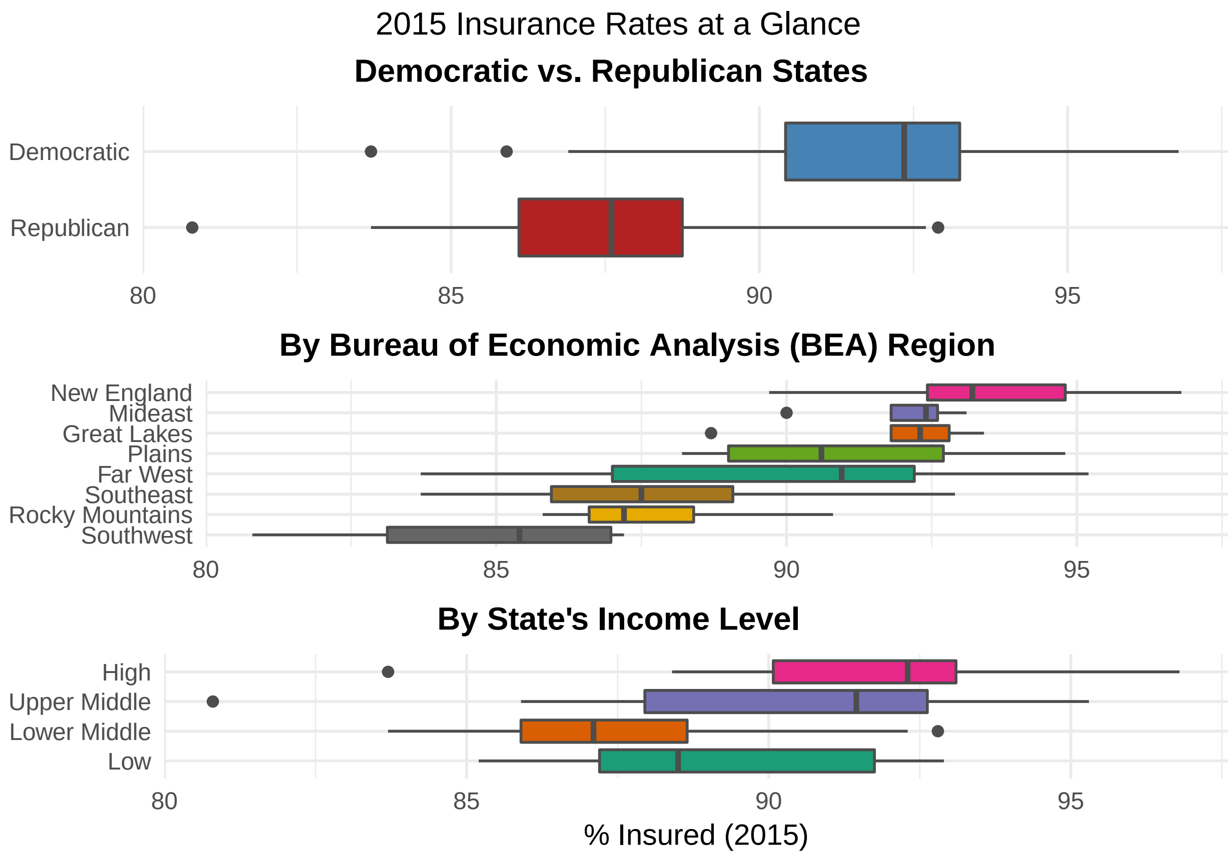
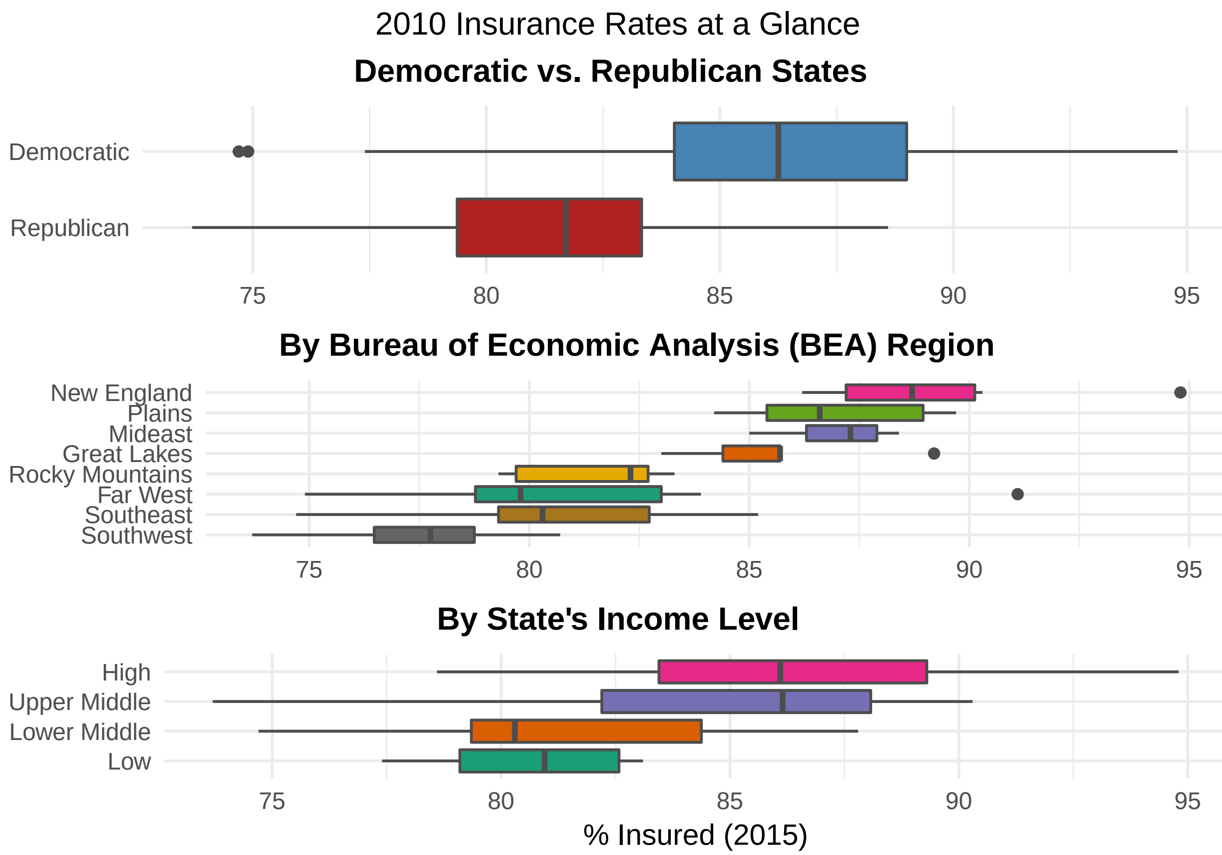

2015 Insurance Rates at a Glance
Democratic vs. Republican States
Democratic
Republican
80
85
90
95
By Bureau of Economic Analysis (BEA) Region
New England
Mideast
Great Lakes
Plains
Far West
Southeast
Rocky Mountains
Southwest
80
85
90
95
By State's Income Level
High
Upper Middle
Lower Middle
Low
80
85
90
95
% Insured (2015)

Figure 6B

In response to student comments to show how states did both before and after the ACA was rolled out, we initially wanted to put in a similar plot for the 2010 (pre-ACA) rates, as below.

```
# PROCESS PLOT

# Boxplots by party
plot3a1 <- (ggplot(insurance_2010_2015, aes(x = reorder(party_2012, insured_pct_2010),
    y = insured_pct_2010)) + geom_boxplot(aes(fill = party_2012, label = insured_pct_2010),
    color = "gray30") + coord_flip() + scale_fill_manual(values = c("steel blue",
    "firebrick")) + ggtitle("Democratic vs. Republican States") + theme_minimal() +
    theme(plot.title = element_text(face = "bold", hjust = 0.37, size = 12),
        legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank()))

# Boxplots by BEA region
plot3b1 <- (ggplot(insurance_2010_2015, aes(x = reorder(BEA_region, insured_pct_2010),
    y = insured_pct_2010)) + geom_boxplot(aes(fill = BEA_region, label = insured_pct_2010),
    color = "gray30") + coord_flip() + scale_fill_manual(values = brewer.pal(n = 8,
    name = "Dark2")) + ggtitle("By Bureau of Economic Analysis (BEA) Region") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.24,
    size = 12), legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank(

# Boxplots by income level
plot3c1 <- (ggplot(insurance_2010_2015, aes(x = income_level, y = insured_pct_2010)) +
    geom_boxplot(aes(fill = income_level, label = insured_pct_2010), color = "gray30") +
    coord_flip() + scale_fill_manual(values = brewer.pal(n = 4, name = "Dark2")) +
    labs(x = "", y = "% Insured (2015)") + ggtitle("By State's Income Level") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.39,
    size = 12), legend.position = "none", axis.title.y = element_blank()))

# Combine plots vertically
grid.arrange(plot3a1, plot3b1, plot3c1, nrow = 3, top = "2010 Insurance Rates at a Glance")
```


2010 Insurance Rates at a Glance
Democratic vs. Republican States
Democratic
Republican
75
80
85
90
95
By Bureau of Economic Analysis (BEA) Region
New England
Plains
Mideast
Great Lakes
Rocky Mountains
Far West
Southeast
Southwest
75
80
85
90
95
By State's Income Level
High
Upper Middle
Lower Middle
Low
75
80
85
90
95
% Insured (2015)

However, we though this didn't really give us much new information, the rankings were roughly the same. Yet, when tried a new plot that compared *change in insured rate from 2010 to 2015* instead, we found the interesting result that the rankings were often flipped, indicating that there was some 'catch-up' (but not enough).

Final figure 6B:

```
## PLOT 6B

# Boxplots by party
plot6b1 <- (ggplot(insurance_2010_2015, aes(x = reorder(party_2012, ins_pct_incr_10_15),
    y = ins_pct_incr_10_15)) + geom_boxplot(aes(fill = party_2012, label = ins_pct_incr_10_15),
    color = "gray30") + coord_flip() + scale_fill_manual(values = c("steel blue",
    "firebrick")) + ggtitle("Democratic vs. Republican States") + theme_minimal() +
    theme(plot.title = element_text(face = "bold", hjust = 0.37, size = 12),
        legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank()))

# Boxplots by BEA region
plot6b2 <- (ggplot(insurance_2010_2015, aes(x = reorder(BEA_region, ins_pct_incr_10_15),
    y = ins_pct_incr_10_15)) + geom_boxplot(aes(fill = BEA_region, label = ins_pct_incr_10_15),
    color = "gray30") + coord_flip() + scale_fill_manual(values = brewer.pal(n = 8,
    name = "Dark2")) + ggtitle("By Bureau of Economic Analysis (BEA) Region") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.24,
    size = 12), legend.position = "none", axis.title.x = element_blank(), axis.title.y = element_blank(

# Boxplots by income level
plot6b3 <- (ggplot(insurance_2010_2015, aes(x = income_level, y = ins_pct_incr_10_15)) +
    geom_boxplot(aes(fill = income_level, label = ins_pct_incr_10_15), color = "gray30") +
    coord_flip() + scale_fill_manual(values = brewer.pal(n = 4, name = "Dark2")) +
    labs(x = "", y = "Increase in % Insured") + ggtitle("By State's Income Level") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.39,
    size = 12), legend.position = "none", axis.title.y = element_blank()))

# Combine plots vertically
grid.arrange(plot6b1, plot6b2, plot6b3, nrow = 3, top = "Increase in Insurance Rates (2010-2015) at a Gl
```

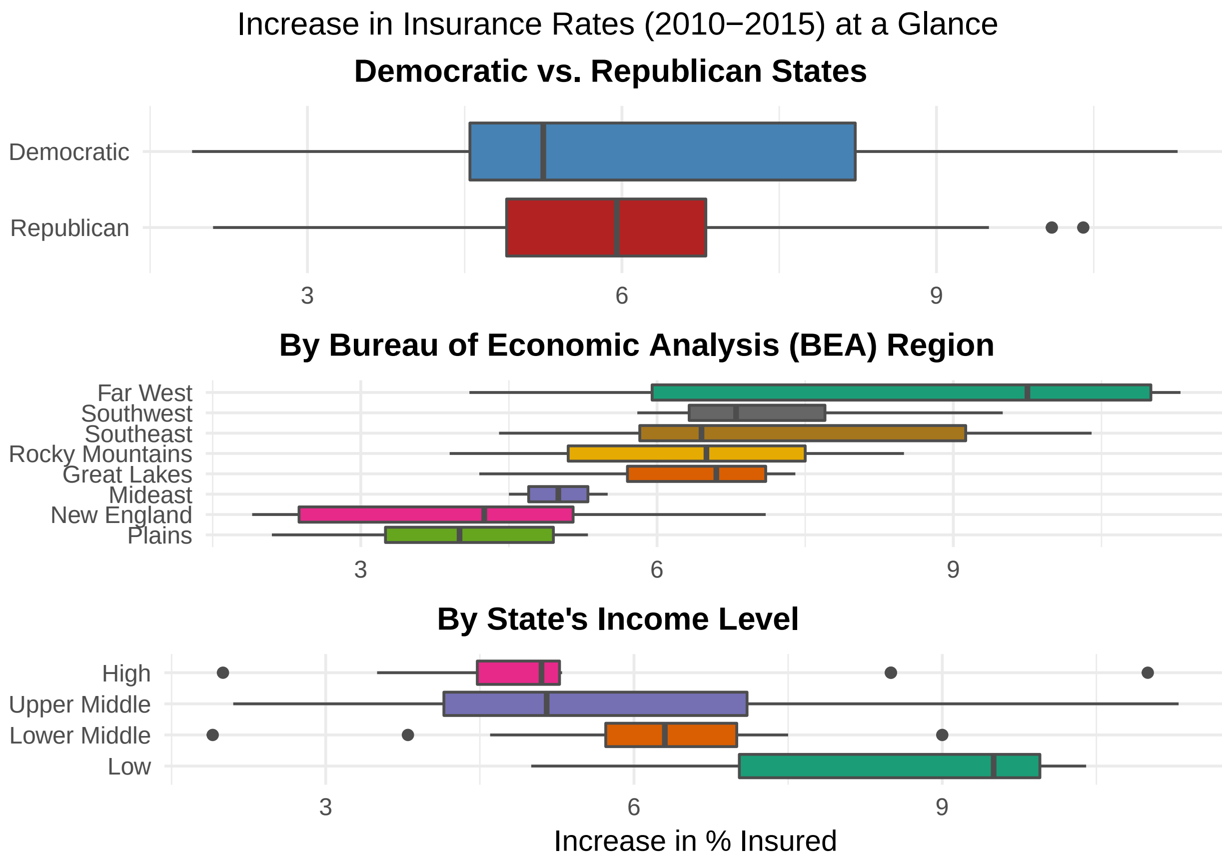

Increase in Insurance Rates (2010−2015) at a Glance
Democratic vs. Republican States
Democratic
Republican
3
6
9
By Bureau of Economic Analysis (BEA) Region
Far West
Southwest
Southeast
Rocky Mountains
Great Lakes
Mideast
New England
Plains
3
6
9
By State's Income Level
High
Upper Middle
Lower Middle
Low
3
6
9
Increase in % Insured

Figure 7

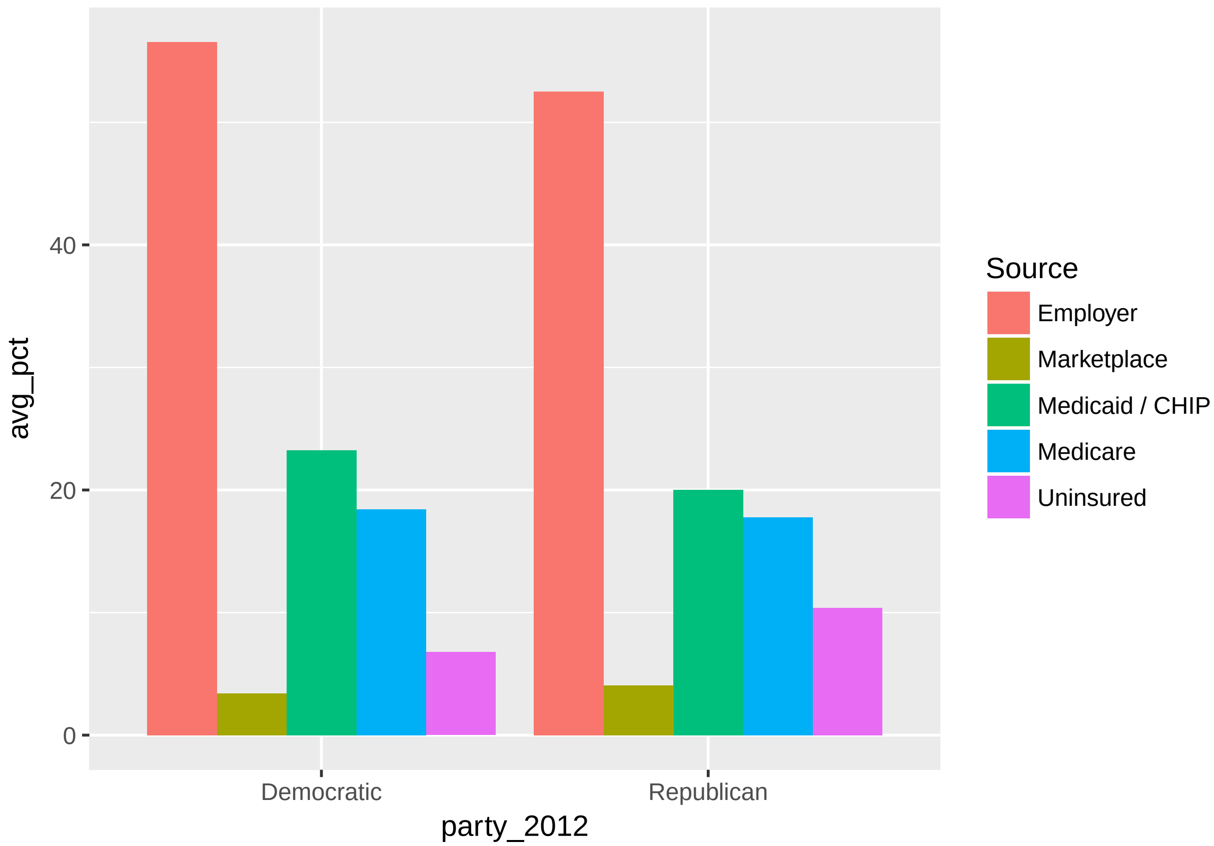
```
## SOME DATA RESHAPING

# Reshape aca_general (long), keeping the interesting quantities Import
# uninsured number (2015)
aca_general$unins_all_15 <- insurance$uninsured_num_2015[match(aca_general$state,
    insurance$state)]
# Keep only select variables
aca_general_long <- aca_general[, c("state", "state_abb", "cov_emp", "cov_mkt_plan_16",
    "medicaid_enroll_16", "medicare_enroll_16", "unins_all_15")]
# The actual reshaping
aca_general_long <- reshape(aca_general_long, varying = c("cov_emp", "cov_mkt_plan_16",
    "medicaid_enroll_16", "medicare_enroll_16", "unins_all_15"), v.names = "Number",
    timevar = "Source", times = c("Employer", "Marketplace", "Medicaid / CHIP",
        "Medicare", "Uninsured"), new.row.names = 1:1000, direction = "long")

# Look-up information into aca_general_long, on: Party (2012)
aca_general_long$party_2012 <- election$party_2012[match(aca_general_long$state,
    election$state)]
# BEA region
aca_general_long$BEA_region <- demographics1$BEA_region[match(aca_general_long$state,
    demographics1$state)]
# Income level
aca_general_long$income_level <- demographics1$income_level[match(aca_general_long$state,
    demographics1$state)]
# Population
aca_general_long$population_2015 <- demographics2$population_2015[match(aca_general_long$state,
    demographics2$state)]

# Calculate percentage from number and population
aca_general_long$percentage <- aca_general_long$Number/aca_general_long$population_2015 *
    100
```

As we had planned from the start, we wanted to do a graph on the different sources where people got their insurance. And we wanted to do some sort of correlation with state characteristics, including party, BEA region and income level.

Therefore, we made exploratory plots for each of these dimensions.

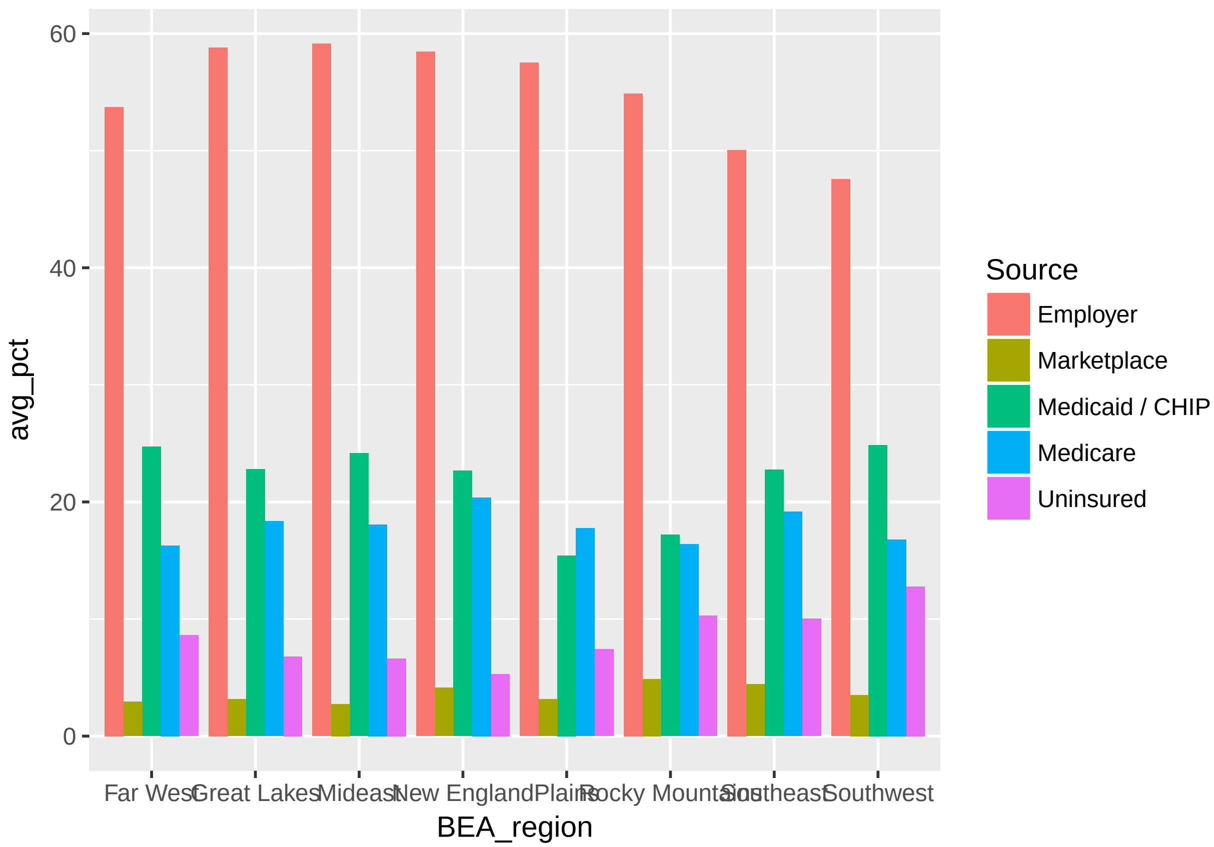
By party

```
## PROCESS PLOT
aca_general_long_by_party <- ddply(aca_general_long, c("party_2012", "Source"),
    summarise, avg_pct = mean(percentage))

(ggplot(aca_general_long_by_party, aes(party_2012, avg_pct, fill = Source)) +
    geom_col(position = "dodge"))
```



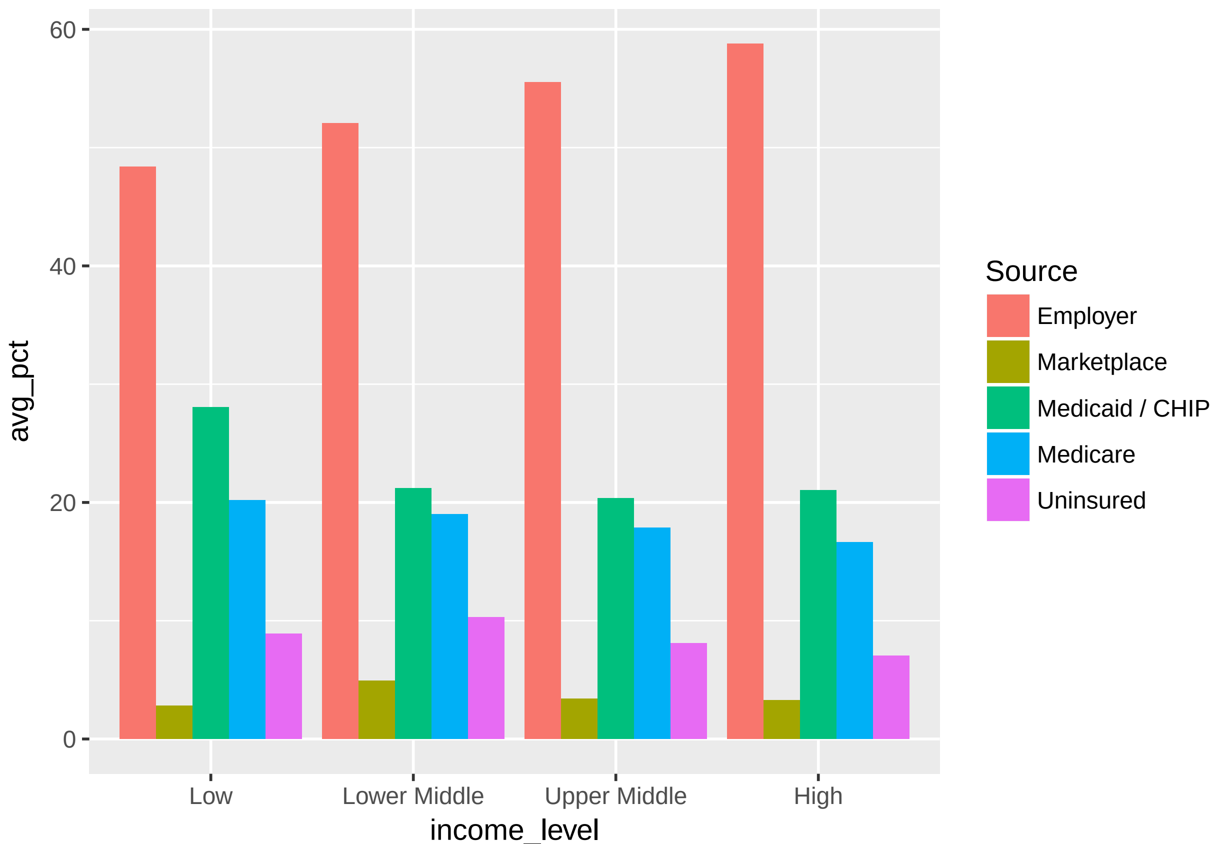
By region

```
## PROCESS PLOT
aca_general_long_by_region <- ddply(aca_general_long, c("BEA_region", "Source"),
    summarise, avg_pct = mean(percentage))

(ggplot(aca_general_long_by_region, aes(BEA_region, avg_pct, fill = Source)) +
    geom_col(position = "dodge"))
```



By income level

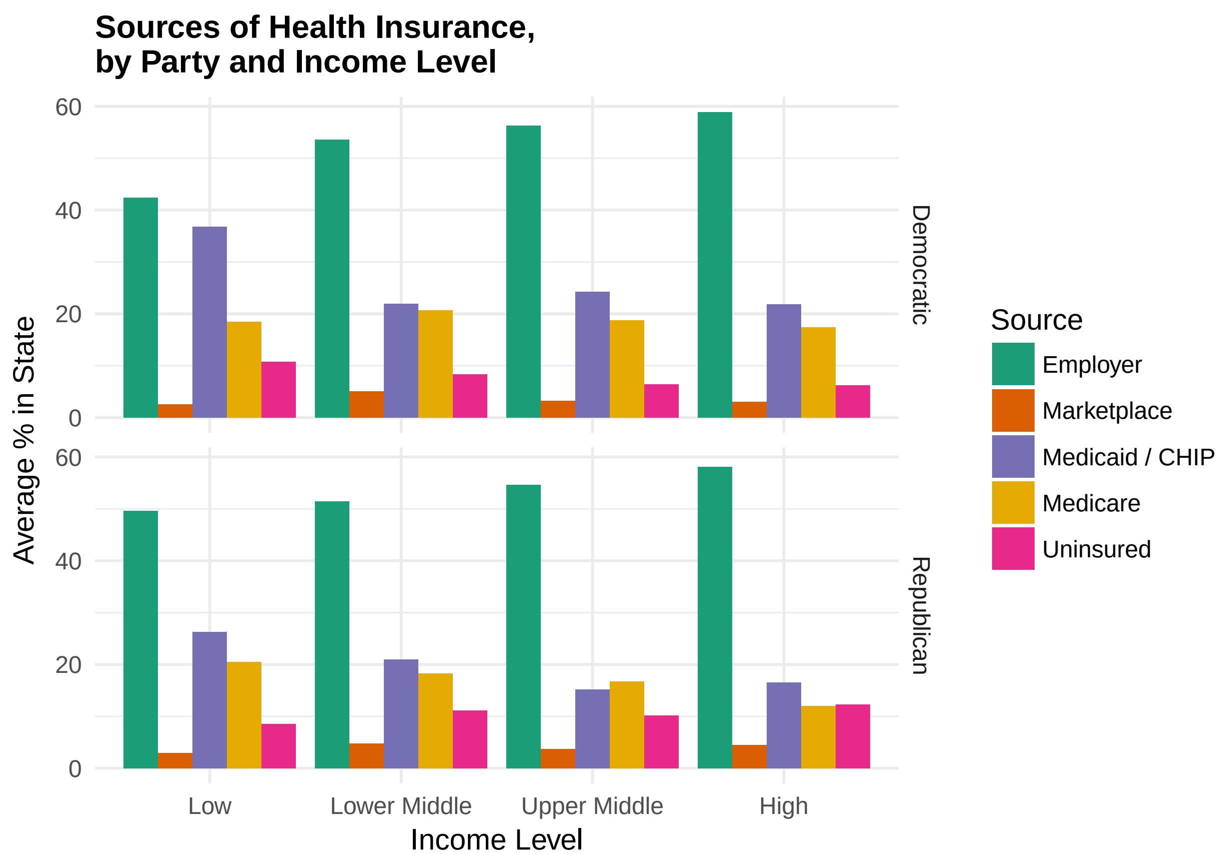
```
## PROCESS PLOT
aca_general_long_by_income <- ddply(aca_general_long, c("income_level", "Source"),
    summarise, avg_pct = mean(percentage))

(ggplot(aca_general_long_by_income, aes(income_level, avg_pct, fill = Source)) +
    geom_col(position = "dodge"))
```



We felt that the trends in the party and the income level plots were most significant adn meaningful. So we dropped regional differences and presented trends by party and income level in the final faceted plot.

Final figure 7. This plot is interactive on the website.

```
## PLOT 7

aca_general_long_by_party_income <- ddply(aca_general_long, c("party_2012",
    "income_level", "Source"), summarise, avg_pct = mean(percentage))
aca_general_long_by_party_income$avg_pct <- round(aca_general_long_by_party_income$avg_pct,
    digits = 1)

plot7 <- (ggplot(aca_general_long_by_party_income, aes(income_level, avg_pct,
    fill = Source)) + geom_col(position = "dodge") + facet_grid(party_2012 ~
    .) + labs(x = "Income Level", y = "Average % in State", color = "Source of \nInsurance") +
    scale_fill_manual(values = brewer.pal(n = 6, name = "Dark2")[c(1:3, 6, 4)]) +
    ggtitle("Sources of Health Insurance, \nby Party and Income Level") + theme_minimal() +
    theme(plot.title = element_text(face = "bold", hjust = 0, size = 12)))
plot7
```



States outcomes - health and financial

To prepare for comparing the states across the different health and financial outcomes, we create a dataframe of (10+) key indicators in these two dimensions. This required going through our 100+ outcome variables and selecting those that were relevant to our question, while still keeping enough breadth.

For example, we asked what represented a good health outcome? We wanted to capture aspects of mortality itself, access to health care (routine), and access to preventative healthcare like cancer screenings and vaccines.

And how about what represented a good financial outcome? That was trickier as there were many trade offs in terms of who bore the most financial burden in each state. Thus, we tried to include information on all these financial metrics, like OOP (private, incidental cost), premium (private, pooled cost), and federal spending (public cost).

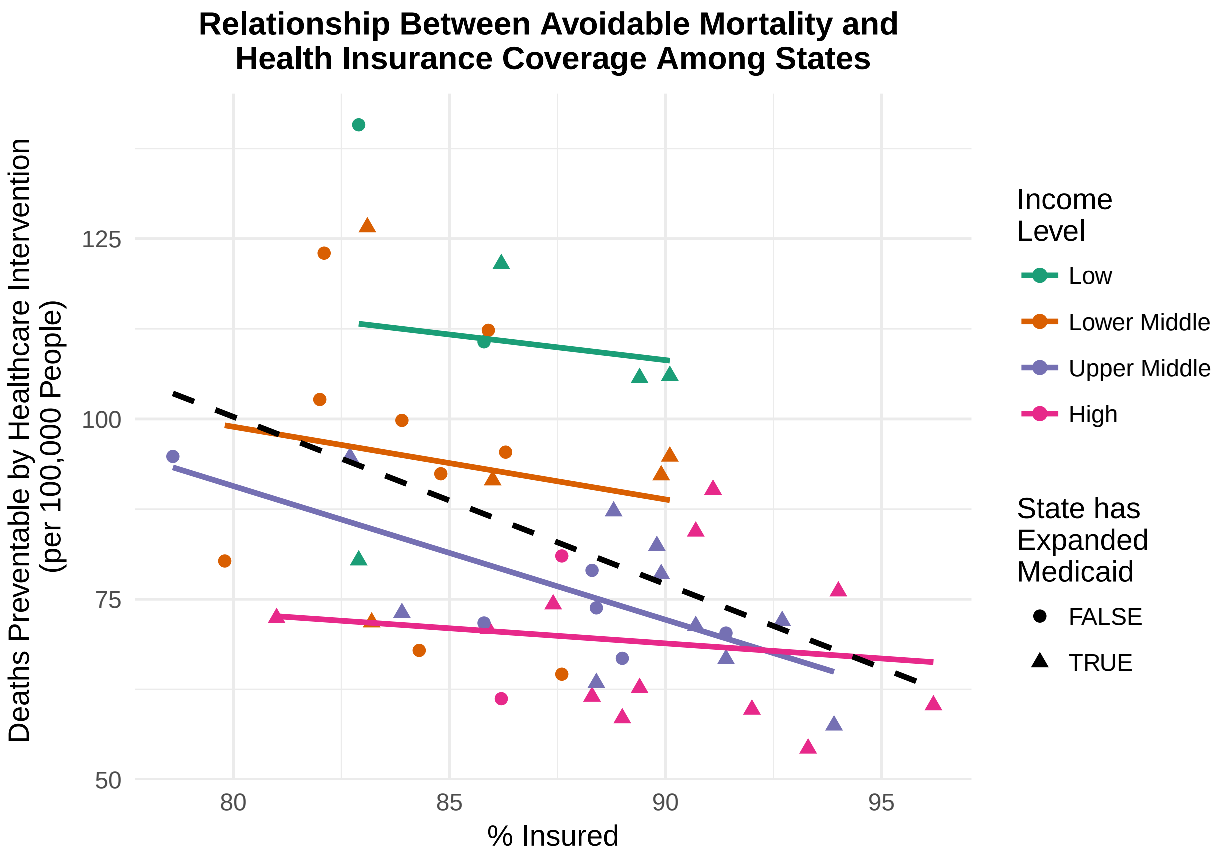
```
## PREPARATION FOR FURTHER PLOTS (INCLUDING DATA TABLE)

# Want to combine information on insurance coverage with key information on
# health outcomes/access and affordability

key_indicators <- state_abb_lookup

## Merge in information to the data table on the following

# Demographic Information Income level
key_indicators$income_level <- demographics1$income_level[match(key_indicators$state_abb,
    demographics1$state_abb)]

# Insurance Coverage Percentage insured (2015)
key_indicators$insured_pct_2015 <- insurance$insured_pct_2015[match(key_indicators$state_abb,
    insurance$state_abb)]
# Percentage insured (2014)
key_indicators$insured_pct_2014 <- insurance$insured_pct_2014[match(key_indicators$state_abb,
    insurance$state_abb)]
# Indicator: Whether state has expanded medicare
key_indicators$state_has_expanded <- aca_general$state_has_expanded[match(key_indicators$state_abb,
    aca_general$state_abb)]

# Health outcomes and access Mortality amenable to health care, deaths per
# 10000 population
key_indicators$deaths_amenable_2014 <- health_ind$h.deaths_amenable.2014[match(key_indicators$state_abb
    health_ind$state_abb)]
# Years of Potential Life Lost before 75
key_indicators$years_life_lost <- health_ind$h.yrs_lost_potential_life_before75.2014[match(key_indicato
    health_ind$state_abb)]
# Adults with a usual source of care (%)
key_indicators$usual_care_2015 <- health_ind$q.with_usual_care_adult.2015[match(key_indicators$state_ab
    health_ind$state_abb)]
# Adults with age/gender-appropriate cancer screenings (%)
key_indicators$with_cancer_screening <- health_ind$q.with_cancer_screening_adult.2014[match(key_indicat
    health_ind$state_abb)]
# Adults with age-appropriate vaccines (%)
key_indicators$with_vaccines <- health_ind$q.with_vaccines_adult.2015[match(key_indicators$state_abb,
    health_ind$state_abb)]

# Affordability and cost-efficiency Individuals under age 65 with high OOP
# medical costs relative to annual household income
```

```
key_indicators$high_OOP_relative <- health_ind$a.high_OOP_relative_under65.2015[match(key_indicators$sta
    health_ind$state_abb)]
# Average annual growth in family premiums for employer coverage (between
# 2010 and 2015)
key_indicators$premium_ann_growth_10_15 <- aca_general$premium_emp_avg_growth_pct_10to15[match(key_indic
    aca_general$state_abb)]
# Marketplace consumers who could select a plan for less than $100
key_indicators$IM_plan_under_100 <- aca_general$cov_mkt_under100D_pct[match(key_indicators$state_abb,
    aca_general$state_abb)]
# Net increase in federal spending (millions)
key_indicators$incr_fed_spending_mil <- aca_general$fed_spending_net_incr_inMil[match(key_indicators$sta
    aca_general$state_abb)]
```

Figure 8

We tried correlating mortality measures with insurance rates, as planned in our very early discussions.

```
## PROCESS PLOT
plot8_old <- (ggplot(key_indicators, aes(insured_pct_2014, deaths_amenable_2014)) +
    geom_point(aes(color = income_level, shape = state_has_expanded), size = 2) +
    geom_smooth(aes(color = income_level, group = income_level), method = "lm",
        lwd = 1, se = FALSE) + geom_smooth(color = "black", method = "lm", linetype = 2,
    lwd = 1, se = FALSE) + labs(x = "% Insured", y = "Deaths Preventable by Healthcare Intervention \n(p
    color = "Income \nLevel", shape = "State has \nExpanded \nMedicaid") + scale_color_manual(values = 
    name = "Dark2")) + ggtitle("Relationship Between Avoidable Mortality and \nHealth Insurance Coverage
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.5,
    size = 12)))
plot8_old
```



Ideally, we wanted to make the plot interactive so that viewers could find out which point represented which state. However, ggplotly messed up our legend so we abandoned that thought.

```
## PROCESS PLOT
ggplotly(plot8_old)
```

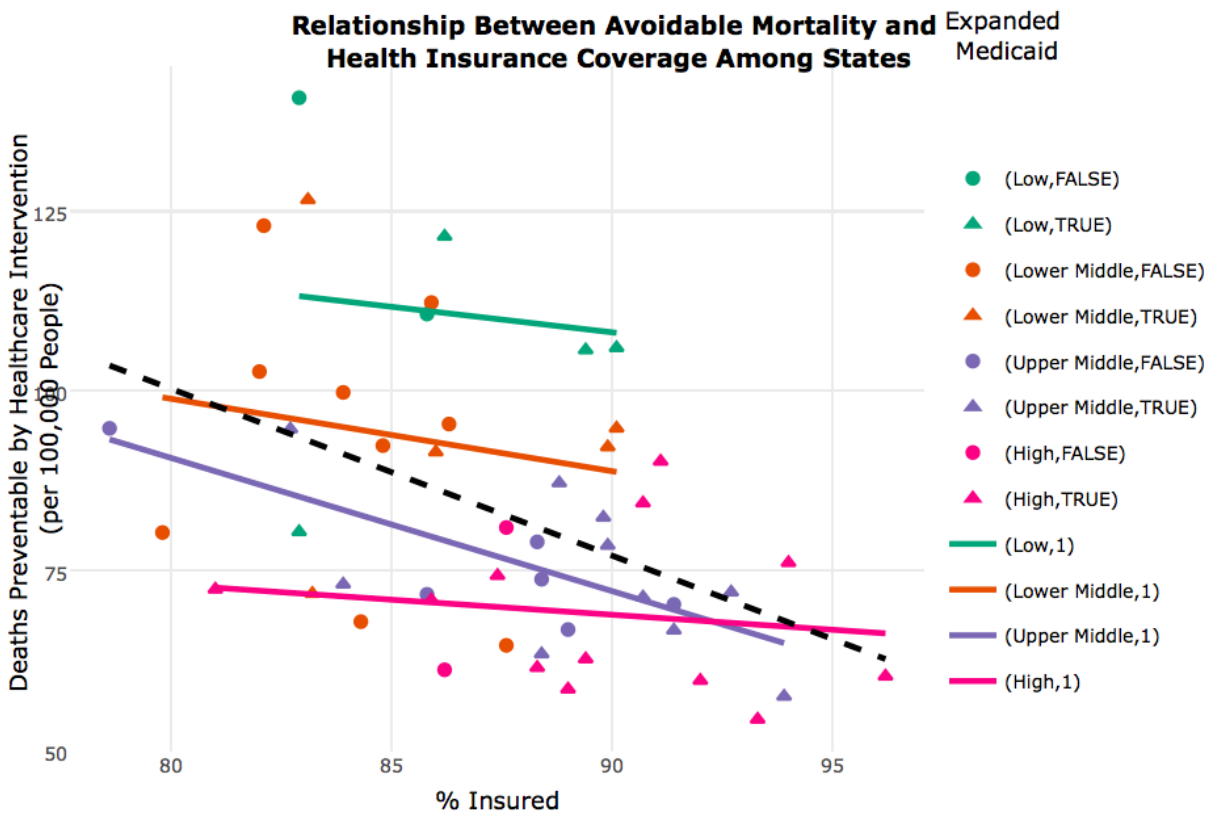
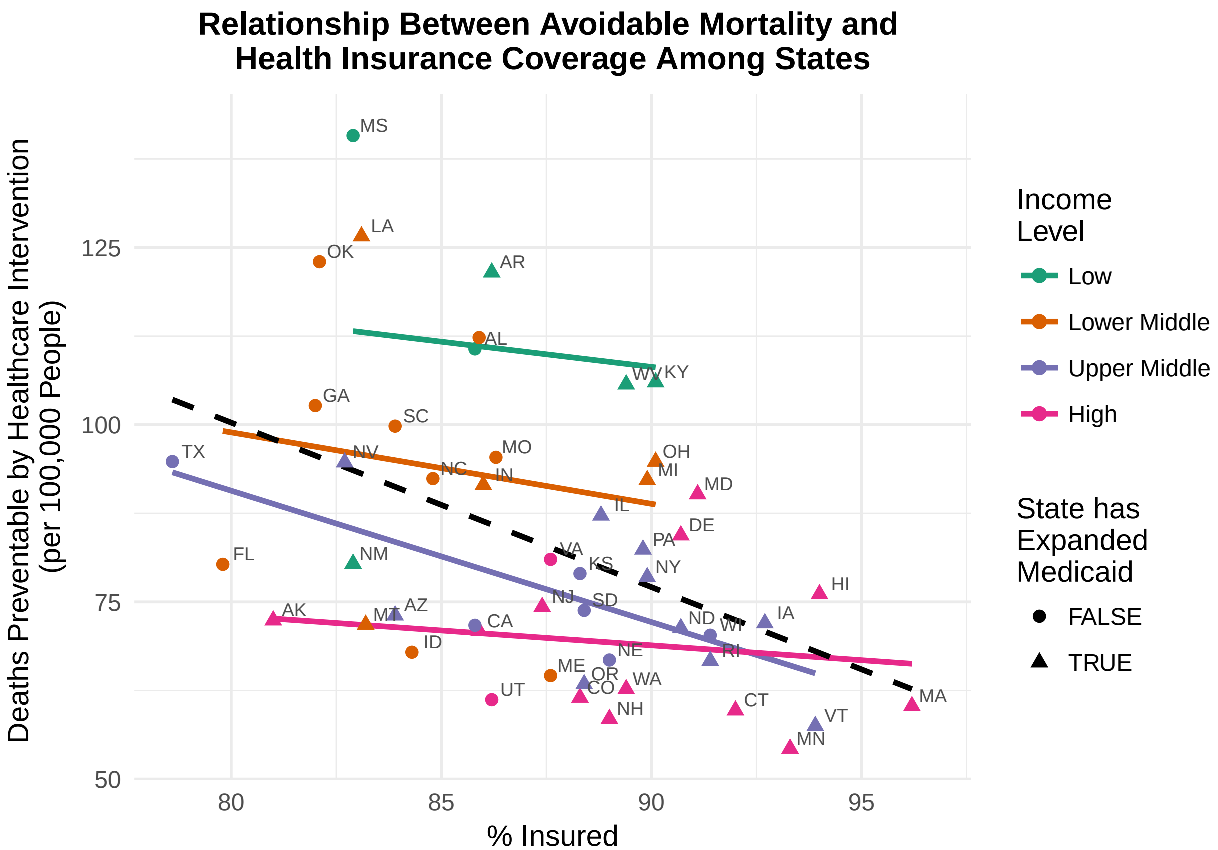


Figure 12:

Ultimately, we went with labeling the state abbreviations on the plot itself, ensuring that it was small enough not to be too distracting and that it didn't overlap with the points.

Final figure 8:

```
## PLOT 8
plot8 <- (ggplot(key_indicators, aes(insured_pct_2014, deaths_amenable_2014)) +
    geom_smooth(aes(color = income_level, group = income_level), method = "lm",
        lwd = 1, se = FALSE) + geom_smooth(color = "black", method = "lm", linetype = 2,
    lwd = 1, se = FALSE) + geom_point(aes(color = income_level, shape = state_has_expanded),
    size = 2) + geom_text(aes(label = state_abb), nudge_x = 0.5, nudge_y = 1.5,
    check_overlap = TRUE, size = 2.5, color = "gray30") + labs(x = "% Insured",
    y = "Deaths Preventable by Healthcare Intervention \n(per 100,000 People)",
    color = "Income \nLevel", shape = "State has \nExpanded \nMedicaid") + scale_color_manual(values = 
    name = "Dark2")) + ggtitle("Relationship Between Avoidable Mortality and \nHealth Insurance Coverage
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0.5,
    size = 12)))
plot8
```



Next, to prepare our data for the leaflet maps sections, we did a lot of pre-processing of our map_states spatial object. Things we had to deal with included the fact that some states had multiple polygons (e.g. Michigan which is split up), yet when evaluating variables we had to match the correct colors to all the polygons within the state. The following code was how we figured out the process.

```
## DATA PROCESSING FOR MAPS

# Obtain shape files of US states
map_states = map("state", fill = TRUE, plot = FALSE)

# Standardize name format with rest of assignment Function to conver to
# proper case
properCase <- function(x) {
    s <- strsplit(x, " ")[[1]]
    paste(toupper(substring(s, 1, 1)), substring(s, 2), sep = "", collapse = " ")
}
# Convert to proper case
map_states$state <- sapply(map_states$names, properCase)
# Rename the main component of states with multiple parts in the map
map_states$state[map_states$state == "Massachusetts:main"] <- "Massachusetts"
map_states$state[map_states$state == "Michigan:south"] <- "Michigan"
map_states$state[map_states$state == "New York:main"] <- "New York"
map_states$state[map_states$state == "North Carolina:main"] <- "North Carolina"
map_states$state[map_states$state == "Virginia:main"] <- "Virginia"
map_states$state[map_states$state == "Washington:main"] <- "Washington"
# Match with state abbreviations
map_states$state_abb <- state_abb_lookup$state_abb[match(map_states$state, state_abb_lookup$state)]
# Then strip off all the extra location information for states with multiple
# parts (leaving just the state name)
for (i in 1:length(map_states$state)) {
    map_states$state[i] <- unlist(strsplit(map_states$state[i], ":"))[1]
}

# Get coordinates of state centers
states_centers <- state.center
state.center <- cbind(states_centers, state_abb_lookup)

# Health outcomes: Import key indicators data, matched by state Remember to
# later match information based on state, not state abbreviation (due to how
# we labelled above)
map_states$deaths_amenable_2014 <- key_indicators$deaths_amenable_2014[match(map_states$state,
    key_indicators$state)]
map_states$usual_care_2015 <- key_indicators$usual_care_2015[match(map_states$state,
    key_indicators$state)]
map_states$with_cancer_screening <- key_indicators$with_cancer_screening[match(map_states$state,
    key_indicators$state)]
map_states$with_vaccines <- key_indicators$with_vaccines[match(map_states$state,
    key_indicators$state)]

# Cost outcomes: Import key indicators data, matched by state Remember to
# later match information based on state, not state abbreviation (due to how
# we labelled above)
map_states$high_OOP_relative <- key_indicators$high_OOP_relative[match(map_states$state,
    key_indicators$state)]
```

```
map_states$premium_ann_growth_10_15 <- key_indicators$premium_ann_growth_10_15[match(map_states$state,
    key_indicators$state)]
map_states$IM_plan_under_100 <- key_indicators$IM_plan_under_100[match(map_states$state,
    key_indicators$state)]
map_states$incr_fed_spending_mil <- key_indicators$incr_fed_spending_mil[match(map_states$state,
    key_indicators$state)]
```

Figure 9

We deliberated with what kind of scales to use for our leaflet map. We tried out a numeric scale and then a 6-color quantile scale but found that the color was actually too much detail and distraction. We decided that 4 bins was actually ideal, and so stuck with the default.

Numeric scale

```
# PROCESS PLOT
(leaflet(map_states) %>% setView(lat = 39.8282, lng = -96, zoom = 4) %>% addPolygons(color = "#333333",
    weight = 1, smoothFactor = 0.5, fillOpacity = 0) %>% addPolygons(group = "Preventable Deaths",
    fillColor = ~colorNumeric("RdYlGn", -deaths_amenable_2014)(-deaths_amenable_2014),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Preventable Mortality</b>", "<br/>",
        "<b>per 100,000: </b>", map_states$deaths_amenable_2014)) %>% addPolygons(group = "Access to Usu
    fillColor = ~colorNumeric("RdYlGn", usual_care_2015)(usual_care_2015), smoothFactor = 0.5,
    stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>", map_states$state,
        "<br/>", "<b>Access to Usual</b>", "<br/>", "<b>Source of Care: </b>",
        map_states$usual_care_2015, "%")) %>% addPolygons(group = "Cancer Screenings Rate",
    fillColor = ~colorNumeric("RdYlGn", with_cancer_screening)(with_cancer_screening),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Cancer Screenings</b>", "<br/>", "<b>Rate: </b>",
        map_states$with_cancer_screening, "%")) %>% addPolygons(group = "Adult Vaccination Rate",
    fillColor = ~colorNumeric("RdYlGn", with_vaccines)(with_vaccines), smoothFactor = 0.5,
    stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>", map_states$state,
        "<br/>", "<b>Vaccination</b>", "<br/>", "<b>Rate: </b>", map_states$with_vaccines,
        "%")) %>% addLabelOnlyMarkers(data = filter(state.center, state_abb !=
    "AK" & state_abb != "HI"), lng = ~x, lat = ~y, label = ~state_abb, labelOptions = labelOptions(noHid
    direction = "top", textOnly = T)) %>% addLayersControl(baseGroups = c("Preventable Deaths",
    "Access to Usual Care", "Cancer Screenings Rate", "Adult Vaccination Rate"),
    options = layersControlOptions(collapsed = FALSE)))
```

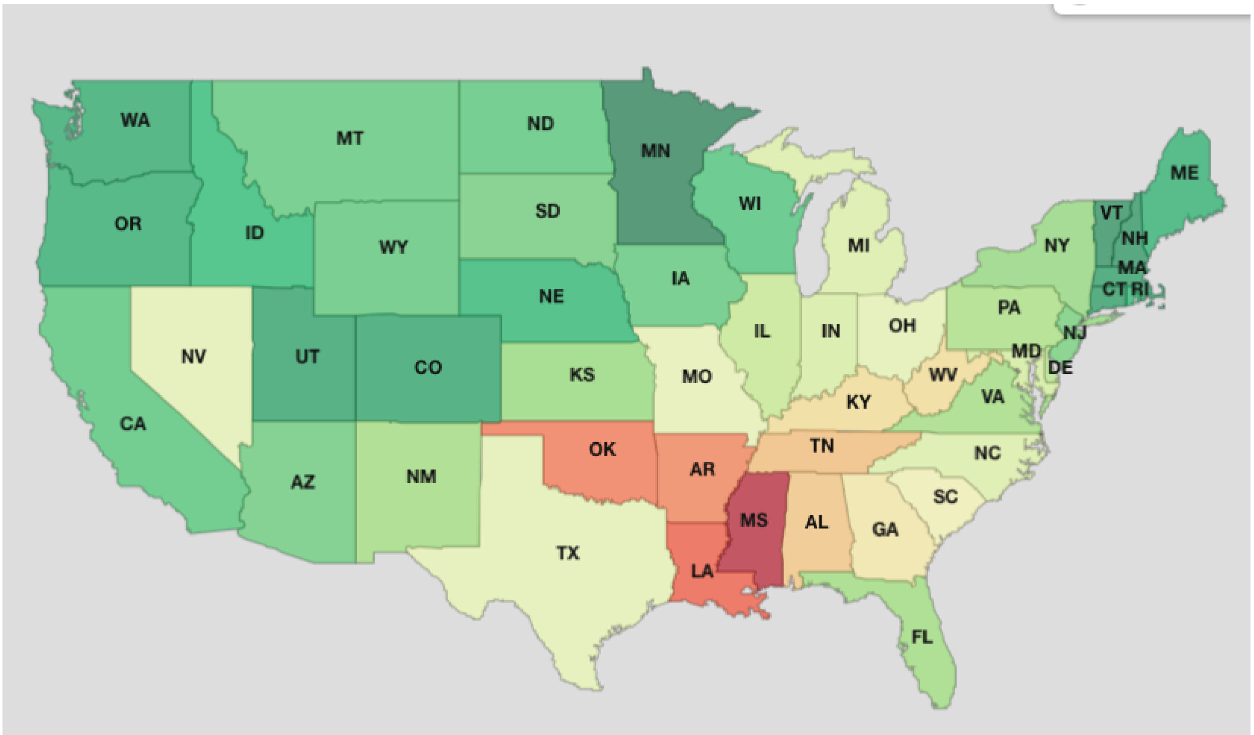


Figure 13:

6-level quantile scale

```
## PROCESS PLOT
(leaflet(map_states) %>% setView(lat = 39.8282, lng = -96, zoom = 4) %>% addPolygons(color = "#333333",
    weight = 1, smoothFactor = 0.5, fillOpacity = 0) %>% addPolygons(group = "Preventable Deaths",
    fillColor = ~colorQuantile("RdYlGn", -deaths_amenable_2014, n = 6)(-deaths_amenable_2014),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Preventable Mortality</b>", "<br/>",
        "<b>per 100,000: </b>", map_states$deaths_amenable_2014)) %>% addPolygons(group = "Access to Us
    fillColor = ~colorQuantile("RdYlGn", usual_care_2015, n = 6)(usual_care_2015),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Access to Usual</b>", "<br/>", "<b>Source of Care: </b>",
        map_states$usual_care_2015, "%")) %>% addPolygons(group = "Cancer Screenings Rate",
    fillColor = ~colorQuantile("RdYlGn", with_cancer_screening, n = 6)(with_cancer_screening),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Cancer Screenings</b>", "<br/>", "<b>Rate: </b>",
        map_states$with_cancer_screening, "%")) %>% addPolygons(group = "Adult Vaccination Rate",
    fillColor = ~colorQuantile("RdYlGn", with_vaccines, n = 6)(with_vaccines),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Vaccination</b>", "<br/>", "<b>Rate: </b>",
        map_states$with_vaccines, "%")) %>% addLabelOnlyMarkers(data = filter(state.center,
    state_abb != "AK" & state_abb != "HI"), lng = ~x, lat = ~y, label = ~state_abb,
    labelOptions = labelOptions(noHide = T, direction = "top", textOnly = T)) %>%
    addLayersControl(baseGroups = c("Preventable Deaths", "Access to Usual Care",
        "Cancer Screenings Rate", "Adult Vaccination Rate"), options = layersControlOptions(collapsed =
```

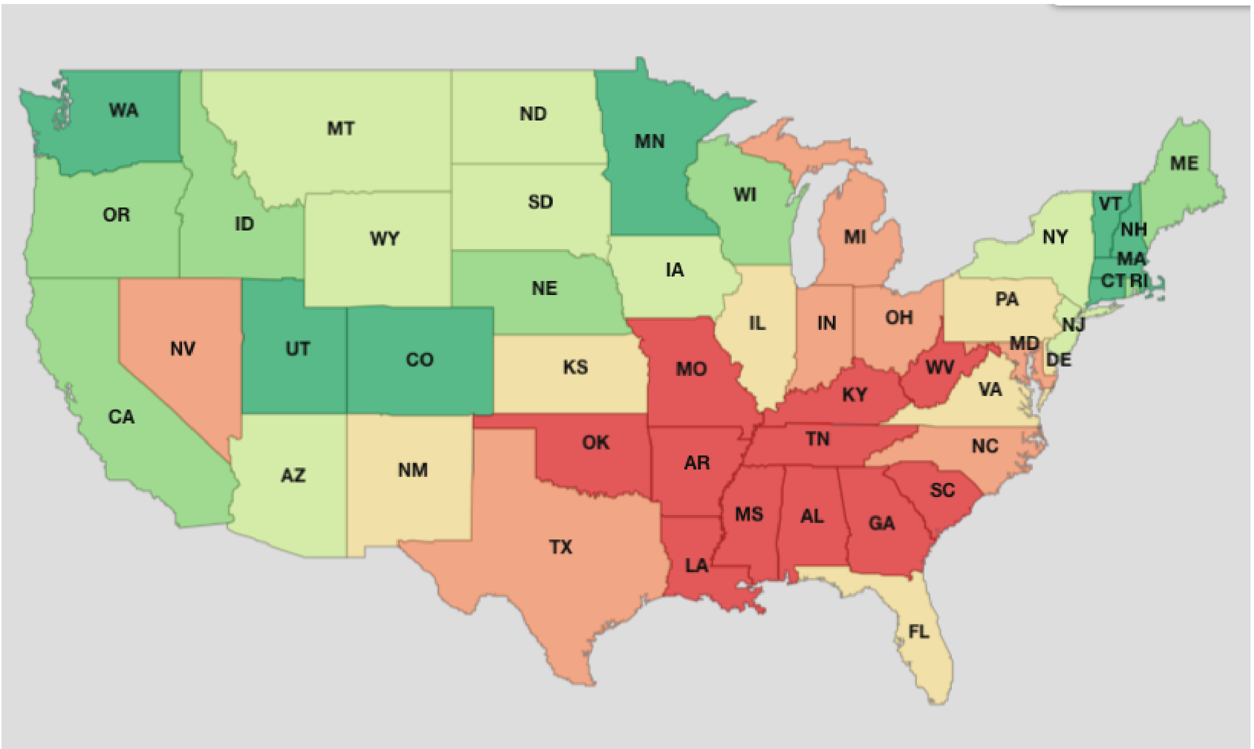


Figure 14:

Final figure 9:

4-level quantile scale

```
## PLOT 9: LEAFLET MAP - HEALTH OUTCOMES
(leaflet(map_states) %>% setView(lat = 39.8282, lng = -96, zoom = 4) %>% addPolygons(color = "#333333",
    weight = 1, smoothFactor = 0.5, fillOpacity = 0) %>% addPolygons(group = "Preventable Deaths (2014)
    fillColor = ~colorQuantile("RdYlGn", -deaths_amenable_2014)(-deaths_amenable_2014),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Preventable Mortality</b>", "<br/>",
        "<b>per 100,000: </b>", map_states$deaths_amenable_2014)) %>% addPolygons(group = "Access to Usu
    fillColor = ~colorQuantile("RdYlGn", usual_care_2015)(usual_care_2015),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Access to Usual</b>", "<br/>", "<b>Source of Care: </b>",
        map_states$usual_care_2015, "%")) %>% addPolygons(group = "Cancer Screenings Rate (2014)",
    fillColor = ~colorQuantile("RdYlGn", with_cancer_screening)(with_cancer_screening),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Cancer Screenings</b>", "<br/>", "<b>Rate: </b>",
        map_states$with_cancer_screening, "%")) %>% addPolygons(group = "Adult Vaccination Rate (2015)"
    fillColor = ~colorQuantile("RdYlGn", with_vaccines)(with_vaccines), smoothFactor = 0.5,
    stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>", map_states$state,
        "<br/>", "<b>Vaccination</b>", "<br/>", "<b>Rate: </b>", map_states$with_vaccines,
        "%")) %>% addLabelOnlyMarkers(data = filter(state.center, state_abb !=
    "AK" & state_abb != "HI"), lng = ~x, lat = ~y, label = ~state_abb, labelOptions = labelOptions(noHid
    direction = "top", textOnly = T)) %>% addLayersControl(baseGroups = c("Preventable Deaths (2014)",
    "Access to Usual Care (2015)", "Cancer Screenings Rate (2014)", "Adult Vaccination Rate (2015)"),
    options = layersControlOptions(collapsed = FALSE)))
```

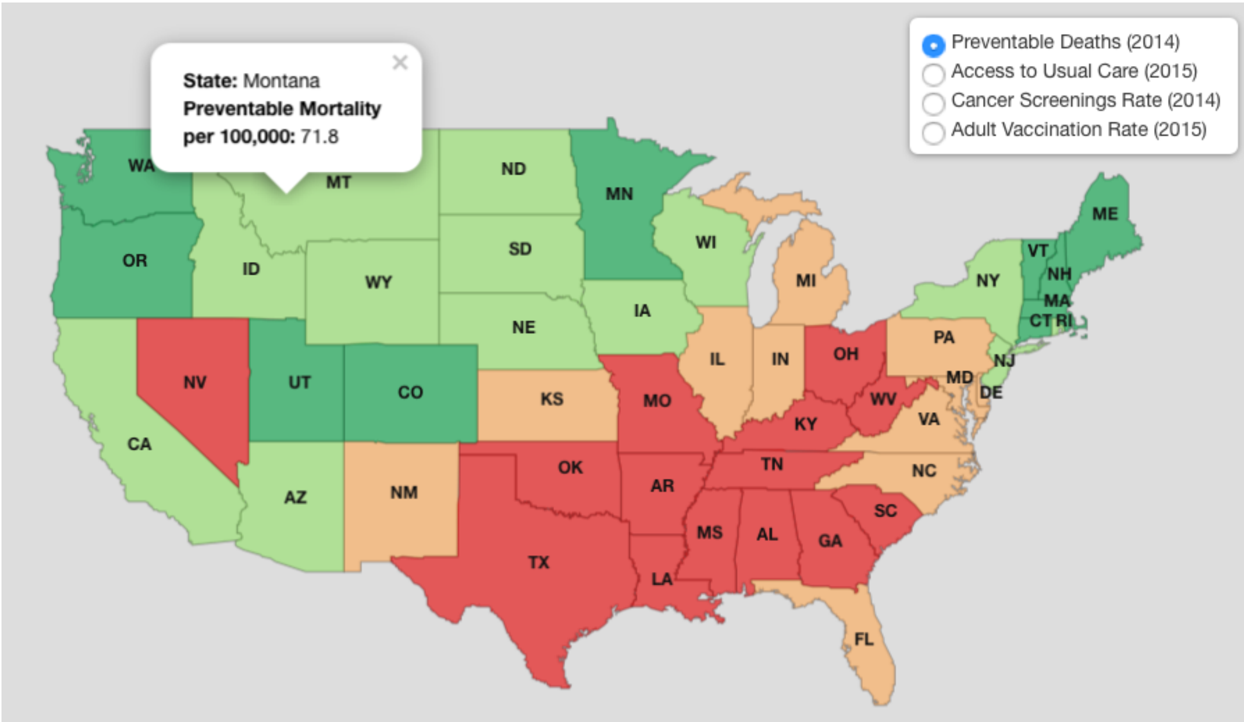


Figure 15:

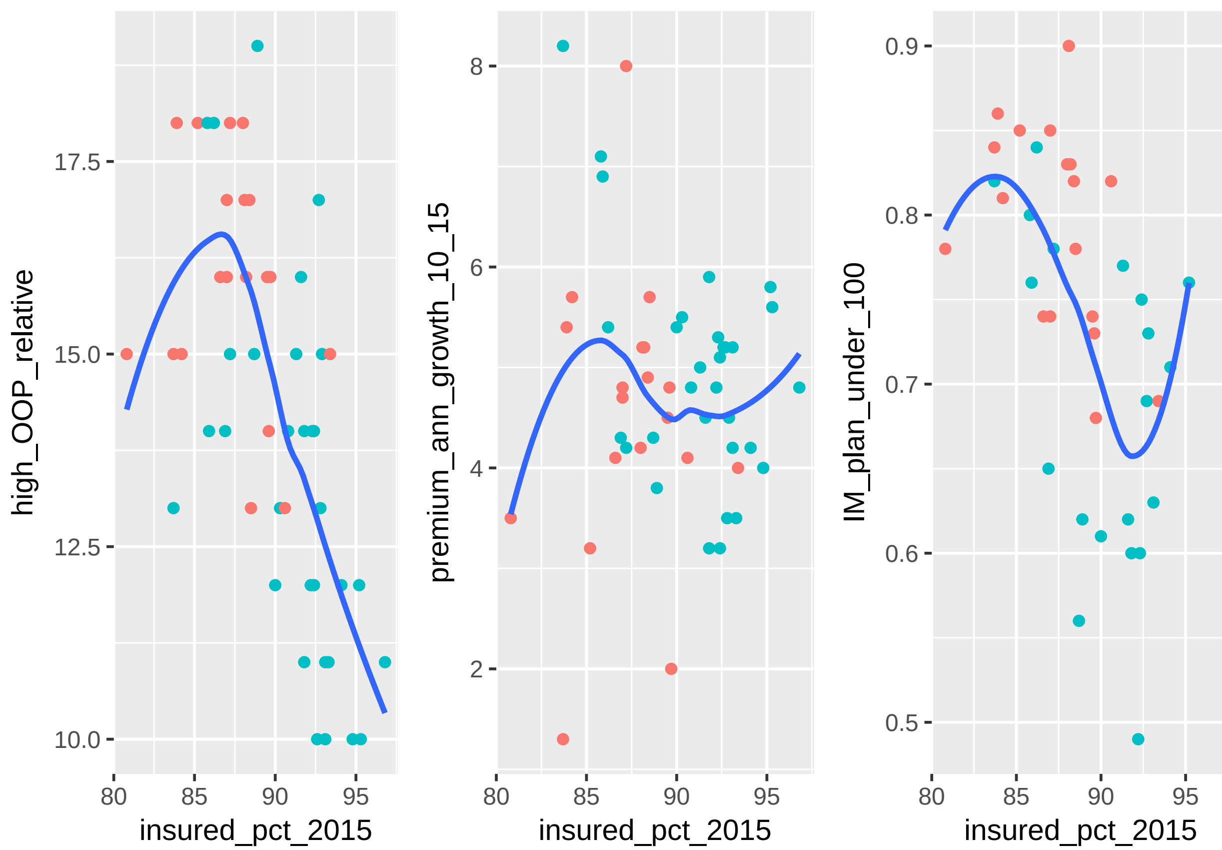
Figure 10

This was a new plot from after our presentation, but we realized we did not have a plot explicitly correlating financial outcomes to insurance rates (like what we had for our health outcomes section).

However, there were 3-4 key financial varaibles that correlated differently with insured rates. We thus explored correlating each of them separately, to see which had a signficiant trend we could report.

```r
# PROCESS PLOTS
plot10a <- (ggplot(key_indicators, aes(insured_pct_2015, high_OOP_relative)) +
    geom_point(aes(color = state_has_expanded)) + geom_smooth(se = FALSE) +
    theme(legend.position = "none"))
plot10b <- (ggplot(key_indicators, aes(insured_pct_2015, premium_ann_growth_10_15)) +
    geom_point(aes(color = state_has_expanded)) + geom_smooth(se = FALSE) +
    theme(legend.position = "none"))
plot10c <- (ggplot(key_indicators, aes(insured_pct_2015, IM_plan_under_100)) +
    geom_point(aes(color = state_has_expanded)) + geom_smooth(se = FALSE) +
    theme(legend.position = "none"))
grid.arrange(plot10a, plot10b, plot10c, ncol = 3)
```

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


high_OOP_relative
17.5
15.0
12.5
10.0
80
85
90
95
insured_pct_2015
premium_ann_growth_10_15
8
6
4
2
80
85
90
95
insured_pct_2015
IM_plan_under_100
0.9
0.8
0.7
0.6
0.5
80
85
90
95
insured_pct_2015

The only variable with a significant correlation trend with insured percentage was the percentage of high OOP people. Therefore, we used that as our y-variable, and create a graph that looked analogous to our health outcomes plot.

Final figure 10:

```
## PLOT 10
plot10 <- (ggplot(key_indicators, aes(insured_pct_2015, high_OOP_relative, label = state_abb))
            + geom_smooth(aes(color = income_level), method = "lm", lwd = 1, se = FALSE)
            + geom_smooth(color = "black", method = "lm", linetype = 2, lwd = 1, se = FALSE)
            + geom_point(aes(color = income_level, shape = state_has_expanded), size = 2)
           + geom_text(aes(label = state_abb), nudge_y = -0.2, check_overlap = TRUE, size = 2.5, color =

           + scale_color_manual(values = brewer.pal(n = 4, name = "Dark2"))
           + labs(x = "% Insured", y = "People with High OOP Costs Relative to Household Income", color
           + ggtitle("Relationship Between OOP Costs and Health \nInsurance Coverage Among States")
           + theme_minimal()
           + theme(plot.title = element_text(face="bold", hjust = 0.5, size = 12)))
plot10
```

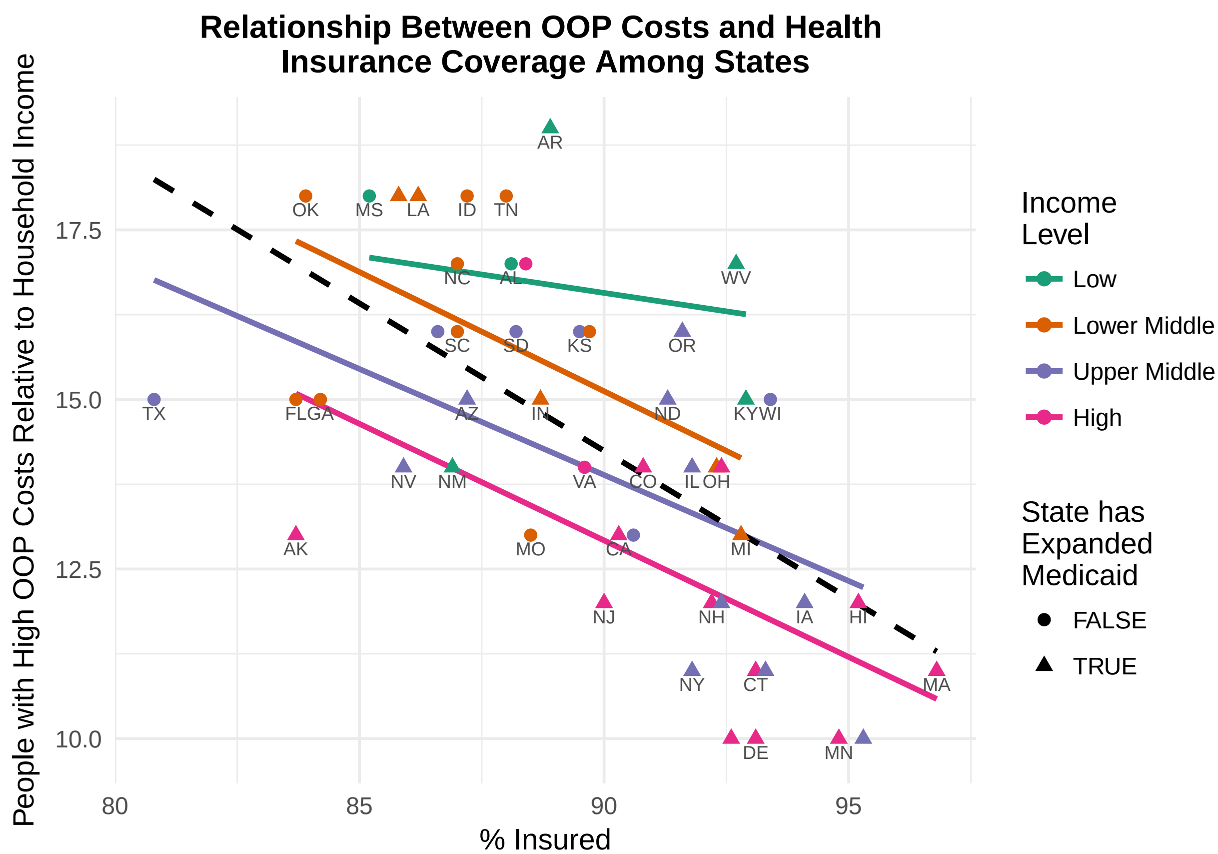


Figure 11

Final figure 11 - leaflet plot of cost outcomes:

```
## PLOT 11: LEAFLET MAP - AFFORDABILITY / COST OUTCOMES
marketplace_pct <- map_states$IM_plan_under_100
marketplace_pct <- marketplace_pct * 100
marketplace_pct[is.na(marketplace_pct)] <- "No State Marketplace"
marketplace_pct_symbol <- ifelse(marketplace_pct == "No State Marketplace",
    "", "%")

(leaflet(map_states) %>% setView(lat = 39.8282, lng = -96, zoom = 4) %>% addPolygons(color = "#333333",
    weight = 1, smoothFactor = 0.5, fillOpacity = 0) %>% addPolygons(group = "Contained OOP Costs (2015
    fillColor = ~colorQuantile("RdYlGn", -high_OOP_relative)(-high_OOP_relative),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>People with High OOP</b>", "<br/>", "<b>Relative to Income: </b>
        map_states$high_OOP_relative, "%")) %>% addPolygons(group = "Premium Growth Rate (2010-2015)",
    fillColor = ~colorQuantile("RdYlGn", -premium_ann_growth_10_15)(-premium_ann_growth_10_15),
    smoothFactor = 0.5, stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>",
        map_states$state, "<br/>", "<b>Annual Premium Growth</b>", "<br/>",
        "<b>Rate (2010-2015): </b>", map_states$premium_ann_growth_10_15, "%")) %>%
    addPolygons(group = "Affordable Marketplace Plan (2017)", fillColor = ~colorQuantile("RdYlGn",
        IM_plan_under_100)(IM_plan_under_100), smoothFactor = 0.5, stroke = FALSE,
        fillOpacity = 0.6, popup = paste("<b>State: </b>", map_states$state,
            "<br/>", "<b>Marketplace Consumers</b>", "<br/>", "<b>Who Can Select Plan</b>",
            "<br/>", "<b>Under $100: </b>", marketplace_pct, marketplace_pct_symbol)) %>%
    addPolygons(group = "Increase in Federal Spending (2016)", fillColor = ~colorQuantile("RdYlGn",
        -incr_fed_spending_mil)(-incr_fed_spending_mil), smoothFactor = 0.5,
        stroke = FALSE, fillOpacity = 0.6, popup = paste("<b>State: </b>", map_states$state,
            "<br/>", "<b>Net Increase in Federal</b>", "<br/>", "<b>Spending</b>: $",
            map_states$incr_fed_spending_mil, "million")) %>% addLabelOnlyMarkers(data = filter(state.c
    state_abb != "AK" & state_abb != "HI"), lng = ~x, lat = ~y, label = ~state_abb,
    labelOptions = labelOptions(noHide = T, direction = "top", textOnly = T)) %>%
    addLayersControl(baseGroups = c("Contained OOP Costs (2015)", "Premium Growth Rate (2010-2015)",
        "Affordable Marketplace Plan (2017)", "Increase in Federal Spending (2016)"),
        options = layersControlOptions(collapsed = FALSE)))
```

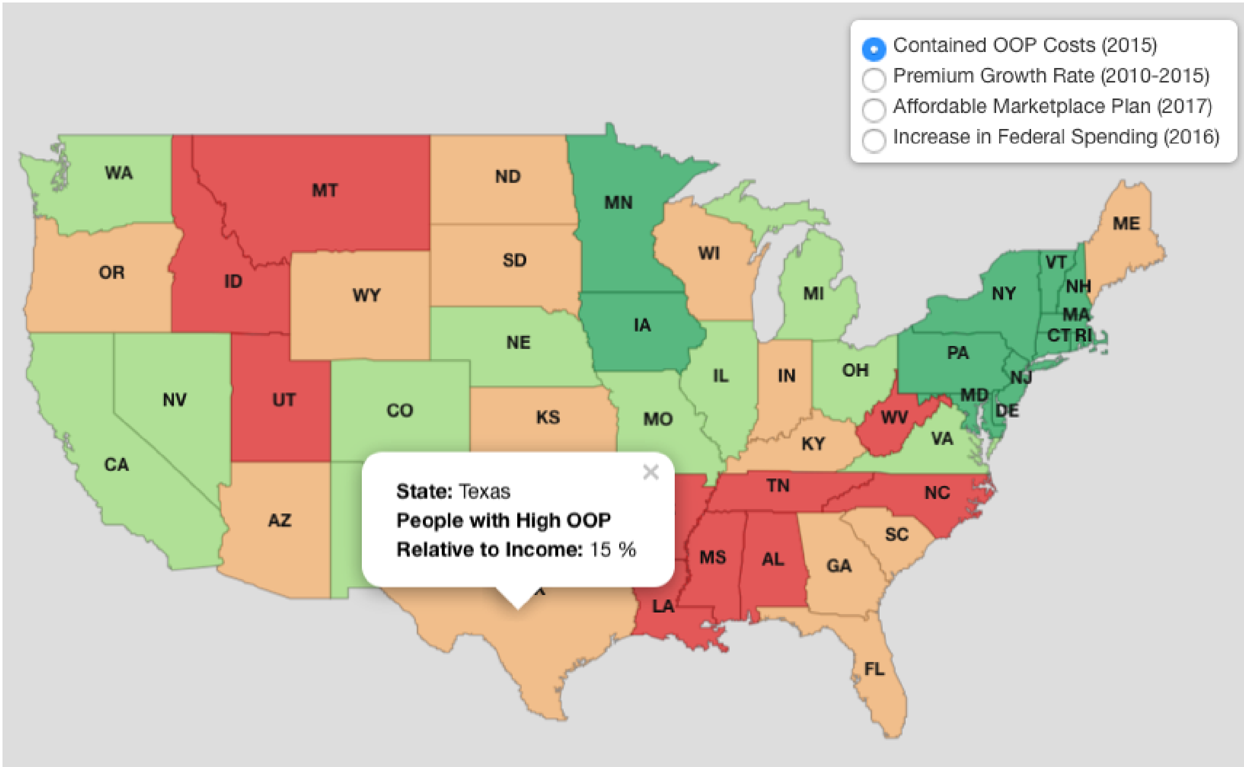


Figure 16:

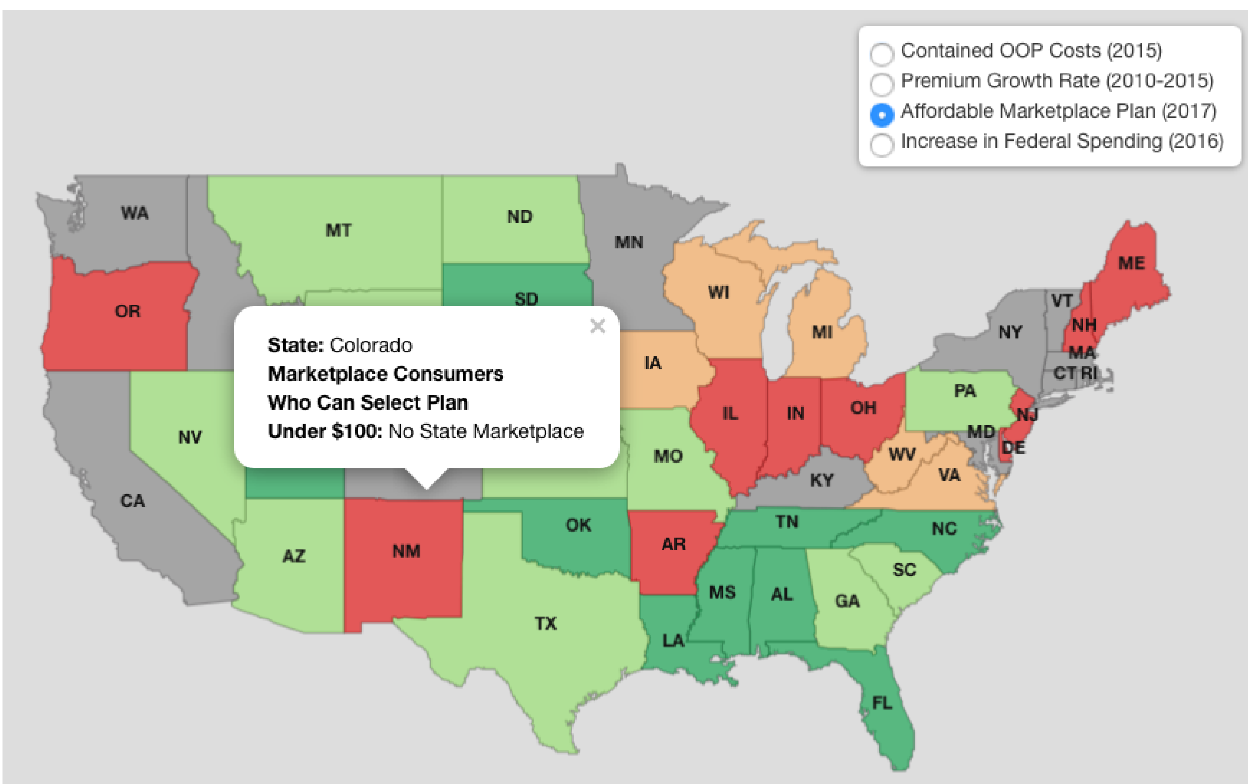


Figure 17:

Finally, we prepared to rank states on various measures using a data table. We distilled form our 10+ key indicators to 6 dimensions to rank states on. We made a version of the data table with values, and a version with the explicit ranks.

```
# Create data table reporting values
data_table_values <- key_indicators[, c("state", "insured_pct_2015", "deaths_amenable_2014",
    "usual_care_2015", "high_OOP_relative", "premium_ann_growth_10_15", "incr_fed_spending_mil")]

# Create data table reporting rank
data_table_rank <- data_table_values[, c("state", "insured_pct_2015", "deaths_amenable_2014",
    "usual_care_2015", "high_OOP_relative", "premium_ann_growth_10_15")]

# Convert each function to rank Rank 1 is always more favorable (regardless
# of definition of variable)
data_table_rank$r_insured_pct_2015 <- rank(-data_table_values$insured_pct_2015,
    na.last = "keep", ties.method = "min")
data_table_rank$r_deaths_amenable_2014 <- rank(data_table_values$deaths_amenable_2014,
    na.last = "keep", ties.method = "min")
data_table_rank$r_usual_care_2015 <- rank(-data_table_values$usual_care_2015,
    na.last = "keep", ties.method = "min")
data_table_rank$r_high_OOP_relative <- rank(data_table_values$high_OOP_relative,
    na.last = "keep", ties.method = "min")
data_table_rank$r_premium_ann_growth_10_15 <- rank(data_table_values$premium_ann_growth_10_15,
    na.last = "keep", ties.method = "min")
data_table_rank$r_incr_fed_spending_mil <- rank(data_table_values$incr_fed_spending_mil,
    na.last = "keep", ties.method = "min")
# Remove value variables
data_table_rank$insured_pct_2015 <- NULL
data_table_rank$deaths_amenable_2014 <- NULL
data_table_rank$usual_care_2015 <- NULL
data_table_rank$high_OOP_relative <- NULL
data_table_rank$premium_ann_growth_10_15 <- NULL
data_table_rank$incr_fed_spending_mil <- NULL
```

Figure 12

We intially thought that putting the values into the data table would add additional useful information. And we thought that the rank would be apparent just by looking at the relative position in the data table. But because there were 50 rows, it was unfeasible for a viewer to count which row the state belonged to. Thus, we decided to report the explicit ranks instead, since ranking was the objective of this section. The detailed values could be seen in our leaflet maps anyway.

By values

```
## PROCESS PLOT
datatable(data_table_values, rownames = FALSE, colnames = c("State", "Percentage Insured",
    "Preventable Deaths Per 100,000", "Usual Care Access (%)", "High OOP Relative to Income",
    "Annual Premium Growth", "Increase in Federal Spending ($M)"), caption = "Performance of the 50 Sta
```

Show 10 entries Search:

Performance of the 50 States in Insurance Coverage and Health Outcomes

State	Percentage Insured	Preventable Deaths Per 100,000	Usual Care Access (%)	High OOP Relative to Income	Annual Premium Growth	Increase in Federal Spending ($M)
Massachusetts	96.8	60.3	89	11	4.8	670
Vermont	95.3	57.5	88	10	5.6	110
Hawaii	95.2	76.1	85	12	5.8	280
Minnesota	94.8	54.3	77	10	4	400
Iowa	94.1	72	81	12	4.2	270
Wisconsin	93.4	70.3	81	15	4	280
Rhode Island	93.3	66.7	88	11	3.5	270
Connecticut	93.1	59.7	85	11	4.2	710
Delaware	93.1	84.4	85	10	5.2	170
Kentucky	92.9	106	83	15	4.5	1640

Showing 1 to 10 of 50 entries Previous 1 2 3 4 5 Next

Figure 18:

Final figure 12 (data table):

By ranks

```
## PLOT 12: DATA TABLE RANKING STATES ON VARIOUS MEASURES
datatable(data_table_rank, rownames = FALSE, colnames = c("State", "Percentage Insured",
    "Fewer Preventable Deaths", "Usual Care Access", "OOP Contained", "Lower Annual Premium Growth",
    "Lower Increase in Federal Spending"), caption = "Rank of the 50 States in Insurance Coverage, Heal
```

Show 10 entries

Search:

Rank of the 50 States in Insurance Coverage, Health and Cost Outcomes

State	Percentage Insured	Fewer Preventable Deaths	Usual Care Access	OOP Contained	Lower Annual Premium Growth	Lower Increase in Federal Spending
Massachusetts	1	5	1	5	24	22
Vermont	2	2	2	1	42	2
Hawaii	3	24	7	9	45	14
Minnesota	4	1	34	1	10	18
Iowa	5	19	18	9	14	12
Wisconsin	6	14	18	26	10	14
Rhode Island	7	11	2	5	6	12
Connecticut	8	4	7	5	14	24
Delaware	8	31	7	1	32	5
Kentucky	10	44	12	26	20	40

Showing 1 to 10 of 50 entries

Previous 1 2 3 4 5 Next

Figure 19:

After the presentation, where we tried to verbally assess the rankings of the states, we realized that what we needed was an overall rank. Yes, it is a simplification in many regards, as we try to combine outcomes in three dimensions and reflect it in an overall rank. But as a journalistic piece, often the viewers would wonder... so which state is the best in the end?

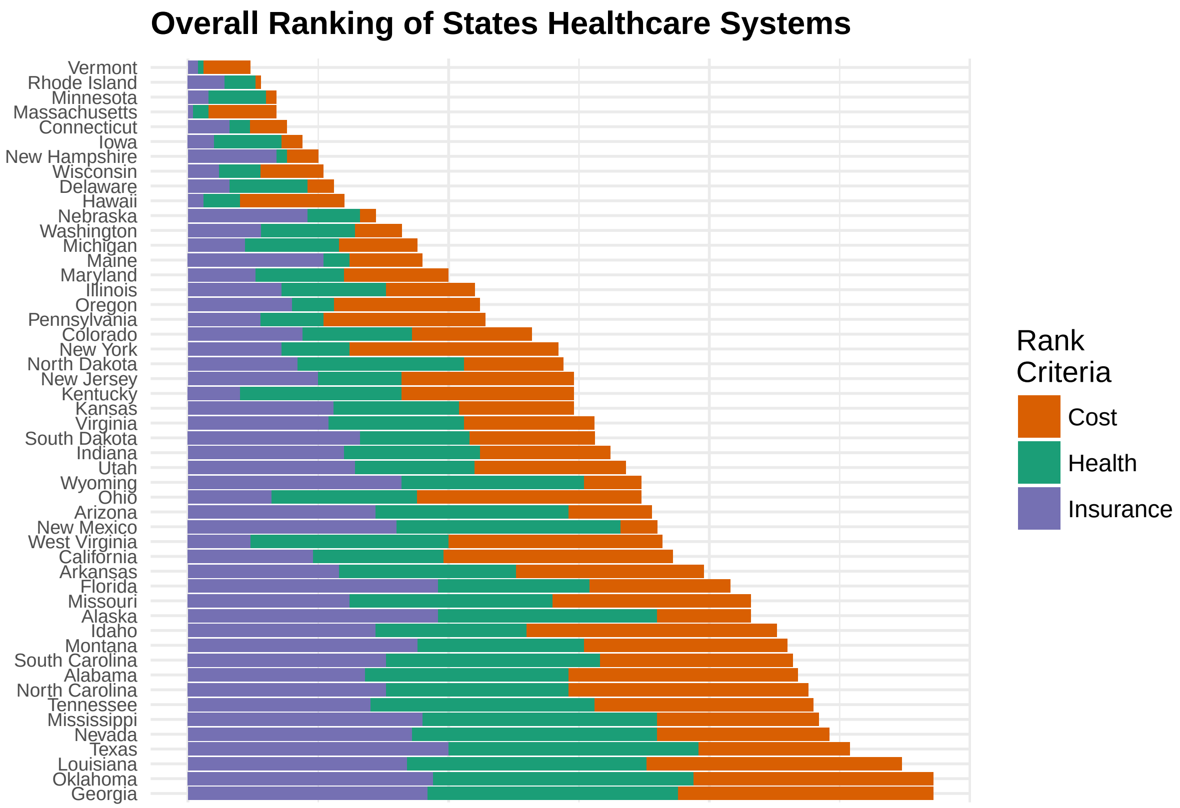
Thus, we used a simple method of aggregating the rankings of the states in the above six health measures into an overall rank. We rank each state on each of three dimensions, insurance coverage, health outcomes, and cost control outcomes. For the health and cost control outcomes, since they have multiple subcomponents (2 and 3 respectively, as in the data table), we take an average over these subcomponent ranks. We then add up the ranks in the three dimensions to find an overall rank to order the states.

```
# Compute overall ranking of states by each of three measures
data_table_rank_cat <- data_table_rank
# Insurance coverage
data_table_rank_cat$insurance_rank <- data_table_rank$r_insured_pct_2015
# Health
data_table_rank_cat$health_rank_sum <- data_table_rank$r_deaths_amenable_2014 +
    data_table_rank$r_usual_care_2015
data_table_rank_cat$health_rank <- rank(data_table_rank_cat$health_rank_sum,
    na.last = "keep", ties.method = "min")
# Finance
data_table_rank_cat$cost_rank_sum <- data_table_rank$r_high_OOP_relative + data_table_rank$r_premium_ann
    data_table_rank$r_incr_fed_spending_mil
data_table_rank_cat$cost_rank <- rank(data_table_rank_cat$cost_rank_sum, na.last = "keep",
    ties.method = "min")

data_table_rank_cat <- data_table_rank_cat[, c("state", "insurance_rank", "health_rank",
    "cost_rank")]
data_table_rank_cat$sum_of_ranks <- data_table_rank_cat$insurance_rank + data_table_rank_cat$health_rank
    data_table_rank_cat$cost_rank
data_table_rank_cat$overall_rank <- rank(data_table_rank_cat$sum_of_ranks, na.last = "keep",
    ties.method = "min")

data_table_rank_cat_long <- data_table_rank_cat
data_table_rank_cat_long <- reshape(data_table_rank_cat, varying = c("insurance_rank",
    "health_rank", "cost_rank"), v.names = "Rank", timevar = "Category", times = c("Insurance",
    "Health", "Cost"), new.row.names = 1:1000, direction = "long")
```

Figure 13

Final figure 13. Our actual plot is an interactive one, where you can toggle the legend to chose which ranks to 'add up' in the figure.

```
plot13 <- (ggplot(data_table_rank_cat_long, aes(x = reorder(state, -overall_rank),
    y = Rank, fill = Category, label = overall_rank)) + geom_col(position = "stack") +
    coord_flip() + labs(x = "", y = "", fill = "Rank \nCriteria") + scale_fill_manual(values = brewer.pa
    name = "Dark2")[c(2, 1, 3)]) + ggtitle("Overall Ranking of States Healthcare Systems") +
    theme_minimal() + theme(plot.title = element_text(face = "bold", hjust = 0,
    size = 12), axis.text.y = element_text(size = 6.5), axis.ticks.x = element_blank(),
    axis.text.x = element_blank()))
plot13
```



Toggling the legend

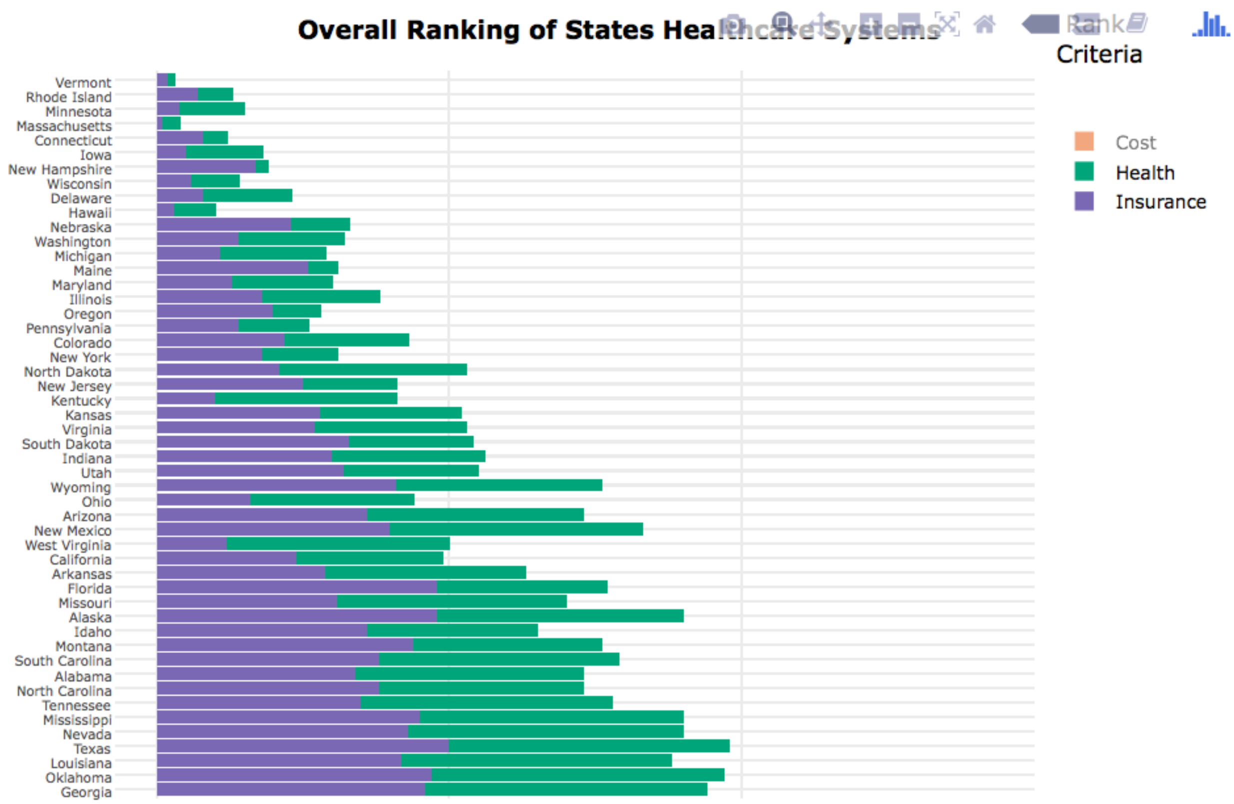
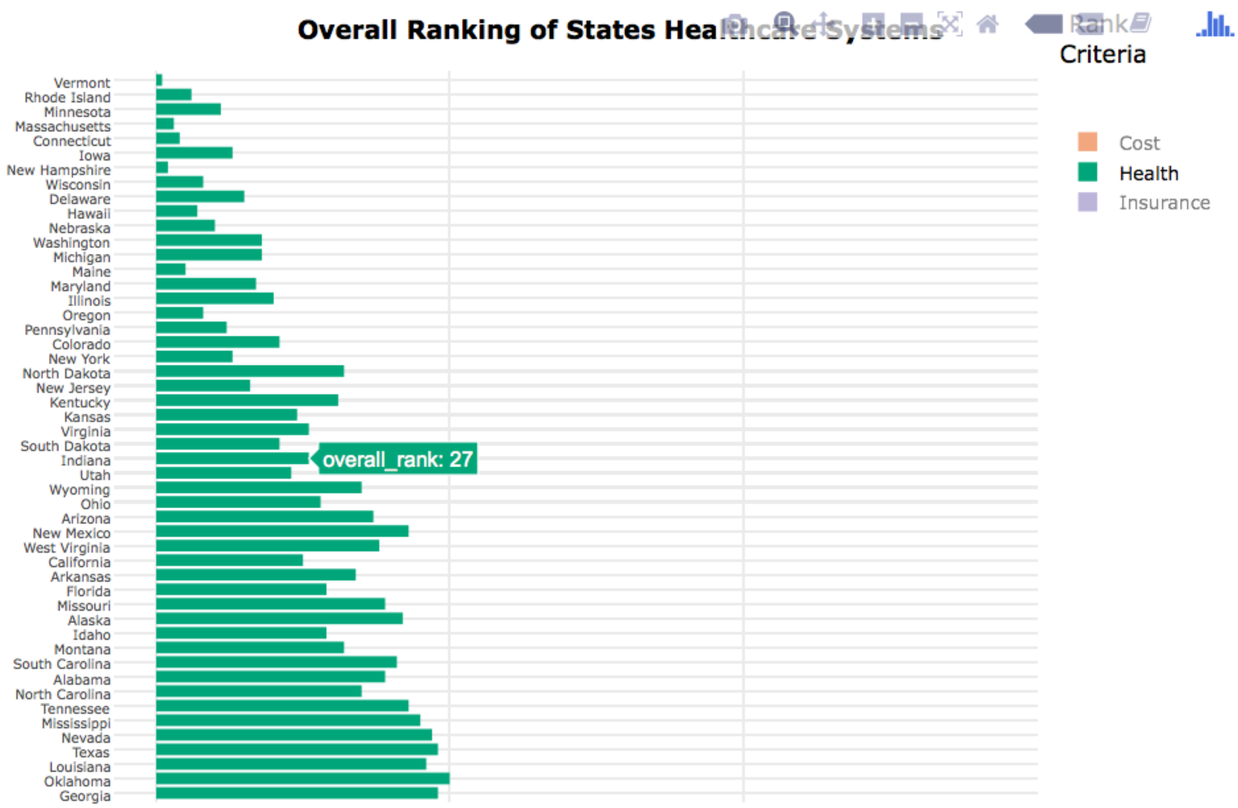


Figure 20:



Overall Ranking of States Healthcare Systems
Rank
Criteria
Cost
Health
Insurance
Vermont
Rhode Island
Minnesota
Massachusetts
Connecticut
Iowa
New Hampshire
Wisconsin
Delaware
Hawaii
Nebraska
Washington
Michigan
Maine
Maryland
Illinois
Oregon
Pennsylvania
Colorado
New York
North Dakota
New Jersey
Kentucky
Kansas
Virginia
South Dakota
Indiana
Utah
Wyoming
Ohio
Arizona
New Mexico
West Virginia
California
Arkansas
Florida
Missouri
Alaska
Idaho
Montana
South Carolina
Alabama
North Carolina
Tennessee
Mississippi
Nevada
Texas
Louisiana
Oklahoma
Georgia
overall_rank: 27

Figure 21:

Perceptions of ACA - NY Times Analysis

We are also interested in how people think about Affordable care act(ACA). We want to have a look of the tweets and newspaper related to ACA. To realize this, we use API to download articles realted to ACA on New york times from 2011-2017 and transform them into corpus to do the text analysis.

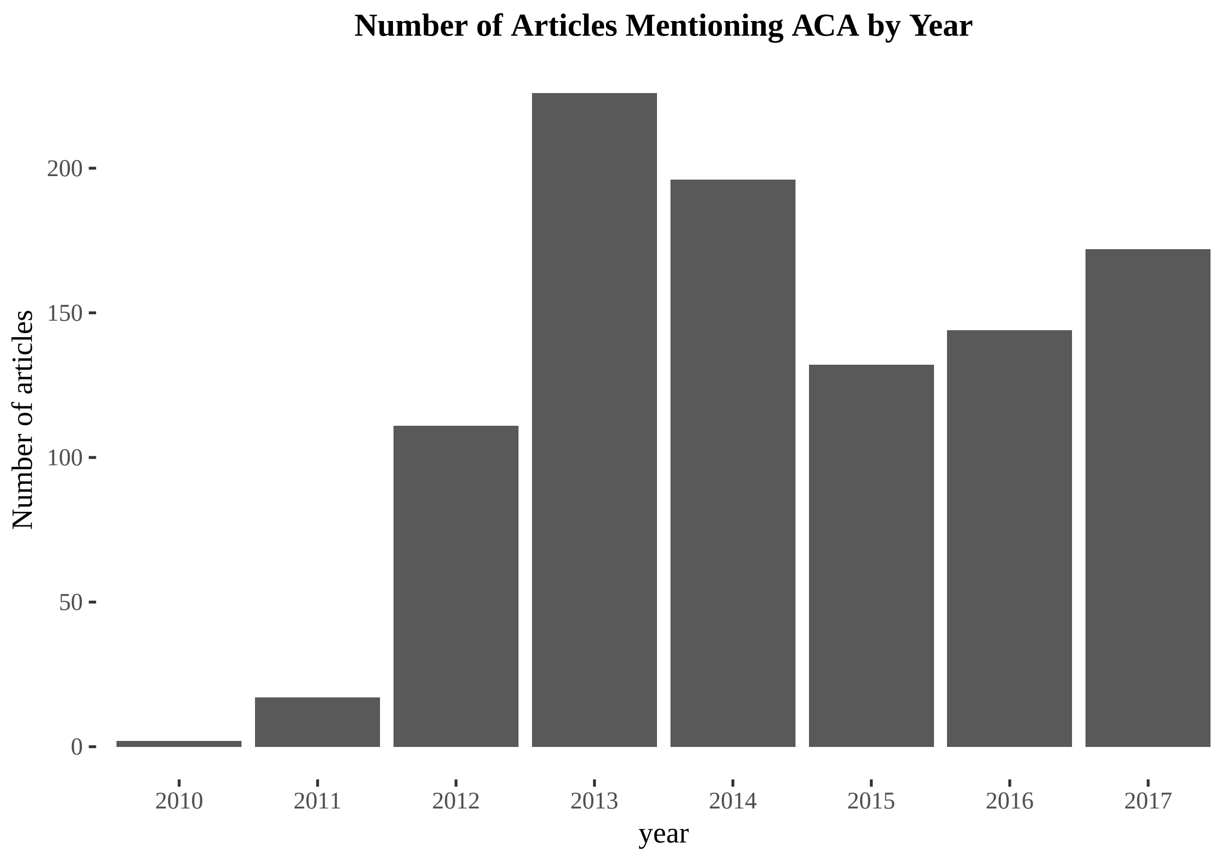
Research Questions: 1. The overll trend of public attention, which can be shown as the number of articles related to ACA in different years. 2. What people discuss about when they discuss ACA, which can be shown as the word frequencies. We could see that people discussed ACA a lot when obama first signed it, and the election in 2017 made it a hot topic again. And not suprisingly, peopel always talk about Trump when they talk about ACA

Figure 14

We could see that people discussed ACA a lot when Obama first signed it, and the election in 2017 made it a hot topic again. And not suprisingly, people always talk about Trump when they talk about ACA.

Draft plot:

```
dat <- data.frame(date = corpus$documents$datetimestamp, levels = c(1:1000))
ggplot(data = dat, aes(x = date)) + geom_bar(stat = "count") + ggtitle("Number of Articles Mentioning AC
    theme_tufte() + ylab("Number of articles") + xlab("year") + theme(plot.title = element_text(face =
    hjust = 0.5, size = 12))
```



Final figure 14:

```
dat <- data.frame(date = corpus$documents$datetimestamp, levels = c(1:1000))
ggplot(data = dat, aes(x = date)) + geom_bar(stat = "count") + ggtitle("Number of Articles Mentioning AC
    theme_tufte() + ylab("Number of articles") + xlab("year") + theme(plot.title = element_text(face =
    hjust = 0.5, size = 12)) + theme_minimal()
```

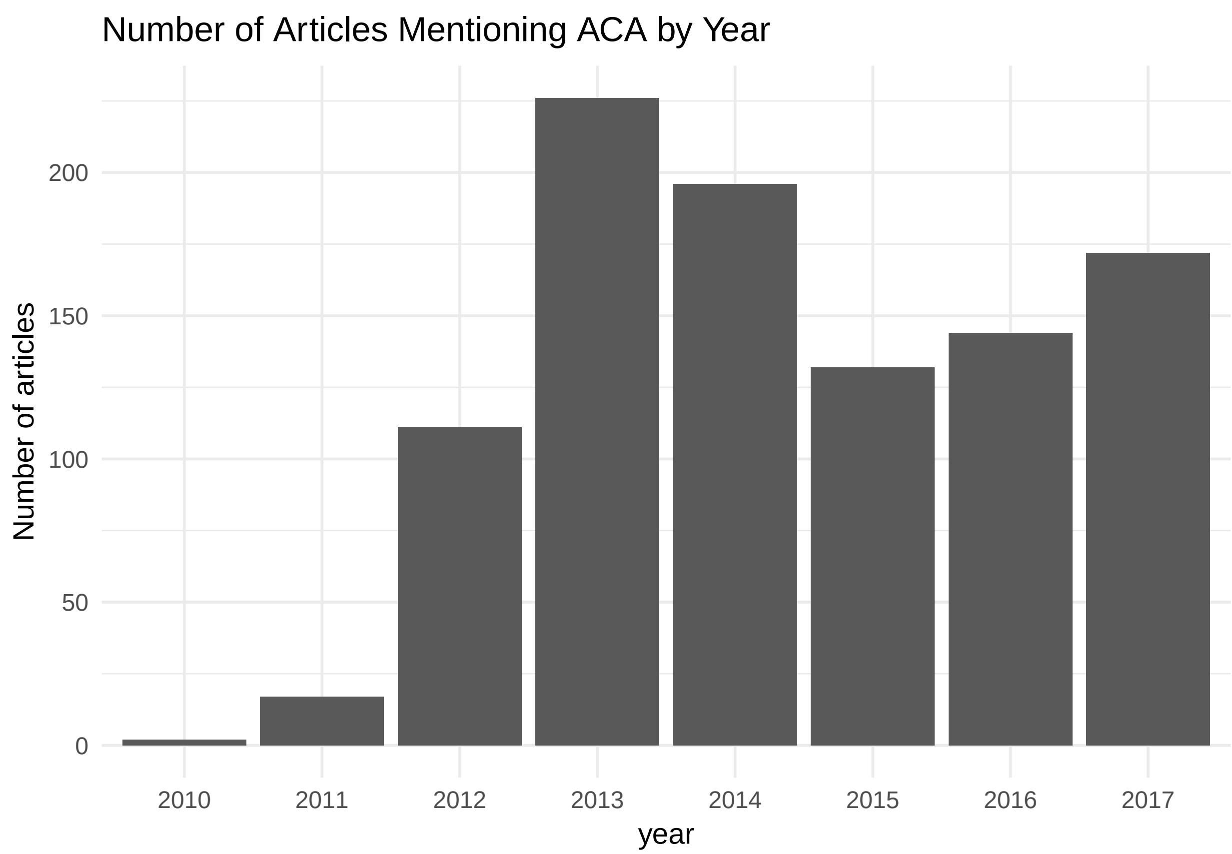


Figure 15

```
load("nytimes.rda")
dfmtotal <- dfm(corpus, remove = stopwords("english"), stem = TRUE, removePunct = TRUE,
    removeNumbers = TRUE, tolower = TRUE, verbose = TRUE)
```

```
## Creating a dfm from a corpus ...

##    ... lowercasing

##    ... tokenizing

##    ... found 1,000 documents, 22,386 features

## ...

## removed 169 features, from 174 supplied (glob) feature types

## ... stemming features (English)

## , trimmed 7834 feature variants

##    ... created a 1,000 x 14,383 sparse dfm
##    ... complete.
## Elapsed time: 1.89 seconds.
```

```
dfmtotal[, 1:5]
```

```
## Document-feature matrix of: 1,000 documents, 5 features (88.9% sparse).
```

```
head(stopwords("english"), 20)
```

```
##  [1] "i"          "me"         "my"         "myself"     "we"
```

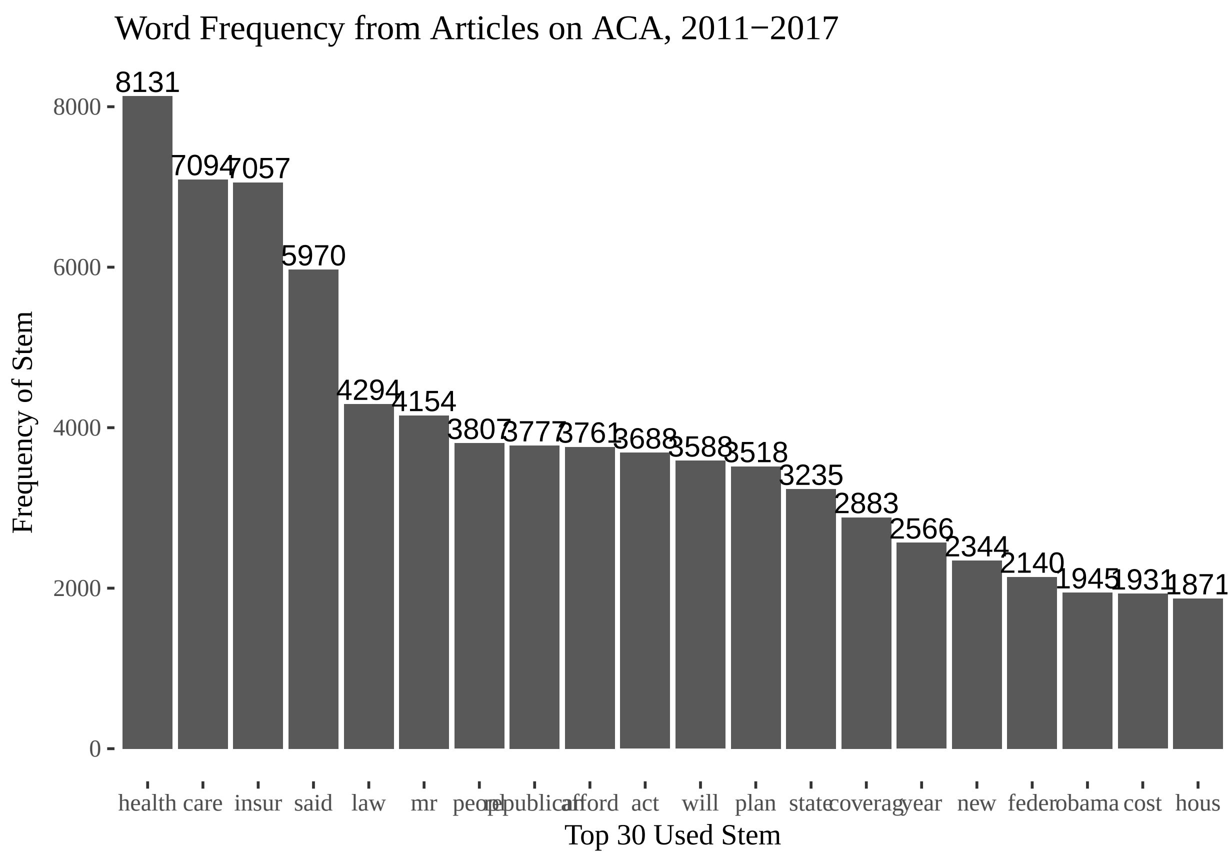
```
##  [6] "our"          "ours"         "ourselves"  "you"          "your"
## [11] "yours"        "yourself"     "yourselves" "he"           "him"
## [16] "his"          "himself"      "she"        "her"          "hers"
```

```
freq <- topfeatures(dfmtotal, 20)
wf <- data.frame(word = names(freq), freq = freq)
```

The graph shows that the frequently use words are health, insur and care. And other words like law, people and republican are also frequently used. It is not surprising that the articles most 40 features on the health care issues, and yet the topics of people, policy and politics are also highly concerned in New York Times.

Draft plot:

```
p <- ggplot(subset(wf, freq > 50), aes(word, freq))
p <- p + geom_bar(stat = "identity")
p <- p + geom_text(aes(label = freq), vjust = -0.2) + scale_x_discrete(limits = wf$word)
p <- p + theme(axis.text.x = element_text(angle = 45, hjust = 1), plot.title = element_text(face = "bol
    hjust = 0.5, size = 12)) + ggtitle("Word Frequency from Articles on ACA, 2011-2017") +
    theme_tufte() + ylab("Frequency of Stem") + xlab("Top 30 Used Stem")
p
```



Final figure 15 - included interactivity with plotly.

```
pp <- plot_ly(subset(wf, freq > 50), x = ~freq, y = ~reorder(word, freq), type = "bar",
    orientation = "h") %>% layout(title = "Word Frequency from Articles on ACA, 2011-2017",
    xaxis = list(title = "Top 30 Used Stem"), yaxis = list(title = "Frequency of Stem"),
    margin = list(l = 120, r = 10, t = 80, b = 80))
pp
```

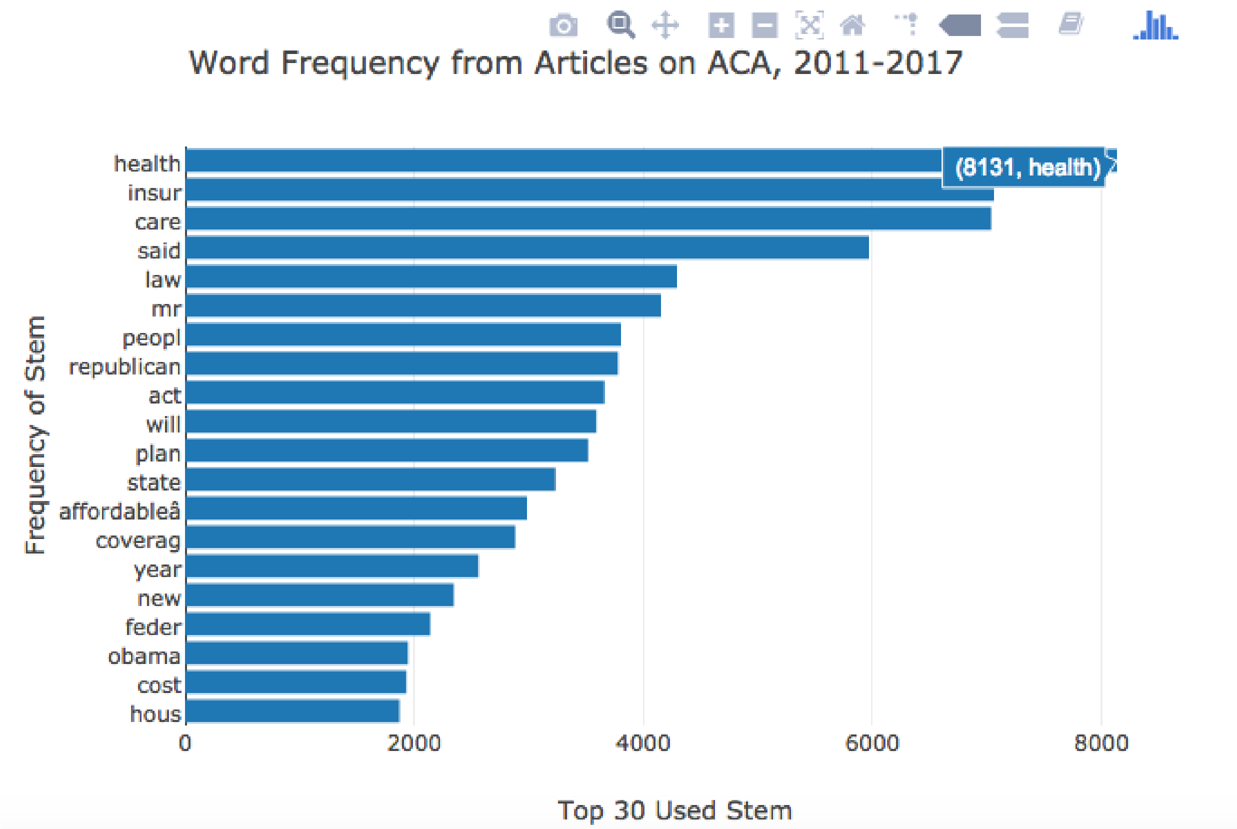


Figure 22:

Figure 16

```
load("nytimes.rda")
NYT_source <- VectorSource(corpus$documents[, 1])
NYT <- VCorpus(NYT_source)
documents <- corpus$documents
removeNumPunct <- function(x) {
    gsub("[^[:alpha:][:space:]]*", "", x)
}
clean_corpus <- function(corpus) {
    corpus <- tm_map(corpus, removePunctuation)
    corpus <- tm_map(corpus, content_transformer(tolower))
    corpus <- tm_map(corpus, content_transformer(replace_symbol))
    corpus <- tm_map(corpus, removeWords, c(stopwords("english"), "will", "can"))
    # We could add more stop words as above
    corpus <- tm_map(corpus, stripWhitespace)
    corpus <- tm_map(corpus, removeNumbers)
    corpus <- tm_map(corpus, content_transformer(removeNumPunct))
    return(corpus)
}
NYT_clean <- clean_corpus(NYT)
NYT_stem <- tm_map(NYT_clean, stemDocument)
meta(NYT_stem, type = "local", tag = "author") <- documents$author
NYT_dtm <- DocumentTermMatrix(NYT_stem)
NYT_tdm <- TermDocumentMatrix(NYT_stem)

NYT_tdm2 <- tidy(NYT_tdm)
NYT_dtm2 <- tidy(NYT_dtm)
NYT_tidy <- tidy(NYT_stem)
NYT_tdm2 <- merge(NYT_tdm2, NYT_tidy, by.x = "document", by.y = "id", all.x = TRUE)
NYT_dtm2 <- merge(NYT_dtm2, NYT_tidy, by.x = "document", by.y = "id", all.x = TRUE)
```

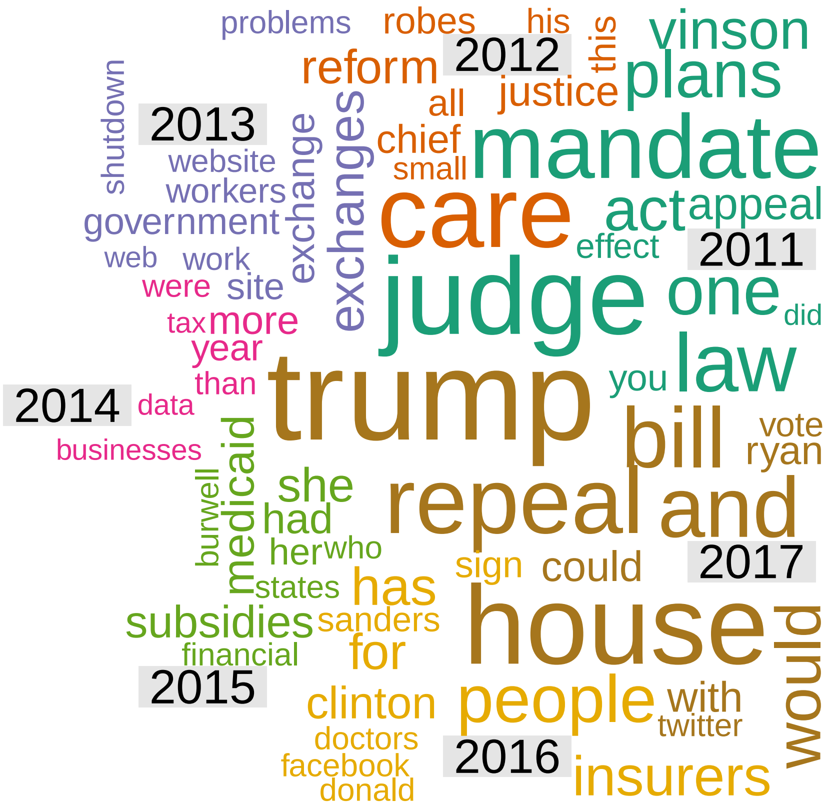
```
atxt <- documents$texts[documents$datetimestamp == "2011"]
btxt <- documents$texts[documents$datetimestamp == "2012"]
ctxt <- documents$texts[documents$datetimestamp == "2013"]
dtxt <- documents$texts[documents$datetimestamp == "2014"]
etxt <- documents$texts[documents$datetimestamp == "2015"]
ftxt <- documents$texts[documents$datetimestamp == "2016"]
gtxt <- documents$texts[documents$datetimestamp == "2017"]

clean.text <- function(x) {
    # tolower
    x = tolower(x)
    # remove rt
    x = gsub("rt", "", x)
    # remove at
    x = gsub("@\\w+", "", x)
    # remove punctuation
    x = gsub("[[:punct:]]", "", x)
    # remove numbers
    x = gsub("[[:digit:]]", "", x)
    # remove links http
    x = gsub("http\\w+", "", x)
    # remove tabs
    x = gsub("[ |\t]{2,}", "", x)
```

```r
    # remove blank spaces at the beginning
    x = gsub("^ ", "", x)
    # remove blank spaces at the end
    x = gsub(" $", "", x)
    return(x)
}

aclean <- clean.text(atxt)
bclean <- clean.text(btxt)
cclean <- clean.text(ctxt)
dclean <- clean.text(dtxt)
eclean <- clean.text(etxt)
fclean <- clean.text(ftxt)
gclean <- clean.text(gtxt)

a <- paste(aclean, collapse = " ")
b <- paste(bclean, collapse = " ")
c <- paste(cclean, collapse = " ")
d <- paste(dclean, collapse = " ")
e <- paste(eclean, collapse = " ")
f <- paste(fclean, collapse = " ")
g <- paste(gclean, collapse = " ")

# put everything in a single vector
all <- c(a, b, c, d, e, f, g)
all <- removeWords(all, c("will", "can", "the", "that", "are", "mrs", "not",
    "said", "cou"))
```

Wordcloud by Year: In the beginning, the articles mainly focused on the law issues as we can see from the frequently used words such as federal, act, mandate, judge, reform, etc. As time passed, the topics mainly focused on financial issues, the words that articles frequently mentioned are workers, business, tax, financial, subsidies, medicaid, etc. In the latest two years, due to the presidential election, the topics are changed to political concerns. For example, the words like people, trump, republicans, house, and repeal are frequently used in the articles.

Final figure 16:

```r
# create corpus
corpus2 <- Corpus(VectorSource(all))
# create term-document matrix
cloudtdm <- TermDocumentMatrix(corpus2)
# convert as matrix
cloudtdm <- as.matrix(cloudtdm)
# add column names
colnames(cloudtdm) <- c("2011", "2012", "2013", "2014", "2015", "2016", "2017")
comparison.cloud(cloudtdm, random.order = FALSE, colors = brewer.pal(8, "Dark2"),
    title.size = 1.5, max.words = 200)
```


2011
vinson
plans
mandate
act
appeal
effect
judge
one
did
law
you
2012
robes
his
this
reform
justice
all
chief
small
care
2013
problems
shutdown
website
workers
government
exchange
exchanges
web
work
site
2014
were
tax
more
year
than
data
businesses
2015
medicaid
burwell
she
had
her
who
states
subsidies
financial
2016
has
sign
sanders
for
clinton
people
doctors
facebook
donald
insurers
2017
trump
bill
vote
ryan
repeal
and
could
house
would
with
twitter

Perceptions of ACA - Twitter API Analaysis

```r
library(httr)
```

```
##
## Attaching package: 'httr'
## The following object is masked from 'package:NLP':
##
##     content
## The following object is masked from 'package:plotly':
##
##     config
```

```r
# library(oauth)
library(ROAuth)
library(twitteR)
```

```
##
## Attaching package: 'twitteR'
## The following object is masked from 'package:qdapTools':
##
##     id
## The following objects are masked from 'package:dplyr':
##
##     id, location
## The following object is masked from 'package:plyr':
##
##     id
```

```r
library(RCurl)
```

```
## Loading required package: bitops
##
## Attaching package: 'RCurl'
## The following object is masked from 'package:tidyr':
##
##     complete
```

```r
library(RJSONIO)
library(stringr)
# secretkey myapp <- oauth_app('twitter', key = 'liLn6XJFenGjtvWFwi5LnDS1M',
# secret = 'dsCBm9Kyaeu9GMKlM9xwKl7eKmDn6qsjP31LQtwMGkF6OQdLh6') Get OAuth
# credentials twitter_token <- oauth1.0_token(oauth_endpoints('twitter'),
# myapp)

# Declare Twitter API Credentials
api_key <- "liLn6XJFenGjtvWFwi5LnDS1M"  # From dev.twitter.com
api_secret <- "dsCBm9Kyaeu9GMKlM9xwKl7eKmDn6qsjP31LQtwMGkF6OQdLh6"  # From dev.twitter.com
token <- "772176811455381505-PYuNAEqhHFc02r83WS9Y5dnsZciIY5v"  # From dev.twitter.com
token_secret <- "mgRPwKeHZEw9Y486h2GMtCBxDztPfQLX1Msd5vog1hiwv"  # From dev.twitter.com

# Create Twitter Connection
```

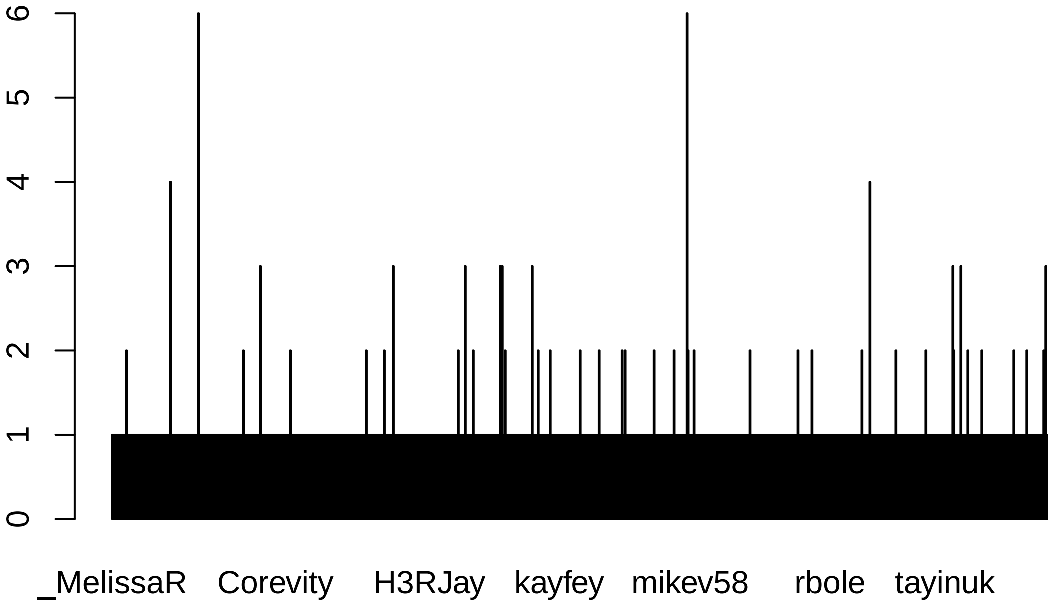
```r
library("base64enc")
setup_twitter_oauth(api_key, api_secret, token, token_secret)
```

```
## [1] "Using direct authentication"
```

```r
# Run Twitter Search. Format is searchTwitter('Search Terms', n=100,
# lang='en', geocode='lat,lng', also accepts since and until).
tweets <- searchTwitter("Obamacare OR ACA OR 'Affordable Care Act' OR #ACA",
    n = 1000, lang = "en", since = "2014-08-20")

# Transform tweets list into a data frame
tweets.df <- twListToDF(tweets)
head(tweets.df, 3)
```

```
##
## 1 Republicans wasted eight yrs. complaining &amp; whipping frenzi over what 76% Majority of American
## 2      I'm grateful my pre-existing condition (SIN) was covered by the Affordable Care Act of JESUS
## 3                                                               How the Affordable Care Act Drove Do
##   favorited favoriteCount replyToSN             created truncated
## 1     FALSE             0      <NA> 2017-05-09 17:14:00      TRUE
## 2     FALSE             0      <NA> 2017-05-09 17:13:20     FALSE
## 3     FALSE             0      <NA> 2017-05-09 17:13:00     FALSE
##   replyToSID                 id replyToUID
## 1       <NA> 861992660822704128       <NA>
## 2       <NA> 861992493352669186       <NA>
## 3       <NA> 861992411777683456       <NA>
##                                                             statusSource
## 1    <a href="http://twitter.com" rel="nofollow">Twitter Web Client</a>
## 2 <a href="http://www.facebook.com/twitter" rel="nofollow">Facebook</a>
## 3              <a href="http://bufferapp.com" rel="nofollow">Buffer</a>
##       screenName retweetCount isRetweet retweeted longitude latitude
## 1       etecbill            0     FALSE     FALSE        NA       NA
## 2 wildlyfejblood            0     FALSE     FALSE        NA       NA
## 3    spediatrics            0     FALSE     FALSE        NA       NA
```

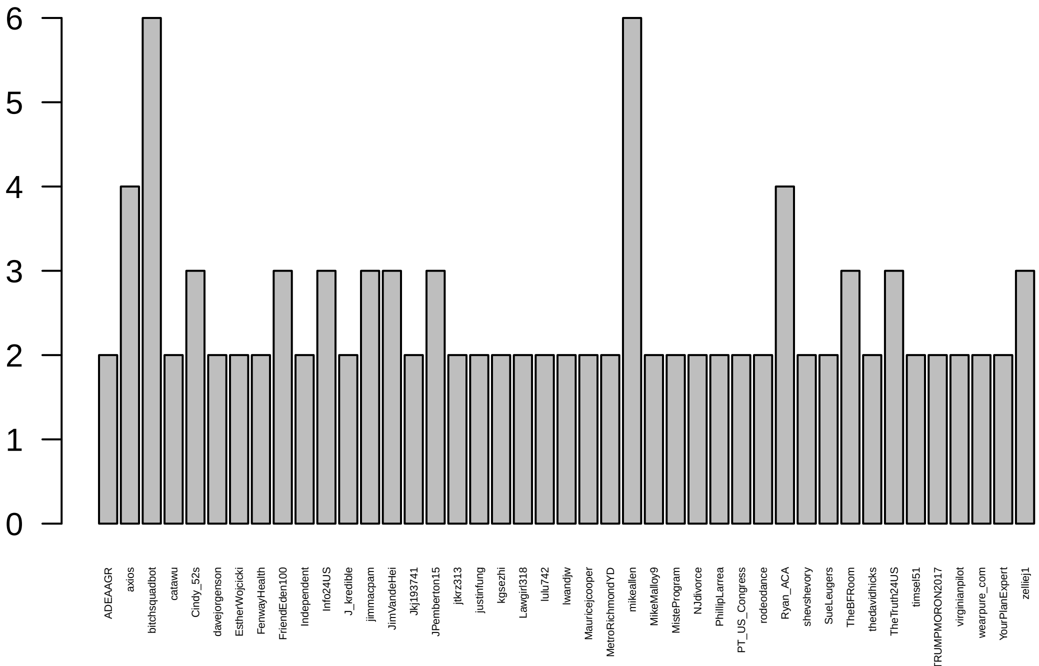
```r
counts = table(tweets.df$screenName)
barplot(counts)
```



Some data exploration

Let's do something hacky: Limit the data set to show only folk who tweeted twice or more in the sample.

```
cc = subset(counts, counts > 1)
barplot(cc, las = 2, cex.names = 0.3)
```



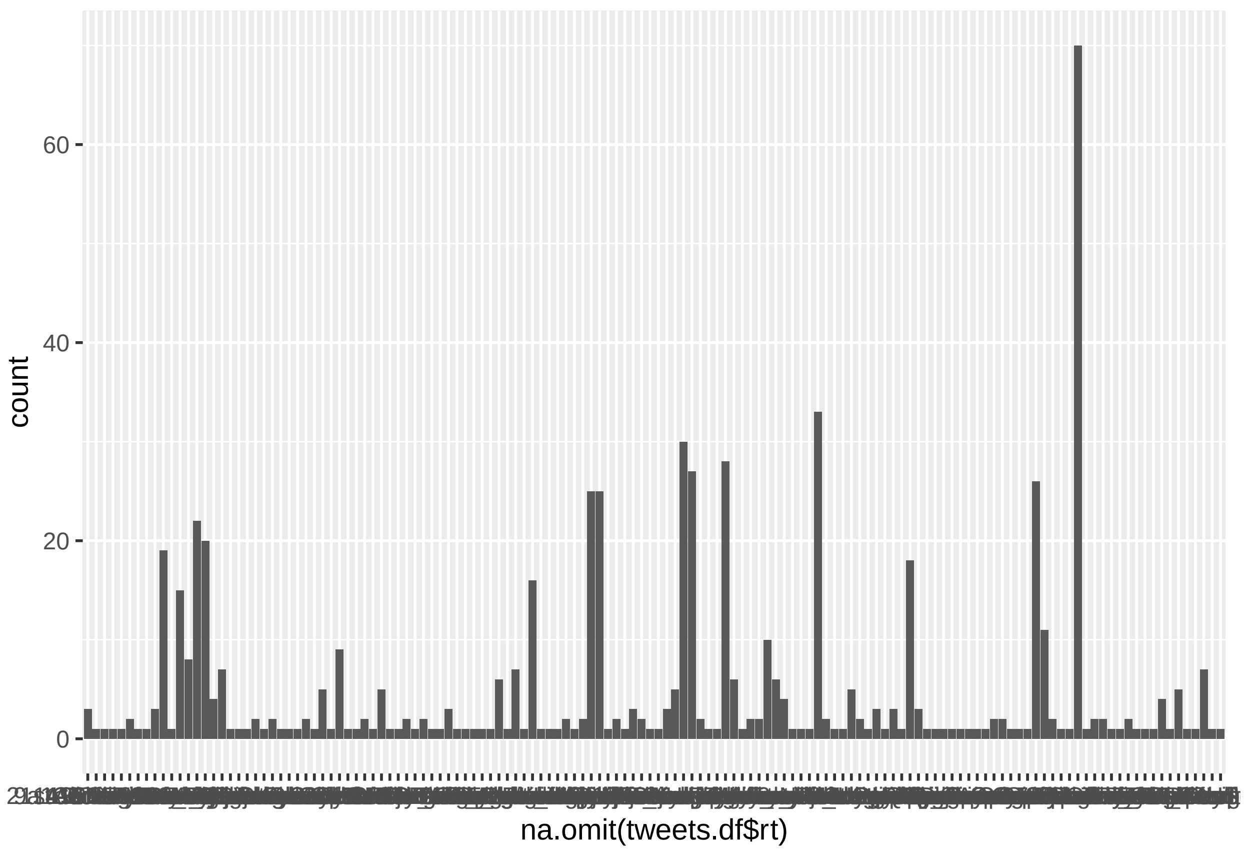

```
tweets.df$text = sapply(tweets.df$text, function(row) iconv(row, to = "UTF-8"))
```

```
# A helper function to remove @ symbols from user names...
trim <- function(x) sub("@", "", x)

# A couple of tweet parsing functions that add columns to the dataframe
library(stringr)
```

```
# Pull out who a message is to
tweets.df$to = sapply(tweets.df$text, function(tweet) str_extract(tweet, "^(@[[:alnum:]_]*)"))
tweets.df$to = sapply(tweets.df$to, function(name) trim(name))

# And here's a way of grabbing who's been RT'd
tweets.df$rt = sapply(tweets.df$text, function(tweet) trim(str_match(tweet,
    "^RT (@[[:alnum:]_]*)")[2]))
```

Now we can plot a chart showing how often a particular person was RT'd in our sample.

```
library(ggplot2)
ggplot() + geom_bar(aes(x = na.omit(tweets.df$rt)))
```




```
library(tidyr)
library(dplyr)
library(purrr)
```

```
##
## Attaching package: 'purrr'

## The following object is masked from 'package:qdap':
##
##     %>%

## The following objects are masked from 'package:dplyr':
##
##     contains, order_by

## The following object is masked from 'package:plyr':
```

```
##
##     compact
## The following object is masked from 'package:magrittr':
##
##     set_names
## The following object is masked from 'package:maps':
##
##     map
```

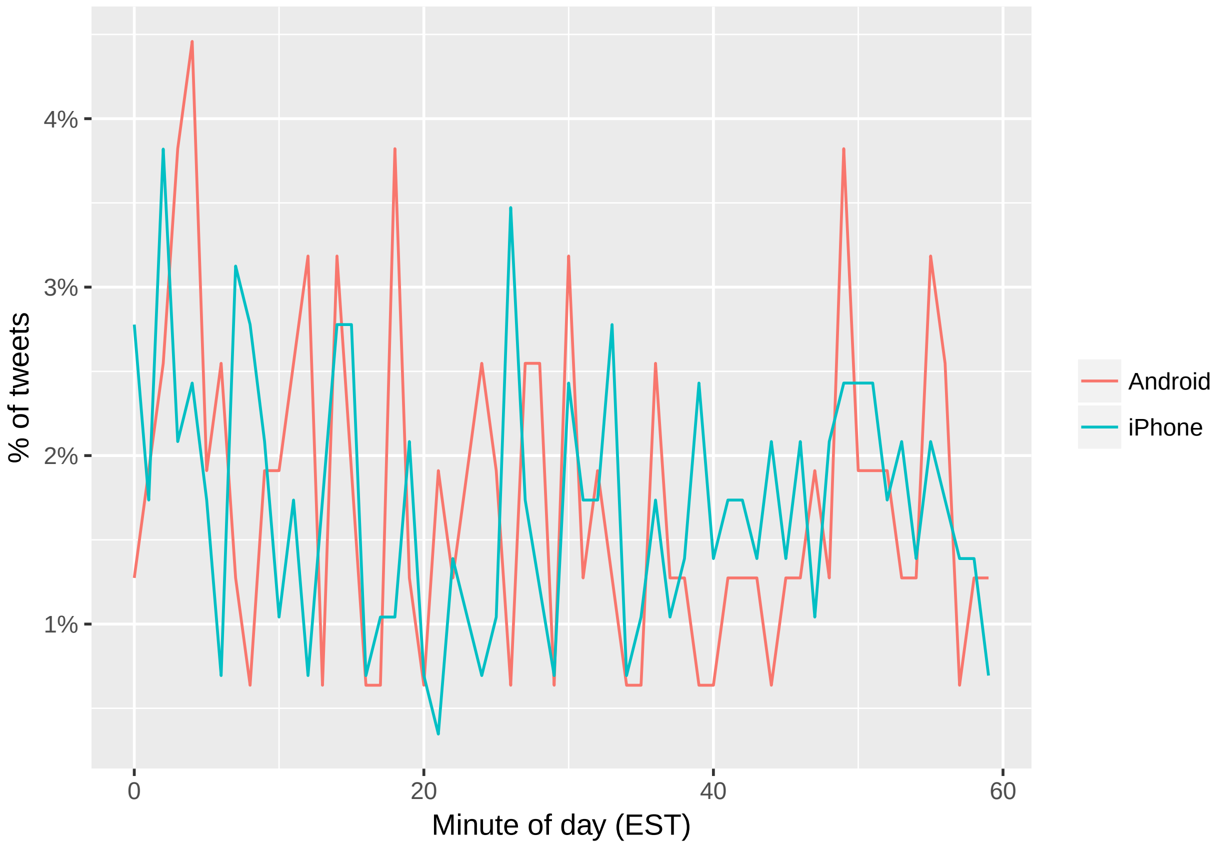
```
tweets <- tweets.df %>% select(id, statusSource, text, created) %>% extract(statusSource,
    "source", "Twitter for (.*?)<") %>% filter(source %in% c("iPhone", "Android"))
table(tweets$source)
```

```
##
## Android  iPhone
##     157     288
```

```
library(lubridate)
library(scales)
```

```
##
## Attaching package: 'scales'
## The following object is masked from 'package:purrr':
##
##     discard
## The following objects are masked from 'package:readr':
##
##     col_factor, col_numeric
```

```
tweets %>% count(source, minute = minute(with_tz(created, "EST"))) %>% mutate(percent = n/sum(n)) %>%
    ggplot(aes(minute, percent, color = source)) + geom_line() + scale_y_continuous(labels = percent_fo
    labs(x = "Minute of day (EST)", y = "% of tweets", color = "")
```




```
tweet_picture_counts <- tweets %>% filter(!str_detect(text, "^\"")) %>% count(source,
    picture = ifelse(str_detect(text, "t.co"), "Picture/link", "No picture/link"))

ggplot(tweet_picture_counts, aes(source, n, fill = picture)) + geom_bar(stat = "identity",
    position = "dodge") + labs(x = "", y = "Number of tweets", fill = "")
```

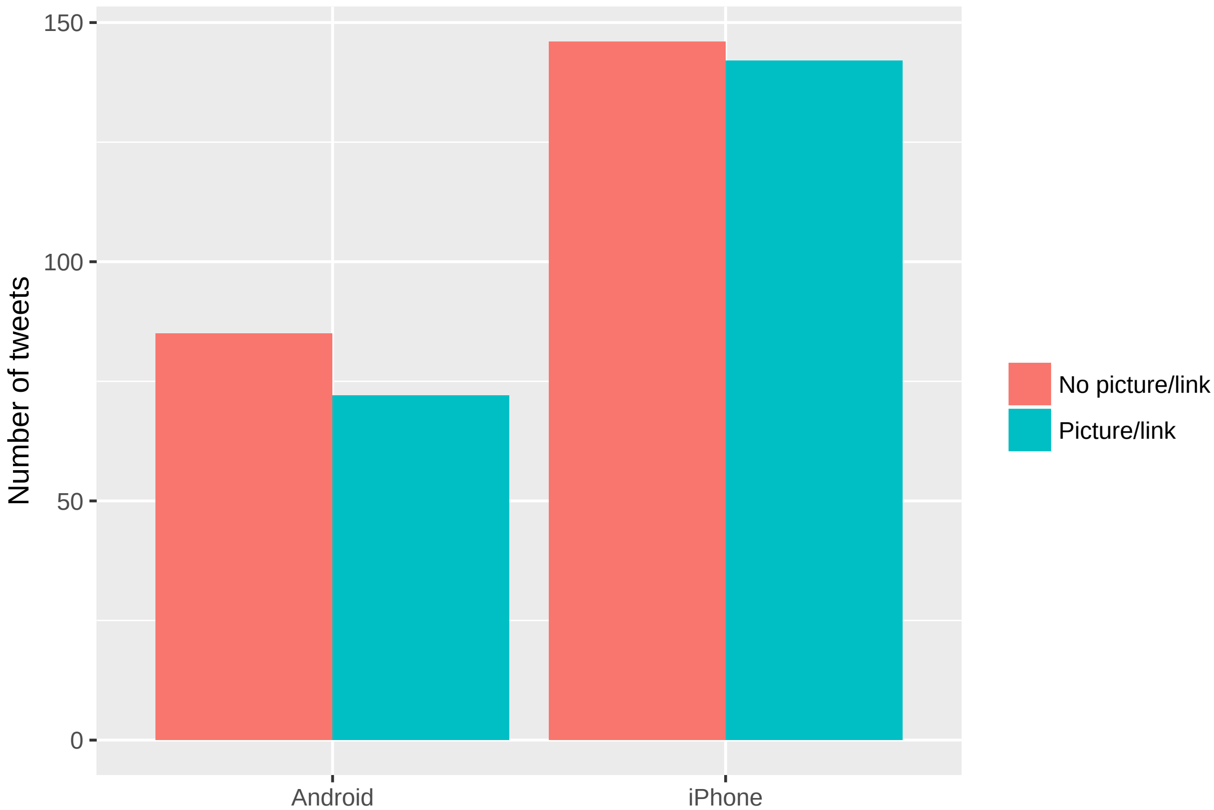
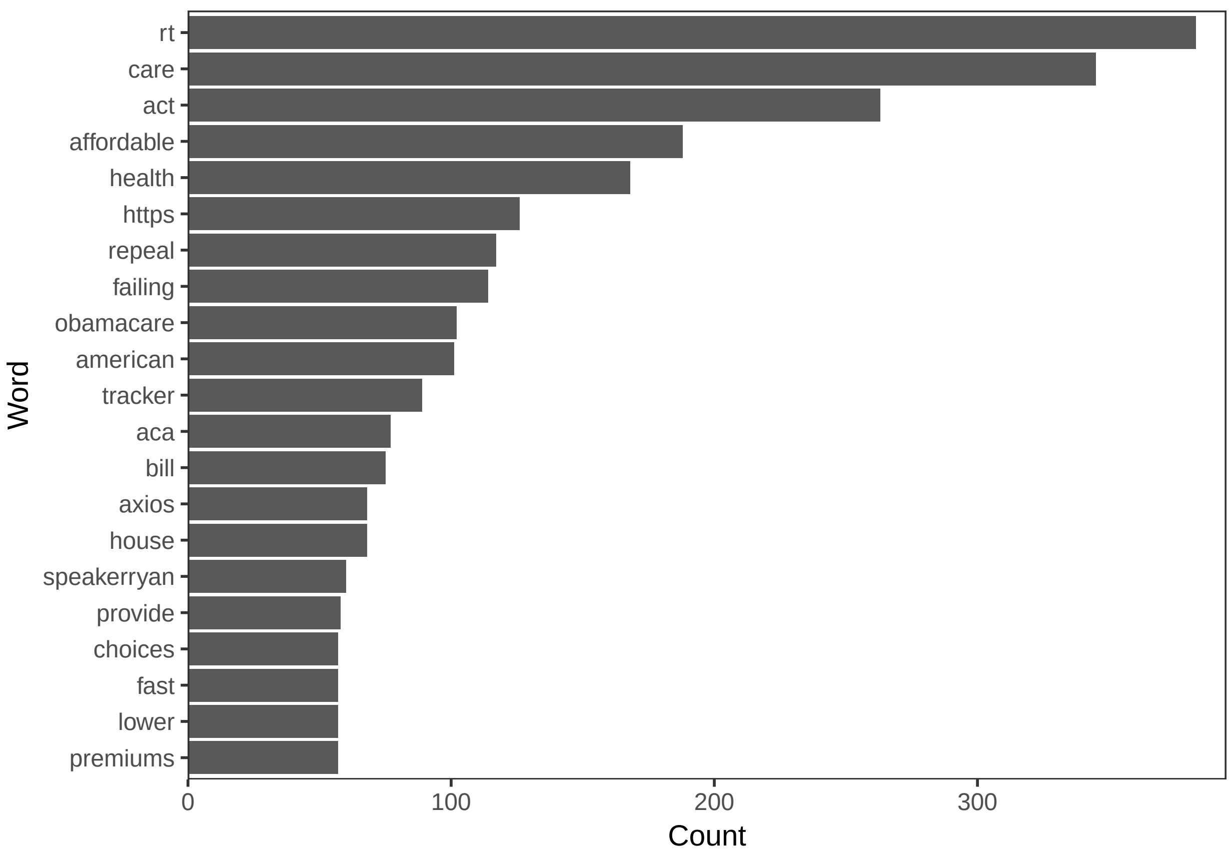

150
100
50
0
Number of tweets
Android
iPhone
No picture/link
Picture/link

Figure 17

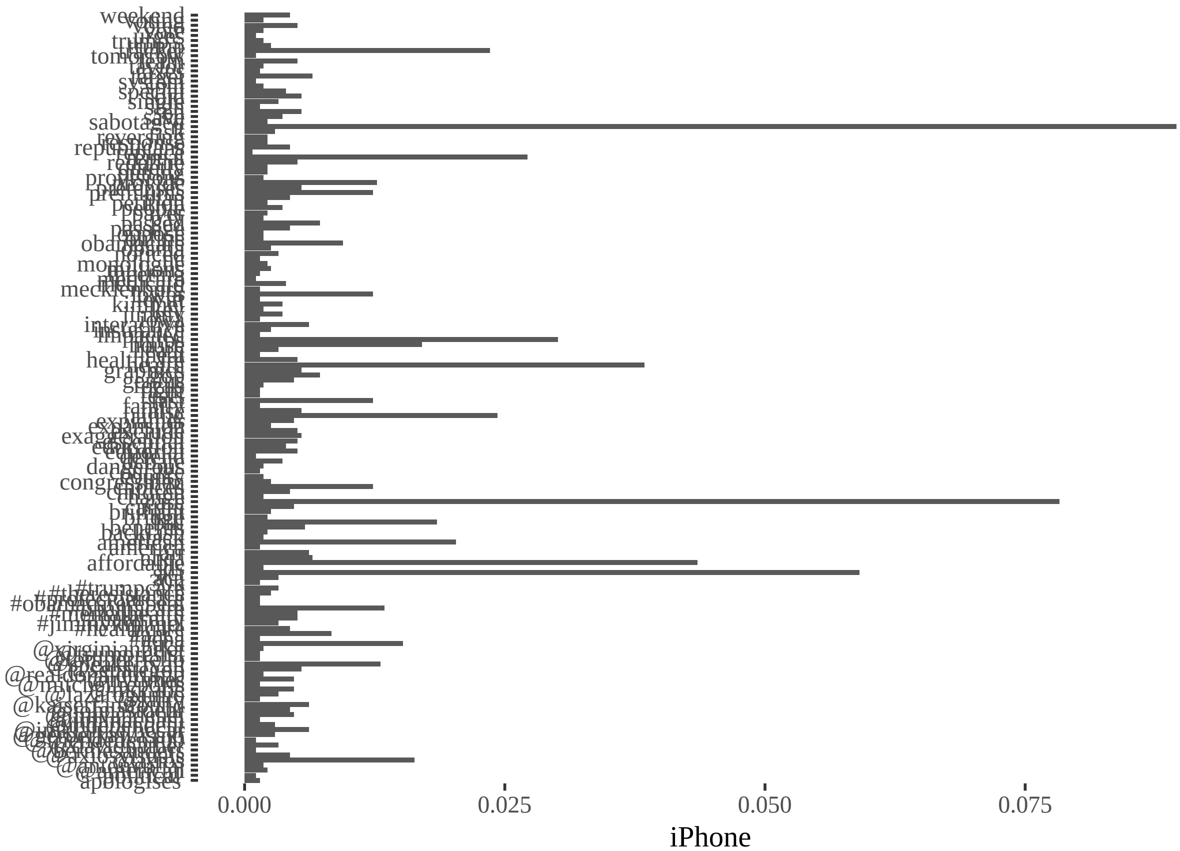
Comparison of words Now that we're sure there's a difference, what can we say about the difference in the content? We'll use the tidytext package. We start by dividing into individual words using the unnest_tokens function, and removing some common stopwords. As we can see the most frequently used word when discussing ACA on Twitter is rt (Republican Party), care, act, afforadable, health, obama, and so on. From the table, we found that most people discussed about political issues (i.e., the topics of republican and democratic parties), affordable care act itself, and finanical concerns (i.e., bill, pay, etc).

```
library(tidytext)
reg <- "([^A-Za-z\\d#@']|'(?![A-Za-z\\d#@]))"
tweet_words <- tweets %>% filter(!str_detect(text, "^\"")) %>% mutate(text = str_replace_all(text,
    "https://t.co/[A-Za-z\\d]+|&amp;", "")) %>% unnest_tokens(word, text, token = "regex",
    pattern = reg) %>% filter(!word %in% stop_words$word, str_detect(word, "[a-z]"))
library(qdap)
# Find the 20 most frequent terms: term_count
term_count1 <- freq_terms(tweet_words$word, 20)
# Plot term_count
plot(term_count1, main = "Frequently used words on Twitter regarding ACA")
```




```
android_iphone_ratios <- tweet_words %>% count(word, source) %>% filter(sum(n) >=
    5) %>% spread(source, n, fill = 0) %>% ungroup() %>% mutate_each(funs((. +
    1)/sum(. + 1)), -word) %>% mutate(logratio = log2(Android/iPhone)) %>% arrange(desc(logratio))
```

```
ggplot(data = android_iphone_ratios, aes(x = word, y = iPhone)) + geom_bar(stat = "identity") +
    # geom_text(label= Android, color='red') +
xlab(NULL) + coord_flip() + theme_tufte()
```



Sentiment analysis: Since we've observed a difference in sentiment between the Android and iPhone tweets, let's try quantifying it. We'll work with the NRC Word-Emotion Association lexicon, available from the tidytext package, which associates words with 10 sentiments: positive, negative, anger, anticipation, disgust, fear, joy, sadness, surprise, and trust.

To measure the sentiment of the Android and iPhone tweets, we can count the number of words in each category. (For example, we see that 41 of the 2331 words in the Android tweets were associated with "anger"). We then want to measure how much more likely the Android account is to use an emotionally-charged term relative to the iPhone account. Since this is count data, we can use a Poisson test to measure the difference:

```
nrc <- sentiments %>% filter(lexicon == "nrc") %>% dplyr::select(word, sentiment)

sources <- tweet_words %>% group_by(source) %>% mutate(total_words = n()) %>%
    ungroup() %>% distinct(id, source, total_words)

by_source_sentiment <- tweet_words %>% inner_join(nrc, by = "word") %>% count(sentiment,
    id) %>% ungroup() %>% # complete(sentiment, id, fill = list(n = 0)) %>%
inner_join(sources) %>% group_by(source, sentiment, total_words) %>% summarize(words = sum(n)) %>%
    ungroup()
```

```
## Joining, by = "id"
```

```
head(by_source_sentiment)
```

```
## # A tibble: 6 × 4
##    source    sentiment total_words words
##     <chr>        <chr>       <int> <int>
## 1 Android        anger        1812    66
## 2 Android anticipation        1812    89
```

```
## 3 Android      disgust        1812     8
## 4 Android         fear        1812    81
## 5 Android          joy        1812    30
## 6 Android     negative        1812   113
```

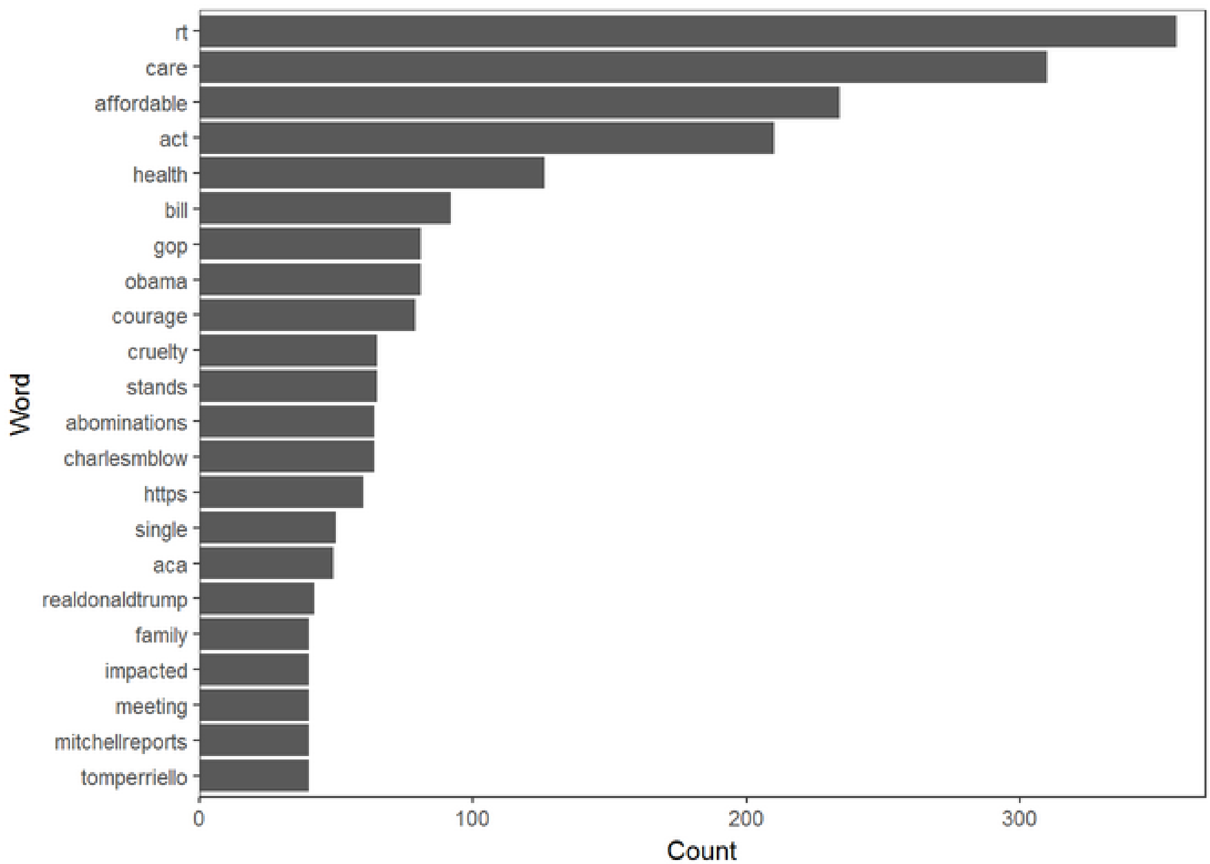
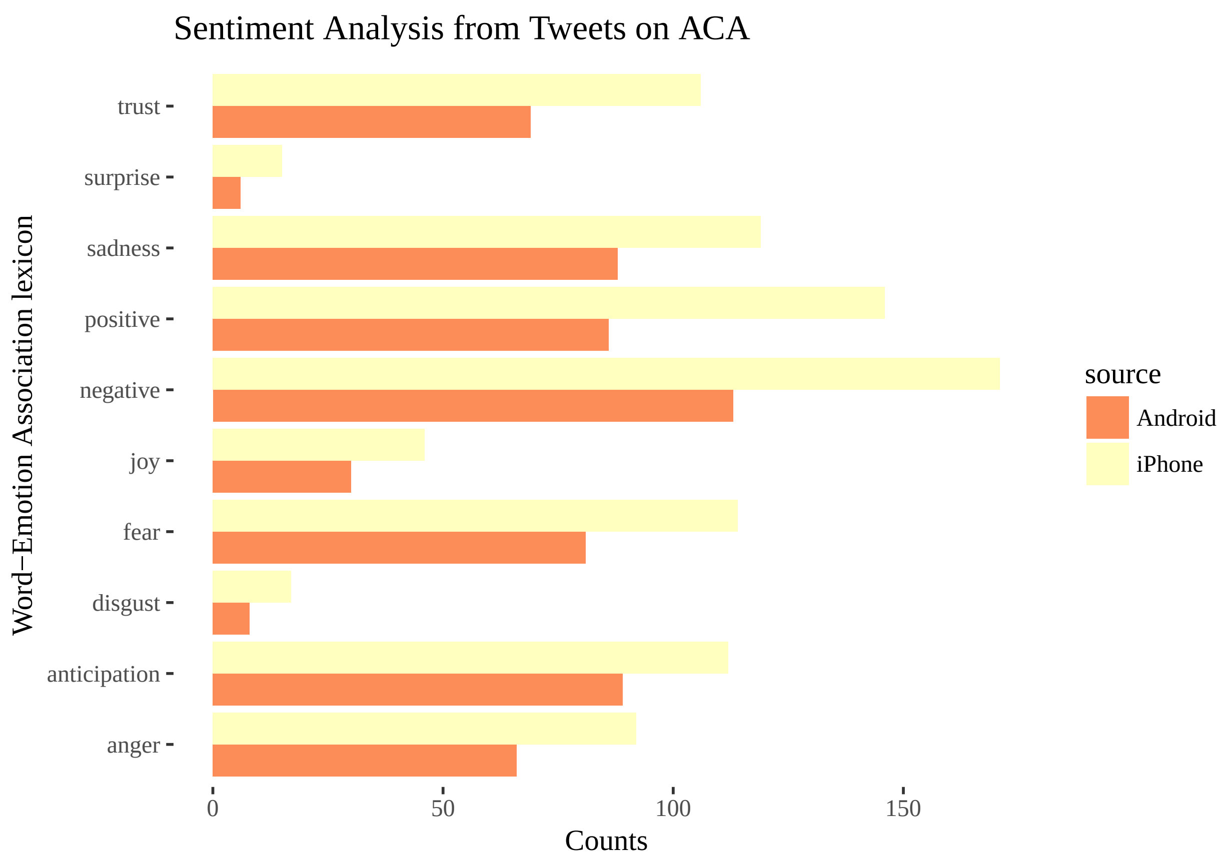


Figure 23:

Figure 18

From the below table, we could see that most people feel quite positive about ACA, the words they used are expressed their trust, anticipation, joy and positive feeling.

```
sentiment_differences <- by_source_sentiment %>% group_by(sentiment) %>% do(tidy(poisson.test(.$words,
    .$total_words)))
# sentiment_differences library(reshape2) df.long <-
# melt(by_source_sentiment) df.long <- df.long[21:40,]
ggplot(by_source_sentiment, aes(x = sentiment, y = words)) + geom_bar(aes(fill = source),
    stat = "identity", position = "dodge") + scale_fill_brewer(palette = "Spectral") +
    coord_flip() + theme(axis.text.x = element_text(angle = 45, hjust = 1),
    plot.title = element_text(face = "bold", hjust = 0.5, size = 12)) + ggtitle("Sentiment Analysis fro
    theme_tufte() + ylab("Counts") + xlab("Word-Emotion Association lexicon")
```


Sentiment Analysis from Tweets on ACA
trust
surprise
sadness
positive
negative
joy
fear
disgust
anticipation
anger
Word−Emotion Association lexicon
0
50
100
150
Counts
source
Android
iPhone

Data Preparation using Twitter: The Twitter search API does not return an exhaustive list of tweets that match your search criteria, as Twitter only makes available a sample of recent tweets. For a more comprehensive search, we will need to use the Twitter streaming API, creating a database of results and regularly updating them, or use an online service that can do this. Now that we have tweet texts, we need to clean them up before doing any analysis. This involves removing content, such as punctuation, that has no emotional content, and removing any content that causes errors.

```r
#text cleaning
library(tm)
# build a corpus, and specify the source to be character vectors
myCorpus <- Corpus(VectorSource(tweets.df$text))
# convert to lower case
# tm v0.6
myCorpus <- tm_map(myCorpus, content_transformer(tolower))
# tm v0.5-10
# myCorpus <- tm_map(myCorpus, tolower)
# remove URLs
removeURL <- function(x) gsub("http[^[:space:]]*", "", x)
# tm v0.6
myCorpus <- tm_map(myCorpus, content_transformer(removeURL))
# tm v0.5-10
# myCorpus <- tm_map(myCorpus, removeURL)
# remove anything other than English letters or space
removeNumPunct <- function(x) gsub("[^[:alpha:][:space:]]*", "", x)
myCorpus <- tm_map(myCorpus, content_transformer(removeNumPunct))
# remove punctuation
# myCorpus <- tm_map(myCorpus, removePunctuation)
# remove numbers
# myCorpus <- tm_map(myCorpus, removeNumbers)
# add two extra stop words: "available" and "via"
myStopwords <- c(stopwords('english'), "available", "via")
# remove "r" and "big" from stopwords
myStopwords <- setdiff(myStopwords, c("r", "big"))
# remove stopwords from corpus
myCorpus <- tm_map(myCorpus, removeWords, myStopwords)
# remove extra whitespace
myCorpus <- tm_map(myCorpus, stripWhitespace)
# keep a copy of corpus to use later as a dictionary for stem completion
myCorpusCopy <- myCorpus
# stem words
myCorpus <- tm_map(myCorpus, stemDocument)
# inspect the first 5 documents (tweets)
# inspect(myCorpus[1:5])
# The code below is used for to make text fit for paper width
for (i in c(1:2, 320)) {
cat(paste0("[", i, "] ")) writeLines(strwrap(as.character(myCorpus[[i]]), 60))
}
## [1] exampl call java code r
## [2] simul mapreduc r big data analysi use flight data rblogger
## [320] r refer card data mine now cran list mani use r function
## packag data mine applic
# tm v0.5-10
# myCorpus <- tm_map(myCorpus, stemCompletion)
# tm v0.6
```

```r
stemCompletion2 <- function(x, dictionary) {
x <- unlist(strsplit(as.character(x), " "))
# Unexpectedly, stemCompletion completes an empty string to
# a word in dictionary. Remove empty string to avoid above issue.
x <- x[x != ""]
x <- stemCompletion(x, dictionary=dictionary)
x <- paste(x, sep="", collapse=" ") PlainTextDocument(stripWhitespace(x))
}
#myCorpus <- lapply(myCorpus, stemCompletion2, dictionary=myCorpusCopy) myCorpus <- Corpus(VectorSource
# count frequency of "mining"
miningCases <- lapply(myCorpusCopy,
function(x) { grep(as.character(x), pattern = "\\<mining")} )
sum(unlist(miningCases))
## [1] 82
# count frequency of "miner"
minerCases <- lapply(myCorpusCopy,
function(x) {grep(as.character(x), pattern = "\\<miner")} )
sum(unlist(minerCases))
## [1] 5
# replace "miner" with "mining"
myCorpus <- tm_map(myCorpus, content_transformer(gsub),
pattern = "miner", replacement = "mining")
tdm <- TermDocumentMatrix(myCorpus,
control = list(wordLengths = c(1, Inf)))
tdm
## <<TermDocumentMatrix (terms: 822, documents: 320)>>
## Non-/sparse entries: 2460/260580
## Sparsity : 99%
## Maximal term length: 27
## Weighting : term frequency (tf)
#__figure 17___
#Frequent Words and Associations
term.freq <- rowSums(as.matrix(tdm))
term.freq <- subset(term.freq, term.freq >= 15)
df <- data.frame(term = names(term.freq), freq = term.freq)


ggplot(df, aes(x = term, y = freq)) + geom_bar(stat = "identity") +
xlab("Terms") + ylab("Count") + coord_flip()
```

Figure 19

Unsurprisingly, Word Cloud shows that most frequently mentioned terms are health, care, affordable, aca, and act. Except the discussion of insurance policy itself, people do talk frequently about political-related topics, such as Republican Party (rt), Grand Old Party (gop), and Obama. Also, some terms related to push forward or hold back the policy, such as courage, urge and repeal. Financial words are used, like pay, bill and save. Finally, specific names of people and place were mentioned in the tweets (i.e., Larry Levitt and Mecklenburg).

```
# library(graph) source('http://bioconductor.org/biocLite.R')
# biocLite('Rgraphviz') library('Rgraphviz') plot(tdm, term = term.freq,
# corThreshold = 0.2, weighting = T)
m <- as.matrix(tdm)
# calculate the frequency of words and sort it by frequency
word.freq <- sort(rowSums(m), decreasing = T)
# colors
library(RColorBrewer)
pal <- brewer.pal(9, "BuGn")
pal <- pal[-(1:4)]
# plot word cloud
library(wordcloud)
wordcloud(words = names(word.freq), freq = word.freq, min.freq = 3, random.order = F,
    colors = pal)
```

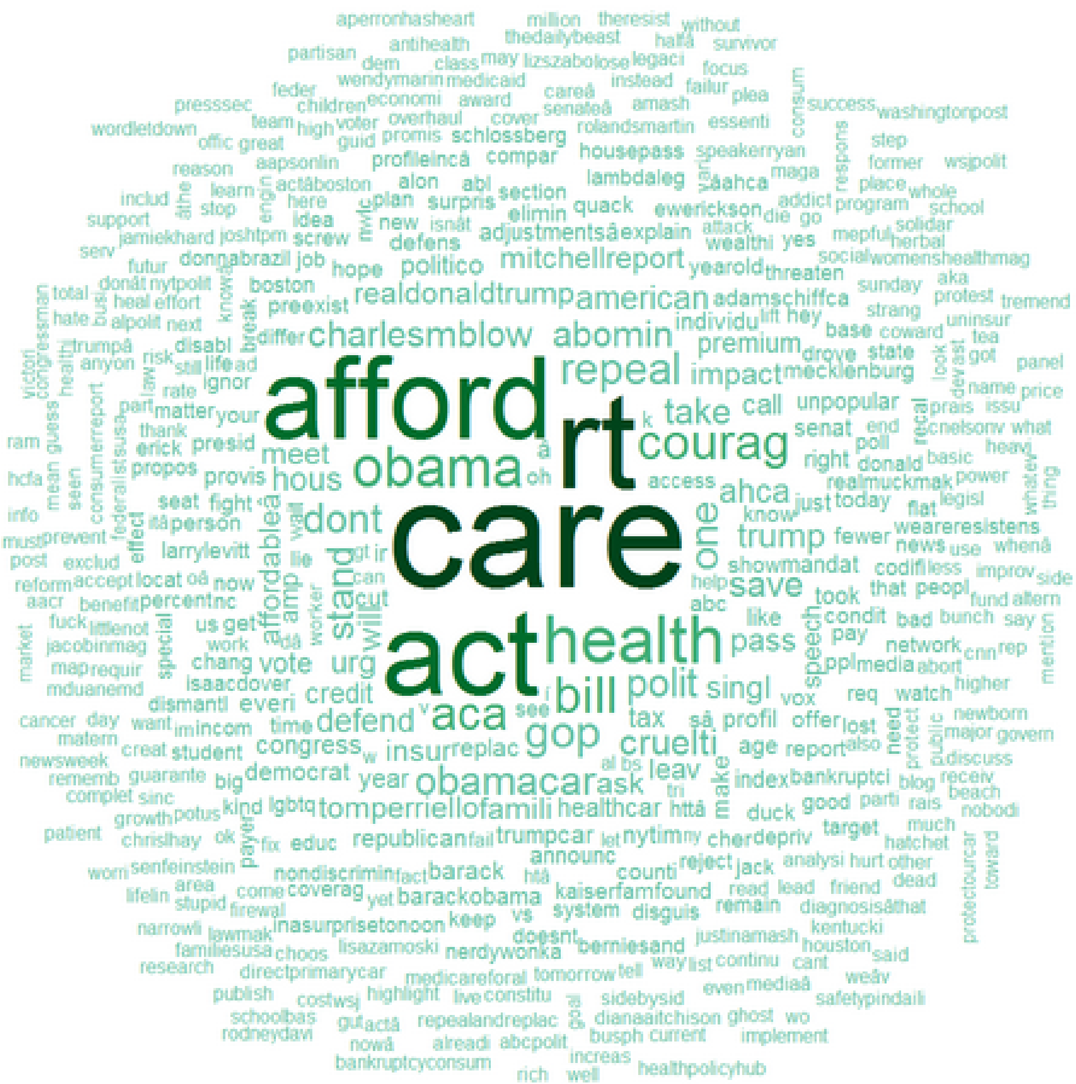

care
rt
afford
act
health
obama
courag
repeal
bill
aca
gop
obamacar
charlesmblow
abomin
realdonaldtrump
american
mitchellreport
dont
stand
trump
save
cruelti
polit
take
ahca
one
hous
meet
defend
tomperriellofamili
pass
singl
impact
premium
call
ask
insur
urg
vote
credit
congress
democrat
republican
barackobama

Figure 24:

Figure 20

Topic Modelling

```
dtm <- as.DocumentTermMatrix(tdm)
library(topicmodels)
lda <- LDA(dtm, k = 8)  # find 8 topics
(term <- terms(lda, 6))  # first 6 terms of every topic
chapter_topics <- tidy(lda, matrix = "beta")
chapter_topics
top_terms <- chapter_topics %>% group_by(topic) %>% top_n(5, beta) %>% ungroup() %>%
    arrange(topic, -beta)
top_terms
```

n text mining, we often have collections of documents, such as social media posts or news articles, that we'd like to divide into natural groups so that we can understand them separately. Topic modeling is a method for unsupervised classification of such documents, similar to clustering on numeric data, which finds natural groups of items even when we're not sure what we're looking for.

Latent Dirichlet allocation (LDA) is a particularly popular method for fitting a topic model. It treats each document as a mixture of topics, and each topic as a mixture of words. This allows documents to "overlap" each other in terms of content, rather than being separated into discrete groups, in a way that mirrors typical use of natural language.

Latent Dirichlet allocation is one of the most common algorithms for topic modeling. Without diving into the math behind the model, we can understand it as being guided by two principles: Every document is a mixture of topics and Every topic is a mixture of words.

This visualization lets us understand the eight topics that were extracted from the tweets. The most common words in topic 1 include "rt", "care", and "health", which suggests it may represent health insurance and republican issues. Those most common in topic 2 include "obama", "act", and "bill", suggeting that this topic represents issues related to obamacare. One important observation about the words in each topic is that some words, such as "act" are common within both topics. This is an advantage of topic modeling as opposed to "hard clustering" methods: topics used in natural language could have some overlap in terms of words.

```
library(ggplot2)
top_terms %>% mutate(term = reorder(term, beta)) %>% ggplot(aes(term, beta,
    fill = factor(topic))) + geom_col(show.legend = FALSE) + facet_wrap(~topic,
    scales = "free") + coord_flip()
```

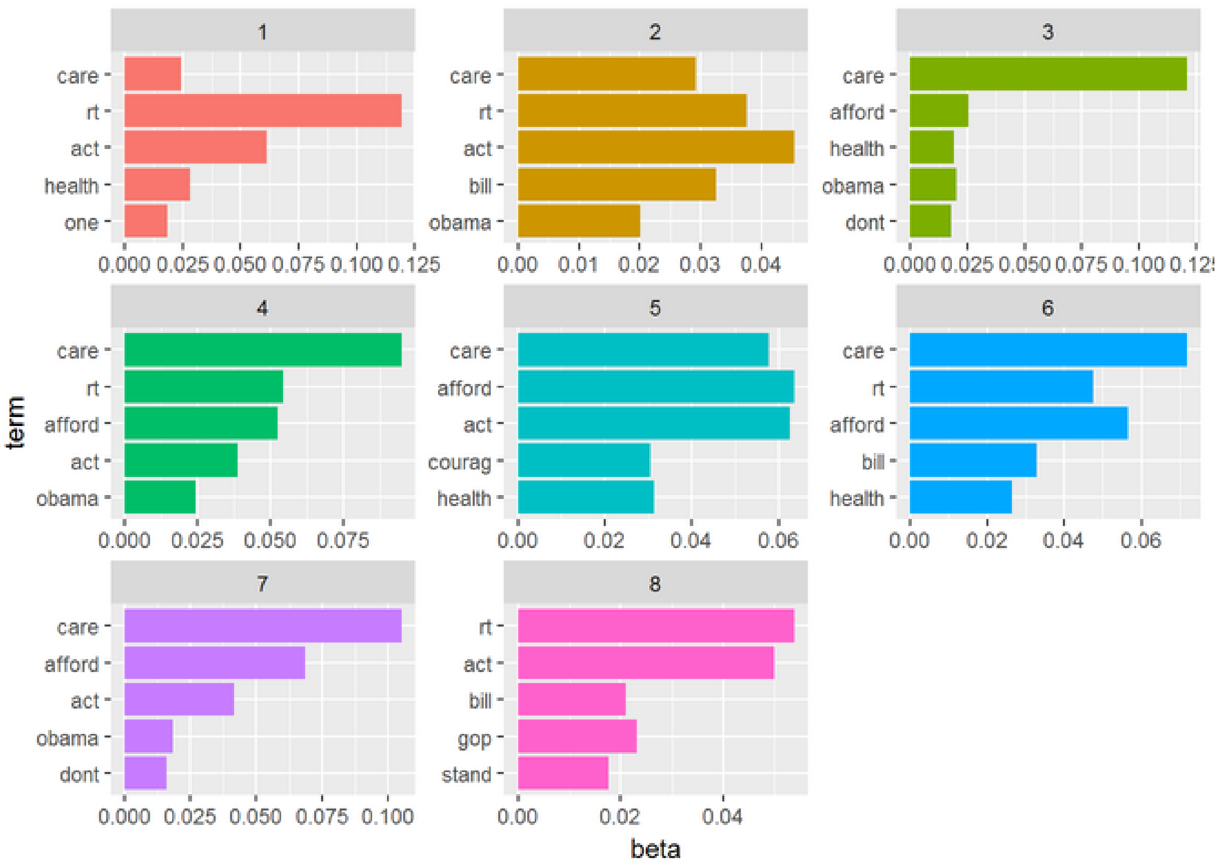

1
care
rt
act
health
one
0.000 0.025 0.050 0.075 0.100 0.125
2
care
rt
act
bill
obama
0.00 0.01 0.02 0.03 0.04
3
care
afford
health
obama
dont
4
care
rt
afford
act
obama
0.000 0.025 0.050 0.075
5
care
afford
act
courag
health
0.00 0.02 0.04 0.06
6
care
rt
afford
bill
health
0.00 0.02 0.04 0.06
7
care
afford
act
obama
dont
0.000 0.025 0.050 0.075 0.100
8
rt
act
bill
gop
stand
0.00 0.02 0.04
term
beta

Figure 25:

Next Steps

We want to do more for the text analysis part, maybe include some sentiment analysis. And for the New York Times analysis, when we convert the articles into metadata, we abandoned many variables, which also limit the possibility to explore some possible relationship, so we might try to do some text analysis while control other variables. As we were not able to download all the tweets we didn't finish the text analysis based on tweets.

One further thing we would like to explore if we had full access to the tweets during the years, is to find out how perceptions of the ACA differed by state. In particular, we can investigate if there are party and regional differences, which would link up well with the earlier section of our project. Given tweets in this full time range, we would have liked to explore how these sentiments changed over time, especially in response to significant policy and political events.