Final

Hyun, Heo

2018년 9월 9일

# 1

## 1-1

(y-XB)’(y-XB) + lambdaB’B를 최소로 하는 B(Beta)를 구하는 문제 or l2 optimization 문제

## 1-2

y-XB가 대칭행렬이기 때문에 B를 구성하는 두 값이 같다

## 1-3

l1 optimization 문제

## 1-4

l1이기 때문에 non-singular 문제를 해결해주는 것이 아니다. 따라서 두 베타 예측값이 다르고 많다

# 2

## 2-1

1. IQ와 GPA가 고정되어 X1, X2, X4가 고정되었고, X5는 constant\*Gender 꼴이 되었기 때문에 계산이 가능하고 평균적으로 학점은 3.5보다 낮은 수준이다.

## 2-2

50-80+7.7+35+4.4-40 = -22.9

## 2-3

거짓. coefficient는 좋은 정보를 주지만 스케일링 등의 고질적 문제가 있기 때문에 계수만으로 모든 것을 합리화 할 수는 없

# 3

EDA 과정을 모두 써야할지 아닐지 고민하다가 특히 3번 문제의 경우 EDA 시간이 매우 길었기에 굳이 EDA 과정을 코드로 모두 쓰진 않기로 했습니다. 대신 문제를 *두가지 방식*으로 풀었는데 첫번째는 수업 중 배운 알고리즘을 활용해서 풀었고, 두번째 방식은 저만의 방식으로 풀었습니다. 이 때 두번째 방식에 EDA 과정에서 어떤 정보를 얻었고 어떻게 활용하겠다는 생각을 했는지 서술하겠습니다.

political$k3[political$k3==2] <- 0 # change 1,2 to 0,1

sum(political$ideo\_self==5)/nrow(political) # at least this

## [1] 0.3064516

우선 k3변수가 다른 변수와 달리 1,2로 처리되어 있어서 0,1로 바꿔주었습니다. 그리고 ideo\_self가 5인 경우가 굉장히 많아 전체를 무조건 5로 예측한 경우를 기준으로 삼기 위해 이 때 예측률을 미리 확인했습니다.

colSums(is.na(political))

## id sex birth age1 age area edu   
## 0 0 0 0 0 0 0   
## income k2 k3 k4 k6 k7 k8   
## 0 227 200 211 177 270 133   
## k10 k12 k13 k14 ideo\_self   
## 278 110 88 120 0

colSums(political[9:18], na.rm = T)

## k2 k3 k4 k6 k7 k8 k10 k12 k13 k14   
## 543 440 453 631 201 355 352 443 260 741

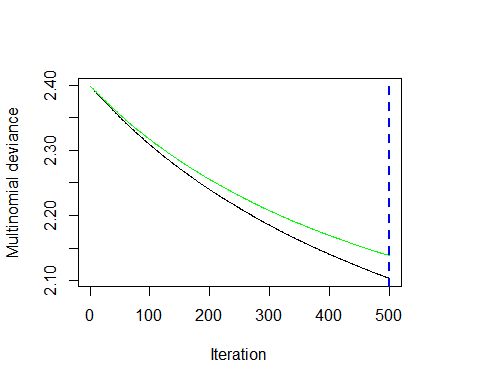
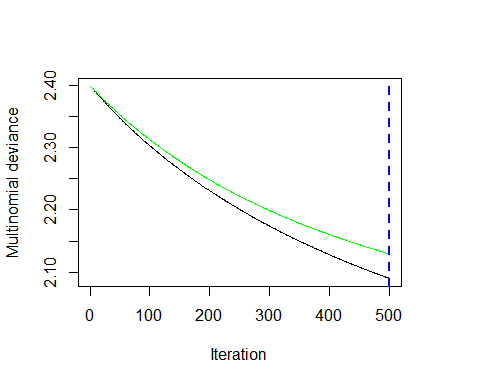
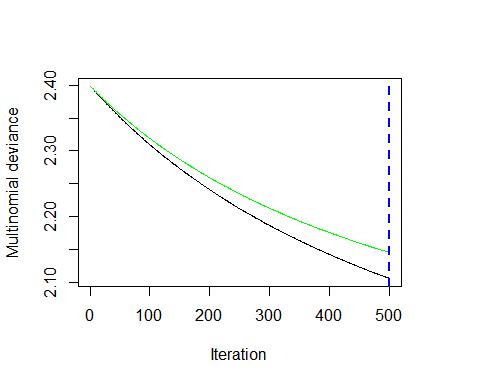
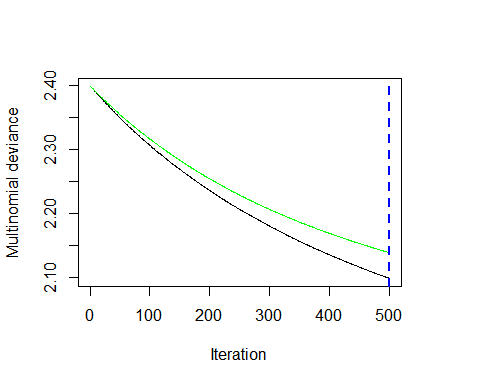
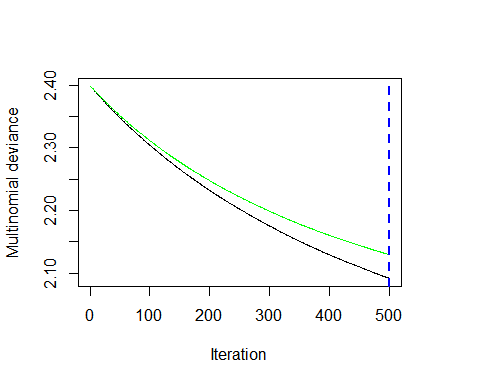
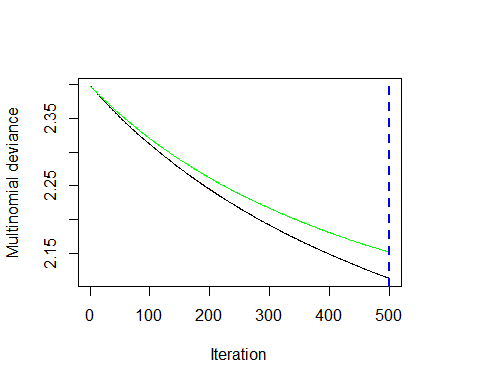
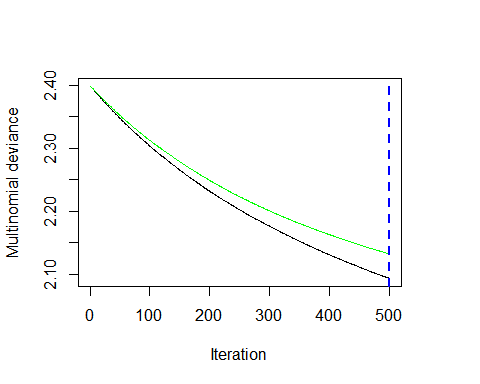
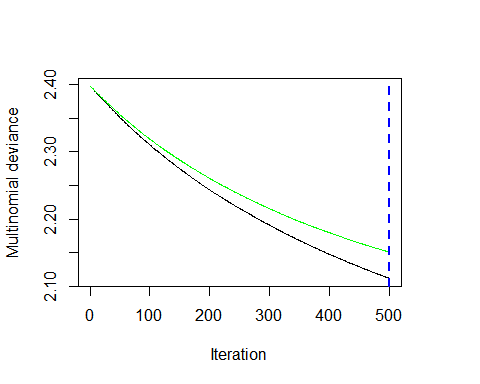
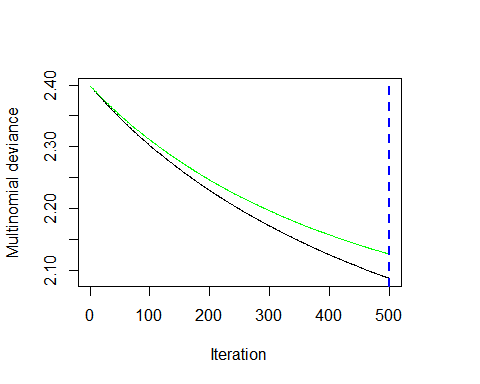
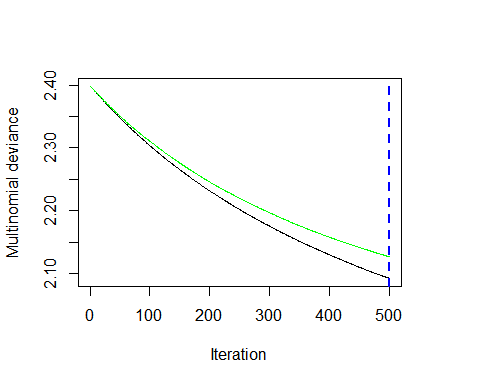
결측값 정보를 확인했고, 확인 결과 모든 결측치는 정치 문항에 대해 패스한 경우에만 발생했다는 것을 확인할수 있었습니다. 일반적으로 패스한 경우 선택이 매우 어려웠거나 중립인 경우에 해당한다는 점을 추후 분석에 고려하였습니다.

poliforest <- political %>%  
 select(k2:k14)  
poliforest[poliforest==0] <- -1 # change 0 to 1  
poliforest[is.na(poliforest)] <- 0 # change NA to 0  
political2 <- political  
political2[,9:18] <- poliforest  
pol2 <- political2 %>%   
 select(sex, age1, area:ideo\_self)  
pol2$ideo\_self <- as.factor(pol2$ideo\_self)  
pol2$area <- as.factor(pol2$area)

위에서 설명했듯 결측값에도 응답자의 정보가 담겨있으므로 이 정보를 살리고자 했습니다. 기존 0,1로 적던 방식을 -1,0,1로 바꾸어 패스한 경우 0으로 처리하였습니다. 또한 두가지 형태의 나이 변수 중 더 많은 정보를 포함한 age1 변수를 선택하였습니다. 추가적으로 factor형으로 꼭 설정해야 한다고 생각한 변수인 ideo\_self와 지역에 대해서는 factor화 했습니다.

## 3-1

fold\_point <- c(seq(0, 900, 100), 1054)  
ten\_tables <- matrix(0, 11, 11)  
colnames(ten\_tables) <- paste0("pred", 0:10)  
rownames(ten\_tables) <- paste0("real", 0:10)  
ten\_tables <- as.table(ten\_tables)  
for (i in 1:10){  
 start\_point <- fold\_point[i]+1  
 end\_point <- fold\_point[i+1]  
 test\_idx <- start\_point:end\_point  
 test\_set <- pol2[test\_idx,]  
 train\_set <- pol2[-test\_idx,]  
 boost.pol=gbm(ideo\_self~.,data=train\_set,  
 distribution="multinomial",n.trees=500, interaction.depth = 1, cv.folds = 5)  
 best.iter = gbm.perf(boost.pol, method="cv")  
 fitControl = trainControl(method="cv", number=5, returnResamp = "all")  
 model2 = train(ideo\_self~.,data=train\_set, method="gbm",distribution="multinomial",  
 trControl=fitControl, verbose=F,  
 tuneGrid=data.frame(.n.trees=best.iter, .shrinkage=0.02, .interaction.depth=1, .n.minobsinnode=1))  
 mPred = predict(model2, test\_set, na.action = na.pass)  
 itable <- table(real = test\_set$ideo\_self, pred = mPred)  
 ten\_tables <- ten\_tables + as.table(itable)  
}



10-fold cv를 수행하기 위해 총 1054개의 관측값을 100 단위로 자르고, 마지막 fold는 901부터 1054까지가 되게 했습니다. statistical learning 알고리즘은 gbm을 이용한 부스팅 트리를 사용했습니다. 이 알고리즘을 사용한 이유는 클래스간 불균형이 매우 컸기 때문입니다. 만약 2가지 중 하나를 예측하는 문제라면 불균형 문제를 부족한 부분의 데이터를 늘려주면서 해결하는 방법을 사용했을 것 같지만, 이 경우 11개의 결과가 존재하기 때문에 어떤 데이터를 얼마나 늘릴지도 판단하기 어려웠습니다. 따라서 점차적으로 예측력이 안 좋은 부분을 개선해나가는 부스팅 트리 방식이 좋다고 생각했습니다. 각 fold 별로 confusion matrix를 만들어서 따로 더했고, 모두 더한 confusion matrix를 최종본으로 썼습니다.

ten\_tables

## pred0 pred1 pred2 pred3 pred4 pred5 pred6 pred7 pred8 pred9 pred10  
## real0 0 0 0 4 0 28 3 1 0 0 1  
## real1 0 0 0 3 2 22 0 0 0 0 0  
## real2 0 0 0 6 5 32 2 0 0 1 0  
## real3 0 0 1 15 7 93 2 0 1 0 0  
## real4 0 0 2 13 2 93 1 3 1 0 1  
## real5 1 0 0 18 8 273 10 6 5 0 2  
## real6 0 0 0 3 5 94 2 6 9 1 1  
## real7 0 0 1 5 2 68 1 14 10 5 0  
## real8 0 0 0 6 2 38 2 5 11 2 7  
## real9 0 0 0 0 0 10 4 4 4 0 3  
## real10 0 0 0 2 1 32 1 7 13 2 3

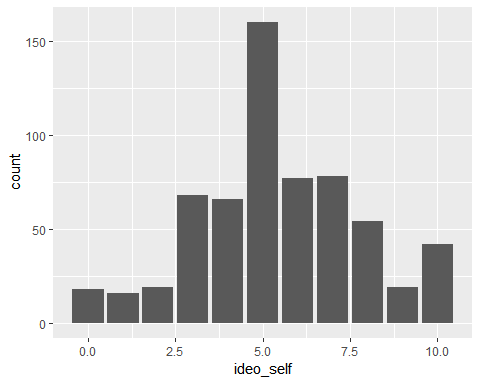
corrected <- sum(diag(ten\_tables))  
corrected/sum(ten\_tables)

## [1] 0.3036053

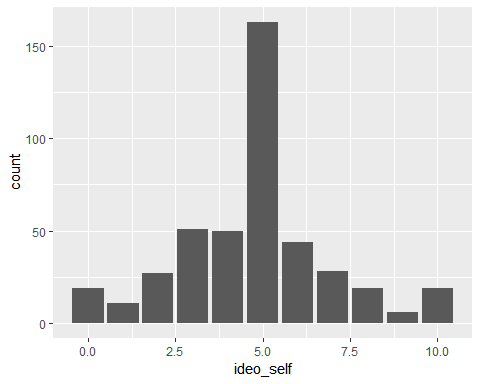
위는 최종 confusion matrix와 accuracy 값입니다.

## 3-2

male <- which(political[,"sex"] == 1)  
ggplot(political[male,]) + geom\_bar(aes(ideo\_self)) # conservative



ggplot(political[-male,]) + geom\_bar(aes(ideo\_self)) # neutral



두번째 방법은 위와 같은 방식으로 그림을 그려서 변수별로 어떤 때 진보적인지 보수적인지를 확인하고 그에 맞게 점수를 주는 rule-based 방식으로 진행했습니다. 예측력은 훨씬 떨어지지만 각 변수에 대해 어떤 답을 했을 때, 혹은 어느 지역에 살 때 더 보수적인지 진보적인지 설명할 수 있다는 점에서 매력적인 접근법이라고 생각했습니다. 또한 바 그래프가 굉장히 한 쪽으로 치우친 경우 추가적으로 점수를 더 많이 주고, 아주 조금 치우친 경우에는 점수를 덜 주는 방식으로 진행했습니다. 다음은 각 변수별 진보/보수 정도 평가 내용입니다.

* sex : 남자면 보수, 여자면 중도
* age1 : 50세 이하면 중도, 초과면 매우 보수
* edu : 1이면 조금 진보, 5면 조금 보수, 그 외 중도
* area : 2, 7, 10, 11, 14이면 보수, 5, 9, 12, 15, 16면 진보
* income : 8, 14, 15면 보수, 2, 3, 5, 6, 7, 9면 진보
* k2 : 1이면 보수, 0이면 진보
* k3 : 1이면 보수, 0이면 진보
* k4 : 1이면 매우 보수, 0이면 진보
* k6 : 1이면 보수, 0이면 진보
* k7 : 1이면 매우매우 보수, 0이면 중도
* k8 : 1이면 매우매우 보수, 0이면 중도
* k10 : 1이면 진보, 0이면 조금 보수
* k12 : 1이면 진보, 0이면 보수
* k13 : 1이면 매우매우매우 보수, 0이면 진보
* k14 : 1이면 조금 보수, 0이면 조금 진보
* 모든 결측값 : 중도

std\_poli <- political %>%  
 mutate(pred = ifelse(is.na(k2), 0, ifelse(k2==1, 1, -1)) +  
 ifelse(is.na(k3), 0, ifelse(k3==1, 1, -1)) +  
 ifelse(is.na(k4), 0, ifelse(k4==1, 1.5, -1)) +  
 ifelse(is.na(k6), 0, ifelse(k6==1, 1, -1)) +  
 ifelse(is.na(k7), 0, ifelse(k7==1, 2, 0)) +  
 ifelse(is.na(k8), 0, ifelse(k8==1, 2, 0)) +  
 ifelse(is.na(k10), 0, ifelse(k10==1, -1, 0.5)) +  
 ifelse(is.na(k12), 0, ifelse(k12==1, -1, 1)) +  
 ifelse(is.na(k13), 0, ifelse(k13==1, 2.5, -1)) +  
 ifelse(is.na(k14), 0, ifelse(k14==1, 0.5, -0.5)) +  
 ifelse(sex==1, 1, 0) +  
 ifelse(age1>50, 1.5, 0) +  
 ifelse(edu==1, -0.5, ifelse(edu==5, 0.5, 0)) +  
 ifelse(area %in% c(2, 7, 10, 11, 14), 1, ifelse(area %in% c(5, 9, 12, 15, 16), -1, 0)) +  
 ifelse(income %in% c(8, 14, 15), 1, ifelse(income %in% c(2, 3, 5, 6, 7, 9), -1, 0))  
 ) %>%  
 mutate(standard\_pred = ifelse(pred < -7, 0,  
 ifelse(pred >= -7 & pred < -5, 1,  
 ifelse(pred >= -5 & pred < -3, 2,  
 ifelse(pred >= -3 & pred < -1.5, 3,   
 ifelse(pred >= -1.5 & pred < 0, 4,  
 ifelse(pred >= 0 & pred < 4, 5,  
 ifelse(pred >= 4 & pred < 6.5, 6,  
 ifelse(pred >= 6.5 & pred < 9, 7,  
 ifelse(pred >= 9 & pred < 12, 8,   
 ifelse(pred >= 12 & pred < 15, 9, 10  
 )))))))))))

위 아이디어에 기반해 점수를 더하거나 빼주었고, 그렇게 나온 점수를 0에서 10으로 할당해주었는데 이 때 구간이 많아지다 보니 양끝 부분이 적어지는 현상과 5에 해당하는 응답자가 많았다는 점을 고려하여 할당해주었습니다. 따라서 두번째 분석에서는 보수/진보/중도를 나누는 것과 구간을 나눠주는 것에서 저의 주관이 어느 정도 개입되었습니다.

table(real = std\_poli$ideo\_self,pred = std\_poli$standard\_pred)

## pred  
## real 0 1 2 3 4 5 6 7 8 9 10  
## 0 0 1 5 7 4 15 4 0 1 0 0  
## 1 1 2 3 9 3 8 1 0 0 0 0  
## 2 2 4 9 9 5 11 4 2 0 0 0  
## 3 1 10 24 12 17 37 8 8 2 0 0  
## 4 4 11 13 12 9 41 10 9 2 4 1  
## 5 1 4 18 27 52 134 38 24 15 8 2  
## 6 1 1 5 6 14 33 19 14 16 10 2  
## 7 0 0 4 3 6 25 13 14 16 18 7  
## 8 1 1 1 2 2 4 6 9 17 26 4  
## 9 0 0 0 0 1 2 1 5 7 8 1  
## 10 0 0 0 0 3 14 5 11 7 14 7

sum(std\_poli$ideo\_self == std\_poli$standard\_pred)/nrow(std\_poli)

## [1] 0.2191651

이 분류법은 train의 과정이 없기 때문에 바로 전체 데이터에 적용해보고 confusion matrix와 예측률을 구했습니다. 예측률 자체는 그렇게 높지 않으나 전체적인 매트릭스가 정확한 정답 근처에 예측을 했다는 점을 감안하면 괜찮은 방법이라고 생각됩니다.

# 4

## 4-1

apply(fin4[,2:51], 1, mean)

## [1] -1.119512e-01 1.262501e-01 -4.314997e-03 -9.015413e-02 -1.899648e-01  
## [6] -2.299286e-01 1.960367e-01 1.281031e-01 2.410717e-01 4.480088e-02  
## [11] 8.180400e-03 -4.542782e-02 -1.179732e-01 -4.586133e-02 -6.488943e-02  
## [16] -9.204562e-02 -2.002221e-01 -1.426771e-01 1.926840e-01 -1.271785e-01  
## [21] -1.116966e-01 7.389528e-02 -2.817975e-01 -1.363722e-01 -1.076601e-01  
## [26] -1.279501e-01 1.100865e-01 -1.349326e-01 -2.199271e-01 3.839812e-02  
## [31] 5.602946e-03 1.488939e-01 1.903632e-01 2.028871e-01 3.577356e-02  
## [36] -5.432994e-02 -1.638337e-01 -1.072551e-01 -6.058164e-02 -1.764308e-01  
## [41] 1.520173e-01 -9.788900e-03 -1.357113e-01 -1.716025e-01 -4.078084e-01  
## [46] 3.066795e-01 -1.504721e-01 1.993871e-01 -4.505282e-02 -1.120795e-01  
## [51] -2.984205e-02 2.781114e-02 5.197680e-02 -1.214505e-01 -1.499517e-01  
## [56] -2.502181e-01 -1.254591e-01 2.485573e-01 -1.327527e-01 2.666812e-01  
## [61] -3.213761e-02 3.313813e-01 -2.561131e-02 -1.789471e-01 6.976530e-03  
## [66] 6.771392e-03 1.384588e-01 1.946355e-02 -1.939967e-02 2.108954e-02  
## [71] 1.020144e-01 5.898723e-03 8.147882e-02 6.346998e-02 -5.116474e-02  
## [76] -1.502789e-02 -2.139979e-02 -1.903841e-02 -4.135843e-02 2.264233e-01  
## [81] 2.227546e-03 -1.395189e-02 1.157653e-01 -5.463045e-02 -4.235509e-02  
## [86] -2.335837e-01 2.922216e-01 2.242060e-02 1.042625e-01 2.018050e-01  
## [91] -6.048530e-02 1.151447e-01 -1.412597e-01 1.427943e-01 2.776339e-02  
## [96] 1.348055e-01 1.130569e-05 1.142923e-01 7.353153e-02 6.112875e-02  
## [101] -6.979923e-02 6.536687e-02 1.367782e-01 -1.110749e-01 3.356722e-02  
## [106] 4.691067e-02 2.192622e-02 6.950367e-02 1.307659e-01 -5.217060e-02  
## [111] 1.314289e-01 2.635678e-01 1.022505e-01 -1.232378e-01 2.303790e-02  
## [116] -4.812981e-02 -1.471847e-01 3.230487e-02 3.167915e-02 -1.343067e-01  
## [121] 1.868543e-01 8.154899e-02 1.191345e-01 1.812880e-01 -1.288577e-01  
## [126] -2.449688e-01 -9.341646e-02 -1.108663e-02 1.338484e-01 2.328865e-01  
## [131] 7.700258e-02 -7.800807e-02 -1.302937e-01 1.287631e-01 1.631027e-01  
## [136] 4.793980e-02 1.604908e-01 8.051813e-02 5.010270e-04 -1.129306e-02  
## [141] -1.748738e-01 -1.137595e-01 -2.192857e-01 6.262273e-02 2.657291e-01  
## [146] -1.150547e-01 7.154587e-02 -9.490369e-02 -2.713105e-01 7.743513e-02  
## [151] 1.752230e-01 -6.565536e-02 6.288620e-02 2.632385e-02 -8.985074e-02  
## [156] -1.126748e-01 -3.126056e-01 -5.568814e-02 1.475850e-01 -2.807722e-01  
## [161] 1.533176e-01 1.028335e-01 3.481990e-01 -1.037915e-01 -1.065142e-01  
## [166] -2.253544e-02 1.128003e-02 5.811299e-02 -7.923484e-02 -2.188061e-01  
## [171] -9.611373e-02 1.040762e-01 2.003621e-01 5.760873e-02 1.411930e-01  
## [176] -1.020769e-02 3.267766e-02 -9.599513e-02 2.217459e-01 -4.822235e-02  
## [181] 2.156790e-03 -1.023285e-01 -6.330377e-02 8.897167e-02 -8.664624e-03  
## [186] 5.379370e-02 1.484801e-01 2.013762e-02 5.241187e-02 -8.010345e-02  
## [191] 4.495722e-02 -1.896731e-01 1.641420e-01 -1.541035e-02 6.891394e-02  
## [196] 1.059217e-01 3.523239e-01 -3.185336e-01 -3.470198e-02 -1.063561e-01  
## [201] -3.459882e-01 1.972004e-01 8.694143e-03 -4.779690e-02 1.161380e-01  
## [206] -2.741643e-02 6.110103e-02 9.816391e-02 -9.529190e-02 -2.400932e-01  
## [211] 1.427696e-01 -6.891162e-02 -6.809762e-02 9.475057e-02 3.070574e-01  
## [216] -1.583553e-01 1.973768e-01 -1.603774e-01 -1.818650e-02 1.951717e-02  
## [221] 1.625309e-01 -5.194020e-02 -1.828057e-02 9.079414e-02 1.046020e-01  
## [226] -9.977886e-02 -1.807811e-01 -8.523140e-03 1.943499e-01 1.715466e-01  
## [231] -2.518610e-02 5.793112e-03 3.131525e-02 2.320404e-01 1.772580e-01  
## [236] -1.445700e-01 -1.206315e-01 9.480981e-02 -4.884062e-02 -6.280649e-02  
## [241] -2.115188e-01 -5.353313e-02 9.054540e-02 9.754300e-03 -3.134720e-02  
## [246] 1.349951e-01 -1.005371e-01 -1.168890e-01 2.776857e-01 1.856306e-01  
## [251] -2.496290e-01 4.113503e-02 -5.425761e-02 -4.749734e-02 -9.347761e-02  
## [256] -6.787000e-02 1.291870e-02 2.613173e-01 -1.339190e-01 3.274006e-02  
## [261] -4.414164e-02 1.117804e-01 6.420671e-02 -7.466135e-02 1.375653e-01  
## [266] 9.861681e-02 -8.738093e-02 1.634123e-01 -1.746975e-01 4.418705e-02  
## [271] 2.180114e-01 7.482151e-02 9.907250e-02 -1.092963e-01 -1.496015e-01  
## [276] 3.261381e-02 -1.673936e-02 1.652781e-02 -9.730586e-02 -1.498447e-01  
## [281] 3.150902e-02 6.078510e-02 -1.389731e-01 -2.326933e-01 -2.893294e-02  
## [286] -3.045940e-02 -1.959347e-01 -6.712156e-02 5.998624e-02 -1.757223e-01  
## [291] 2.718648e-01 -1.476326e-01 1.723452e-02 -3.915583e-02 -7.403858e-02  
## [296] -2.493097e-01 -2.973775e-01 -2.169050e-02 7.423744e-02 -3.816831e-02  
## [301] 7.951975e-02 2.599281e-01 1.336547e-01 4.468687e-02 -2.709296e-01  
## [306] -2.080903e-01 -8.527746e-02 4.690533e-02 -1.613171e-01 9.951123e-03  
## [311] 8.000282e-02 2.069162e-01 5.496566e-03 -1.131072e-01 -6.606084e-02  
## [316] -8.439400e-02 1.787582e-01 8.666511e-02 1.290083e-01 -1.282503e-01  
## [321] 1.129148e-02 1.978475e-01 2.071661e-02 9.227827e-02 9.119849e-02  
## [326] 1.819041e-01 5.237409e-02 1.955147e-02 2.443889e-01 -9.054969e-02  
## [331] -1.651702e-01 -4.005795e-02 -4.013824e-02 -5.624432e-02 -8.386794e-02  
## [336] 1.291580e-01 -4.060949e-02 3.065681e-02 2.452693e-02 -3.757656e-02  
## [341] -1.041084e-01 3.333976e-01 3.533243e-01 1.364444e-01 -8.523478e-02  
## [346] -2.081105e-01 6.911638e-02 -1.106762e-01 7.640941e-02 1.415959e-01  
## [351] 2.896968e-01 -4.465975e-02 -6.186647e-02 -1.879389e-01 -1.600274e-01  
## [356] -1.434957e-01 -4.800463e-02 -1.056202e-01 -4.182718e-02 2.904784e-01  
## [361] 9.357703e-02 2.362266e-01 -1.566214e-01 2.074217e-01 2.038786e-02  
## [366] 7.501852e-02 1.665597e-01 -1.379934e-01 -2.889742e-01 7.837442e-02  
## [371] -1.320514e-01 1.510171e-01 6.856688e-02 -1.021306e-01 2.242696e-01  
## [376] 2.434635e-01 1.550893e-01 -1.443914e-02 8.116718e-03 -1.982226e-01  
## [381] 2.807986e-01 5.656814e-02 3.141409e-01 2.013049e-02 -5.100074e-02  
## [386] 8.535599e-03 5.517798e-02 6.949942e-02 -2.717180e-02 6.928202e-02  
## [391] -1.790813e-01 9.608348e-02 -1.918250e-01 1.162912e-01 5.547237e-02  
## [396] 2.778930e-02 -3.329478e-01 1.215574e-01 -1.026649e-01 4.892799e-03  
## [401] 2.956264e-01 1.090478e-01 -3.025165e-01 -4.566007e-02 -4.753419e-02  
## [406] 8.805013e-02 6.735701e-02 -9.094449e-02 2.320989e-01 -1.313759e-01  
## [411] -4.557778e-02 -1.008263e-02 -1.084272e-01 5.941265e-02 -2.336261e-01  
## [416] -4.680383e-02 1.862769e-01 2.252482e-01 -1.700819e-03 1.379649e-01  
## [421] 1.558473e-01 6.010670e-02 -2.557407e-02 2.832732e-01 1.036100e-01  
## [426] 1.077363e-01 1.859881e-01 6.547686e-02 -4.314990e-02 1.223203e-01  
## [431] 8.659574e-02 3.327793e-02 3.744014e-02 -8.880620e-02 2.391208e-02  
## [436] 1.958763e-01 6.627406e-02 1.669166e-01 2.059512e-01 7.887214e-02  
## [441] 7.970617e-02 -4.087334e-02 1.603839e-01 -8.207172e-02 -7.501798e-02  
## [446] 1.588635e-01 -2.709841e-01 -2.068981e-02 -2.029238e-01 1.372665e-01  
## [451] -2.809583e-01 2.521666e-02 2.447943e-01 1.220005e-01 3.620960e-02  
## [456] -9.244501e-02 9.336316e-02 -9.963286e-02 -1.887110e-01 -8.630544e-02  
## [461] -5.478599e-02 -1.650867e-01 1.482833e-01 -3.617160e-02 3.286079e-01  
## [466] -2.408392e-03 6.900348e-02 -2.037012e-01 4.215251e-01 8.983209e-02  
## [471] -2.284746e-01 -8.745738e-02 -9.797497e-02 1.590182e-01 -5.035448e-02  
## [476] 3.300919e-01 -5.850971e-02 1.526982e-01 2.316638e-01 -1.999968e-02  
## [481] -4.076147e-01 -2.455111e-01 7.277804e-02 -1.122642e-01 -1.358498e-01  
## [486] -3.050137e-01 4.132095e-02 8.712348e-02 7.247664e-02 2.580364e-02  
## [491] 1.269879e-01 1.292463e-01 -1.559918e-01 1.985831e-02 -1.745994e-01  
## [496] 2.233506e-01 3.722501e-02 4.017890e-02 1.341909e-02 1.285696e-01

apply(fin4[,2:51], 1, sd)

## [1] 1.1532362 1.0419449 0.9811217 0.9812712 1.0872637 0.8811231 1.0070331  
## [8] 0.9606337 1.0413859 0.9471223 1.0474860 0.9987741 1.0807639 0.8723761  
## [15] 0.9885048 0.9962271 0.8770738 1.0667361 0.9822108 1.0067490 1.2461354  
## [22] 0.9808990 0.8686349 1.0930241 1.0075558 1.0215193 1.1466281 1.3370443  
## [29] 0.9250061 0.8467346 1.1865981 1.0613769 1.0715136 1.1079710 1.1105514  
## [36] 1.1195752 1.0058387 0.9290791 0.9534196 0.9835141 0.9403938 0.9990899  
## [43] 1.1397213 1.0077247 0.9714979 0.8474572 1.0420272 1.1128262 1.0161654  
## [50] 1.0257325 1.0163073 0.8994848 0.8838584 0.9370156 1.0148747 1.0735132  
## [57] 0.9512399 1.0760045 0.8582662 0.8102162 1.0372378 0.8989769 1.0511419  
## [64] 1.0973280 1.0407318 1.0265635 0.9280073 1.0321383 0.8834577 0.9412213  
## [71] 1.2182585 1.1010357 0.9029594 1.0388332 0.8650615 1.0227023 1.0709345  
## [78] 0.8351609 1.1200896 0.8855556 0.9766476 1.0806505 1.2103187 0.9404074  
## [85] 1.0653646 1.1622005 1.1423697 0.9529719 0.9136199 0.9713156 1.0390778  
## [92] 0.8615918 0.9599331 0.9571883 1.0287722 0.9494822 1.0987771 0.9872729  
## [99] 1.0661742 0.8657089 1.0197938 1.0530005 0.9428893 0.9788138 1.0950390  
## [106] 0.9425017 1.0017766 1.0072377 1.0234579 0.9961689 1.0407637 1.1548588  
## [113] 0.9100306 1.0125331 0.9580228 1.0576699 0.9654127 1.0293254 1.0229209  
## [120] 1.0756764 1.0112132 1.0164886 1.1636230 1.1842481 1.1159620 1.0061281  
## [127] 0.8618263 1.0747972 1.0427168 1.2111869 1.0724815 0.9143582 0.8236116  
## [134] 1.0872029 0.9733685 0.9901583 0.9424155 1.2082788 1.0392340 0.9989389  
## [141] 0.9981984 1.0314437 0.9293597 1.2104687 1.1427377 1.1583436 0.8366106  
## [148] 1.0002005 0.9410981 0.9888415 0.9451042 0.9913319 1.0604484 1.0166318  
## [155] 1.0995634 1.0031009 1.0569333 0.8057157 0.9738212 0.9050445 1.1404780  
## [162] 1.0414971 1.1098543 1.0246157 1.0343003 1.0758624 1.1100919 1.0404841  
## [169] 1.1522524 0.7607031 0.9317253 1.0864962 1.0300117 1.2513285 1.2262892  
## [176] 0.9667432 1.1678014 0.9994044 0.8931618 1.0242095 0.8984801 1.1159061  
## [183] 1.0710948 0.9570691 0.9952074 1.0416720 1.1200241 1.1284416 1.0254256  
## [190] 0.9455890 0.9771578 0.8078726 1.0074436 0.9217743 1.1576951 1.0209985  
## [197] 1.0638288 1.1926845 0.9554443 1.1186538 1.0486746 1.0700489 0.9973982  
## [204] 1.0275665 0.9071477 1.0737719 1.0171534 0.9393732 1.0595827 0.9798713  
## [211] 0.9806382 1.1265987 1.1257715 1.1006956 1.1371293 1.0495291 1.1299737  
## [218] 0.8828857 1.0045640 0.9850761 0.9314841 1.0188944 0.8722989 1.0486473  
## [225] 1.0381898 1.0178064 1.0992708 1.0012914 1.0265006 0.9300567 0.9797883  
## [232] 1.1523199 1.1031555 1.0554135 1.0816835 1.0174310 0.9205483 1.0730364  
## [239] 0.9945261 0.9911223 1.0050459 0.9819955 1.1705026 1.0221134 0.8747595  
## [246] 1.0318802 1.0849847 1.0641638 0.9020472 1.2225636 1.1210797 0.9017882  
## [253] 0.9743328 1.0240877 1.1199170 1.0995207 1.0721132 1.1037581 0.9900501  
## [260] 1.1151573 1.2355680 0.9627200 1.1666280 0.9957926 0.9801395 1.0250940  
## [267] 1.0106947 1.0165736 1.0229537 1.0200482 0.9791611 1.2909150 1.1074901  
## [274] 1.0555280 1.0358247 0.8499831 0.9176087 1.1173398 0.9287653 1.0738231  
## [281] 1.1370840 1.0923277 1.1906191 0.9329515 0.9385350 0.8952450 1.0274728  
## [288] 1.0703566 1.1949390 1.0013525 1.0702532 1.1382435 1.1212605 1.0641425  
## [295] 1.1164844 1.0832275 1.2770326 1.1373785 0.9673448 0.9293238 1.0596261  
## [302] 1.1329474 1.2431549 1.1403805 1.0211278 0.9247309 1.0118187 1.1517549  
## [309] 1.0459697 0.9339682 1.0395302 1.0276853 1.0392842 0.9604418 1.1469337  
## [316] 1.2200158 1.0246585 1.1098209 1.1566148 0.8932337 1.2260352 0.9991388  
## [323] 0.9742771 1.0578023 0.9487181 1.1028129 1.0714101 1.0201006 1.0701141  
## [330] 1.0708928 1.0089631 0.9814779 1.1076784 1.1817280 1.1409408 0.9811582  
## [337] 1.0646979 1.1965029 1.0602240 1.0425771 0.9573637 0.8711056 1.1278203  
## [344] 0.9630591 0.9668003 1.1064090 1.2549297 1.0231092 0.9889615 1.1773131  
## [351] 1.0025707 1.0284955 1.1538093 0.8934601 1.0819823 1.2223484 1.0400590  
## [358] 1.0855267 1.0194557 1.1708470 1.0295843 1.2454734 1.0300240 1.1653975  
## [365] 1.0049141 1.0342587 1.1139437 1.0919284 0.9666606 1.0682418 0.9788719  
## [372] 1.0123300 1.1447944 1.1281008 0.9797513 1.0488273 1.0988655 1.2272969  
## [379] 1.2247413 1.1098055 0.9633341 1.1662885 1.0542478 0.9796622 1.1898470  
## [386] 1.2340356 1.0824097 1.0801206 1.3141903 1.0742801 1.0725456 1.1696033  
## [393] 0.9198053 1.0056503 1.0462385 1.1434193 1.0749518 1.1319949 1.0008641  
## [400] 0.9119081 1.0901681 1.0442986 1.0851174 0.8822149 1.0539136 1.1484680  
## [407] 0.9794957 1.0051999 0.9607919 1.1376958 1.0304846 1.1835506 0.9301016  
## [414] 1.2767603 1.0684135 1.1229693 1.2058793 0.9381450 0.9922082 1.1246862  
## [421] 1.2725079 0.9642690 0.9302761 1.1063103 1.0859884 1.0376829 1.1123554  
## [428] 1.0753819 1.0961068 1.0003923 1.1244743 1.0673338 1.0414913 0.9215108  
## [435] 1.0725150 0.9915168 1.0342620 1.0477822 1.1580343 1.0792751 1.0853052  
## [442] 0.9984506 1.1496844 1.0469082 1.1202898 1.0849803 1.0972857 0.9418902  
## [449] 1.0521280 1.0416136 1.0075969 0.9257314 1.0205799 0.9611488 1.1335227  
## [456] 0.9362684 1.0182019 1.2127173 1.0931231 1.1394878 1.2111272 1.0937192  
## [463] 1.0351007 1.1212283 1.0804250 0.9707126 1.1307119 1.0351570 1.0004664  
## [470] 1.0686864 1.0221388 1.1751058 1.0363847 0.9213066 1.1396839 1.0711568  
## [477] 0.8320639 1.0311364 1.0491481 0.9814310 1.0401747 1.0091426 1.0354748  
## [484] 0.9811130 1.1336886 0.9943882 1.0508727 1.0253745 1.1427029 1.1148785  
## [491] 1.1720707 1.1383365 1.0713640 1.0092040 0.9336815 0.9235910 1.0625834  
## [498] 1.0679184 1.1186236 1.1118536

apply(fin4[,2:51], 2, mean)

## V2 V3 V4 V5 V6   
## 0.2811463705 -0.0122708824 -0.0652210250 0.0569017763 0.0502177919   
## V7 V8 V9 V10 V11   
## 0.0234537238 0.0106091236 -0.0306502724 0.0809840510 0.0496538714   
## V12 V13 V14 V15 V16   
## -0.0373899135 -0.0167574439 -0.0001811560 0.0223851556 -0.0488549805   
## V17 V18 V19 V20 V21   
## -0.0127520222 -0.0402137158 0.0312824057 -0.0395657628 -0.0206464142   
## V22 V23 V24 V25 V26   
## 0.0022973271 0.0007980717 -0.0126960040 0.0213477752 0.0285090482   
## V27 V28 V29 V30 V31   
## 0.0057350303 0.0940303706 -0.0158823942 -0.0556392061 0.0035745996   
## V32 V33 V34 V35 V36   
## -0.0111967045 0.0033150179 -0.0278416960 -0.0493557822 0.0044023595   
## V37 V38 V39 V40 V41   
## 0.0007829558 -0.0175727190 -0.0473393429 0.0172157032 0.0616220233   
## V42 V43 V44 V45 V46   
## 0.0096335674 -0.0071118014 -0.0475064735 0.0610169950 -0.0016780125   
## V47 V48 V49 V50 V51   
## 0.1148059077 0.0373286311 -0.0253855536 -0.0130126981 0.0126579516

apply(fin4[,2:51], 2, sd)

## V2 V3 V4 V5 V6 V7 V8   
## 1.7952222 1.7918847 0.9893549 0.9769410 0.9872264 1.0138380 1.0345095   
## V9 V10 V11 V12 V13 V14 V15   
## 1.1021502 1.0143069 1.0391242 1.0166690 0.9671766 0.9734386 1.0339972   
## V16 V17 V18 V19 V20 V21 V22   
## 0.9978085 1.0125605 0.9883655 0.9122094 1.0016286 0.9469817 1.0361489   
## V23 V24 V25 V26 V27 V28 V29   
## 1.0118095 1.0288463 0.9614021 1.0332780 0.9778156 0.9756520 1.0099502   
## V30 V31 V32 V33 V34 V35 V36   
## 0.9856237 0.9698249 1.0065411 1.0157012 1.0202482 1.0604705 0.9426782   
## V37 V38 V39 V40 V41 V42 V43   
## 0.9959802 1.0257821 0.9829531 0.9759667 1.0541386 0.9497995 0.9572732   
## V44 V45 V46 V47 V48 V49 V50   
## 1.0328617 0.9726076 0.9569821 1.0016979 1.0497450 1.0253434 0.9706059   
## V51   
## 1.0038525

분석에 앞서 데이터 형태에 대한 의문이 생겨 행과 열에 따라 평균과 표준편차를 구했습니다. 전체적으로 평균은 0, 표준편차는 1에 근접했습니다. 저는 이 데이터가 rnorm(0,1)로 만들어졌다는 결론을 내렸습니다. V2부터 V51까지 랜덤으로 만들어진 변수이기 때문에 군집화 과정에 더 중요하거나 덜 중요한 변수는 없다고 판단했고, 모든 변수를 단순하면서도 강력한 K-means 방법으로 군집화하기로 했습니다.

k\_clust <- kmeans(fin4[,2:51], 2, nstart = 20); k\_clust

## K-means clustering with 2 clusters of sizes 272, 228  
##   
## Cluster means:  
## V2 V3 V4 V5 V6 V7  
## 1 1.363546 1.012321 -0.11049361 0.05381766 0.0930607456 0.11862823  
## 2 -1.010137 -1.234591 -0.01121163 0.06058107 -0.0008931001 -0.09008779  
## V8 V9 V10 V11 V12 V13  
## 1 -0.02500781 0.02141346 0.04006784 -0.06682739 -0.003726322 -0.01490503  
## 2 0.05309950 -0.09276139 0.12979637 0.18861398 -0.077549987 -0.01896734  
## V14 V15 V16 V17 V18 V19  
## 1 -0.04144724 -0.03283904 -0.02896781 -0.04956400 -0.01485044 0.03870763  
## 2 0.04904856 0.08826665 -0.07258003 0.03116402 -0.07047166 0.02242424  
## V20 V21 V22 V23 V24 V25  
## 1 -0.009121602 0.04500064 0.01705302 0.0282749 0.01559593 0.05287455  
## 2 -0.075885112 -0.09896220 -0.01530596 -0.0319813 -0.04644778 -0.01626312  
## V26 V27 V28 V29 V30 V31  
## 1 -0.00122659 -0.04573086 0.03777327 -0.021731335 -0.002210502 -0.04789524  
## 2 0.06398314 0.06713293 0.16114410 -0.008904711 -0.119378713 0.06497721  
## V32 V33 V34 V35 V36 V37  
## 1 0.01205472 0.04304904 -0.01298250 -0.11767087 0.003339455 -0.01735875  
## 2 -0.03893524 -0.04408697 -0.04556846 0.03214292 0.005670386 0.02242569  
## V38 V39 V40 V41 V42 V43  
## 1 -0.037474827 0.003891927 0.05589448 0.07632343 0.00144782 0.04171575  
## 2 0.006170147 -0.108457349 -0.02892740 0.04408351 0.01939902 -0.06536221  
## V44 V45 V46 V47 V48 V49  
## 1 -0.084518468 0.08491851 -0.04053401 0.23724827 -0.03068774 0.02485248  
## 2 -0.003351813 0.03250290 0.04467652 -0.03126568 0.11847096 -0.08531865  
## V50 V51  
## 1 -0.006235909 0.029645385  
## 2 -0.021097289 -0.007607759  
##   
## Clustering vector:  
## [1] 2 2 2 2 2 2 2 2 1 1 2 1 2 2 1 2 1 1 1 1 2 1 2 1 2 2 1 2 2 2 1 2 1 1 1  
## [36] 1 2 2 2 1 2 2 2 1 2 1 2 1 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2 2 2 1 1 2 2  
## [71] 2 2 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 1 1 1 2 2 1 2 2 1 2 2 1 2 1 1 1 1 1  
## [106] 1 1 1 1 1 2 1 1 1 2 2 2 1 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 2 1 2 1 1 1 1  
## [141] 2 2 1 2 1 2 1 1 1 2 1 2 2 2 2 2 2 2 1 1 2 1 1 1 1 2 2 2 2 2 1 2 1 1 2  
## [176] 2 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 2 2 1 1 2 1 1 2 1 1 2 2 1 1 1 1 2 2 1  
## [211] 1 2 1 1 1 1 1 2 1 1 1 2 2 2 1 2 1 2 2 1 1 2 2 1 1 2 2 1 2 2 2 2 2 2 2  
## [246] 2 2 2 1 1 2 1 2 1 1 2 2 1 2 2 2 1 2 2 1 1 2 1 1 2 1 2 2 1 2 1 1 1 1 2  
## [281] 2 1 1 2 1 2 2 2 1 1 1 2 1 2 1 2 2 2 1 2 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2  
## [316] 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1  
## [351] 1 2 2 2 2 2 1 1 2 1 1 1 2 1 1 2 1 1 2 1 2 2 1 1 1 1 2 1 2 2 1 1 2 1 2  
## [386] 1 1 2 2 1 1 1 1 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 1 2 2 1 2 1 2 2 2 1 2 1  
## [421] 1 2 2 1 1 1 1 2 2 2 1 2 2 1 1 1 1 1 1 1 1 2 2 1 2 2 2 1 1 1 2 1 1 1 1  
## [456] 2 2 1 2 2 2 2 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 2 2 1 1 1 2 1 1 2 1  
## [491] 1 1 2 2 2 1 1 1 1 1  
##   
## Within cluster sum of squares by cluster:  
## [1] 14086.86 11719.81  
## (between\_SS / total\_SS = 5.1 %)  
##   
## Available components:  
##   
## [1] "cluster" "centers" "totss" "withinss"   
## [5] "tot.withinss" "betweenss" "size" "iter"   
## [9] "ifault"

k\_clust$cluster

## [1] 2 2 2 2 2 2 2 2 1 1 2 1 2 2 1 2 1 1 1 1 2 1 2 1 2 2 1 2 2 2 1 2 1 1 1  
## [36] 1 2 2 2 1 2 2 2 1 2 1 2 1 1 2 2 2 1 1 1 2 1 1 2 1 1 1 2 2 2 2 1 1 2 2  
## [71] 2 2 1 1 1 2 1 1 1 1 2 1 1 1 2 2 2 1 1 1 2 2 1 2 2 1 2 2 1 2 1 1 1 1 1  
## [106] 1 1 1 1 1 2 1 1 1 2 2 2 1 2 2 2 1 1 1 1 2 2 1 1 1 2 2 2 2 1 2 1 1 1 1  
## [141] 2 2 1 2 1 2 1 1 1 2 1 2 2 2 2 2 2 2 1 1 2 1 1 1 1 2 2 2 2 2 1 2 1 1 2  
## [176] 2 1 1 1 2 1 1 2 1 1 1 2 1 2 1 1 2 2 1 1 2 1 1 2 1 1 2 2 1 1 1 1 2 2 1  
## [211] 1 2 1 1 1 1 1 2 1 1 1 2 2 2 1 2 1 2 2 1 1 2 2 1 1 2 2 1 2 2 2 2 2 2 2  
## [246] 2 2 2 1 1 2 1 2 1 1 2 2 1 2 2 2 1 2 2 1 1 2 1 1 2 1 2 2 1 2 1 1 1 1 2  
## [281] 2 1 1 2 1 2 2 2 1 1 1 2 1 2 1 2 2 2 1 2 1 1 1 1 2 2 2 2 2 2 2 2 1 2 2  
## [316] 1 1 2 1 2 2 1 2 1 1 1 1 1 1 1 2 1 2 2 1 1 2 1 1 2 1 1 1 1 2 1 1 1 1 1  
## [351] 1 2 2 2 2 2 1 1 2 1 1 1 2 1 1 2 1 1 2 1 2 2 1 1 1 1 2 1 2 2 1 1 2 1 2  
## [386] 1 1 2 2 1 1 1 1 1 1 1 2 1 2 1 1 1 2 2 1 1 1 1 1 2 2 1 2 1 2 2 2 1 2 1  
## [421] 1 2 2 1 1 1 1 2 2 2 1 2 2 1 1 1 1 1 1 1 1 2 2 1 2 2 2 1 1 1 2 1 1 1 1  
## [456] 2 2 1 2 2 2 2 1 2 1 1 1 2 1 2 1 1 2 1 1 1 1 2 1 1 2 2 1 1 1 2 1 1 2 1  
## [491] 1 1 2 2 2 1 1 1 1 1

## 4-2

n\_fin4 <- fin4[sample(1:500),] # shuffle  
  
ten\_tables2 <- matrix(0, 2, 2)  
colnames(ten\_tables2) <- paste0("pred", 0:1)  
rownames(ten\_tables2) <- paste0("real", 0:1)  
ten\_tables2 <- as.table(ten\_tables2)  
for (i in 1:5){  
 idx <- ((i-1)\*100+1):(i\*100)  
 test\_set <- n\_fin4[idx,]  
 random01 <- sample(0:1, 100, replace = TRUE)  
 itable <- table(test\_set$V1, random01)  
 ten\_tables2 <- ten\_tables2 + as.table(itable)  
}  
ten\_tables2

## pred0 pred1  
## real0 116 134  
## real1 129 121

corrected <- sum(diag(ten\_tables2))  
corrected/sum(ten\_tables2)

## [1] 0.474

랜덤워크라는 생각을 가지고 있기 때문에 복잡한 방법을 쓰기보다 똑같이 랜덤하게 예측하는 방법을 선택했습니다.

# 5

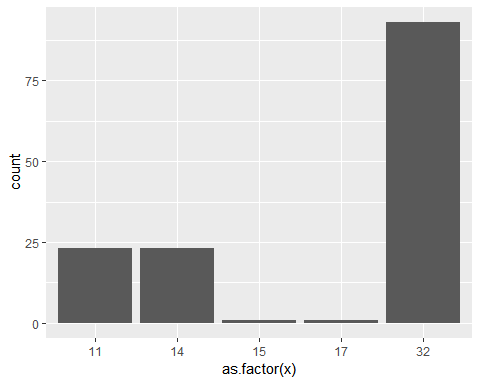
이 데이터를 통한 군집화는 기존에 나와있는 모델을 쓰기보다 제가 가정한 논리에 기반해 직접 군집화하는 것이 더 효과적이라 생각했습니다. 논리는 다음과 같습니다.

* 발의한 법안 수가 많은[적은] 의원 순으로 진행한다.
* 각 의원이 발의한 법안 중 n/k개 이상을 같이했다면 같은 군집에 속한다.
* 속한 군집이 없다면 스스로 군집을 열게 한다.

이 방법을 적용할 때 분석자가 변경할 수 있는 것은 발의한 법안 수에 따라 오름차순으로 정렬할지, 내림차순으로 정렬할지와 같은 군집에 속하는 것으로 묶을 때 각 의원이 발의한 전체 법안 n개 중 n/k를 몇으로 할지, 즉 k 값을 얼마로 할지입니다.

이 때 오름차순으로 하면 비교적 많은 군집이 생기고, 내림차순으로 하면 비교적 적은 군집이 생깁니다. 이는 오름차순으로 할 때 초반에 연결되는 관계 수가 적어 군집이 많이 만들어지고, 내림차순으로 할 때는 초반에 연결되는 관계 수가 많아 군집이 적게 만들어지기 때문입니다. 또한 k값을 얼마로 하는지도 중요한데, k를 작게 할수록 많은 군집이 생기고, k를 크게 할수록 적은 군집이 생긴다. 시행착오를 통해 4~6 정도의 k값이 분석 가능한 수준의 k값임을 확인했습니다.

group\_poli <- rep(0, 141)  
influence\_order <- order(diag(as.matrix(poli\_rel)), decreasing = TRUE)  
j <- 0  
for (i in influence\_order){  
 if(group\_poli[i] == 0){  
 j <- j+1  
 k <- j  
 } else{  
 k <- group\_poli[i]  
 }  
 ith\_poli<- poli\_rel[i,]  
 refer <- poli\_rel[i,i]  
 related <- which(ith\_poli >= (refer/5))  
 group\_poli[related] <- k  
}  
  
ggplot(data.frame(x=group\_poli)) + geom\_bar(aes(as.factor(x)))



poli\_party\_clust <- bind\_cols(poli\_party, clust=group\_poli)  
poli\_party\_clust

## id party clust  
## 1 legi1 바른정당 14  
## 2 legi2 더불어민주당 32  
## 3 legi5 더불어민주당 32  
## 4 legi8 자유한국당 14  
## 5 legi9 더불어민주당 32  
## 6 legi11 자유한국당 11  
## 7 legi12 더불어민주당 32  
## 8 legi13 자유한국당 14  
## 9 legi15 국민의당 32  
## 10 legi16 더불어민주당 32  
## 11 legi17 더불어민주당 32  
## 12 legi18 더불어민주당 32  
## 13 legi19 더불어민주당 32  
## 14 legi20 국민의당 32  
## 15 legi23 자유한국당 11  
## 16 legi24 국민의당 32  
## 17 legi26 자유한국당 11  
## 18 legi27 자유한국당 14  
## 19 legi30 자유한국당 11  
## 20 legi31 바른정당 14  
## 21 legi33 더불어민주당 32  
## 22 legi35 더불어민주당 32  
## 23 legi36 더불어민주당 32  
## 24 legi37 국민의당 32  
## 25 legi38 자유한국당 11  
## 26 legi40 자유한국당 11  
## 27 legi41 자유한국당 14  
## 28 legi42 더불어민주당 32  
## 29 legi43 국민의당 32  
## 30 legi46 자유한국당 14  
## 31 legi50 자유한국당 11  
## 32 legi51 자유한국당 14  
## 33 legi57 바른정당 14  
## 34 legi60 더불어민주당 32  
## 35 legi62 자유한국당 11  
## 36 legi64 더불어민주당 32  
## 37 legi67 자유한국당 11  
## 38 legi71 자유한국당 11  
## 39 legi72 더불어민주당 32  
## 40 legi73 더불어민주당 32  
## 41 legi74 더불어민주당 32  
## 42 legi79 더불어민주당 32  
## 43 legi80 더불어민주당 32  
## 44 legi81 더불어민주당 32  
## 45 legi83 자유한국당 14  
## 46 legi84 더불어민주당 32  
## 47 legi85 더불어민주당 32  
## 48 legi87 더불어민주당 32  
## 49 legi89 자유한국당 11  
## 50 legi92 더불어민주당 32  
## 51 legi94 더불어민주당 32  
## 52 legi95 더불어민주당 32  
## 53 legi101 더불어민주당 32  
## 54 legi106 더불어민주당 32  
## 55 legi107 자유한국당 14  
## 56 legi111 더불어민주당 32  
## 57 legi112 더불어민주당 32  
## 58 legi113 더불어민주당 32  
## 59 legi114 국민의당 32  
## 60 legi116 국민의당 32  
## 61 legi118 더불어민주당 32  
## 62 legi119 자유한국당 11  
## 63 legi120 더불어민주당 32  
## 64 legi126 무소속 32  
## 65 legi128 더불어민주당 32  
## 66 legi131 더불어민주당 32  
## 67 legi132 국민의당 32  
## 68 legi133 더불어민주당 32  
## 69 legi134 국민의당 32  
## 70 legi135 더불어민주당 32  
## 71 legi137 더불어민주당 32  
## 72 legi139 자유한국당 11  
## 73 legi141 더불어민주당 32  
## 74 legi144 국민의당 32  
## 75 legi145 더불어민주당 32  
## 76 legi147 정의당 32  
## 77 legi149 자유한국당 11  
## 78 legi150 더불어민주당 32  
## 79 legi151 더불어민주당 32  
## 80 legi152 자유한국당 17  
## 81 legi153 국민의당 32  
## 82 legi155 더불어민주당 32  
## 83 legi156 더불어민주당 32  
## 84 legi158 자유한국당 14  
## 85 legi159 자유한국당 11  
## 86 legi160 국민의당 32  
## 87 legi161 바른정당 14  
## 88 legi168 더불어민주당 32  
## 89 legi171 자유한국당 11  
## 90 legi175 더불어민주당 32  
## 91 legi177 자유한국당 14  
## 92 legi179 자유한국당 14  
## 93 legi180 자유한국당 15  
## 94 legi183 국민의당 32  
## 95 legi186 자유한국당 11  
## 96 legi188 더불어민주당 32  
## 97 legi189 더불어민주당 32  
## 98 legi190 더불어민주당 32  
## 99 legi192 국민의당 32  
## 100 legi195 국민의당 32  
## 101 legi196 더불어민주당 32  
## 102 legi199 국민의당 32  
## 103 legi200 자유한국당 11  
## 104 legi203 국민의당 32  
## 105 legi206 자유한국당 11  
## 106 legi207 자유한국당 14  
## 107 legi209 자유한국당 11  
## 108 legi210 더불어민주당 32  
## 109 legi213 더불어민주당 32  
## 110 legi214 바른정당 14  
## 111 legi219 국민의당 32  
## 112 legi220 자유한국당 11  
## 113 legi223 더불어민주당 32  
## 114 legi225 국민의당 32  
## 115 legi227 바른정당 14  
## 116 legi228 더불어민주당 32  
## 117 legi229 자유한국당 14  
## 118 legi233 더불어민주당 32  
## 119 legi235 더불어민주당 32  
## 120 legi236 국민의당 32  
## 121 legi239 자유한국당 14  
## 122 legi241 더불어민주당 32  
## 123 legi242 더불어민주당 32  
## 124 legi244 자유한국당 14  
## 125 legi246 국민의당 32  
## 126 legi248 더불어민주당 32  
## 127 legi256 더불어민주당 32  
## 128 legi259 더불어민주당 32  
## 129 legi260 자유한국당 14  
## 130 legi261 더불어민주당 32  
## 131 legi266 더불어민주당 32  
## 132 legi267 더불어민주당 32  
## 133 legi269 자유한국당 11  
## 134 legi271 바른정당 14  
## 135 legi274 더불어민주당 32  
## 136 legi278 국민의당 32  
## 137 legi280 국민의당 32  
## 138 legi284 더불어민주당 32  
## 139 legi285 자유한국당 11  
## 140 legi288 더불어민주당 32  
## 141 legi301 국민의당 32

최종적으로 참고자료 차원으로 있던 party 변수를 이용해 군집화가 잘 되었는지 검토해봤는데, 이 때 재밌는 점을 몇 개 찾을 수 있었습니다. (k와 차순을 바꿔가며 만든 표를 직접 보며 파악한 내용이기 때문에 시각화하거나 테이블을 보여주기엔 분량이 많아 생략했습니다. 직접하시고자 하는 경우 decreasing 부분을 T or F 조절하고, refer/다음에 있는 수를 조절하여 확인할 수 있습니다.)

* 정의당과 무소속은 각각 1명으로 항상 더불어민주당의 대세 군집과 같은 군집에 속한다.
* 더불어민주당은 군집을 아주 많이 만들지 않는 이상 거의 모두가 같은 군집에 속한다.
* 자유한국당은 전체의 군집이 3개 이상일 때부터는 항상 자유한국당 내부에서 군집이 갈리고, 군집이 많을수록 더 다양한 군집이 자유한국당 내부에 있음을 확인할 수 있다. [두 군집일 때 각 군집에 해당하는 자유한국당원 수[세력]가 비슷하다.]
* 국민의당은 군집이 아주 많지 않으면 더불어민주당과 같은 군집에 속한다. 군집이 많아져도 여러 세력으로 잘 갈리지 않는 더불어민주당에 비해 국민의당은 군집이 많아지면 비슷한 규모의 두 세력으로 나뉜다.
* 바른정당은 군집이 아주 많지 않을 때는 자유한국당의 한 세력과 항상 같은 군집에 속하고, 군집이 일정 수준 이상 많아지면 legi1만 혼자 다른 의견을 낸다는 것을 알 수 있다. 이를 통해 바른정당에는 한 명이 다른 생각을 품는다는 것과 자유한국당의 한 세력이 바른정당과 긴밀한 관계라는 것을 유추할 수 있다.