STAT167 HW7 - Spring 2025

Ethan Choi

Contents

Homework #7 instructions 1
Question 1 [40 pt] Relational data analysis with `dplyr` 3
The `USArrests` dataset 8
Question 2 [30 pt] Hierarchical Clustering 8
Question 3 [30 pt] K-Means Clustering 15

Homework #7 instructions

Review the R4DS textbook Chapter 19 "Transform > Joins" and the lecture notes on relational data before answering the homework Question 1.

Review the ISLR textbook Chapter 12 "Unsupervised Learning", and the lecture notes on clustering before answering the homework Questions 2 and 3.

This homework contains 3 questions, each with multiple parts, 100 points in total.

Replace **INSERT_YOUR_ANSWER** with your own answers.

- First open this `rmd` file in RStudio and click `Knit -> Knit to PDF` to render it to PDF format. You need to have `LaTex` installed on the computer to render it to PDF format. If not, you can also render it to HTML format.
- It is best to read this `rmd` file and the rendered `pdf`/`html` file side-by-side, while you are working on this homework.
- If the question asks you to write some R code, remember to put your code into a **R code chunk**. Make sure both your R code chunk and its output are visible in the rendered `pdf`/`html` file.
- For this homework, use **ggplot2** to visualize your data.
- **Please comment your R code thoroughly, and follow the R coding style guideline (https://google.github.io/styleguide/Rguide.xml). Partial credit will be deducted for insufficient commenting or poor coding styles.**
- If you have any question about this homework assignment, we encourage you to post it on **Piazza**.

Homework submission guideline

- **This homework is DUE at *11:59 PM* on *Friday June 6, 2025*.**
- Late submission penalties.
  - Submissions up to 24 hours late will incur a 10% deduction.
  - Submissions up to 48 hours late will incur a 30% deduction.
- **If you are using one or both of your free late days, please state here: INSERT_YOUR_ANSWER**
- After you complete all questions, save your `rmd` file to `FirstnameLastname-SID-HW7.rmd` and save the rendered pdf file to `FirstnameLastname-SID-HW7.pdf`. If you can not knit it to pdf, knit it to html first and then print/save it to pdf format.
- Submit **BOTH your source `rmd` file and the knitted `pdf` file** to **GradeScope**. Do NOT create a zip file. For the `pdf` submission, please tag specific pages that correspond with each question in the assignment.
- You can submit multiple times, you last submission will be graded.

Acknowledgments

Some of the example code were adopted from:

- Class material from Cosma Shalizi @ CMU, and David Dalpiaz @ UIUC
- Colored dendrogram example @ stackoverflow

Please list all the help you have received for completing this homework.

INSERT_YOUR_ANSWER

Install necessary packages

Note that you only need to install each package once. Then you can comment out the following installation lines.

```
# install.packages("Lahman")
# install.packages("ggdendro")
```

Load necessary packages

```
library(tidyverse) # for `ggplot2`, `dplyr`, `tidyr` and more
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.4     v readr     2.1.5
## v forcats   1.0.0     v stringr   1.5.1
## v ggplot2   3.5.2     v tibble    3.2.1
## v lubridate 1.9.4     v tidyr     1.3.1
## v purrr     1.0.4
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
library(Lahman) # for the Lahman dataset
library(nycflights13) # for NYC flights dataset
library(ggdendro) # for plotting dendrogram
```

Set the random seed

```
# set the random seed so that your analysis is reproducible
set.seed(167) # do NOT change this number
```

Question 1 [40 pt] Relational data analysis with `dplyr`

(a) [20 pt] Identify keys

In relational database, a **key** is a variable (or set of variables) that uniquely identifies an observation.

Every join involves a pair of keys: a primary key and a foreign key.

- A **primary key** uniquely identifies an observation in its own table.
- A **foreign key** corresponds to a primary key in another table.

Take the `nycflights13::airlines` table as an example, `airlines$carrier` is the primary key. We can confirm that as follows.

```
glimpse(airlines)
## Rows: 16
## Columns: 2
## $ carrier <chr> "9E", "AA", "AS", "B6", "DL", "EV", "F9", "FL", "HA", "MQ", "O~
## $ name    <chr> "Endeavor Air Inc.", "American Airlines Inc.", "Alaska Airline~

airlines |> count(carrier) |> filter (n>1)
## # A tibble: 0 x 2
## # i 2 variables: carrier <chr>, n <int>
```

The above analysis confirms that there do not exist two airlines in the `airlines` table that have the same `carrier` key. Thus, `airlines$carrier` is the primary key of `airlines` because it uniquely identifies each airline in the `airlines` table.

Next let's take a look at the `nycflights13::flights` table. `flights$carrier` is a foreign key in the `flights` table, because it matches each flight's carrier to a unique airline in the `airlines` table.

```
glimpse(flights)
## Rows: 336,776
## Columns: 19
## $ year           <int> 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2013, 2~
## $ month          <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ day            <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1~
## $ dep_time       <int> 517, 533, 542, 544, 554, 554, 555, 557, 557, 558, 558, ~
## $ sched_dep_time <int> 515, 529, 540, 545, 600, 558, 600, 600, 600, 600, 600, ~
## $ dep_delay      <dbl> 2, 4, 2, -1, -6, -4, -5, -3, -3, -2, -2, -2, -2, -2, -1~
## $ arr_time       <int> 830, 850, 923, 1004, 812, 740, 913, 709, 838, 753, 849,~
## $ sched_arr_time <int> 819, 830, 850, 1022, 837, 728, 854, 723, 846, 745, 851,~
## $ arr_delay      <dbl> 11, 20, 33, -18, -25, 12, 19, -14, -8, 8, -2, -3, 7, -1~
## $ carrier        <chr> "UA", "UA", "AA", "B6", "DL", "UA", "B6", "EV", "B6", "~
## $ flight         <int> 1545, 1714, 1141, 725, 461, 1696, 507, 5708, 79, 301, 4~
## $ tailnum        <chr> "N14228", "N24211", "N619AA", "N804JB", "N668DN", "N394~
## $ origin         <chr> "EWR", "LGA", "JFK", "JFK", "LGA", "EWR", "EWR", "LGA",~
## $ dest           <chr> "IAH", "IAH", "MIA", "BQN", "ATL", "ORD", "FLL", "IAD",~
## $ air_time       <dbl> 227, 227, 160, 183, 116, 150, 158, 53, 140, 138, 149, 1~
## $ distance       <dbl> 1400, 1416, 1089, 1576, 762, 719, 1065, 229, 944, 733, ~
## $ hour           <dbl> 5, 5, 5, 5, 6, 5, 6, 6, 6, 6, 6, 6, 6, 6, 6, 5, 6, 6, 6~
## $ minute         <dbl> 15, 29, 40, 45, 0, 58, 0, 0, 0, 0, 0, 0, 0, 0, 0, 59, 0~
## $ time_hour      <dttm> 2013-01-01 05:00:00, 2013-01-01 05:00:00, 2013-01-01 0~

anti_join(flights, airlines, by="carrier")
## # A tibble: 0 x 19
## # i 19 variables: year <int>, month <int>, day <int>, dep_time <int>,
## #   sched_dep_time <int>, dep_delay <dbl>, arr_time <int>,
## #   sched_arr_time <int>, arr_delay <dbl>, carrier <chr>, flight <int>,
## #   tailnum <chr>, origin <chr>, dest <chr>, air_time <dbl>, distance <dbl>,
## #   hour <dbl>, minute <dbl>, time_hour <dttm>
```

The above `anti_join()` code confirms that all carrier values in the `carrier` column of the `flights` table can be matched to a unique entry in the `airlines` table.

For the following datasets, identify the primary keys and/or foreign keys.

Note: You will need to install and load some packages to access the datasets and read the documentation.

(i) What is the primary key of the `Lahman::Batting` table? Use `count()` and `filter()` to support your claim of primary key.

```
Batting %>%
  count(playerID, yearID, stint) %>%
  filter(n > 1)
```

```
## [1] playerID yearID   stint    n
## <0 rows> (or 0-length row.names)
```

Player id, year id, and stint are all primary keys since the df returns zero

---

(ii) Find another table in the `Lahman` dataset, and identify the foreign key in the second table that can be used to connect with the `Lahman::Batting` table. Use `anti_join()` to support your choice of the foreign key.

```
anti_join(Pitching, Batting, by = c("playerID", "yearID", "stint"))
```

```
##  [1] playerID yearID   stint    teamID   lgID     W        L        G
##  [9] GS       CG       SHO      SV       IPouts   H        ER       HR
## [17] BB       SO       BAOpp    ERA      IBB      WP       HBP      BK
## [25] BFP      GF       R        SH       SF       GIDP
## <0 rows> (or 0-length row.names)
```

All variables exist in Pitching, supporting my choice of the foreign key

---

(iii) Is there a primary key for the `ggplot2::diamonds` table? Use `count()` and `filter()` to justify your answer.

```
diamonds %>%
  count(carat, cut, color, clarity, depth, table, price, x, y, z) %>%
  filter(n > 1)
```

```
## # A tibble: 143 x 11
##    carat cut       color clarity depth table price     x     y     z     n
##    <dbl> <ord>     <ord> <ord>   <dbl> <dbl> <int> <dbl> <dbl> <dbl> <int>
##  1  0.3  Good      J     VS1      63.4    57   394  4.23  4.26  2.69     2
##  2  0.3  Very Good G     VS2      63      55   526  4.29  4.31  2.71     2
##  3  0.3  Very Good J     VS1      63.4    57   506  4.26  4.23  2.69     2
##  4  0.3  Premium   D     SI1      62.2    58   709  4.31  4.28  2.67     2
##  5  0.3  Ideal     G     VS2      63      55   675  4.31  4.29  2.71     2
##  6  0.3  Ideal     G     IF       62.1    55   863  4.32  4.35  2.69     2
##  7  0.3  Ideal     H     SI1      62.2    57   450  4.26  4.29  2.66     2
##  8  0.3  Ideal     H     SI1      62.2    57   450  4.27  4.28  2.66     2
##  9  0.31 Good      D     SI1      63.5    56   571  4.29  4.31  2.73     2
## 10  0.31 Very Good D     SI1      63.5    56   732  4.31  4.29  2.73     2
## # i 133 more rows
```

No, duplicates exist

(iv) In the `nycflights13` diagram (https://r4ds.hadley.nz/diagrams/relational.png), the authors forgot to draw the relationship between `weather` and `airports`.

What is the relationship? Which table has a foreign key that can be used to connect to the other table? Use `anti_join()` to support your choice of the foreign key.

```
anti_join(weather, airports, by = c("origin" = "faa"))
```

```
## # A tibble: 0 x 15
## # i 15 variables: origin <chr>, year <int>, month <int>, day <int>, hour <int>,
## #   temp <dbl>, dewp <dbl>, humid <dbl>, wind_dir <dbl>, wind_speed <dbl>,
## #   wind_gust <dbl>, precip <dbl>, pressure <dbl>, visib <dbl>,
## #   time_hour <dttm>
```

weather$origin can be used as a foreign key to airports$faa

(b) [20 pt] Analysis of the `nycflights13` dataset

(i) Relationship between planes and airlines

You might expect that there's an implicit relationship between plane and airline, because each plane is flown by a single airline. Confirm or reject the hypothesis that each plane only flew for one airline. If you reject the hypothesis, provide a counterexample.

Hint: Generate a table of all unique pairs of `carrier` and `tailnum` for non-canceled flights. Check whether `tailnum` is the primary key of that table.

```
flights_clean <- flights %>%
  filter(!is.na(dep_time), !is.na(arr_time))

flights_clean %>%
  distinct(tailnum, carrier) %>%
  count(tailnum) %>%
  filter(n > 1)
```

```
## # A tibble: 17 x 2
##    tailnum     n
##    <chr>   <int>
##  1 N146PQ      2
##  2 N153PQ      2
##  3 N176PQ      2
##  4 N181PQ      2
##  5 N197PQ      2
##  6 N200PQ      2
##  7 N228PQ      2
##  8 N232PQ      2
##  9 N933AT      2
## 10 N935AT      2
## 11 N977AT      2
## 12 N978AT      2
## 13 N979AT      2
## 14 N981AT      2
## 15 N989AT      2
## 16 N990AT      2
## 17 N994AT      2
```

(ii) Identify all the planes that had flown at least 100 flights. Which plane flew the most? Which airline it belonged to?

```
plane_flight_counts <- flights_clean %>%
  count(tailnum) %>%
  filter(n >= 100)

top_plane <- plane_flight_counts %>%
  arrange(desc(n)) %>%
  slice(1)

flights_clean %>%
  filter(tailnum == top_plane$tailnum) %>%
  count(carrier) %>%
  arrange(desc(n))
```

```
## # A tibble: 1 x 2
##   carrier     n
##   <chr>   <int>
## 1 MQ        546
```

MQ

(iii) Generate a table with 16 rows (one for each airline) and 3 columns: the two-letter carrier abbreviation, the full airline name, and the number of frequently flown planes (planes had flown at least 100 times in 2013) the airline had used. Which airlines did not have any frequently flown plane?

```
frequent_planes <- plane_flight_counts$tailnum

flights_clean %>%
  filter(tailnum %in% frequent_planes) %>%
  distinct(carrier, tailnum) %>%
  count(carrier) %>%
  left_join(airlines, by = "carrier") %>%
  rename(`Airline Name` = name, `Number of Frequent Planes` = n)
```

```
## # A tibble: 9 x 3
##   carrier `Number of Frequent Planes` `Airline Name`
##   <chr>                         <int> <chr>
## 1 9E                               41 Endeavor Air Inc.
## 2 AA                               28 American Airlines Inc.
## 3 B6                              187 JetBlue Airways
## 4 DL                              211 Delta Air Lines Inc.
## 5 EV                              270 ExpressJet Airlines Inc.
## 6 MQ                               80 Envoy Air
## 7 UA                              320 United Air Lines Inc.
## 8 US                               47 US Airways Inc.
## 9 VX                               26 Virgin America
```

The `USArrests` dataset

The `datasets::USArrests` dataset contains statistics, in arrests per 100,000 residents for assault, murder, and rape in each of the 50 US states in 1973. Also given is the percent of the population living in urban areas.

```
?USArrests # `USArrests` helper page
## starting httpd help server ... done
dim(USArrests)
## [1] 50  4
head(USArrests)
##            Murder Assault UrbanPop Rape
## Alabama      13.2     236       58 21.2
## Alaska       10.0     263       48 44.5
## Arizona       8.1     294       80 31.0
## Arkansas      8.8     190       50 19.5
## California    9.0     276       91 40.6
## Colorado      7.9     204       78 38.7
```

Question 2 [30 pt] Hierarchical Clustering

(a) [10 pt] Data scaling

(i) Use `dplyr` function to calculate the mean and the variance for each variable (i.e., each column) in `USArrests` Is there a huge difference?

Hint: Call `summarize_all()` to calculate mean and variance, separately.

```
USArrests %>%
  summarize_all(mean)
```

```
##   Murder Assault UrbanPop   Rape
## 1  7.788  170.76    65.54 21.232
```

```
USArrests %>%
  summarize_all(var)
```

```
##     Murder  Assault UrbanPop     Rape
## 1 18.97047 6945.166 209.5188 87.72916
```

Yes, there is quite a big difference in means.

(ii) We can perform (column-wise) data scaling using the `scale()` function.

```
USArrests.scaled <- scale(USArrests)
head(USArrests.scaled)
##                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
```

Use `dplyr` function to calculate the mean and the variance for each variable (i.e., each column) in `USArrests.scaled`. Is there much difference?

```
USArrests.scaled <- scale(USArrests)

scaled_df <- as_tibble(USArrests.scaled)

scaled_df %>%
  summarize_all(mean)
```

```
## # A tibble: 1 x 4
##      Murder  Assault  UrbanPop     Rape
##       <dbl>    <dbl>     <dbl>    <dbl>
## 1 -7.66e-17 1.11e-16 -4.33e-16 8.94e-17
```

```
scaled_df %>%
  summarize_all(var)
```

```
## # A tibble: 1 x 4
##   Murder Assault UrbanPop  Rape
##    <dbl>   <dbl>    <dbl> <dbl>
## 1      1       1        1     1
```

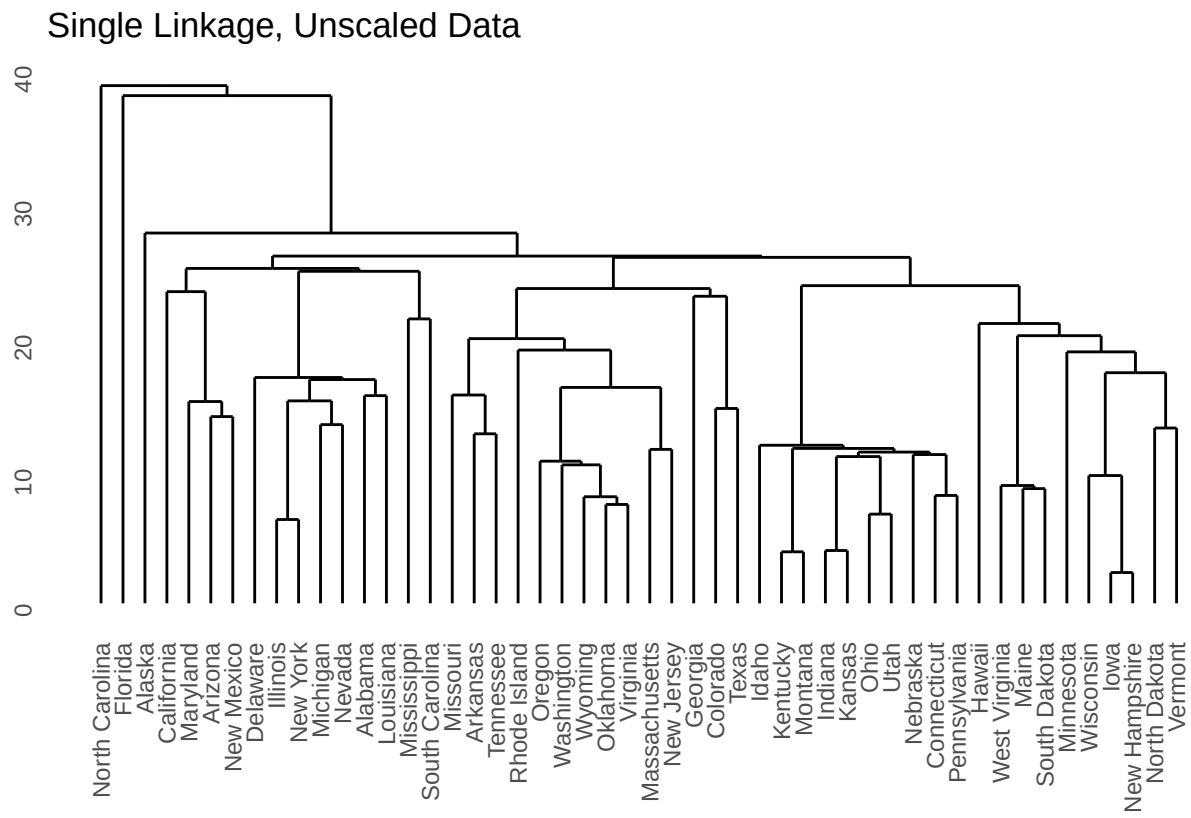
Yes, a difference is notable

(b) [15 pt] Linkages

To perform a hierarchical clustering, you need to decide which linkage method (complete, single, average, or others) to use.

For example, for single linkage on unscaled data:

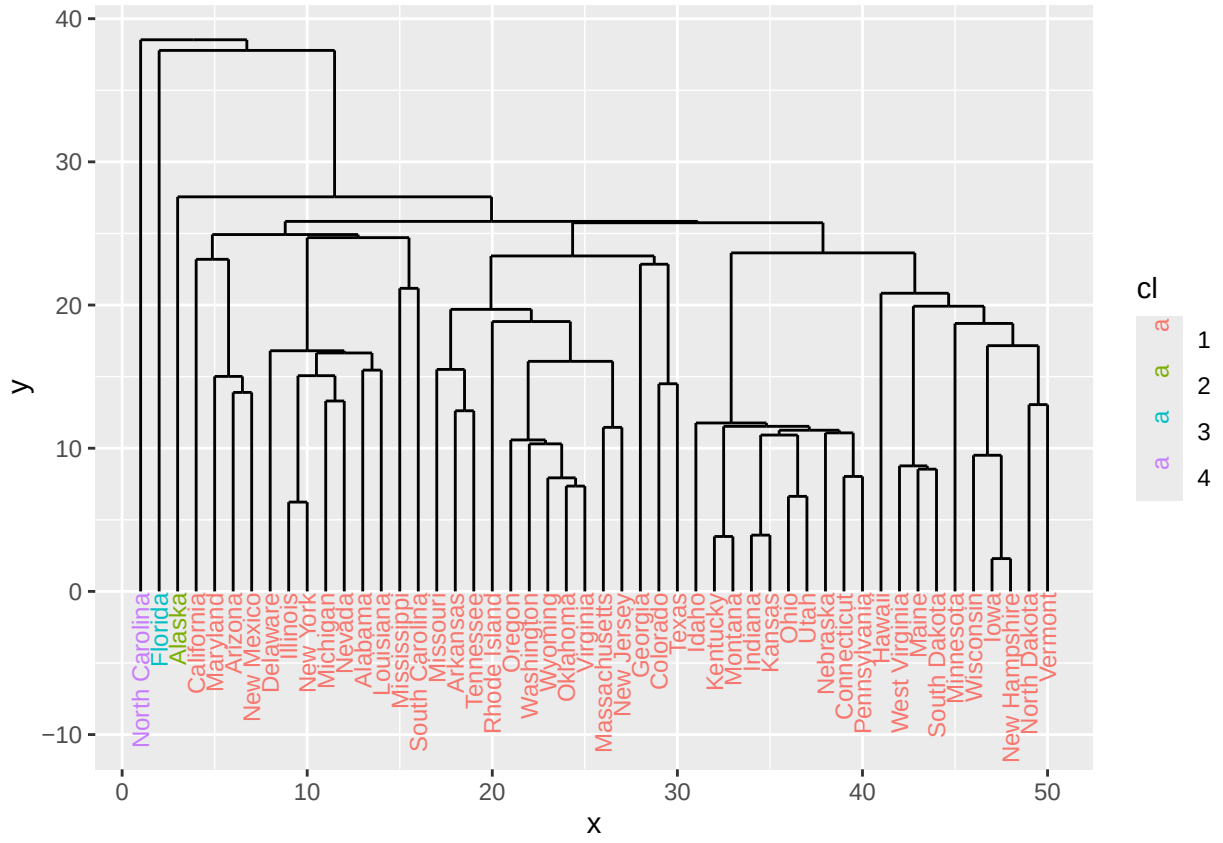
```
hc.single <- hclust(d = dist(USArrests), method = "single")

ggdendrogram(data = hc.single) +
  ggtitle("Single Linkage, Unscaled Data")
```



Suppose we decide to cut the dendrogram at a height that results in 4 distinct clusters. We can plot the dendrogram with a different color for labels in each cluster.

```
# cut the dendrogram to get 4 clusters
hc.cl <- cutree(tree = hc.single, k = 4)
# hc.cl

hc.cl.tb <- tibble(label = names(hc.cl), cl = as.factor(hc.cl))
# hc.cl.tb

# get dendrogram
hc.single.dd <- as.dendrogram(hc.single)
# hc.single.dd

# rectangular lines of the dendrogram
hc.single.dd.data <- dendro_data(hc.single.dd, type = "rectangle")
# hc.single.dd.data

# join dendrogram labels with clustering results
labels <- label(hc.single.dd.data) |> left_join(hc.cl.tb)
## Joining with `by = join_by(label)`
# labels

# plot dendrogram with colored leaf labels
ggplot(data = segment(hc.single.dd.data)) +
  geom_segment(mapping = aes(x = x, y = y, xend = xend, yend = yend)) +
```

```
  geom_text(data = labels,
            mapping = aes(label = label, x = x, y = -.1, color = cl),
            size = 3, hjust = 1, angle = 90) +
  ylim(low = -10, high = NA)
```



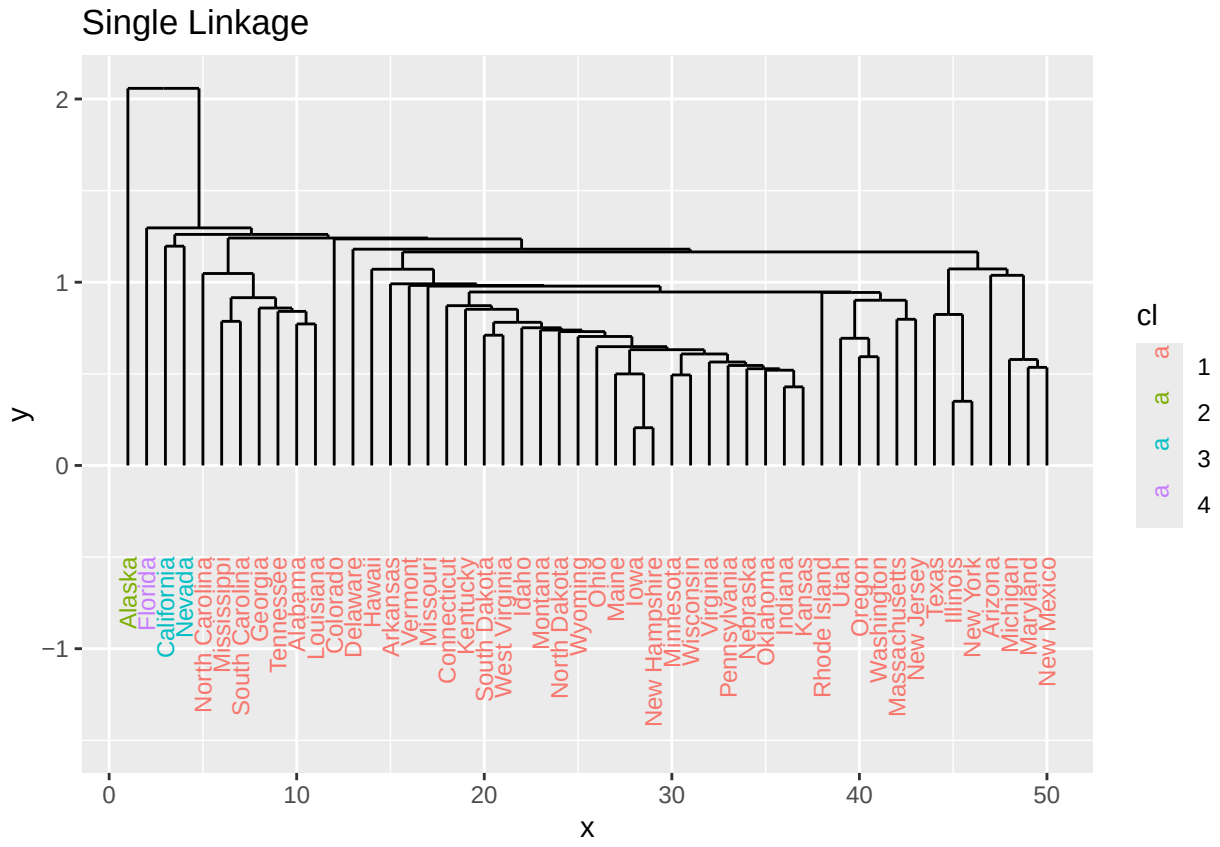
(i) Write a R code to perform hierarchical clustering with single linkage on the scaled data `USArrests.scaled`. Cut the dendrogram at a height that results in 4 distinct clusters. Then plot the dendrogram with a different color for labels in each cluster.

Hint: You might need to adjust `ylim()` to show the full labels.

```
hc.single.scaled <- hclust(dist(USArrests.scaled), method = "single")

hc.cl.single <- cutree(hc.single.scaled, k = 4)
hc.cl.tb <- tibble(label = names(hc.cl.single), cl = as.factor(hc.cl.single))

hc.single.dd <- as.dendrogram(hc.single.scaled)
hc.single.dd.data <- dendro_data(hc.single.dd, type = "rectangle")
labels <- label(hc.single.dd.data) |> left_join(hc.cl.tb)
```

```
## Joining with `by = join_by(label)`
```

```
ggplot(data = segment(hc.single.dd.data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data = labels,
            aes(label = label, x = x, y = -0.5, color = cl),
            size = 3, hjust = 1, angle = 90) +
  ylim(low = -1.5, high = NA) +
  ggtitle("Single Linkage")
```



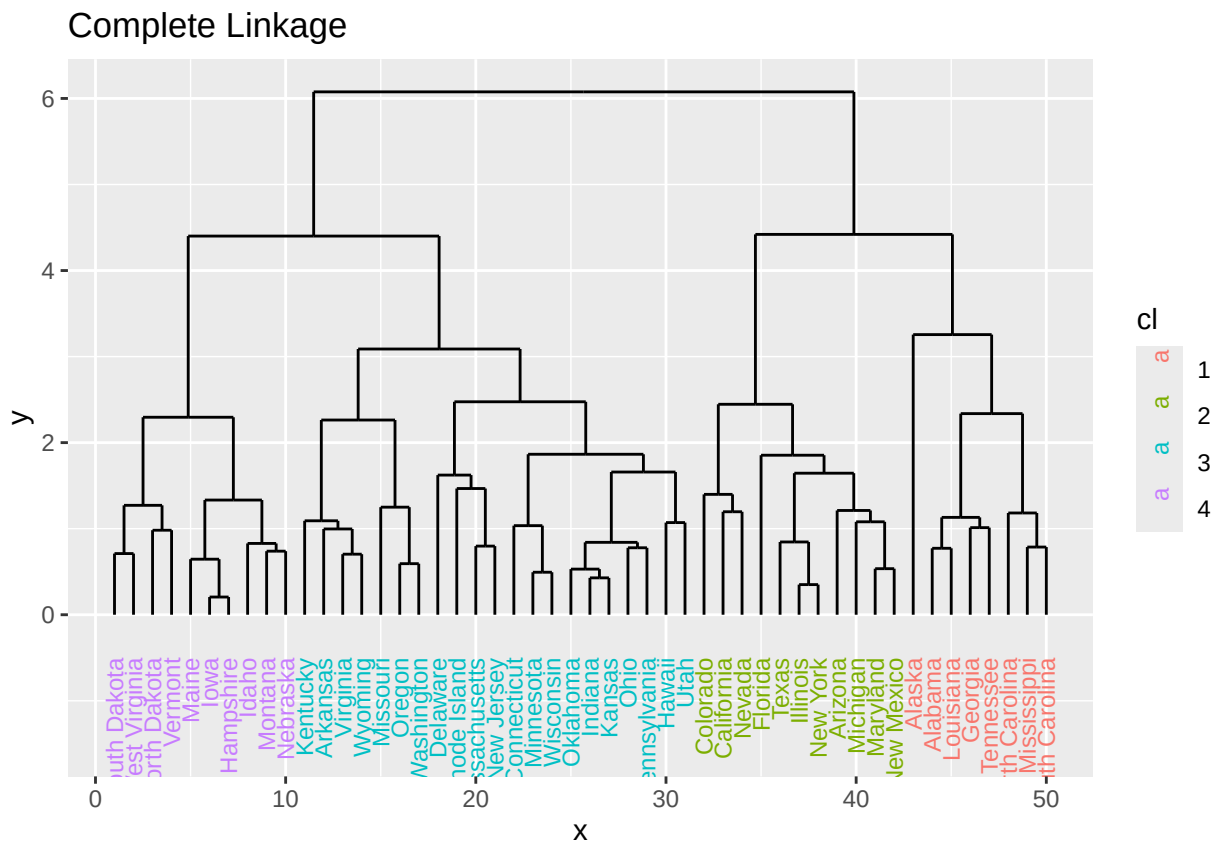
(ii) Choose another linkage method and then perform hierarchical clustering again on the scaled data `USArrests.scaled`. Again, cut the dendrogram at a height that results in 4 distinct clusters. Then plot the dendrogram with a different color for labels in each cluster.

Hint: See the documentation of `hclust` to find other possible linkages.

```
hc.complete.scaled <- hclust(dist(USArrests.scaled), method = "complete")

hc.cl.complete <- cutree(hc.complete.scaled, k = 4)
hc.cl.tb.complete <- tibble(label = names(hc.cl.complete), cl = as.factor(hc.cl.complete))

hc.complete.dd <- as.dendrogram(hc.complete.scaled)
hc.complete.dd.data <- dendro_data(hc.complete.dd, type = "rectangle")
labels.complete <- label(hc.complete.dd.data) |> left_join(hc.cl.tb.complete)
```

```
## Joining with `by = join_by(label)`
```

```
ggplot(data = segment(hc.complete.dd.data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data = labels.complete,
            aes(label = label, x = x, y = -0.5, color = cl),
            size = 3, hjust = 1, angle = 90) +
  ylim(low = -1.5, high = NA) +
  ggtitle("Complete Linkage")
```



(iii) Based on the above plots, do you think any of the results seem more reasonable than the others? Pick your favorite clustering result. Explain your choice.

The complete linkage dendrogram looks better, with more separation between clusters. Single linkage seems to generate clumped clusters, which may not reflect groups well. I prefer the complete linkage clustering result.

(c) [5 pt] Other distance measure options

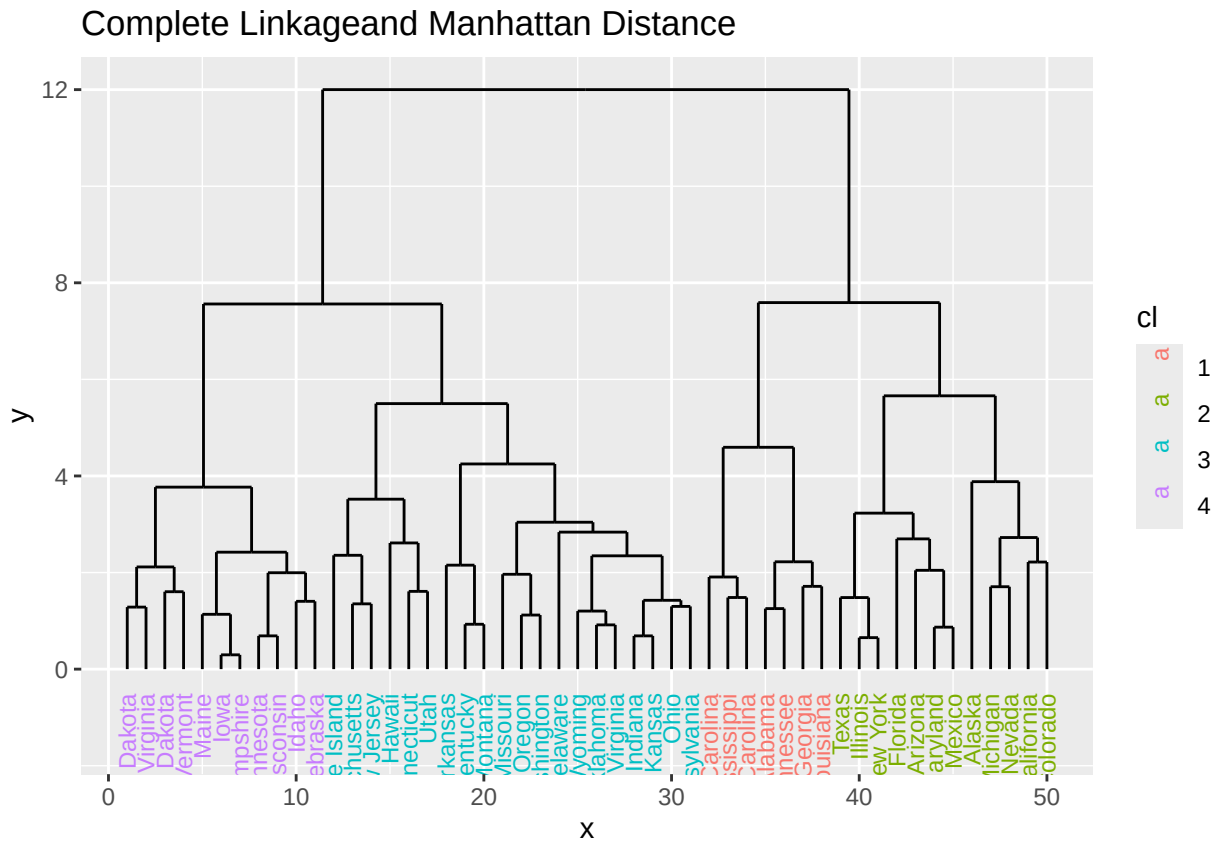
Read the documentation of `dist` and find other possible distance measures. We have been using the default Euclidean distance measure. Try another distance measure (do not use the `binary` method). Visualize

the hierarchical clustering result. Compare the result to your favorite one from part (b). Is there much difference?

```
hc.manhattan <- hclust(dist(USArrests.scaled, method = "manhattan"), method = "complete")

hc.cl.manhattan <- cutree(hc.manhattan, k = 4)
hc.cl.tb.manhattan <- tibble(label = names(hc.cl.manhattan), cl = as.factor(hc.cl.manhattan))

hc.manhattan.dd <- as.dendrogram(hc.manhattan)
hc.manhattan.dd.data <- dendro_data(hc.manhattan.dd, type = "rectangle")
labels.manhattan <- label(hc.manhattan.dd.data) |> left_join(hc.cl.tb.manhattan)
```

```
## Joining with `by = join_by(label)`
```

```
ggplot(data = segment(hc.manhattan.dd.data)) +
  geom_segment(aes(x = x, y = y, xend = xend, yend = yend)) +
  geom_text(data = labels.manhattan,
            aes(label = label, x = x, y = -0.5, color = cl),
            size = 3, hjust = 1, angle = 90) +
  ylim(low = -1.5, high = NA) +
  ggtitle("Complete Linkageand Manhattan Distance")
```



Question 3 [30 pt] K-Means Clustering

The goal of K-means clustering is to find a partition of the data $\{C_1, \cdots, C_k\}$ that minimizes the sum of **within-cluster variations**:

$$\sum_{k=1}^{K} WCV(C_k)$$

Typically, we use the sum of all the pair-wise squared **Euclidean distances** between the observations in each cluster to quantify the **within-cluster variation**:

$$WCV(C_k) = \frac{1}{|C_k|} \sum_{i,i' \in C_k} \|\boldsymbol{x_i} - \boldsymbol{x_{i'}}\|_2^2 = 2 \sum_{i \in C_k} \|\boldsymbol{x_i} - \bar{\boldsymbol{x}}_{\boldsymbol{k}}\|_2^2$$

where $|C_k|$ denotes the number of observations in the kth cluster, and $\bar{\boldsymbol{x}}_{\boldsymbol{k}} = \frac{1}{|C_k|} \sum_{i \in C_k} \boldsymbol{x_i}$ is the average of all the points in the kth cluster, i.e., the cluster mean/centroid.

(a) [15 pt] Choose the number of clusters K

(i) Apply K-means to the `USArrests.scaled` data 15 times, each time with a different number of centers from 1 to 15. Use `nstart = 20` in the `kmeans()` calls and store the `tot.withinss` value from the resulting object.

Hint: Call the `kmeans()` function within a `for` loop.

```
set.seed(123)

wss <- numeric(15)

for (k in 1:15) {
  km <- kmeans(USArrests.scaled, centers = k, nstart = 20)
  wss[k] <- km$tot.withinss
}

print(wss)
```

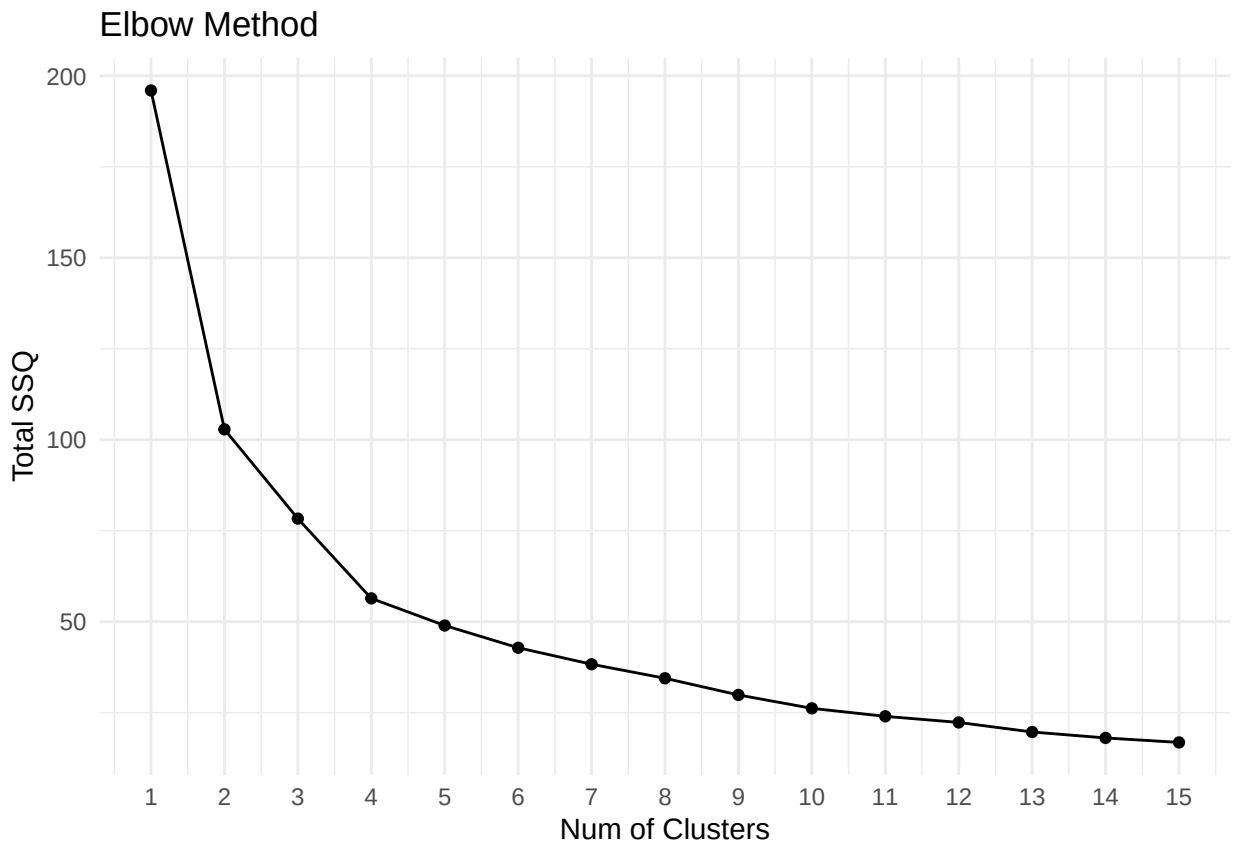
```
##  [1] 196.00000 102.86240  78.32327  56.40317  48.94420  42.83303  38.30416
##  [8]  34.44327  29.86789  26.18348  23.98458  22.30834  19.68082  18.04643
## [15]  16.81152
```

(ii) The total within-cluster sum of squares `tot.withinss`, $\sum_{i \in C_k} \|\boldsymbol{x_i} - \bar{\boldsymbol{x}}_{\boldsymbol{k}}\|_2^2$, measures how variable the observations are within the same cluster, which we would like to be low.

So obviously this value will be lower with more centers, no matter how many clusters there truly are.

Plot the total within-cluster sum of squares `tot.withinss` values against the number of centers K.

```r
tibble(K = 1:15, WSS = wss) %>%
  ggplot(aes(x = K, y = WSS)) +
  geom_line() +
  geom_point() +
  scale_x_continuous(breaks = 1:15) +
  labs(title = "Elbow Method",
       x = "Num of Clusters",
       y = "Total SSQ") +
  theme_minimal()
```



Look for an "elbow", the number of centers where the improvement suddenly drops off. Based on this plot, how many cluster do you think should be used for this data?

Based on the elbow plot, it seems to appear around K = 4 or 5I'd say K = 4 is a good balance between simplicity and accuracy.

(b) [5 pt] Interpret K-means clustering results

Re-apply K-means for your chosen number of centers. How many observations are placed in each cluster? What is the total of within-cluster sum of squares? What is the within-cluster sum of squares for each cluster?

```r
set.seed(123)
kmeans.final <- kmeans(USArrests.scaled, centers = 4, nstart = 25)

kmeans.final$size
```

```
## [1]  8 13 16 13
```

```r
kmeans.final$tot.withinss
```

```
## [1] 56.40317
```

```r
kmeans.final$withinss
```

```
## [1]  8.316061 11.952463 16.212213 19.922437
```

The 50 states are distributed across 4 clusters.

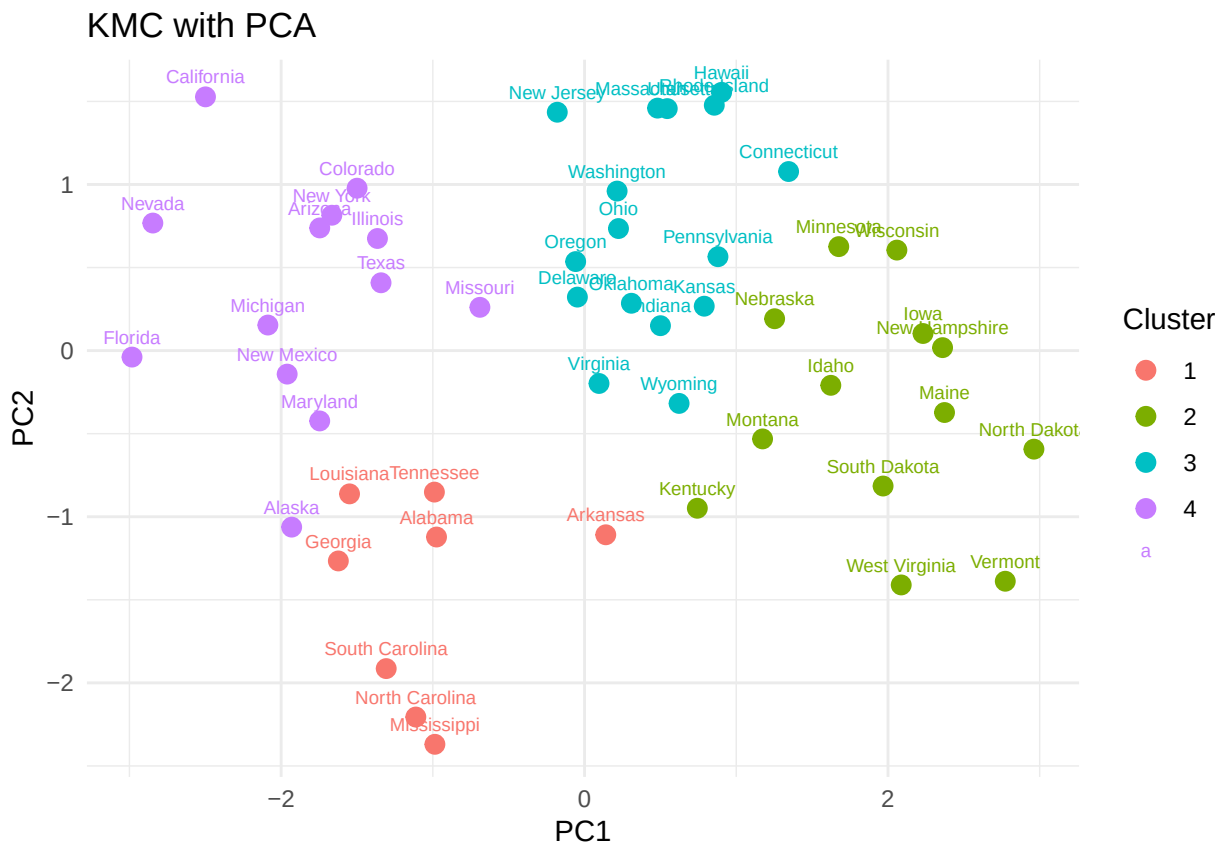
Total within-cluster sum of squares: Around 56.4

Within-cluster sum of squares by cluster: 8.316061 11.952463 16.212213 19.922437

(c) [10 pt] Visualize K-means clustering results

Plot the data using the first two variables and color the points according to the K-means clustering. Based on this plot, do you think you made a good choice for the number of centers? Briefly explain your answer.

```r
pca.results <- prcomp(USArrests.scaled)
pca.df <- as_tibble(pca.results$x[, 1:2]) %>%
  mutate(Cluster = factor(kmeans.final$cluster),
         State = rownames(USArrests))

ggplot(pca.df, aes(x = PC1, y = PC2, color = Cluster, label = State)) +
  geom_point(size = 3) +
  geom_text(size = 2.5, vjust = -1) +
  labs(title = "KMC with PCA") +
  theme_minimal()
```



Since the clusters are pretty well seperated, i'd say that the choice of K=4 is valid. ***