Gov 2018: Lab 8 Clustering and PCA

Your name: Alexandra Norris

Tuesday March 22, 2022

Overview

This lab consists of two parts: (1) model-based clustering and (2) PCA. Part of the explanation is based on the book written by Boehmke & Greenwell (2020), Hands-On Machine Learning with R.

```
# Load Packages
library(ggplot2)
library(MASS) # for mvrnorm, see below
library(mclust) # clustering
```

```
## Package 'mclust' version 5.4.9
## Type 'citation("mclust")' for citing this R package in publications.
```

```
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --
```

```
## v tibble  3.1.6     v dplyr   1.0.7
## v tidyr   1.1.4     v stringr 1.4.0
## v readr   2.0.2     v forcats 0.5.1
## v purrr   0.3.4
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## x purrr::map()    masks mclust::map()
## x dplyr::select() masks MASS::select()
```

Model-based Clustering

The key idea behind model-based clustering is that the data are considered as coming from a mixture of underlying probability distributions. The most popular approach is the Gaussian mixture model (GMM) (Banfield and Raftery 1993) where each observation is assumed to be distributed as one of k multivariate-normal distributions, where k is the number of clusters (commonly referred to as components in model-based clustering).

GMMs are founded on the multivariate normal (Gaussian) distribution where p variables $(X_1, \ldots, X_p)$ are assumed to have means $\mu = (\mu_1, \ldots, \mu_p)$ and a covariance matrix Σ, which describes the joint variability (i.e., covariance) between each pair of variables. GMMs assume the clusters can be created using k such Gaussian distributions. For example, if there are two variables (X_1 and X_2) then each observation (X_{1i}, X_{2i}) is modeled as having been sampled from one of k multivariate Gaussian distributions: e.g., $(X_{1i}, X_{2i}) \mid \text{Cluster}_i = k, \mu, \Sigma \sim \mathcal{N}_2(\mu_k, \Sigma_k)$).

Question 1. Clustering with synthetic data

In this question, we will fit GMM using synthetic data with two variables ($p = 2$).

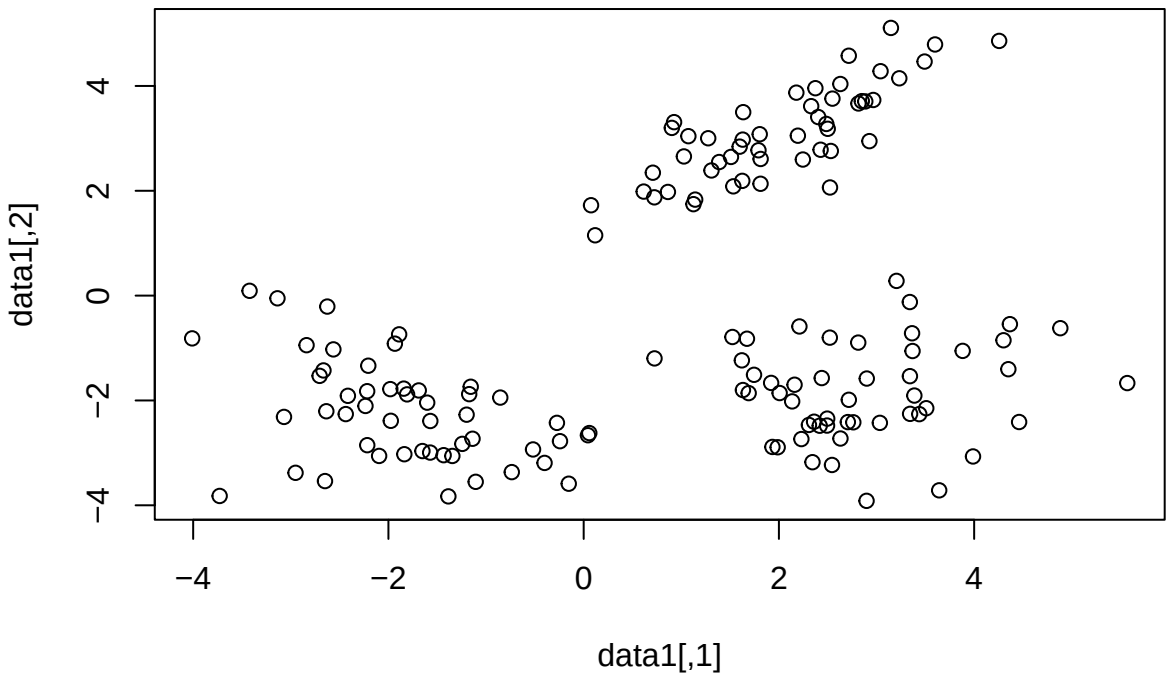
1.1

Generate synthetic data using the following codes:

```
set.seed(123)
# three sets of random correlated variables (that is, clusters)
# cluster 1
mu1 <- c(2, 3)
Sigma1 <- matrix(c(1, 0.8, 0.8, 1), ncol = 2)
cluster1 <- mvrnorm(n = 50, mu = mu1, Sigma = Sigma1)

# cluster 2
mu2 <- c(-2, -2)
Sigma2 <- matrix(c(1, -0.4, -0.4, 1), ncol = 2)
cluster2 <- mvrnorm(n = 50, mu = mu2, Sigma = Sigma2)

# cluster 3
mu3 <- c(3, -2)
Sigma3 <- matrix(c(1, 0.1, 0.1, 1), ncol = 2)
cluster3 <- mvrnorm(n = 50, mu = mu3, Sigma = Sigma3)
```

Now, combine these three cluster into one data frame using `rbind` and standardize each columns (`V1` and `V2`). Visualize the data using scatterplot.

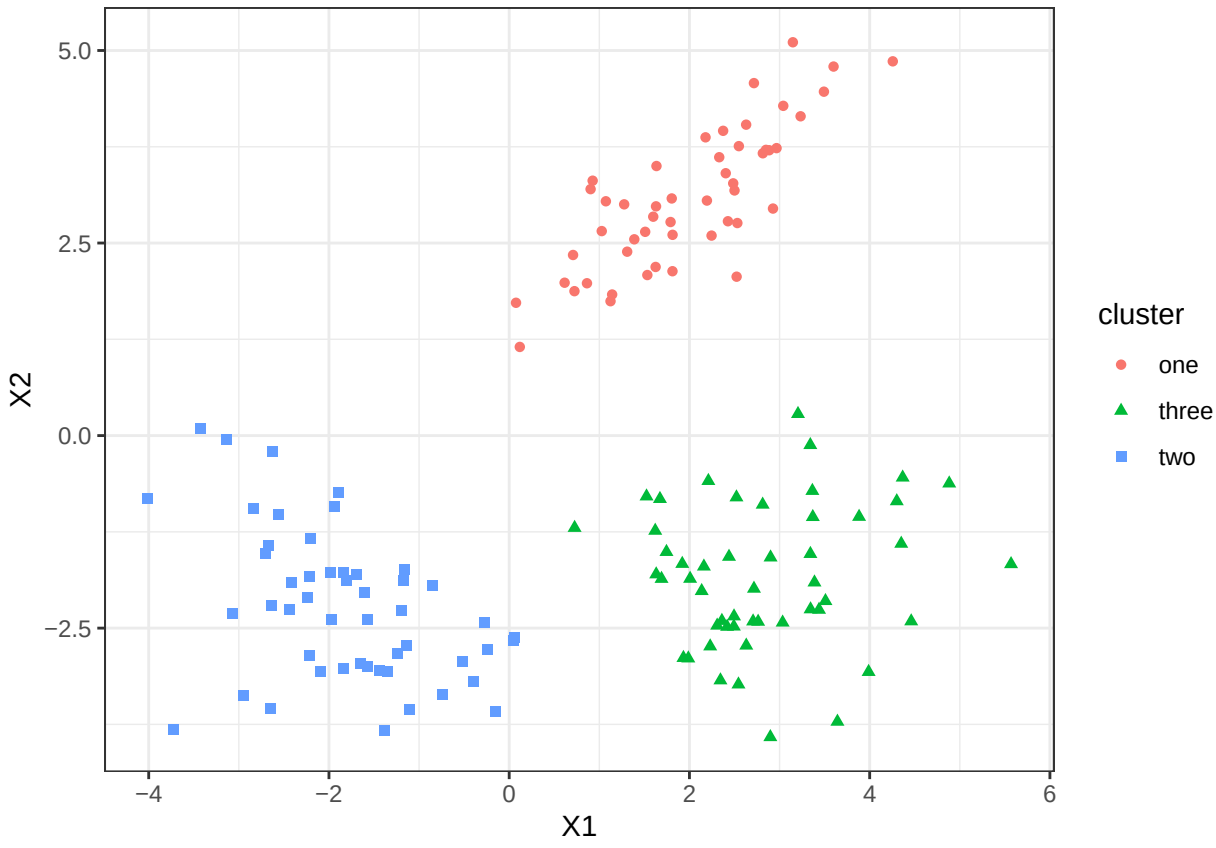
```
data1 <- rbind(cluster1, cluster2, cluster3)

plot(data1)
```



Create another scatterplot with different shape/color for each cluster. Note that this shows you the true

cluster we would like to estimate using GMM.

```
c1 <- data.frame(cluster1) %>%
  mutate(cluster = "one")

c2 <- data.frame(cluster2) %>%
  mutate(cluster = "two")

c3 <- data.frame(cluster3) %>%
  mutate(cluster = "three")

data <- data.frame(rbind(c1,c2,c3))

data %>%
  ggplot(aes(x = X1, y = X2, shape = cluster, color = cluster)) +
  geom_point() +
  theme_bw()
```



1.2

The covariance matrix Σ describes the geometry of the clusters; namely, the volume, shape, and orientation of the clusters. GMMs allow for flexible clustering structures by adding constraints to the covariance matrix. These constraints can be one or more of the following:

- volume: each cluster has approximately the same number of observations;
- shape: each cluster has approximately the same variance so that the distribution is spherical;
- orientation: each cluster is forced to be axis-aligned.

The various combinations of the above constraints have been classified into three main families of models: spherical, diagonal, and general (also referred to as ellipsoidal) families (Celeux and Govaert 1995). For example, EII denotes a model with spherical family, equal volume, equal shape, no orientation.

`Mclust` function from `mclust` package fit models with different Σ types and different number of clusters (k) by EM algorithm, and then select the best model using a selection criteria such as BIC.

Fit GMM using the following codes:

```
set.seed(123)
clustering1 <- Mclust(data1)
summary(clustering1)
```

```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
##
## Mclust EVE (ellipsoidal, equal volume and orientation) model with 3 components:
##
##  log-likelihood   n df       BIC       ICL
##       -543.2584 150 13 -1151.655 -1151.948
##
## Clustering table:
##  1  2  3
## 50 50 50
```

Visualize the performance of different models using BIC: `plot(clustering1, what = "BIC")`. Compare different models and briefly explain the best model.

```
# model selection based on BIC
plot(clustering1, what = "BIC")
```

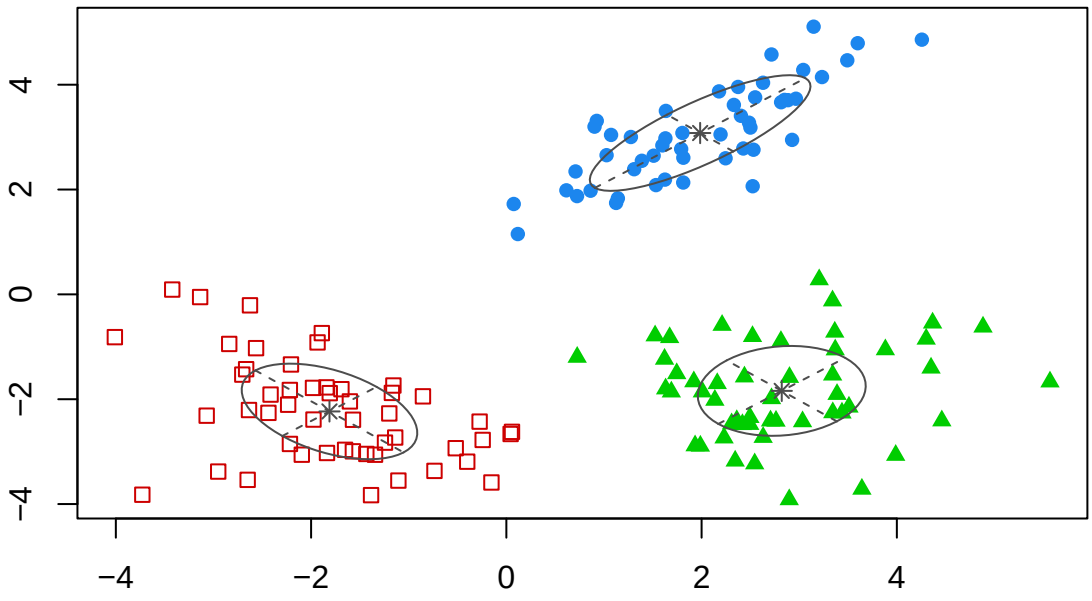
Create a contingency table with true clusters and estimates from the best fitted model. Visualize the clusters of the best fitted model: `plot(clustering1, what = "classification")`.

Hint: Use `classification` item from the Mclust object.

```
table(clustering1$classification, data$cluster)
```

```
##
##     one three two
##   1  50     0   0
##   2   0     0  50
##   3   0    50   0
```

```
plot(clustering1, what = "classification")
```



1.3

One of the advantages of using GMM for clustering lies in its use of probability and measure of uncertainty.

Draw an uncertainty plot where it visualizes the observations with high uncertainty (low probability) in its cluster assignment as larger points: `plot(clustering1, what = "uncertainty")`. Briefly discuss the results.

```
# estimated uncertainty of classification
plot(clustering1, what = "uncertainty")
```

Which observation has the maximum uncertainty? How the estimated probability of belonging to a particular cluster for that observation look like?

Hint: Use `uncertainty` (the uncertainty associated with the classification) and `z` (a matrix whose `[i,k]`th entry is the probability that observation i in the test data belongs to the kth class) items from the Mclust object.

```
round(max(clustering1$uncertainty), 3)
```

```
## [1] 0.048
```

```
# probability of uncertainty belinging to a cluster
round(clustering1$z[which.max(clustering1$uncertainty),], 3)
```

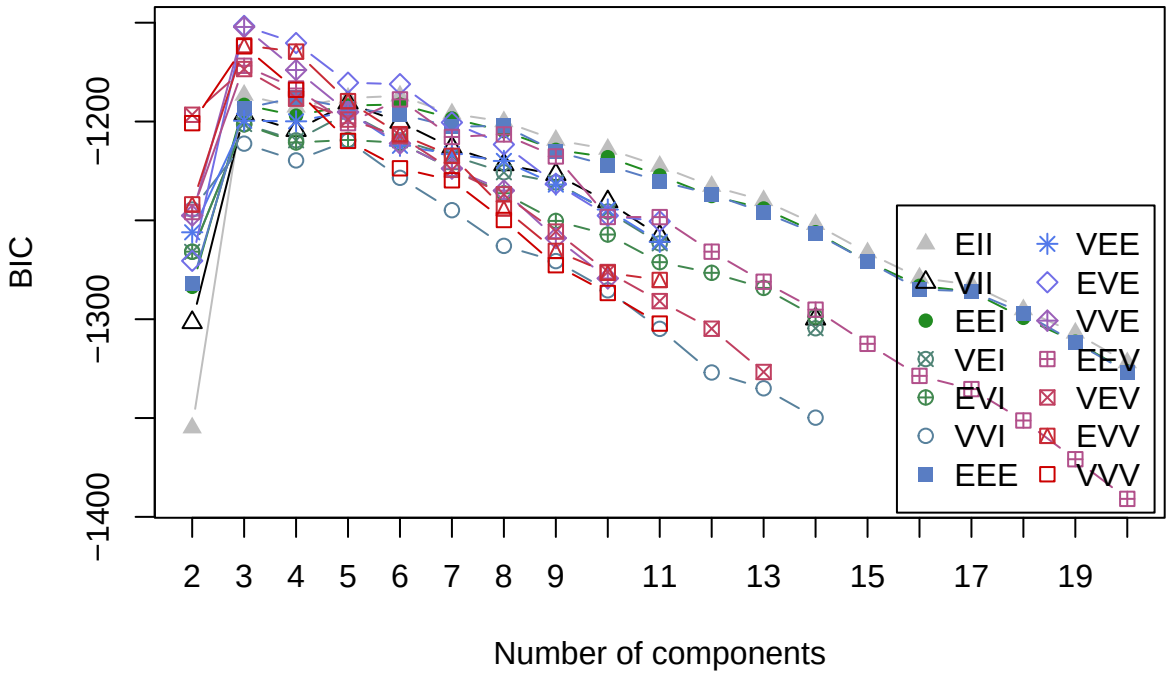
```
## [1] 0.000 0.952 0.048
```

1.4

So far, we did not specify k but used the default values of `G = 1:9` to fit the models for each value and find the best model. (Note that `G` argument from `Mclust` function is an integer vector specifying the numbers of mixture components/clusters for which the BIC is to be calculated.)

Fit GMM with `G = 2:20` and visualize its BIC and classification result of the best fitted model as before.

```
set.seed(123)
clustering2 <- Mclust(data1, G = 2:20)
plot(clustering2, what = "BIC")
```



Question 2. Clustering with real data

In this question, we will use `data("diabetes")` dataset with three variables ($p = 3$).

```
# a real data set ------
data("diabetes")
head(diabetes)
```

```
##    class glucose insulin sspg
## 1 Normal      80     356  124
## 2 Normal      97     289  117
## 3 Normal     105     319  143
## 4 Normal      90     356  199
## 5 Normal      90     323  240
## 6 Normal      86     381  157
```

```
diabetes_unlabeled <- diabetes[, -1]
diabetes_unlabeled <- scale(diabetes_unlabeled)
```

Fit GMM using `diabetes_unlabeled` and create a contingency table. Visualize BIC, classification, uncertainty as before and observe the results.

Bonus: Additionally, you can try `fviz_mclust` function from `factoextra` for more flexible visualization.

```
clustering3 <- Mclust(diabetes_unlabeled)
summary(clustering3)
```

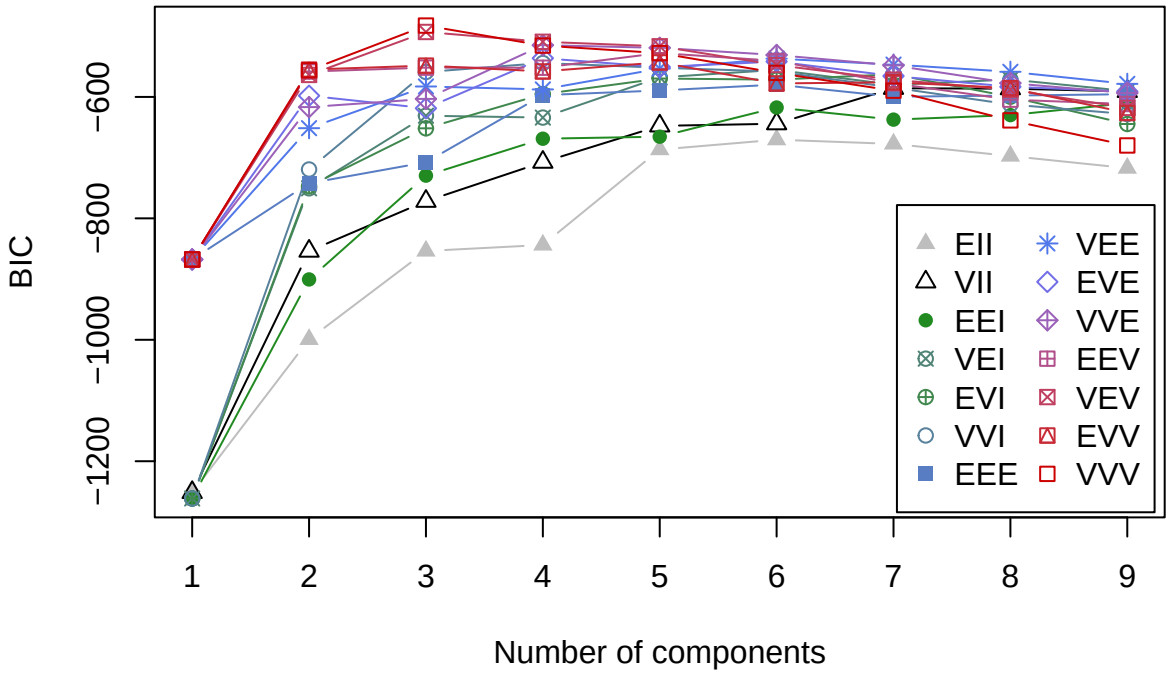
```
## ----------------------------------------------------
## Gaussian finite mixture model fitted by EM algorithm
## ----------------------------------------------------
```

```
##
## Mclust VVV (ellipsoidal, varying volume, shape, and orientation) model with 3
## components:
##
##  log-likelihood   n df       BIC       ICL
##       -169.0908 145 29 -482.5069 -501.4662
##
## Clustering table:
##  1  2  3
## 81 36 28
```

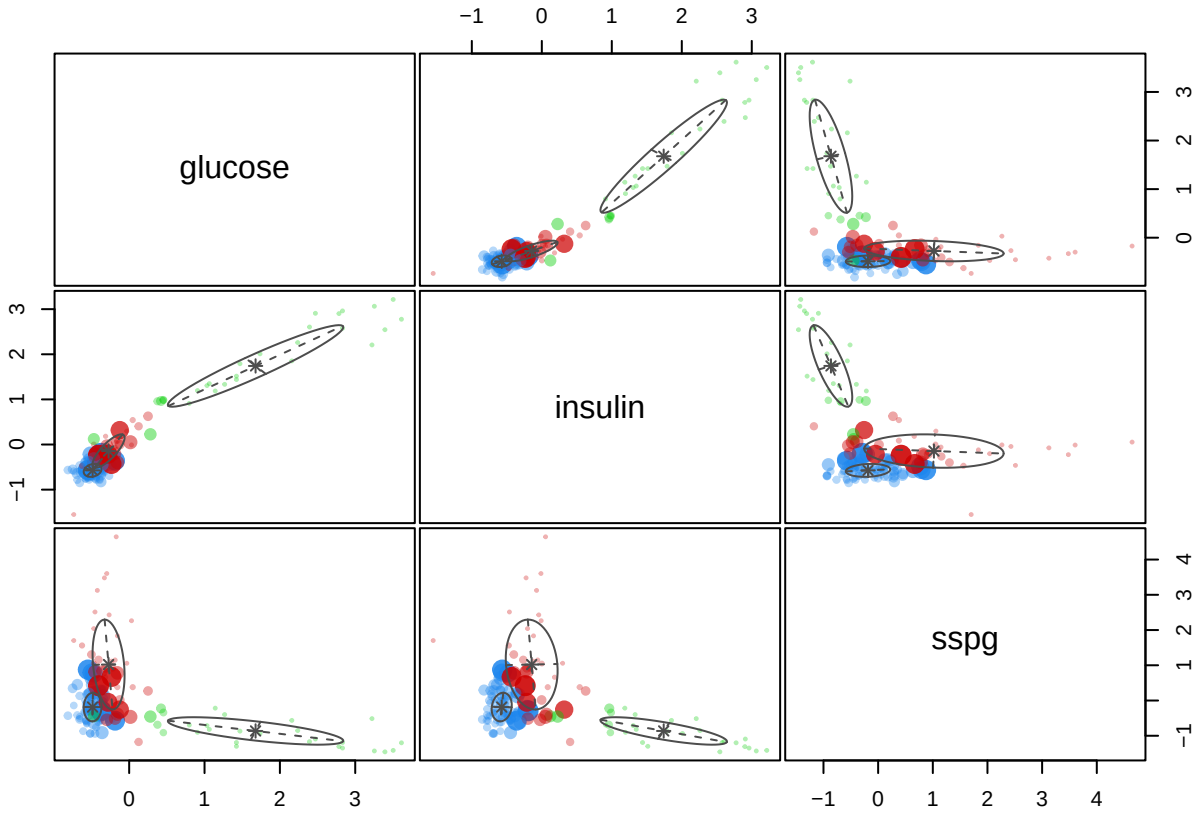
```
table(clustering3$classification, diabetes$class)
```

```
##
##     Chemical Normal Overt
##   1        9     72     0
##   2       26      4     6
##   3        1      0    27
```

```
plot(clustering3, what = "BIC")
```

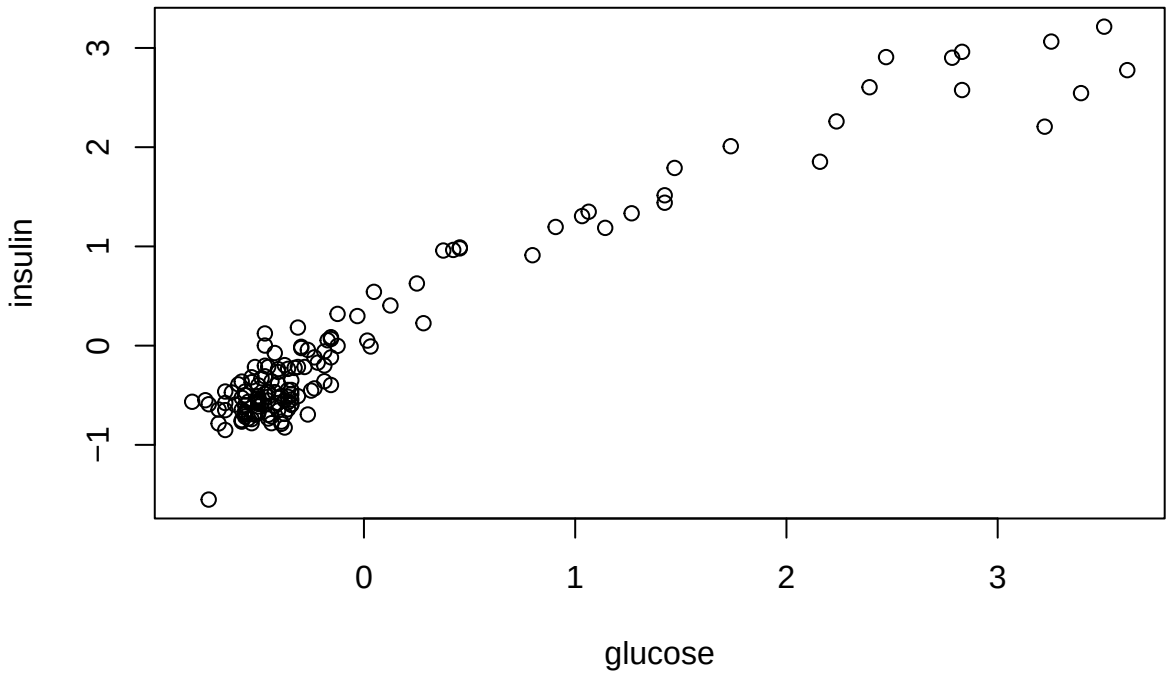



```
plot(clustering3, what = "uncertainty")
```




```
plot(diabetes_unlabeled, what = "classification")
```

```
## Warning in plot.window(...): "what" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "what" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "what" is not a
## graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "what" is not a
## graphical parameter

## Warning in box(...): "what" is not a graphical parameter

## Warning in title(...): "what" is not a graphical parameter
```



Principle Component Analysis

The second part of the lab will be a brief demonstration of principle component analysis. We'll be using PCA to explore and visualize data on world countries.

The data consists of 107 countries and 21 features about each country from the year 2000. These features capture a range of information, from demographics and development to politics and human rights.

The full list of variables include:

1. `idealpoint`: idealpoint preference based on UN votes, relative to US.
2. `polity`: combined polity score, indicating regime type (autocatic<—>democratic)
3. `polity2`: combined polity score 2, indicating regime type (autocatic<—>democratic)
4. `democ`: Institutionalized Democracy
5. `autoc`: Institutionalized Autocracy
6. `unreg`: UN region
7. `physint`: CIRI physical integrity score (ciri)
8. `speech`: CIRI freedom of speech score (ciri)
9. `new_empinx`: CIRI Empowerment Rights Index (ciri)
10. `wecon`: CIRI Women's Economic Rights (ciri)
11. `wepol`: CIRI Women's Political Rights (ciri)
12. `wosoc`: CIRI Women's Social Rights (ciri)
13. `elecsd`: CIRI Electoral Self-Determination: (ciri)
14. `gdp.pc.wdi`: GDP per capita, current US$ (World Bank Development Indicators)
15. `gdp.pc.un`: GDP per capita, current US$ (UN data)
16. `pop.wdi`: WDI population mid-year estimates (World Bank Development Indicators)
17. `amnesty`: Political Terror Score measuring physical integrity rights violations worldwide, from Amnesty International
18. `statedept`: Political Terror Score measuring physical integrity rights violations worldwide, from Amnesty International
19. `milper`: Size of Military, (military personnel, thousands) from Correlates of War
20. `cinc`: Composite Index of Military Capabilities from Correlates of War

21. `domestic9`: Domestic Conflict / Stability Index, from Banks 2012.

First load your data.

```
rm(list=ls())
# makes the first column the rownames.
dat <- read.csv("countries.csv", row.names = 1)
```

Question 1. Computing PCA

We'll be using the `prcomp` function to conduct PCA. There are many other functions that compute PCA, such as `princomp`. They all have slightly different functionality.

1.1

Use `prcomp` to compute the principle components of the standardized data.

Hint: `prcomp` contains options that automatically `scale` and `center` our data for us.

```
pca1 <- prcomp(dat, scale = TRUE, center = TRUE)
```

1.2

The `summary` method describes the importance of the PCs. The first row describes the standard deviation (i.e., the square roots of the eigenvalue) associated with each PC. Larger numbers means this component is more "interesting" or "important" in the sense that it captures more variation.

The second row shows the proportion of the variance in the data explained by each component while the third row describe the cumulative proportion of explained variance.

Print `summary()` of your pca object and observe the result.

```
summary(pca1)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     2.9173 1.8600 1.6439 1.10713 1.07631 0.91289 0.78181
## Proportion of Variance 0.4053 0.1648 0.1287 0.05837 0.05516 0.03968 0.02911
## Cumulative Proportion  0.4053 0.5700 0.6987 0.75708 0.81225 0.85193 0.88104
##                           PC8     PC9    PC10    PC11    PC12    PC13   PC14
## Standard deviation     0.72948 0.64421 0.58703 0.55164 0.49341 0.46337 0.3995
## Proportion of Variance 0.02534 0.01976 0.01641 0.01449 0.01159 0.01022 0.0076
## Cumulative Proportion  0.90638 0.92614 0.94255 0.95704 0.96864 0.97886 0.9865
##                           PC15    PC16    PC17    PC18    PC19      PC20
## Standard deviation     0.32765 0.29011 0.24347 0.18215 0.01990 8.602e-16
## Proportion of Variance 0.00511 0.00401 0.00282 0.00158 0.00002 0.000e+00
## Cumulative Proportion  0.99157 0.99558 0.99840 0.99998 1.00000 1.000e+00
##                           PC21
## Standard deviation     2.409e-16
## Proportion of Variance 0.000e+00
## Cumulative Proportion  1.000e+00
```

1.3

Retrive the loadings and scores your pca object and observe the results.

Hint: Loadings are contained in the `rotation` item and scores are contained in the `x` item.

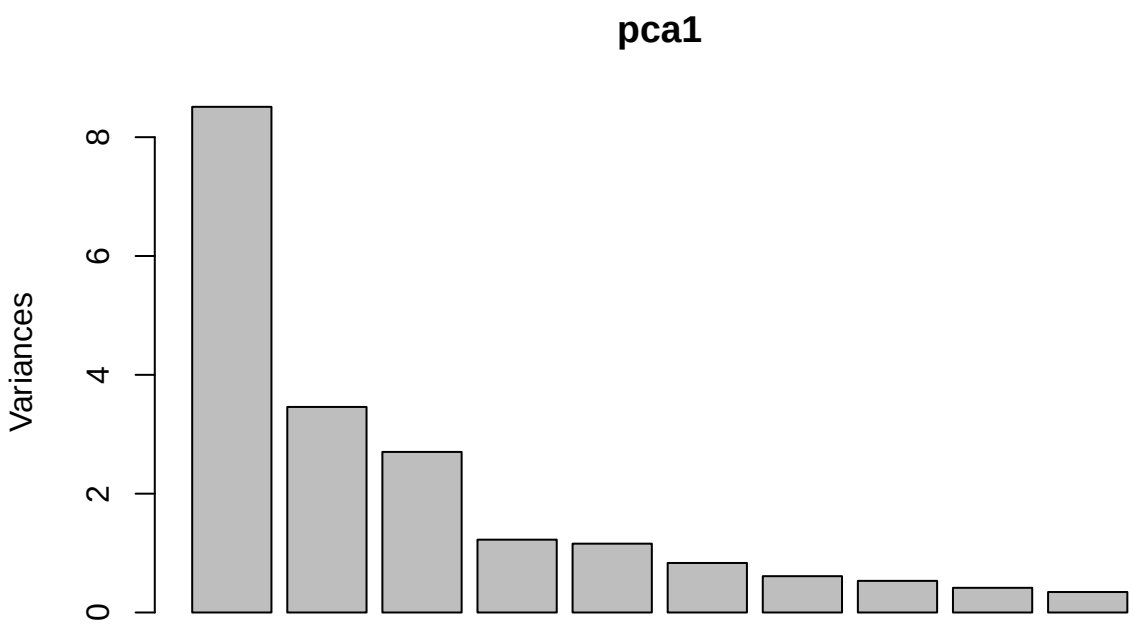
```
loadings <- pca1$rotation
loadings[1:5, 1:5]
```

```
##                   PC1        PC2         PC3         PC4         PC5
## idealpoint  0.2590509  0.0411834 -0.09502442 -0.00940566 -0.04815128
## polity      0.3024428 -0.2099553  0.07988403  0.03264443 -0.01088367
## polity2     0.3024428 -0.2099553  0.07988403  0.03264443 -0.01088367
## democ       0.3132604 -0.1661158  0.01740663  0.01322431  0.04334737
## autoc      -0.2643837  0.2505050 -0.15504195 -0.05538732  0.08075072
```

1.4

We can call `plot` on our PCA object, which returns a scree plot of the variances (y-axis) associated with the PCs (x-axis). Create a scree plot to decide how many PCs to retain for further analysis.

```
plot(pca1)
```

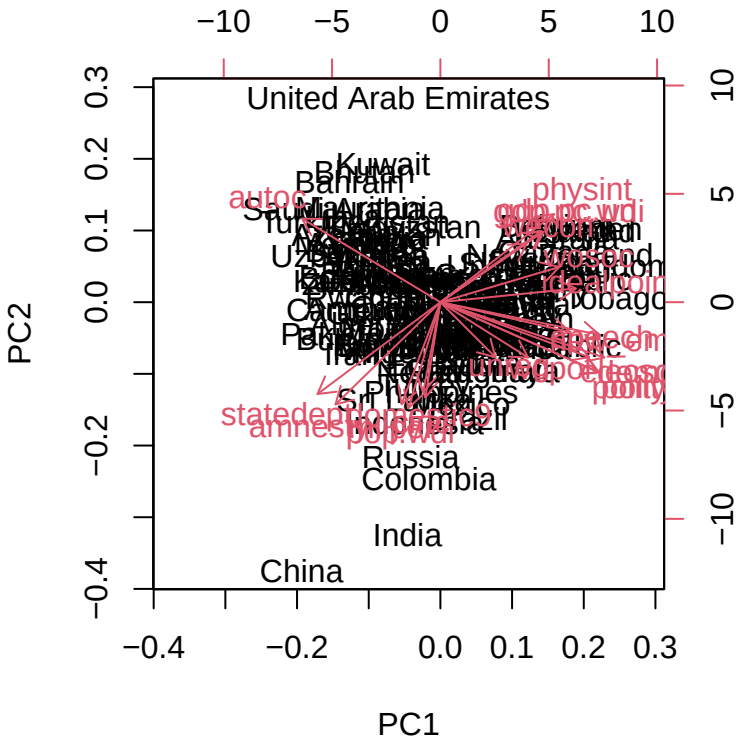


Question 2. Using PCA to visualize our data

2.1

Run `biplot` funcion with your pca object to visualize both PC loadings and scores.

```
biplot(pca1)
```



2.2: ggfortify

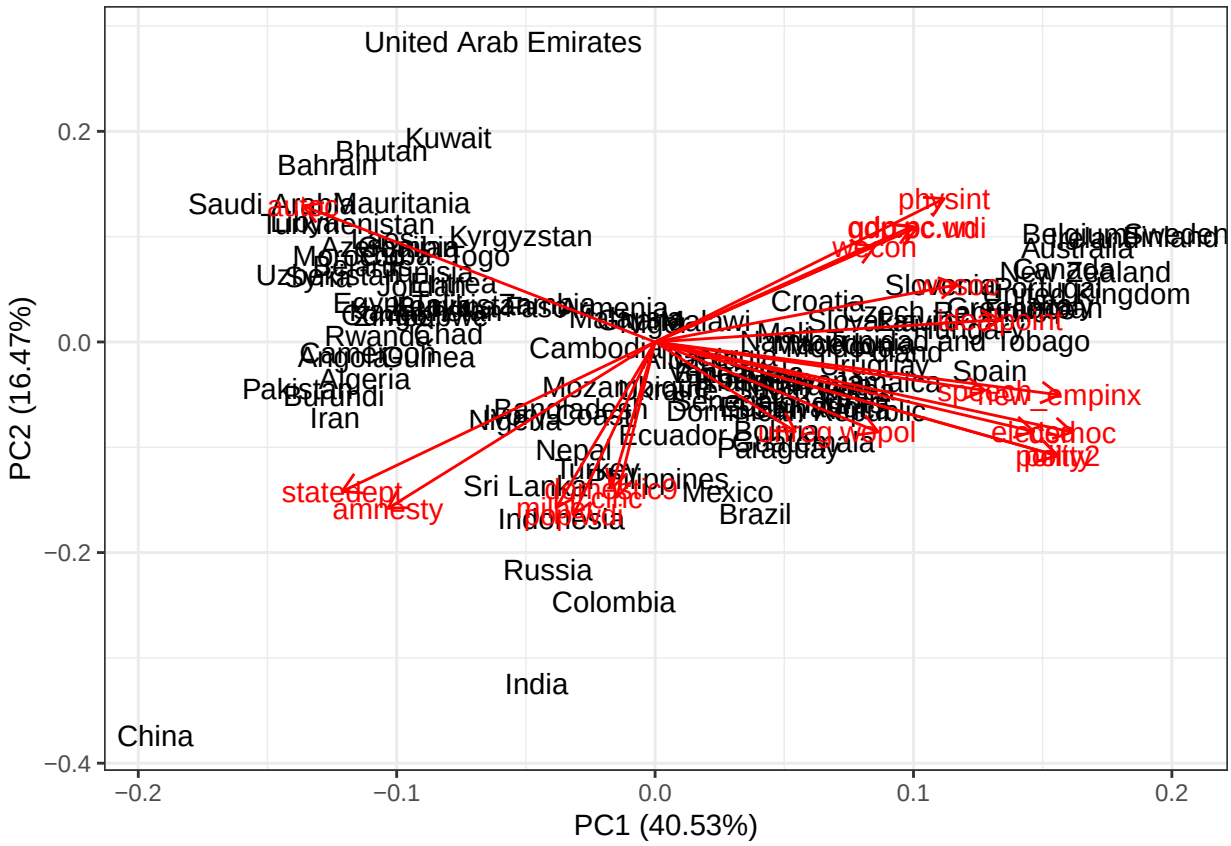
The `ggfortify` package lets `ggplot2` know how to interpret PCA objects. After installing and loading `ggfortify`, run the `autoplot` function with your pca object.

Hint: Set `shape = FALSE` to plot with labels instead of points and `loadings = TRUE` to include loadings in plot.

```
library(ggplot2)
library(ggfortify)
```

```
## Warning: package 'ggfortify' was built under R version 4.1.2
```

```
autoplot(pca1, shape=FALSE, loadings=TRUE, loadings.label = TRUE) +
  theme_bw()
```


United Arab Emirates
Kuwait
Bhutan
Bahrain
Mauritania
Kyrgyzstan
Togo
Rwanda
Chad
Guinea
Algeria
Burundi
Iran
Nepal
Sri Lanka
Indonesia
Mexico
Brazil
Russia
Colombia
India
China
Croatia
Spain
Australia
Canada
New Zealand
Trinidad and Tobago
physint
wdi
wecon
statedept
amnesty
spanish_empinx
PC1 (40.53%)
PC2 (16.47%)
0.2
0.0
−0.2
−0.4
−0.2
−0.1
0.0
0.1
0.2