# Lab 4 - Machine Learning for Social Science (Solutions)

```r
knitr::opts_chunk$set(echo = TRUE)
mywd <- "/Users/marar08/Documents/Teaching/MLSS_HT2025/"
setwd(mywd)
library(data.table)
library(ggplot2)
library(mclust)
```

```
## Package 'mclust' version 6.1.1
## Type 'citation("mclust")' for citing this R package in publications.
```

```r
library(elasticnet)
```

```
## Loading required package: lars
```

```
## Loaded lars 1.3
```

## Part 1: Taste clustering and influence

In the first part of this lab, we will consider a (simulated) data set which contains information about a sample of (fictive) individuals' music tastes as well as a measure of their influence on others.

1. Begin by importing the file "`taste_influence.csv`". Report the number of rows and columns of the data set, and the genres contained in it. Create a scatter-plot of two combinations of genres of your choice. Based on this, do you get any indication that the data is clustered along musical tastes?

```r
dt <- fread('/Users/marar08/Documents/Teaching/MLSS_HT2025/Labs/W4/taste_influence.csv')
dim(dt)
```

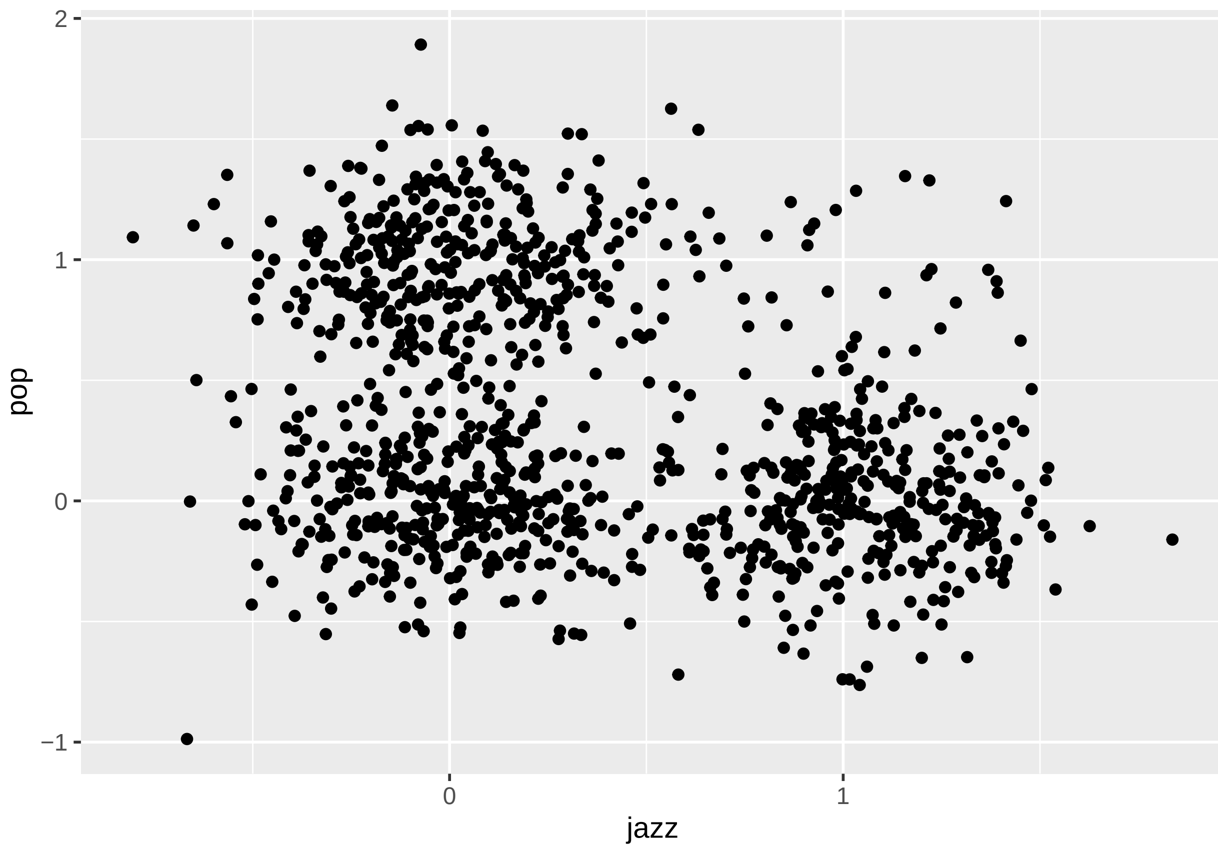
```
## [1] 1075    4
```

```r
head(dt)
```

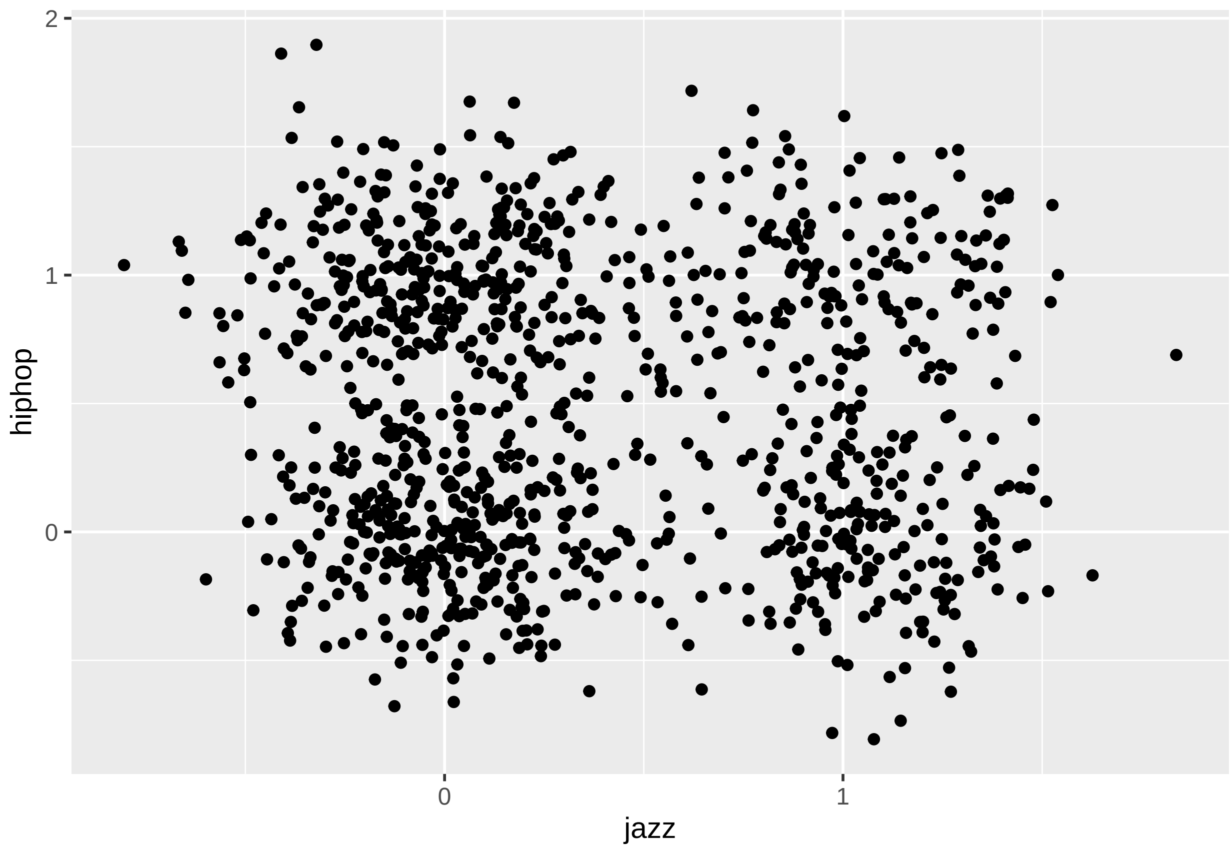
```
##            jazz         pop      hiphop   influence
##           <num>       <num>       <num>       <num>
## 1: -0.03743187 -0.02836584  1.17409185  1.67634621
## 2:  0.55748147  0.15872313 -0.03007249  0.07487850
## 3: -0.08302068 -0.02067557 -0.11942695 -0.09607821
## 4:  0.11750772  1.39635593 -0.06703518  0.36594376
## 5:  0.24060609  1.01726019  0.66100663  1.81186663
## 6:  0.93673898  0.04925761 -0.05323063  1.77677839
```

The data sets contains 1075 rows and 4 columns. Three of the variables/columns describe the users taste (genres: jazz, pop, and hiphop), and one their influence on others.

```r
ggplot(dt, aes(x=jazz,y=pop)) + geom_point()
```




```r
ggplot(dt, aes(x=jazz,y=hiphop)) + geom_point()
```




Based on these bivariate scatterplots, I would say there are indeed some indications of clustering along the taste dimensions. Although the separation is *not super-clear cut*, there are clear *differences in density* of data points in different regions. Taking jazz–hiphop as an example, I would say there looks like there are four dense regions of data points with centers at [0,0], [0,1], [1,0], and [1,1], and where there is a decline in desity at their respective borders.

2. Now you shall do some clustering. To prepare the data, do the following: (*i*) store/copy the data to a new `R` object, and subset it so that it only contains the three "taste columns"— these are the columns you will cluster based upon, (*ii*) standardize this data table (hint: you can e.g., use `scale()` for this purpose), (*iii*) transform it into a matrix (hint: e.g., by using `as.matrix()`).

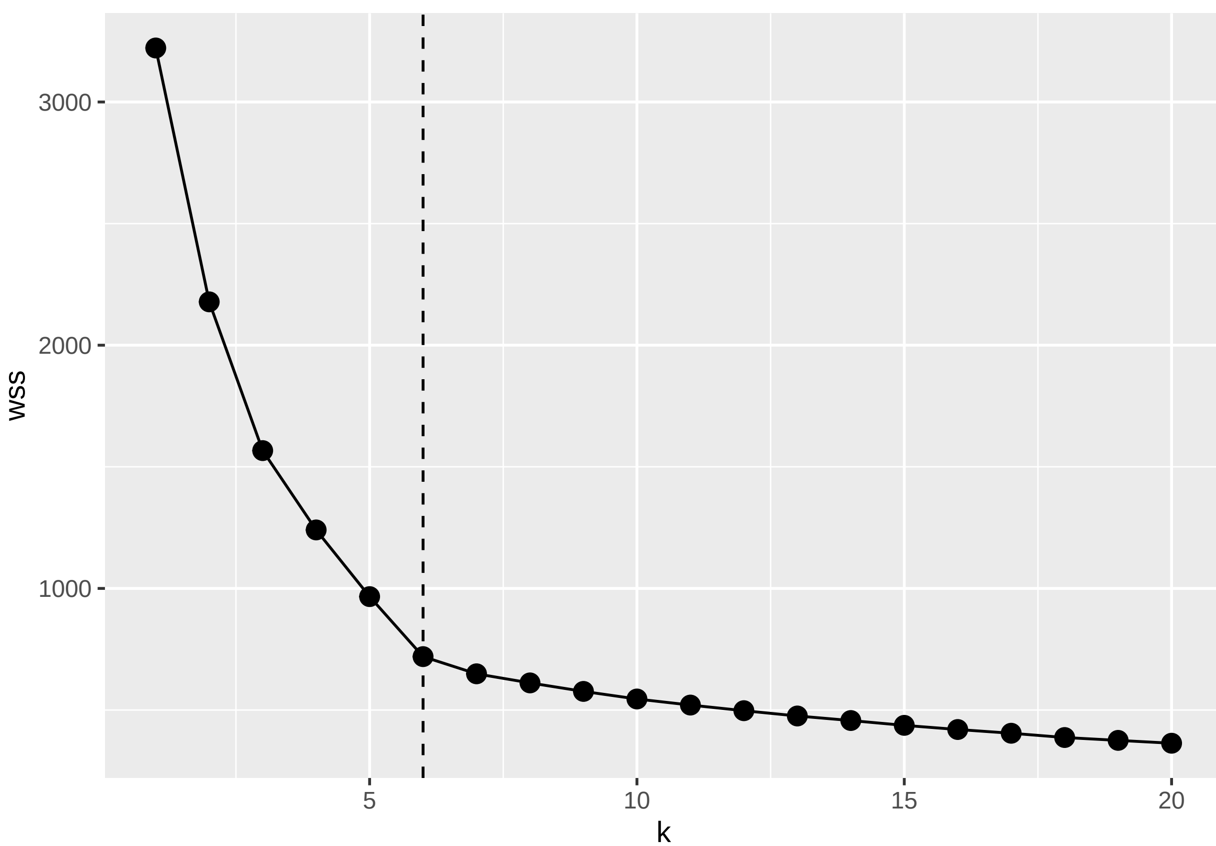
```r
# (i)
taste_mat <- copy(dt)
taste_mat$influence <- NULL
# (ii)
taste_mat <- as.matrix(taste_mat)
# (iii)
taste_mat <- scale(taste_mat)
# Inspect
head(taste_mat)
```

```
##            jazz        pop     hiphop
## [1,] -0.7253625 -0.6712253  1.2209825
## [2,]  0.3576840 -0.3239461 -0.9018071
## [3,] -0.8083574 -0.6569504 -1.0593277
## [4,] -0.4432932  1.9733787 -0.9669676
```

```
## [5,] -0.2191912  1.2696917  0.3164781
## [6,]  1.0481267 -0.5271387 -0.9426319
```

3. Having formatted the data according to #2, you shall now use the *kmeans* algorithm to cluster your data. Recall that a requisite for running *kmeans* is that the parameter $k$ has been specified. In practice—and as is the case here—we often do not know the appropriate number of clusters a priori. Therefore, you shall implement a loop that, at every iteration, runs kmeans with a different number of clusters, and extracts the *total within cluster sum of squares* (hint 1: which can be extracted using `$tot.withinss` | hint 2: set the argument `nstart=100` to ensure robustness of the local optima you find). Consider no. clusters ranging from 1 to 20, with an interval of 1. Plot $k$ against `tot.withinss`. Which number of clusters do you find appropriate? Motivate.

```r
ks <- 1:20
wss <- c()
for(i in 1:length(ks)){
  temp <- kmeans(x = taste_mat,
                 centers = ks[i],
                 nstart = 100,
                 iter.max = 1000)
  wss[i] <- sum(temp$tot.withinss)
}
ggplot(data.table(k=ks,wss=wss),aes(x=k,y=wss)) +
  geom_point(size=3) +
  geom_line() +
  geom_vline(xintercept = 6, linetype='dashed')
```




The relative gain in terms of additional *total within sum of squares* per extra cluster starts to decline sharply after $k = 6$. Hence, seeking to find a good balance between variance accounted for and parsimony/complexity, $k = 6$ appears to be a good choice here.

4. For the specification (of $k$) that you decided on in #3, extract the *centroids* and interpret each cluster in terms of what distinguishes it from the rest. Do the clusters seem meaningfully distinct?

```
set.seed(1234)
finalk <- kmeans(x = taste_mat,
                 centers = 6,
                 nstart = 100,
                 iter.max = 1000)
finalk$centers
```

```
##         jazz        pop     hiphop
## 1 -0.6606565 -0.6264451 -0.8310363
## 2  1.1975400 -0.5153682  0.9374598
## 3 -0.6034402  1.2157422 -0.8781105
## 4 -0.5225678  1.2650179  0.8785221
## 5  1.2617093 -0.6363058 -0.8861036
## 6 -0.7075403 -0.6519389  0.9275712
```

The clusters does indeed seem to capture meaningfully distinct taste-profiles. I would label each cluster as follows (note that scale is a bit funny because we have standardized our data; e.g., $+1$ = one positive standard deviation from the mean, which is 0):

- Cluster 1: People who do not like either jazz, pop or hiphop
- Cluster 2: Jazz and hiphop fans
- Cluster 3: Pop fans
- Cluster 4: Pop and hiphop fans
- Cluster 5: Jazz fans
- Cluster 6: Hiphop fans.

5. To get a feeling for the role that the choice of $k$ plays, estimate another *kmeans* model but this time with $k = 2$. Inspecting the centroids, how does your clustering change; how does it alter your understanding of the population?

```
set.seed(1234)
k2 <- kmeans(x = taste_mat,
             centers = 2,
             nstart = 100,
             iter.max = 1000)
k2$centers
```

```
##         jazz        pop      hiphop
## 1  1.1107205 -0.6203654 -0.08308994
## 2 -0.6634616  0.3705600  0.04963173
```

Partitioned this way, it would appear we have jazz fans—who are OK with hiphop but does not like pop, on the one hand (cluster 1). And then moderate pop fans, who are OK with hophop but does not like jazz. In other words, this gives a very different picture of the population. By being so coarse, it blends together different tastes, and we only see pooled averages.

6. Clustering provides a tool for discovering underlying structures in our data. Once these structures have been discovered, they can be studied in separate analyses. That is what you shall do now. We want to examine whether different "taste types" have differential degree of influence on others. To do so, (*i*) create a new column in your original data set storing the the retrieved cluster assignments (hint: you find the cluster assignments using `$cluster`). Then (*ii*) estimate a linear regression with the *influence score* (`infuence`) as the outcome variable, and the clusters (formatted as a factor) as predictors. Interpret the results: are there any difference in influence between the clusters?

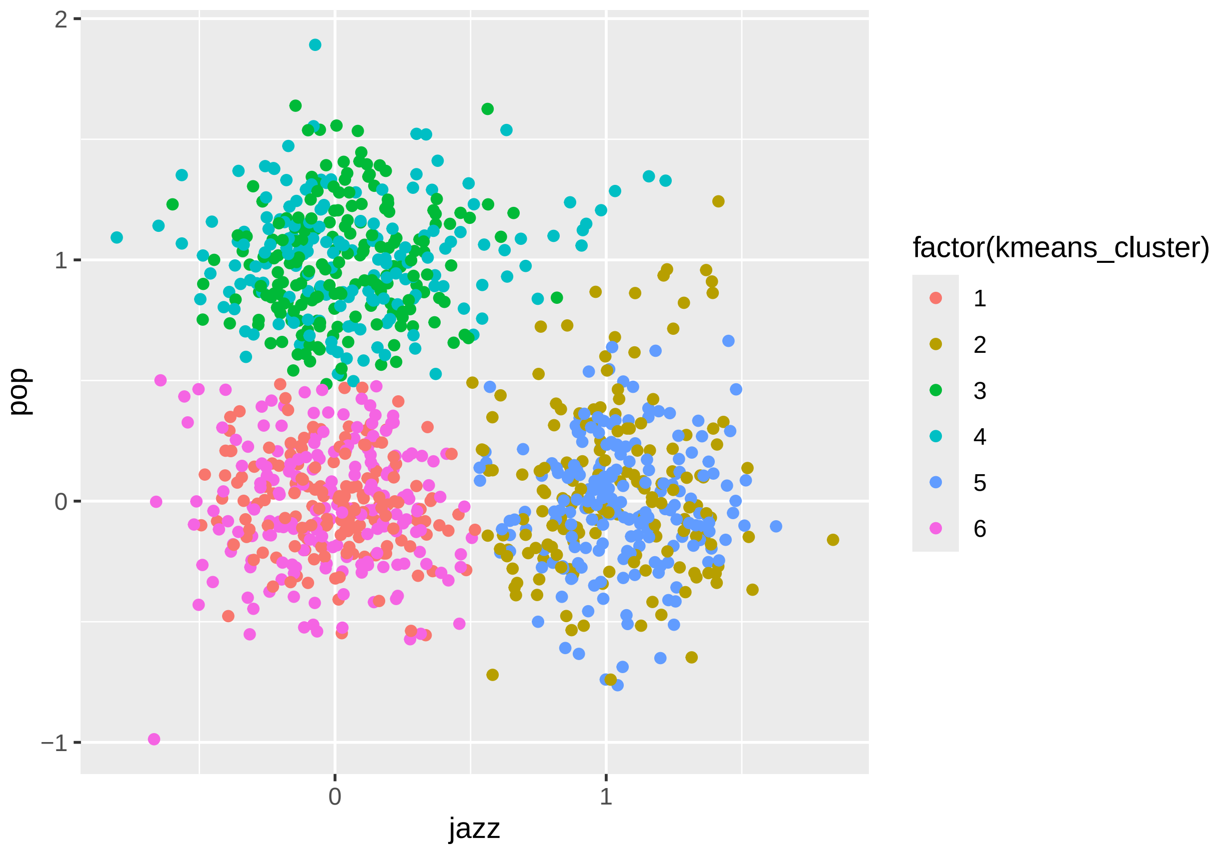
```
dt[,kmeans_cluster := factor(finalk$cluster)]
summary(lm(influence~kmeans_cluster,data=dt))
```

```
##
## Call:
## lm(formula = influence ~ kmeans_cluster, data = dt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5574 -0.5409  0.0660  0.6140  3.4105
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)      0.19778    0.07516   2.631  0.00863 **
## kmeans_cluster2  2.42014    0.10662  22.698  < 2e-16 ***
## kmeans_cluster3  0.52539    0.10300   5.101 3.99e-07 ***
## kmeans_cluster4  1.79207    0.10598  16.910  < 2e-16 ***
## kmeans_cluster5  1.57779    0.10166  15.521  < 2e-16 ***
## kmeans_cluster6  1.60424    0.10237  15.671  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9655 on 1069 degrees of freedom
## Multiple R-squared:  0.4019, Adjusted R-squared:  0.3991
## F-statistic: 143.6 on 5 and 1069 DF,  p-value: < 2.2e-16
```

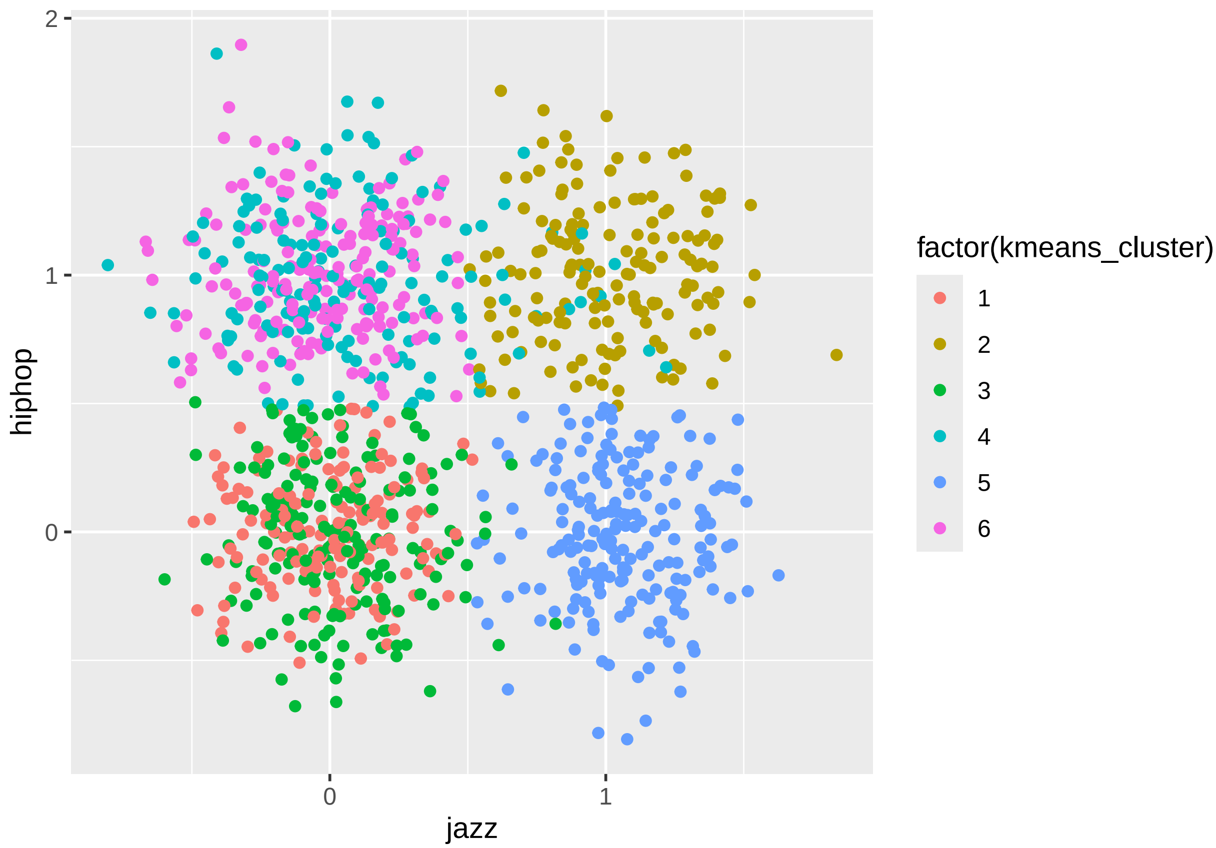
There does indeed seem to be differences influence across the 6 clusters. The most influential is cluster 2, which we labeled as those who like both jazz and hiphop. The least influential is the base category, cluster 1, which reflects individuals who neither likes jazz, hiphop or pop. One might also note that differences in taste explain as much as 40% of the variation in infliuence.

7. Now that you have merged the cluster assignments to the original data, produce the same plots as you did in #1, but now colored by the cluster assignments. Does it look like *kmeans* have picked up on the patterns you observed in #1? Further—what you think of the separation between the clusters? Is there clear spacing betweeen the clusters, or are the borders almost touching each other (note that there will be certain overlap due to plotting the data in 2D)?

```
ggplot(dt,aes(x=jazz,pop,color=factor(kmeans_cluster))) + geom_point()
```



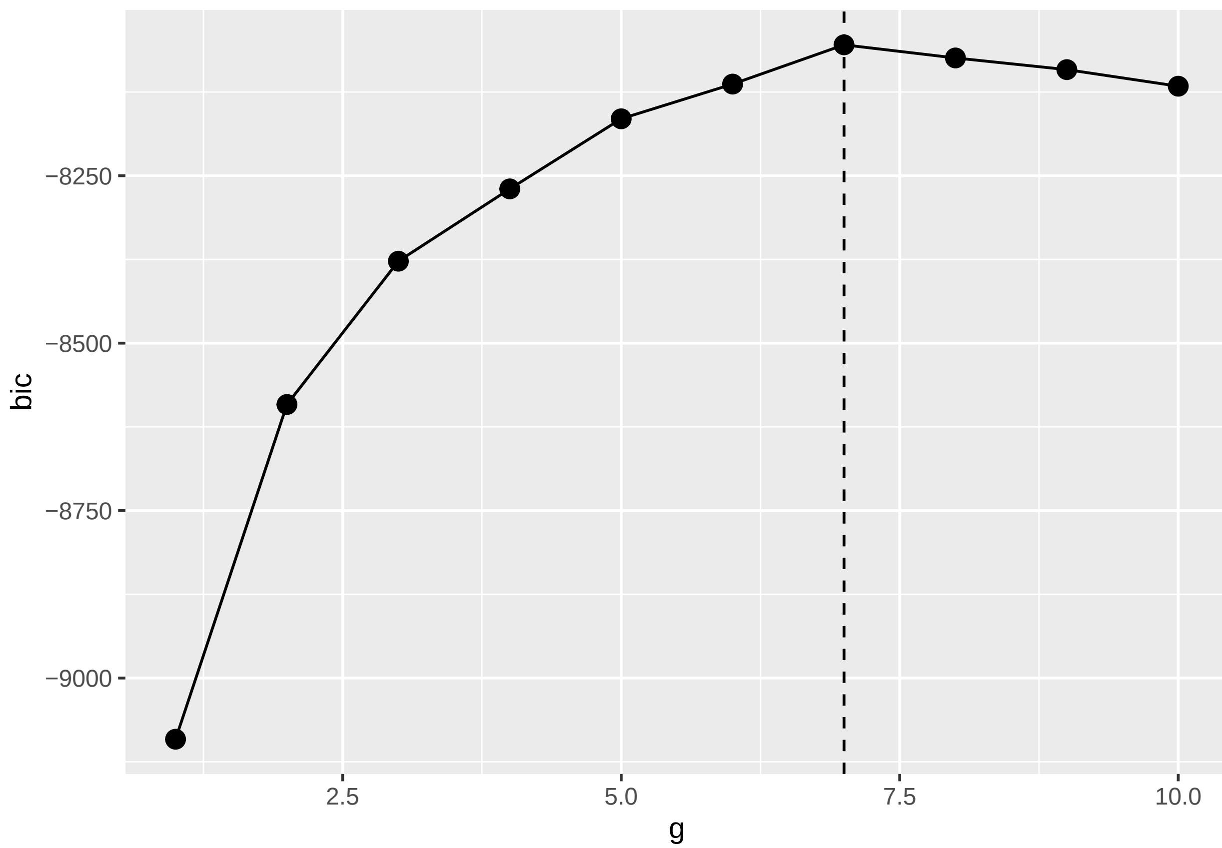

```r
ggplot(dt,aes(x=jazz,hiphop,color=factor(kmeans_cluster))) + geom_point()
```




*kmeans* have indeed picked up on the density differences spotted earlier. Note: the reason for the two-colors-per-cluster pattern is that clusters—as we know from the labeling—are defined by three dimensions, not two. Had we applied PCA here, and plotted the clusters, we would see a much cleaner separation.

8. Repeat step 3–7 (but skip #5) using now instead a *Gaussian mixture model*. For this, you may use `mclust`'s function `Mclust()` (specifying the number of components with the argument `G`). For #3: Note that, because this is a probabilistic model, we retrieve a *likelihood score* (or, more specifically *BIC* which is based upon the likehliood score but also penalizes for complexity) to measure its performance instead of *total within cluster sum of squares* (hint: you can extract the BIC by `$bic` on the model object). For #4, you can use `$parameters$mean` to extract the means/centroids of each cluster. For #6, you shall extract the hard cluster assignments (which you can do using `$classification`).

```
# Step #3: deciding k (or G)
lls <- bics <- c()
ks <- 1:10
for(i in 1:10){
  temp <- mclust::Mclust(data = taste_mat,
                         G = ks[i])
  lls[i] <- temp$loglik
  bics[i] <- temp$bic
}
ggplot(data.table(g=ks,bic=bics),aes(x=g,y=bic)) +
  geom_point(size=3) +
  geom_line() +
  geom_vline(xintercept = 7,linetype='dashed')
```




Using BIC as the criterion to select $k$, we see that it is maximized at $k = 7$, after which the penality for added clusters exeeds the gain in fit of the data.

```
# Refit for best k/G=7
set.seed(1)
gmm <- mclust::Mclust(data = taste_mat, G = 7)

# Step #4: extract means for interpretation
t(gmm$parameters$mean)
```

```
##            jazz        pop     hiphop
## [1,] -0.6881655 -0.6460921  0.9196229
## [2,] -0.6493541 -0.6091402 -0.8299227
## [3,] -0.5909529  1.2169070 -0.8628565
## [4,] -0.6951585  1.2151297  0.8672693
## [5,]  1.2606871 -0.6421940 -0.8762022
## [6,]  1.1720341 -0.6766994  0.9056929
## [7,]  1.1080265  1.1541214  0.9649712
```

These clusters also seem capture meaningfully distinct taste-profiles. The difference here is that we have one more additional cluster. I would label each cluster as follows:

- Cluster 1: Hiphop fans
- Cluster 2: People who do not like either jazz, pop or hiphop

- Cluster 3: Pop fans
- Cluster 4: Pop and hiphop fans
- Cluster 5: Jazz fans
- Cluster 6: Jazz and hiphop fans
- Cluster 7: People who like all three genres (new)

```r
# Step #6:
gmm_z <- as.data.table(gmm$z)
setnames(gmm_z,
         old = paste0('V',1:7),
         new = paste0('C',1:7))
dt[,kmeans_cluster := NULL]
dt <- cbind(dt,gmm_z)
dt[,gmm_sharp_cluster_assignment := factor(gmm$classification)]
summary(lm(influence ~ gmm_sharp_cluster_assignment,data=dt))
```

```
##
## Call:
## lm(formula = influence ~ gmm_sharp_cluster_assignment, data = dt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.5475 -0.5226  0.0730  0.6061  2.5033
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    1.79642    0.06836  26.280  < 2e-16 ***
## gmm_sharp_cluster_assignment2 -1.59863    0.10069 -15.877  < 2e-16 ***
## gmm_sharp_cluster_assignment3 -1.08408    0.09705 -11.170  < 2e-16 ***
## gmm_sharp_cluster_assignment4  0.13308    0.10318   1.290    0.197
## gmm_sharp_cluster_assignment5 -0.03081    0.09582  -0.322    0.748
## gmm_sharp_cluster_assignment6  0.78398    0.10437   7.512 1.23e-13 ***
## gmm_sharp_cluster_assignment7  1.28185    0.18375   6.976 5.32e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9497 on 1068 degrees of freedom
## Multiple R-squared:  0.4219, Adjusted R-squared:  0.4186
## F-statistic: 129.9 on 6 and 1068 DF,  p-value: < 2.2e-16
```

Considering the resulting coefficients, let me just make a few remarks. First, and most interestingly, we see that the "new" cluster—the one capturing individuals who like all of the three genres—is the most influential cluster. The second most influential are the hiphop and jazz fans. The least influential, as we also found with kmeans, are the people who don't like any of the three genres.

9. Something which `Mclust()` also provides is a score for each observation how *uncertain* we are about its assignment. As mentioned during the lecture, "border-observations" can sometimes be substantively meaningful to study. You shall do so here. Extract the vector `$uncertainity` from the Gaussian mixture model fit, and store it in the original data. Then yet again fit a linear regression (together with the taste variables), but this time additionally with the `uncertainity` variable.

```
dt[,uncertainity := gmm$uncertainty]
summary(lm(influence ~ gmm_sharp_cluster_assignment + uncertainity,data=dt))
```

```
##
## Call:
## lm(formula = influence ~ gmm_sharp_cluster_assignment + uncertainity,
##     data = dt)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.8151 -0.5115 -0.0157  0.5060  3.3028
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                    2.06168    0.05860  35.182  < 2e-16 ***
## gmm_sharp_cluster_assignment2 -1.61357    0.08435 -19.129  < 2e-16 ***
## gmm_sharp_cluster_assignment3 -1.07142    0.08131 -13.178  < 2e-16 ***
## gmm_sharp_cluster_assignment4  0.24291    0.08659   2.805  0.00512 **
## gmm_sharp_cluster_assignment5 -0.09050    0.08032  -1.127  0.26010
## gmm_sharp_cluster_assignment6  0.81508    0.08744   9.321  < 2e-16 ***
## gmm_sharp_cluster_assignment7  1.43922    0.15411   9.339  < 2e-16 ***
## uncertainity                  -4.42904    0.20767 -21.327  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7956 on 1067 degrees of freedom
## Multiple R-squared:  0.5947, Adjusted R-squared:  0.592
## F-statistic: 223.6 on 7 and 1067 DF,  p-value: < 2.2e-16
```

A first remark: including the uncertainity variable, we see a substantial bump in the explained $R^2$. What is its relation to influence? We see a large negative coefficient; suggesting that—in this fictive population—the more atypical an individual's taste, the less influential he/she is. This not a totally crazy idea based on what studies on homophily have found.

### Part 2: Regional variation

In the second part of the lab, we will consider another simulated data set. This time, containing information about both the (fictive) individuals themselves, but also their social environments.

1. Begin by importing the file "`neighborhood.csv`". Report the number of rows and columns of the data set, and make a brief note on the types of columns contained in it.

```
nh <- fread(input = 'neighborhood.csv')
dim(nh)
```

```
## [1] 200  25
```

```
colnames(nh)
```

```
##  [1] "taste_jazz"               "taste_classical"
```

```
##  [3] "taste_blues"             "taste_pop"
##  [5] "taste_country"           "taste_raegge"
##  [7] "income"                  "nbhood_avg_income"
##  [9] "education"               "nbhood_avg_education"
## [11] "nhood_crime"             "nbhood_unemployment"
## [13] "nhbood_avg_temp"         "nhbood_pop"
## [15] "nhbood_nr_lights"        "nhbood_nr_pizzerias"
## [17] "city_avg_taste_jazz"     "city_avg_taste_classical"
## [19] "city_avg_taste_blues"    "city_avg_taste_pop"
## [21] "city_avg_taste_country"  "city_avg_taste_raegge"
## [23] "taste_film_action"       "taste_film_romcom"
## [25] "taste_film_documentary"
```

The data set contains 200 rows and 25 columns. These variables are made up of both *individual level* variables (music taste, film taste, income, education), *neighborhood level* variables (music taste, crime, education, unemployment, no. pizzerias, no. lights, average temperature), and *city level* variables (music taste).

2. Based on the types of variables we find, we have some suspicion that there may exist considerable correlation between different variables in this data set. To explore whether we can capture key aspects of our data using fewer dimensions, we will use *PCA* and its extensions. Begin by estimating a *principal components* model *without* doing any standardization (hint: to estimate a PCA, use `prcomp()`). Why is this problematic (hint: examine the principal loadings)

```
nh_mat <- as.matrix(nh)
nh_pca <- prcomp(nh_mat)
summary_nh_pca <- summary(nh_pca)
nh_pca$rotation[,c(1:3)]
```

```
##                                   PC1           PC2          PC3
## taste_jazz               -1.130734e-02 -0.0023440600 -0.014436501
## taste_classical          -7.785547e-03  0.0075890578 -0.011243550
## taste_blues              -1.094852e-02  0.0036306694 -0.017388315
## taste_pop                -7.826432e-05 -0.0116550537  0.028459962
## taste_country             1.995948e-03 -0.0017553217  0.024188748
## taste_raegge              4.944613e-03 -0.0147713147  0.025235482
## income                    5.135923e-03  0.0159201632 -0.023204570
## nbhood_avg_income         6.926408e-03  0.0164944063 -0.018198678
## education                -2.588437e-03  0.0090498047 -0.013038645
## nbhood_avg_education     -5.864163e-03  0.0133028161 -0.010250448
## nhood_crime               4.772406e-03 -0.0105344496  0.015323486
## nbhood_unemployment       4.138436e-03 -0.0096372982  0.013718324
## nhbood_avg_temp           4.802633e-02  0.5671282890  0.807538197
## nhbood_pop               -2.325264e-01  0.4612561449 -0.155298040
## nhbood_nr_lights         -9.704813e-01 -0.1031315060  0.094911157
## nhbood_nr_pizzerias      -2.950091e-02  0.6727586139 -0.555567040
## city_avg_taste_jazz      -5.433612e-03 -0.0054625216  0.019601534
## city_avg_taste_classical  7.555348e-03 -0.0079776471  0.008232628
## city_avg_taste_blues      3.312960e-03 -0.0006816765  0.026113769
## city_avg_taste_pop       -6.397509e-03  0.0016809645 -0.017835786
## city_avg_taste_country    3.592366e-03  0.0052711518 -0.005572193
## city_avg_taste_raegge     1.429978e-03  0.0080628332 -0.013574051
## taste_film_action         1.278440e-02 -0.0064266489  0.006172812
## taste_film_romcom        -8.944392e-03 -0.0154741603  0.003254971
## taste_film_documentary   -5.620100e-03  0.0218192137 -0.012201239
```

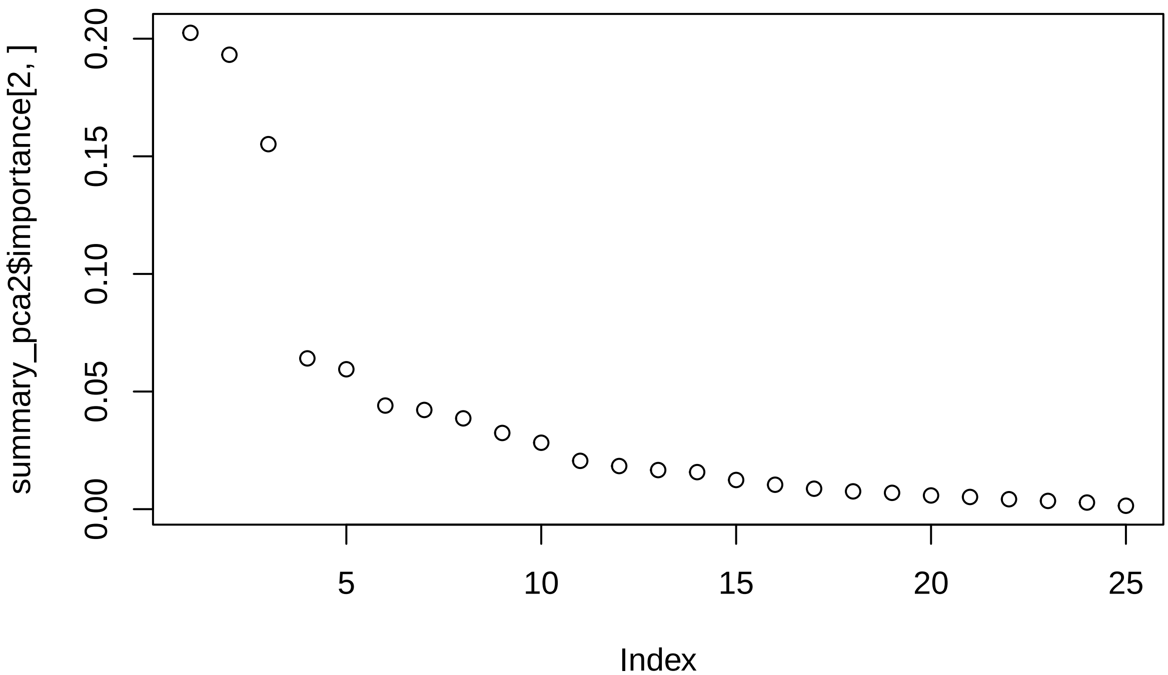
```r
colMeans(nh)
```

```
##             taste_jazz          taste_classical              taste_blues
##            -0.07608428              -0.05200800              -0.05217688
##              taste_pop            taste_country             taste_raegge
##             0.08899830               0.04512450               0.09513898
##                 income         nbhood_avg_income                education
##            -0.03255688              -0.05084456               0.02212464
##   nbhood_avg_education              nhood_crime      nbhood_unemployment
##             0.02774459               0.01057689               0.01665329
##        nhbood_avg_temp               nhbood_pop         nhbood_nr_lights
##             9.66232846               9.97013680               9.32930979
##     nhbood_nr_pizzerias      city_avg_taste_jazz city_avg_taste_classical
##             9.54769551              10.00736259               9.96904671
##    city_avg_taste_blues       city_avg_taste_pop   city_avg_taste_country
##             9.95416957              10.09838065              10.03703774
##   city_avg_taste_raegge        taste_film_action        taste_film_romcom
##            10.02680726              -0.07699345               0.09774521
## taste_film_documentary
##            -0.02571752
```

The principal loadings reveal that a set of the neighborhood dimensions (nbhoodpop, nhboodnrlights, nhboodnrpizzerias, nbhoodavgtemp) dominate the first PCs—the ones which explain by far the most variance. As we see from the means calculation, however, this is an artefact of these columns having a much greater scale.

3. Now, standardize your data, and then fit a PCA on this standardized data set. Plot the *proportion variance explained*. Interpret and decide on an appropriate number of principal components.

```r
mat_scaled <- scale(as.matrix(nh))
nh_pca2 <- prcomp(mat_scaled)
summary_pca2 <- summary(nh_pca2)
plot(summary_pca2$importance[2,])
```




The first two PC explains ≈ 20% each. Based only on this plot, I would go with either three or five *PCs*. After this point, the additional PCs explain very little additional variance.

4. Interpret the retrieved principal components based on their loadings. Do they provide easy and substantively expected interpretations?

```
nh_pca2$rotation[,c(1:5)]
```

```
##                               PC1          PC2          PC3          PC4
## taste_jazz             -0.165598284 -0.315796907  0.188815999  0.008645344
## taste_classical        -0.186439330 -0.282337949  0.220597843  0.034722749
## taste_blues            -0.197560893 -0.315298024  0.166961965 -0.021606793
## taste_pop               0.218843741  0.270293966 -0.188021321 -0.046135465
## taste_country           0.205061458  0.277200903 -0.206469793  0.021552088
## taste_raegge            0.205283637  0.279349334 -0.207769006  0.016002989
## income                 -0.334078396  0.119288960 -0.200122814  0.036660873
## nbhood_avg_income      -0.340921831  0.100533268 -0.182486414  0.070252855
## education              -0.341538161  0.082650349 -0.207921612  0.023059624
## nbhood_avg_education   -0.349351488  0.090315358 -0.180684902  0.005989955
## nhood_crime             0.329488500 -0.103288082  0.217348454  0.014278135
## nbhood_unemployment     0.346934465 -0.113247576  0.191109908 -0.006426588
## nhbood_avg_temp         0.033748193  0.010148644 -0.045986155 -0.050748419
## nhbood_pop             -0.036709545 -0.046398844 -0.014724121 -0.211302634
## nhbood_nr_lights       -0.006280977 -0.004203786  0.018981095 -0.189007367
## nhbood_nr_pizzerias    -0.110745654  0.026056839  0.061859046 -0.186292300
```

```
## city_avg_taste_jazz       0.095354373 -0.243906113 -0.308085582 -0.023559477
## city_avg_taste_classical  0.067175435 -0.280305639 -0.289200006  0.047647626
## city_avg_taste_blues      0.097818272 -0.278059681 -0.271576796  0.001225083
## city_avg_taste_pop       -0.076748148  0.263499641  0.319545930  0.060475896
## city_avg_taste_country   -0.091673374  0.247513332  0.295280303  0.009527403
## city_avg_taste_raegge    -0.112170433  0.248201800  0.290919907  0.084001215
## taste_film_action         0.020026975 -0.063442131  0.002728314  0.711698717
## taste_film_romcom         0.049294821  0.113532618 -0.006786904 -0.227452260
## taste_film_documentary   -0.059787432 -0.053353936  0.029209916 -0.546605735
##                                    PC5
## taste_jazz               -4.807130e-02
## taste_classical          -6.322271e-02
## taste_blues              -8.290845e-02
## taste_pop                 4.656127e-02
## taste_country             5.467145e-02
## taste_raegge              5.140658e-02
## income                   -3.582694e-05
## nbhood_avg_income         6.787990e-03
## education                -2.630559e-02
## nbhood_avg_education     -1.014768e-02
## nhood_crime               4.273240e-02
## nbhood_unemployment       1.377604e-02
## nhbood_avg_temp           2.151341e-01
## nhbood_pop                3.700403e-01
## nhbood_nr_lights         -1.519667e-01
## nhbood_nr_pizzerias       2.468936e-01
## city_avg_taste_jazz      -4.694237e-02
## city_avg_taste_classical -3.406103e-02
## city_avg_taste_blues     -1.690488e-02
## city_avg_taste_pop       -5.199699e-03
## city_avg_taste_country    5.539677e-03
## city_avg_taste_raegge     2.086289e-02
## taste_film_action         2.548018e-01
## taste_film_romcom        -6.802694e-01
## taste_film_documentary    4.207922e-01
```

A high value on PC1 indicates (*a*) that the focal individual likes pop, country or raegge (and dislikes blues, classical and jazz), (*b*) a low income and education, both for the individual in question and their neighborhood, (*c*) high crime and unemployment in their neighborhood, (*d*) to a lesser but not negligible: city liking jazz, classical and blue (but disliking pop, country, reggae). Thus, as we can see, the loadings combine multiple types of dimensions, and this makes interpretation a bit tricky.

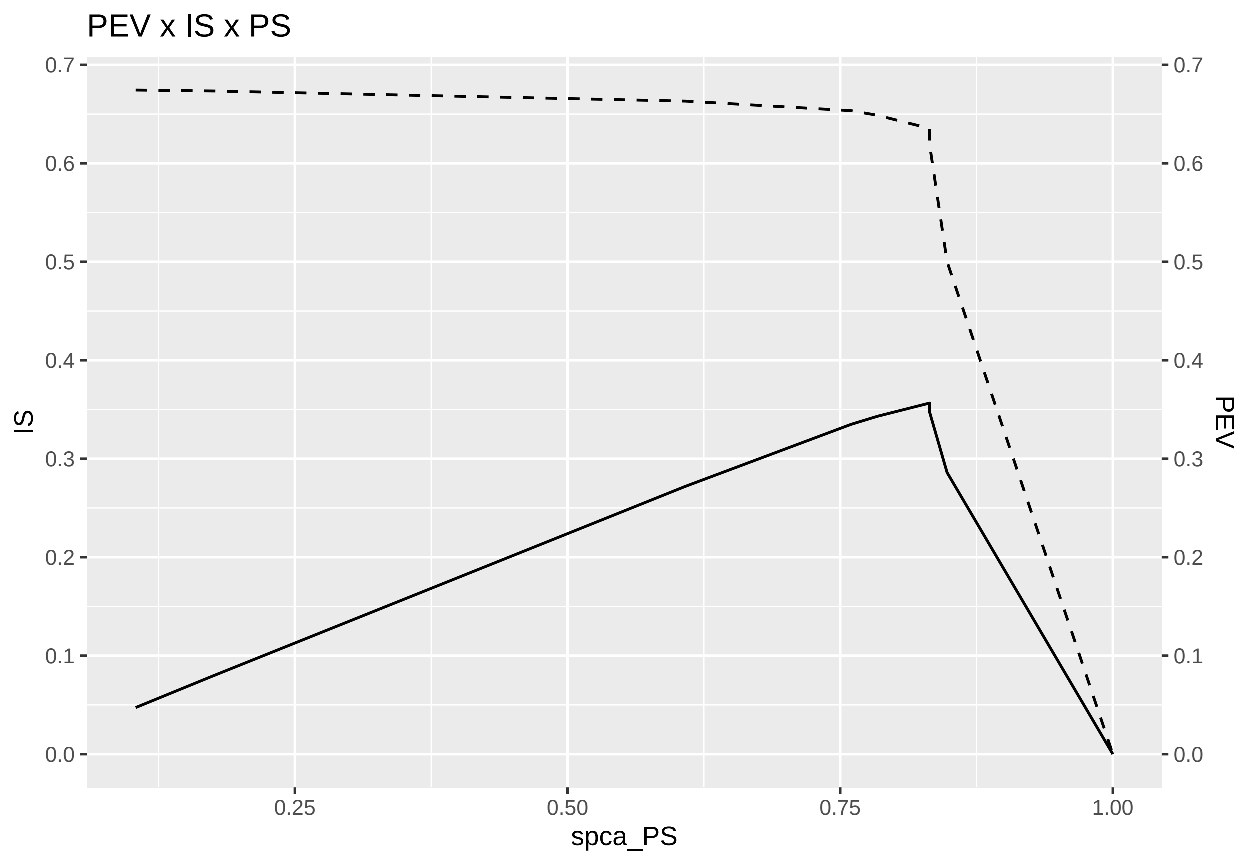
5. Because of the conclusions in #4, we will now consider the *sparse PCA*. Use the same number of principal components that you did for the standard PCA in #2. In comparison to the standard PCA, we have an additional parameter $\lambda$ in the sparse PCA. Use the *IS* index to determine an appropriate $\lambda$. Inspect the principal loadings for the resulting configuration. Interpret each dimension. Which do you think was easier to interpret; the sparse PCA or the standard PCA? Are there any downsides to sparse PCA?

```r
# Sparse PCA comparison
penalities <- c(0,1,5,10,20,80,100,200,400,800)
spca_pevs <- c()
spca_ps <- c()
```

```r
for(i in 1:length(penalities)){
  temp <- spca(x = mat_scaled,
                 K = 5,
                 type = 'predictor',
                 para = c(rep(penalities[i],ncol(mat_scaled))),
                 sparse = 'penalty')
  spca_pevs[i] <- sum(temp$pev)
  spca_loadings <- temp$loadings
  ps <- length(spca_loadings[abs(spca_loadings)<=0.01]) / length(spca_loadings)
  spca_ps[i] <- ps
  if(length(ps)==0){ps <- 0}
}

is_pev_dt <- data.table(lambda=penalities,
                        spca_PEV=spca_pevs,
                        spca_PS=spca_ps)
standard_PCA_PEV <- is_pev_dt[lambda==0]$spca_PEV
is_pev_dt[,IS:=standard_PCA_PEV * spca_PEV * spca_PS]

my_plot <- ggplot(is_pev_dt, aes(x=spca_PS)) +
  geom_line( aes(y=IS)) +
  geom_line( aes(y=spca_PEV),linetype='dashed') +
  scale_y_continuous(name = "IS",
    sec.axis = sec_axis(~.*1, name="PEV",breaks = scales::pretty_breaks(n=8)),
    breaks = scales::pretty_breaks(n=8)) +
  ggtitle('PEV x IS x PS') +
  theme_gray(base_size = 10)
plot(my_plot)
```




The IS score is maximized for $\lambda = 80$: at this value, we only see a marginal reduction in variable explained, while having principle loadings which are more than 80% sparse — nice!

```
# Re-fit best spec
spcaout <- spca(x = mat_scaled,
                K = 5,
                type = 'predictor',
                para = c(rep(80,5)), sparse = 'penalty')
# Print loadings
spcaout$loadings
```

```
##                              PC1        PC2       PC3       PC4 PC5
## taste_jazz             0.0000000 -0.3884487 0.0000000 0.0000000   0
## taste_classical        0.0000000 -0.4026162 0.0000000 0.0000000   0
## taste_blues            0.0000000 -0.4352764 0.0000000 0.0000000   0
## taste_pop              0.0000000  0.3953759 0.0000000 0.0000000   0
## taste_country          0.0000000  0.3932364 0.0000000 0.0000000   0
## taste_raegge           0.0000000  0.4319552 0.0000000 0.0000000   0
## income                -0.4233497  0.0000000 0.0000000 0.0000000   0
## nbhood_avg_income     -0.3904130  0.0000000 0.0000000 0.0000000   0
## education             -0.4083279  0.0000000 0.0000000 0.0000000   0
## nbhood_avg_education  -0.3905200  0.0000000 0.0000000 0.0000000   0
## nhood_crime            0.3864943  0.0000000 0.0000000 0.0000000   0
## nbhood_unemployment    0.4469199  0.0000000 0.0000000 0.0000000   0
## nhbood_avg_temp        0.0000000  0.0000000 0.0000000 0.0000000   0
## nhbood_pop             0.0000000  0.0000000 0.0000000 0.0000000   0
```

```
## nhbood_nr_lights          0.0000000  0.0000000  0.0000000  0.0000000   0
## nhbood_nr_pizzerias       0.0000000  0.0000000  0.0000000  0.0000000   0
## city_avg_taste_jazz       0.0000000  0.0000000 -0.3978269  0.0000000   0
## city_avg_taste_classical  0.0000000  0.0000000 -0.3952236  0.0000000   0
## city_avg_taste_blues      0.0000000  0.0000000 -0.4116225  0.0000000   0
## city_avg_taste_pop        0.0000000  0.0000000  0.4615278  0.0000000   0
## city_avg_taste_country    0.0000000  0.0000000  0.3825235  0.0000000   0
## city_avg_taste_raegge     0.0000000  0.0000000  0.3959380  0.0000000   0
## taste_film_action         0.0000000  0.0000000  0.0000000  0.6298242   0
## taste_film_romcom         0.0000000  0.0000000  0.0000000  0.0000000  -1
## taste_film_documentary    0.0000000  0.0000000  0.0000000 -0.7767377   0
```

The sparsity makes it considerably easier to interpret these loadings. For example, PC1 isolates properties about person- and neighborhood SES, while PC2 captures musical tastes.

6. As a last exercise for today, you shall simulate your own data. Generate a dataset of 50 observations and 50 independent variables using the function provided below:

```
gen_data <- function(n,p){
  df <- c()
  for(i in 1:p){
    ith_var <- rnorm(n = n, mean = 0, sd = 1)
    df <- cbind(df,ith_var)
  }
  return(df)
}
```

- Once you have generated the data, process your data as you did above for the `neighborhood` data set (standardize, making into a matrix). Then, estimate a standard PCA. What do you find: could the PCA help us effectively reduce the dimensionality of our data or not? Why?