HW8 Questions

Sruti Kanthan

Problem Set 8: Unsupervised Machine Learning

```
library(tidyverse)
library(ggplot2)
library(dplyr)
library(skimr)
library(dendextend) # for "cutree" function
library(ggdendro)
library(patchwork)
library(GGally)
library(tictoc)
library(factoextra)
library(clValid)
library(here)
library(sparcl)
library(ggthemes)
```

Data Mining (Waggoner, SISRM 2020)

Answer the following questions to the best of your ability. Be sure to show *all* code in-line, in addition to full written responses to each of the questions. Write in complete sentences where appropriate. **A complete submission is a single rendered PDF document**.

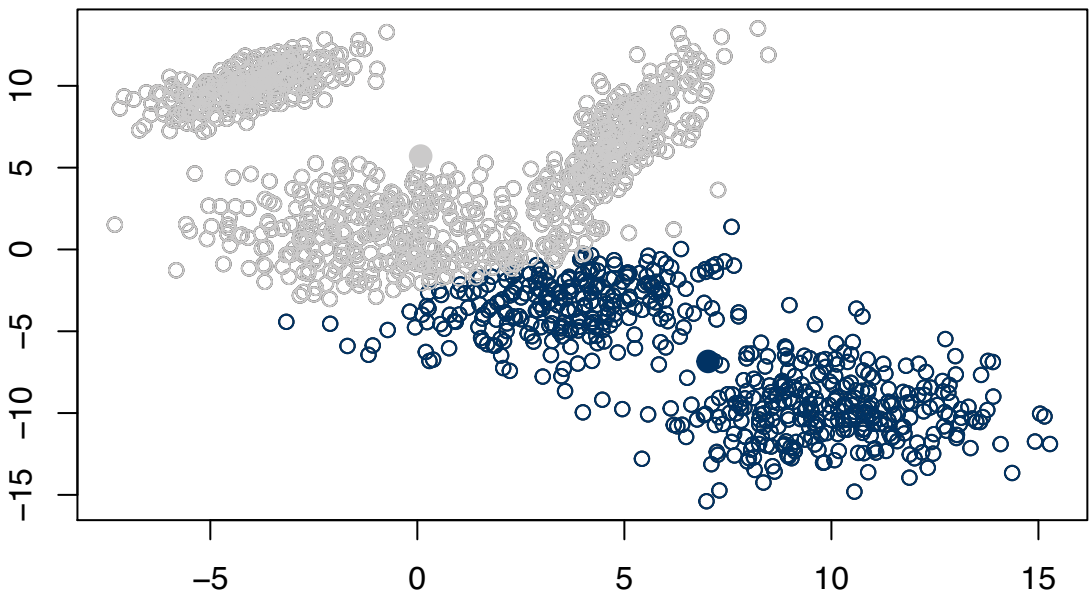
For question 1, use the following simulated data.

```
x <- list(
  `1` = MASS::mvrnorm(n = 300, c(-4,10), matrix(c(1.5,1,1,1.5),2)),
  `2` = MASS::mvrnorm(n = 300, c(5,7), matrix(c(1,2,2,6),2)),
  `3` = MASS::mvrnorm(n = 300, c(-1,1), matrix(c(4,0,0,4),2)),
  `4` = MASS::mvrnorm(n = 300, c(10,-10), matrix(c(4,0,0,4),2)),
  `5` = MASS::mvrnorm(n = 300, c(3,-3), matrix(c(4,0,0,4),2))
) %>%
  map_df(as_tibble, .id = "cluster") %>%
  rename(x = V1,
         y = V2)

x <- x %>%
  as_tibble() %>%
  select(x,y)
```

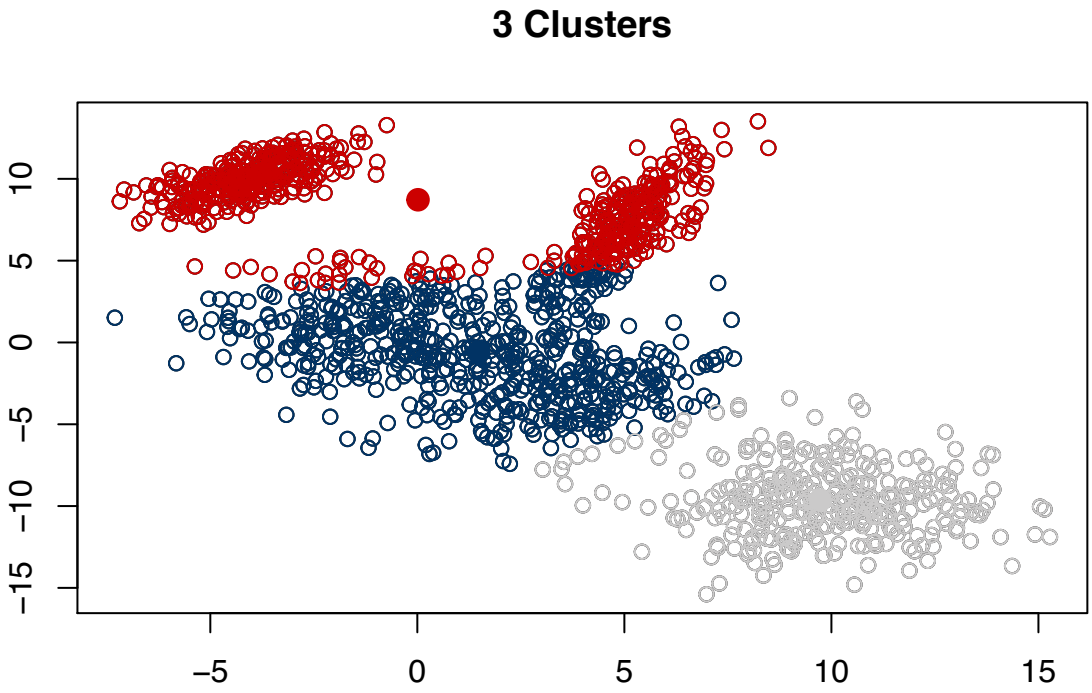
1. (10 points) k-means will give you however many clusters you search for, whether these are meaningful or not. Explore this. Using the simulated data, `x`, fit 6 different k-means models, with k ranging from 2 to 7 and initialized at `nstart = 20`. Plot your configurations varying the color based on `k` and note how configurations change, regardless of any substantive meaning.

```
set.seed(666)

k5 <- kmeans(x, 5, nstart=20)
k6 <- kmeans(x, 6, nstart=20)
k7 <- kmeans(x, 7, nstart=20)
```

```
colors <- amerika::amerika_palette(3,
                                   name = "Dem_Ind_Rep3",
                                   type = "discrete")

kmeans_plot <- function(M, C, title) {
  plot(M, main = title, xlab = "", ylab = "")
  for(i in 1:k) {
    points(C[i, , drop = FALSE], pch = 19, cex = 1.5, col = colors[i])
    points(M[A == i, , drop = FALSE], col = colors[i])
  }
}



k<-2
k2 <- kmeans(x, 2, iter.max=100, nstart=20)
A=k2$cluster


kmeans_plot(x, k2$centers, "2 Clusters")
```

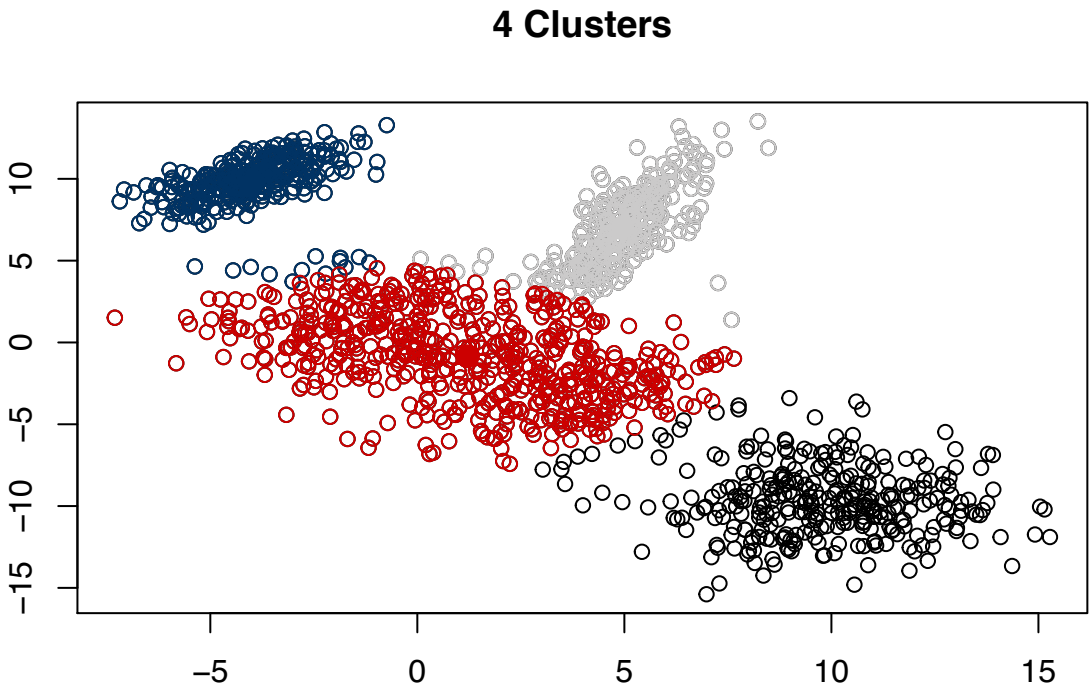



```
k<-3
k3 <- kmeans(x, 3, iter.max=100, nstart=20)
A=k3$cluster


kmeans_plot(x, k3$centers, "3 Clusters")
```



It doesn't look like 3 clusters is enough - the blue is split across two clusters.

```
k<-4
k4 <- kmeans(x, 4, iter.max=100, nstart=20)
A=k4$cluster


kmeans_plot(x, k4$centers, "4 Clusters")
```



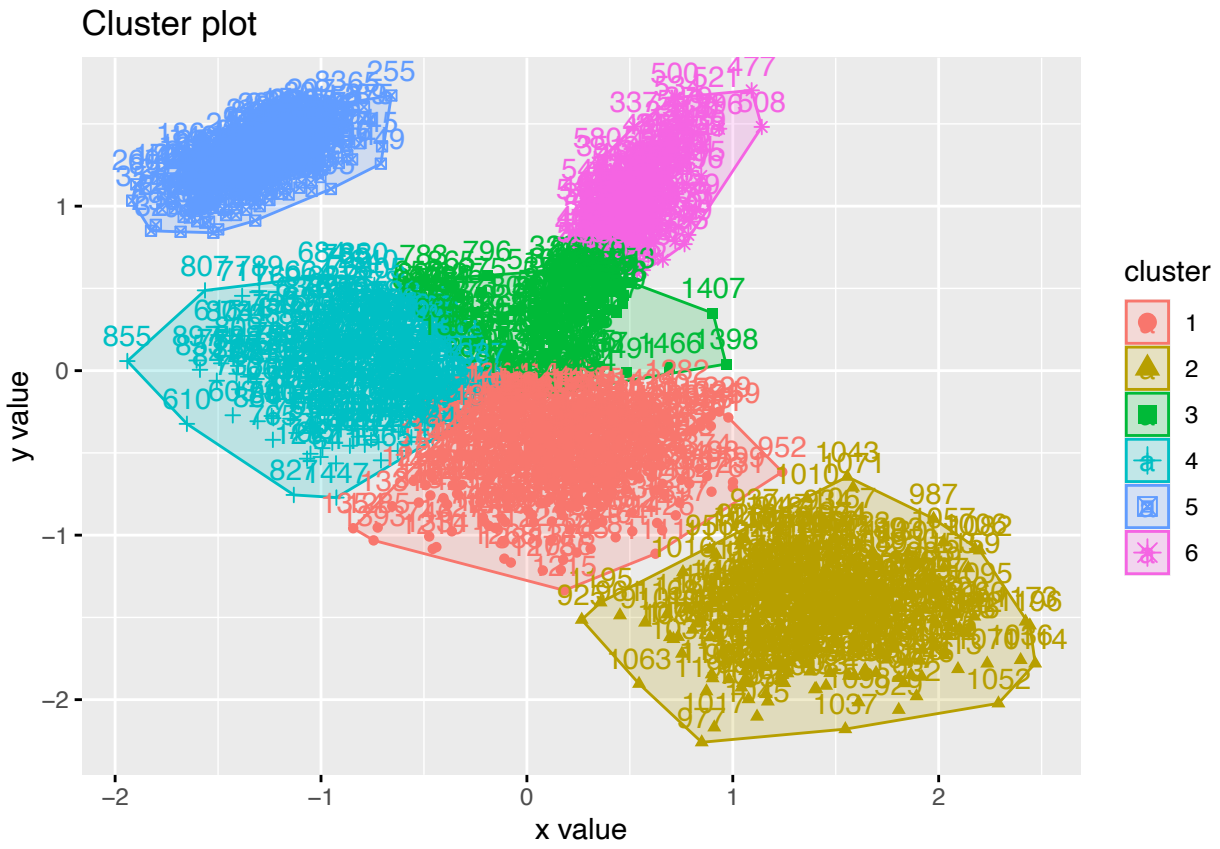
It looks like four clusters might be adequate - one color for each cluster, and it doesn't seem like there's a cluster that isn't being covered.

There weren't 5+ colors in the palette, so I used fviz for k=5:7:

```
fviz_cluster(k5, x)
```

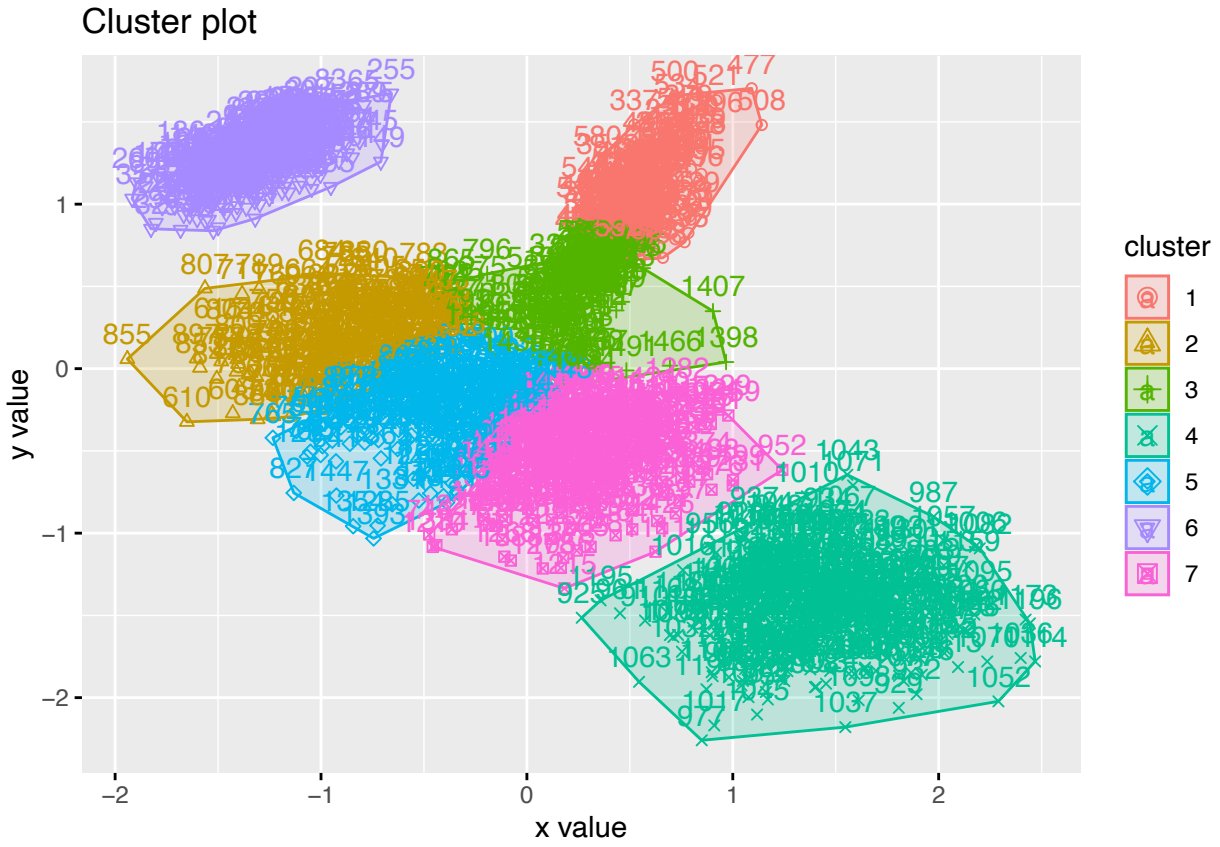
It looks like k=5 might actually work, sicne there's little no no overlap between the clusters - the red cluster from k=4 was just split up into two subclusters.

```
fviz_cluster(k6, x)
```



k=6 might work, since I don't see too much overlap in the clusters. However, I think the 4-5 cluster range still suffices- maybe k=5 is a happy medium.

```
fviz_cluster(k7, x)
```



We can start to see overlap when k=7 clusters.

For questions 2-6, use the `USArrests` data (preloaded in R) for another look at k-means, but with real data.

```
df <- USArrests %>%
  na.omit %>%
  scale()

head(df)
```

```
##               Murder   Assault   UrbanPop         Rape
## Alabama    1.24256408 0.7828393 -0.5209066 -0.003416473
## Alaska     0.50786248 1.1068225 -1.2117642  2.484202941
## Arizona    0.07163341 1.4788032  0.9989801  1.042878388
## Arkansas   0.23234938 0.2308680 -1.0735927 -0.184916602
## California 0.27826823 1.2628144  1.7589234  2.067820292
## Colorado   0.02571456 0.3988593  0.8608085  1.864967207
```

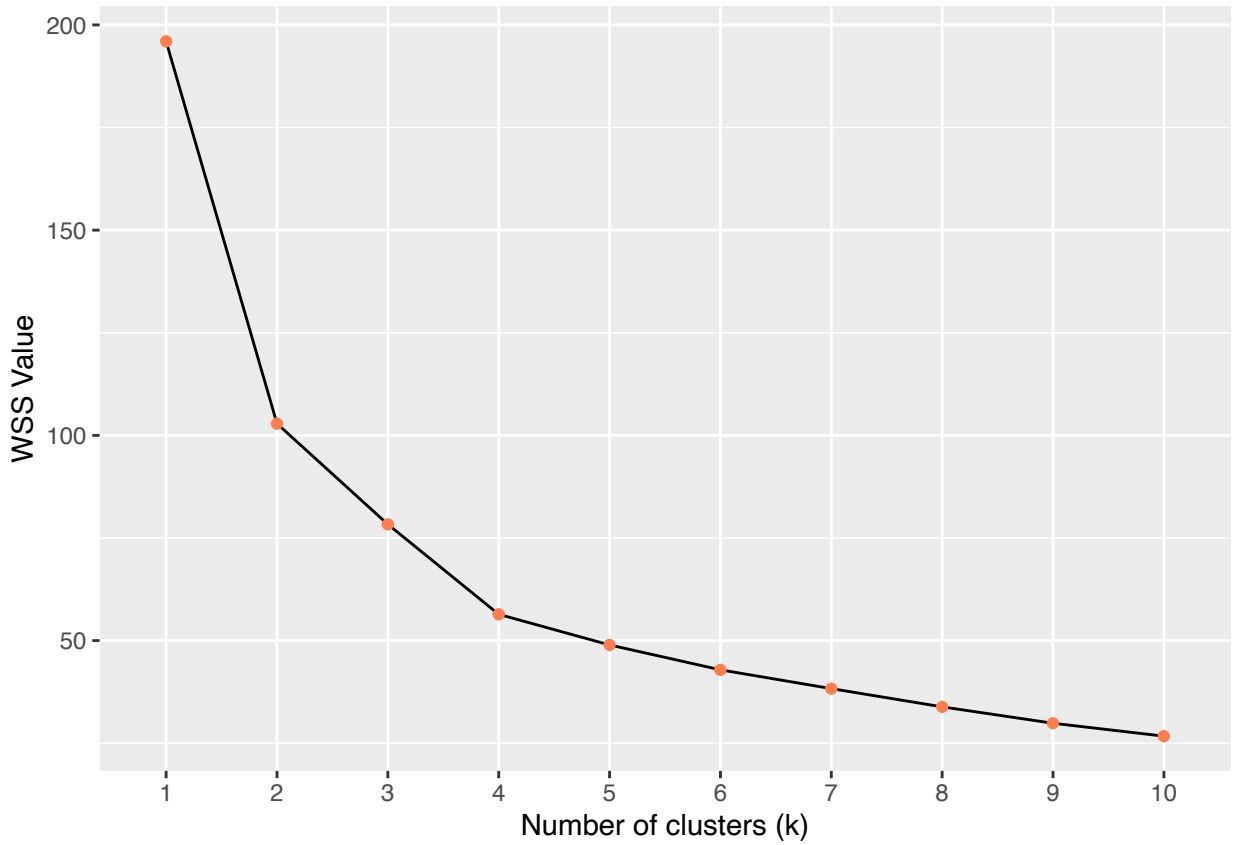
2. (5 points) Building on what we learned about internal validation (implemented via the `clValid` package in class), write your own function to calculate *total within-cluster sum of squares* over a range of values for `k`.

```
wcss_find <- function(data,k_end) {
```

```
wss <- c(length(k_end))
  for (i in 1:k_end) {
    wss[i] <- sum(kmeans(data, centers=i, nstart=20)$withinss)
  }
# turn wss data into a data frame for plotting
wss_data <- as.data.frame(wss) %>%
  mutate(k=row_number())

return(wss_data)

}
```

3. (10 points) Using your helper function created in the previous question, plot the within sums of squares for all k-means models, with k ranging from 1 to 10. In a few sentences, discuss what you find.

```
wcss_plot <- wcss_find(df, 10) %>%
 mutate(k=as.factor(k)) %>%
 rename("Number of clusters (k)"=k,
         "WSS Value"=`wss`)

ggplot(wcss_plot, aes(x=`Number of clusters (k)`, y=`WSS Value`, group=1))+
  geom_line() +
  geom_point(color="coral") +
  scale_x_discrete()
```



My function took 2 parameters, a dataframe and k_end value. I set an empty vector to the size of length(k_end), and then created a for loop that calculated kmeans from 1 through k_end, used those values to generate wcss, and then saved that information into a dataframe that the function returned.

It looks like as k increases, the wss decreases consistently (from 196 to ~26). Through the elbow method, we should be choosing 4 clusters. The break in the elbow isn't very pronounced. I've seen in some plots that the k might increase back up a little bit before going down, but this looks like it's decreasing consistently. I had to make k a factor for the numbers to order correctly on the plot.

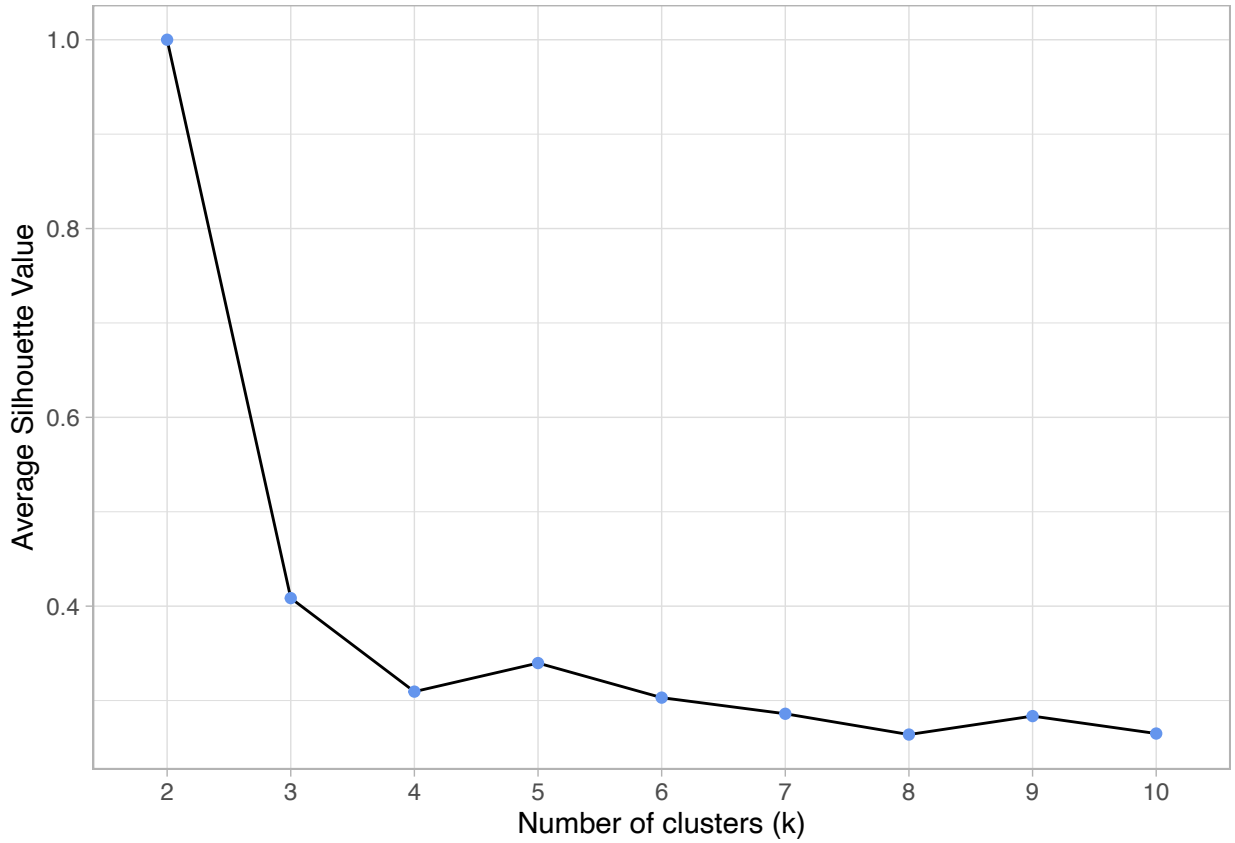
4. (5 points) Building on what we learned about internal validation (implemented via the `clValid` package in class), write your own function to calculate the *average silhouette width* over a range of models varying by `k` (*hint*: consider `silhouette()`).

```
sil_find <- function(data,k_end) {

sil_mean <- c(length(k_end))

  for (i in 2:k_end) {
    k_means <- kmeans(data, centers=i, nstart=20)
    sil <- silhouette(k_means$cluster, dist(data))
    sil_mean[i] <- mean(sil[, 3])
  }
# turn wss data into a data frame for plotting
s_data <- as.data.frame(sil_mean) %>%
  mutate(k=row_number()+1)
```

```
return(s_data)

}
```

5. (10 points) Using your helper function created in the previous question, plot the average silhouette width for all k-means models, with k ranging from 2-10. In a few sentences, discuss what you find.

```
sil_plot <- sil_find(df, 9) %>%
 mutate(k=as.factor(k)) %>%
 rename("Number of clusters (k)"=k,
        "Average Silhouette Value"=`sil_mean`)

ggplot(sil_plot, aes(x=`Number of clusters (k)`, y=`Average Silhouette Value`, group=1))+
  geom_line() +
  geom_point(color="cornflowerblue") +
  scale_x_discrete() +
  theme_light()
```



My silhouette function is essentially the same as my kmeans function, except that I had to make sure to only select a specific section of the mean output for the returned data given the way that the silhouette function is structured. I also had to save silhouette as an object and calculate the mean sil after, which was one additional step.

It looks like silhouette is generally decreasing, with a small uptick at k=5. The highest value is 1 and then it drops down with each additional cluster. Through this method, we should be choosing a k of 2, since that has the highest y value - the next highest values are for k=3 and k=5.

6. (5 points) Discuss the optimal model (value of `k`) for both within sum of squares and average silhouette width. Do these metrics give the same results? Different? If this were to happen in your own research, how would you proceed and why?

The optimal value for k is 4 for wcss and technically 2 for silhouette, though we do see a small uptick for k=5. So, the optimal values differ between the metrics. If this were to happen in my own research, I would probably look at both plots and then look at the data directly to see which optimal value assessment I agree with more. In this case, I think that wcss is closer to being correct - k=4 does seem like a fine choice, especially if the research question is looking for broad categories that emerge. That being said, I wonder if the small uptick at k=5 is a sign that k=5 would have also been a good option. I wonder if running iterations of k means and looking at error rates by cluster size could also shed light on which would've been the better option, but I am not sure about this. It really would depend on whether I'm interested in finding subcategories (would go for k=6 here) or seeing general categories (would choose k=4 most likely).

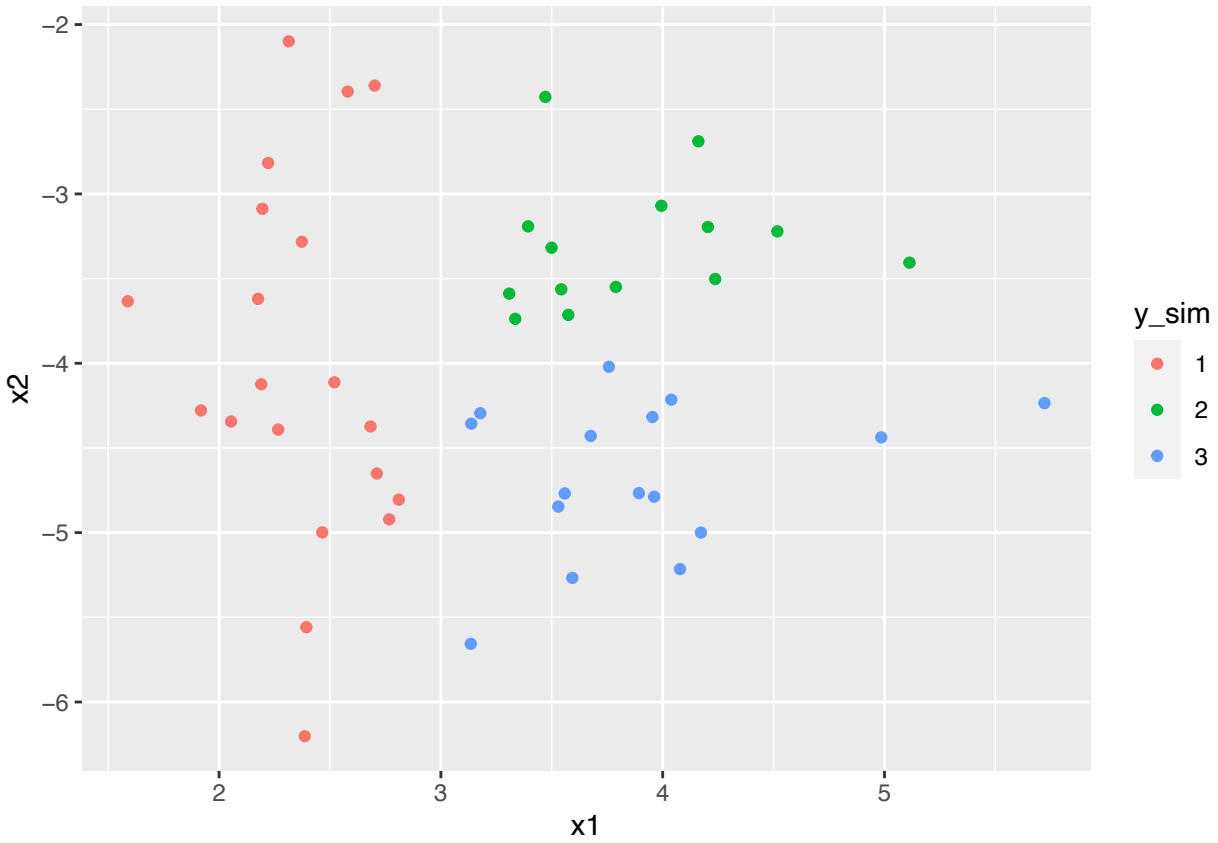
For questions 7-10, use the simulated data.

I renamed the variable x, since we already had one of the same name earlier in the homework. I also renamed the other vars to `x1`, `x2`, and `ysim`:

```
x_sim <- tibble(x1 = rnorm(50) + 3,
            x2 = rnorm(50) - 4,
            y_sim = ifelse(x1 < 3, "1",
                       ifelse(x2 > -4, "2", "3")))
```

7. (5 points) Get to know the data. Plot the continuous features from the simulated data object `x`, with color varying by values of `y`. Discuss the patterns you see (e.g., think about natural groupings, spread of the data and so on).

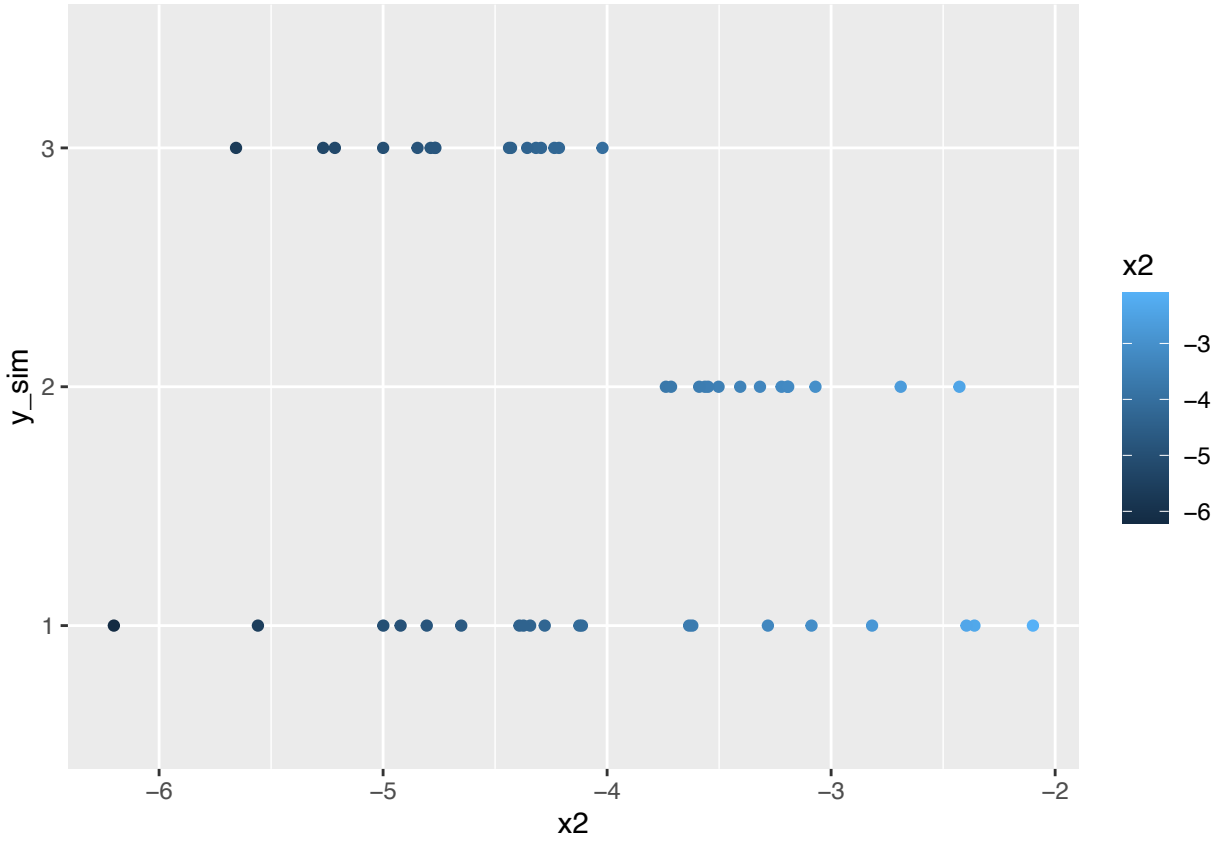
```
ggplot(data=x_sim, aes(x=x1, y=x2, color=y_sim)) +
  geom_point()
```



Plotting x1 and x2, filled in by y_sim.

The data appear to range from x1= -0.5<x1<6 or so, and x2 = -6.5<x<-3 ; and, y_sim is 1, 2, or 3. It looks like the y_sims are clustering such that the 1's are to the left, the 2's are to the top right, and the 3's are just below the 2's. This makes sense, given how the data were generated; y_sim values contingent on x1 and x2's values, in levels. I'm seeing potential outliers at the far right (y_sim =3) and bottom of the graph (y_sim=1).

```
ggplot(data=x_sim, aes(x=x2, y=y_sim, color=x2)) +
  geom_point()
```



(Just another way to view the data, we can see the 3 groupings this way too with x_2 plotted against y_sim and filled by x1)

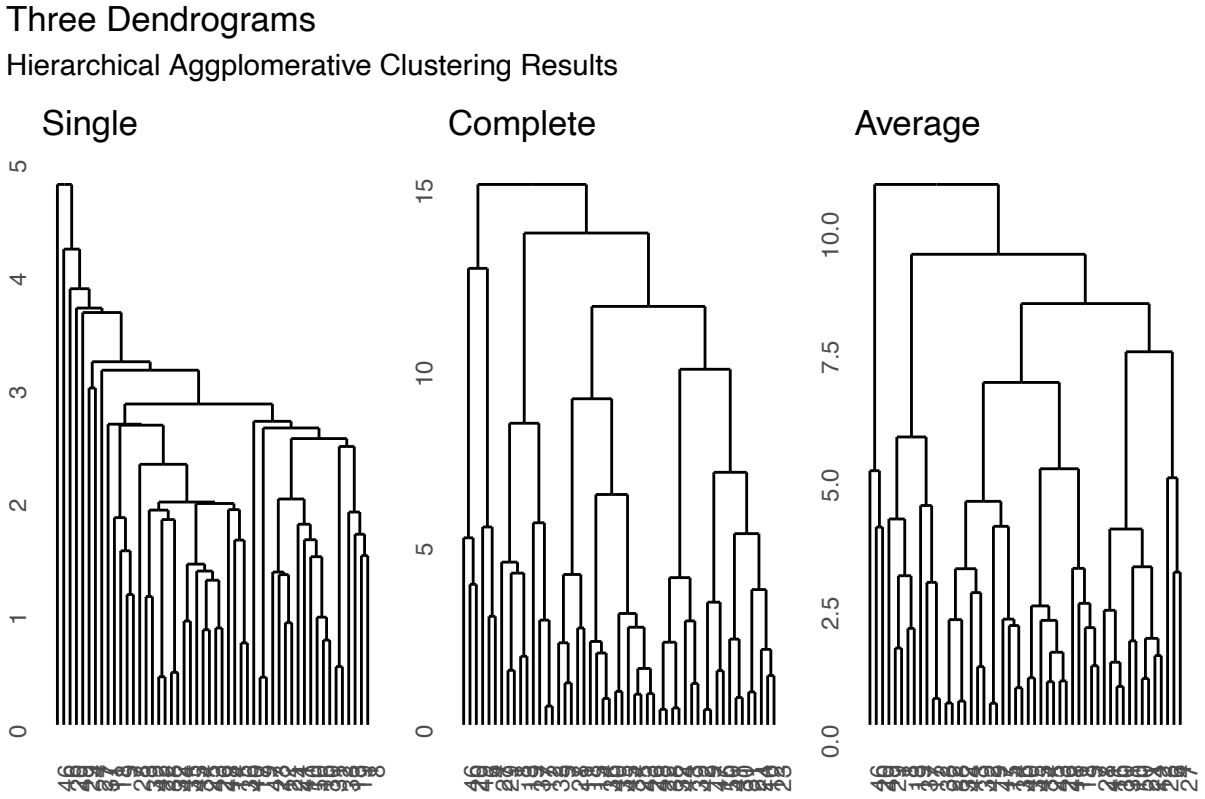
8. (10 points) Fit 3 different hierarchical agglomerative clustering models for each of the following linkage methods: `complete`, `single`, and `average`. Visualize these models as dendrograms. Discuss a few differences you see across the models (you may reference observation numbers to answer this, as these are simulated data).

```
x_scale <- x_sim[1:2] %>%
  scale()
```

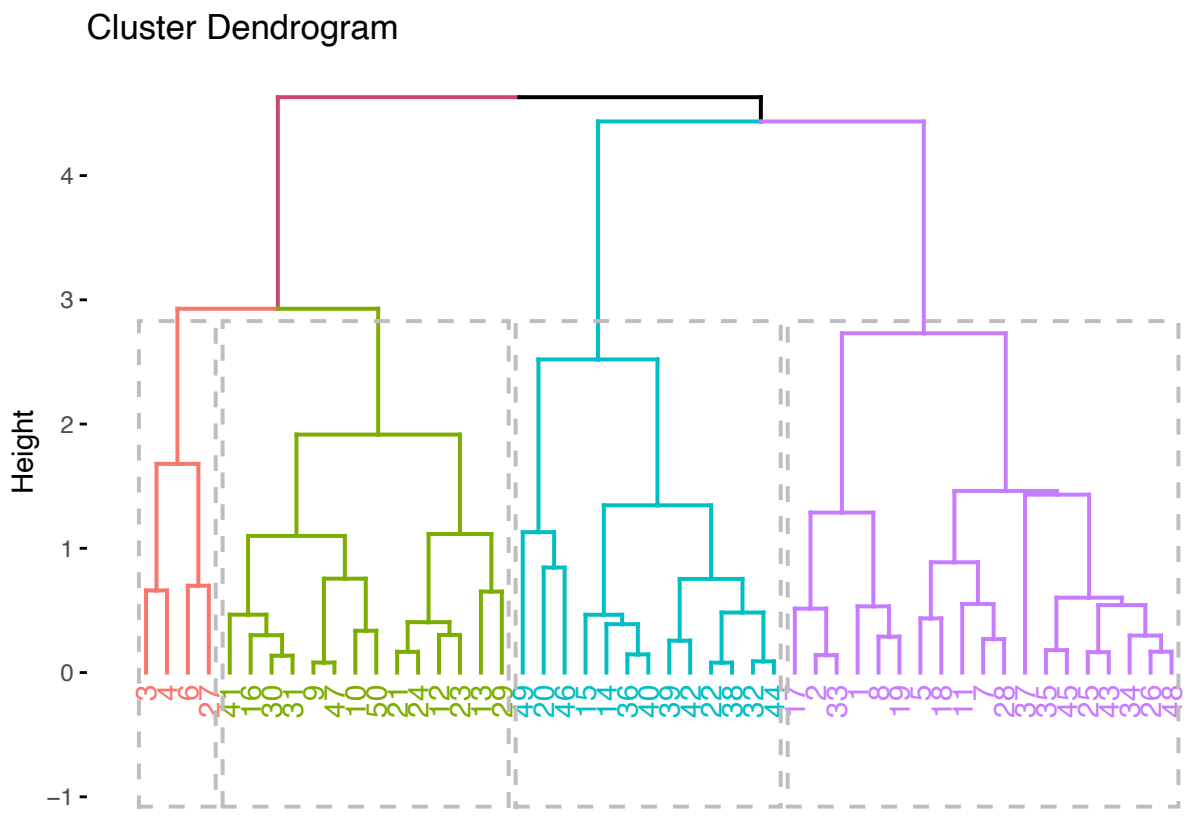
```
set.seed(666)

x_sim_dend <- x_sim %>% #scaling first
  select(-y_sim) %>%
  drop_na()%>%
  scale() %>%
  dist()

x_dist <- x_sim_dend %>%
  dist(method="euclidean")

h1<- hclust(x_dist, method = "single")
```

```r
hc_single <- ggdendrogram(h1) + labs(title = "Single")

h2 <- hclust(x_dist,
             method = "complete")
hc_complete <- ggdendrogram(h2) + labs(title = "Complete")

h3 <- hclust(x_dist,
             method = "average")
hc_average <- ggdendrogram(h3) + labs(title = "Average")
```

```r
plots <- hc_single +
  hc_complete +
  hc_average


plots + plot_annotation(
  title = 'Three Dendrograms',
  subtitle = 'Hierarchical Aggplomerative Clustering Results')
```



Given that longer branches typically indicate a cluster - I see three long branches in the 'complete' (15 to 10 or so along Y) and 'average' dendrograms (~9 to 6). There are some relatively long branches within these clusters, though, which suggests that the clusters are not compact and there's some space between the points. This matches what we saw when we plotted the data - the three clusters were relatively spread out. The single plot appears to have more outliers compared to the rest (clades with just one leaf) - it also seems to have fewer merges/way more leaves. Average seems to have the second most outliers, while complete has the least.



According to this visualization, there are actually six clusters. I used this website to confirm this count. These clusters represent how close certain points of the data are together, while the clusters themselves have a broad spread.

Use the `global_literacy_rates.csv` data set for questions 11-12. The data file contains global literacy rate data by country, region, gender, year, and include other useful features. First run the following code I wrote to preprocess the data *prior to answering the questions that follow.*

```
lit <- here("data", "global_literacy_rates.csv") %>%
  read_csv()

# region -> numeric
lit$region[lit$Region == "Central and Southern Asia"] <- 1
lit$region[lit$Region == "Eastern and South-Eastern Asia"] <- 2
lit$region[lit$Region == "Europe and Northern America"] <- 3
lit$region[lit$Region == "Latin America and the Caribbean"] <- 4
lit$region[lit$Region == "Northern Africa and Western Asia"] <- 5
lit$region[lit$Region == "Oceania"] <- 6
lit$region[lit$Region == "Sub-Saharan Africa"] <- 7

# age -> numeric
lit$age[lit$Age == "15+"] <- 1
lit$age[lit$Age == "15-24"] <- 2
lit$age[lit$Age == "25-64"] <- 3
lit$age[lit$Age == "65+"] <- 4

lit$year <- lit$Year
```

```
# gender -> numeric
lit$male[lit$Gender == "female"] <- 1
lit$male[lit$Gender == "male"] <- 2
lit$male[lit$Gender == "total"] <- NA

# literacy rate -> numeric
lit$lit_rate <- as.numeric(sub("%", "", lit$`Literacy rate`))

literacy <- lit %>%
  select(region, age, year, male, lit_rate) %>%
  drop_na()
```

11. (10 points) Perform PCA on the `literacy` data and use `ggplot2`-style tools (i.e., not `biplot()`) to visualize the observations on the first *and* second principal components. Describe your results, e.g., Are features strongly correlated? If so, along which component? If not, why do you think?

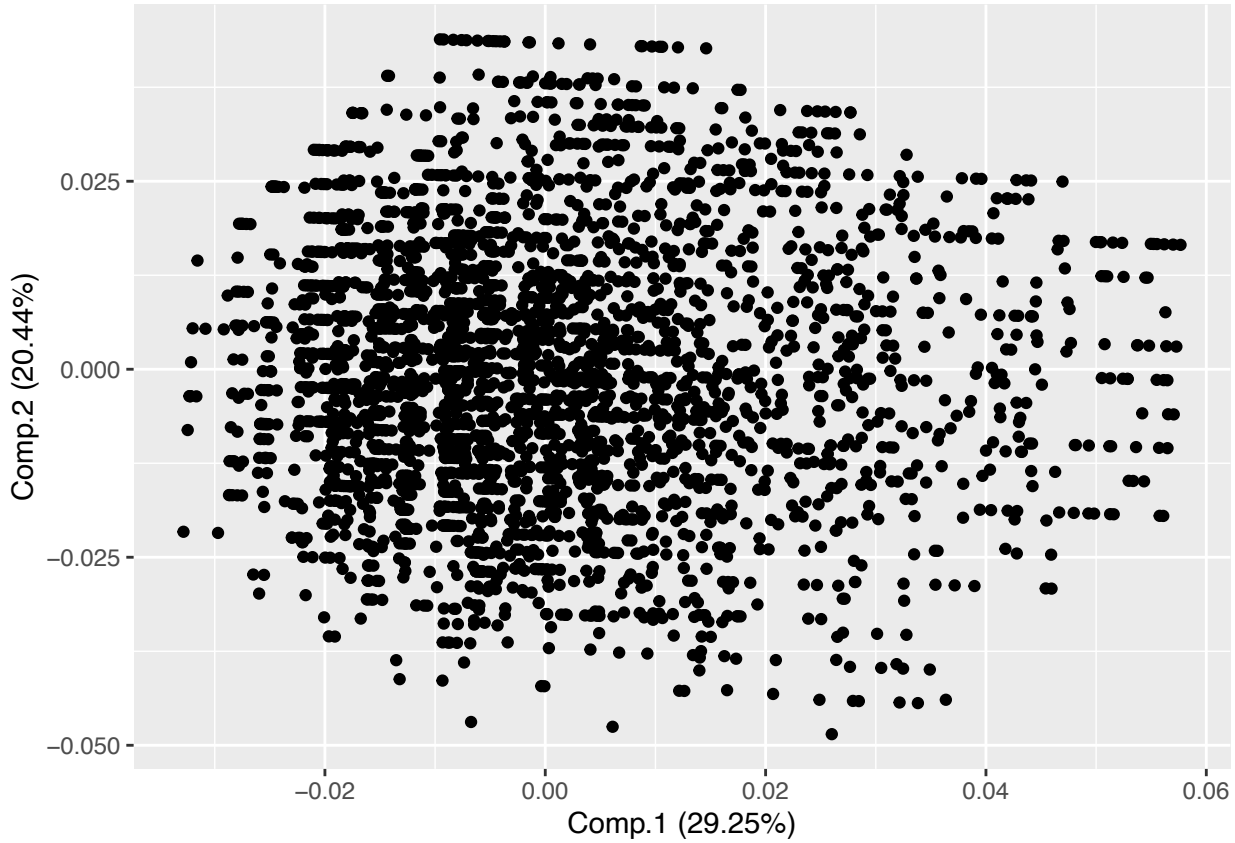
```
literacy_as_tibble <- literacy %>%
  as_tibble()

pca_fit <- literacy %>%
  scale() %>%
  prcomp()

summary(pca_fit)
```

```
## Importance of components:
##                           PC1    PC2    PC3    PC4    PC5
## Standard deviation     1.2093 1.0109 0.9998 0.9881 0.7347
## Proportion of Variance 0.2925 0.2044 0.1999 0.1953 0.1080
## Cumulative Proportion  0.2925 0.4969 0.6968 0.8920 1.0000
```

```
literacy_as_tibble <- literacy %>%
  as_tibble()

pca_fit <- literacy %>%
  scale() %>%
  princomp(cor=TRUE)

summary(pca_fit)
```

```
## Importance of components:
##                            Comp.1    Comp.2    Comp.3    Comp.4    Comp.5
## Standard deviation     1.2093017 1.0108550 0.9998194 0.9880676 0.7347416
## Proportion of Variance 0.2924821 0.2043656 0.1999278 0.1952555 0.1079690
## Cumulative Proportion  0.2924821 0.4968477 0.6967755 0.8920310 1.0000000
```

```
library(ggfortify)

autoplot(pca_fit, data = literacy)
```



General distribution (below, points are colored by age)

```
data_for_plot <- literacy_as_tibble %>%
  mutate(PC1 = pca_fit$x[, 1],
         PC2 = pca_fit$x[, 2])
```

```
#data_for_plot %>%
#  ggplot(aes(PC1, PC2, color=as.factor(age))) +
#  geom_point() +
#  stat_ellipse() +
#  labs(title = "PCA Solution",
#       x = "PC1",
#       y = "PC2") +
#  theme_minimal()
```

For some reason, the code above wouldn't plot when I was trying to knit the document a second time - kept saying "PC1 wasn't found", even though I defined it in the previous cell. Here is what that graph looked like. I had originally made plots by age (shown) and gender.

```
library(FactoMineR)

contrib <- PCA(literacy, scale.unit = TRUE, ncp = 5, graph = TRUE)
```

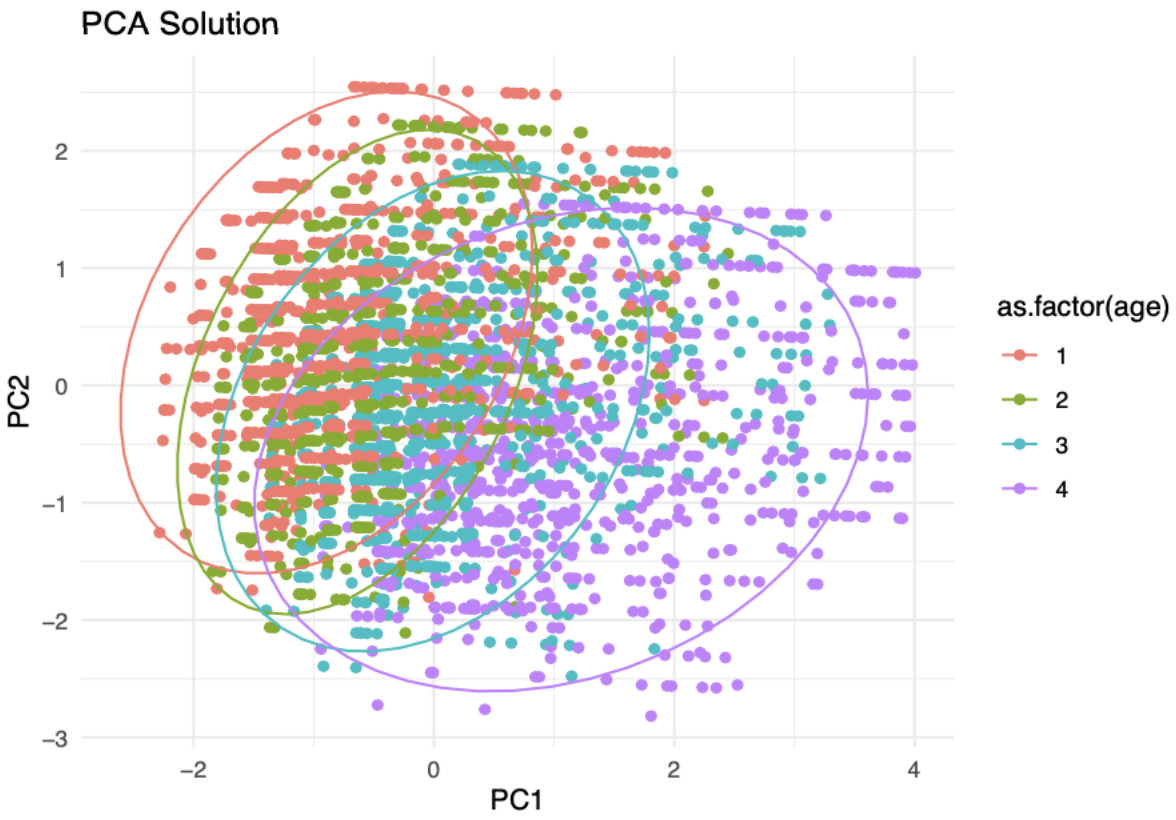
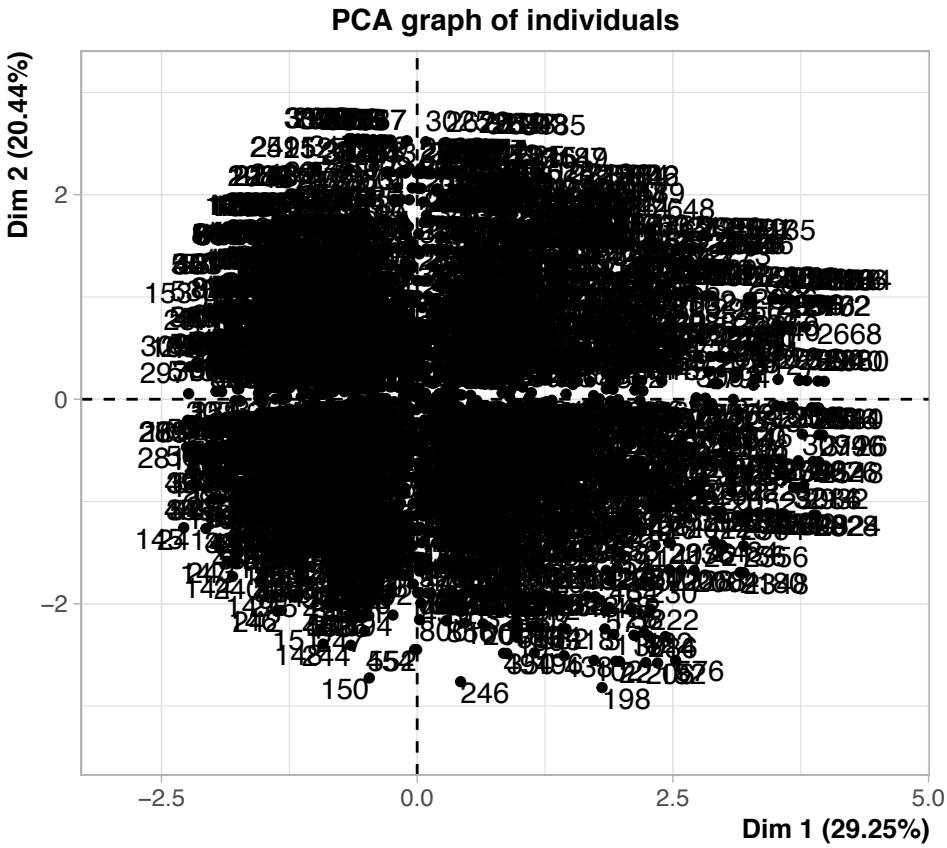
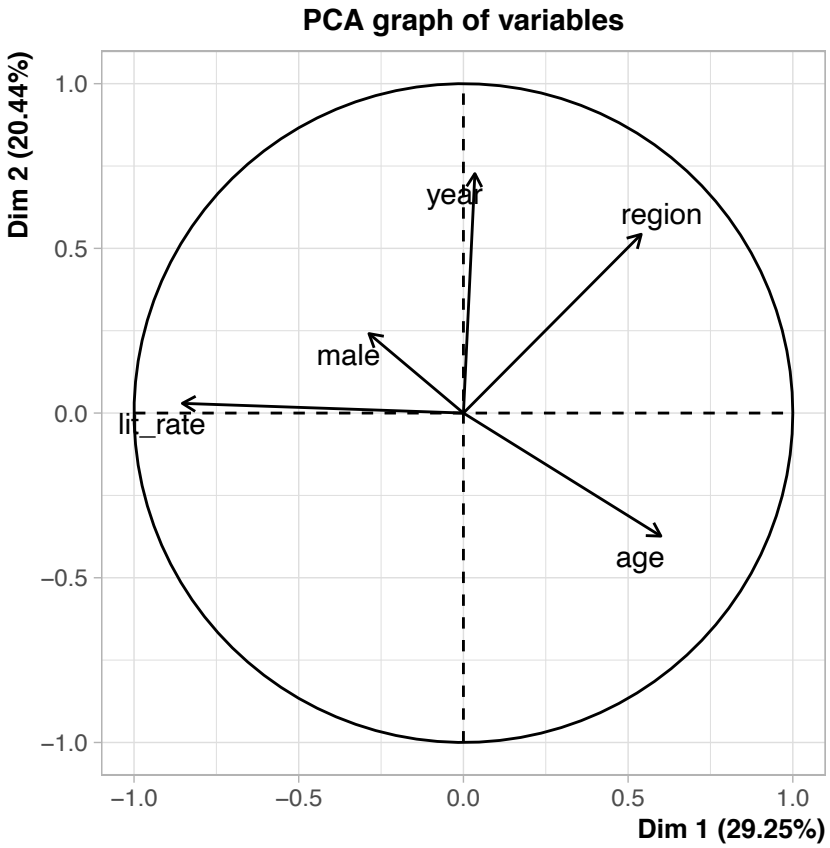


Figure 1: pca


PCA graph of individuals
Dim 2 (20.44%)
Dim 1 (29.25%)
2
0
−2
−2.5
0.0
2.5
5.0
150
246
198



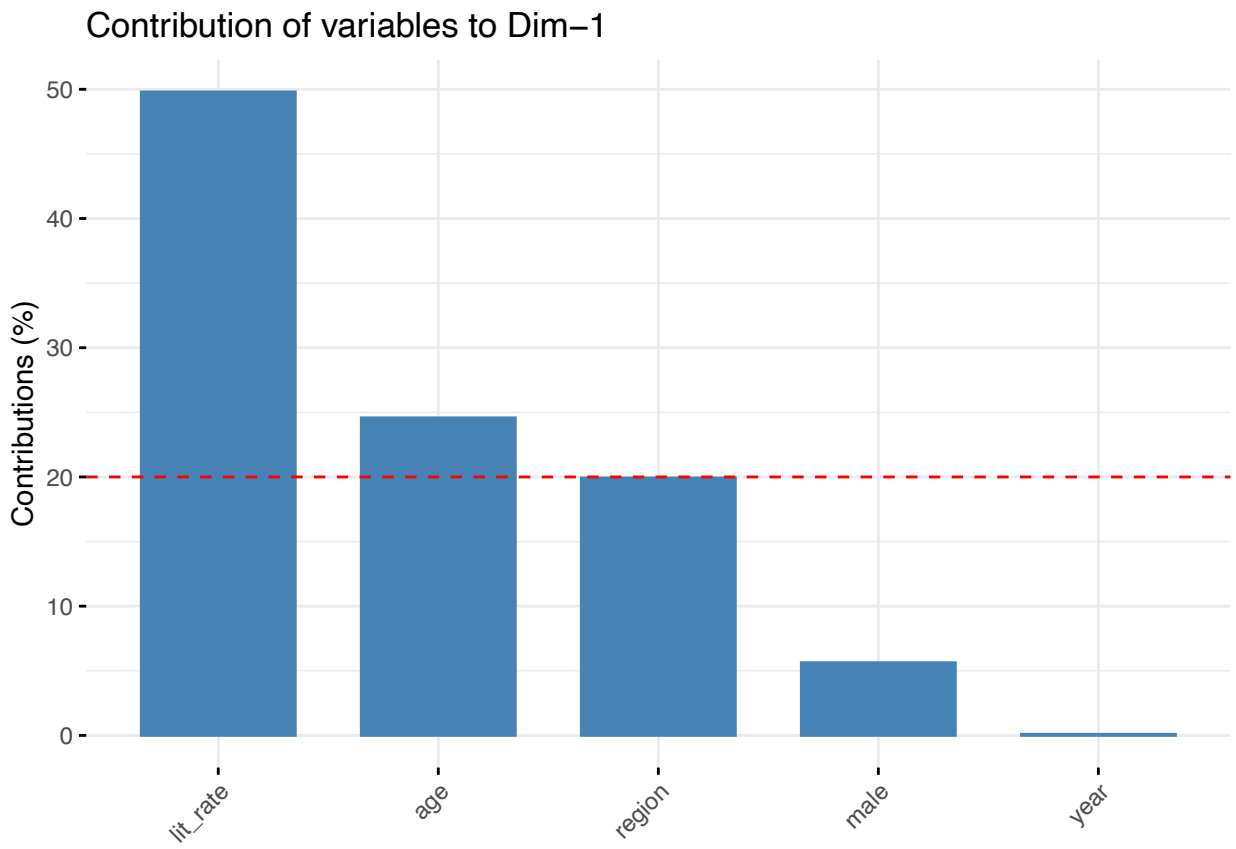

```
contrib$var$contrib
```

```
##                Dim.1     Dim.2        Dim.3       Dim.4       Dim.5
## region   19.92074921 28.841129 1.170063e-02 31.46188150 19.76453921
## age      24.57425299 13.656682 1.941789e+01 17.94921628 24.40195703
## year      0.08262656 51.719119 3.927251e-01 47.73010297  0.07542662
## male      5.62371156  5.698149 8.017763e+01  2.78546540  5.71504763
## lit_rate 49.79865968  0.084921 5.594057e-05  0.07333386 50.04302951
```

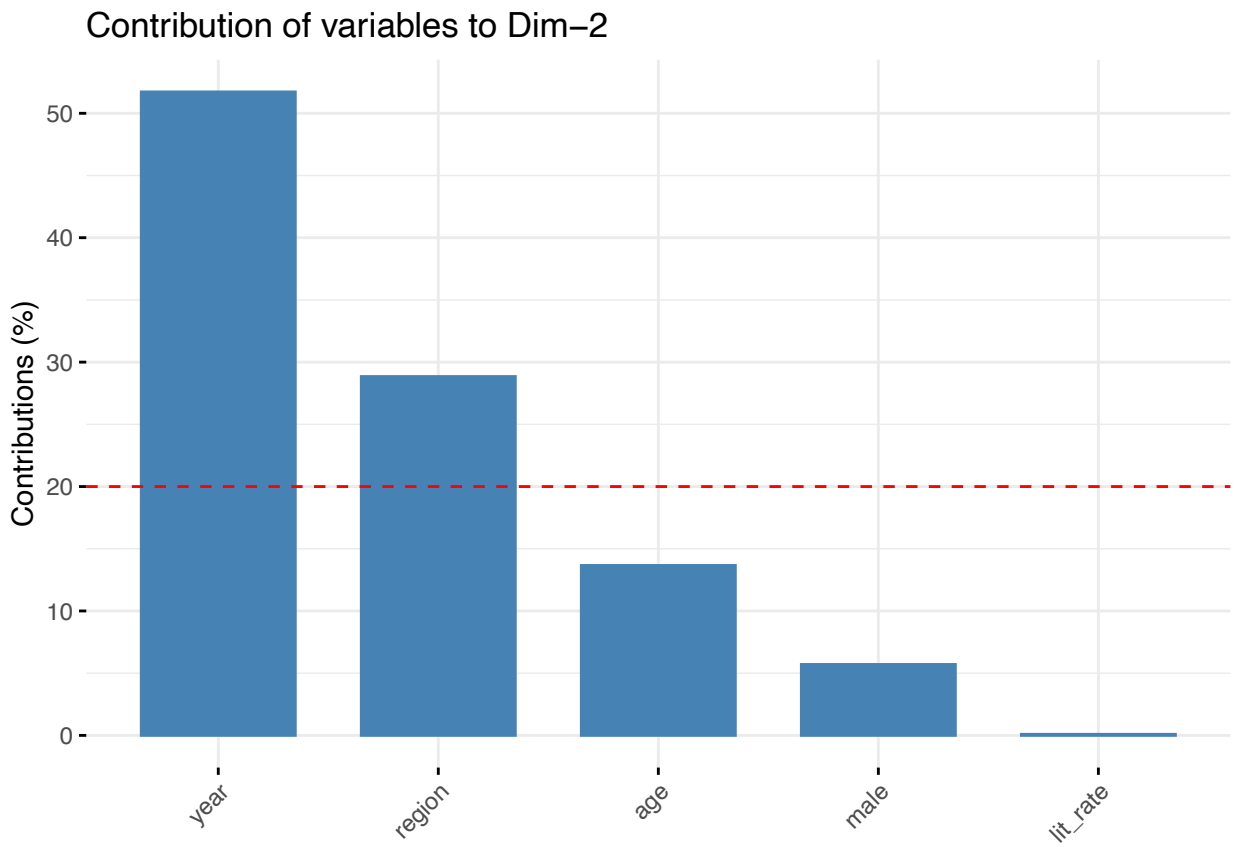
The larger the contribution, the more the variable contributes to the component. The most variance could be explained by the first and second components (as further corroborated by the plots in #12), so I will focus on those.

I'll use functions from fviz to create a barplot of how much each variable contributes to PC1 and PC2 respectively:

```
# Contributions of variables to PC1
fviz_contrib(contrib, choice = "var", axes = 1, top = 10) #fewer than 10 present but that doesn't seem
```




```
# Contributions of variables to PC2
fviz_contrib(contrib, choice = "var", axes = 2, top = 10)
```



As seen in the contrib*var*contrib output, `lit_rate`, `age`, and `region` contribute most to PC1. `year`, and `region` contribute most to PC2. contrib*var*contrib didn't tell us what the cutoff was for significant contribution the way that these bar charts do.‘

Finally, we can see which features are correlated and how the associate with given PCs:

```
contrib.desc <- dimdesc(contrib, axes = c(1,2), proba = 0.05)
# Description of dimension 1
contrib.desc$Dim.1
```

```
## $quanti
##          correlation       p.value
## age       0.59948018 1.462434e-321
## region    0.53974360 3.757583e-249
## year      0.03476118  4.575650e-02
## male     -0.28677823  1.467690e-63
## lit_rate -0.85338203  0.000000e+00
##
## attr(,"class")
## [1] "condes" "list "
```

While the previous barplots could show us degree of contribution, they couldn't tell us the directionality or correlation. `Lit_rate` has a strong *negative* correlation for PC1, while `age` and `region` both have positive correlations. The first PC1 is then strongly correlated with 3 of the variables, which seems to suggest that these variables vary together.

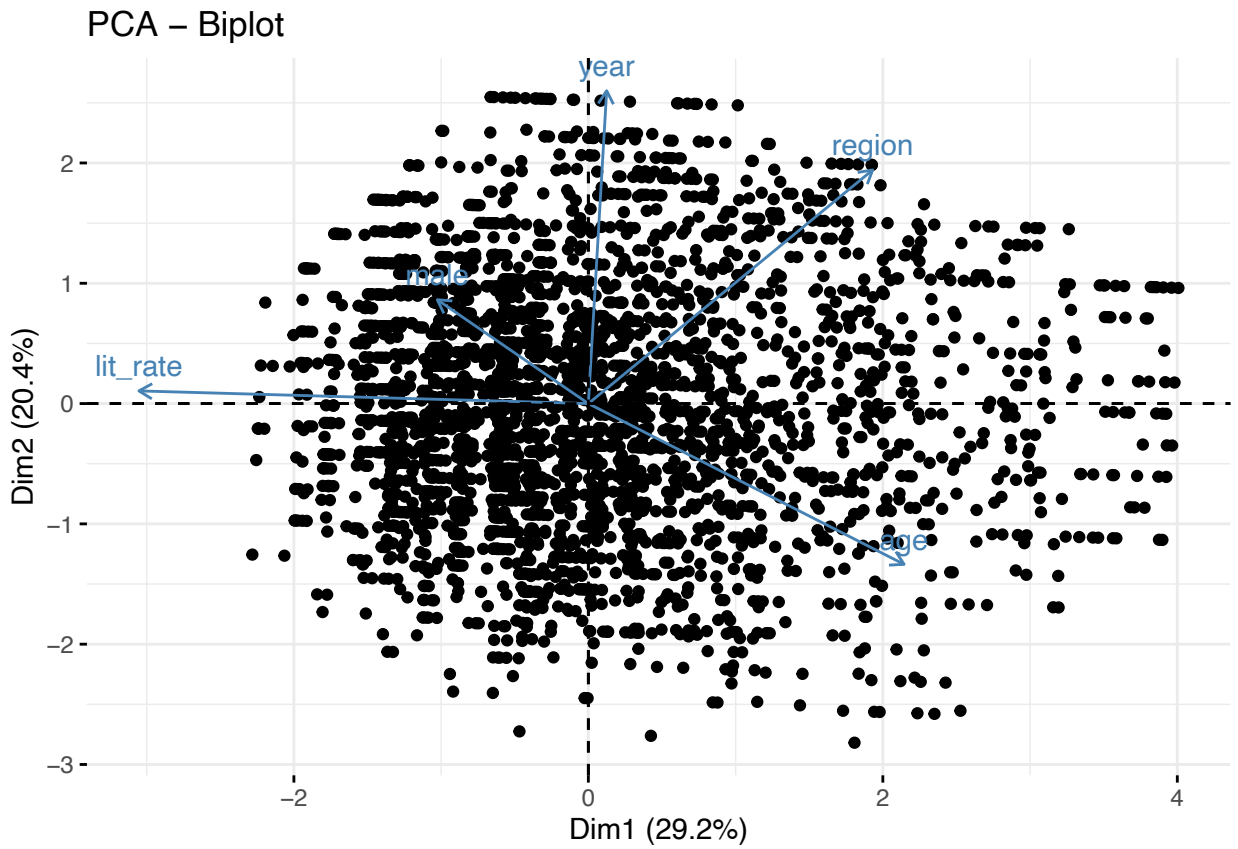
```
#Description of dimension 2

contrib.desc$Dim.2
```

```
## $quanti
##        correlation       p.value
## year     0.7269666  0.000000e+00
## region   0.5428690 1.386314e-252
## male     0.2412991  5.721381e-45
## age     -0.3735609 6.853822e-110
##
## attr(,"class")
## [1] "condes" "list "
```

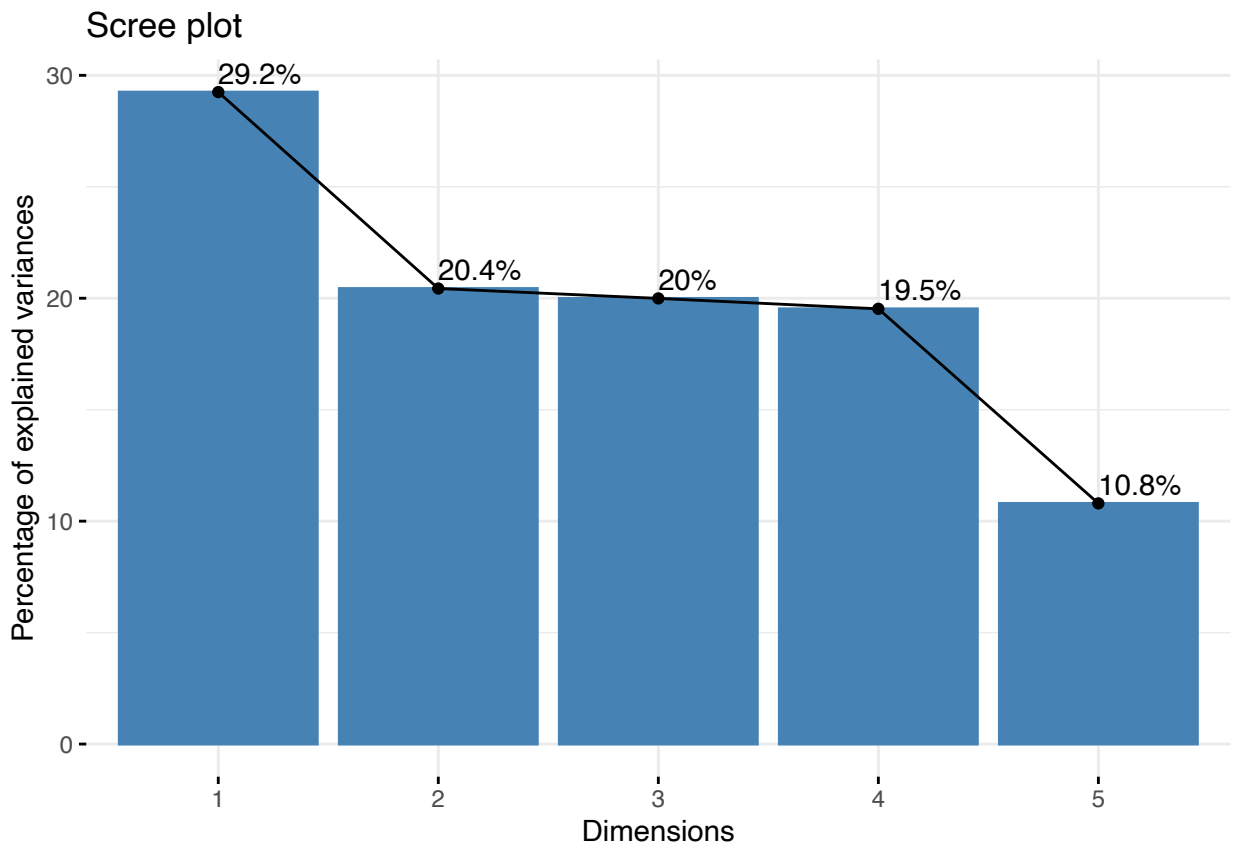
`year` and `region` both have positive correlations with PC2 (and vary together).

12. (10 points) Calculate and use `ggplot2`-style tools to visualize the proportion of variance explained (PVE) *and* the cumulative PVE (CPVE) for all principal components. Approximately how much of the variance is explained by the first two components?

```
fviz_pca_biplot(pca_fit, label = "var")
```



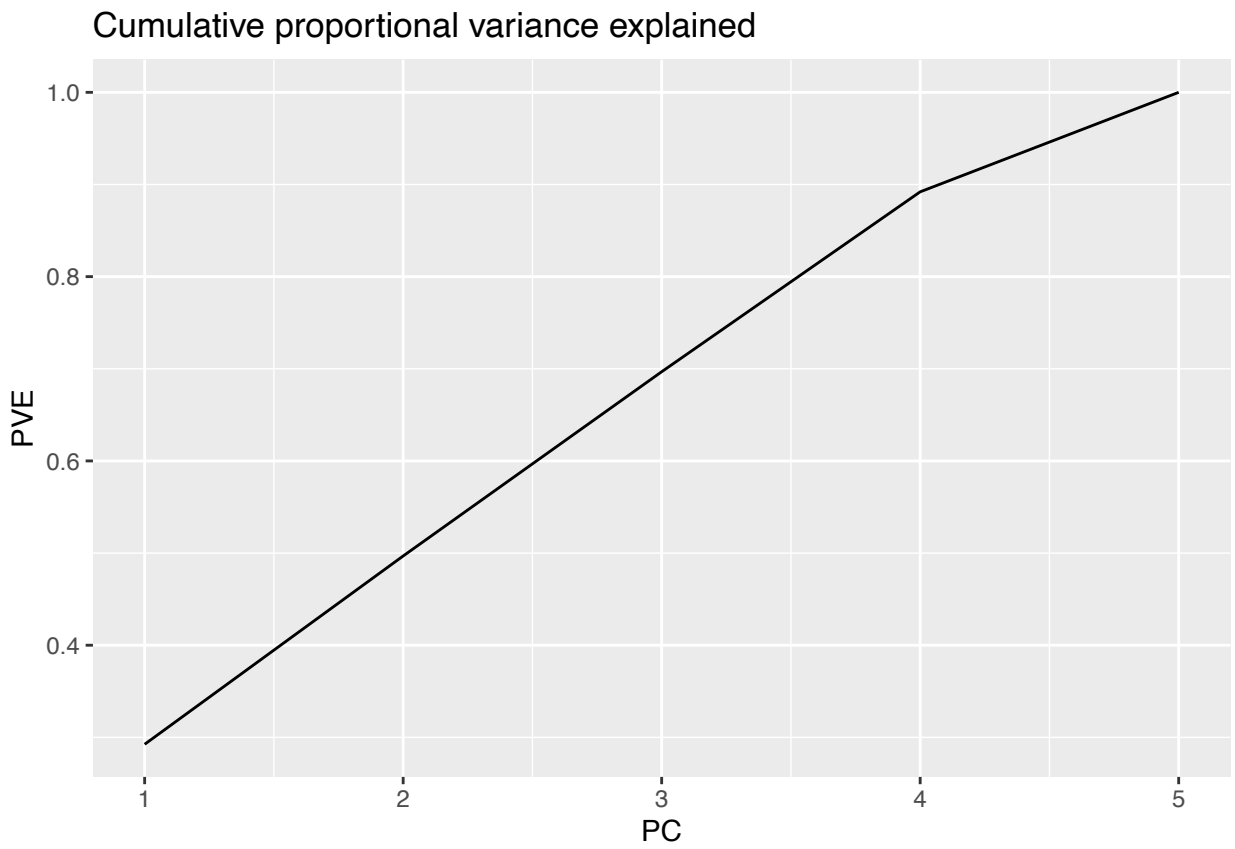
```
fviz_screeplot(pca_fit, addlabels = TRUE, choice = "variance")
```



About 50% (49.6%) of the variance is explained by the first two principal components. Below, I plot PVE and cumulative PVE.

```
#plotting cumulative PVE

covariances <- cov(literacy %>%
                     scale())
# calculate eigen
eigenvectors <- eigen(covariances)
```

```
var_prop <- eigenvectors$values / sum(eigenvectors$values)
# calculate cumulative vals
cumulative_prop <- cumsum(var_prop) %>%
  enframe(name="PC Number", "PVE")


# plot
ggplot(cumulative_prop, aes(x = `PC Number`, y=PVE)) +
  geom_line() +
  labs(title = "Cumulative proportional variance explained",
       y = "PVE",
       x = "PC")
```




```
PVE <- eigenvectors$values / sum(eigenvectors$values)
round(PVE, 2)
```

```
## [1] 0.29 0.20 0.20 0.20 0.11
```

```
library(gridExtra)
```

```
##
## Attaching package: 'gridExtra'

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

```
# PVE (aka scree) plot
PVEplot <- qplot(c(1:5), PVE) +
  geom_line() +
  xlab("Principal Component") +
  ylab("PVE") +
  ggtitle("Scree Plot") +
  ylim(0, 1)

# Cumulative PVE plot
cumPVE <- qplot(c(1:5), cumsum(PVE)) +
```

```
  geom_line() +
  xlab("Principal Component") +
  ylab(NULL) +
  ggtitle("Cumulative Scree Plot") +
  ylim(0,1)

grid.arrange(PVEplot, cumPVE, ncol = 2)
```