Mini Project # 2
Name: Niveditha Varadha Chandrasekaran

Exercise 1

Problem Statement:
Use R to make a map of the states/provinces/regions showing GDP or income inequality in a country of your choice (except USA). Please mention the source of the data you used to create the map.

Read the handout about map making in R for a prototype example.

Simply replace "USA" with "the country of your choice" in the Data function in the handout to get a shape file of the states/provinces/regions in the country of your choice.

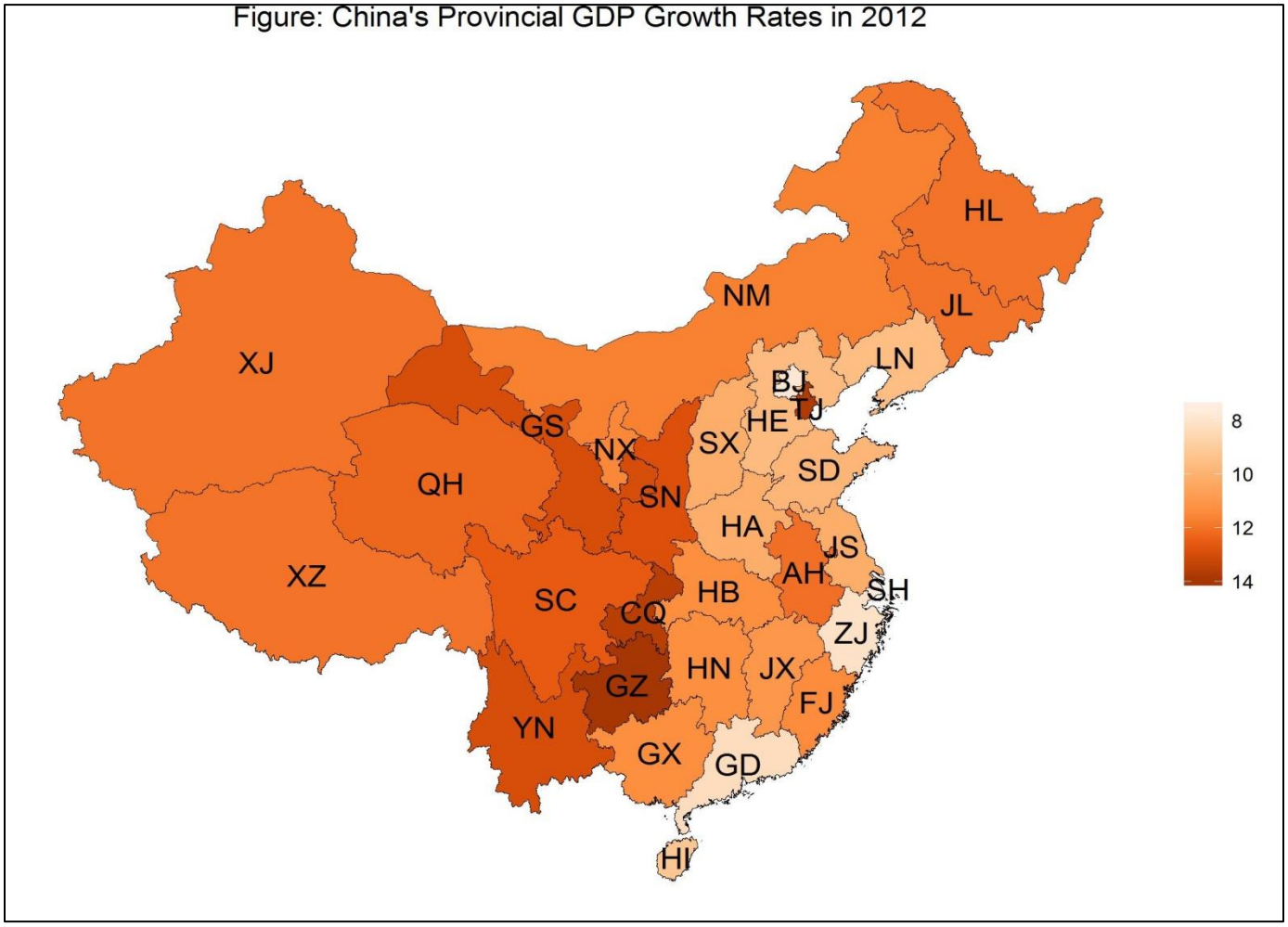
The maps should label the states/regions/provinces. Label the states with their abbreviations as has been done on the maps shown on the website:
http://www.shsu.edu/eco_mwf/inequality.html.
You should be able to figure this out by googling.

Solution:

Let us consider provincial GDP growth rate of **"China"** in 2012.
Source of data:
http://www.china-briefing.com/news/2013/05/16/chinas-provincial-gdp-figures-in-2012.html

Province	provincial GDP growth rate (%) - 2012	Province	provincial GDP growth rate (%) - 2012	Province	provincial GDP growth rate (%) - 2012
Anhui	12.1	Hubei	11.3	Shanxi	10.1
Beijing	7.7	Hunan	11.3	Sichuan	12.6
Chongqing	13.6	Nei Mongol	11.7	Tianjin	13.8
Fujian	11.4	Jiangsu	10.1	Xizang	12.0
Gansu	13.0	Jiangxi	11.0	Xinjiang Uygur	12.0
Guangdong	8.2	Jilin	12.0	Yunnan	13.0
Guangxi	11.3	Liaoning	9.5	Zhejiang	8.0
Guizhou	14.0	Ningxia Hui	11.5		
Hainan	9.1	Qinghai	12.3		
Hebei	9.6	Shaanxi	12.9		
Heilongjiang	12.0	Shandong	9.8		
Henan	10.1	Shanghai	7.5		

Output:



Exercise 2

Problem Statement:

Mixture models form one of the most fundamental classes of generative models for clustered data. This will have a multimodal distribution.

The following are run times (in seconds) on unix servers in 100 universities.

30.16 30.36 97.83 101.59 106.42 30.75 100.10 103.30 101.73 25.48 98.90 31.41 26.33 32.35
96.52 31.93 108.32 99.72 101.11 103.92 97.87 97.83 99.22 97.51 103.24 29.31 29.82 98.42
34.28 27.12 99.28 103.77 102.61 27.22 97.71 105.96 102.41 30.38 101.73 98.59 100.14

99.09 27.44 100.37 99.84 97.34 101.17 99.14 97.41 99.92 101.31 104.61 100.71 30.62 103.57 28.35 108.12 100.05 31.84 28.80
98.47 27.99 105.05 33.33 100.09 23.57 101.68 95.62 102.10 98.77 100.93 98.68 27.00 102.04 100.88 98.79 102.58 27.40 29.01 29.57 97.16 96.60 105.35 97.74 100.97 101.88 96.75 29.01 98.08 99.63 99.41 101.96 26.70 31.66 98.29 103.51 99.28 99.10 33.36 100.36

Using mixture normal distribution (a bimodal distribution)
$0.7\times N(\mu_1,\sigma_1^2)+0.3\times N(\mu_2,\sigma_2^2)$

(a) (4 points) find the maximum likelihood estimates of the unknown parameters and their standard errors. Use any appropriate R package or otherwise.
(b) (3 points) Draw a histogram of the data and superimpose the density of the above mixture normal distribution using maximum likelihood estimates of the unknown parameters.

Solution:

(a) Using R we get the maximum likelihood estimates of the unknown parameters (μ_1, σ_1, μ_2, σ_2) and their standard errors (SE) as:

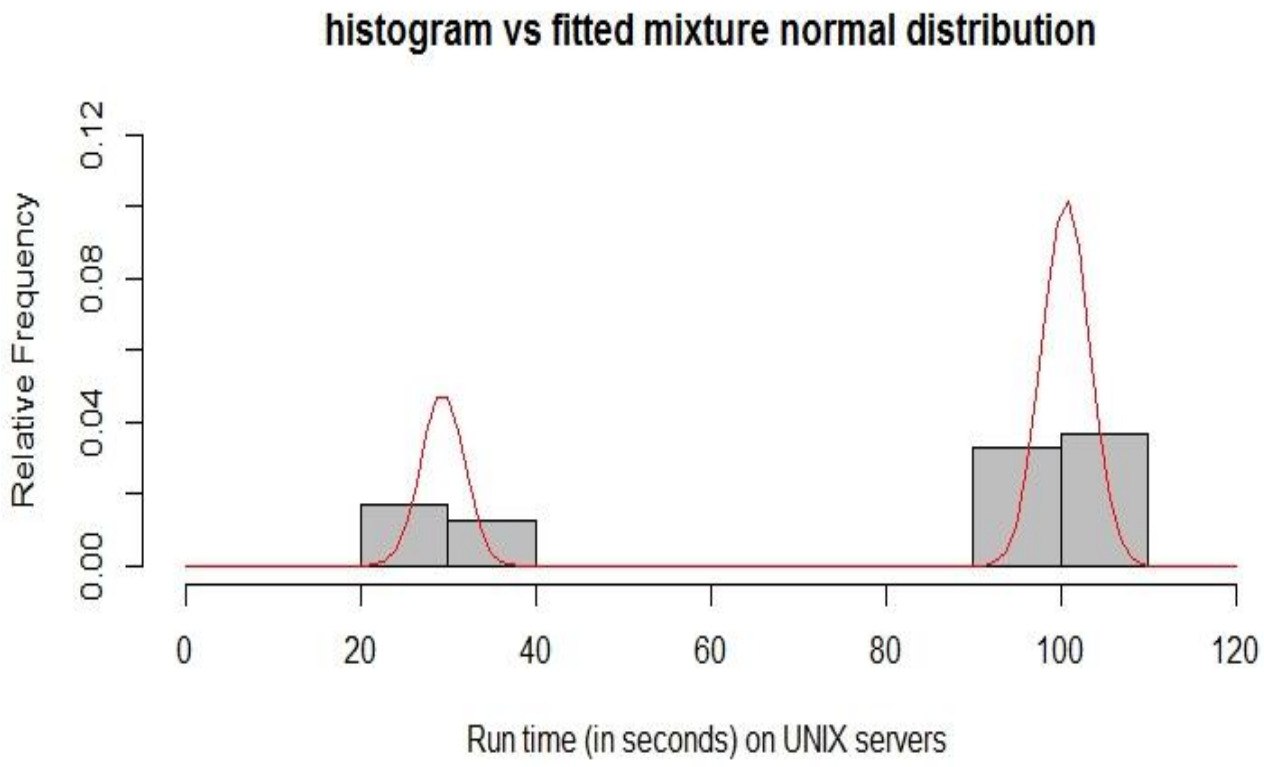
- $\mu_{1=}$ 29.418338
- $\sigma_{1=}$ 2.479900

- $\mu_{2=}$ 100.573572
- $\sigma_{2\,=}$ 2.736789

- SE(μ_1) = 0.4527657
- SE(σ_1) = 0.3201538

- SE(μ_2) = 0.3271088
- SE(σ_2) = 0.2313010

(b) The histogram of the data and the superimposed density of the above mixture normal distri bution using maximum likelihood estimates of the unknown parameters is as given below.



R Code:

Exercise 1:

```
> library(raster) # to get map shape file
> library(ggplot2) # for plotting and miscellaneuous things
> library(ggmap) # for plotting
> library(plyr) # for merging datasets
> library(scales) # to get nice looking legends
> library(maps)
> # Get a shape file of provinces of china
> china.shape <- getData("GADM", country = "chn", level = 1)
> par(mar = rep(2, 4))
> plot(china.shape)
```

```
> # To merge population data to the shape file, convert the shape file into a dataframe
> china.df <- fortify(china.shape)
Regions defined for each Polygons
> str(china.df)
'data.frame':         894920 obs. of  7 variables:
 $ long : num  116 116 116 116 116 ...
 $ lat  : num  34.6 34.6 34.6 34.6 34.6 ...
 $ order: int  1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 732 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ id   : chr  "1" "1" "1" "1" ...
 $ group: Factor w/ 2021 levels "1.1","2.1","3.1",..: 1 1 1 1 1 1 1 1 1 1 ...
> # Issue: the dataframe does not have state names, only id's
> # Add the state names to the data frame
> china.df$id <- as.numeric(china.df$id)
> state.names <- data.frame(id = 1:length(china.shape$NAME_1), state = china.shape$NAME_1
)
>
> # Merge the shape data with the population data by state name
> china.df <- join(china.df, state.names, by = "id", type = "inner")
> china.df$state<-tolower(china.df$state)
> str(china.df)
'data.frame':         894920 obs. of  8 variables:
 $ long : num  116 116 116 116 116 ...
 $ lat  : num  34.6 34.6 34.6 34.6 34.6 ...
 $ order: int  1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 732 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ id   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ group: Factor w/ 2021 levels "1.1","2.1","3.1",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ state: chr  "anhui" "anhui" "anhui" "anhui" ...
> china.dat <- read.table("C:\\Users\\Niveditha\\Desktop\\china_data.csv",
+                         header = T, sep = ",")
> china.dat$state <- tolower(china.dat$state)
> china.df <- join(china.df, china.dat, by = "state", type = "inner")
> str(china.df)
'data.frame':         894920 obs. of  10 variables:
 $ long : num  116 116 116 116 116 ...
 $ lat  : num  34.6 34.6 34.6 34.6 34.6 ...
 $ order: int  1 2 3 4 5 6 7 8 9 10 ...
 $ hole : logi  FALSE FALSE FALSE FALSE FALSE FALSE ...
 $ piece: Factor w/ 732 levels "1","2","3","4",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ id   : num  1 1 1 1 1 1 1 1 1 1 ...
 $ group: Factor w/ 2021 levels "1.1","2.1","3.1",..: 1 1 1 1 1 1 1 1 1 1 ...
```

```
 $ state: chr  "anhui" "anhui" "anhui" "anhui" ...
 $ year : int  2012 2012 2012 2012 2012 2012 2012 2012 2012 2012 ...
 $ PGDP : num  12.1 12.1 12.1 12.1 12.1 12.1 12.1 12.1 12.1 12.1 ...
>
> china.2012 <- china.df[china.df$year == 2012, ]
>
> # Divide Percentage GDP into class intervals --- there will be one color for each interval
> # provide only the upper limits of the intervals (the break points)
> brks.2012 <- seq(6, 16, by = 2)
>
> #Provide title for the map
> title <- "Figure: China's Provincial GDP Growth Rates in 2012"
>
> #Get coordinates for all the provinces in china
> china.center.df <- data.frame(long = coordinates(china.shape)[, 1],
+                               lat = coordinates(china.shape)[, 2])
> china.center.df[, 'ID_1'] <- china.shape$data[,'ID_1']
>
> # Add the state abbreviation to the data frame
> china.center.df[, 'abbr'] <- c("AH", "BJ", "CQ", "FJ", "GS","GD","GX","GZ","HI",
+                        "HE","HL","HA","HB","HN","JS","JX","JL","LN","NM",
+                        "NX","QH","SN","SD","SH","SX","SC","TJ","XJ","XZ",
+                        "YN","ZJ")
>
> #adjust the coordinates
> china.center.df$lat[10] = china.center.df$lat[10] - 1
> china.center.df$long[10] = china.center.df$long[10] - 1
> china.center.df$lat[27] = china.center.df$lat[27] - 0.25
> china.center.df$long[27] = china.center.df$long[27] + 0.10
> china.center.df$long[5] = china.center.df$long[5] + 1
> china.center.df$lat[5] = china.center.df$lat[5] + 0.5
> china.center.df$lat[2] = china.center.df$lat[2] + 0.10
> china.center.df$long[24] = china.center.df$long[24] + 0.8
>
> # Plot the china map with the data and get the heatmap and save it as an image
> p <- function(data, brks, title) {
+   ggp <- ggplot() + geom_polygon(data = data, aes(x = long, y = lat,
+                                       group = group, fill = PGDP),
+                          color = "black", size = 0.15) +
+     scale_fill_distiller(palette = "Oranges", trans = "reverse", breaks = brks) +
+     theme_nothing(legend = TRUE) + labs(title = title, fill = "") +
+     geom_text(data = china.center.df, aes(x = long, y = lat, label = abbr), size = 5)
+   return(ggp)
+ }
```

```
>
> ggsave(p(china.df, brks.2012, title), height = 6, width = 8,
+      file = "C:\\Users\\Niveditha\\Desktop\\china_map_2012.jpg")
```

Exercise 2:

```
> RunTime <- scan(file="  C:\\Users\\Niveditha\\Desktop\\runtime.txt")
> # On seeing the histogram we get:
> mu1= 29.49  #μ1
> s1 = 2.52   #standard deviation 1
> mu2= 100.57 #μ2
> s2= 2.76    #standard deviation 2
>
> # Negative of log-likelihood function assuming mixture normal parent distri
bution
> neg.loglik.fun <- function(par,dat)
+ {
+   result <- sum(log(0.3*dnorm(dat, mean=par[1],sd=par[2])+
+                       0.7*dnorm(dat, mean=par[3],sd=par[4])))
+   return(-result)
+ }
> # Minimize -log (L), i.e., maximize log (L)
> ml.est <- optim(par=c(mu1,s1,mu2,s2), fn=neg.loglik.fun, method = "L-BFGS-B
",
+                 lower=rep(0,4), hessian=TRUE,dat=RunTime)
>
> # their standard errors
> sqrt(diag(solve(ml.est$hessian)))
[1] 0.4527657 0.3201538 0.3271088 0.2313010
>
> # How well the fitted model represents the data?
>
> # density function of mixture normal distribution
> pdf <- function(x) {
+   (0.3*dnorm(x, mean=ml.est$par[1],sd=ml.est$par[2]))+
+     (0.7*dnorm(x, mean=ml.est$par[3],sd=ml.est$par[4]))
+ }
> # relative frequency (density) histogram
> hist(RunTime, probability = TRUE,col = "grey",
+      xlab="Run time (in seconds) on UNIX servers ",
+      ylab="Relative Frequency",
+      main="histogram vs fitted mixture normal distribution",
+      xlim=c(0,120),ylim=c(0,0.12))
>
> # superimpose the fitted density
> curve(pdf, xlim=c(0,120),ylim=c(0,0.12),col = "red",
+       add= TRUE, yaxt = "n",xaxt = "n",
+       xlab = "", ylab = "")
>
> # MLE
> ml.est$par
```

```
[1]  29.418338   2.479900 100.573572   2.736789
> ml.est$value
[1] 300.7017
> ml.est$counts
function gradient
       7        7
> ml.est$convergence
[1] 0
> ml.est$message
[1] "CONVERGENCE: REL_REDUCTION_OF_F <= FACTR*EPSMCH"
> #hessian matirx
> ml.est$hessian
              [,1]          [,2]          [,3]          [,4]
[1,]  4.878126e+00 -1.693934e-05 -1.421085e-08  7.105427e-09
[2,] -1.693934e-05  9.756244e+00  0.000000e+00  1.421085e-08
[3,] -1.421085e-08  0.000000e+00  9.345777e+00 -5.968559e-07
[4,]  7.105427e-09  1.421085e-08 -5.968559e-07  1.869154e+01
> # their standard errors
> sqrt(diag(solve(ml.est$hessian)))
[1] 0.4527657 0.3201538 0.3271088 0.2313010
```