```
install.packages('factoextra', quiet = TRUE)
```

```
also installing the dependencies ‘matrixStats’, ‘RcppArmadillo’, ‘numDeriv’, ‘SparseM’, ‘MatrixModels’, ‘conquer’, ‘sp’, ‘oper
```

Data

```
data <- read.table("vectors2.csv",sep=",",header=TRUE)
data <- data[, 3:43]
data
```

A data.frame: 200 × 41

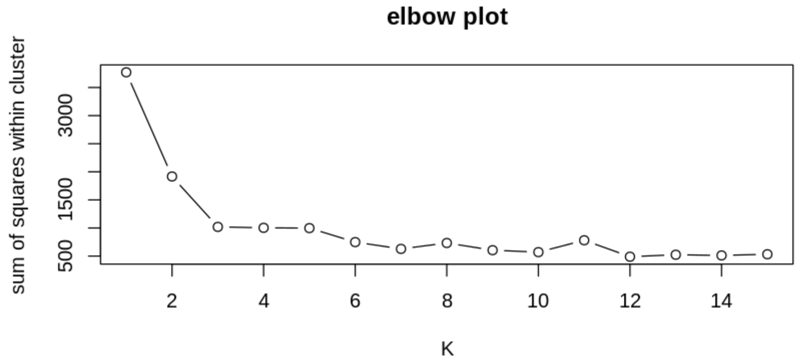
class	v0	v1	v2	v3	v4	v5	v6	v7	v8	⋯	v30	
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	⋯	<dbl>	
human	2.6002622	1.44048011	-1.2125840	-1.5646441	-1.41500854	1.7523570	-0.6888220	-0.8314072	1.40226841	⋯	-1.0081722	1.048
human	2.2157688	1.03172469	-0.8887914	-1.2748611	-1.33984053	1.3295089	-0.5240245	-1.0537314	1.19442236	⋯	-1.4297421	0.621
human	2.9760008	1.63780832	-1.2346615	-1.5561976	-1.22166657	1.8647786	-0.6280878	-1.1813650	1.58883548	⋯	-1.2211341	1.035
human	1.6182292	1.68940926	-1.2624258	-0.5248858	-0.99792296	1.3417084	-0.6341772	-2.1757324	0.60869700	⋯	-1.8267305	0.557
human	1.6839134	1.68101740	-0.6077973	-1.1181115	-1.53544188	1.1893209	-0.9949601	-1.0366992	0.45895031	⋯	-1.8144143	1.099
human	2.3967137	0.86961067	-1.1279887	-1.5695435	-1.81571853	1.6694503	-0.3483628	-0.3701709	1.37537301	⋯	-0.9840603	0.825
human	1.4089483	3.10270905	-1.2953923	-0.7504808	-1.00633419	1.1089976	-1.1454989	-1.8507205	0.26207894	⋯	-2.2533493	0.277
human	1.2286259	3.11897492	-1.1619078	-1.1058580	-1.23588371	1.2711140	-0.7079484	-1.5657444	0.51212144	⋯	-1.9786599	0.586
human	0.1582623	2.76319265	-1.2495297	-1.0579451	-1.10284162	1.3138505	-1.3044717	-1.3073403	0.55921078	⋯	-1.4195971	1.105
human	0.5362926	2.53814507	-1.4477805	-1.1432642	-0.76792246	0.8279900	-1.2931086	-1.3260915	0.50838137	⋯	-1.1723914	0.568
human	-0.3412297	1.83045518	-0.7174654	-0.7853898	-1.21853793	0.5915188	-0.6725301	-0.4000489	0.15973842	⋯	-0.7259036	0.200
human	-0.6835908	1.95631003	-1.5721922	-0.6626856	0.09793150	0.9661801	-0.4417855	-1.9962054	-0.36830732	⋯	-0.3823577	-0.292
human	-0.1554935	2.19747496	-1.5194000	-0.6601118	-0.31410673	0.5465780	-0.6167886	-1.6708215	-0.23363005	⋯	-1.4189301	0.186
human	-0.2621725	2.23505116	-1.7367764	-0.8124174	-0.06961089	0.5102980	-0.7536057	-1.9797951	-0.13061519	⋯	-0.8454061	-0.199
human	0.7789880	1.83514452	-1.5670017	-0.7146733	-0.49398434	0.4144911	-0.5048890	-1.0093025	0.50358540	⋯	-0.8092731	0.352
human	0.4798467	1.52457511	-1.6577194	-0.8720924	-0.60783249	1.1561311	-1.3307598	-1.2843437	0.29418337	⋯	-1.0465782	0.408
human	1.1897928	1.26390207	-1.5219202	-0.9675056	-0.41653448	1.0907078	-0.4028623	-1.1677103	0.88214546	⋯	-0.2576329	0.286
human	0.7179680	0.87458223	-1.3510417	-0.9207808	-1.20464408	0.8755190	-1.0254772	-0.4963304	0.31916550	⋯	-0.4790622	0.585
human	-0.6649646	2.16206455	-1.3498904	-0.5153473	-0.42721024	0.4012737	-1.1893753	-1.2433535	-0.04809623	⋯	-0.6534677	0.010
human	1.1432834	2.20961690	-1.2179929	-1.0344322	-0.51923919	0.8803607	-0.3984118	-0.8131989	1.20665181	⋯	-0.5468500	0.421
human	2.8959241	0.34382114	-0.9134481	-1.0877239	-1.74490499	1.8839951	-0.3476616	-1.0831336	0.88909280	⋯	-1.2558346	1.265
human	2.1600919	0.77130175	-0.8954796	-1.4803972	-2.37020206	1.5525680	-0.1877291	-0.1551805	0.90475678	⋯	-1.0635607	1.009
human	2.6835163	1.11130869	-1.2433289	-1.2418070	-1.39735198	1.6839832	-0.7750425	-0.6335618	1.44516659	⋯	-1.2877517	1.025
human	2.1821909	0.89844996	-0.9826505	-1.2898159	-0.85285729	1.7988576	-0.5756720	-2.0449686	0.74755251	⋯	-1.6075255	0.663
human	2.6053896	0.05115666	-1.0464821	-1.4855775	-1.39245033	1.6851696	-0.5920713	-1.1345772	0.96375698	⋯	-1.2920569	0.601
human	2.1578369	1.17817080	-1.0859376	-1.7099813	-1.70924211	1.0101749	-0.4743942	0.2785887	1.15109539	⋯	-1.0261079	0.722
human	2.7794163	0.88002133	-1.5379760	-1.5640072	-1.72650969	1.4508874	-0.7772204	-0.1728411	1.57218206	⋯	-1.1858189	1.099
human	2.5539362	0.32000905	-1.0632042	-1.1694876	-1.68217218	1.3779832	-0.6117345	0.5915592	1.26069355	⋯	-0.9087094	1.158
human	2.4145074	0.87877029	-1.2541808	-1.3107020	-2.01024699	1.2688193	-0.3930501	-0.3186818	1.24710310	⋯	-1.4600054	0.632
human	2.2471879	1.53314388	-1.0601480	-1.3981249	-1.24242842	2.0320995	-0.5094057	-1.7262914	1.25275052	⋯	-1.4591299	0.843
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋱	⋮	
bot	0.274199843	0.4734849	-1.3021432	0.189958766	-6.635840e-01	0.2022090	-0.21016467	0.13394189	-0.4501034319	⋯	-0.1013611	0.66
bot	0.204511881	0.2078215	-1.0952222	-0.247913197	-5.309961e-01	0.2162134	-0.43332335	0.34848374	-0.3328803480	⋯	0.1419856	0.96
bot	0.013565375	0.8105545	-1.3828044	-0.060451977	-6.835277e-02	0.5260870	-0.13439047	-0.21048053	-0.1486476809	⋯	0.3734888	0.76
bot	0.379952312	0.5292860	-0.9601328	0.048931684	-4.602781e-01	0.4357498	-0.11991909	0.06528260	-0.2989134192	⋯	0.1830565	0.94
					-5.598539e-							
bot	0.199923351	0.4140221	-1.0760778	-0.289936125	[illegible]	0.4354911	-0.26338628	-0.09649979	0.0201591253	⋯	0.5795038	0.79
					-3.233636e-							

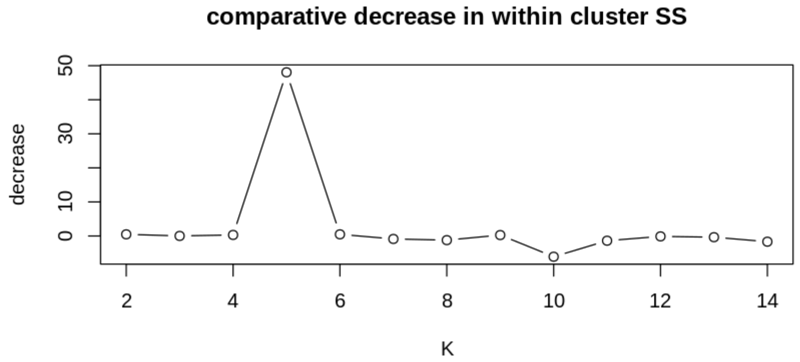
```
train_sp <- sample(200, round(0.7*200), replace=FALSE)

train_df <- data[train_sp, ]
test_df <- data[-train_sp, ]
```

Simple K-means

```
km_ss <- function(k) {
  kmk.out <- kmeans(train_df[, 2:41], k)
  return(kmk.out$tot.withinss)
}

K <- 15
par(mfcol = c(2,1))
```

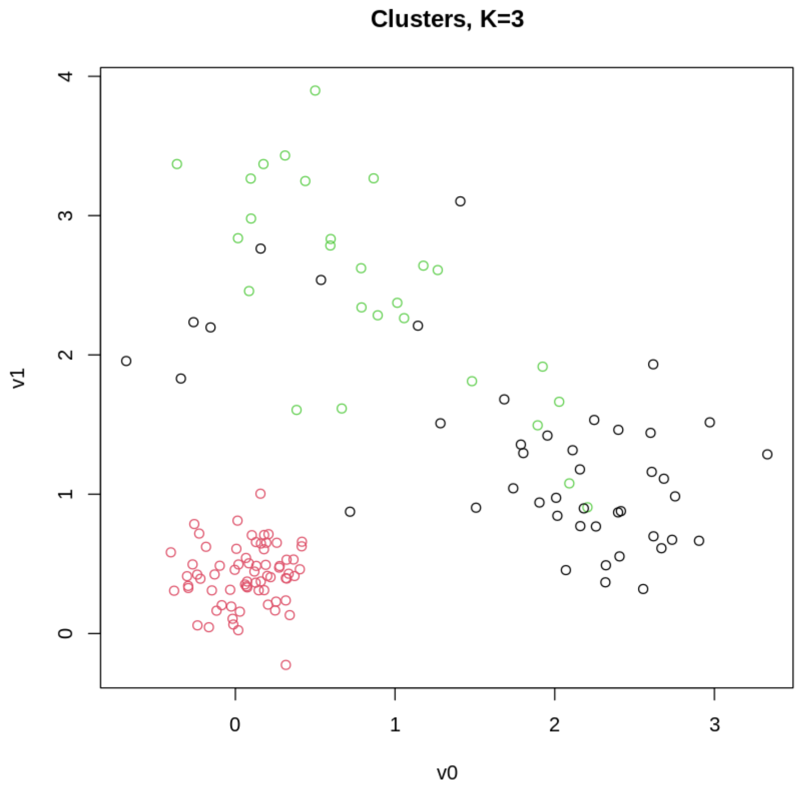
```
k_ss <- sapply(1:K, km_ss)
plot(1:K, k_ss, ylab='sum of squares within cluster', xlab='K', main='elbow plot', type='b')
k_ss_compare <- sapply(2:(K-1), function(x) (k_ss[x]-k_ss[x+1])/(k_ss[x-1]-k_ss[x]))
plot(2:(K-1), k_ss_compare, ylab='decrease', main='comparative decrease in within cluster SS', xlab='K', type='b')
```



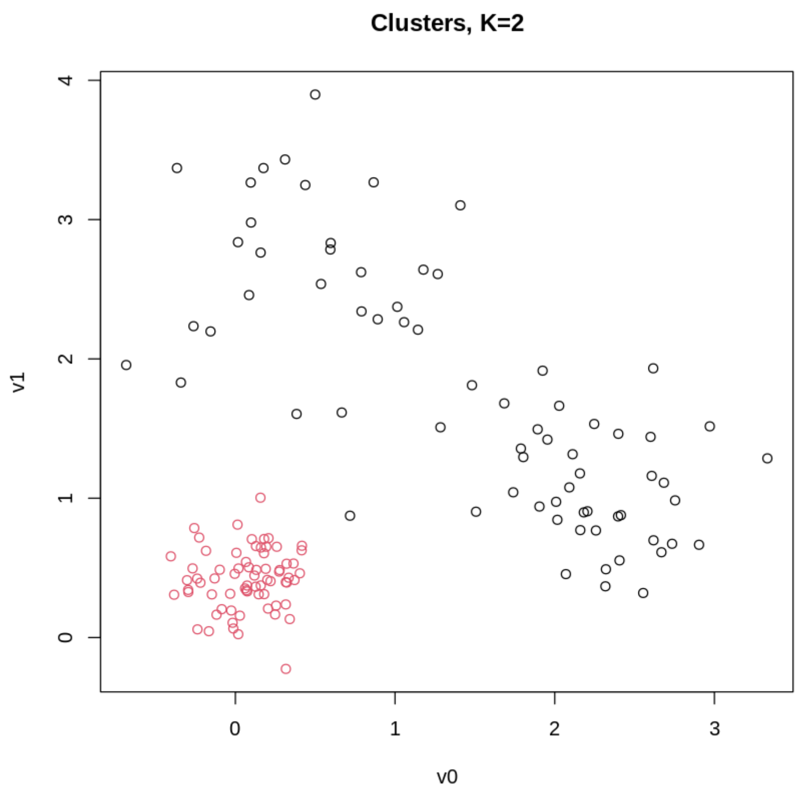


bot	0.315666944	0.2377967	1.3142340	0.066260375	-5.332912e-	0.3318657	0.56951797	0.38794681	0.2917513251	0.4398451	0.92

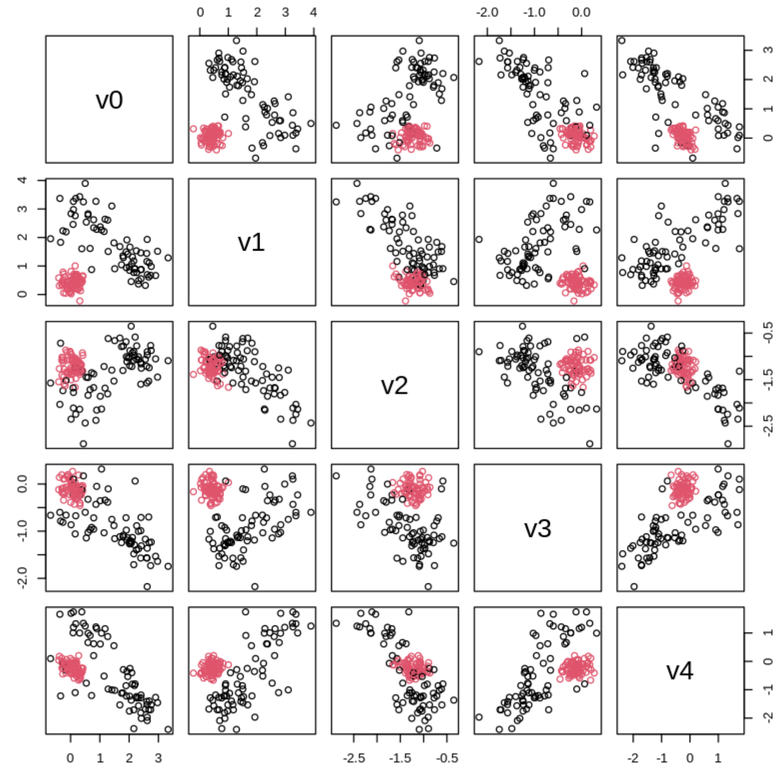
```
km_raw <- kmeans(train_df[, 2:41], 3, nstart = 20)
plot(train_df[, 2:3], col=(km_raw$cluster), main="Clusters, K=3")
```




```
km_raw <- kmeans(train_df[, 2:41], 2, nstart = 20)
plot(train_df[, 2:3], col=(km_raw$cluster), main="Clusters, K=2")
```




```
pairs(train_df[, 2:6], col=(km_raw$cluster))
```




```
km_predict <- function(object, newdata) {

  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)

  return(best_clusters)
}
```

```
test_df$raw_predict <- km_predict(km_raw, test_df[, 2:41])
```

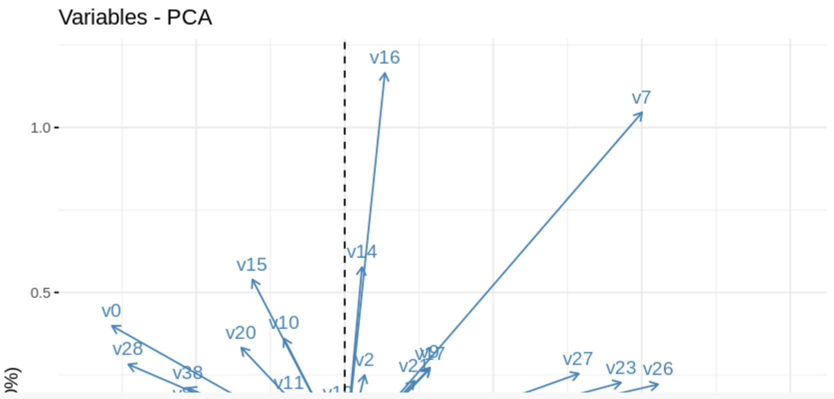
```
test_df
```

A data.frame: 60 × 42

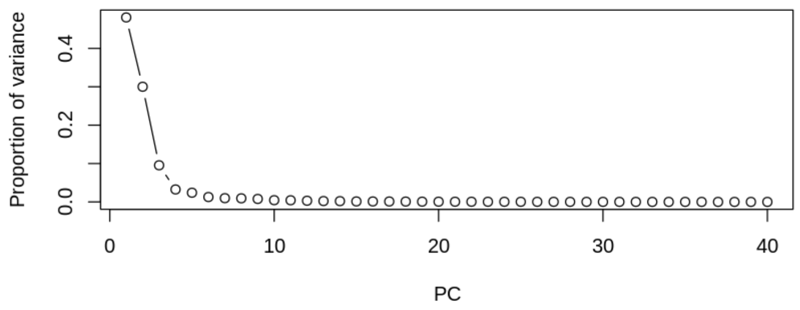
v7	v8	⋯	v31	v32	v33	v34	v35	v36	v37	v38	v39
<dbl>	<dbl>	⋯	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
-1.05373144	1.1944223642	⋯	0.62197065	-0.61060405	-0.59617287	-0.86288226	-1.20323312	-2.11993361	-1.9731960	1.040921926	0.35502931
-1.18136501	1.5888354778	⋯	1.03537953	-0.48806697	-0.86176187	-0.71411574	-1.39863300	-2.04405808	-1.4652017	1.210512280	0.58417004
-2.17573237	0.6086969972	⋯	0.55778617	0.25647852	0.04344850	-0.72558475	-0.76405078	-2.65441179	-1.9175396	0.562621534	-0.10496024
-1.56574440	0.5121214390	⋯	0.58680964	-0.40755525	0.44083628	-0.90433502	-0.06124768	-3.15770626	-1.8261822	0.373324662	-0.25455505
-1.00930250	0.5035853982	⋯	0.35233882	-0.27946404	0.10557976	-0.90751529	0.46662197	-1.76957524	-1.4781467	0.827279627	-0.34023026
-1.28434372	0.2941833735	⋯	0.40800962	-1.03687036	0.47104740	-1.03750670	0.52476645	-2.93649411	-2.0066812	0.855190277	-0.99642324
-1.16771030	0.8821454644	⋯	0.28653064	-0.41048709	-0.18306607	-1.17963159	0.03329696	-2.05090976	-0.7615325	0.918242991	-0.24069342
-1.24335349	-0.0480962284	⋯	0.01087571	-0.19002353	1.16251290	-0.93248010	0.47091782	-2.85473037	-2.0290823	0.709987938	-1.12229550
-1.08313358	0.8890928030	⋯	1.26534820	-0.11416221	-1.18711531	-0.17582756	-2.60877633	-0.64842343	-1.8278294	0.821273029	0.50176686
-1.13457716	0.9637569785	⋯	0.60122377	-0.01108570	-0.80521792	0.36021316	-2.56136131	-1.85802341	-1.7363559	0.938004911	0.35631743
-0.17284107	1.5721820593	⋯	1.09943485	-1.13988173	-1.32700360	-0.99020696	-0.94199955	-1.23188841	-2.2193739	0.778383315	-0.28499559
-1.77891648	0.9099979401	⋯	2.08955002	0.32610139	-0.59073025	0.17579620	-1.73881137	-1.58883023	-1.7269771	0.941369116	-0.32824546
-0.54625678	1.4678690434	⋯	1.27739120	-0.75206608	-1.48408020	-0.67813230	-0.67730749	-0.86977446	-1.3327212	0.932143629	-0.07032982
-3.10790563	0.2468841374	⋯	-0.03066412	0.05352650	0.56382710	-0.23470905	-0.06966221	-3.05481434	-2.4613216	0.708495378	-0.14202645
-1.14924705	1.3662321568	⋯	1.40700507	-0.25780028	-1.19338763	-1.43520916	-1.44012642	-1.61061358	-1.4195974	1.831676722	0.41652146
0.11618628	0.4066341221	⋯	0.59855688	-0.73124373	-0.94860208	-1.24008811	-0.30329436	-1.03524554	-1.3653032	0.451304883	-0.60737747
-3.60719800	0.7924073339	⋯	-0.06092231	1.04434788	0.47763404	-1.33786929	-0.31478640	-2.51232982	-1.4018497	0.231797978	0.51757163
-3.52251387	1.0049568415	⋯	1.30554092	3.03634310	-0.19314130	-0.92495030	-2.84797740	-0.14430602	-0.6637834	0.526974082	1.61614954
-3.94267154	0.1511492282	⋯	0.81007940	1.87298131	-0.60036820	-0.39770821	-1.01433301	-0.67872322	-0.2874364	-0.039798919	0.26750040
-3.70012093	0.5214017630	⋯	0.47986332	1.46297896	-0.16343357	-0.88348353	-0.60384721	-1.06618321	-1.1388642	0.057916693	0.45135173
-4.14000463	0.3240755200	⋯	0.72263205	3.34032869	0.02638787	-0.83636898	-0.13316429	0.42502603	-0.8171065	-0.081681237	0.57422143
-2.95685697	0.4205506742	⋯	0.09887560	1.23891401	-0.19305192	-0.79947549	0.09225606	-0.49399140	-0.3511732	-0.237031534	0.50155306
-3.10952759	0.2274720073	⋯	0.08200826	1.63894439	-0.33149347	-0.73077959	-0.39101630	-0.09722253	-0.4151224	-0.422359794	0.59977776
-3.64597654	0.5440257192	⋯	0.18102543	1.68529975	-0.20135687	-0.84329170	-0.08318584	-0.59342456	-0.4687667	-0.309512466	0.84135985
-2.47609019	0.1930978745	⋯	0.22903132	0.95560950	-0.17013662	-1.24462807	0.34954131	-0.29323909	-0.9651977	-0.391939610	-0.03180460
-1.56680059	0.1714082658	⋯	0.08713070	0.56128538	-0.95431036	1.34680343	-2.86071014	-1.63921189	-0.6453359	-0.455962002	0.50041997
-3.75903440	0.4060684741	⋯	0.79511756	1.72964990	0.13936473	-0.53674257	-0.27814743	-0.74782622	-0.8521619	0.005054687	0.58668321
-2.86661720	0.2128315717	⋯	0.87574303	2.08925986	-0.36771259	0.36322799	-2.53912401	-1.37055933	-0.9139228	0.638175607	0.61342967
-3.95027614	0.6597155333	⋯	1.34077811	2.03062558	-0.47663072	-0.22919703	0.08368511	-0.02944917	-0.2732891	-0.067772761	0.73471916
0.45884442	-0.1975209564	⋯	0.76679707	0.22068323	-1.10768688	-0.03464926	-0.57438552	1.44274151	-1.2801262	-0.461643457	-0.89030820
0.24087717	-0.3547609448	⋯	0.34580866	0.46108323	-1.30383146	-0.22512214	-0.42396328	1.56548023	-1.1377822	-0.676897526	-1.01388299
0.33985013	-0.2217625678	⋯	1.02053356	0.72883791	-1.26958799	-0.44051564	-0.58221412	2.03559637	-1.1074196	-0.479330122	-0.86233038
-0.33829391	-0.2715214193	⋯	1.18061233	1.01035714	-0.99373358	0.18067107	-0.46507454	1.66202581	-1.3328507	-0.186734319	-0.91266978
0.07574807	-0.5763864517	⋯	0.67119253	0.21193330	-0.77107775	-0.11566154	-0.18904334	0.87375456	-1.8319130	-0.461206675	-1.37627351
0.28008971	-0.5012071729	⋯	0.87238854	0.61815822	-0.75598377	0.33539799	-0.51890320	1.57420599	-1.0717559	-0.414887577	-0.69688666

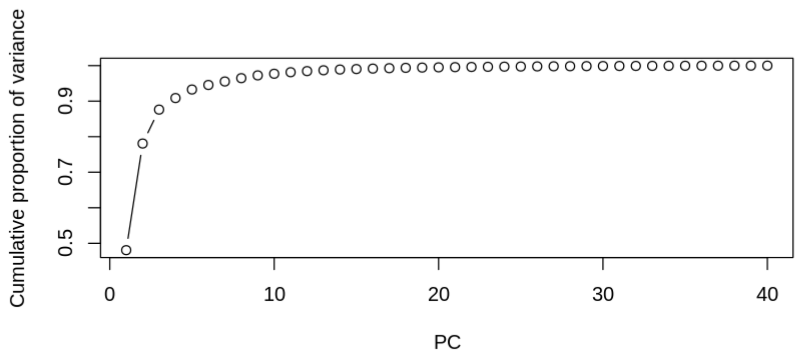
PCA

```
library(factoextra)

pr.out = prcomp(train_df[, 2:41], scale=FALSE) #features have same scale
fviz_pca_var(pr.out, col.var = "steelblue")
```



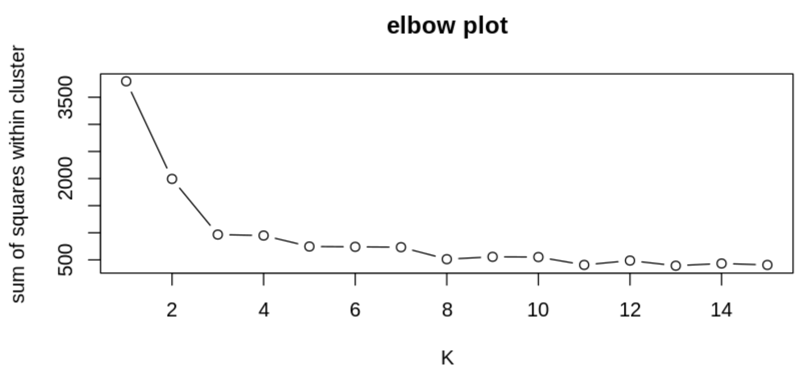

```
pr.var = pr.out$sdev^2
pve = pr.var/sum(pr.var)

par(mfrow = c(2,1))
plot(pve, xlab='PC', ylab='Proportion of variance', type='b')
plot(cumsum(pve), xlab='PC', ylab='Cumulative proportion of variance', type='b')
```

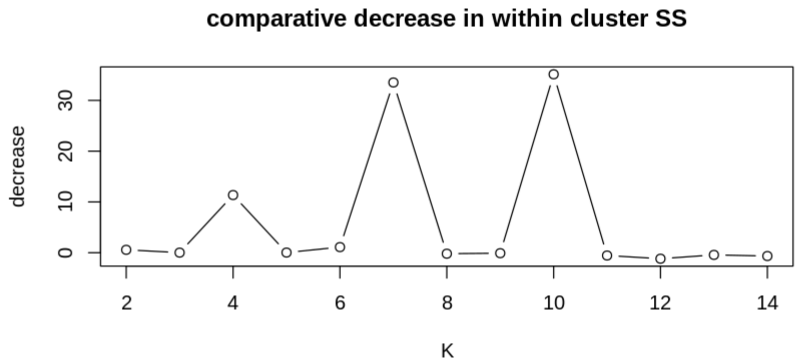




PCA K-means

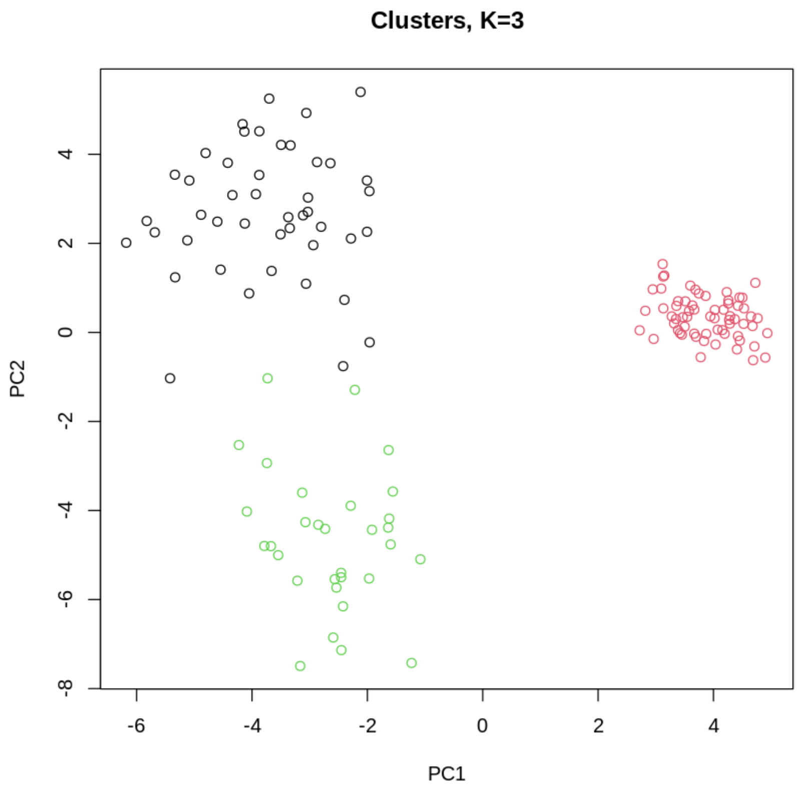
```
km_ss <- function(k) {
  kmk.out <- kmeans(pr.out$x[, 1:7], k)
  return(kmk.out$tot.withinss)
}

K <- 15
par(mfcol = c(2,1))
k_ss <- sapply(1:K, km_ss)
plot(1:K, k_ss, ylab='sum of squares within cluster', xlab='K', main='elbow plot', type='b')
k_ss_compare <- sapply(2:(K-1), function(x) (k_ss[x]-k_ss[x+1])/(k_ss[x-1]-k_ss[x]))
plot(2:(K-1), k_ss_compare, ylab='decrease', main='comparative decrease in within cluster SS', xlab='K', type='b')
```





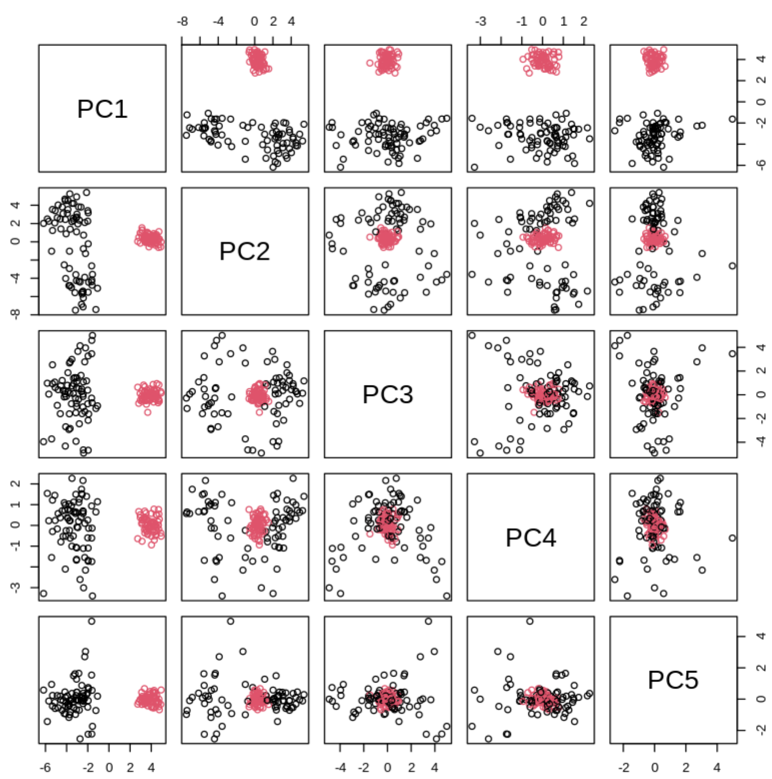

```
km.out=kmeans(pr.out$x[,1:7], 3, nstart = 20)
```

```
plot(pr.out$x[, 1:7], col=(km.out$cluster), main="Clusters, K=3")
```




```
km.out=kmeans(pr.out$x[,1:7], 2, nstart = 20)
plot(pr.out$x[, 1:10], col=(km.out$cluster), main="Clusters, K=2")
```


```
pairs(pr.out$x[, 1:5], col=(km.out$cluster))
```




```
pr_test.out = prcomp(test_df[, 2:41], scale=FALSE) #features have same scale
test_df$pca_predict <- km_predict(km.out, pr_test.out$x[, 1:7])
```