Data Wrangling HW 9 and 10

Yaniv Bronshtein

4/19/2021

Import the necessary libraries

```
## Warning: package 'sna' was built under R version 4.0.5

## Loading required package: statnet.common

## Warning: package 'statnet.common' was built under R version 4.0.5

##
## Attaching package: 'statnet.common'

## The following object is masked from 'package:base':
##
##     order

## Loading required package: network

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

## network: Classes for Relational Data
## Version 1.16.1 created on 2020-10-06.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##                     Mark S. Handcock, University of California -- Los Angeles
##                     David R. Hunter, Penn State University
##                     Martina Morris, University of Washington
##                     Skye Bender-deMoll, University of Washington
##  For citation information, type citation("network").
##  Type help("network-package") to get started.

## sna: Tools for Social Network Analysis
## Version 2.6 created on 2020-10-5.
## copyright (c) 2005, Carter T. Butts, University of California-Irvine
##  For citation information, type citation("sna").
##  Type help(package="sna") to get started.

## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.3     v purrr   0.3.4
## v tibble  3.0.5     v dplyr   1.0.3
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.4.0     v forcats 0.5.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

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

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union

## The following objects are masked from 'package:purrr':
##
##     compose, simplify

## The following object is masked from 'package:tidyr':
##
##     crossing

## The following object is masked from 'package:tibble':
##
##     as_data_frame

## The following objects are masked from 'package:sna':
##
##     betweenness, bonpow, closeness, components, degree, dyad.census,
##     evcent, hierarchy, is.connected, neighborhood, triad.census

## The following objects are masked from 'package:network':
##
##     %c%, %s%, add.edges, add.vertices, delete.edges, delete.vertices,
##     get.edge.attribute, get.edges, get.vertex.attribute, is.bipartite,
##     is.directed, list.edge.attributes, list.vertex.attributes,
##     set.edge.attribute, set.vertex.attribute

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union

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

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

## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
## Loading required package: xml2

##
## Attaching package: 'rvest'

## The following object is masked from 'package:purrr':
##
##     pluck

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

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

## Loading required package: acs

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

## Loading required package: XML

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

##
## Attaching package: 'XML'

## The following object is masked from 'package:rvest':
##
##     xml

##
## Attaching package: 'acs'

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

## The following object is masked from 'package:base':
##
##     apply

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

## Loading required package: lattice

## Loading required package: survival

## Loading required package: Formula
```

```
##
## Attaching package: 'Hmisc'

## The following object is masked from 'package:rvest':
##
##     html

## The following objects are masked from 'package:dplyr':
##
##     src, summarize

## The following object is masked from 'package:network':
##
##     is.discrete

## The following objects are masked from 'package:base':
##
##     format.pval, units

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

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

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:acs':
##
##     combine

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

Problem 1

1.Generate an undirected random graph with 20 vertices and probability of a tie (forming an edge) equal 0.7. Calculate the mean degree from these graphs. Call it MD20. 2.Now, keeping the probability of a tie fixed at 0.7, repeat the above step with 30, 40, 50, 60, 70, 80, 90 and 100 vertices, by writing a simple R function. In each case calculate the average mean degree. Denote them by MD30, . . . , MD 100. 3.Plot the average mean degrees against the number of vertices. #Is the plot result consistent with your expectation? Explain.

1.1 Generate MD20

```
set.seed(1)
net <- rgraph(n = 20, tprob = 0.7, mode = "graph")
MD20 <- mean(sna::degree(net, gmode = "graph"))
MD20
```

```
## [1] 14.4
```

1.2 repeat the above step with 30, 40, 50, 60, 70, 80, 90 and 100 vertices, by writing a simple R function.

```
get_mean_degree <- function(num_vertices) {
  set.seed(1)
  net <- rgraph(n = num_vertices, tprob = 0.7, mode = "graph")
  return(mean(sna::degree(net, gmode = "graph")))
}
```

Call the function

```
mean_list <- NULL
for (i in seq(from = 20, to = 100, by = 10)) {
  mean_list <- c(mean_list, get_mean_degree(i))
}
```

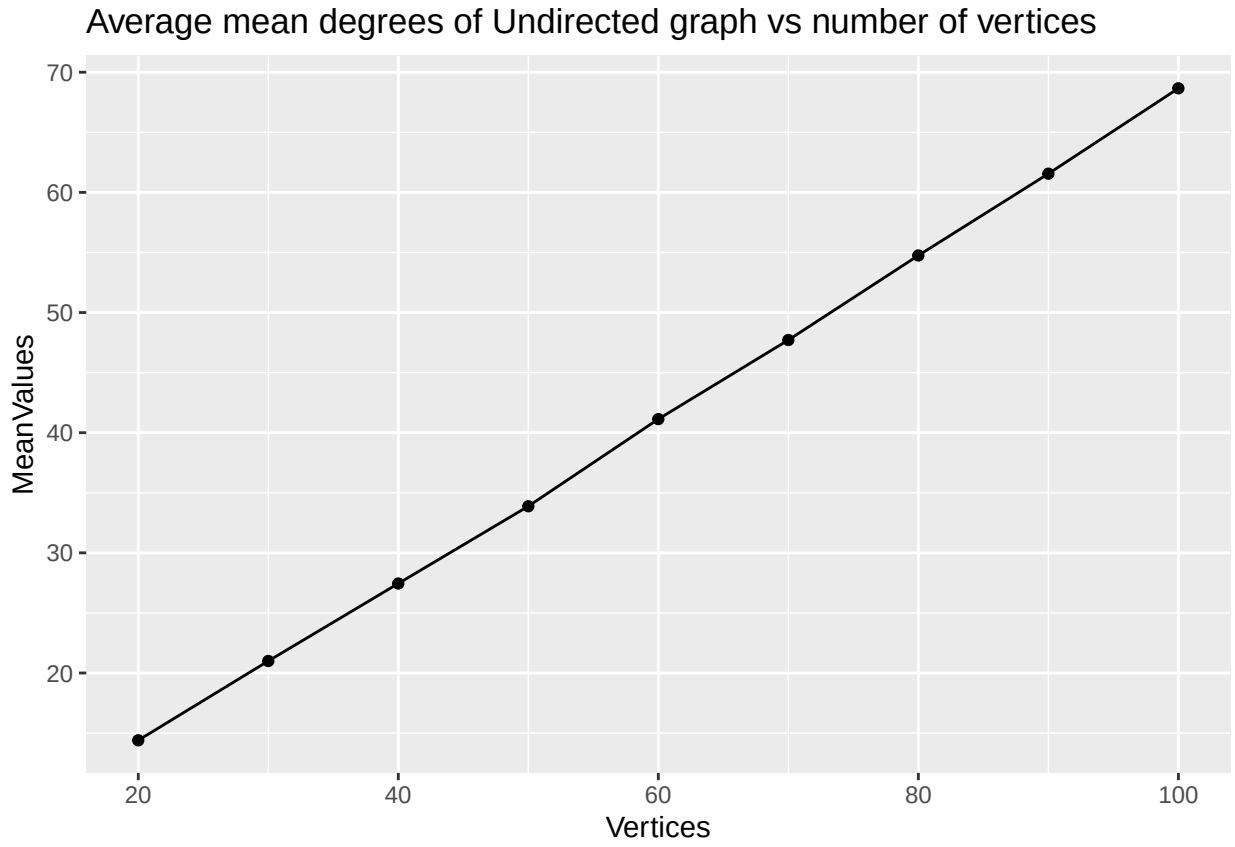
1.3 Plot the average mean degrees against the number of vertices. Is the plot result consistent with your expectation? Explain.

```
#create the tibble
mean_degrees <- c("MD20","MD30", "MD40", "MD50", "MD60", "MD70", "MD80", "MD90", "MD100")
graph_means_t <- tibble(Vertices = seq(from = 20, to = 100, by = 10),
                        MeanDegrees = mean_degrees,
                        MeanValues = mean_list)
graph_means_t
```

```
## # A tibble: 9 x 3
##   Vertices MeanDegrees MeanValues
##      <dbl> <chr>            <dbl>
## 1       20 MD20              14.4
## 2       30 MD30              21
## 3       40 MD40              27.4
## 4       50 MD50              33.9
## 5       60 MD60              41.1
## 6       70 MD70              47.7
## 7       80 MD80              54.8
## 8       90 MD90              61.6
## 9      100 MD100             68.7
```

Create the plot

```
ggplot(data = graph_means_t, mapping = aes(x = Vertices, y = MeanValues)) +
  geom_point() +
  geom_line() +
  labs(title = "Average mean degrees of Undirected graph vs number of vertices")
```



The results indicate a linear relationship. As the number of of vertexes increase, so does the average degree

Problem 2

1.Download the data frames "Dataset1-Media-Example-NODES.csv and"Dataset1-Media-Example-EDGES.csv" from https:// (Links to an external site.)net/network-visualization (Links to an external site.), and create an igraph network object. 2.Convert the igraph to a network object in the sna package 3. Plot the network using, using separate colors for the nodes based on the vertex attribute media.type and make the size of the nodes proportional to the vertex attribute audience.size. [Hint: Use network::get.vertex.attribute] 4. Calculate the mean degree and density of the network using appropriate functions in the sna package.

2.1 Create an igraph from the data frame

```
nodes <- read_csv("C:\\Users\\Julia\\Downloads\\Dataset1-Media-Example-NODES.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   id = col_character(),
##   media = col_character(),
##   media.type = col_double(),
##   type.label = col_character(),
##   audience.size = col_double()
## )
```

```
links <- read_csv("C:\\Users\\Julia\\Downloads\\Dataset1-Media-Example-EDGES.csv")
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   from = col_character(),
##   to = col_character(),
##   type = col_character(),
##   weight = col_double()
## )
```

```
net_2 <- graph_from_data_frame(d=links, vertices = nodes, directed = TRUE) %>%
  simplify(remove.multiple = F, remove.loops = T)
```

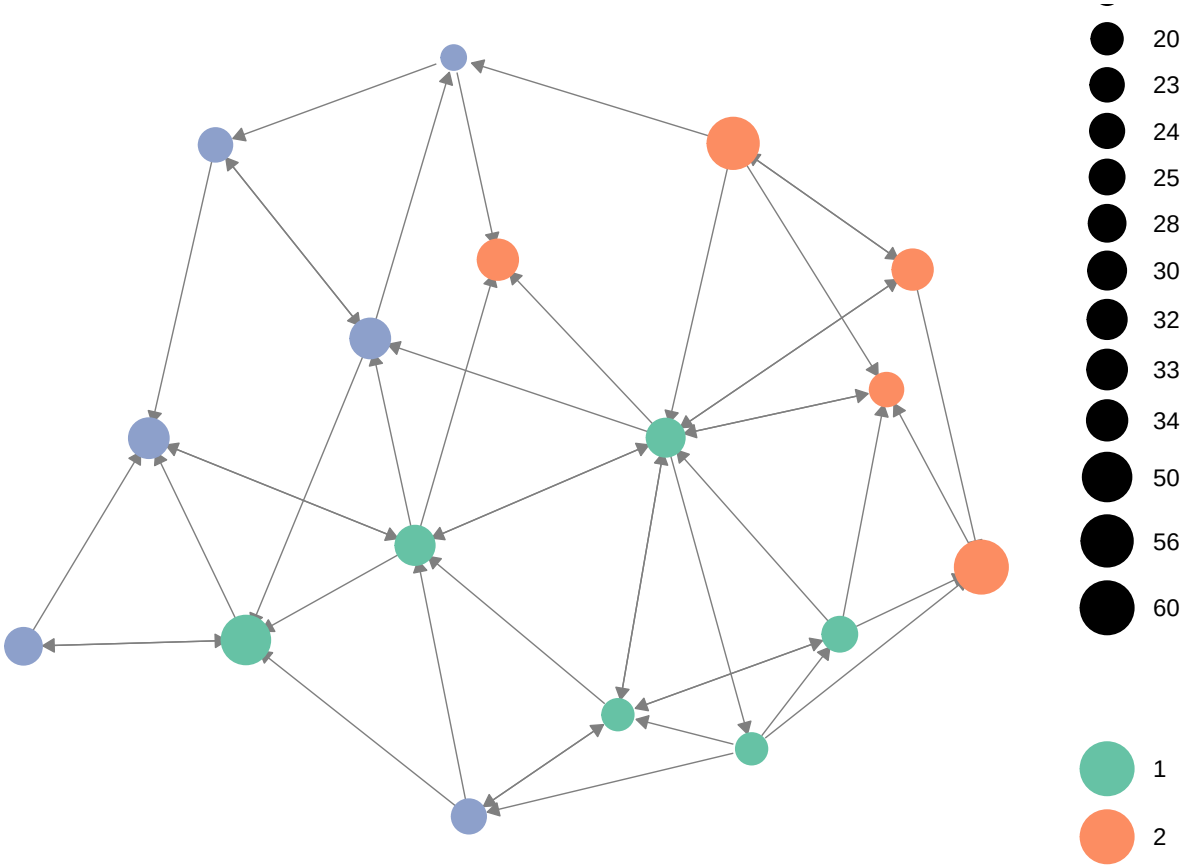
2.2 Convert the igrpah to network object in the sna package

```
network_obj <- asNetwork(net_2)
```

2.3 Plot the network using, using separate colors for the nodes based on the vertex attribute media.type and make the size of the nodes proportional to the vertex attribute audience.size. [Hint: Use network::get.vertex.attribute]

```
g_size <- network::get.vertex.attribute(network_obj, "audience.size")
g_color <- network::get.vertex.attribute(network_obj, "media.type")

ggnet2(network_obj, arrow.size = 5, arrow.gap = 0.02, size = g_size,
       label = FALSE,  color = g_color, palette = "Set2")
```



2.4 Calculate the mean degree and density of the network using appropriate functions in the sna package. Density

```
density <- network.density(network_obj)
density
```

```
## [1] 0.1764706
```

Mean degree

```
mean_degree <- mean(sna::degree(network_obj, gmode = "digraph"))
mean_degree
```

```
## [1] 5.647059
```

PROBLEM 3:

Scrape the country-wise population data from https://www.worldometers.info/world-population/population-by-country/ Plot the population density (P/Km2) obtained from this table on a country-wise choropleth map. Make sure to:

1.Clean the data to make it compatible with the country-wise world choropleth map 2.Maximize the overlap between the two data frames (the one obtained from the scraped data and the choropleth country data frame), i.e., if a country appears in both data frames, possibly with different names, it must be plotted 3.List

the countries, if any, in the scraped data frame that do not appear in the choropleth country data frame (after appropriate cleaning) 4.List the countries, if any, in the choropleth country data frame that do not appear in the scraped data frame (after appropriate cleaning)

Scrape data using rvest

```
data("country.regions") #Will be necessary as cross-reference
url <- "https://www.worldometers.info/world-population/population-by-country/"
worldometer_data <- url %>%
  read_html() %>%
  html_nodes("table") %>%
  html_table(fill = TRUE)

population_density_t <- worldometer_data[[1]] %>% dplyr::select(-1)
```

```
clean_column <- function(data, col_name) {
  #Cast data as a tibble and extract the col_name as a vector
  col_vec <- as_tibble(data) %>%
    pull(col_name)

  #If we are dealing with the value column, remove commas and convert to numeric
  if (col_name == "value") {
    return(
      col_vec %>%
        str_replace_all("[,]", "") %>%
        as.numeric())
  }
  #Else we are dealing with region column. We can only do initial processing here
  #which means lowercasing the column
  else {
    return(
      col_vec %>% tolower()
    )
  }
}
```

Call the helper function to perform initial processing and select necessary columns

```
#Extract necessary region and value columns from population_density_t
cleaned_t <- population_density_t %>%
  dplyr::select(1, 5)
names(cleaned_t)[1] <- "region"
names(cleaned_t)[2] <- "value"

#Call the clean_column() function to perform initial preprocessing on the data.
#Sort by the country/region ascending
cleaned_t <- cleaned_t %>%
  mutate(region = clean_column(cleaned_t, "region"),
         value = clean_column(cleaned_t, "value")) %>%
  arrange(region)
```

use anti_join() to see the countries that are different between the country.regions and the ones in our web scraped data

```
country_regions <- country.regions
misfits = anti_join(country_regions, cleaned_t)
```

```
## Joining, by = "region"
```

```
misfits %>% head(10)
```

```
##                              region iso2c
## 1                         macedonia    MK
## 2          united states of america    US
## 3                        somaliland    SO
## 4                republic of serbia    RS
## 5                         swaziland    SZ
## 6                        east timor    TL
## 7       united republic of tanzania    TZ
## 8                       the bahamas    BS
## 9                       ivory coast    CI
## 10 democratic republic of the congo    CD
```

Doing some outside work, we list the respective countries

```
choropleth_countries <- c("united states of america","republic of serbia",
               "united republic of tanzania","the bahamas",
               "democratic republic of the congo",
               "republic of congo","czech republic","guinea bissau",
               "ivory coast","macedonia","east timor","swaziland")
worldometer_countries <- c("united states","serbia","tanzania","bahamas","dr congo",
                 "congo","czech republic (czechia)","guinea-bissau",
                 "côte d'ivoire","north macedonia","timor-leste","eswatini")
```

Replace the scraped versions of the misfits with the ones compatible with choropleth

```
for (i in seq(worldometer_countries)) {
  cleaned_t$region[cleaned_t$region==worldometer_countries[i]] <-
    choropleth_countries[i]
}
```

Below are the countries not present at all in the choropleth dataframe

```
misfits_choro <- anti_join(cleaned_t, country_regions)
```

```
## Joining, by = "region"
```

```
misfits_choro %>% head(10)
```

```
##                    region value
## 1          american samoa   276
## 2                 andorra   164
## 3                anguilla   167
## 4     antigua and barbuda   223
```

```
## 5                   aruba   593
## 6                 bahrain  2239
## 7                barbados   668
## 8                 bermuda  1246
## 9  british virgin islands   202
## 10             cabo verde   138
```

```
nrow(misfits_choro)
```

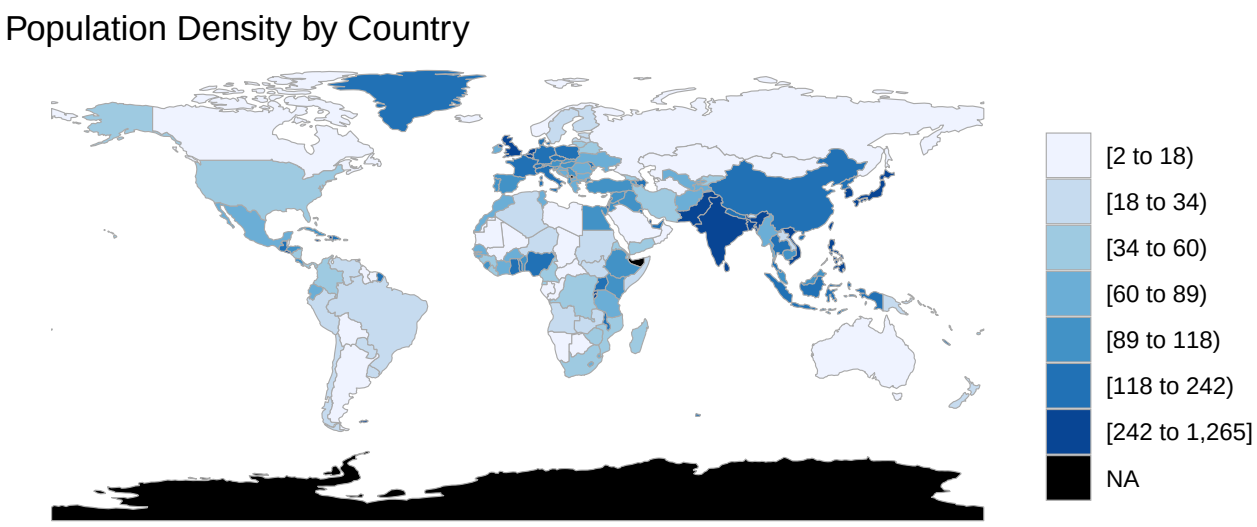
```
## [1] 68
```

Generate the choropleth

```
choroplethr::country_choropleth(cleaned_t, title = "Population Density by Country")
```

```
## Warning in super$initialize(country.map, user.df): Your data.frame contains the
## following regions which are not mappable: american samoa, andorra, anguilla,
## antigua and barbuda, aruba, bahrain, barbados, bermuda, british virgin islands,
## cabo verde, caribbean netherlands, cayman islands, channel islands, comoros,
## cook islands, curaçao, dominica, faeroe islands, falkland islands, french
## guiana, french polynesia, gibraltar, greenland, grenada, guadeloupe, guam,
## holy see, hong kong, isle of man, kiribati, liechtenstein, macao, maldives,
## malta, marshall islands, martinique, mauritius, mayotte, micronesia, monaco,
## montserrat, nauru, new caledonia, niue, northern mariana islands, palau, puerto
## rico, réunion, saint barthelemy, saint helena, saint kitts & nevis, saint lucia,
## saint martin, saint pierre & miquelon, samoa, san marino, sao tome & principe,
## seychelles, singapore, sint maarten, st. vincent & grenadines, state of
## palestine, tokelau, tonga, turks and caicos, tuvalu, u.s. virgin islands, wallis
## & futuna
```

```
## Warning in self$bind(): The following regions were missing and are being set to
## NA: somaliland, northern cyprus, antarctica, kosovo
```



PROBLEM 4:

Obtain 2015-2019 5-year aggregated ACS tract-wise data on NJ median household income and rental. Combine them into a single data frame.

1.Plot the tract-wise rental against the median household income and comment. 2.Fit a linear regression equation of rental against median household income and report the summary. 3.Looking at the plot, suggest ways to improve the model fit. Fit the improved model and report the R2. 4.Obtain the rental data for year=2019 (5-year aggregate from 2015-2019) and year=2014 (5-year aggregate from 2010-2014), and plot the percentage changes for each county in a column or bar diagram, in increasing or decreasing order of percentage increase.

4.1 Plot the tract-wise rental against median household income

```
my_census_key <- "68b91a0cb82e99d93fcf934623a6b9ad5ba04986"
census_api_key(my_census_key, install = TRUE, overwrite = TRUE)
```

```
## Your original .Renviron will be backed up and stored in your R HOME directory if needed.
```

```
## Your API key has been stored in your .Renviron and can be accessed by Sys.getenv("CENSUS_API_KEY").
## To use now, restart R or run `readRenviron("~/.Renviron")`
```

```
## [1] "68b91a0cb82e99d93fcf934623a6b9ad5ba04986"
```

```
v19 <- load_variables(2019, "acs5", cache = TRUE) #Load the variables

acs_data_t <- get_acs(geography = "tract",
```

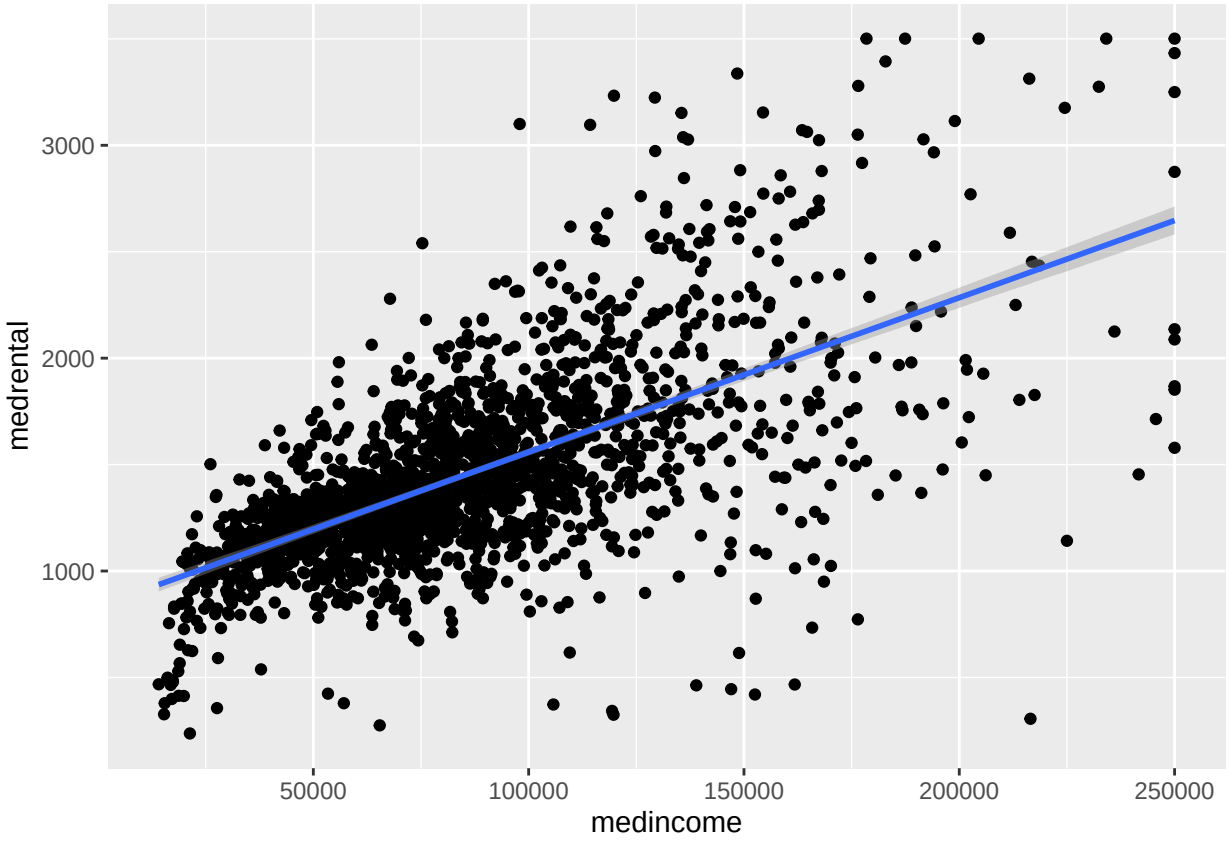
```
                        variables = c(medincome = "B19013_001",
                                      medrental = "B25064_001"),
                        state = "NJ")
```

```
## Getting data from the 2015-2019 5-year ACS
```

```
acs_data_t <- acs_data_t %>% select(NAME, variable, estimate) %>%
  pivot_wider(names_from = variable, values_from = estimate)

acs_data_t <- acs_data_t %>% filter(!is.na(medrental), !is.na(medincome))
ggplot(data = acs_data_t, mapping = aes(x = medincome, y = medrental)) +
  geom_point() +
  geom_smooth(method = "lm")
```

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



It seems as though the two variables are positively correlated

4.2Fit a linear regression equation of rental against median household income

```
acs_lm <- lm(medrental ~ medincome, data = acs_data_t)

summary_lm <- summary(acs_lm)
lm_rsq <- summary_lm$r.squared
lm_rsq
```

```
## [1] 0.4123706
```

4.3 Looking at the plot, suggest ways to improve the model fit. Looking at the plot, suggest ways to improve the model fit.

```
acs_poly <- lm(medrental ~ poly(medincome, 5), data = acs_data_t)
summary_poly <- summary(acs_poly)
poly_rsq <- summary_poly$r.squared
poly_rsq
```

```
## [1] 0.421725
```

Fitting a poly model with order=5 increased the R^2 by almost 0.01

4.4 Obtain the rental data for year=2019 (5-year aggregate from 2015-2019) and year=2014 (5-year aggregate from 2010-2014), and plot the percentage changes for each county in a column or bar diagram, in increasing or decreasing order of percentage increase.

Get the 2014 and 2019 data

```
rental_14 <- get_acs(geography = "county",
                     variables = c(medrental= "B25064_001"),
                     year = 2014, state = "NJ") %>%
  rename(region = NAME, value = estimate)
```

```
## Getting data from the 2010-2014 5-year ACS
```

```
rental_19 <- get_acs(geography = "county",
                     variables = c(medrental= "B25064_001"),
                     year = 2019, state = "NJ") %>%
  rename(region = NAME, value = estimate)
```

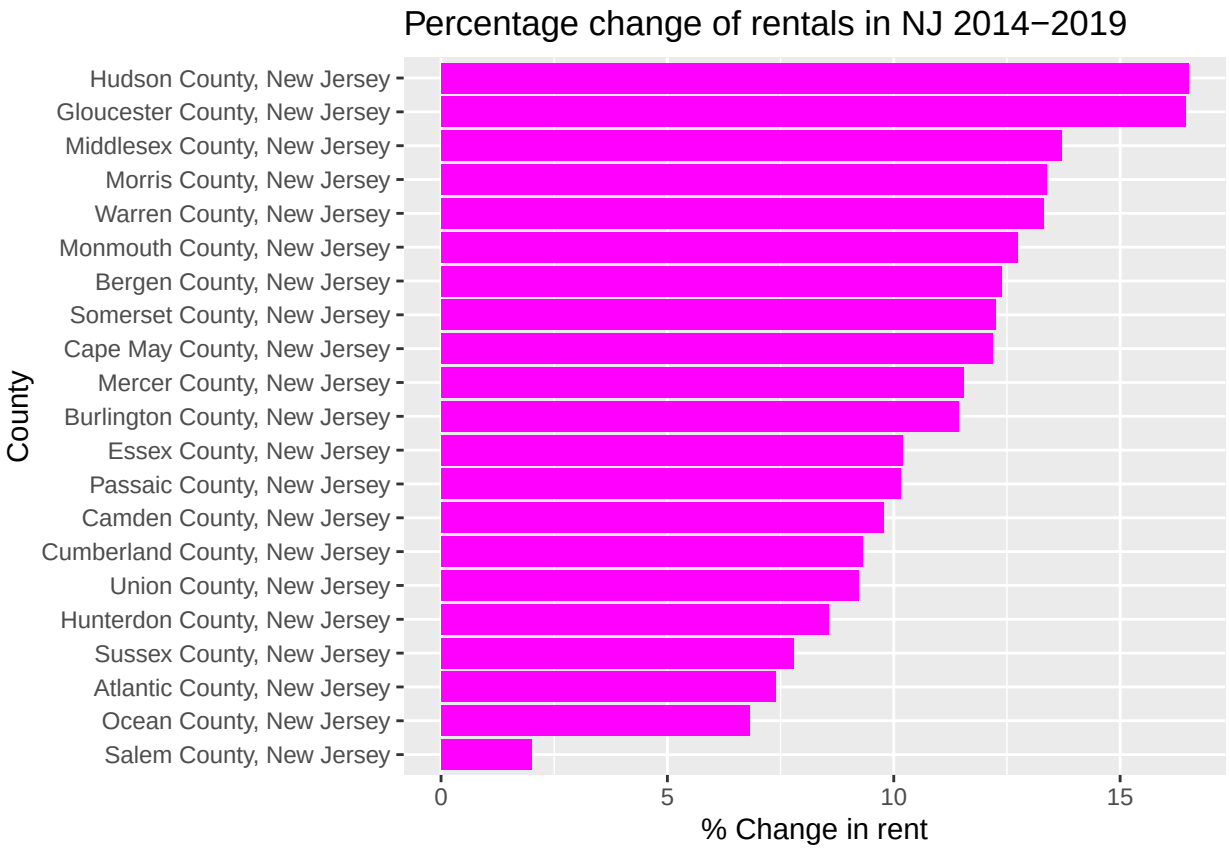
```
## Getting data from the 2015-2019 5-year ACS
```

Use choroplethr to calculate the percent change and store in a dataframe

```
percent_change_t <- choroplethr::calculate_percent_change(rental_14, rental_19)
```

Create a ggplot() column graph to show to show information on rental changes

```
ggplot(data = percent_change_t,
       mapping = aes(x = value, y = fct_reorder(region,value))) +
  geom_col(fill = "magenta") +
  labs(y="County",
       x="% Change in rent",
       title = "Percentage change of rentals in NJ 2014-2019")
```


Percentage change of rentals in NJ 2014−2019
Hudson County, New Jersey
Gloucester County, New Jersey
Middlesex County, New Jersey
Morris County, New Jersey
Warren County, New Jersey
Monmouth County, New Jersey
Bergen County, New Jersey
Somerset County, New Jersey
Cape May County, New Jersey
Mercer County, New Jersey
Burlington County, New Jersey
Essex County, New Jersey
Passaic County, New Jersey
Camden County, New Jersey
Cumberland County, New Jersey
Union County, New Jersey
Hunterdon County, New Jersey
Sussex County, New Jersey
Atlantic County, New Jersey
Ocean County, New Jersey
Salem County, New Jersey
County
0
5
10
15
% Change in rent