Project

2023-11-25

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
df <- read.csv('train.csv')

# Identify indices of majority and minority classes
churn_indices <- which(df$Churn == 1)
no_churn_indices <- which(df$Churn == 0)

# Randomly undersample the majority class to match the size of the minority class
set.seed(123)  # for reproducibility
no_churn_sampled_indices <- sample(no_churn_indices, length(churn_indices))
df <- df[c(churn_indices, no_churn_sampled_indices), ]


# Shuffle the rows
df <- df[sample(nrow(df)), ]

stratified_sample <- df %>%
  group_by(Churn) %>%
  sample_n(3000)

# Replace the original dataframe with the sampled data
df <- data.frame(stratified_sample)
df_ni = df
df_main = df
```

```
str(df)
```

```
## 'data.frame':    6000 obs. of  21 variables:
##  $ AccountAge              : int  48 5 15 75 85 67 114 8 55 22 ...
##  $ MonthlyCharges          : num  19.82 19.71 5.22 15.8 14.05 ...
##  $ TotalCharges            : num  951.1 98.5 78.3 1185.3 1194.5 ...
##  $ SubscriptionType        : chr  "Basic" "Basic" "Standard" "Premium" ...
##  $ PaymentMethod           : chr  "Credit card" "Bank transfer" "Mailed check" "Ma
iled check" ...
##  $ PaperlessBilling        : chr  "No" "Yes" "No" "Yes" ...
##  $ ContentType             : chr  "TV Shows" "Movies" "TV Shows" "Both" ...
##  $ MultiDeviceAccess       : chr  "No" "Yes" "Yes" "No" ...
##  $ DeviceRegistered        : chr  "Mobile" "Computer" "Tablet" "Computer" ...
##  $ ViewingHoursPerWeek     : num  36 17.9 15.8 33.9 35.4 ...
##  $ AverageViewingDuration  : num  141.8 55 91 108.2 47.4 ...
##  $ ContentDownloadsPerMonth: int  13 43 12 2 47 10 17 10 38 45 ...
##  $ GenrePreference         : chr  "Action" "Drama" "Sci-Fi" "Drama" ...
##  $ UserRating              : num  2.18 2.34 4.11 2.45 3.48 ...
##  $ SupportTicketsPerMonth  : int  7 0 3 7 3 1 8 3 0 6 ...
##  $ Gender                  : chr  "Female" "Female" "Female" "Female" ...
##  $ WatchlistSize           : int  7 18 23 5 9 24 9 16 16 2 ...
##  $ ParentalControl         : chr  "No" "Yes" "No" "No" ...
##  $ SubtitlesEnabled        : chr  "No" "No" "Yes" "No" ...
##  $ CustomerID              : chr  "WNXOZZL9ET" "Y7YQAS70DV" "3BWM3W0RX1" "VSIWM8W3
EB" ...
##  $ Churn                   : int  0 0 0 0 0 0 0 0 0 0 ...
```

#CHECKING FOR NA VALUES

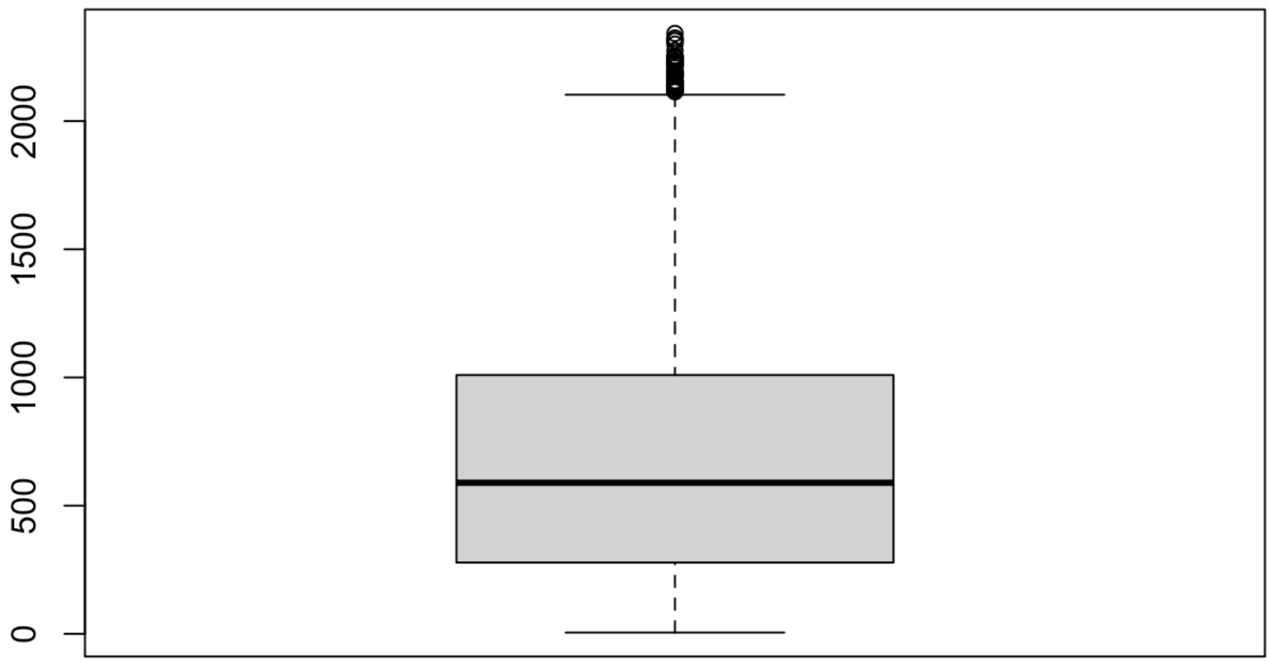
```
na_checking <- any(is.na(df))
na_checking
```

```
## [1] FALSE
```

#Checking outliers and disturbution

```
numeric_columns <- sapply(df, is.numeric)
numeric_data <- df[, numeric_columns]

par(mfrow = c(3, 4))

for (i in 1:ncol(numeric_data)) {
  boxplot(numeric_data[, i], col = "lightblue", main = names(numeric_data)[i])
}

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


```
boxplot(df$TotalCharges)
```



#

Removing outliers can lead to a loss of valuable information and variability in your data in total charges column.It represent legitimate and meaningful information about your dataset.

#categorical features Names

```
numeric_column = sapply(df, is.numeric)
categorical_colmumn = sapply(df, function(i) is.factor(i) || is.character(i))

categorical_col_names = names(df[categorical_colmumn])
categorical_col_names
```

```
##  [1] "SubscriptionType"  "PaymentMethod"     "PaperlessBilling"
##  [4] "ContentType"       "MultiDeviceAccess" "DeviceRegistered"
##  [7] "GenrePreference"   "Gender"            "ParentalControl"
## [10] "SubtitlesEnabled"  "CustomerID"
```

#categorical features Names and unique values

```
cat("Subscription Types:", paste(unique(df$SubscriptionType), collapse = ", "), "\n")
```

```
## Subscription Types: Basic, Standard, Premium
```

```
cat("PaymentMethod:", paste(unique(df$PaymentMethod), collapse = ", "), "\n")
```

```
## PaymentMethod: Credit card, Bank transfer, Mailed check, Electronic check
```

```
cat("PaperlessBilling:", paste(unique(df$PaperlessBilling), collapse = ", "), "\n")
```

```
## PaperlessBilling: No, Yes
```

```
cat("ContentType:", paste(unique(df$ContentType), collapse = ", "), "\n")
```

```
## ContentType: TV Shows, Movies, Both
```

```
cat("MultiDeviceAccess:", paste(unique(df$MultiDeviceAccess), collapse = ", "), "\n")
```

```
## MultiDeviceAccess: No, Yes
```

```
cat("DeviceRegistered:", paste(unique(df$DeviceRegistered), collapse = ", "), "\n")
```

```
## DeviceRegistered: Mobile, Computer, Tablet, TV
```

```
cat("GenrePreference:", paste(unique(df$GenrePreference), collapse = ", "), "\n")
```

```
## GenrePreference: Action, Drama, Sci-Fi, Comedy, Fantasy
```

```
cat("Gender:", paste(unique(df$Gender), collapse = ", "), "\n")
```

```
## Gender: Female, Male
```

```
cat("ParentalControl:", paste(unique(df$ParentalControl), collapse = ", "), "\n")
```

```
## ParentalControl: No, Yes
```

```
cat("SubtitlesEnabled:", paste(unique(df$SubtitlesEnabled), collapse = ", "), "\n")
```

```
## SubtitlesEnabled: No, Yes
```

#One hot encoding

```
df$PaperlessBilling = as.numeric(df$PaperlessBilling == "No")
df$MultiDeviceAccess = as.numeric(df$MultiDeviceAccess == "Yes")
df$ParentalControl = as.numeric(df$ParentalControl == "Yes")
df$SubtitlesEnabled = as.numeric(df$SubtitlesEnabled == "Yes")
df$Gender = as.numeric(df$Gender == "Female")
```

Dummy Variable Encoding

```
df = fastDummies::dummy_cols(df, select_columns = "ContentType")
df = fastDummies::dummy_cols(df, select_columns = "PaymentMethod")
df = fastDummies::dummy_cols(df, select_columns = "DeviceRegistered")
df = fastDummies::dummy_cols(df, select_columns = "GenrePreference")
```

#Ordinal Encoding

```
df$SubscriptionType <- sapply(df$SubscriptionType, switch,
  "Premium"=3,
  "Basic"=1,
  "Standard"=2,
  )
```

#REMOVING UNWANTED COLUMNS

```
columns_to_remove1 <- c("PaymentMethod","PaymentMethod_Bank transfer", "ContentTyp
e","ContentType_Both","DeviceRegistered_Computer", "DeviceRegistered","GenrePreferenc
e","GenrePreference_Sci-Fi","CustomerID")

columns_to_remove2 = c("PaymentMethod","PaymentMethod_Bank transfer", "ContentType","
ContentType_Both","DeviceRegistered_Computer", "DeviceRegistered","GenrePreference","
GenrePreference_Sci-Fi","CustomerID", "Churn","SubscriptionType")
df1 = df[, setdiff(names(df), columns_to_remove1)]
df <- df[, setdiff(names(df), columns_to_remove2)]
```

```
df
```

AccountA... <int>	MonthlyCharges <dbl>	TotalCharges <dbl>	PaperlessBilling <dbl>	MultiDeviceAccess <dbl>	Viewin
48	19.815454	951.141809	1	0	
5	19.707053	98.535265	0	1	
15	5.222221	78.333312	1	1	
75	15.804611	1185.345852	0	0	
85	14.053068	1194.510801	1	0	
67	17.852482	1196.116303	1	0	
114	9.475375	1080.192790	1	1	
8	7.853356	62.826847	0	0	
55	11.752374	646.380591	1	1	
22	12.265846	269.848609	1	0	

1-10 of 6,000 rows | 1-6 of 26 columns Previous **1** 2 3 4 5 6 ... 600 Next

```
cor_matrix <- cor(df)
correlated_features <- findCorrelation(cor_matrix, cutoff = 0.9)
correlated_features
```

```
## integer(0)
```

#Indicates that there are no highly correlated features among themself to be removed.

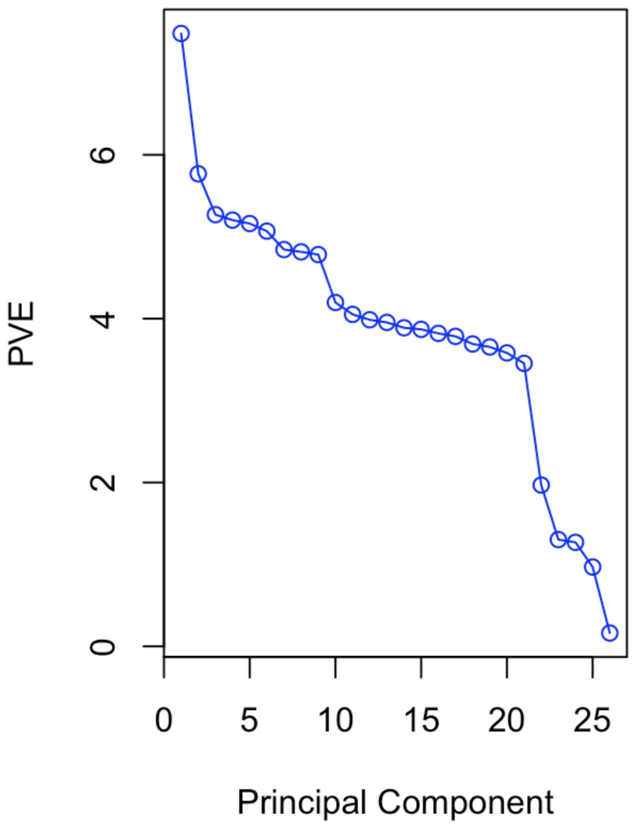
#SCALING DATA

```
r df_scale <- scale(df)
```

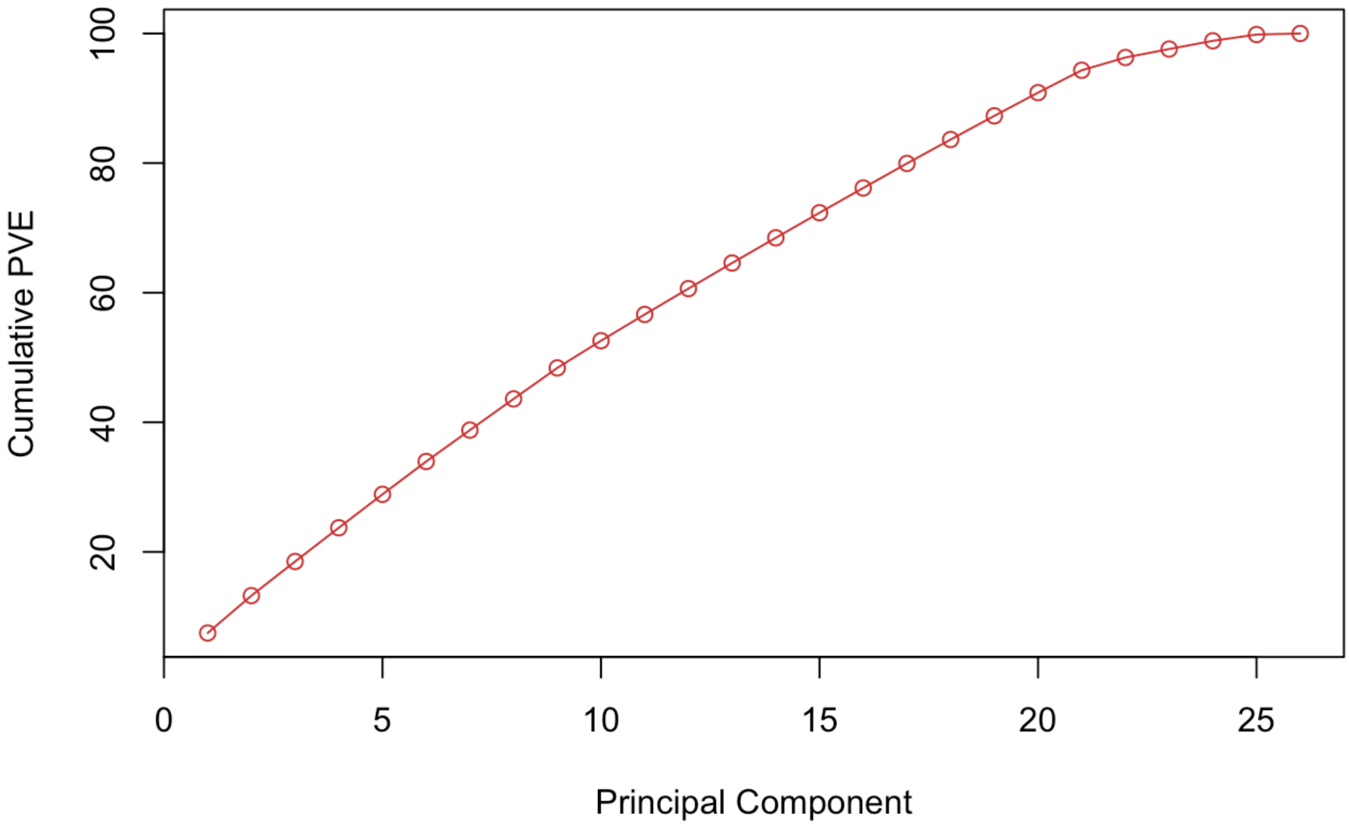
##PCA

```
pc.out <- prcomp(df_scale, scale = T)
#pc.out
```

```
pve <- 100 * pc.out$sdev^2 / sum(pc.out$sdev^2)
par(mfrow = c(1, 2))
plot(pve,  type = "o", ylab = "PVE",
    xlab = "Principal Component", col = "blue")
```

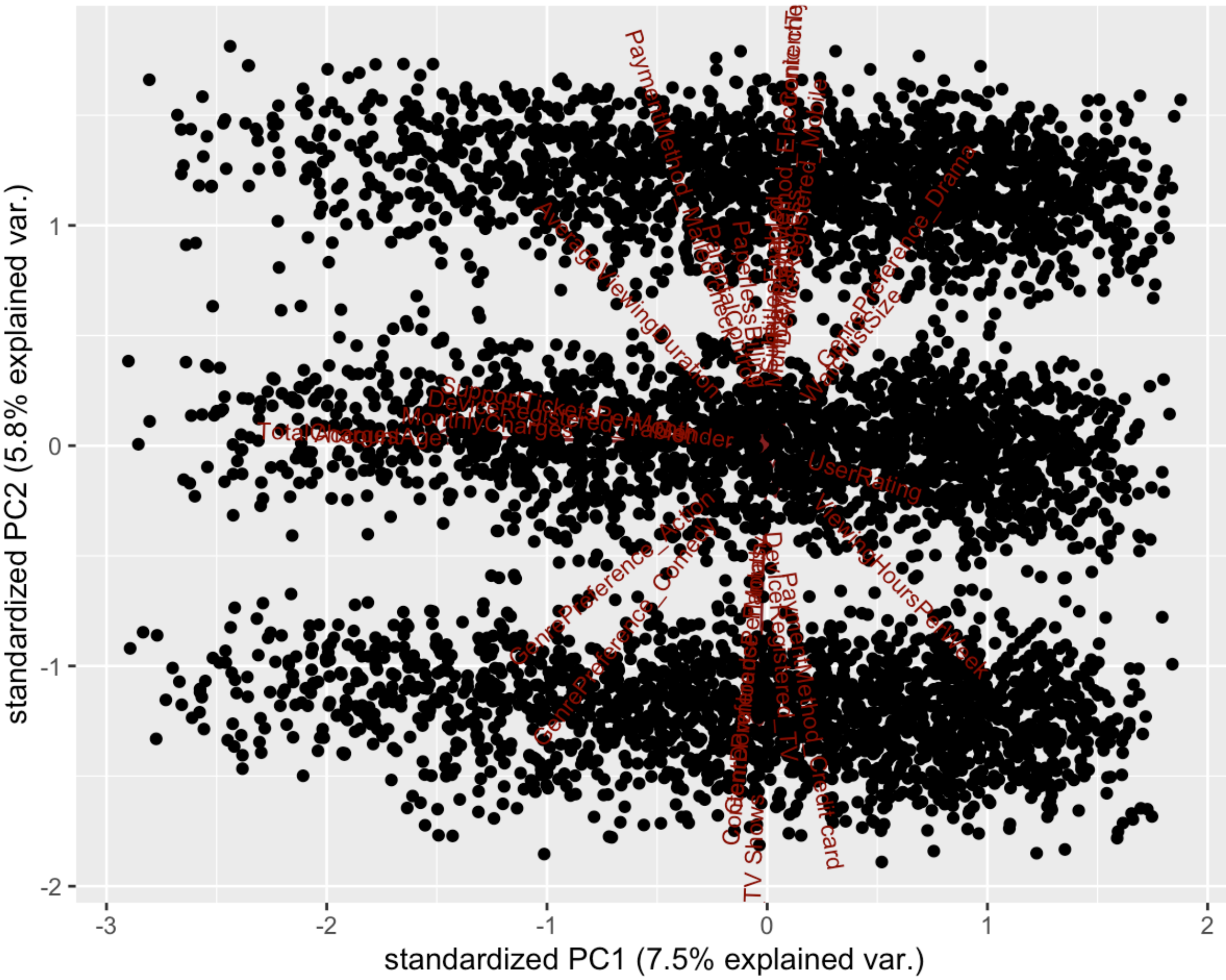



```
plot(cumsum(pve), type = "o", ylab = "Cumulative PVE",
    xlab = "Principal Component", col = "brown3")
```

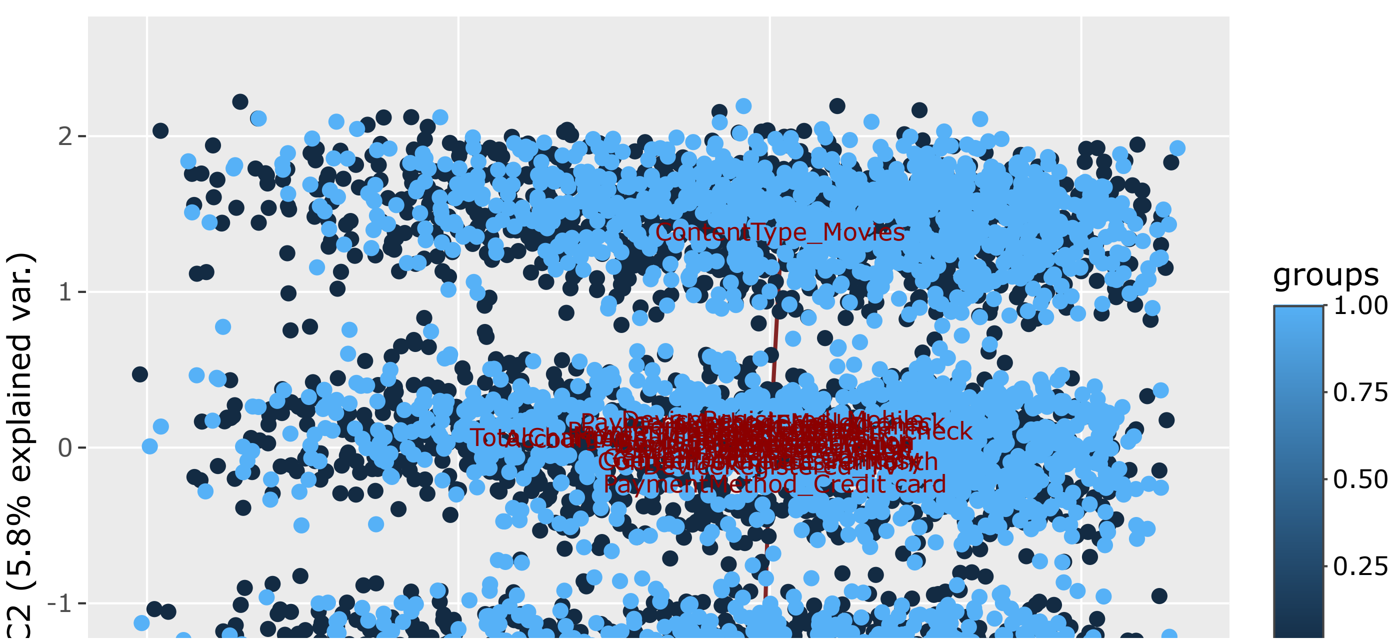


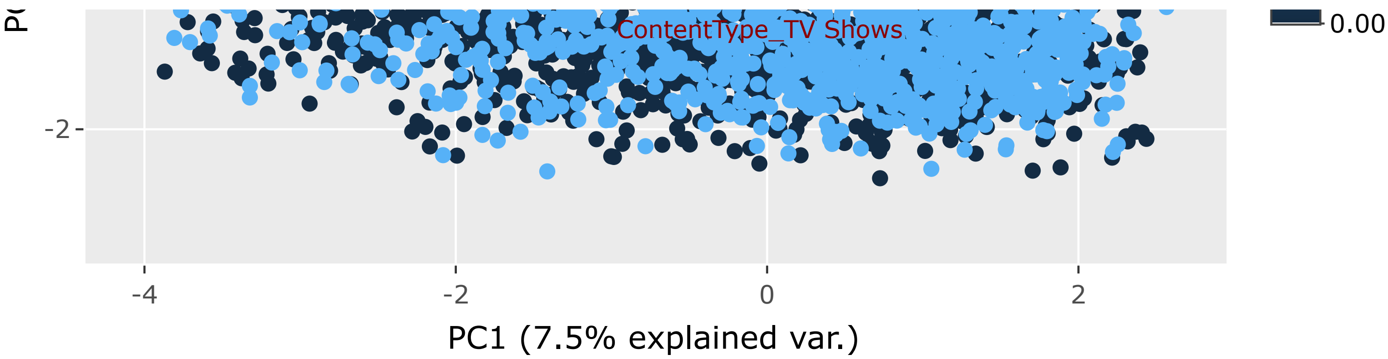
20 PCs.

```
library(ggbiplot)
ggbiplot(pc.out, scale = T, labels=rownames(pc.out$x))
```



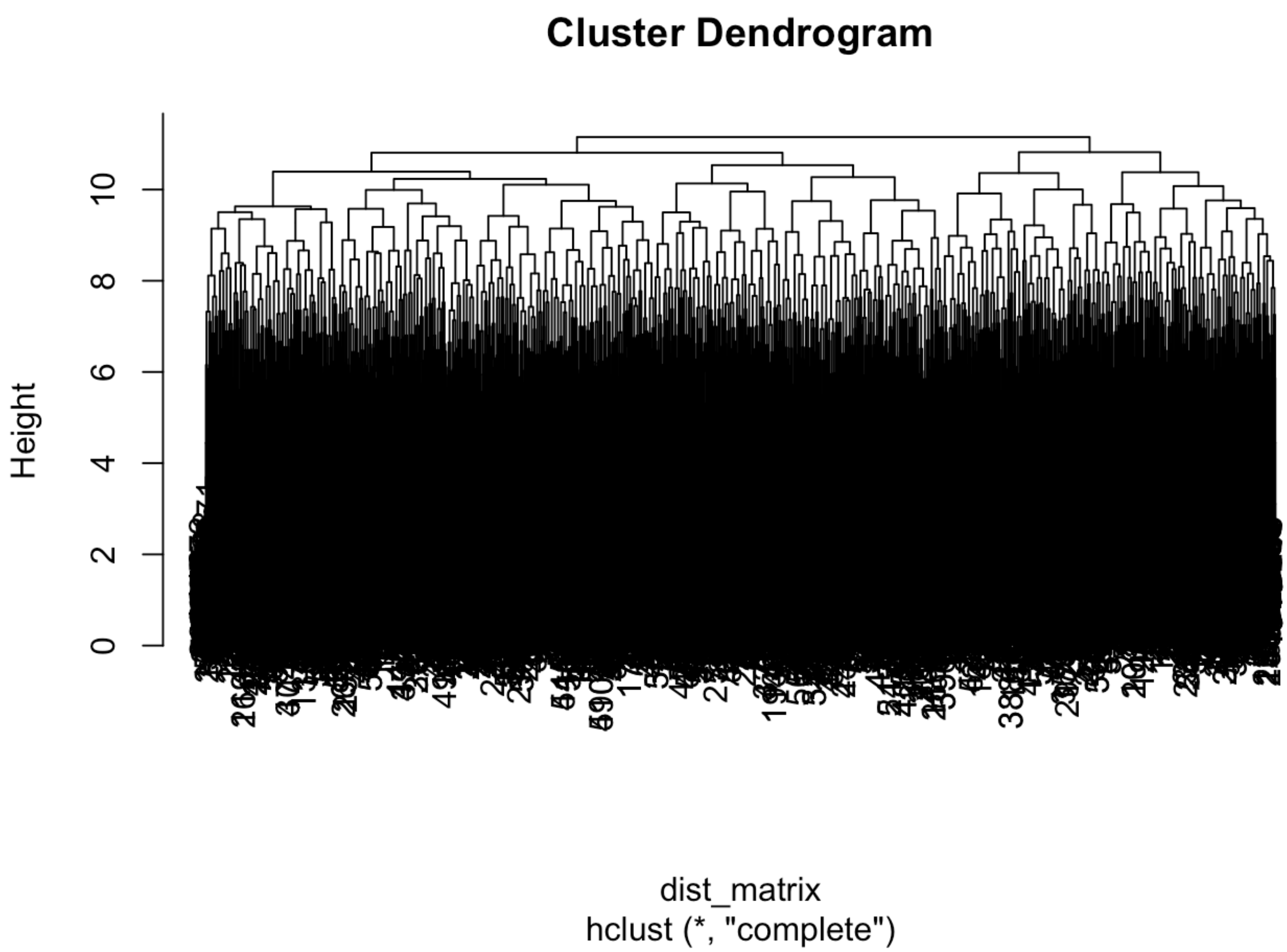

```
ggplotly(ggbiplot(pc.out, scale = -0, groups=df1$Churn))
```



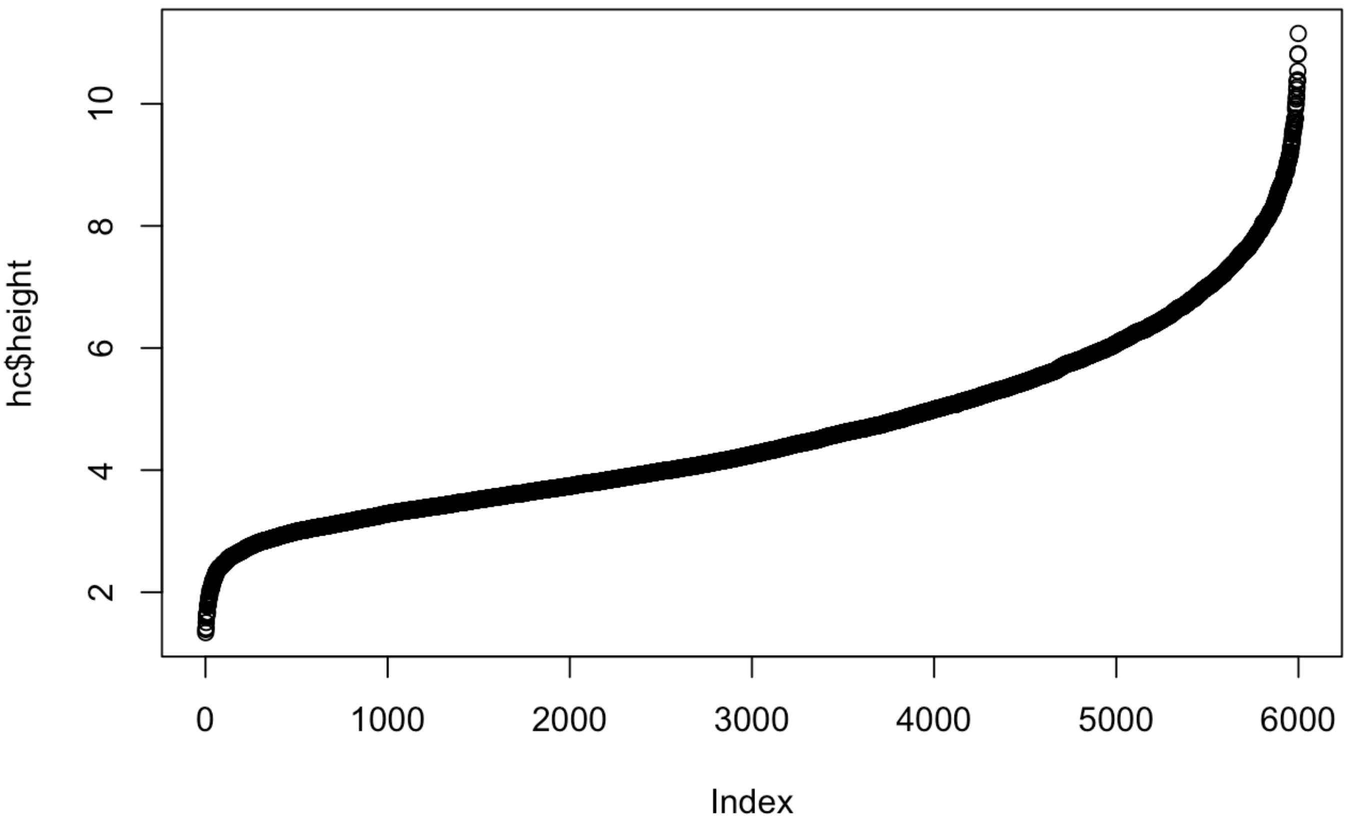


#SHILA MODELS

```
dist_matrix <- dist(df_scale, method = "euclidean")

hc <- hclust(dist_matrix, method = "complete")
plot(hc)
```




```
plot(hc$height, type = "b")
```




```r
# Silhouette method
k_min <- 3
k_max <- 10
sil_width <- numeric(k_max - k_min + 1)
```

```r
# Loop over the number of clusters
for (k in k_min:k_max) {
  clustering <- cutree(hclust(dist_matrix, method = "complete"), k)
  silhouette_obj <- silhouette(clustering, dist_matrix)
  sil_width[k - k_min + 1] <- mean(silhouette_obj[, "sil_width"])
}
```

```r
# Find the number of clusters that gives the maximum average silhouette width
optimal_clusters <- which.max(sil_width) + k_min - 1
```

```r
# Print the optimal number of clusters
print(paste("Optimal number of clusters: ", optimal_clusters))
```

```
## [1] "Optimal number of clusters:  3"
```

```
clusters <- cutree(hc, k = 3)
```

```
# Add the cluster assignments to your dataframe
df2 <- df1[, -c(4:9,13,16,18:20)]
df2$Cluster <- clusters
aggregate(. ~ Cluster, data = df2, mean)
```

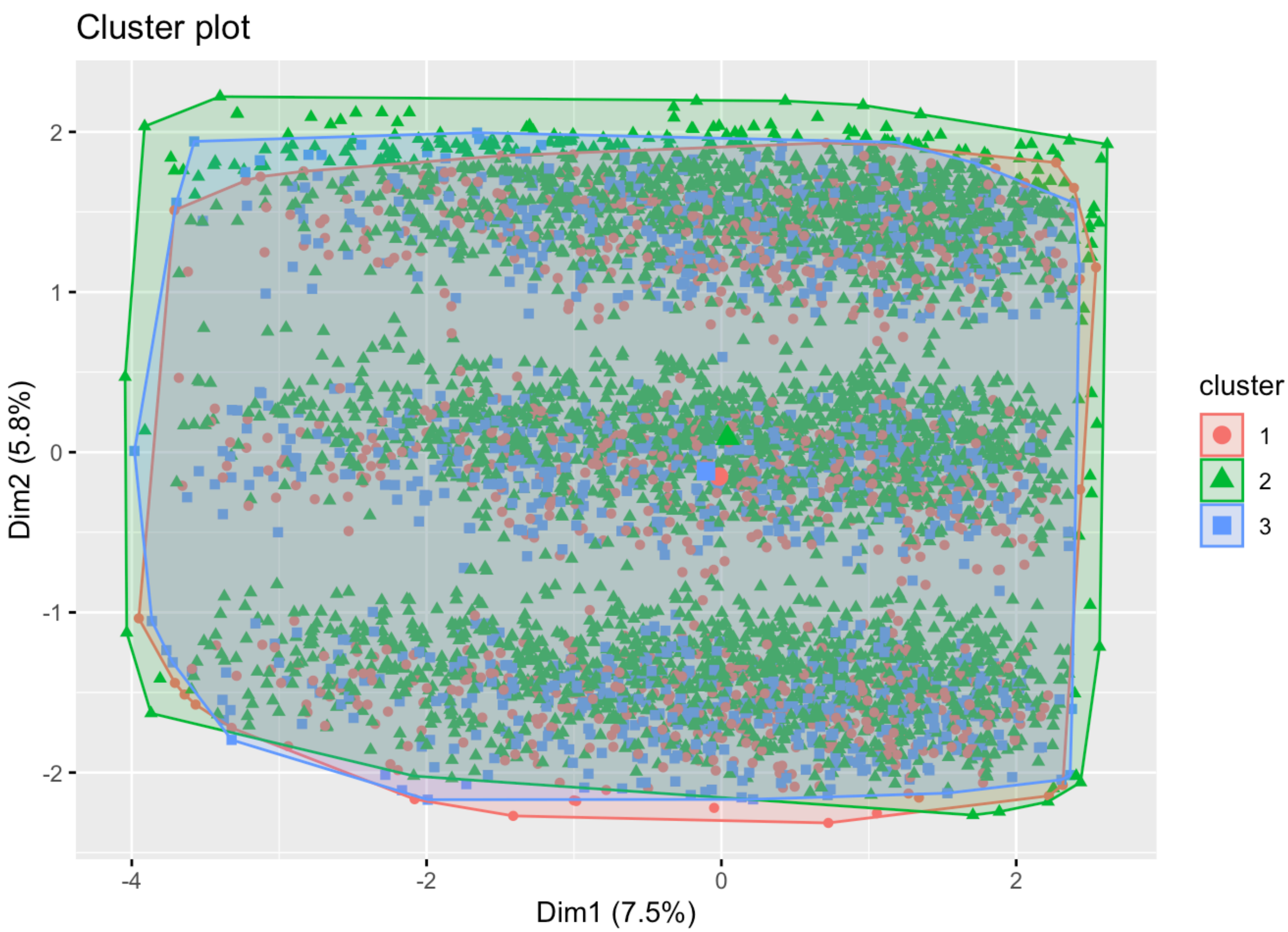
Cluster <int>	AccountA... <dbl>	MonthlyCharges <dbl>	TotalCharges <dbl>	UserRating <dbl>	SupportTicketsPerMonth <dbl>
1	85.26510	15.07035	1278.6673	2.974772	4.889458
2	43.33470	12.09975	498.8914	3.037486	4.697219
3	72.54054	13.26615	943.1113	3.084541	4.415541

3 rows | 1-7 of 18 columns

Gaussian Mixture model

```
opt_gmm = Optimal_Clusters_GMM(df_scale, max_clusters = 10, criterion = "BIC",

                               dist_mode = "maha_dist", seed_mode = "random_subset",

                               km_iter = 10, em_iter = 10, var_floor = 1e-10,

                               plot_data = T)
```

```
# Run GMM clustering
gmm_model <- Mclust(df_scale, G = 3)  # Choose the number of components (k)

# Add cluster assignment to the original dataset
churn_data_gmm <- cbind(df_scale, cluster = as.factor(gmm_model$classification))

# Visualize the clusters
fviz_cluster(gmm_model, data = df_scale, geom = "point", stand = FALSE)
```

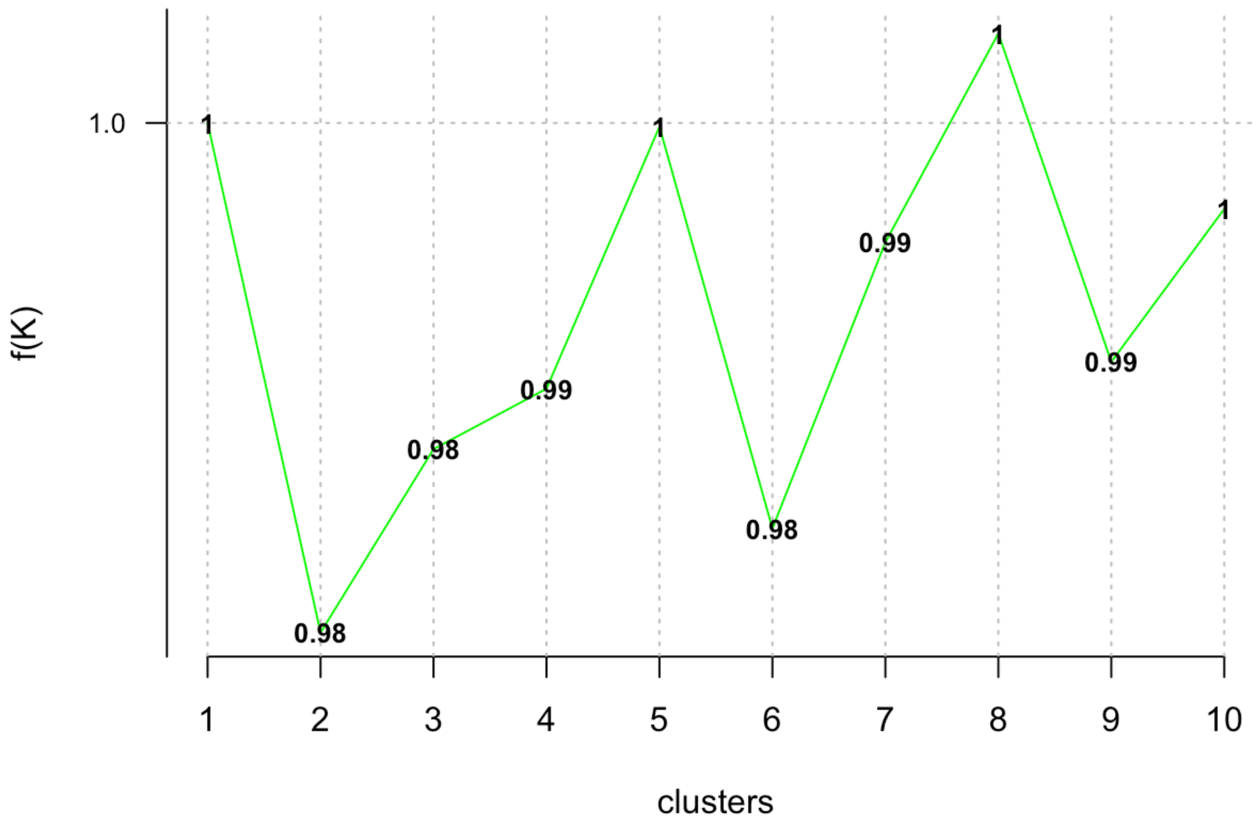


In case of model selection, among a specific number of models, the model with the lowest BIC should be preferred, which is true here for a number of clusters equal to 3.

```
km_rc = KMeans_rcpp(df_scale, clusters = 5, num_init = 5, max_iters = 100,
                    initializer = 'optimal_init', verbose = F)

km_rc$between.SS_DIV_total.SS
```

```
## [1] 0.1425753
```

```
opt = Optimal_Clusters_KMeans(df_scale, max_clusters = 10, plot_clusters = T,

                              criterion = 'distortion_fK', fK_threshold = 0.85,

                              initializer = 'optimal_init', tol_optimal_init = 0.2)
```



Values below the fixed threshold (here fK_threshold = 0.85) could be recommended for clustering, however there are multiple optimal clusterings and this highlights the fact that f(K) should only be used to suggest a guide value for the number of clusters and the final decision as to which value to adopt has to be left at the discretion of the user.

#K MEANS

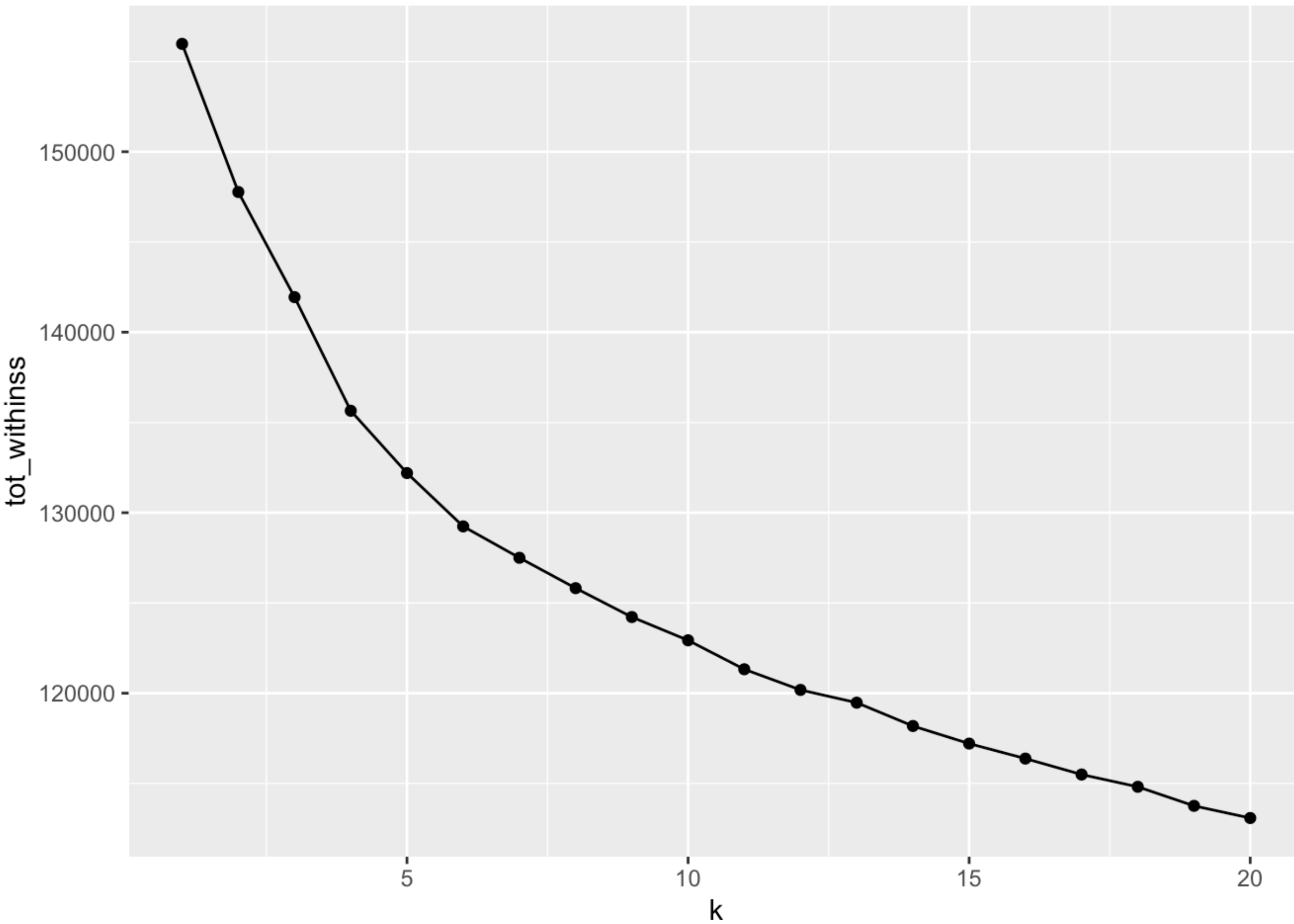
```
km_out_list <- lapply(1:20, function(k) list(
  k=k,
  km_out=kmeans(df_scale, k, nstart = 20)))


km_results <- data.frame(
  k=sapply(km_out_list, function(k) k$k),
  totss=sapply(km_out_list, function(k) k$km_out$totss),
  tot_withinss=sapply(km_out_list, function(k) k$km_out$tot.withinss)
  )
km_results
```

k	totss	tot_withinss

<int>	<dbl>	<dbl>
1	155974	155974.0
2	155974	147767.5
3	155974	141942.0
4	155974	135647.5
5	155974	132195.4
6	155974	129241.0
7	155974	127502.5
8	155974	125819.4
9	155974	124219.9
10	155974	122930.2

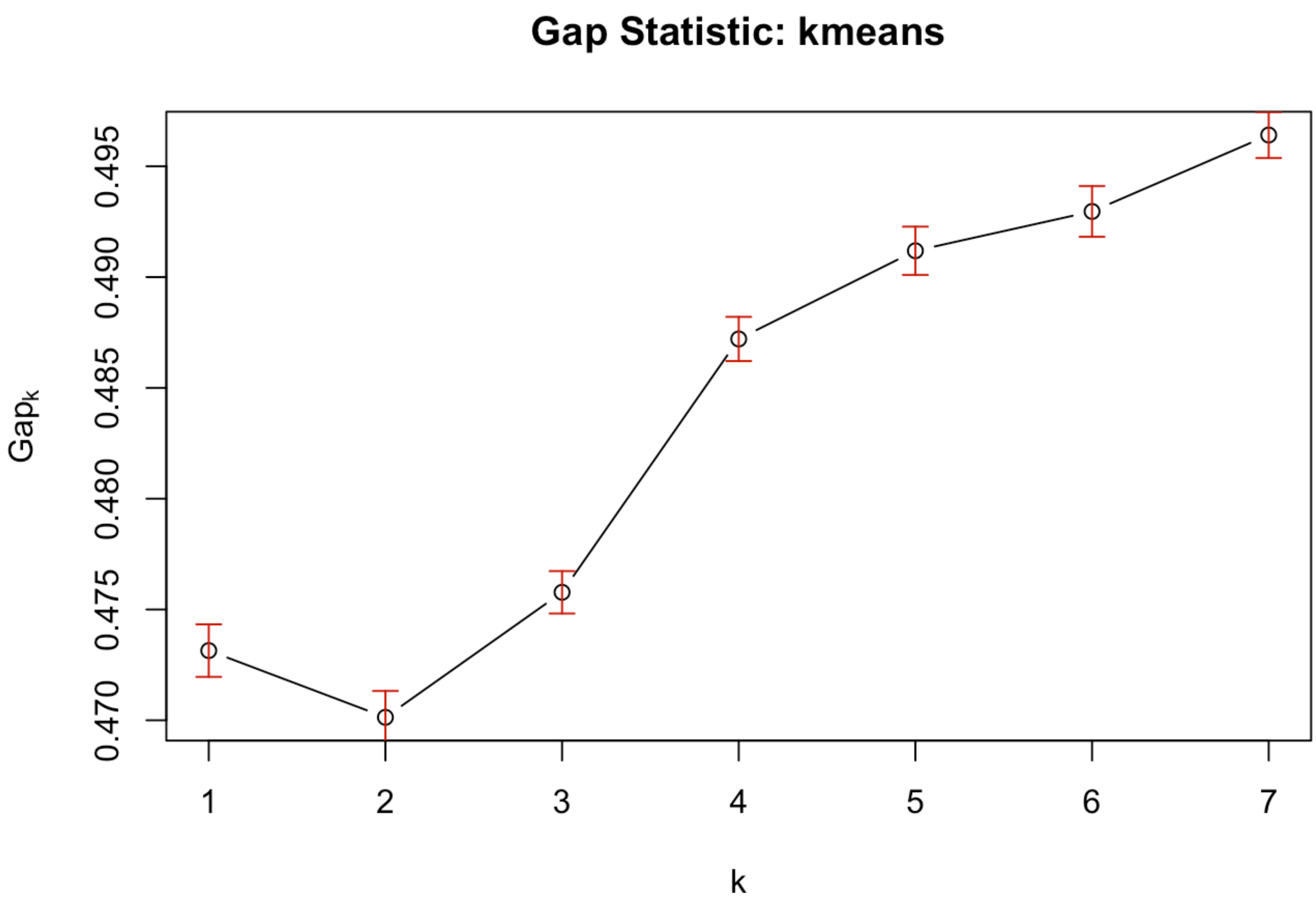
1-10 of 20 rows Previous **1** 2 Next

```
ggplot(km_results,aes(x=k,y=tot_withinss))+geom_line()+geom_point()
```

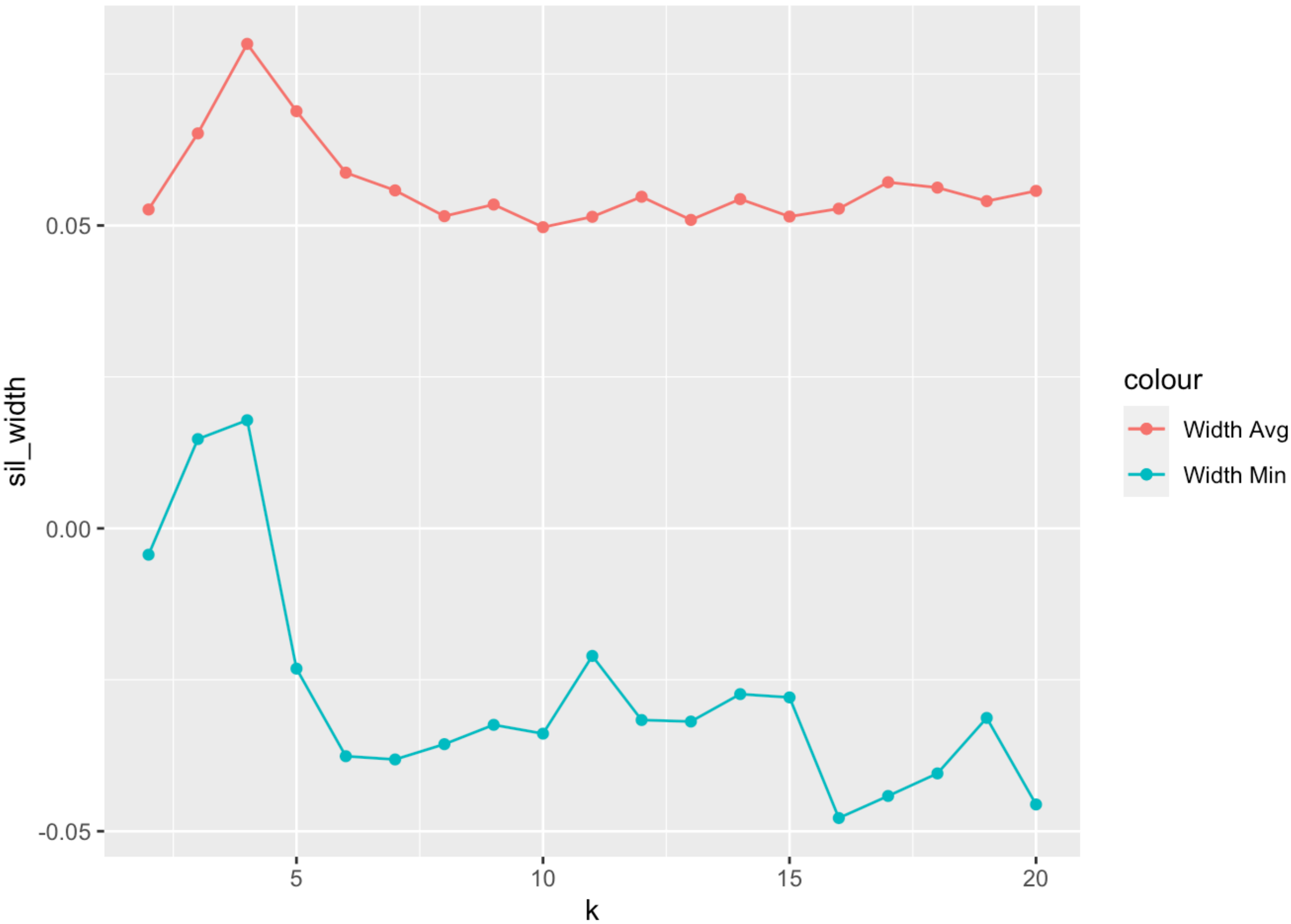


select optimal number of clusters using gap statistic

```
set.seed(1)
gap_kmeans <- clusGap(df_scale, kmeans, nstart = 20, K.max = 7, B = 10)

plot(gap_kmeans, main = "Gap Statistic: kmeans")
```




```
#Silhouette
set.seed(1)
results <- lapply(2:20, function(k) {
  kmeans_cluster <- kmeans(df_scale, k, nstart=20, iter.max=20)
  si <- silhouette(kmeans_cluster$cluster, dist = dist(df_scale))
  data.frame(k=k,sil_width=mean(si[,'sil_width']),sil_width_min=min(si[,'sil_widt
h']))
})
si_df <- bind_rows(results)

ggplot(si_df, aes(x=k,y=sil_width,color="Width Avg"))+geom_point()+geom_line()+
  geom_point(aes(y=sil_width_min,color="Width Min"))+geom_line(aes(y=sil_width_min,co
lor="Width Min"))
```



#

3 WILL BE A GOOD CHOICE.

DBScan Clustering

Desity Based Clustering group objects into cluster

#various shapes and sizes also less noise to outliers like k means

```
unwanted_columns <- c("PaymentMethod", "PaperlessBilling", "ContentType", "MultiDevic
eAccess", "DeviceRegistered", "GenrePreference", "SubtitlesEnabled", "Gender", "Paren
talControl", "Churn")

df_ni <- df_ni %>%
  select(-any_of(unwanted_columns))

df_ni$SubscriptionType = factor(df_ni$SubscriptionType, levels = unique(df_ni$Subscri
ptionType), labels = c(3L, 1L, 2L), ordered = TRUE)

head(df_ni)
```

	AccountA... <int>	MonthlyCharges <dbl>	TotalCharges <dbl>	SubscriptionType <ord>	ViewingHoursPerWeek <dbl>
1	48	19.815454	951.14181	3	35.96337
2	5	19.707053	98.53526	3	17.92241
3	15	5.222221	78.33331	1	15.79676
4	75	15.804611	1185.34585	2	33.87384
5	85	14.053068	1194.51080	1	35.44470
6	67	17.852482	1196.11630	2	14.35675

6 rows | 1-6 of 12 columns

```
selected_features <- c("AccountAge", "MonthlyCharges", "TotalCharges", "SubscriptionT
ype", "ViewingHoursPerWeek", "AverageViewingDuration", "ContentDownloadsPerMonth", "U
serRating", "SupportTicketsPerMonth", "WatchlistSize")

df_ni_selected <- select(df_ni, selected_features)
```

```r
selected_features <- as.data.frame(lapply(df_ni_selected, as.numeric))
preprocess <- preProcess(selected_features, method = c("center", "scale"))
scaled_features_caret <- predict(preprocess, selected_features)

scaled_features_caret<-apply(scaled_features_caret, 2, function(x) (x - min(x)) / (ma
x(x) - min(x)))

unwanted_columns <- c("AccountAge", "MonthlyCharges", "TotalCharges", "SubscriptionTy
pe", "ViewingHoursPerWeek", "AverageViewingDuration", "AverageViewingDuration", "Cont
entDownloadsPerMonth", "UserRating", "SupportTicketsPerMonth", "WatchlistSize")

df_ni <- select(df_ni, -one_of(unwanted_columns))
df_ni <- cbind(df_ni, scaled_features_caret)
```

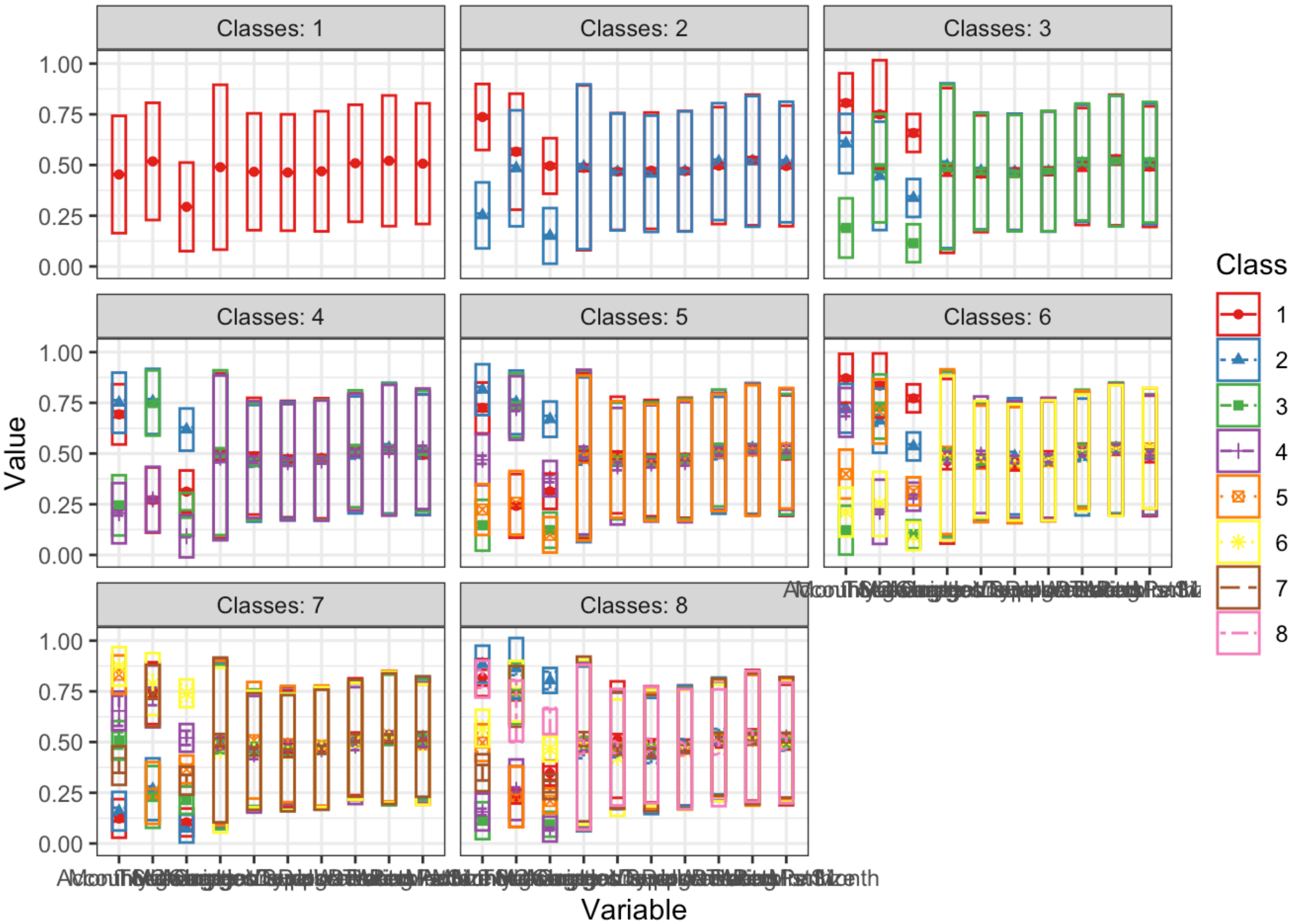
```r
library(tidyLPA)
```

```
## You can use the function citation('tidyLPA') to create a citation for the use of {
tidyLPA}.
## Mplus is not installed. Use only package = 'mclust' when calling estimate_profile
s().
```

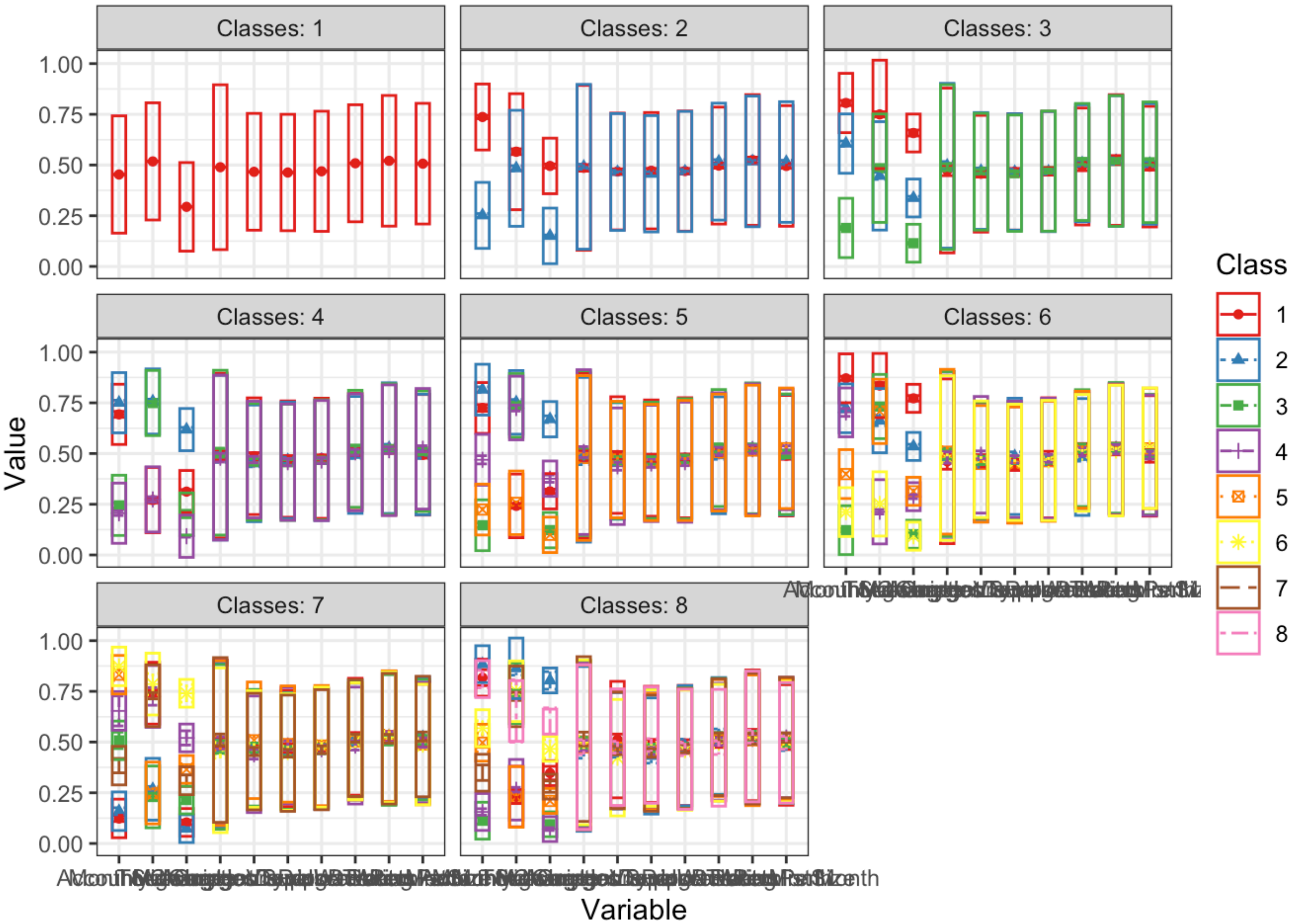
```r
VLPA <- df_ni[,-1] %>% estimate_profiles(1:8)
VLPA
```

```
## tidyLPA analysis using mclust:
##
##  Model Classes AIC      BIC      Entropy prob_min prob_max n_min n_max BLRT_p
##  1     1       23859.27 23993.26 1.00    1.00     1.00     1.00  1.00
##  1     2       18091.15 18298.84 0.84    0.94     0.96     0.41  0.59  0.01
##  1     3       15117.42 15398.80 0.86    0.92     0.95     0.17  0.45  0.01
##  1     4       13974.83 14329.91 0.85    0.90     0.95     0.23  0.28  0.01
##  1     5       11915.55 12344.32 0.86    0.87     0.95     0.17  0.25  0.01
##  1     6       10513.16 11015.63 0.87    0.88     0.92     0.07  0.24  0.01
##  1     7       9082.07  9658.23  0.87    0.87     0.95     0.10  0.20  0.01
##  1     8       8185.93  8835.79  0.87    0.87     0.92     0.06  0.19  0.01
```

```r
plot_profiles(VLPA, rawdata = FALSE)
```




```
plot_profiles(VLPA, rawdata = FALSE)
```




```
par(las = 2)
```

```
df_ni
```

Customer... <chr>	AccountAge <dbl>	MonthlyCharges <dbl>	TotalCharges <dbl>	SubscriptionType <dbl>	ViewingHoursP
WNXOZZL9ET	0.398305085	9.886728e-01	4.048709e-01	0.0	0.896
Y7YQAS70DV	0.033898305	9.814422e-01	4.001835e-02	0.0	0.433
3BWM3W0RX1	0.118644068	1.527547e-02	3.137341e-02	0.5	0.379
VSIWM8W3EB	0.627118644	7.211417e-01	5.050930e-01	1.0	0.842
Q5F4H0Q0TV	0.711864407	6.043103e-01	5.090149e-01	0.5	0.883
PSJ6SREKVP	0.559322034	8.577387e-01	5.097019e-01	1.0	0.342
5WH7LLTPW5	0.957627119	2.989692e-01	4.600952e-01	1.0	0.722

FEO69EG20G	0.059322034	1.907773e-01	2.473778e-02	0.0	0.371
OXV0DP85BH	0.457627119	4.508495e-01	2.744557e-01	1.0	0.210
XUNE0YQ6O2	0.177966102	4.850991e-01	1.133278e-01	0.0	0.178

1-10 of 6,000 rows | 1-6 of 11 columns Previous **1** 2 3 4 5 6 ... 600 Next