# R Notebook

## 1

• import customers_data.csv as a data frame with header and make a summary of its variables. (5 points) • extract the variables FRESH and FROZEN and store them in a separate data frame, called customers_2. (5 points) • from this new data frame, provide a scatter-plot matrix of FRESH and FROZEN via the function ggpairs from the package GGally. (5 points)

```
library(GGally)
```

```
## Loading required package: ggplot2
```

```
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
```

```
setwd("/Users/quilviohernandez/Desktop")

#import the data
#1
customer_data <- read.csv("customers_data.csv")
head(customer_data)
```

```
##   Channel Region Fresh Milk Grocery Frozen Detergents_Paper Delicassen
## 1       2      3 12669 9656    7561    214             2674       1338
## 2       2      3  7057 9810    9568   1762             3293       1776
## 3       2      3  6353 8808    7684   2405             3516       7844
## 4       1      3 13265 1196    4221   6404              507       1788
## 5       2      3 22615 5410    7198   3915             1777       5185
## 6       2      3  9413 8259    5126    666             1795       1451
```

```
summary(customer_data)
```

```
##     Channel          Region          Fresh             Milk
##  Min.   :1.000   Min.   :1.000   Min.   :     3   Min.   :   55
##  1st Qu.:1.000   1st Qu.:2.000   1st Qu.:  3128   1st Qu.: 1533
##  Median :1.000   Median :3.000   Median :  8504   Median : 3627
##  Mean   :1.323   Mean   :2.543   Mean   : 12000   Mean   : 5796
##  3rd Qu.:2.000   3rd Qu.:3.000   3rd Qu.: 16934   3rd Qu.: 7190
##  Max.   :2.000   Max.   :3.000   Max.   :112151   Max.   :73498
##     Grocery          Frozen        Detergents_Paper    Delicassen
##  Min.   :    3   Min.   :   25.0   Min.   :    3.0   Min.   :    3.0
##  1st Qu.: 2153   1st Qu.:  742.2   1st Qu.:  256.8   1st Qu.:  408.2
##  Median : 4756   Median : 1526.0   Median :  816.5   Median :  965.5
##  Mean   : 7951   Mean   : 3071.9   Mean   : 2881.5   Mean   : 1524.9
##  3rd Qu.:10656   3rd Qu.: 3554.2   3rd Qu.: 3922.0   3rd Qu.: 1820.2
##  Max.   :92780   Max.   :60869.0   Max.   :40827.0   Max.   :47943.0
```

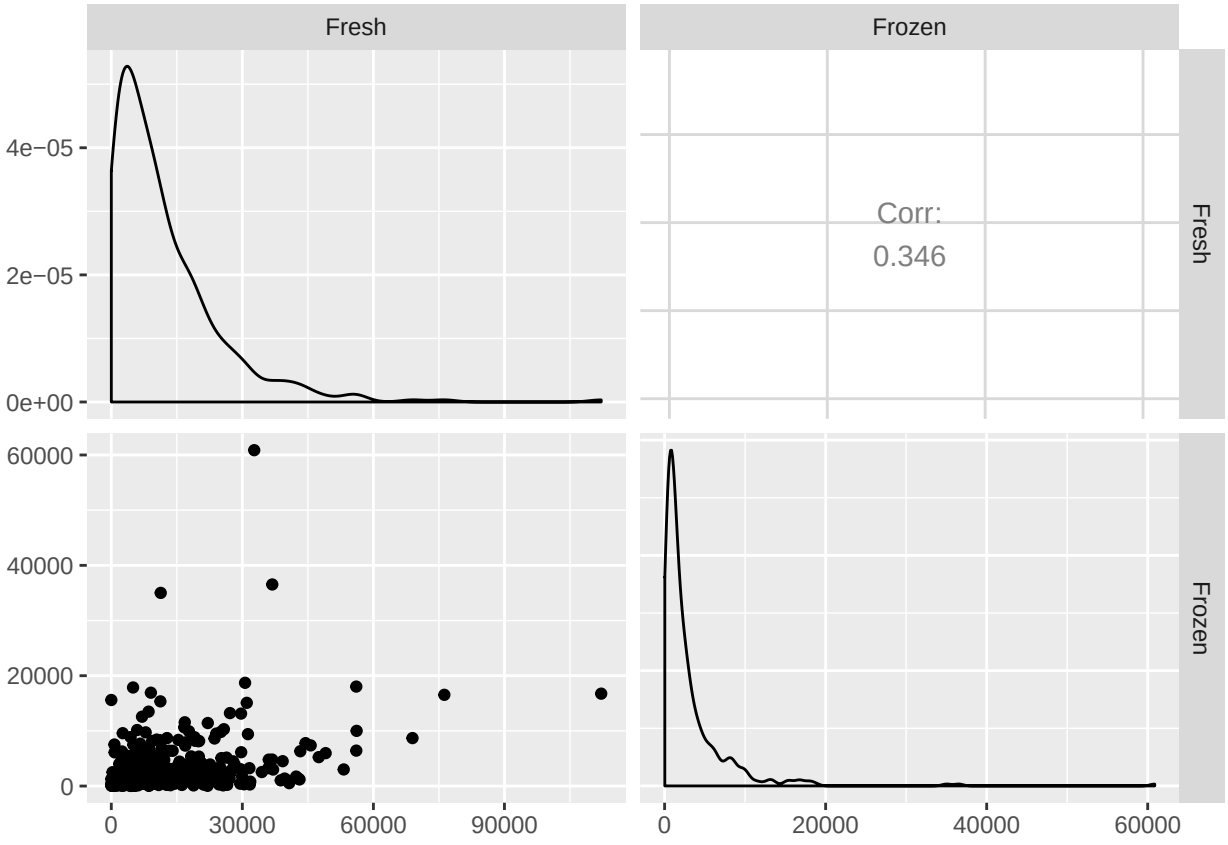
```
customers_2 <- customer_data[,c(3,6)]
customers_2 <- na.omit(customers_2)
names(customers_2)
```

```
## [1] "Fresh"  "Frozen"
```

```
dim(customers_2)
```

```
## [1] 440   2
```

```
ggpairs(customers_2)
```




## 2

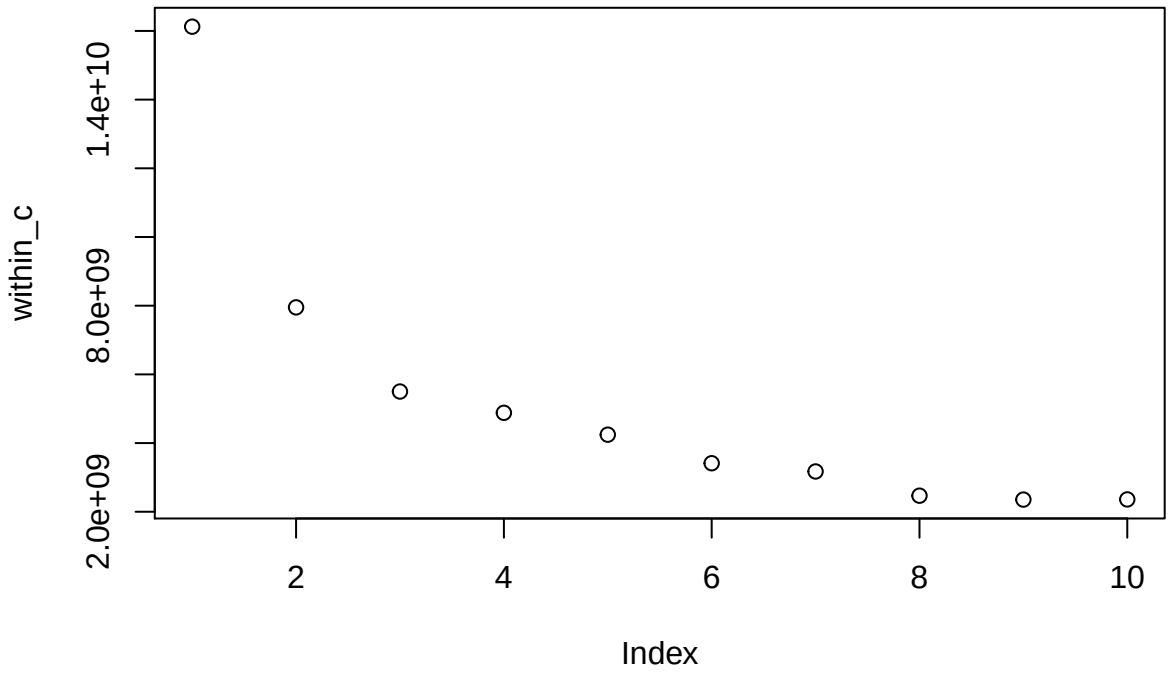
• Estimatea k-means clustering partitionon FRESH and FROZEN.For k=1,. . . ,10, run100 times the following procedure: (5 points) • draw randomly the 80% of observations in customer_2 and use them as a training data-set. The remaining observations will constitute the test data-set. (5 points) • run the function kmeans on the training data-set with k centers, 20 random starts and 100 maximum iterations. (5 points) • use the estimated centers to allocate the observations of the test data-set to a specific group and derive the relative vector of assignments. (10 points) • calculate the deviance within estimated groups in the test data-set. (10 points)

```
withinss <- function(group, x, centers, assignments) {
  cent <- centers[group, ]
  m <- rbind(cent, x[assignments==group, 1:2])
  sum((as.matrix(dist(m))[1, ])^2)
}
```

```
predict.kmeans <- function(object,
                           newdata,
                           method = c("centers", "classes")) {
  method <- match.arg(method)

  centers <- object$centers
  ss_by_center <- apply(centers, 1, function(x) {
    colSums((t(newdata) - x) ^ 2)
  })
  best_clusters <- apply(ss_by_center, 1, which.min)

  if (method == "centers") {
    centers[best_clusters, ]
  } else {
    best_clusters
  }
}

set.seed(12)
train_p=0.8
n=dim(customers_2)[1]
N=100
n_clust=10
within_c_n=matrix(0,n_clust,N)
within_c=vector('numeric',0)
dfxy=customers_2



for(cl in 1:n_clust){
  for(iter in 1:N){
    train=sample(1:n,train_p*n)
    train=sort(train)
    test=sort(setdiff(1:n,train))
    dfxy.km_train=kmeans(dfxy[train,1:2],centers=cl, nstart = 20, iter.max = 100)
    centers <- dfxy.km_train$centers
    assignments <-  as.numeric(row.names(predict.kmeans(dfxy.km_train, (dfxy[test,1:2]))))
    #assignments <- predict.kmeans(dfxy.km_train, (dfxy[test,1:2]),method = "classes")
    within_c_n[cl,iter]=sum(sapply(seq(nrow(centers)), function(y){withinss(group=y,x = dfxy[test,1:2],
  }
  within_c[cl]=(mean(within_c_n[cl,]))
}
within_c
```

```
##  [1] 16123668468  7951142311  5500510756  4880624956  4244403334  3412735472
##  [7]  3173091650  2467738735  2354705337  2358766521
```

```
par(mfrow=c(1,1))
plot(within_c)
```

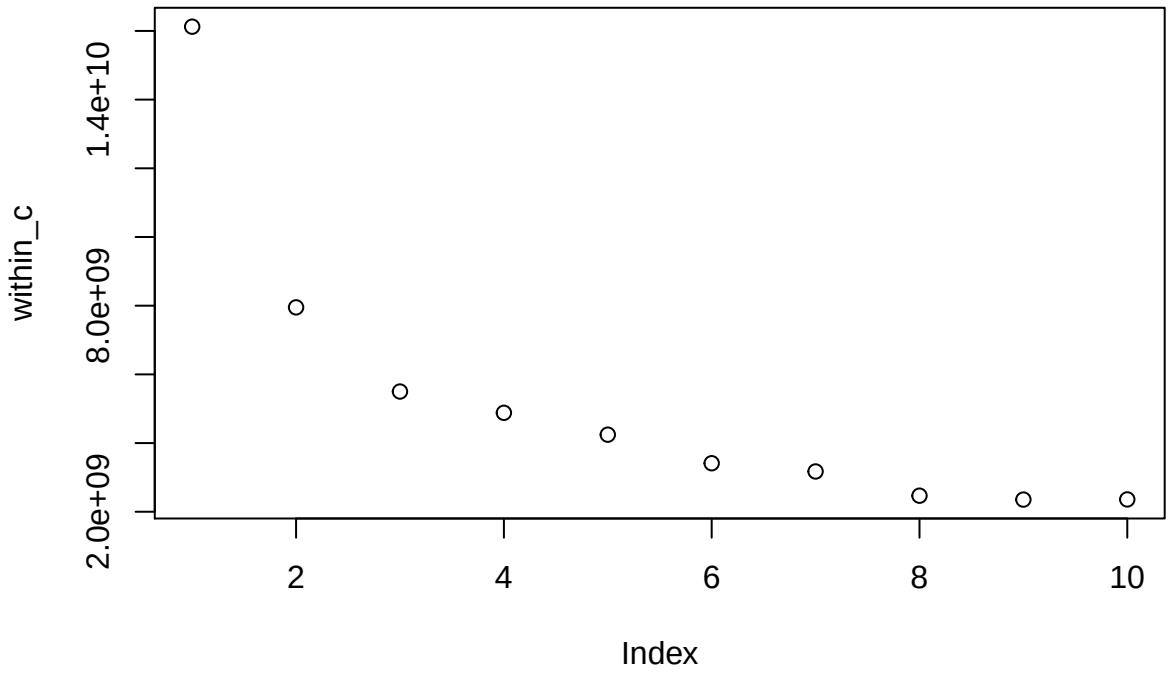



```
centers
```

```
##        Fresh     Frozen
## 1  28787.870  2141.0870
## 2  27080.000 12071.8750
## 3  26959.333 44137.3333
## 4  94194.000 16641.5000
## 5  11073.750 10236.0625
## 6   5823.000  2307.3671
## 7  46304.727  5527.7273
## 8  11020.297  1672.3438
## 9  18721.462  2551.1731
## 10  1633.096   999.1702
```

• Then, for each k, average the deviance within groups over the 100 runs. (5 points) Finally, • plot this average over the number of clusters and decide the optimal number of clusters using the elbow criterion. (5 points) • re-apply kmeans with the selected number of clusters, 20 random starts and 100 max- imum iterations, and derive the estimated cluster memberships. (5 points) • provide a scatter-plot matrix of FRESH and FROZEN conditional on the estimated clus- ter memberships via the function ggpairs from the package GGally and comment about the shape of FRESH and FROZEN over groups. (5+5 points)

```
plot(within_c)
```




```
dfxy.km = kmeans(dfxy[,1:2],centers=2, nstart = 20, iter.max = 100)
dfxy.km$cluster
```

```
##   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18  19  20
##   2   2   2   2   1   2   2   2   2   2   2   2   1   1   1   2   2   2   2   2
##  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40
##   2   2   1   1   1   2   2   2   2   1   2   2   1   1   2   2   1   2   2   1
##  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60
##   1   2   2   2   2   2   2   1   2   2   2   2   1   2   1   2   2   2   2   2
##  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80
##   2   1   2   2   2   2   2   2   2   2   2   2   2   1   2   1   2   2   2   2
##  81  82  83  84  85  86  87  88  89  90  91  92  93  94  95  96  97  98  99 100
##   2   2   2   1   2   2   1   1   2   1   2   2   2   2   2   2   2   2   2   2
## 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120
##   2   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2
## 121 122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140
##   2   2   2   2   1   1   2   1   2   1   2   2   2   2   2   2   2   2   2   2
## 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160
##   2   1   1   2   2   1   2   2   2   1   2   2   2   2   2   2   2   2   2   2
## 161 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180
##   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2
## 181 182 183 184 185 186 187 188 189 190 191 192 193 194 195 196 197 198 199 200
##   2   1   2   1   2   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2
## 201 202 203 204 205 206 207 208 209 210 211 212 213 214 215 216 217 218 219 220
##   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2   2
## 221 222 223 224 225 226 227 228 229 230 231 232 233 234 235 236 237 238 239 240
##   2   2   2   2   2   2   1   2   2   2   2   2   1   2   2   2   2   2   2   1
## 241 242 243 244 245 246 247 248 249 250 251 252 253 254 255 256 257 258 259 260
##   1   1   2   2   2   2   2   2   2   2   2   2   2   1   2   1   2   2   1   1
## 261 262 263 264 265 266 267 268 269 270 271 272 273 274 275 276 277 278 279 280
##   2   2   1   2   2   2   2   1   2   2   2   2   2   1   2   2   1   2   2   2
## 281 282 283 284 285 286 287 288 289 290 291 292 293 294 295 296 297 298 299 300
##   2   2   1   1   1   1   2   2   2   1   2   2   2   2   1   2   2   2   2   2
```

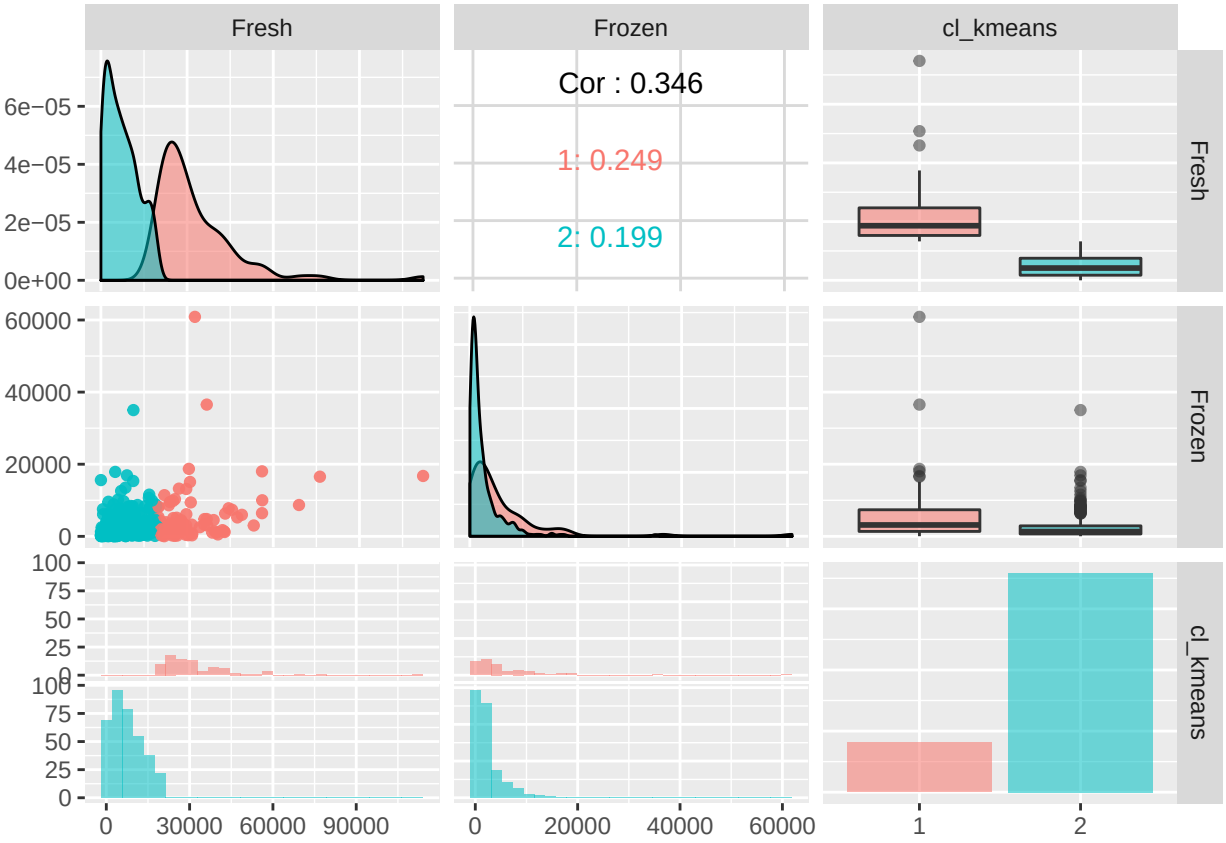
```
## 301 302 303 304 305 306 307 308 309 310 311 312 313 314 315 316 317 318 319 320
##   2   2   2   2   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   2
## 321 322 323 324 325 326 327 328 329 330 331 332 333 334 335 336 337 338 339 340
##   2   2   2   2   1   1   2   2   2   2   2   2   1   2   2   1   2   2   2   2
## 341 342 343 344 345 346 347 348 349 350 351 352 353 354 355 356 357 358 359 360
##   2   2   2   2   2   2   2   1   2   2   2   2   2   2   2   2   1   2   2   2
## 361 362 363 364 365 366 367 368 369 370 371 372 373 374 375 376 377 378 379 380
##   2   2   2   2   2   2   2   2   1   2   1   1   2   2   2   2   2   1   2   2
## 381 382 383 384 385 386 387 388 389 390 391 392 393 394 395 396 397 398 399 400
##   1   2   1   2   2   2   2   1   2   2   2   2   2   1   2   2   2   2   2   2
## 401 402 403 404 405 406 407 408 409 410 411 412 413 414 415 416 417 418 419 420
##   2   1   1   1   2   2   1   2   2   2   2   2   2   2   2   2   2   2   2   2
## 421 422 423 424 425 426 427 428 429 430 431 432 433 434 435 436 437 438 439 440
##   2   2   1   2   2   2   2   1   2   2   2   2   1   2   2   1   1   2   2   2
```

```
dfxy.km$size
```

```
## [1]  81 359
```

```
customers_2$cl_kmeans=as.factor(dfxy.km$cluster)

ggpairs(customers_2, 1:3, mapping = ggplot2::aes(color = cl_kmeans, alpha = 0.5),
        diag = list(continuous = wrap("densityDiag")),
        lower=list(continuous = wrap("points", alpha=0.9)))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




Estimate a hierachical partition on FRESH and FROZEN by the complete linkage using the number of

clusters selected above. (5 points) • Derive the estimated cluster memberships. (5 points) • Provide a scatter-plot matrix of FRESH and FROZEN conditional on the estimated clus- ter memberships via the function ggpairs from the package GGally.(5 points) • Comment about the shape of FRESH and FROZEN over the estimated groups and com- pare this outcome to the k-means one. (5+5 points)

```
ass_hclust=function(i,method='complete',test){
  dist_hclust=as.matrix(dist(rbind(dfxy[test,1:2],dfxy[assignment_train==i,1:2])))
  n_dist=length(test)+length(which(assignment_train==i))
  dist_hclust=as.matrix(dist_hclust)
  if(method=='complete'){
    return(sapply(1:length(test),function(j){max(dist_hclust[(length(test)+1):n_dist,j])}))}
  if(method=='single'){
    return(sapply(1:length(test),function(j){min(dist_hclust[(length(test)+1):n_dist,j])}))}
  if(method=='average'){
    return(sapply(1:length(test),function(j){mean(dist_hclust[(length(test)+1):n_dist,j])}))}
}

set.seed(12)
train_p=0.8
N=100
n_clust=20
within_c_n=matrix(0,n_clust,N)
within_c=vector('numeric',0)
dfxy=customers_2
summary(dfxy)
```

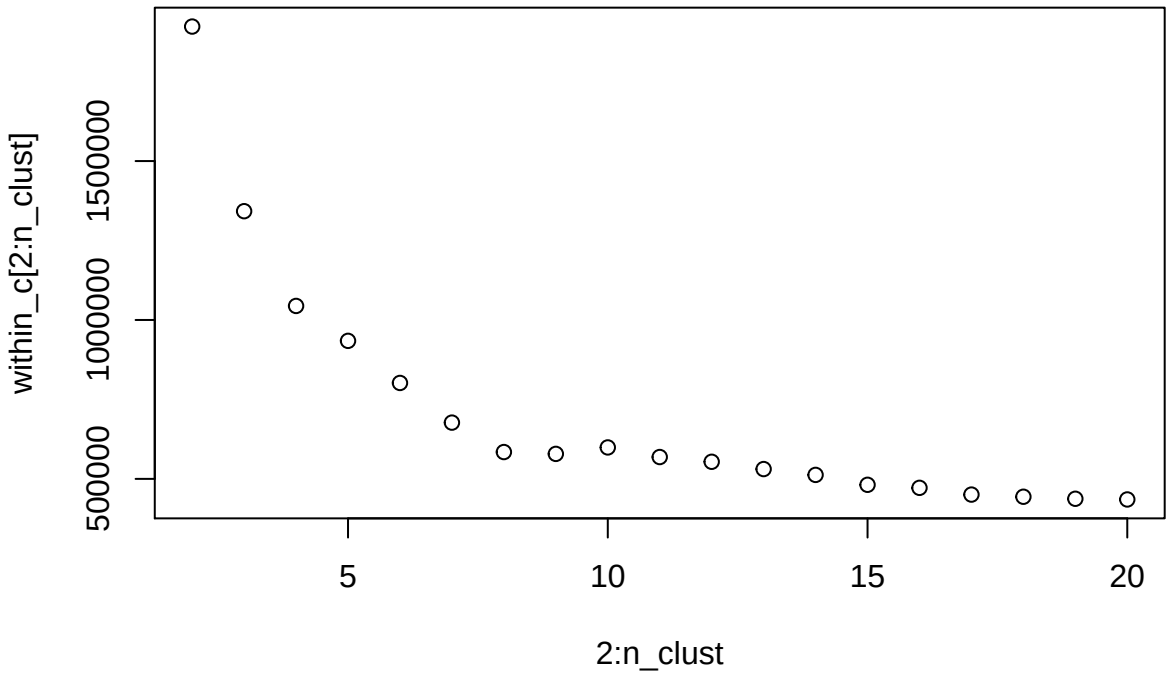
```
##      Fresh            Frozen        cl_kmeans
##  Min.   :     3   Min.   :   25.0   1: 81
##  1st Qu.:  3128   1st Qu.:  742.2   2:359
##  Median :  8504   Median : 1526.0
##  Mean   : 12000   Mean   : 3071.9
##  3rd Qu.: 16934   3rd Qu.: 3554.2
##  Max.   :112151   Max.   :60869.0
```

```
for(cl in 2:n_clust){
  for(iter in 1:N){
    train=sample(1:n,train_p*n)
    train=sort(train)
    test=sort(setdiff(1:n,train))
    n_train=length(train)
    test=sort(setdiff(1:n,train))
    n_test=length(test)
    within_c_n[1,iter]=var(dfxy[test,1])*(n_test-1)+var(dfxy[test,2])*(n_test-1)
    datamat = dfxy[train,1:2]
    dfxy.complete_train = hclust(dist(datamat), method = "complete")
    assignment_train=cutree(dfxy.complete_train , cl)

    #centers <- sapply(seq(cl),function(y){centers_compute(y,train=train,assignment_train=assignment_tr
    #colnames(centers)=as.numeric((n+1):(n+cl))
    #dist_center=dist(rbind(t(centers),dfxy[test,1:2]))
    #dist_center=(as.matrix(dist_center))
    #dist_true=dist_center[(cl+1):(n_test+cl),1:cl]
    #dist_hc=function(i){return(as.numeric(which(dist_true[i,]==min(dist_true[i,]))))}
    #assignments <- sapply(seq(n_test),dist_hc)
    clust_distance <- sapply(seq(cl),function(l){ass_hclust(l,method='complete',test=test)})
```

```
    #assignments=sapply(1:length(test),function(j){which(clust_distance[j,]==min(clust_distance[j,]))})
    within_c_n[cl,iter]=sum(sapply(1:length(test),function(j){min(clust_distance[j,])}))
    #sum(sapply(seq(nrow(centers)), function(y){withinss(n=y,x = dfxy[test,1:2], centers = centers, ass
  }
  within_c[cl]=(mean(within_c_n[cl,],na.rm=TRUE))
}
within_c
```

```
##  [1]        NA 1923145.1 1342238.5 1044112.3  934258.3  801788.3  677008.8
##  [8]  584358.7  578580.2  598878.1  568710.7  553786.0  530836.5  512513.8
## [15]  481341.9  471808.7  450568.3  443776.8  437437.3  435243.9
```

```
par(mfrow=c(1,1))
plot(2:n_clust,within_c[2:n_clust])
```




```
dfxy.complete = hclust(dist(customers_2[,1:2]), method = "complete")
assignment=cutree(dfxy.complete , 5)
table(assignment)
```

```
## assignment
##   1   2   3   4   5
## 417  17   3   2   1
```

```
customers_2$cl_complete=as.factor(assignment)

ggpairs(customers_2, c(1:2,4), mapping = ggplot2::aes(color = cl_complete, alpha = 0.5),
        diag = list(continuous = wrap("densityDiag")),
        lower=list(continuous = wrap("points", alpha=0.9)))
```

```
## Warning: Groups with fewer than two data points have been dropped.

## Warning: Groups with fewer than two data points have been dropped.

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```