Lab 4: PCA and Clustering

Kaori Hirano

2023-06-29

Packages

```
# load packages here
suppressPackageStartupMessages(library(tidyverse))
suppressPackageStartupMessages(library(readr))
library(ISLR2)
library(broom) # for tidy function
library(patchwork) # for plot placement
library(ggdendro) # for dendrograms
library(mdsr) # for later examples
library(ggplot2)
```

Data

```
# load data here
load("data/vendor_data.RData")
```

Data Wrangling

Q1

First, let's fix a small issue and create a new, categorical version of income. 1. One particular respondent reported owning 80,000 houses. This is most likely untrue (and seriously affects the standard deviation, which, as you know from the reading and the example code, plays a key role in PCA in particular). Let's replace this value with NA. 2. Use the cut_number() function on the income variable to create a 10-level version of this variable. Call this hh_income_cat10

```
vendor_data$houses[vendor_data$houses == 80000] <- NA # replaces likely mistake with NA

# summary(vendor_data$houses) checks if works
vendor_data$hh_income_cat10 <- cut_number(vendor_data$hh_income_trim_99, 10) # cuts income
```

PCA

Q2

What are the mean and variance of the possessions variables? Does it seem like we should scale them before doing PCA?

```
# prints means and variances for all posession variables
cbind(mean(vendor_data$houses, na.rm = TRUE), var(vendor_data$houses, na.rm= TRUE))
```

```
         [,1]    [,2]
[1,] 1.228153 1.64927
```

```
cbind(mean(vendor_data$acres_farmland, na.rm = TRUE), var(vendor_data$acres_farmland, na.r
```

```
         [,1]     [,2]
[1,] 1.735929 7.909405
```

```
cbind(mean(vendor_data$bicycles, na.rm = TRUE), var(vendor_data$bicycles, na.rm= TRUE))
```

```
          [,1]     [,2]
[1,] 0.9592885 1.165997
```

```
cbind(mean(vendor_data$chickens, na.rm = TRUE), var(vendor_data$chickens, na.rm= TRUE))
```

```
         [,1]     [,2]
[1,] 3.927806 56.28764
```

```r
cbind(mean(vendor_data$goats, na.rm = TRUE), var(vendor_data$goats, na.rm= TRUE))
```

```
         [,1]     [,2]
[1,] 1.254452 15.23174
```

```r
cbind(mean(vendor_data$basic_cell_phones, na.rm = TRUE), var(vendor_data$basic_cell_phones
```

```
         [,1]      [,2]
[1,] 1.194071 0.8721673
```

```r
cbind(mean(vendor_data$smart_phones, na.rm = TRUE), var(vendor_data$smart_phones, na.rm= T
```

```
          [,1]      [,2]
[1,] 0.3610781 0.4916962
```

```r
??na.omit

# drops all nas for simplicity's sake
vd <- na.omit(vendor_data)
 apply(vd[, 12:18], 2, mean) # we need to subset to columns 2 through 5
```

```
           houses    acres_farmland          bicycles          chickens
        1.2627291         1.7993126         0.9857434         4.0814664
            goats basic_cell_phones      smart_phones
        1.3136456         1.2021385         0.3655804
```

```r
#pr_vd$center
                                # because first column is character column
 apply(vd[, 12:18], 2, var)
```

```
           houses    acres_farmland          bicycles          chickens
        1.8035822         8.8925963         1.2468675        56.7411948
            goats basic_cell_phones      smart_phones
       17.2882294         0.8714982         0.5152895
```

```
#pr_vd$scale
```

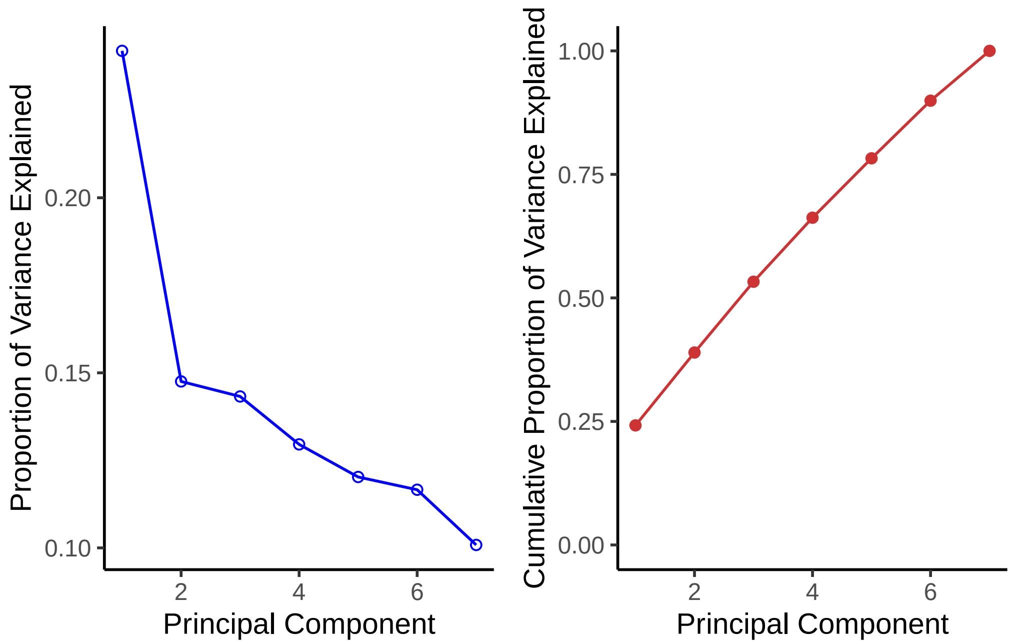
The mean and variances of the possession variables range from .5-56.7 (variance) and .3 to 4.0 (mean). I think it would make sense to scale them because you can have a lot more chickens than you can phones, so we don't want to give too much influence to one variable that doesn't necessarily warrant it when we want to know about the differences not the overall amount.

Q3

Use the prcomp() function to do PCA. Create two scree plots like the ones in Figure 12.3 of the textbook. Is there an elbow? How many components does it seem are sufficient?

```
# pca creation
pr_vd <- prcomp(vd[, 12:18], scale = TRUE)

comp_vd <- pr_vd$x %>%
  as_tibble()

pve_vd <- tibble(component = 1:ncol(comp_vd),
              var = pr_vd$sdev^2,
              pve = var/sum(var),
              cumulative = cumsum(pve))

#creates plot
pve_vd_plot <- ggplot(pve_vd) +
  geom_point(aes(x = component,
                 y = pve),
             color = "blue",
             shape = 1) +
  geom_line(aes(x = component,
                y = pve),
            color = "blue") +
  labs(x = "Principal Component",
       y = "Proportion of Variance Explained") +
  theme_classic()

cpve_vd_plot <- ggplot(pve_vd) +
  geom_point(aes(x = component,
                 y = cumulative),
             color = "brown3") +
  geom_line(aes(x = component,
```

```
                 y = cumulative),
              color = "brown3") +
  labs(x = "Principal Component",
       y = "Cumulative Proportion of Variance Explained") +
  ylim(c(0, 1)) +
  theme_classic()

# using patchwork package again
pve_vd_plot + cpve_vd_plot
```




```
pr_vd
```

```
Standard deviations (1, .., p=7):
[1] 1.3014155 1.0162523 1.0013681 0.9523320 0.9174713 0.9034611 0.8401661

Rotation (n x k) = (7 x 7):
                     PC1         PC2        PC3        PC4         PC5
houses         0.4252236 -0.12479018  0.2012112 -0.4379147  0.57698491
acres_farmland 0.4027297  0.37511002 -0.4848129 -0.2143706  0.21737205
bicycles       0.4152664 -0.19750494  0.2106372  0.5912293 -0.06938089
chickens       0.4277457 -0.04863014  0.2787870  0.3392641  0.20465307
```

```
goats              0.4184615  0.07007486 -0.5618568  0.1501649 -0.38041120
basic_cell_phones 0.3255098  0.29549847  0.5038891 -0.4143543 -0.61255342
smart_phones      0.1432292 -0.84268438 -0.1810883 -0.3215350 -0.23066610
                          PC6         PC7
houses             0.36385795  0.32565757
acres_farmland    -0.14983264 -0.58855973
bicycles           0.49390017 -0.38193515
chickens          -0.74167164  0.17287778
goats              0.12159006  0.56767082
basic_cell_phones -0.01820749 -0.07467999
smart_phones      -0.18980965 -0.20965723
```

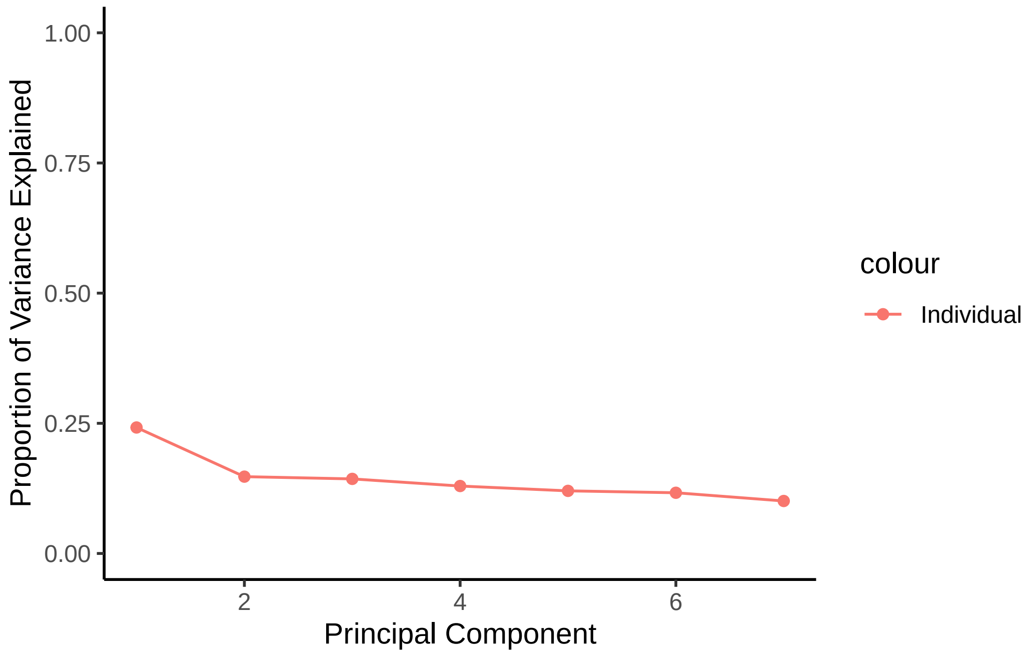
```r
# standard deviation of each principal component
pr_vd$sdev
```

```
[1] 1.3014155 1.0162523 1.0013681 0.9523320 0.9174713 0.9034611 0.8401661
```

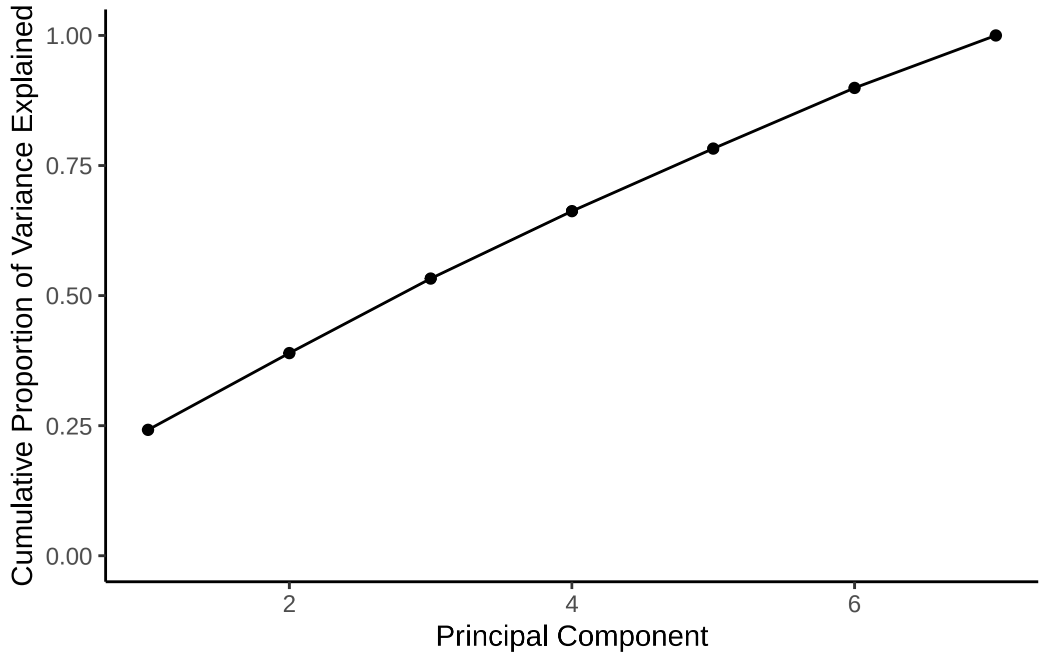
```r
# variance explained
pr_vd$sdev^2
```

```
[1] 1.6936822 1.0327686 1.0027381 0.9069362 0.8417537 0.8162420 0.7058791
```

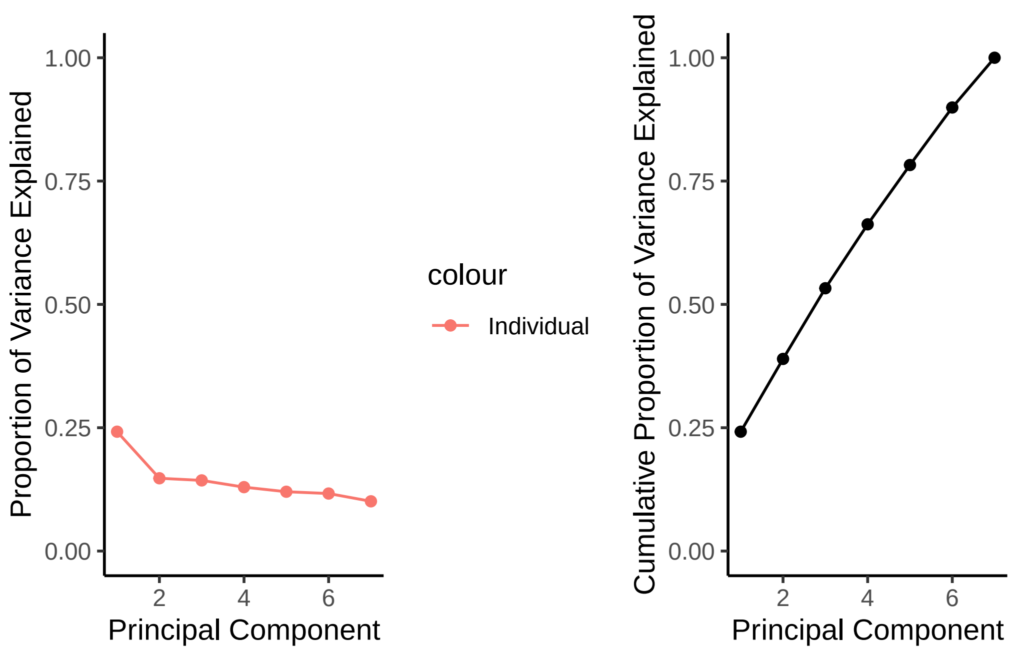
```r
# proportion of variance explained
pve <- pr_vd$sdev^2/sum(pr_vd$sdev^2)

pve_plot <- ggplot(pve_vd) +
  geom_point(aes(x = component,
                 y = pve, color = "Individual")) +
  geom_line(aes(x = component,
                y = pve, color = "Individual")) +
  labs(x = "Principal Component",
       y = "Proportion of Variance Explained") +
  ylim(c(0, 1)) +
  theme_classic()
pve_plot
```




```
cpve_plot <- ggplot(pve_vd) +
  geom_point(aes(x = component,
                 y = cumulative)) +
  geom_line(aes(x = component,
                y = cumulative)) +
  labs(x = "Principal Component",
       y = "Cumulative Proportion of Variance Explained") +
  ylim(c(0, 1)) +
  theme_classic()
cpve_plot
```




```
# using patchwork package again
pve_plot + cpve_plot
```

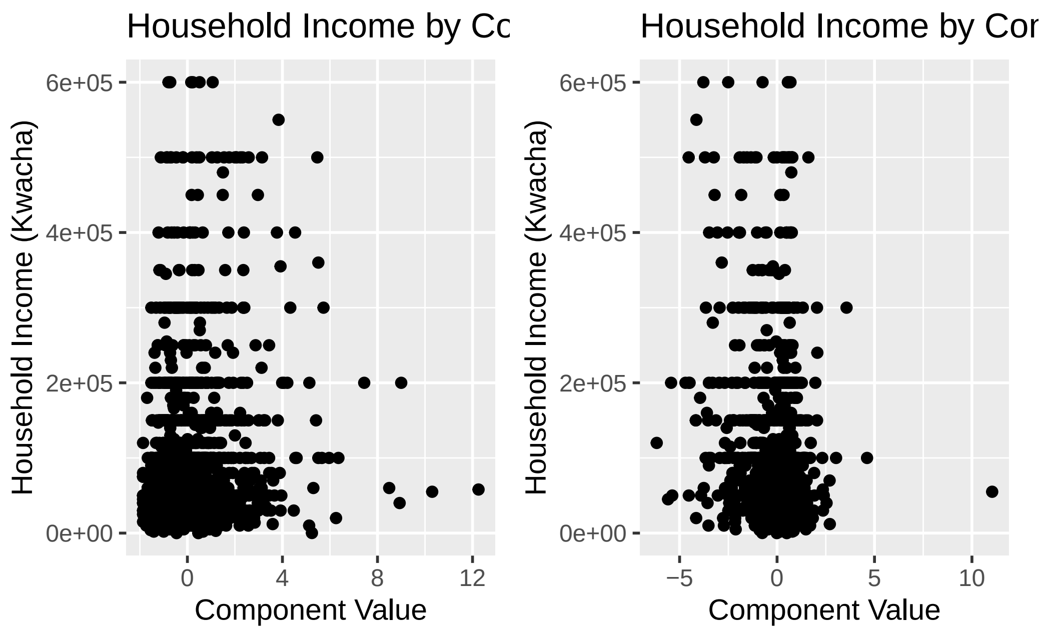


There is an elbow, and it appears to be at the second component. 2 components seems to be sufficient based on this.

Q4

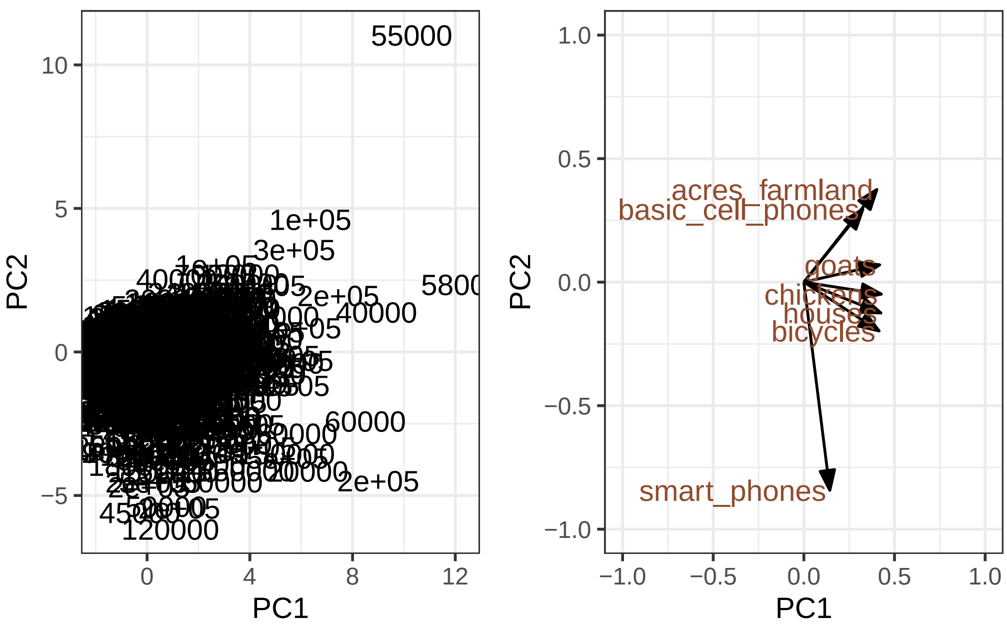
Plot each of these components (the optimal number based on Q3) against household income. Arrange the plots using the patchwork package as demonstrated in this module's example code. Which component(s) seem to proxy for income, if any?

```
# comp_vd this has all of the points and the associated component values
# so we need to extract the exact row that corresponds with the component we care about
# that was not in the example tho
# so maybe not

# plots the component values against income
pca1 <- ggplot(vd, aes(x = comp_vd$PC1, y = hh_income_trim_99)) +
 geom_point() +
  labs(title = 'Household Income by Component 1', x = 'Component Value', y = 'Household In

pca2 <- ggplot(vd, aes(x = comp_vd$PC2, y = hh_income_trim_99)) +
 geom_point() +
 labs(title = 'Household Income by Component 2', x = 'Component Value', y = 'Household Inc

pca1 + pca2
```




```
# these are the side by side plots from the code example
# define arrow style for plotting
arrow_style <- arrow(
  angle = 20, ends = "first", type = "closed", length = grid::unit(8, "pt")
)

rot <- pr_vd %>%
  tidy(matrix = "rotation") %>%
  pivot_wider(names_from = "PC", names_prefix = "PC", values_from = "value")

scores <- pr_vd$x %>%
  as_tibble() %>%  # turn matrix into tibble for ggplot
  mutate(id = 1:n()) %>% # add ID vector in to identify observations
  ggplot(aes(PC1, PC2)) +
  geom_text(aes(label = vd$hh_income_trim_99)) +
  theme_bw()

loadings <- rot %>%
  ggplot(aes(PC1, PC2)) +
  geom_segment(xend = 0, yend = 0, arrow = arrow_style) +
  geom_text(
    aes(label = column),
    hjust = 1, nudge_x = -0.02,
```

```r
    color = "#904C2F"
  ) +
  xlim(-1, 1) + ylim(-1, 1) +
  theme_bw()

scores + loadings
```



Smartphones, basic cell phones, and acres of farmland appear to be the closest to income because of the direction and placement of the arrows.

Clustering

Q5

```r
#sets seed
set.seed(67)

# subsets so only numerical possessive are included
vd_pos <- vd[, 12:18]

# k means
```

```r
km_out <- kmeans(vd_pos, 10, nstart = 30)
```

Q5p1

```r
# gets cluster assignment for each point
km_clusters <- km_out$cluster

# need to group by cluster
vd_pos$cluster <- km_clusters
vd$cluster_km <- km_clusters

# gets cluster avgs for each possession variable
houses_km <- with(vd_pos, tapply(houses, cluster, mean))
af_km <- with(vd_pos, tapply(acres_farmland, cluster, mean))
bikes_km <- with(vd_pos, tapply(bicycles, cluster, mean))
chickens_km <- with(vd_pos, tapply(chickens, cluster, mean))
goats_km <- with(vd_pos, tapply(goats, cluster, mean))
cell_km <- with(vd_pos, tapply(basic_cell_phones, cluster, mean))
smart_km <- with(vd_pos, tapply(smart_phones, cluster, mean))

# prints all cluster avgs
# cbind(houses_km, af_km, bikes_km, chickens_km, goats_km, cell_km, smart_km)
```

```r
# ptings all cluster avgs
km_out$centers
```

```
     houses acres_farmland  bicycles  chickens      goats basic_cell_phones
1  2.666667       7.166667 1.3333333 71.833333  1.1666667          1.500000
2  1.931034       4.840517 1.7241379  5.741379 13.5517241          1.344828
3  1.000000      80.000000 0.0000000  0.000000  0.0000000          3.000000
4  1.224274       1.654881 1.0791557  4.730871  1.0897098          1.229551
5  1.035714       1.004487 0.7692308  0.217033  0.4395604          1.092491
6  1.687500       2.011719 1.2968750 10.062500  1.4427083          1.312500
7  1.606061       2.550505 1.7373737 17.949495  2.2020202          1.545455
8  2.000000      11.000000 1.5000000  0.000000 90.0000000          1.500000
9  1.727273       2.426136 1.3636364 31.636364  2.7045455          1.636364
10 2.120879       7.016484 1.0439560  1.329670  1.0989011          1.450549
   smart_phones
1     0.8333333
```

```
2      0.6379310
3      0.0000000
4      0.2955145
5      0.3608059
6      0.3072917
7      0.5151515
8      1.0000000
9      0.4318182
10     0.4285714
```

Q5p2

```
# gets hh income mean of each cluster
cluster_mean <- with(vd, tapply(hh_income_trim_99, cluster_km, mean))
cluster_mean
```

```
        1          2          3          4          5          6          7          8
186666.67 118312.93  55000.00  71823.22  79798.63  89765.64 119676.77  49000.00
        9         10
125409.09  85582.42
```

Q5p3

```
# shows the number of instances of a household income by each cluster
vd %>%
  group_by(cluster_km, hh_income_cat10) %>%
  tally() %>%
  spread(cluster_km, n)
```

```
# A tibble: 10 x 11
   hh_income_cat10     `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`
   <fct>             <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
 1 [2,1.5e+04]          NA     3    NA    42   117    20     3    NA    NA    11
 2 (1.5e+04,2.5e+04]    NA     6    NA    47   105    13     6    NA     1     9
 3 (2.5e+04,3.5e+04]    NA     3    NA    57   125    16    13    NA     6     9
 4 (3.5e+04,4.5e+04]    NA     4    NA    38    84    16     5     1     2     6
 5 (4.5e+04,5e+04]      NA     5    NA    47   159    27    17    NA     9    11
 6 (5e+04,6e+04]         1     3     1    22    72    17     7     1     3     6
 7 (6e+04,8.74e+04]     NA     9    NA    33   104    16     7    NA     2     5
```

```
 8 (8.74e+04,1e+05]       1      9     NA     37    135     21     11     NA      6     10
 9 (1e+05,2e+05]          3      8     NA     36    127     30     18     NA      8     17
10 (2e+05,6e+05]          1      8     NA     20     64     16     12     NA      7      7
```

```
# maybe this will be easier to rank...
vd %>%
  group_by(hh_income_cat10, cluster_km) %>%
  tally() %>%
  spread(hh_income_cat10, n)
```

```
# A tibble: 10 x 11
   cluster_km `[2,1.5e+04]` `(1.5e+04,2.5e+04]` `(2.5e+04,3.5e+04]`
        <int>         <int>                 <int>                 <int>
 1          1            NA                    NA                    NA
 2          2             3                     6                     3
 3          3            NA                    NA                    NA
 4          4            42                    47                    57
 5          5           117                   105                   125
 6          6            20                    13                    16
 7          7             3                     6                    13
 8          8            NA                    NA                    NA
 9          9            NA                     1                     6
10         10            11                     9                     9
# i 7 more variables: `(3.5e+04,4.5e+04]` <int>, `(4.5e+04,5e+04]` <int>,
#   `(5e+04,6e+04]` <int>, `(6e+04,8.74e+04]` <int>, `(8.74e+04,1e+05]` <int>,
#   `(1e+05,2e+05]` <int>, `(2e+05,6e+05]` <int>
```

The clustering didn't seem to work very well. It was really hard to rank income clusters by income groups because there was so much variation within clusters. I would rank the clusters from 2, 8, 3, 1, 10, 9, 5, 6, 4, 7. I chose this order by trying to balance the number of households in each income bracket, with the ones with the highest proportion of higher income households being in the wealthier positions on the list and the ones with more lower income households on the less wealthy side of the list.

Q6

Repeat the previous, but this time use hierarchical clustering with average linkage. Cut the tree a 10 clusters. Answer the same three questions. Finally: is there any overlap between the clusters created through hierarchical clustering and the clusters created using -means? Note: If focusing on prediction, we could use cross-validation to try to see which linkage approach works best, as the section on clustering in ISLR points out. We'll talk more about that during the homework for this module

```
# gets rid of nas in response
vendor_data_rm <- na.omit(vendor_data)
# remember not to scale because of instructions

# sets up distances
data_dist <- dist(vendor_data_rm)

# clusters
al_nci <- hclust(data_dist, method = "average") %>%
  ggdendrogram() +
  labs(title = "Average Linkage")

# cuts tree into 10 clusters
hc_out <- hclust(dist(vendor_data_rm))
hc_clusters <- cutree(hc_out, 10)
table(hc_clusters)
```

```
hc_clusters
   1    2    3    4    5    6    7    8    9   10
 356 1278  110   44  112   28    7   27    1    1
```

Q5p1

```
vd_pos$hc_clusters <- hc_clusters
vd$hc_clusters <- hc_clusters

# gets cluster avgs for each possession variable
houses_hc <- with(vd_pos, tapply(houses, hc_clusters, mean))
af_hc <- with(vd_pos, tapply(acres_farmland, hc_clusters, mean))
bikes_hc <- with(vd_pos, tapply(bicycles, hc_clusters, mean))
chickens_hc <- with(vd_pos, tapply(chickens, hc_clusters, mean))
goats_hc <- with(vd_pos, tapply(goats, hc_clusters, mean))
cell_hc <- with(vd_pos, tapply(basic_cell_phones, hc_clusters, mean))
smart_hc <- with(vd_pos, tapply(smart_phones, hc_clusters, mean))

# prints all cluster avgs
cbind(houses_hc, af_hc, bikes_hc, chickens_hc, goats_hc, cell_hc, smart_hc)
```

```
   houses_hc    af_hc  bikes_hc chickens_hc goats_hc  cell_hc  smart_hc
1   1.233146 1.786376 0.9803371    3.896067 1.398876 1.272472 0.4662921
2   1.211268 1.751487 0.9381847    3.695618 1.160407 1.135368 0.2480438
3   1.509091 1.940909 1.0000000    4.781818 1.800000 1.354545 0.7090909
4   1.454545 2.488636 1.0227273    4.272727 2.068182 1.340909 0.7045455
5   1.428571 1.968750 1.2500000    5.589286 1.428571 1.401786 0.5892857
6   1.321429 1.732143 1.4285714   10.821429 2.535714 1.071429 0.8571429
7   1.142857 1.142857 1.0000000    4.000000 2.714286 2.142857 1.7142857
8   2.000000 2.129630 1.5185185    7.962963 2.222222 1.666667 0.8518519
9   1.000000 1.000000 4.0000000   20.000000 0.000000 1.000000 1.0000000
10  3.000000 1.000000 1.0000000    0.000000 0.000000 1.000000 0.0000000
```

Q6p2

```r
# I wasn't sure if categories meant the possession variables or clusters
# I think it meant the variables but I added both just in case
# prints avg household income in each of the clusters
with(vd, tapply(hh_income_trim_99, hc_clusters, mean))
```

```
        1         2         3         4         5         6         7         8
 97955.06  37837.37 150610.00 298409.09 208705.36 378928.57 592857.14 491851.85
        9        10
500000.00  45000.00
```

```r
# prints the avg household income of each of the clusters
hc_mean <- with(vd, tapply(hh_income_trim_99, hc_clusters, mean))
hc_mean
```

```
        1         2         3         4         5         6         7         8
 97955.06  37837.37 150610.00 298409.09 208705.36 378928.57 592857.14 491851.85
        9        10
500000.00  45000.00
```

```r
# prints avg household income in each of the possession categories
#houses_hc_avg <- with(vd, tapply(hh_income_trim_99,houses, mean))
#af_hc_avg <- with(vd, tapply(hh_income_trim_99,acres_farmland, mean))
##bikes_hc_avg <- with(vd, tapply(hh_income_trim_99,bicycles, mean))
#chickens_hc_avg <- with(vd, tapply(hh_income_trim_99,chickens, mean))
#goats_hc_avg <- with(vd, tapply(hh_income_trim_99, goats, mean))
```

```
#cell_hc_avg <- with(vd, tapply(hh_income_trim_99, basic_cell_phones, mean))
#smart_hc_avg <- with(vd, tapply(hh_income_trim_99, smart_phones, mean))
# prints all category avgs
#cbind(houses_hc_avg, af_hc_avg, bikes_hc_avg, chickens_hc_avg, goats_hc_avg, cell_hc_avg,
```

Q6p3

```
# shows the instances of each cluster as the number
# of times it shows up in the corresponding hh income level
vd %>%
  group_by(hc_clusters, hh_income_cat10) %>%
  tally() %>%
  spread(hc_clusters, n)
```

```
# A tibble: 10 x 11
   hh_income_cat10     `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`
   <fct>             <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
 1 [2,1.5e+04]          NA   196    NA    NA    NA    NA    NA    NA    NA    NA
 2 (1.5e+04,2.5e+04]    NA   187    NA    NA    NA    NA    NA    NA    NA    NA
 3 (2.5e+04,3.5e+04]    NA   229    NA    NA    NA    NA    NA    NA    NA    NA
 4 (3.5e+04,4.5e+04]    NA   155    NA    NA    NA    NA    NA    NA    NA     1
 5 (4.5e+04,5e+04]      NA   275    NA    NA    NA    NA    NA    NA    NA    NA
 6 (5e+04,6e+04]        NA   133    NA    NA    NA    NA    NA    NA    NA    NA
 7 (6e+04,8.74e+04]     73   103    NA    NA    NA    NA    NA    NA    NA    NA
 8 (8.74e+04,1e+05]    230    NA    NA    NA    NA    NA    NA    NA    NA    NA
 9 (1e+05,2e+05]        53    NA   110    NA    84    NA    NA    NA    NA    NA
10 (2e+05,6e+05]        NA    NA    NA    44    28    28     7    27     1    NA
```

The clustering here seemed to work much much better. There appears to be a much clearer grouping of clusters based on the distribution of income groups included in the clusters. The ranking from least to most wealthy would be 2, 10, 1, 3, 5, 9, 7, 8, 6, 4. This was based on the number of households in each of the categories then by the total number in the specific category because there were a lot in the last one.

k means order: 2, 8, 3, 1, 10, 9, 5, 6, 4, 7.

Q6P4

4. is there any overlap between the clusters created through hierarchical clustering and the clusters created using -means?

Yes, there is some overlap, but seems mostly because the K-means clustering is so spread out that it happens to overlap with some of the data points on the hierarchical clustering. In terms of ordering, there was some overlap, with 2 being the lowest and 4/6 being toward the higher end, but overall not a lot because the distribution in the k means was so spread out compared to the hc.