R Notebook

# Download all the libraries rerquied

# Principal Component Analysis

The dataset decathlon contains athletes’ performance during two sporting meetings. The first ten columns corresponds to the performance of the athletes for the 10 events of the decathlon. The columns 11 and 12 correspond respectively to the rank and the points obtained. The last column is a categorical variable corresponding to the sporting event (2004 Olympic Game or 2004 Decastar).

# load the dataset  
data ("decathlon")  
  
# Create a copy of dataset  
decathlon\_copy <- decathlon  
  
# Inspect the dataset  
head(decathlon)

## 100m Long.jump Shot.put High.jump 400m 110m.hurdle Discus Pole.vault  
## SEBRLE 11.04 7.58 14.83 2.07 49.81 14.69 43.75 5.02  
## CLAY 10.76 7.40 14.26 1.86 49.37 14.05 50.72 4.92  
## KARPOV 11.02 7.30 14.77 2.04 48.37 14.09 48.95 4.92  
## BERNARD 11.02 7.23 14.25 1.92 48.93 14.99 40.87 5.32  
## YURKOV 11.34 7.09 15.19 2.10 50.42 15.31 46.26 4.72  
## WARNERS 11.11 7.60 14.31 1.98 48.68 14.23 41.10 4.92  
## Javeline 1500m Rank Points Competition  
## SEBRLE 63.19 291.7 1 8217 Decastar  
## CLAY 60.15 301.5 2 8122 Decastar  
## KARPOV 50.31 300.2 3 8099 Decastar  
## BERNARD 62.77 280.1 4 8067 Decastar  
## YURKOV 63.44 276.4 5 8036 Decastar  
## WARNERS 51.77 278.1 6 8030 Decastar

# Inspect the broader overview of dataset  
skim (decathlon)

Data summary

|  |  |
| --- | --- |
| Name | decathlon |
| Number of rows | 41 |
| Number of columns | 13 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Column type frequency: |  |
| factor | 1 |
| numeric | 12 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |  |
| Group variables | None |

**Variable type: factor**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | ordered | n\_unique | top\_counts |
| Competition | 0 | 1 | FALSE | 2 | Oly: 28, Dec: 13 |

**Variable type: numeric**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| skim\_variable | n\_missing | complete\_rate | mean | sd | p0 | p25 | p50 | p75 | p100 | hist |
| 100m | 0 | 1 | 11.00 | 0.26 | 10.44 | 10.85 | 10.98 | 11.14 | 11.64 | ▃▇▇▅▁ |
| Long.jump | 0 | 1 | 7.26 | 0.32 | 6.61 | 7.03 | 7.30 | 7.48 | 7.96 | ▃▆▇▅▂ |
| Shot.put | 0 | 1 | 14.48 | 0.82 | 12.68 | 13.88 | 14.57 | 14.97 | 16.36 | ▂▅▇▅▂ |
| High.jump | 0 | 1 | 1.98 | 0.09 | 1.85 | 1.92 | 1.95 | 2.04 | 2.15 | ▆▇▃▃▃ |
| 400m | 0 | 1 | 49.62 | 1.15 | 46.81 | 48.93 | 49.40 | 50.30 | 53.20 | ▁▇▆▃▁ |
| 110m.hurdle | 0 | 1 | 14.61 | 0.47 | 13.97 | 14.21 | 14.48 | 14.98 | 15.67 | ▇▃▅▃▂ |
| Discus | 0 | 1 | 44.33 | 3.38 | 37.92 | 41.90 | 44.41 | 46.07 | 51.65 | ▃▆▇▃▃ |
| Pole.vault | 0 | 1 | 4.76 | 0.28 | 4.20 | 4.50 | 4.80 | 4.92 | 5.40 | ▃▃▇▃▂ |
| Javeline | 0 | 1 | 58.32 | 4.83 | 50.31 | 55.27 | 58.36 | 60.89 | 70.52 | ▅▇▇▃▁ |
| 1500m | 0 | 1 | 279.02 | 11.67 | 262.10 | 271.02 | 278.05 | 285.10 | 317.00 | ▆▇▃▂▁ |
| Rank | 0 | 1 | 12.12 | 7.92 | 1.00 | 6.00 | 11.00 | 18.00 | 28.00 | ▇▇▅▃▃ |
| Points | 0 | 1 | 8005.37 | 342.39 | 7313.00 | 7802.00 | 8021.00 | 8122.00 | 8893.00 | ▂▅▇▁▁ |

The above result gives a clear view that the dataset does not have any missing values

# To Stimulate the real world scenario lets divide the dataset into training and test.  
# The training dataset will be 90% and the remaining 10% is test data set  
# Feature Rank and point is ignored   
# First 10 features will be the input and Competition will be the target feature  
  
train\_data <- createDataPartition(decathlon\_copy$Competition, p=0.9, list=FALSE)  
decathlon\_train <- decathlon\_copy[train\_data, ]  
decathlon\_test <- decathlon\_copy[-train\_data, ]  
  
# verify the dimensions of test and train data set created  
dim\_train<- as.vector(dim (decathlon\_train))  
dim\_test<- as.vector(dim (decathlon\_test))  
  
paste0 ("Training dataset has ", dim\_train[1], " records")

## [1] "Training dataset has 38 records"

paste0 ("Test dataset has ", dim\_test[1], " records")

## [1] "Test dataset has 3 records"

# Applying PCA for the first 10 columns and also performing scaling  
# From the proportion of variance we can see that PC1 is having having the highest variance with respect to data  
decathlon\_pca <- prcomp(decathlon\_train[, 1:10], center = TRUE, scale. = TRUE)  
summary(decathlon\_pca)

## Importance of components:  
## PC1 PC2 PC3 PC4 PC5 PC6 PC7  
## Standard deviation 1.8280 1.3037 1.1913 1.0254 0.83540 0.76234 0.69129  
## Proportion of Variance 0.3342 0.1700 0.1419 0.1051 0.06979 0.05812 0.04779  
## Cumulative Proportion 0.3342 0.5041 0.6461 0.7512 0.82099 0.87910 0.92689  
## PC8 PC9 PC10  
## Standard deviation 0.58952 0.46341 0.41087  
## Proportion of Variance 0.03475 0.02147 0.01688  
## Cumulative Proportion 0.96164 0.98312 1.00000

#Displaying the eigen vectors for every PCA and feature  
decathlon\_pca$rotation

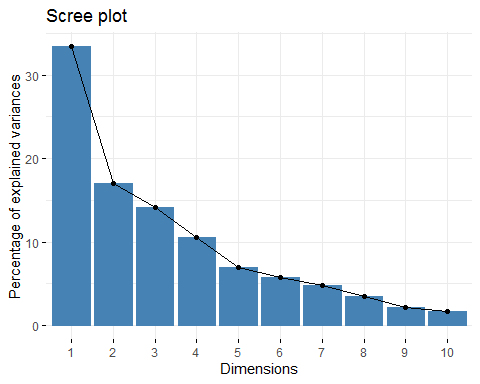
## PC1 PC2 PC3 PC4 PC5  
## 100m 0.424170598 -0.1123345 0.19136071 -0.03934588 0.354483898  
## Long.jump -0.397965480 0.2756193 -0.16588873 0.05431862 0.006644098  
## Shot.put -0.347680943 -0.4514282 0.03426316 0.21095029 0.231184693  
## High.jump -0.323256605 -0.2252864 0.24476917 -0.18473798 0.590684661  
## 400m 0.361904938 -0.4488781 -0.08138943 0.03235594 -0.144743988  
## 110m.hurdle 0.410832520 -0.1802731 0.10159200 0.25426812 0.238636886  
## Discus -0.328507991 -0.4543874 -0.03209255 -0.18185127 -0.073458524  
## Pole.vault -0.007528503 0.1424340 -0.57063255 0.56269749 0.435115651  
## Javeline -0.164684041 -0.1896351 0.35769302 0.69056868 -0.395107836  
## 1500m 0.038371284 -0.4002085 -0.63500802 -0.15681392 -0.207874767  
## PC6 PC7 PC8 PC9 PC10  
## 100m -0.28691333 0.30508662 -0.598347743 -0.12098131 -0.3091614485  
## Long.jump 0.24547815 0.67298052 -0.133407073 0.40001413 -0.2101730276  
## Shot.put -0.27757047 -0.28167430 0.148670108 0.37380063 -0.5079317752  
## High.jump 0.55680748 -0.05139605 -0.068228209 -0.24763904 0.1521396727  
## 400m 0.37416767 -0.07094862 -0.255696980 0.58858275 0.2864708361  
## 110m.hurdle 0.06615465 0.44396182 0.681732921 -0.02364598 -0.0001399003  
## Discus -0.48275285 0.37295526 -0.022122377 -0.06723036 0.5186167787  
## Pole.vault -0.10845691 -0.11506015 -0.151234681 -0.05216896 0.3122258963  
## Javeline 0.16944743 0.09054566 -0.209556843 -0.30792025 -0.0183902850  
## 1500m 0.22899290 0.09679163 -0.002919258 -0.41863471 -0.3609060729

# Extract the Eigen values and identify the number of PC's to be considered for 80% of over all variability  
  
eigen\_values <- get\_eig(decathlon\_pca)  
eigen\_values

## eigenvalue variance.percent cumulative.variance.percent  
## Dim.1 3.3416757 33.416757 33.41676  
## Dim.2 1.6996349 16.996349 50.41311  
## Dim.3 1.4191784 14.191784 64.60489  
## Dim.4 1.0514927 10.514927 75.11982  
## Dim.5 0.6978953 6.978953 82.09877  
## Dim.6 0.5811595 5.811595 87.91037  
## Dim.7 0.4778750 4.778750 92.68911  
## Dim.8 0.3475335 3.475335 96.16445  
## Dim.9 0.2147443 2.147443 98.31189  
## Dim.10 0.1688106 1.688106 100.00000

# From the below result we can confirm that PC1 to PC% should be considered for 80% variability

# Scree plot the vairance with respect to PC  
  
fviz\_eig(decathlon\_pca)



# From the below plot it is very clear that there is very little variability after PC5

# Inspect the variance of the data with respect to PC  
  
result <- get\_pca\_var(decathlon\_pca)  
result

## Principal Component Analysis Results for variables  
## ===================================================  
## Name Description   
## 1 "$coord" "Coordinates for the variables"   
## 2 "$cor" "Correlations between variables and dimensions"  
## 3 "$cos2" "Cos2 for the variables"   
## 4 "$contrib" "contributions of the variables"

# Let identify the correlation between the each variable and the PCi  
  
res\_cor<- result$cor  
res\_cor

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## 100m 0.77539449 -0.1464506 0.22796663 -0.04034617 0.296136312  
## Long.jump -0.72749087 0.3593250 -0.19762203 0.05569957 0.005550488  
## Shot.put -0.63556948 -0.5885272 0.04081746 0.21631332 0.193131995  
## High.jump -0.59092118 -0.2937059 0.29159173 -0.18943461 0.493458739  
## 400m 0.66157130 -0.5852026 -0.09695864 0.03317854 -0.120919317  
## 110m.hurdle 0.75101215 -0.2350222 0.12102582 0.26073243 0.199357567  
## Discus -0.60052084 -0.5923851 -0.03823162 -0.18647451 -0.061367347  
## Pole.vault -0.01376229 0.1856913 -0.67979041 0.57700306 0.363496185  
## Javeline -0.30104656 -0.2472273 0.42611710 0.70812515 -0.330073603  
## 1500m 0.07014367 -0.5217520 -0.75648044 -0.16080063 -0.173658852  
## Dim.6 Dim.7 Dim.8 Dim.9 Dim.10  
## 100m -0.21872498 0.21090181 -0.35273781 -0.05606336 -1.270238e-01  
## Long.jump 0.18713736 0.46522135 -0.07864610 0.18536861 -8.635289e-02  
## Shot.put -0.21160256 -0.19471723 0.08764397 0.17322114 -2.086917e-01  
## High.jump 0.42447559 -0.03552932 -0.04022188 -0.11475721 6.250897e-02  
## 400m 0.28524230 -0.04904572 -0.15073842 0.27275229 1.177010e-01  
## 110m.hurdle 0.05043221 0.30690415 0.40189502 -0.01095767 -5.748023e-05  
## Discus -0.36802092 0.25781838 -0.01304158 -0.03115489 2.130818e-01  
## Pole.vault -0.08268084 -0.07953936 -0.08915583 -0.02417536 1.282829e-01  
## Javeline 0.12917624 0.06259286 -0.12353790 -0.14269183 -7.555937e-03  
## 1500m 0.17457002 0.06691061 -0.00172096 -0.19399749 -1.482839e-01

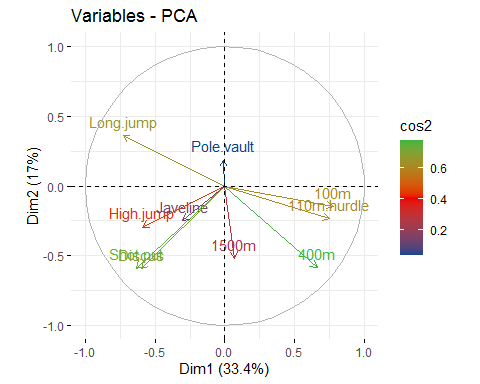
# Identify the proportion of variance between each variance and its PCi  
  
res\_cos2 <- result$cos2  
res\_cos2

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5  
## 100m 0.6012366123 0.02144776 0.051968783 0.001627814 8.769672e-02  
## Long.jump 0.5292429732 0.12911446 0.039054468 0.003102442 3.080792e-05  
## Shot.put 0.4039485644 0.34636424 0.001666065 0.046791453 3.729997e-02  
## High.jump 0.3491878393 0.08626318 0.085025739 0.035885471 2.435015e-01  
## 400m 0.4376765869 0.34246207 0.009400978 0.001100815 1.462148e-02  
## 110m.hurdle 0.5640192444 0.05523542 0.014647248 0.067981401 3.974344e-02  
## Discus 0.3606252833 0.35092007 0.001461657 0.034772743 3.765951e-03  
## Pole.vault 0.0001894007 0.03448124 0.462115005 0.332932528 1.321295e-01  
## Javeline 0.0906290285 0.06112136 0.181575779 0.501441221 1.089486e-01  
## 1500m 0.0049201343 0.27222514 0.572262657 0.025856843 3.015740e-02  
## Dim.6 Dim.7 Dim.8 Dim.9 Dim.10  
## 100m 0.047840617 0.044479574 1.244240e-01 0.0031431008 1.613505e-02  
## Long.jump 0.035020392 0.216430905 6.185210e-03 0.0343615221 7.456821e-03  
## Shot.put 0.044775645 0.037914798 7.681465e-03 0.0300055635 4.355224e-02  
## High.jump 0.180179527 0.001262332 1.617799e-03 0.0131692170 3.907372e-03  
## 400m 0.081363168 0.002405482 2.272207e-02 0.0743938093 1.385354e-02  
## 110m.hurdle 0.002543408 0.094190160 1.615196e-01 0.0001200706 3.303977e-09  
## Discus 0.135439397 0.066470317 1.700828e-04 0.0009706274 4.540387e-02  
## Pole.vault 0.006836121 0.006326510 7.948762e-03 0.0005844482 1.645651e-02  
## Javeline 0.016686501 0.003917866 1.526161e-02 0.0203609597 5.709219e-05  
## 1500m 0.030474694 0.004477029 2.961705e-06 0.0376350246 2.198812e-02

# Let us try to visualize the variance for the first 2 PCi  
result\_plot <- cbind (res\_cos2[,1:2], variability\_explained = rowSums(res\_cos2[,1:2]))  
result\_plot

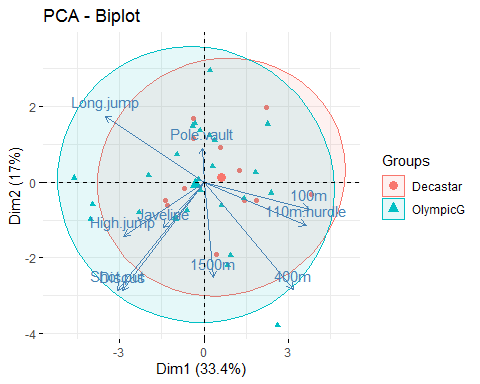
## Dim.1 Dim.2 variability\_explained  
## 100m 0.6012366123 0.02144776 0.62268438  
## Long.jump 0.5292429732 0.12911446 0.65835743  
## Shot.put 0.4039485644 0.34636424 0.75031280  
## High.jump 0.3491878393 0.08626318 0.43545102  
## 400m 0.4376765869 0.34246207 0.78013866  
## 110m.hurdle 0.5640192444 0.05523542 0.61925466  
## Discus 0.3606252833 0.35092007 0.71154535  
## Pole.vault 0.0001894007 0.03448124 0.03467064  
## Javeline 0.0906290285 0.06112136 0.15175039  
## 1500m 0.0049201343 0.27222514 0.27714527

fviz\_pca\_var(decathlon\_pca, col.var = "cos2",gradient.cols = "lancet")



# From the below visualization we observe that variable "400m" has the highest variance wrt the PCi

# Plot all the observation against the first two PCs  
# Colour individuals by the competition type  
  
  
fviz\_pca\_biplot(decathlon\_pca, label="var", habillage = decathlon\_train$Competition, addEllipses = TRUE)



# Creating the updated training dataset with PC1 to PC5 because the overall 80% vraibility reflects within PC1 to PC5  
  
decathlon\_train\_pca\_transformed <- decathlon\_pca$x[ ,1:5] %>%  
 as.data.frame() %>%  
 mutate (Competition = decathlon\_train$Competition)  
  
decathlon\_train\_pca\_transformed

## PC1 PC2 PC3 PC4 PC5  
## CLAY -1.3066062 -0.60334964 -2.14960882 -0.08893746 -1.82045576  
## KARPOV -1.3861340 -0.49402013 -1.95072396 -1.65956611 0.83203547  
## BERNARD 0.5720813 0.92057508 -0.86014722 2.16929814 0.42861355  
## YURKOV 0.4560050 -1.89537170 1.33768708 0.84447765 1.23574261  
## WARNERS -0.3760721 1.67250404 -0.86187497 -0.62043641 0.91315700  
## ZSIVOCZKY -0.3731863 1.15633026 1.11282880 -1.57191968 -0.09644205  
## McMULLEN -0.6911333 -0.17352047 0.35079661 -1.56892782 -0.05555793  
## MARTINEAU 1.8657948 -0.49000070 0.75655012 -0.35285183 1.75801283  
## HERNU 1.4261302 -0.48513735 -0.77426339 0.37647683 -0.16310254  
## BARRAS 1.2498259 0.30486554 -0.00760441 -0.59335942 0.28351734  
## NOOL 2.2315627 1.98389363 1.25604115 -0.01595492 0.58298116  
## BOURGUIGNON 3.8436954 -0.33478172 -1.18231469 0.53330433 0.33649960  
## Sebrle -4.0451794 -0.99559188 0.28982492 1.93843883 0.40326749  
## Clay -3.9607685 -0.58404843 -0.29963301 1.51752789 -1.06441020  
## Karpov -4.6198388 0.11214372 -0.16711027 -1.16769094 0.32559297  
## Macey -2.3179384 -0.80705725 1.74872793 -0.72811987 0.96218821  
## Zsivoczky -1.0170502 -0.94516635 1.45992853 0.77217186 0.86191577  
## Hernu -0.9499671 0.73152478 0.77632210 -0.08508675 0.63249238  
## Nool -0.3116758 1.55936967 -1.39696097 2.17779984 0.07690698  
## Bernard -1.9729989 0.17291756 0.60666276 -1.47026852 0.17009231  
## Schwarzl -0.1276490 1.35726641 -0.89242900 0.41160374 0.23036630  
## Pogorelov -0.5953837 -0.74399845 -1.30803224 -0.48130200 0.89817841  
## Schoenbeck -0.1886958 0.06314679 -0.76461234 1.03233533 -0.74120028  
## Barras -0.1121649 -0.20969743 1.50261885 0.69759845 -0.57959942  
## Smith -1.0063843 -0.98053730 1.49609653 -0.85684977 -1.80456809  
## Averyanov -0.3999490 1.48469393 -0.44758546 -0.05002845 -0.38244612  
## Smirnov 0.3841473 1.10880407 1.09669979 0.55766799 -0.39626995  
## Qi 0.3052332 0.40314553 0.98484568 -0.21531176 -0.63463855  
## Drews 0.2211417 2.94577756 -1.23721346 -0.62080375 -0.26259619  
## Parkhomenko 0.9488951 -1.93021959 1.11377647 1.54653890 0.33997373  
## Terek 0.6360175 -0.62495825 -2.13687356 0.27549225 1.31109556  
## Gomez 0.2013430 1.22338808 1.14422630 0.08709668 -1.17410248  
## Turi 1.4265905 -0.43099597 -0.44285037 -0.17623146 -0.33327075  
## Lorenzo 2.2681250 1.53693847 1.37031041 0.24242868 -0.65023313  
## Karlivans 1.8518346 0.26596303 0.32047926 -1.28811292 0.21685239  
## Korkizoglou 0.8477763 -2.20698103 -2.44762711 -1.04109952 -0.72771500  
## Uldal 2.3935592 -0.28512556 0.42801869 0.02048634 -1.25072251  
## Casarsa 2.6290171 -3.78268896 0.17502325 -0.54788437 -0.66215110  
## Competition  
## CLAY Decastar  
## KARPOV Decastar  
## BERNARD Decastar  
## YURKOV Decastar  
## WARNERS Decastar  
## ZSIVOCZKY Decastar  
## McMULLEN Decastar  
## MARTINEAU Decastar  
## HERNU Decastar  
## BARRAS Decastar  
## NOOL Decastar  
## BOURGUIGNON Decastar  
## Sebrle OlympicG  
## Clay OlympicG  
## Karpov OlympicG  
## Macey OlympicG  
## Zsivoczky OlympicG  
## Hernu OlympicG  
## Nool OlympicG  
## Bernard OlympicG  
## Schwarzl OlympicG  
## Pogorelov OlympicG  
## Schoenbeck OlympicG  
## Barras OlympicG  
## Smith OlympicG  
## Averyanov OlympicG  
## Smirnov OlympicG  
## Qi OlympicG  
## Drews OlympicG  
## Parkhomenko OlympicG  
## Terek OlympicG  
## Gomez OlympicG  
## Turi OlympicG  
## Lorenzo OlympicG  
## Karlivans OlympicG  
## Korkizoglou OlympicG  
## Uldal OlympicG  
## Casarsa OlympicG

# Predict the test result for the PCA mode created   
  
decathlon\_test\_pca <- predict(decathlon\_pca, decathlon\_test[, 1:10])  
  
decathlon\_test\_pca\_transformed <- decathlon\_test\_pca[,1:5] %>%  
 as.data.frame() %>%  
 mutate(Competition = decathlon\_test$Competition)  
  
decathlon\_test\_pca\_transformed

## PC1 PC2 PC3 PC4 PC5 Competition  
## SEBRLE -0.8414152 -0.6182834 -0.7405037 1.0902417 0.5776087 Decastar  
## Warners -2.1690268 1.8050182 -1.0327088 -0.2673807 -0.1676260 OlympicG  
## Ojaniemi -0.4437401 0.7833132 0.2515883 0.5868240 -0.4518257 OlympicG