Clustering Example 9: College Scorecard and k-means Clustering

Amy Wagaman, Nathan Carter

July 2020

Load necessary libraries.

```
library(mosaic)
library(cluster)
```

This requires you to have access to the `college.rda` file. It is available in the book's GitHub repository at the following URL.

https://github.com/ds4m/ds4m.github.io/tree/master/chapter-5-resources/college.rda

If you run this R code, place the data file in the same folder as the code file.

```
load("college.rda")
```

The data was originally retrieved from https://collegescorecard.ed.gov/data/ and a corresponding data dictionary can be found at https://collegescorecard.ed.gov/data/documentation/. We summarize a portion of the data dictionary here for convenience.

Code	Control level
1	Public
2	Private nonprofit
3	Private for-profit

Code	Region
0	U.S. Service Schools - none in this subset
1	New England (CT, ME, MA, NH, RI, VT)
2	Mid East (DE, DC, MD, NJ, NY, PA)
3	Great Lakes (IL, IN, MI, OH, WI)
4	Plains (IA, KS, MN, MO, NE, ND, SD)
5	Southeast (AL, AR, FL, GA, KY, LA, MS, NC, SC, TN, VA, WV)
6	Southwest (AZ, NM, OK, TX)
7	Rocky Mountains (CO, ID, MT, UT, WY)
8	Far West (AK, CA, HI, NV, OR, WA)
9	Outlying Areas (AS, FM, GU, MH, MP, PR, PW, VI)

Code	Locale
11	City: Large
12	City: Midsize
13	City: Small
21	Suburb: Large

Code	Locale
22	Suburb: Midsize
23	Suburb: Small
31	Town: Fringe
32	Town: Distant
33	Town: Remote
41	Rural: Fringe
42	Rural: Distant
43	Rural: Remote

Select relevant variables and convert their data types.

```
data <- select(CollegeScorecardMostRecent, INSTNM, STABBR, CONTROL, REGION,
               LOCALE, ADM_RATE, ACTCMMID, SAT_AVG, UGDS, TUITFTE, AVGFACSAL,
               PCTPELL, PFTFAC, COMP_ORIG_YR4_RT, COMP_ORIG_YR6_RT,
               PAR_ED_PCT_1STGEN, GRAD_DEBT_MDN, AGE_ENTRY,MD_FAMINC,
               MEDIAN_HH_INC)

data <- mutate(data,
               COMP_ORIG_YR4_RT = as.numeric(COMP_ORIG_YR4_RT),
               COMP_ORIG_YR6_RT = as.numeric(COMP_ORIG_YR6_RT),
               PAR_ED_PCT_1STGEN = as.numeric(PAR_ED_PCT_1STGEN),
               GRAD_DEBT_MDN = as.numeric(GRAD_DEBT_MDN),
               AGE_ENTRY = as.numeric(AGE_ENTRY),
               MD_FAMINC = as.numeric(MD_FAMINC),
               MEDIAN_HH_INC = as.numeric(MEDIAN_HH_INC))
```

```
## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion

## Warning: NAs introduced by coercion
```

Remove observations with missing values (NAs).

Also print summary information for the dataset; this includes showing how many values in each column are missing (NA's).

```
complete <- select(data, -INSTNM, -STABBR, -CONTROL, -REGION, -LOCALE,
                   -MEDIAN_HH_INC, -ACTCMMID, -ADM_RATE, -SAT_AVG, -PFTFAC,
                   -AVGFACSAL)
summary(complete)
```

```
##       UGDS           TUITFTE           PCTPELL        COMP_ORIG_YR4_RT
##  Min.   :    0   Min.   :     0   Min.   :0.0000   Min.   :0.0062
```

```
##  1st Qu.:  106   1st Qu.:  4918   1st Qu.:0.3117   1st Qu.:0.2652
##  Median :  401   Median :  9156   Median :0.4630   Median :0.5000
##  Mean   : 2427   Mean   : 10767   Mean   :0.4821   Mean   :0.4710
##  3rd Qu.: 2019   3rd Qu.: 13906   3rd Qu.:0.6522   3rd Qu.:0.6552
##  Max.   :77269   Max.   :712078   Max.   :1.0000   Max.   :0.9515
##  NA's   :748     NA's   :468      NA's   :770      NA's   :1373
##  COMP_ORIG_YR6_RT PAR_ED_PCT_1STGEN GRAD_DEBT_MDN     AGE_ENTRY
##  Min.   :0.0029   Min.   :0.0887    Min.   : 1510   Min.   :17.43
##  1st Qu.:0.2187   1st Qu.:0.3762    1st Qu.: 9500   1st Qu.:23.17
##  Median :0.4672   Median :0.4762    Median :13826   Median :25.78
##  Mean   :0.4378   Mean   :0.4555    Mean   :16317   Mean   :26.01
##  3rd Qu.:0.6210   3rd Qu.:0.5435    3rd Qu.:23584   3rd Qu.:28.50
##  Max.   :0.9294   Max.   :0.9573    Max.   :47000   Max.   :58.90
##  NA's   :1437     NA's   :1247      NA's   :1453    NA's   :500
##    MD_FAMINC
##  Min.   :     0
##  1st Qu.: 16094
##  Median : 21994
##  Mean   : 28133
##  3rd Qu.: 33989
##  Max.   :179864
##  NA's   :500
```

```
complete <- complete[complete.cases(complete),]
glimpse(complete)
```

```
## Observations: 4,546
## Variables: 9
## $ UGDS              <dbl> 4824, 12866, 322, 6917, 4189, 32387, 1404, 2801, 42...
## $ TUITFTE           <dbl> 9227, 11612, 14738, 8727, 9003, 13574, 1580, 6713, ...
## $ PCTPELL           <dbl> 0.7100, 0.3532, 0.7415, 0.2765, 0.7377, 0.1800, 0.4...
## $ COMP_ORIG_YR4_RT  <dbl> 0.19082126, 0.45396707, 0.11707317, 0.44937833, 0.2...
## $ COMP_ORIG_YR6_RT  <dbl> 0.29772727, 0.50694981, 0.22222222, 0.47475642, 0.2...
## $ PAR_ED_PCT_1STGEN <dbl> 0.3658281, 0.3412237, 0.5125000, 0.3101322, 0.34343...
## $ GRAD_DEBT_MDN     <dbl> 34500, 22500, 25002, 22021, 32637, 23250, 12447, 18...
## $ AGE_ENTRY         <dbl> 20.28374, 23.60797, 33.67230, 22.72792, 20.13099, 2...
## $ MD_FAMINC         <dbl> 23553.0, 34489.0, 15033.5, 44787.0, 22080.5, 66733....
```

Run k-means clustering for $k = 1$ to $k = 20$.

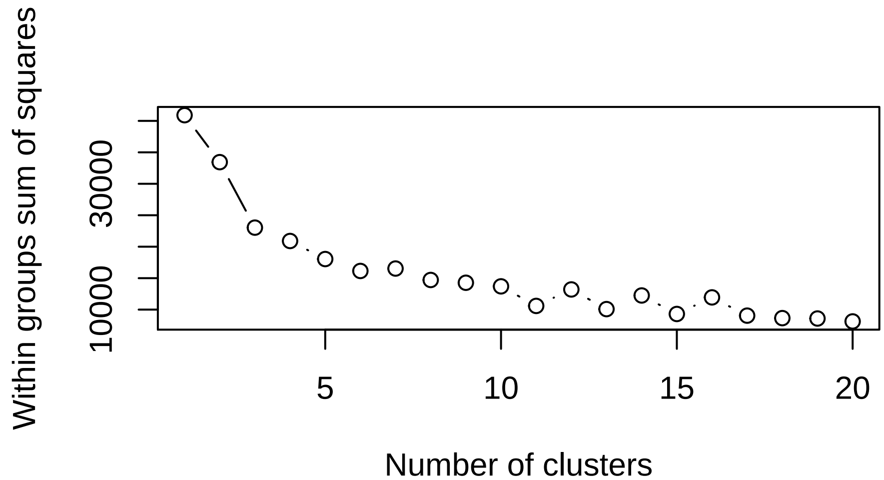
We plot the within-groups sum of squares for each run, so that we can assess which k value may be best. Note the choice of a random number seed, for reproducibility.

```
set.seed(240)
wss <- rep(0, 20)
for (i in 1:20) {
  wss[i] <- sum(kmeans(scale(complete), centers = i)$withinss)
}
```

```
## Warning: did not converge in 10 iterations

## Warning: did not converge in 10 iterations
```

```
plot(1:20, wss, type = "b",
     xlab = "Number of clusters", ylab = "Within groups sum of squares")
```



Run k-means with the chosen value of $k = 4$.

And print the results, which include cluster centroids, the clustering partition as a vector, and the within-groups sum of squares.

```r
set.seed(304)
Ksol1 <- kmeans(scale(complete), centers = 4) #centers is the # of clusters
list(Ksol1)
```

```
## [[1]]
## K-means clustering with 4 clusters of sizes 739, 1559, 1203, 1045
##
## Cluster means:
##         UGDS     TUITFTE    PCTPELL COMP_ORIG_YR4_RT COMP_ORIG_YR6_RT
## 1 -0.3112588  0.34844677  0.6337379       -0.8723507       -0.7602608
## 2 -0.4580810 -0.02966638  0.6934096        0.8190212        0.7344870
## 3  0.3055813  0.32718985 -0.8473379        0.4428018        0.5142920
## 4  0.5517265 -0.57881499 -0.5071869       -1.1147152       -1.1501682
##   PAR_ED_PCT_1STGEN GRAD_DEBT_MDN  AGE_ENTRY  MD_FAMINC
## 1         0.4115626     1.1205933  0.9423235 -0.4708501
## 2         0.6950675    -0.7846063  0.5024887 -0.6269660
## 3        -1.2440464     0.8437308 -1.0114104  1.3529469
## 4         0.1041463    -0.5932300 -0.2517036 -0.2891837
##
## Clustering vector:
##    [1] 1 3 1 3 1 3 4 2 4 3 3 4 1 1 4 4 1 4 1 3 3 4 4 1 1 4 1 1 3 4 3 4 3 3 4 3 1
##   [38] 1 4 3 4 4 4 2 2 2 2 2 2 2 1 2 2 3 4 3 1 4 1 2 4 2 2 2 2 3 4 4 4 4 4 2 3 1
##   [75] 4 2 4 2 2 1 2 4 4 4 4 2 1 2 4 4 2 1 2 3 2 3 1 4 4 4 4 2 2 4 4 4 4 4 2 2 4
##  [112] 4 3 4 3 2 3 2 2 4 3 4 4 3 4 1 4 4 2 2 3 4 4 4 4 1 2 4 3 2 4 4 3 2 2 3 4 2
##  [149] 3 2 1 4 4 3 3 3 3 4 4 2 2 4 3 2 4 4 4 4 4 4 4 3 3 3 3 3 3 3 3 2 2 3 3 4 4
##  [186] 2 2 2 2 2 2 2 2 4 4 3 2 3 1 4 4 2 4 3 1 3 4 4 4 4 2 4 3 4 4 2 2 4 3 3 3 4
##  [223] 2 4 4 1 4 2 1 4 4 3 4 1 2 2 4 2 1 4 3 1 1 3 2 4 3 3 4 4 4 4 4 4 3 4 2 4 4
##  [260] 1 3 2 2 2 2 2 2 2 2 4 3 4 3 4 3 4 4 2 3 4 4 4 2 4 2 1 1 2 2 2 2 2 1 3 4 3
##  [297] 3 2 1 3 3 4 4 3 3 2 3 4 3 4 2 2 4 4 1 4 4 3 4 1 4 4 3 3 4 3 4 2 3 4 4 4 2
##  [334] 2 4 3 4 4 4 4 2 4 2 4 3 4 3 1 3 2 4 3 2 2 2 4 4 4 1 4 4 2 3 3 3 2 4 4 4 3
##  [371] 2 3 3 2 3 1 3 2 1 4 4 3 2 3 1 4 1 2 4 2 3 3 4 2 2 4 4 4 4 3 1 1 4 3 4 2 4
##  [408] 4 4 3 1 4 2 2 4 2 3 1 4 2 1 1 3 1 2 3 2 3 1 2 3 3 2 4 2 3 4 4 4 4 3 4 2 2
##  [445] 3 4 4 2 1 3 3 3 4 3 2 3 4 3 3 3 2 4 3 3 3 2 1 1 3 3 1 2 3 3 3 1 1 2 1 1 1
##  [482] 1 3 4 4 2 4 3 3 4 1 3 4 4 3 3 4 4 3 3 2 2 2 1 1 4 4 1 2 2 3 3 2 3 4 1 4 4
```

```
##  [519] 2 2 4 1 2 1 3 1 4 4 2 1 4 1 4 3 2 2 3 3 4 3 4 4 4 4 3 3 1 4 4 1 4 4 4 3 3
##  [556] 3 3 2 4 1 3 4 1 4 1 3 1 4 4 3 1 1 4 1 4 1 2 1 3 4 2 3 4 3 1 4 4 4 1 4 4 4
##  [593] 4 2 1 4 3 3 4 1 3 4 3 3 4 3 4 4 2 2 4 3 4 1 3 3 4 3 1 3 4 4 3 1 1 3 1 3 3
##  [630] 4 4 3 3 3 1 4 3 2 3 4 4 4 4 4 4 4 2 3 2 4 3 3 4 2 4 3 4 4 3 3 3 3 2 4 4 1
##  [667] 3 2 3 2 2 2 2 2 4 1 3 4 4 4 4 4 4 4 3 3 2 4 2 3 2 4 1 3 4 3 3 2 2 2 3 2 2
##  [704] 2 4 3 3 3 3 4 3 4 3 1 3 4 4 4 4 3 4 4 4 3 4 3 4 4 3 4 4 4 2 3 2 2 1 3 4 3
##  [741] 3 1 3 3 4 4 4 2 1 3 3 3 1 3 4 4 3 4 4 3 1 1 4 4 1 4 2 3 3 3 3 3 3 3 4 2 2
##  [778] 3 3 3 1 4 2 4 2 3 4 4 3 3 1 3 1 1 2 3 3 3 3 3 3 3 3 2 3 4 3 3 1 3 1 3 4 4
##  [815] 4 3 4 4 4 2 2 2 1 2 2 3 3 1 1 1 3 3 2 2 3 2 3 1 3 2 3 3 3 4 3 3 2 2 3 3 3
##  [852] 3 4 3 3 3 4 4 3 3 3 1 1 2 4 4 4 4 2 3 3 4 3 4 2 2 2 3 3 1 4 3 3 3 1 4 3 3
##  [889] 4 3 2 3 2 2 3 4 4 2 1 1 3 2 4 1 4 3 4 3 3 3 1 4 1 4 4 4 4 4 4 3 4 4 1 1 4
##  [926] 4 2 3 4 4 4 4 2 4 3 3 2 3 3 4 4 4 3 3 3 4 3 4 1 1 3 4 3 2 4 1 3 3 4 4 3 2
##  [963] 4 3 4 2 3 2 2 4 1 4 4 3 2 2 4 3 4 3 2 4 2 4 4 4 2 1 1 3 3 3 1 3 4 4 1 3 3
## [1000] 4 3 1 4 1 4 1 2 2 2 2 4 4 3 1 3 3 2 4 3 4 2 4 2 3 2 2 4 2 1 4 1 2 3 4 3 4
## [1037] 4 2 2 4 3 3 4 4 4 3 4 1 1 2 2 4 2 4 1 1 4 3 4 3 3 1 3 2 3 2 4 3 4 4 1 3 4
## [1074] 4 3 3 4 3 4 2 3 3 2 4 3 3 3 4 4 4 2 2 2 1 4 3 4 4 1 1 2 4 3 4 3 1 4 4 3 4
## [1111] 2 3 2 3 4 2 3 3 3 3 2 2 4 3 3 1 3 4 2 2 3 3 2 3 3 3 3 4 1 3 3 3 3 3 2 1 2
## [1148] 3 3 3 4 2 3 3 3 3 4 2 4 1 4 2 1 3 3 3 2 3 3 3 3 2 1 3 3 2 3 4 3 3 3 4 3 1
## [1185] 3 3 2 3 3 4 4 3 3 3 3 4 3 4 3 3 3 4 1 3 3 4 3 4 3 1 4 4 3 3 2 3 3 3 4 3 3
## [1222] 3 2 3 3 3 3 3 3 3 3 3 3 3 2 3 4 3 3 4 4 3 2 3 4 2 1 3 3 1 2 4 2 3 2 2 3 3
## [1259] 2 4 3 4 1 4 3 4 3 4 3 4 4 4 2 4 3 4 3 4 1 1 1 3 3 3 3 3 3 4 4 4 2 4 2 4 3 4
## [1296] 3 4 3 3 2 2 2 3 4 1 4 4 2 3 3 2 2 4 4 4 4 3 1 3 4 4 3 4 3 3 4 3 3 3 4 4 4
## [1333] 4 3 3 4 4 3 4 4 3 4 3 1 4 4 3 3 2 3 4 1 2 3 3 3 3 4 1 4 3 4 3 3 4 4 4 3 4
## [1370] 3 3 3 3 3 3 3 2 3 1 4 3 4 3 4 4 3 4 1 1 4 1 4 4 4 4 2 4 4 4 1 4 4 3 3 2 1
## [1407] 3 4 3 4 4 4 1 4 2 1 1 3 2 2 1 2 3 1 3 1 3 4 3 1 4 2 2 3 3 4 4 1 2 2 4 3 3
## [1444] 4 3 3 2 4 3 2 2 2 4 3 4 4 4 3 3 3 3 4 3 3 3 4 1 2 3 2 3 3 4 3 4 2 4 3 3 3
## [1481] 4 4 3 3 3 3 3 4 3 4 4 4 3 4 4 4 3 3 4 3 4 4 1 3 4 4 2 3 3 3 3 3 2 3 1 3 4
## [1518] 4 3 3 3 1 3 3 3 4 4 3 4 3 3 3 4 3 2 2 4 1 2 4 3 4 3 4 4 3 3 2 2 3 2 1 4 3
## [1555] 3 3 3 4 4 4 4 4 3 3 1 2 4 2 4 1 2 1 4 4 3 4 3 2 2 2 4 4 3 2 2 3 1 3 2 3 3
## [1592] 4 2 2 2 4 4 3 2 2 2 4 3 3 2 2 2 3 4 2 2 4 2 3 3 2 3 3 3 3 1 4 3 2 1 4 3 3
## [1629] 2 4 3 4 3 2 4 2 4 4 4 4 1 4 4 4 3 4 4 4 4 4 4 4 4 4 4 4 3 1 4 4 1 3 4 1 3 3
## [1666] 2 3 1 3 2 3 3 2 1 2 2 1 1 2 2 4 1 1 1 1 1 1 3 2 2 4 3 2 3 4 3 1 3 4 2 1 2 2
## [1703] 2 3 3 4 3 3 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 3 3 3 4 1 3 4 3 4 3 3 2 4
## [1740] 4 3 3 2 4 3 3 3 3 4 2 3 3 4 2 4 1 3 3 1 3 2 1 1 2 3 3 1 2 3 3 1 1 2 2 1 2
## [1777] 4 3 2 4 3 3 4 3 1 3 2 3 4 3 4 3 1 4 4 3 3 2 1 3 3 3 3 3 3 4 2 3 3 3 3 3 3
## [1814] 3 2 3 4 3 3 3 2 3 2 3 2 4 3 4 4 4 4 4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4 3
## [1851] 2 1 3 3 2 3 4 4 1 4 4 3 3 3 1 3 3 3 1 4 3 3 1 3 4 3 4 3 1 4 3 3 4 3 4 3 4
## [1888] 4 1 4 4 3 3 4 3 1 1 3 4 1 4 3 3 3 4 3 1 1 4 3 3 1 4 3 3 1 1 2 3 1 2 4 3 3
## [1925] 3 3 3 3 3 1 3 3 4 3 4 3 1 4 1 3 1 2 3 4 4 3 4 3 4 4 4 4 3 3 3 1 3 3 2 3 4
## [1962] 3 3 3 3 4 4 3 4 4 4 3 2 1 1 3 3 1 2 3 1 2 2 3 4 3 2 3 3 1 2 2 2 3 3 3 4 2
## [1999] 1 1 2 3 3 3 4 3 3 1 4 3 2 3 2 3 4 4 1 3 3 3 4 2 1 3 1 4 3 2 2 3 3 3 4 2 1
## [2036] 4 3 2 2 3 3 3 3 3 3 3 3 3 3 4 2 4 2 2 4 3 3 3 4 2 3 3 3 1 2 3 3 3 3 4 3 2
## [2073] 4 4 2 3 3 3 2 3 2 3 3 3 3 3 3 2 2 2 3 3 3 3 3 3 3 2 3 3 4 2 2 2 1 1 3 4 4
## [2110] 1 4 1 3 4 4 4 3 2 2 3 1 3 4 1 3 3 3 3 3 3 4 2 1 3 3 2 4 4 2 4 4 4 4 4 2 1
## [2147] 1 4 2 4 4 4 4 3 3 4 3 4 3 4 3 3 4 3 2 4 4 2 4 4 4 1 4 4 2 3 4 2 1 2 4 4 4
## [2184] 4 4 2 2 3 2 2 1 2 3 2 4 3 3 4 2 1 2 4 3 3 2 3 3 3 3 3 2 2 2 2 4 4 3 4 2 4
## [2221] 4 4 4 1 3 2 3 4 2 3 3 2 4 3 2 3 3 2 3 1 2 1 2 3 4 3 3 2 3 3 4 2 4 3 3 3 3
## [2258] 3 1 3 1 1 2 2 3 2 2 4 3 3 2 3 3 3 3 3 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 3 3
## [2295] 2 3 1 4 3 3 3 2 3 3 2 2 2 3 3 3 2 3 3 3 4 3 3 2 2 3 4 3 2 2 2 1 3 4 3 1 3
## [2332] 3 1 2 3 3 2 3 3 4 3 3 3 3 2 3 2 4 2 2 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
## [2369] 3 3 1 2 2 2 3 3 3 3 4 3 3 3 3 3 2 2 3 4 2 1 2 3 2 1 3 3 2 3 3 3 2 3 3 2 2
## [2406] 2 3 3 2 3 3 3 2 2 2 2 3 3 2 3 3 3 2 3 2 3 3 3 3 3 2 3 3 3 1 2 3 3 4 3 3 3
## [2443] 3 2 1 4 1 3 3 4 1 3 1 2 3 3 3 3 1 3 3 1 3 4 3 4 3 3 4 4 2 2 3 1 4 1 3 3 4
## [2480] 4 3 3 3 3 3 4 4 4 3 1 3 4 4 4 4 4 1 3 4 3 2 3 3 3 2 3 3 3 1 1 3 3 3 3 3 4
```

```
## [2517] 3 3 4 4 1 3 1 4 3 4 3 4 4 2 3 3 4 3 1 3 3 4 3 2 1 1 1 3 2 4 3 2 3 3 3 4 2
## [2554] 2 4 3 4 3 4 3 3 3 3 3 2 1 3 1 4 1 3 3 4 4 2 3 4 4 4 1 2 1 3 4 2 2 3 4 2 2
## [2591] 4 4 4 4 1 4 3 2 3 1 2 3 4 3 2 4 4 4 4 2 2 1 2 4 4 4 1 3 4 4 3 2 4 4 1 4 4
## [2628] 3 3 2 1 2 4 4 4 4 4 4 4 3 2 3 4 3 4 3 2 4 3 1 2 4 2 2 4 4 4 3 4 2 4 2 2 1
## [2665] 4 4 4 4 1 1 4 2 3 4 2 3 4 3 3 4 4 3 2 3 4 2 2 3 2 4 1 3 3 3 3 4 4 4 4 4 2
## [2702] 3 4 4 3 4 3 3 4 4 3 1 3 4 4 2 1 3 1 4 3 2 2 4 2 2 4 2 4 1 4 3 4 4 1 1 3 3
## [2739] 4 2 2 4 2 2 4 4 4 1 1 4 4 4 3 4 3 3 3 3 4 2 3 1 3 4 3 3 1 3 3 3 2 2 3 4 1
## [2776] 3 3 1 4 3 1 1 3 4 4 1 3 3 1 3 4 3 3 4 3 4 4 3 4 3 1 3 3 1 4 3 2 1 4 3 4 1
## [2813] 4 3 2 3 3 2 4 3 3 2 3 3 3 1 3 4 4 4 3 4 3 3 3 1 1 1 3 4 2 1 4 4 4 3 1 2 4
## [2850] 4 4 3 2 3 4 4 2 2 4 3 4 4 1 4 2 2 4 4 4 2 4 3 4 2 3 4 3 4 3 4 4 3 3 4 4 4
## [2887] 4 4 2 4 4 3 3 3 4 3 4 3 3 4 3 3 4 2 3 2 4 3 4 4 1 3 2 1 2 3 4 3 1 3 4 4 3
## [2924] 3 3 2 2 1 4 3 3 2 1 4 3 4 1 3 3 3 3 4 4 1 4 3 3 3 3 4 4 3 3 4 1 4 4 4 3 3
## [2961] 3 1 4 1 4 3 4 4 4 3 2 2 3 3 3 3 3 4 1 3 3 3 3 3 3 3 4 4 4 4 4 4 4 2 2 3 4
## [2998] 4 4 2 4 4 2 4 4 4 4 2 4 3 1 2 2 4 4 4 4 4 4 4 4 2 4 2 4 4 4 4 4 4 4 4 2 4
## [3035] 4 4 4 2 4 4 4 4 3 3 2 1 1 2 4 1 2 2 2 1 2 2 4 1 1 1 1 1 1 2 2 2 2 2 4 4 2 1
## [3072] 2 2 4 1 4 4 2 2 2 3 1 1 2 2 2 2 2 1 2 2 2 4 1 1 2 2 2 1 1 2 2 1 2 2 2 2 4
## [3109] 1 3 2 2 2 2 2 2 2 2 4 2 2 4 4 1 4 4 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 4 2 4 2
## [3146] 1 4 1 2 1 2 1 2 2 1 2 4 1 1 2 2 1 2 2 2 4 2 2 1 2 2 1 1 2 2 2 2 2 2 2 2 2
## [3183] 2 2 2 2 1 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 3 3 2 2 2
## [3220] 2 2 2 2 2 2 4 2 4 1 2 2 2 2 2 2 2 2 1 2 2 1 2 2 4 2 4 2 2 2 2 1 4 2 2 4 4
## [3257] 1 2 2 2 1 1 2 1 1 1 2 1 2 2 2 3 2 2 2 1 2 2 2 2 1 2 4 4 4 2 1 1 1 4 2 2 1
## [3294] 2 2 4 1 2 2 2 2 1 3 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 1 2 2 2
## [3331] 1 1 1 2 2 2 2 2 2 1 2 2 2 2 2 2 2 4 2 2 3 2 3 4 2 2 2 2 2 2 2 2 2 1 2 2 2
## [3368] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 2 2 2 1 2
## [3405] 2 2 2 1 1 2 2 1 1 2 2 4 4 4 2 2 4 3 1 2 2 3 1 2 2 4 2 1 3 2 1 2 2 1 1 1 4
## [3442] 1 2 1 1 1 2 2 2 2 2 2 2 3 2 2 2 2 2 3 2 2 2 2 1 1 2 2 2 1 2 2 3 2 4 1 2 1
## [3479] 3 2 4 2 1 2 2 2 2 2 1 4 1 2 2 2 1 2 2 1 2 1 2 4 2 1 2 1 2 2 2 2 3 3 3 1 4
## [3516] 3 2 2 1 2 2 2 2 1 2 1 1 2 2 2 1 2 1 2 4 1 2 2 1 2 2 4 4 3 1 2 1 2 2 2 2 2
## [3553] 1 2 2 1 2 2 2 2 1 2 2 2 2 2 2 1 4 1 1 1 2 1 1 1 1 1 3 2 1 1 1 2 2 1 2 2 2
## [3590] 2 2 2 1 1 2 2 1 2 1 2 2 2 2 2 2 4 1 2 2 4 1 1 2 1 2 1 1 2 2 3 2 1 2 1 1 2
## [3627] 2 2 1 2 1 2 2 2 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 3 2 1 3 2 4 1 2 2 2 2 1 2
## [3664] 2 1 2 1 1 2 2 1 2 2 2 2 1 2 4 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 4 2
## [3701] 3 1 1 1 2 2 2 2 2 2 4 1 2 2 1 1 2 2 1 2 2 2 2 3 3 1 3 2 1 2 2 2 2 2 2 3 2
## [3738] 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 1 1 4 1 2 2 2 1 2 2 2 2 2 2 1 1 1 2 2 2 2 1
## [3775] 1 2 1 2 2 4 2 2 2 2 4 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 3 1 1 1 2 2 1 1
## [3812] 1 4 1 2 2 1 1 1 1 1 2 2 3 2 3 2 2 2 1 1 2 2 2 1 2 2 2 2 2 2 4 2 2 2 2 2 2
## [3849] 2 2 2 2 2 2 2 2 2 2 2 1 2 3 2 2 2 2 2 1 2 2 2 2 1 1 2 2 2 1 1 1 1 2 2 1 1
## [3886] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 3 2
## [3923] 2 2 2 2 2 2 2 1 3 1 2 2 1 1 1 2 2 1 1 2 2 1 2 2 1 1 2 1 2 2 1 1 1 1 2 2 2
## [3960] 1 1 1 2 1 2 2 2 1 2 3 1 2 2 2 2 1 2 3 1 2 2 2 2 2 2 2 2 4 2 2 2 2 2 2 2 2
## [3997] 2 2 2 2 2 2 2 1 1 1 2 1 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 2 2 2 2 2
## [4034] 2 1 1 2 2 1 1 2 4 4 2 2 2 1 2 2 1 1 1 2 2 2 2 1 3 2 2 1 2 2 2 2 2 1 2 2 2
## [4071] 2 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 3 3 1 2 2
## [4108] 2 2 1 2 1 1 1 2 1 1 1 1 2 2 2 2 2 2 2 2 2 1 1 1 1 1 1 1 2 2 2 2 2 2 2 4 1 2
## [4145] 2 2 2 2 2 2 2 1 1 2 2 2 2 2 2 2 2 2 2 2 1 1 2 2 1 2 2 1 1 2 2 2 2 2 2 2 2
## [4182] 2 2 1 3 2 4 2 2 4 2 2 2 2 2 2 2 2 1 2 2 2 2 1 2 1 1 1 1 1 2 1 1 2 1 2 2 2
## [4219] 1 1 3 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 1 1 2 1 1 2 2 2 2
## [4256] 2 2 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 1 1 1 2 2 2 2
## [4293] 2 2 1 2 1 2 1 3 2 3 2 2 1 1 2 1 2 2 1 1 1 1 2 1 1 1 2 1 2 2 2 2 1 2 1 3 1
## [4330] 1 1 1 2 2 2 1 2 2 2 2 2 2 3 1 1 1 1 1 2 3 2 3 1 2 2 2 2 2 1 2 2 2 1 1 2 1
## [4367] 2 2 2 2 2 2 2 1 2 2 2 2 2 2 2 1 2 2 1 2 2 2 2 2 2 3 4 1 2 1 1 1 1 1 1 1 1
## [4404] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 4 4 3 1 1 2 2 2 4 3 4 1 2 2 2 2 2 2 2 2 2 2
## [4441] 2 2 2 2 3 2 4 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 4 2 1 2 2 4 2 2 1 3 1 2 2
## [4478] 2 2 2 4 2 2 2 2 2 2 2 2 3 2 2 2 3 1 1 1 3 2 2 3 3 2 2 4 2 2 2 2 2 2 2 2 2
```

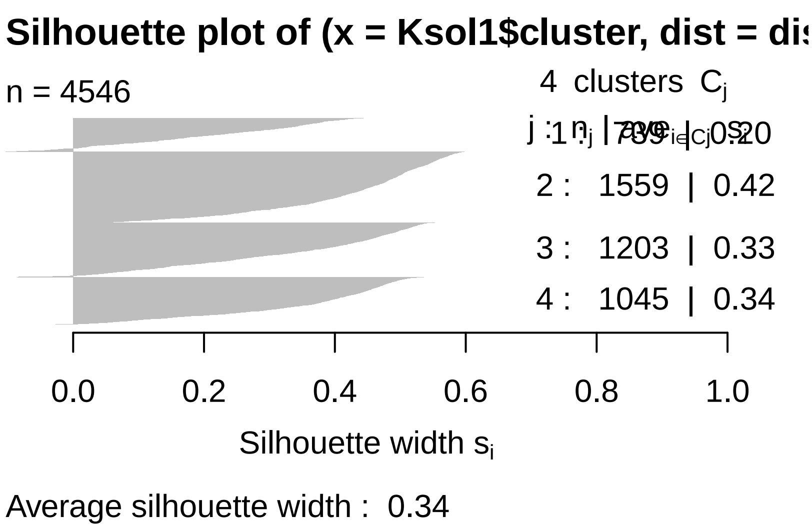
```
## [4515] 2 2 2 2 2 3 2 2 2 3 2 1 2 2 2 2 2 1 2 2 2 2 2 2 1 1 2 2 2 1 1 1
##
## Within cluster sum of squares by cluster:
## [1] 5989.212 4208.402 5635.969 4105.863
##  (between_SS / total_SS =  51.3 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

Create corresponding silhouette plot.

```
kmeansSil <- silhouette(Ksol1$cluster, dist(scale(complete)))
silsum <- summary(kmeansSil)
silsum
```

```
## Silhouette of 4546 units in 4 clusters from silhouette.default(x = Ksol1$cluster, dist = dist(scale(
##  Cluster sizes and average silhouette widths:
##       739      1559      1203      1045
## 0.2040882 0.4200223 0.3269803 0.3438813
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.1032  0.2319  0.3824  0.3428  0.4745  0.5978
```

```
plot(kmeansSil)
```



Individual lines in the silhouette plot are not readable with that many observations, but we can still benefit from the output shown on the right.

Report average silhouette width more precisely.

```
silsum$avg.width
```

```
## [1] 0.3427957
```

Compute average silhouette width for all possible values of k.

Thus k ranges from 2 to $n-1$ (where n is the number of observations, here 101) and we report summary statistics for the collection of silhouette widths.

```
set.seed(304)
n <- 101
mydist <- dist(scale(complete))
avgwidths <- rep(0, 99)

for (i in 2:(n-1)) {
  set.seed(304)
  Ksol <- kmeans(scale(complete), centers = i) #centers is the # of clusters
  kmeansSil <- silhouette(Ksol$cluster, mydist)
  avgwidths[i-1] <- summary(kmeansSil)$avg.width
}
```

```
## Warning: did not converge in 10 iterations
```

```
summary(avgwidths)
```

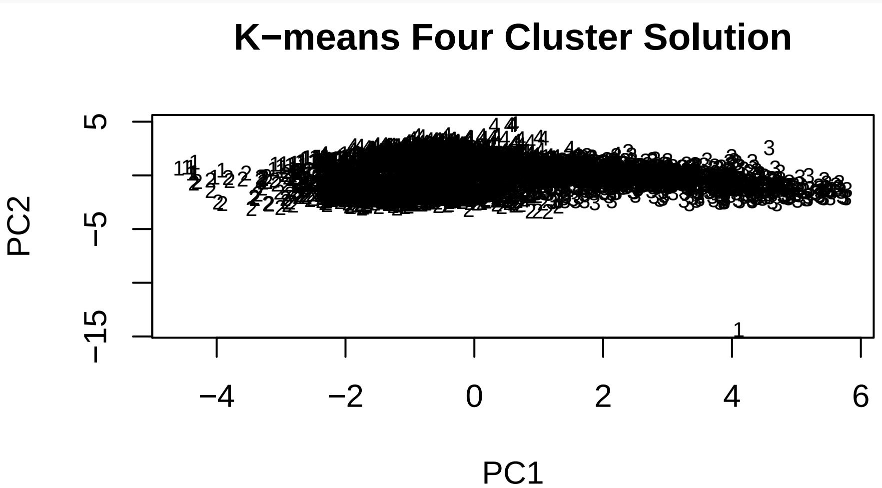
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.1482  0.1522  0.1567  0.1705  0.1688  0.3428
```

Plot the $k = 4$ solution in principal component space.

However, because there are so many observations, we can gain very little benefit from such a plot.

There is, however, one outlier clearly shown at about $(4, -15)$.

```
set.seed(1)
csPCAs <- princomp(complete, cor = TRUE)
plot(csPCAs$scores[, 1:2], type = "n",
     xlab = "PC1", ylab = "PC2", main="K-means Four Cluster Solution")
text(csPCAs$scores[, 1:2], labels = Ksol1$cluster, cex = 0.7)
```




```
summary(csPCAs)
```

```
## Importance of components:
##                           Comp.1    Comp.2    Comp.3     Comp.4     Comp.5
## Standard deviation     1.8277294 1.4513753 1.0950314 0.90480782 0.77922210
## Proportion of Variance 0.3711772 0.2340545 0.1332326 0.09096413 0.06746523
## Cumulative Proportion  0.3711772 0.6052317 0.7384643 0.82942844 0.89689367
```

```
##                           Comp.6     Comp.7     Comp.8      Comp.9
## Standard deviation     0.71988714 0.45321184 0.37141307 0.257625315
## Proportion of Variance 0.05758194 0.02282233 0.01532752 0.007374534
## Cumulative Proportion  0.95447562 0.97729795 0.99262547 1.000000000
```

Examine statistics for some key variables by cluster.

```r
favstats(GRAD_DEBT_MDN ~ Ksol1$cluster, data = complete)
```

```
##   Ksol1$cluster  min      Q1 median      Q3   max     mean       sd    n
## 1             1 4500 21643.5  25084 30344.5 47000 25503.92 6463.374  739
## 2             2 1974  8028.0   9500 11978.0 26250 10172.27 3426.845 1559
## 3             3 5500 21120.5  24500 26000.0 36314 23275.93 4076.457 1203
## 4             4 2069  8750.0  10500 14000.0 32813 11712.32 4765.294 1045
##   missing
## 1       0
## 2       0
## 3       0
## 4       0
```

```r
favstats(TUITFTE ~ Ksol1$cluster, data = complete)
```

```
##   Ksol1$cluster min    Q1 median      Q3    max      mean        sd    n
## 1             1 643 10159  13191 16885.5 712078 15409.465 26919.063  739
## 2             2 415  7828   9710 12587.0  55500 10573.824  4940.852 1559
## 3             3 594  9636  13659 18961.5  49006 15137.613  7656.973 1203
## 4             4 228  1949   2982  4402.0  27488  3550.831  2560.968 1045
##   missing
## 1       0
## 2       0
## 3       0
## 4       0
```

Compare clustering solution to some key variables.

We need a copy of the data with just the columns that make it possible to compare with our clustering solution. Create that data and do the comparison.

```r
for_comparison <- select(data, -STABBR, -MEDIAN_HH_INC, -ACTCMMID, -ADM_RATE, -SAT_AVG,
                     -PFTFAC, -AVGFACSAL)
for_comparison <- for_comparison[complete.cases(for_comparison),]
tally(for_comparison$CONTROL ~ Ksol1$cluster)
```

```
##                         Ksol1$cluster
## for_comparison$CONTROL    1    2    3    4
##                      1   44   98  399  963
##                      2  255  131  773   71
##                      3  440 1330   31   11
```

```r
tally(for_comparison$REGION ~ Ksol1$cluster)
```

```
##                        Ksol1$cluster
## for_comparison$REGION   1   2   3   4
##                     1  25  72 123  46
##                     2  89 287 265 105
##                     3 108 236 216 130
```

```
##                     4  69  71 156 107
##                     5 262 330 244 256
##                     6  87 202  59 144
##                     7  31  57  29  54
##                     8  65 279 110 165
##                     9   3  25   1  38
```

```r
tally(for_comparison$LOCALE ~ Ksol1$cluster)
```

```
##                      Ksol1$cluster
## for_comparison$LOCALE   1   2   3   4
##                    11 233 434 258 149
##                    12 119 187 145  95
##                    13  80 208 165 156
##                    21 201 539 242 162
##                    22  11  47  43  36
##                    23   4  33  32  25
##                    31   7  13  53  21
##                    32  34  33 135  96
##                    33  14  20  73 133
##                    41  22  40  34 127
##                    42   9   4  15  31
##                    43   5   1   8  14
```

To try hierarchical clustering, we need a distance matrix.

```r
cs.dist.scale <- dist(scale(complete))
```

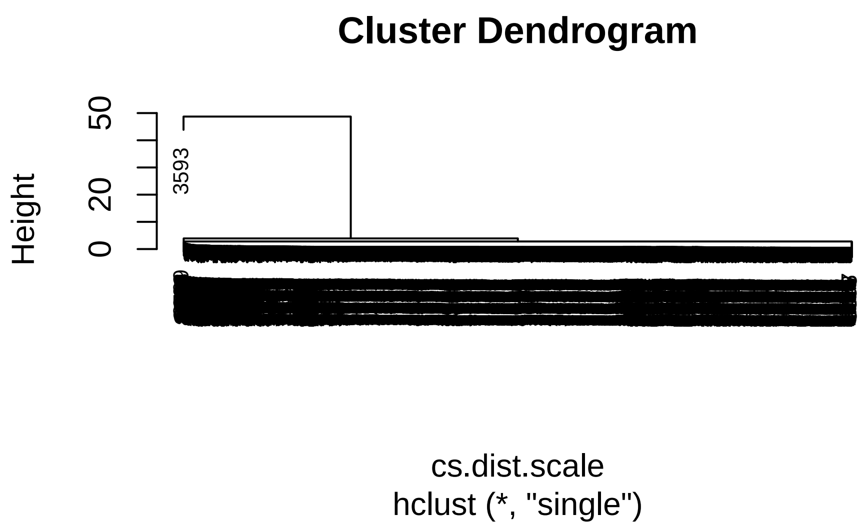
Perform hierarchical clustering with single linkage.

```r
hcsingle <- hclust(cs.dist.scale, method = "single")
list(hcsingle) # reminds you of properties of the solution, if desired
```

```
## [[1]]
##
## Call:
## hclust(d = cs.dist.scale, method = "single")
##
## Cluster method   : single
## Distance         : euclidean
## Number of objects: 4546
```

Plot the corresponding dendrogram.

```r
plot(hcsingle, cex = 0.7)
```



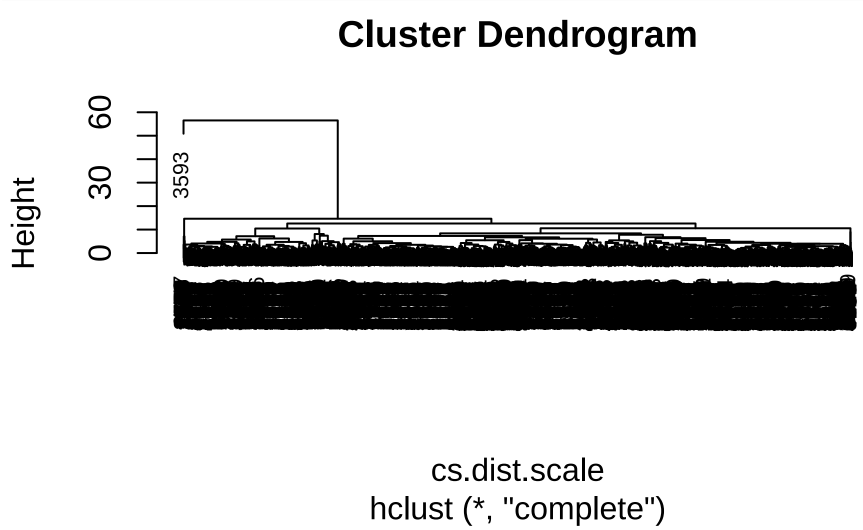
Observation 3593 is a major outlier. Not much else beneficial to be gained here.

```
for_comparison[3593,1:4]
```

```
## # A tibble: 1 x 4
##   INSTNM                              CONTROL REGION LOCALE
##   <chr>                                 <dbl>  <dbl>  <dbl>
## 1 Ultimate Medical Academy-Clearwater       2      5     12
```

Perform hierarchical clustering with complete linkage.

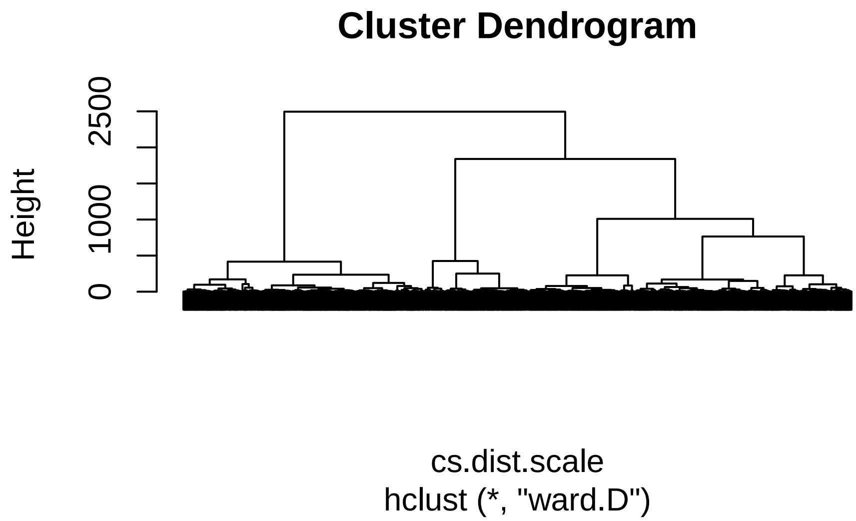
```
hccomp <- hclust(cs.dist.scale, method = "complete")
plot(hccomp, cex = 0.7)
```



Observation 3593 is still a major outlier.

Perform hierarchical clustering with Ward's method.

```
hcward <- hclust(cs.dist.scale, method = "ward.D")
plot(hcward, cex = 0.3, labels = FALSE)
```




```
wardSol <- (cutree(hcward, k= 5))
summary(as.factor(wardSol)) #as factor to get table
```

```
##    1    2    3    4    5
##  939  543  713  715 1636
```

Compute and report average silhouette width more precisely.

```
WardSil <- silhouette(wardSol, dist(scale(complete)))
silsum2 <- summary(WardSil)
silsum2
```

```
## Silhouette of 4546 units in 5 clusters from silhouette.default(x = wardSol, dist = dist(scale(comple
##  Cluster sizes and average silhouette widths:
##       939       543       713       715      1636
## 0.1721134 0.2105865 0.2185532 0.3728725 0.2989987
## Individual silhouette widths:
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -0.4333  0.1617  0.3130  0.2612  0.4023  0.5234
```

```
silsum2$avg.width
```

```
## [1] 0.2612312
```

Compare clustering solution to some key variables.

(The `for_comparison` data frame was defined earlier.)

```
tally(for_comparison$CONTROL ~ wardSol)
```

```
##                         wardSol
## for_comparison$CONTROL    1    2    3    4    5
##                       1  438    9  241  702  114
##                       2  427  141  468    7  187
##                       3   74  393    4    6 1335
```

```
tally(for_comparison$REGION ~ wardSol)
```

```
##                        wardSol
## for_comparison$REGION    1    2    3    4    5
```

```
##                      1  50  18  88  36  74
##                      2 132  59 177  79 299
##                      3 135  86 125  99 245
##                      4 140  62  77  49  75
##                      5 275 156 121 182 358
##                      6  93  57  33 103 206
##                      7  44  29  15  27  56
##                      8  67  75  77 140 260
##                      9   3   1   0   0  63
```

```
tally(for_comparison$LOCALE ~ wardSol)
```

```
##                       wardSol
## for_comparison$LOCALE   1   2   3   4   5
##                      11 179 186 151 104 454
##                      12  95  79  91  67 214
##                      13 128  56 108 100 217
##                      21 150 168 146 124 556
##                      22  22   9  29  31  46
##                      23  17   3  24  16  34
##                      31  36   5  25  13  15
##                      32 108  10  84  65  31
##                      33 117   7  28  64  24
##                      41  51  15  16 100  41
##                      42  22   3   8  23   3
##                      43  14   2   3   8   1
```

Compare clustering solutions built using k-means and Ward's method.

```
tally(wardSol ~ Ksol1$cluster)
```

```
##        Ksol1$cluster
## wardSol    1    2    3    4
##       1  206   34  425  274
##       2  464   63   12    4
##       3    0    1  694   18
##       4    6    0    0  709
##       5   63 1461   72   40
```