Projet Final R

Victor Simonin & Alexandre Lemonnier

Dataset : `decathlon.csv`

Import des bibliothèques

Pour commencer il est nécessaire d'importer les bibliothèques qui nous permettront d'utiliser leurs fonctions par la suite du projet

In [89]:
```
library("VIM") # Pour la visualisation des valeurs manquantes
library("corrplot") # Pour les matrices de corrélation.
library("factoextra") # Pour les graphes de l'ACP
library("FactoMineR")
library("PerformanceAnalytics") # Diagramme de dispersion
library("pander") # Pour la régression logistique
```

Lecture du dataset

In [90]:
```
df = read.table('data/decathlon.csv',header = T ,sep = ';', dec = '.', row.names = 1)
head(df)
```

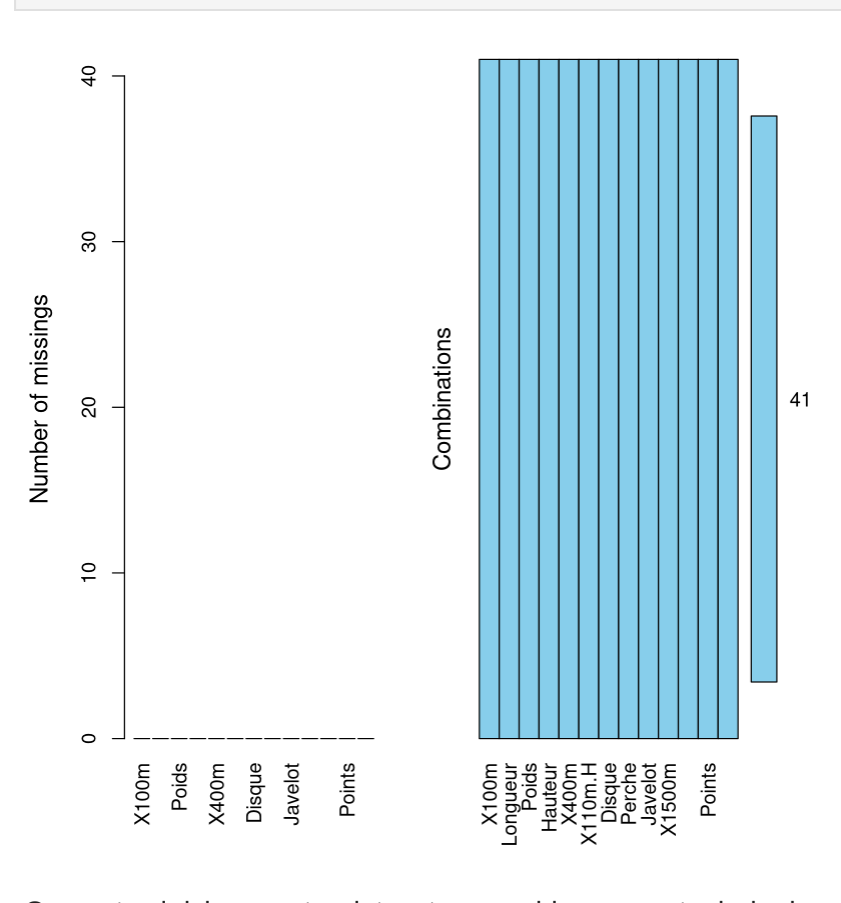
A data.frame: 6 × 13

	X100m	Longueur	Poids	Hauteur	X400m	X110m.H	Disque	Perche	Javelot	X1500m	Classement	Points	Competition
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<chr>
Sebrle	10.85	7.84	16.36	2.12	48.36	14.05	48.72	5.0	70.52	280.01	1	8893	JO
Clay	10.44	7.96	15.23	2.06	49.19	14.13	50.11	4.9	69.71	282.00	2	8820	JO
Karpov	10.50	7.81	15.93	2.09	46.81	13.97	51.65	4.6	55.54	278.11	3	8725	JO
Macey	10.89	7.47	15.73	2.15	48.97	14.56	48.34	4.4	58.46	265.42	4	8414	JO
Warners	10.62	7.74	14.48	1.97	47.97	14.01	43.73	4.9	55.39	278.05	5	8343	JO
Zsivoczky	10.91	7.14	15.31	2.12	49.40	14.95	45.62	4.7	63.45	269.54	6	8287	JO

Le dataset ci-dessus est composé de plusieurs données assez différentes. Celles-ci représentent les scores obtenues par des athlètes de l'épreuve du decathlon au **Jeux Olympiques** et au **Decastar**, donc les scores des 10 épreuves ainsi que leur classement et leurs points.

Visualisation des données manquantes

In [91]:
```
missdata = aggr(df, prop = F, number = T)
```



On peut voir ici que notre dataset ne semble pas contenir de données manquantes, ce qui facilite notre analyse et nous permet de continuer sereinement. Deux choix étaient possibles si jamais le dataset contenait des données manquantes, soit la suppression des lignes qui ne sont pas complètes, ou alors il est possible d'inférer ces données manquantes par plusieurs algorithmes.

Il faudra néanmoins vérifier qu'il n'y a aucune valeures aberrantes.

Description du dataset

Cette section va nous servir à comprendre un peu mieux notre dataset, notamment sur sa taille, ses attributs et quelques statistiques descriptives sur les colonnes.

En premier lieu, nous allons passer la colonne `Compétition` qui est une colonne catégorique en colonne numérique.

In [92]:
```
table(df$Competition)
competition = df$competition
```

```
Decastar       JO
      13       28
```

On peut voir ici que notre dataset contient les résultats de deux compétitions, les `JO` et le `Decastar`, on va donc simplement remplacer ces deux valeurs par des valeurs numériques en suivant : `JO` = 1 et `Decastar` = 2.

In [93]:
```
df[1:28, 13] = 1
df[29:41, 13] = 2
df$Competition = as.numeric(as.character(df$Competition))
```

In [94]:
```
df
```

A data.frame: 41 × 13

	X100m	Longueur	Poids	Hauteur	X400m	X110m.H	Disque	Perche	Javelot	X1500m	Classement	Points	Competition
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<int>	<int>	<dbl>
Sebrle	10.85	7.84	16.36	2.12	48.36	14.05	48.72	5.00	70.52	280.01	1	8893	1
Clay	10.44	7.96	15.23	2.06	49.19	14.13	50.11	4.90	69.71	282.00	2	8820	1
Karpov	10.50	7.81	15.93	2.09	46.81	13.97	51.65	4.60	55.54	278.11	3	8725	1
Macey	10.89	7.47	15.73	2.15	48.97	14.56	48.34	4.40	58.46	265.42	4	8414	1
Warners	10.62	7.74	14.48	1.97	47.97	14.01	43.73	4.90	55.39	278.05	5	8343	1
Zsivoczky	10.91	7.14	15.31	2.12	49.40	14.95	45.62	4.70	63.45	269.54	6	8287	1
Hernu	10.97	7.19	14.65	2.03	48.73	14.25	44.72	4.80	57.76	264.35	7	8237	1
Nool	10.80	7.53	14.26	1.88	48.81	14.80	42.05	5.40	61.33	276.33	8	8235	1
Bernard	10.69	7.48	14.80	2.12	49.13	14.17	44.75	4.40	55.27	276.31	9	8225	1
Schwarzl	10.98	7.49	14.01	1.94	49.76	14.25	42.43	5.10	56.32	273.56	10	8102	1
Pogorelov	10.95	7.31	15.10	2.06	50.79	14.21	44.60	5.00	53.45	287.63	11	8084	1
Schoenbeck	10.90	7.30	14.77	1.88	50.30	14.34	44.41	5.00	60.89	278.82	12	8077	1
Barras	11.14	6.99	14.91	1.94	49.41	14.37	44.83	4.60	64.55	267.09	13	8067	1
Smith	10.85	6.81	15.24	1.91	49.27	14.01	49.02	4.20	61.52	272.74	14	8023	1
Averyanov	10.55	7.34	14.44	1.94	49.72	14.39	39.88	4.80	54.51	271.02	15	8021	1
Ojaniemi	10.68	7.50	14.97	1.94	49.12	15.01	40.35	4.60	59.26	275.71	16	8006	1
Smirnov	10.89	7.07	13.88	1.94	49.11	14.77	42.47	4.70	60.88	263.31	17	7993	1
Qi	11.06	7.34	13.55	1.97	49.65	14.78	45.13	4.50	60.79	272.63	18	7934	1
Drews	10.87	7.38	13.07	1.88	48.51	14.01	40.11	5.00	51.53	274.21	19	7926	1
Parkhomenko	11.14	6.61	15.69	2.03	51.04	14.88	41.90	4.80	65.82	277.94	20	7918	1
Terek	10.92	6.94	15.15	1.94	49.56	15.12	45.62	5.30	50.62	290.36	21	7893	1
Gomez	11.08	7.26	14.57	1.85	48.61	14.41	40.95	4.40	60.71	269.70	22	7865	1
Turi	11.08	6.91	13.62	2.03	51.67	14.26	39.83	4.80	59.34	290.01	23	7708	1
Lorenzo	11.10	7.03	13.22	1.85	49.34	15.38	40.22	4.50	58.36	263.08	24	7592	1
Karlivans	11.33	7.26	13.30	1.97	50.54	14.98	43.34	4.50	52.92	278.67	25	7583	1
Korkizoglou	10.86	7.07	14.81	1.94	51.16	14.96	46.07	4.70	53.05	317.00	26	7573	1
Uldal	11.23	6.99	13.53	1.85	50.95	15.09	43.01	4.50	60.00	281.70	27	7495	1
Casarsa	11.36	6.68	14.92	1.94	53.20	15.39	48.66	4.40	58.62	296.12	28	7404	1
SEBRLE	11.04	7.58	14.83	2.07	49.81	14.69	43.75	5.02	63.19	291.70	1	8217	2
CLAY	10.76	7.40	14.26	1.86	49.37	14.05	50.72	4.92	60.15	301.50	2	8122	2
KARPOV	11.02	7.30	14.77	2.04	48.37	14.09	48.95	4.92	50.31	300.20	3	8099	2
BERNARD	11.02	7.23	14.25	1.92	48.93	14.99	40.87	5.32	62.77	280.10	4	8067	2
YURKOV	11.34	7.09	15.19	2.10	50.42	15.31	46.26	4.72	63.44	276.40	5	8036	2
WARNERS	11.11	7.60	14.31	1.98	48.68	14.23	41.10	4.92	51.77	278.10	6	8030	2
ZSIVOCZKY	11.13	7.30	13.48	2.01	48.62	14.17	45.67	4.42	55.37	268.00	7	8004	2
McMULLEN	10.83	7.31	13.76	2.13	49.91	14.38	44.41	4.42	56.37	285.10	8	7995	2
MARTINEAU	11.64	6.81	14.57	1.95	50.14	14.93	47.60	4.92	52.33	262.10	9	7802	2
HERNU	11.37	7.56	14.41	1.86	51.10	15.06	44.99	4.82	57.19	285.10	10	7733	2
BARRAS	11.33	6.97	14.09	1.95	49.48	14.48	42.10	4.72	55.40	282.00	11	7708	2
NOOL	11.33	7.27	12.68	1.98	49.20	15.29	37.92	4.62	57.44	266.60	12	7651	2
BOURGUIGNON	11.36	6.80	13.46	1.86	51.16	15.67	40.49	5.02	54.68	291.70	13	7313	2

Nos compétitions ont bien été modifié, ainsi que le type de la colonne, on a donc maintenant uniquement des colonnes numériques dans notre dataset.

In [95]:
```
dim(df)
```

41 · 13

On comprend donc ici que notre dataset se compose de 41 individus qui ont chacun 13 données dans les colonnes qui servent à les qualifier et à les décrire.

In [96]:
```
attributes(df)
```

$names 'X100m' · 'Longueur' · 'Poids' · 'Hauteur' · 'X400m' · 'X110m.H' · 'Disque' · 'Perche' · 'Javelot' · 'X1500m' · 'Classement' · 'Points' · 'Competition'

$row.names 'Sebrle' · 'Clay' · 'Karpov' · 'Macey' · 'Warners' · 'Zsivoczky' · 'Hernu' · 'Nool' · 'Bernard' · 'Schwarzl' · 'Pogorelov' · 'Schoenbeck' · 'Barras' · 'Smith' · 'Averyanov' · 'Ojaniemi' · 'Smirnov' · 'Qi' · 'Drews' · 'Parkhomenko' · 'Terek' · 'Gomez' · 'Turi' · 'Lorenzo' · 'Karlivans' · 'Korkizoglou' · 'Uldal' · 'Casarsa' · 'SEBRLE' · 'CLAY' · 'KARPOV' · 'BERNARD' · 'YURKOV' · 'WARNERS' · 'ZSIVOCZKY' · 'McMULLEN' · 'MARTINEAU' · 'HERNU' · 'BARRAS' · 'NOOL' · 'BOURGUIGNON'

$class 'data.frame'

On retrouve donc les différents attributs de notre dataset avec les noms des différentes colonnes qui représentent les épreuves des **Jeux Olympiques** et du **Decastar**. \ Les lignes représentent les individus et donc les scores des athlètes participant à ces compétitions.

In [97]:
```
summary(df)
```

```
     X100m          Longueur         Poids          Hauteur          X400m      
 Min.   :10.44   Min.   :6.61   Min.   :12.68   Min.   :1.850   Min.   :46.81  
 1st Qu.:10.85   1st Qu.:7.03   1st Qu.:13.88   1st Qu.:1.920   1st Qu.:48.93  
 Median :10.98   Median :7.30   Median :14.57   Median :1.950   Median :49.40  
 Mean   :11.00   Mean   :7.26   Mean   :14.48   Mean   :1.977   Mean   :49.62  
 3rd Qu.:11.14   3rd Qu.:7.48   3rd Qu.:14.97   3rd Qu.:2.040   3rd Qu.:50.30  
 Max.   :11.64   Max.   :7.96   Max.   :16.36   Max.   :2.150   Max.   :53.20  
    X110m.H          Disque          Perche          Javelot     
 Min.   :13.97   Min.   :37.92   Min.   :4.200   Min.   :50.31  
 1st Qu.:14.21   1st Qu.:41.90   1st Qu.:4.500   1st Qu.:55.27  
 Median :14.48   Median :44.41   Median :4.800   Median :58.36  
 Mean   :14.61   Mean   :44.33   Mean   :4.762   Mean   :58.32  
 3rd Qu.:14.98   3rd Qu.:46.07   3rd Qu.:4.920   3rd Qu.:60.89  
 Max.   :15.67   Max.   :51.65   Max.   :5.400   Max.   :70.52  
     X1500m        Classement        Points      Competition   
 Min.   :262.1   Min.   : 1.00   Min.   :7313   Min.   :1.000  
 1st Qu.:271.0   1st Qu.: 6.00   1st Qu.:7802   1st Qu.:1.000  
 Median :278.1   Median :11.00   Median :8021   Median :1.000  
 Mean   :279.0   Mean   :12.12   Mean   :8005   Mean   :1.317  
 3rd Qu.:285.1   3rd Qu.:18.00   3rd Qu.:8122   3rd Qu.:2.000  
 Max.   :317.0   Max.   :28.00   Max.   :8893   Max.   :2.000  
```

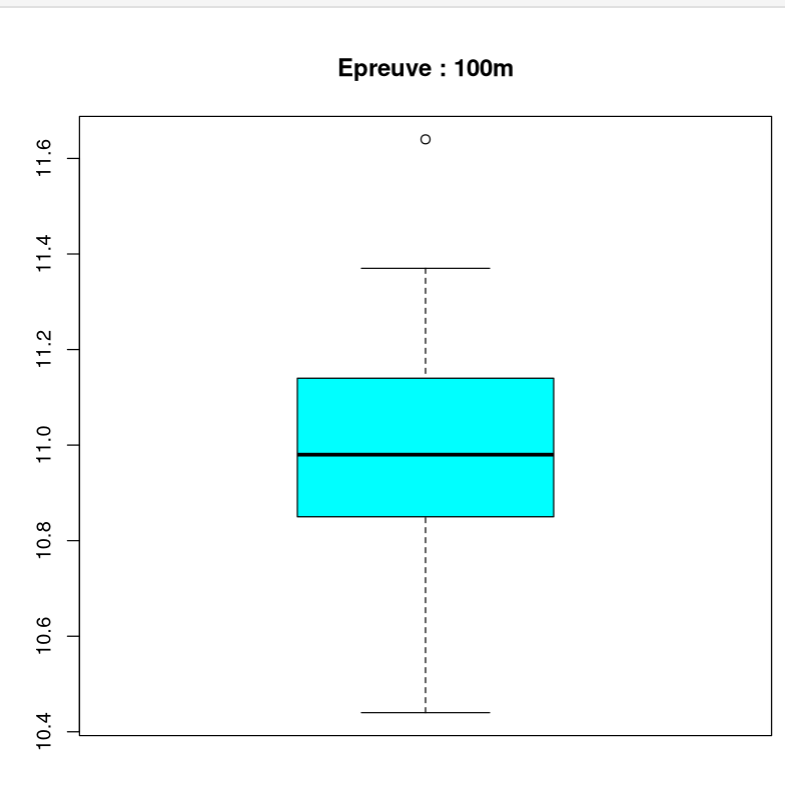
La commande `summary` nous permet d'afficher quelques informations statistiques intéressantes sur nos données.

On retrouve les calculs suivant :

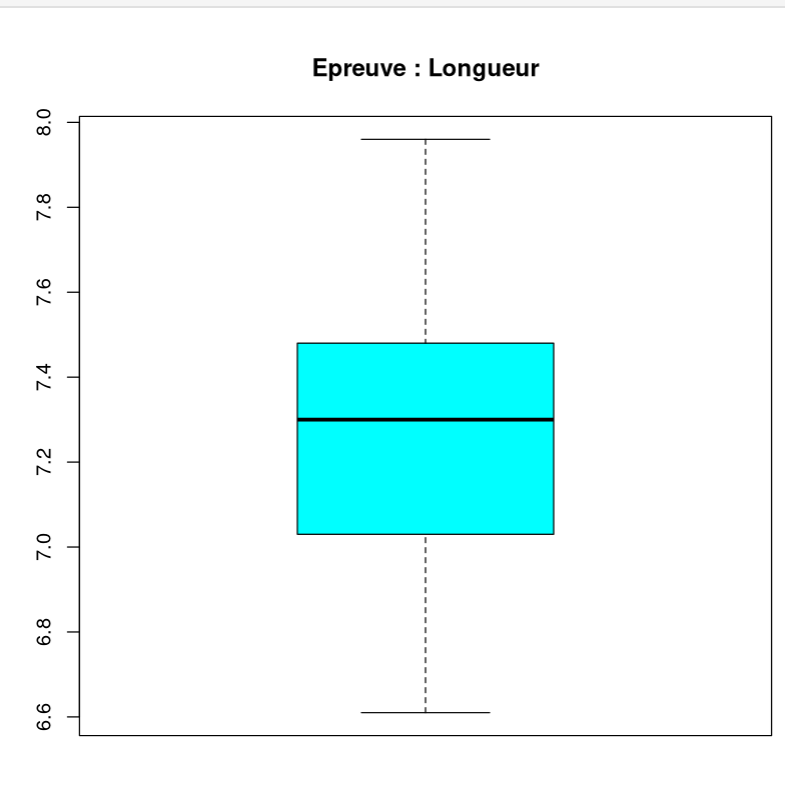
- Minimum
- 1er quartile
- Médiane
- Moyenne
- 3ème quartile
- Maximum

Cela nous permet de comprendre un peu mieux notre dataset, et aussi de commencer à repérer certains outliers et valeurs aberrantes. \ Ici les colonnes avec les valeurs les plus dispersés sont le **javelot** et le **1500m**.\ Les statistiques sur le classement et la compétition ne sont pas intéressantes.

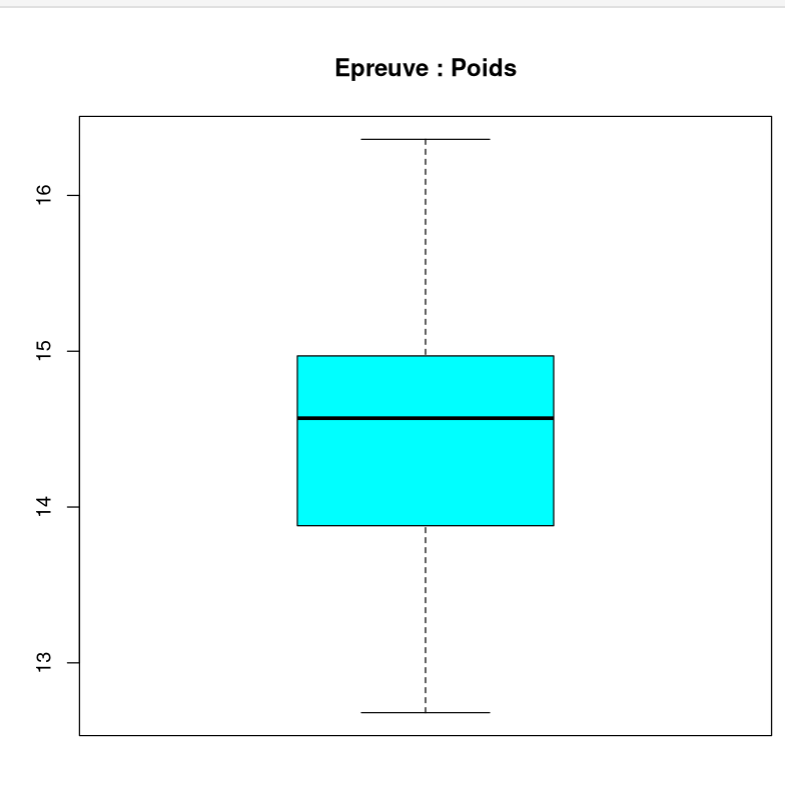
In [98]:
```
boxplot(df$X100m, col ='cyan', main = "Epreuve : 100m")
```



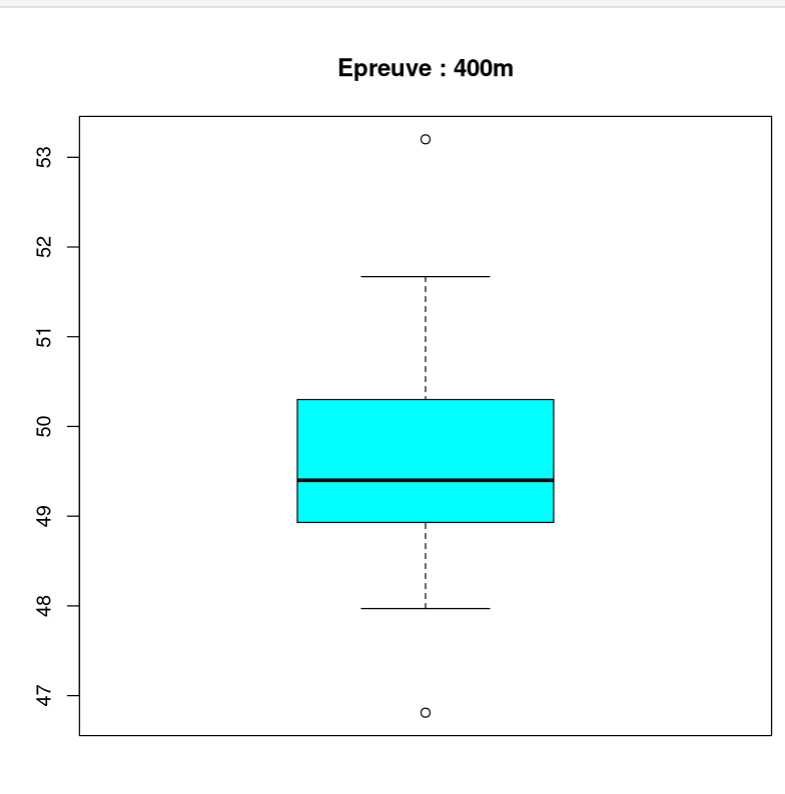
In [99]:
```
boxplot(df$Longueur, col ='cyan', main = "Epreuve : Longueur")
```



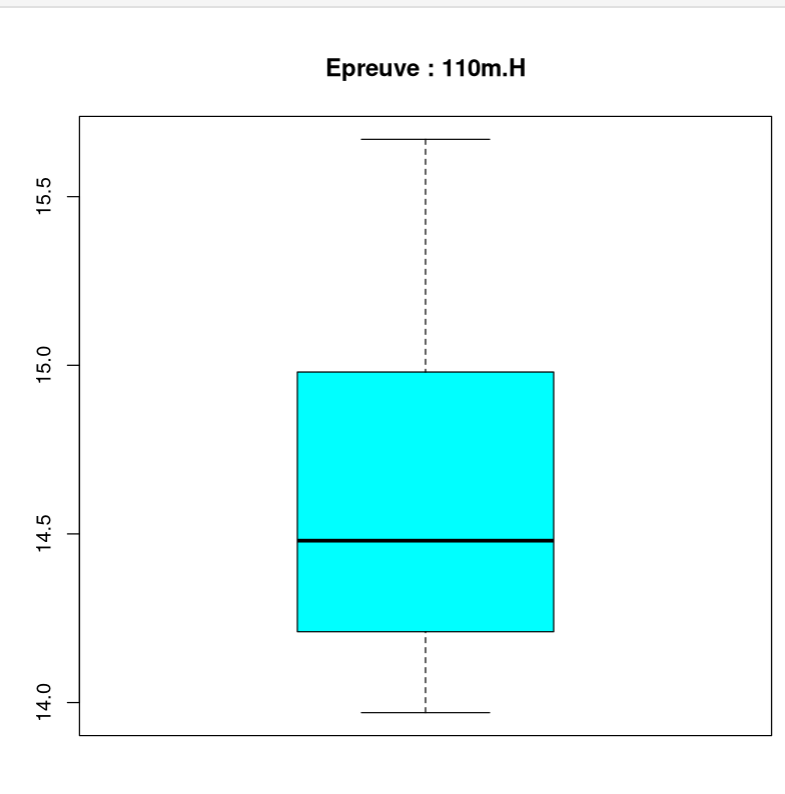
In [100…
```
boxplot(df$Poids, col ='cyan', main = "Epreuve : Poids")
```



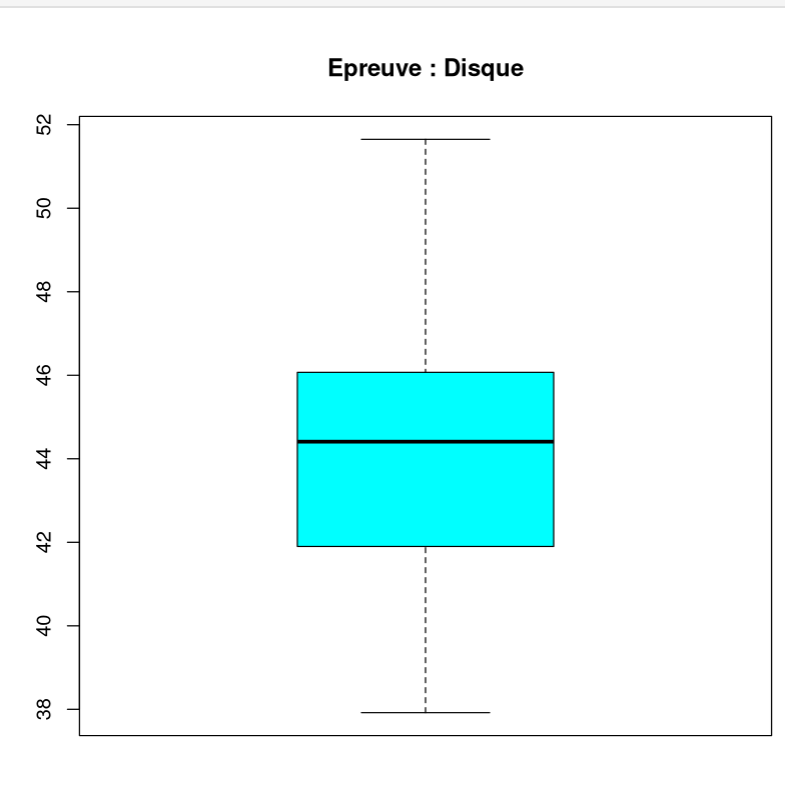
In [101…
```
boxplot(df$X400m, col ='cyan', main = "Epreuve : 400m")
```



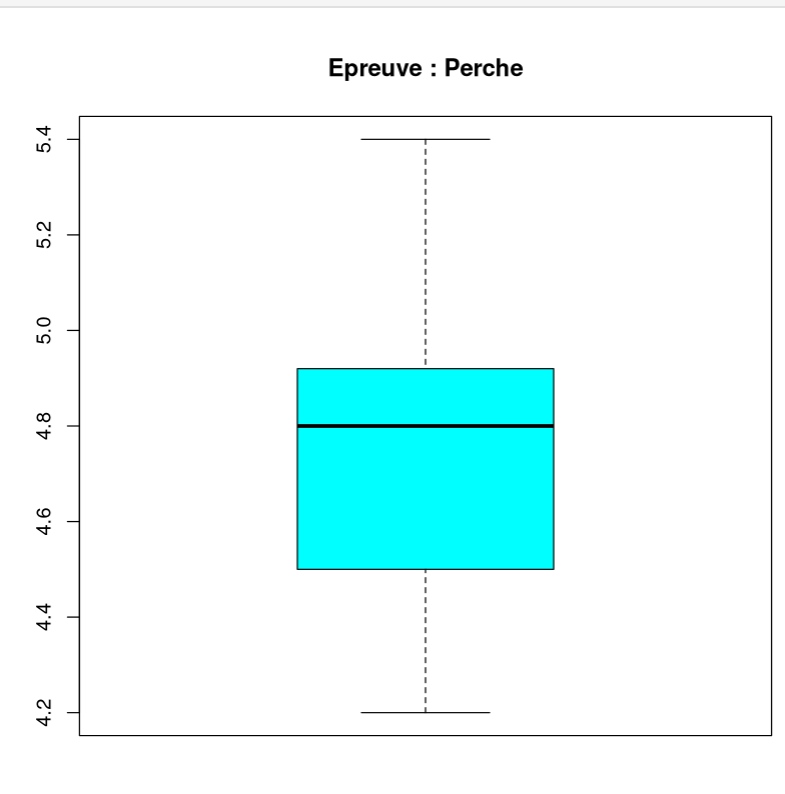
In [102…
```
boxplot(df$X110m.H, col ='cyan', main = "Epreuve : 110m.H")
```



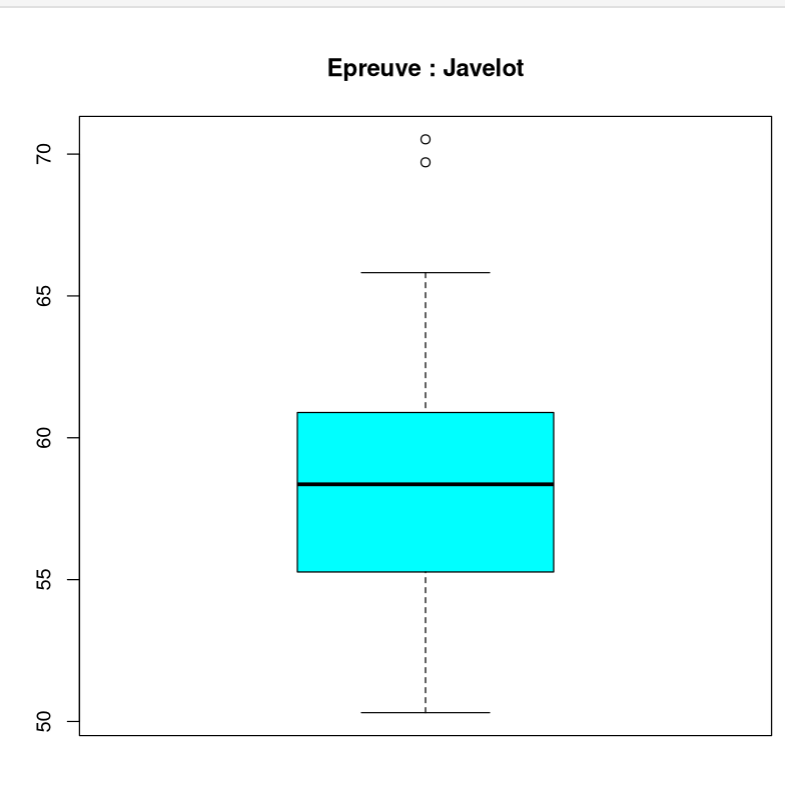
In [103…
```
boxplot(df$Disque, col ='cyan', main = "Epreuve : Disque")
```



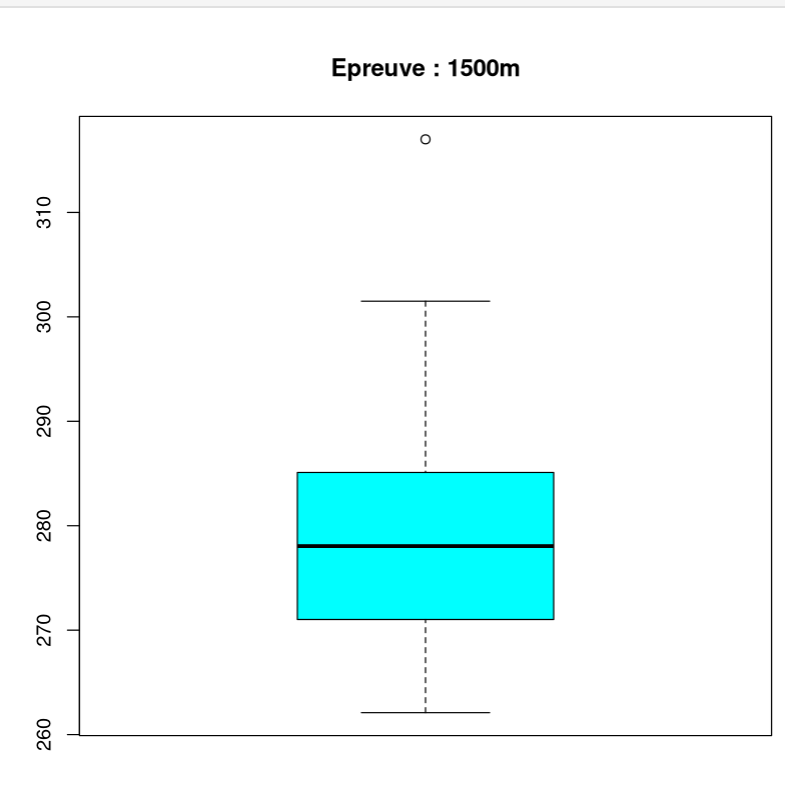
In [104…
```
boxplot(df$Perche, col ='cyan', main = "Epreuve : Perche")
```



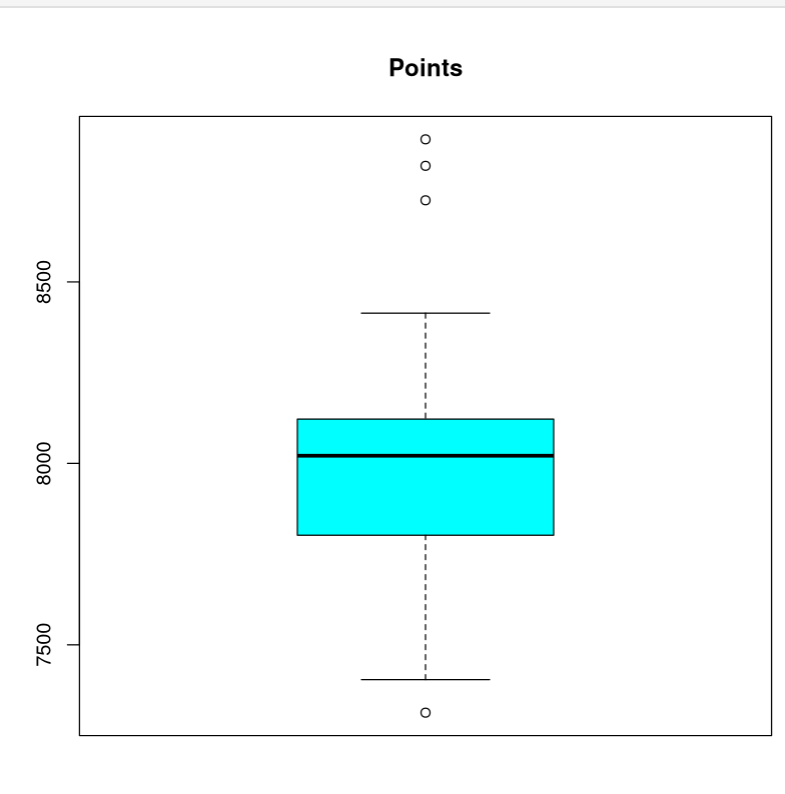
In [105…
```
boxplot(df$Javelot, col ='cyan', main = "Epreuve : Javelot")
```



In [106…
```
boxplot(df$X1500m, col ='cyan', main = "Epreuve : 1500m")
```

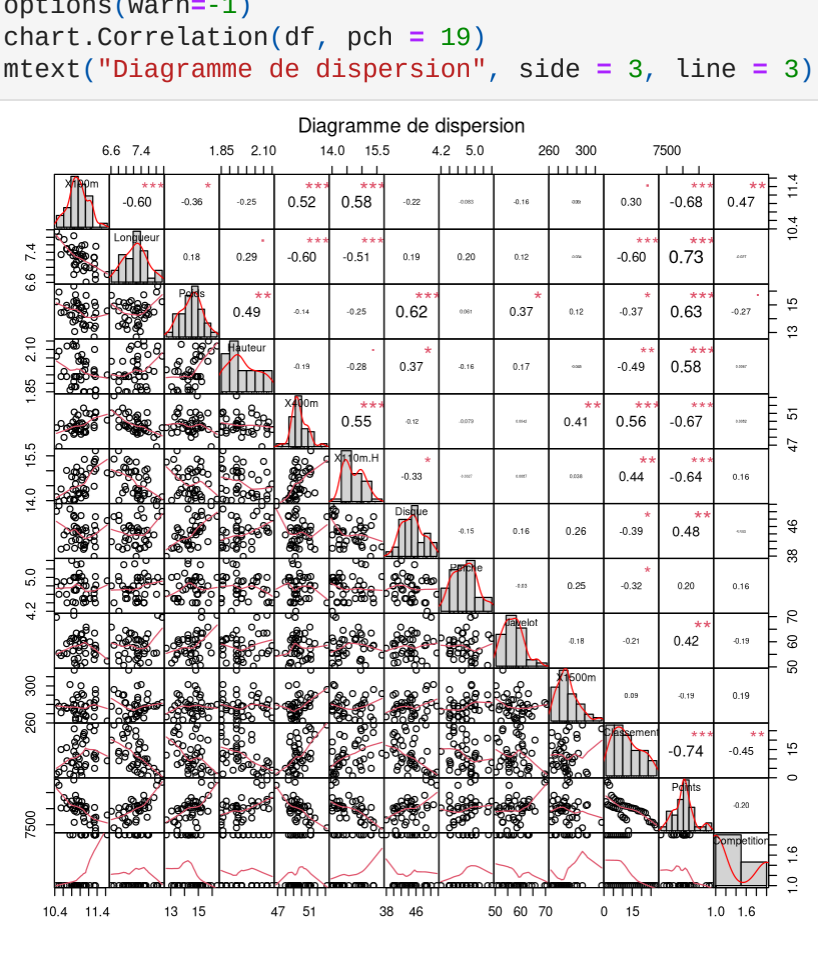


In [107…
```
boxplot(df$Points, col ='cyan', main = "Points")
```



Grâce aux boxplots, il est possible de mieux comprendre la dispersion de nos données dans les différentes colonnes.\ On voit ici comme avec le `summary` que la plupart des données ne présentent pas d'anomalies. Les seules données qui pourraient soulever des questions sont dans le **400m**, le **javelot** ou le **1500m**, mais celles-ci restent dans des normes tout à fait acceptable.

In [108…
```
options(warn=-1)
chart.Correlation(df, pch = 19)
mtext("Diagramme de dispersion", side = 3, line = 3)
```



On retrouve ici le **Diagramme de dispersion** appliqué sur nos données ainsi que les valeurs de corrélations entre les différentes colonnes.

Il est notamment assez utile pour retrouver des applications linéaires entre les colonnes de nos données.

Nos observations sont les suivantes :

- La ligne de la colonne `Points` est linéairement corrélée à toutes les autres variables, que ce soit négativement ou positivement. Cela est logique puisque plus le résultats d'une épreuve est élevé (ou faible si c'est une course) plus le nombre de points est élevé aussi.
- On remarque que la valeur de l'épreuve `Poids` est fortement corrélée a celle de l'épreuve `Disque`.

ACP

On va maintenant effectuer une **ACP** (Analyse en Composantes Principales).\

Celle-ci nous permet d'analyser et de visualiser notre jeu de données `Decathlon` qui contient des individus décrits par plusieurs variables quantitatives.

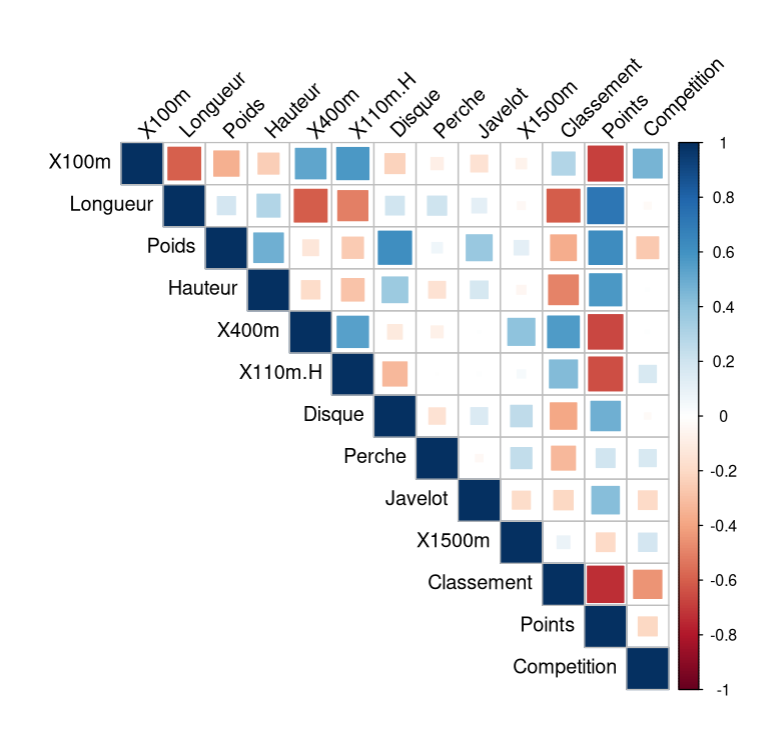
L'**ACP** est une méthode qui nous permet d'explorer des données avec plusieurs variables. Chaque variable pourrait être considérée comme une dimension différente. Cela est utile car il pourrait être très difficile de visualiser nos données dans une "hyper-espace" multidimensionnelle.

L'analyse en composantes principales est utilisée pour extraire et de visualiser les informations importantes contenues dans une table de données multivariées. L'**ACP** synthétise cette information en seulement quelques nouvelles variables appelées composantes principales. Ces nouvelles variables correspondent à une combinaison linéaire des variables originels. Le nombre de composantes principales est inférieur ou égal au nombre de variables d'origine.

L'information contenue dans un jeu de données correspond à la variance ou l'inertie totale qu'il contient. L'objectif de l'**ACP** est d'identifier les directions (les axes principaux ou composantes principales) le long desquelles la variation des données est maximale.

En d'autres termes, l'**ACP** réduit les dimensions d'une donnée multivariée à deux ou trois composantes principales, qui peuvent être visualisées graphiquement, en perdant le moins possible d'information.

In [109…
```
matrice_cor = cor(df)
corrplot(matrice_cor, type="upper", order="original", tl.col="black", tl.srt=45, method = "square")
```



On confirme nos observations précédentes dans ce graphe de corrélation.

On peut ajouter que :

- les résultats des épreuves de courses courtes, donc `X100m`, `X400m` et `X110m.H`, sont fortement corrélés entre elles. Assez étonnamment elles sont cependant négativement corrélées avec le classement final, contrairement aux épreuves de `Longueur`, `Poids` et `Hauteur`.
- l'épreuve de `Longueur` et celle du `X400m` sont assez fortement négativement corrélées.

Scaling des données

Pour passer aux étapes suivantes, il est nécessaire de normaliser nos données, ce qui est fait avec la fonction `scale` et qui nous permet d'obtenir un dataset normalisé.

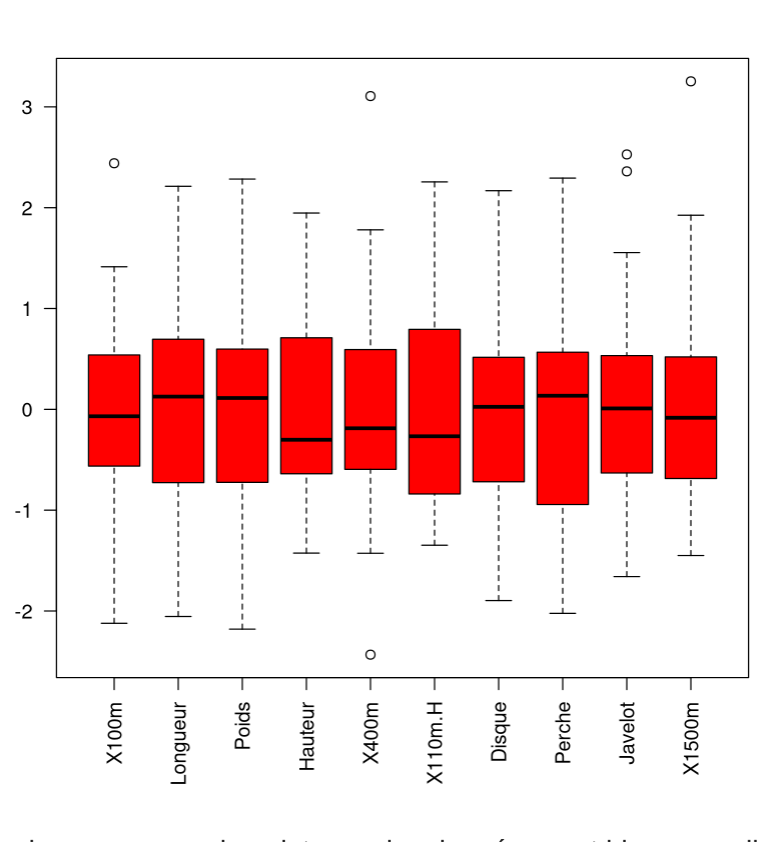
In [110…
```
df_scaled = scale(df[1:10], center = TRUE, scale = TRUE)
df_scaled = cbind(df_scaled, df[c("Classement", "Points", "Competition")])
```

In [111…
```
head(df_scaled)
```

A data.frame: 6 × 13

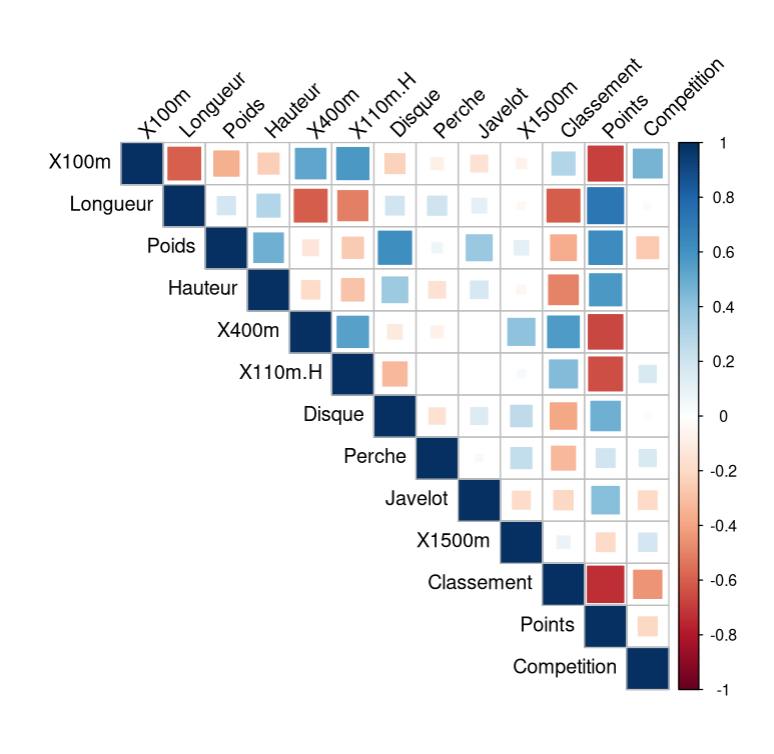
	X100m	Longueur	Poids	Hauteur	X400m	X110m.H	Disque	Perche	Javelot	X1500m	Class
	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	
Sebrle	-0.5628739	1.8331131	2.283919585	1.60955474	-1.0892025	-1.17818270	1.3009450	0.8545364	2.52825135	0.08439142	
Clay	-2.1216730	2.2123779	0.913271989	0.93502243	-0.3696226	-1.00861538	1.7124500	0.4948240	2.36043901	0.25486670	
Karpov	-1.8935560	1.7382969	1.762345721	1.27228858	-2.4329962	-1.34775002	2.1683620	-0.5843134	-0.57524110	-0.07837391	
Macey	-0.4107959	0.6637134	1.519753226	1.94682089	-0.5603546	-0.09719103	1.1884472	-1.3037383	0.02971203	-1.16547502	
Warners	-1.4373221	1.5170591	0.003550134	-0.07677604	-1.4273183	-1.26296636	-0.1763283	0.4948240	-0.60631746	-0.08351387	
Zsivoczky	-0.3347570	-0.3792648	1.010308987	1.60955474	-0.1875602	0.72944966	0.3832000	-0.2246009	1.06351893	-0.81253124	

In [112…
```
boxplot(df_scaled[1:10], col = 'red', las=2)
```



On observe par ces boxplots que les données sont bien normalisées.

In [113…
```
matrice_cor = cor(df_scaled)
corrplot(matrice_cor, type="upper", order="original", tl.col="black", tl.srt=45, method = "square")
```



On observe que la normalisation des données n'impacte pas la corrélation.

Réalisation de l'ACP

On effectue ici l'ACP avec la fonction prcomp().

In [114…
```
res_acp = prcomp(df_scaled[1:10], scale = FALSE)
res_acp
```

```
Standard deviations (1, .., p=10):
 [1] 1.8088409 1.3180027 1.1852918 1.0280323 0.8275104 0.7741245 0.6717405
 [8] 0.6299814 0.4634812 0.4268811

Rotation (n x k) = (10 x 10):
                PC1        PC2         PC3         PC4         PC5         PC6
X100m     0.42829627  0.1419891 -0.15557953 -0.03678703 -0.36518741  0.29607739
Longueur -0.41015201 -0.2620794  0.15372674  0.09901016 -0.04432336 -0.30612478
Poids    -0.34414444  0.4539470 -0.01972378  0.18539458 -0.13431954  0.30547299
Hauteur  -0.31619436  0.2657761 -0.21894349 -0.13189684 -0.67121760 -0.46777116
X400m     0.37571570  0.4320460  0.11091758  0.02850297  0.10597034 -0.33252178
X110m.H   0.41255442  0.1735910 -0.07815576  0.28290068 -0.19857266 -0.09963776
Disque   -0.30542571  0.4600244  0.03623770 -0.25259074  0.12667770  0.44937288
Perche   -0.02783081 -0.1368411  0.58361717  0.53649480 -0.39873734  0.26166458
Javelot  -0.15319802  0.2405071 -0.32874217  0.69285498  0.36873120 -0.16320268
X1500m    0.03210733  0.3598049  0.65987362 -0.15669648  0.18557094 -0.29826888
                PC7         PC8         PC9        PC10
X100m     0.38177608  0.46160211 -0.10475771  0.42428269
Longueur  0.62769317 -0.02101165 -0.48266910  0.08104448
Poids    -0.30972542 -0.31393005 -0.42729075  0.39028424
Hauteur  -0.09145002  0.12509166  0.24366054 -0.10642724
X400m    -0.12442114  0.21339819 -0.55212939 -0.41399532
X110m.H   0.35733030 -0.71111429  0.15013429 -0.09086448
Disque    0.42988982  0.03838986  0.15480715 -0.44916580
Perche   -0.09796019  0.17803824  0.08297769 -0.27645138
Javelot   0.10674519  0.29614206  0.24732691  0.08777340
X1500m    0.08362898  0.01371744  0.30773397  0.42923132
```

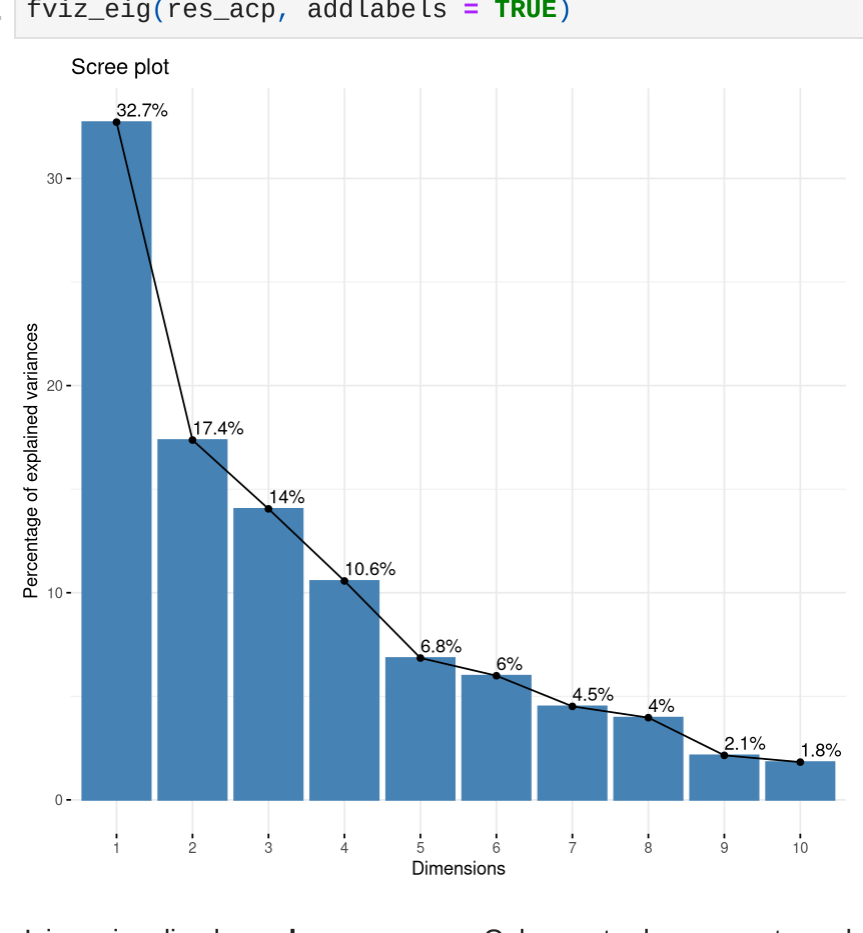
In [115…
```
get_eigenvalue(res_acp)
```

A data.frame: 10 × 3

	eigenvalue	variance.percent	cumulative.variance.percent
	<dbl>	<dbl>	<dbl>
Dim.1	3.2719055	32.719055	32.71906
Dim.2	1.7371310	17.371310	50.09037
Dim.3	1.4049167	14.049167	64.13953
Dim.4	1.0568504	10.568504	74.70804
Dim.5	0.6847735	6.847735	81.55577
Dim.6	0.5992687	5.992687	87.54846
Dim.7	0.4512353	4.512353	92.06081
Dim.8	0.3968766	3.968766	96.02958
Dim.9	0.2148149	2.148149	98.17773
Dim.10	0.1822275	1.822275	100.00000

On récupere ici les valeurs propres suite a notre ACP qui représentent la quantité de variance expliquée par chaque composante principale.

In [116…
```
fviz_eig(res_acp, addlabels = TRUE)
```

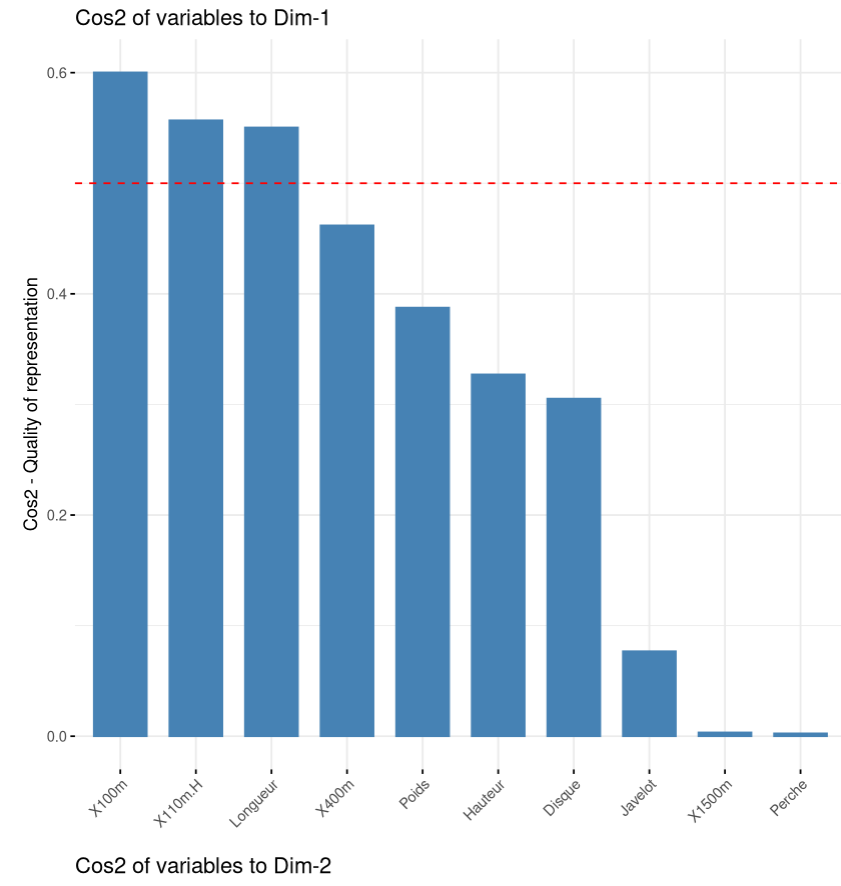


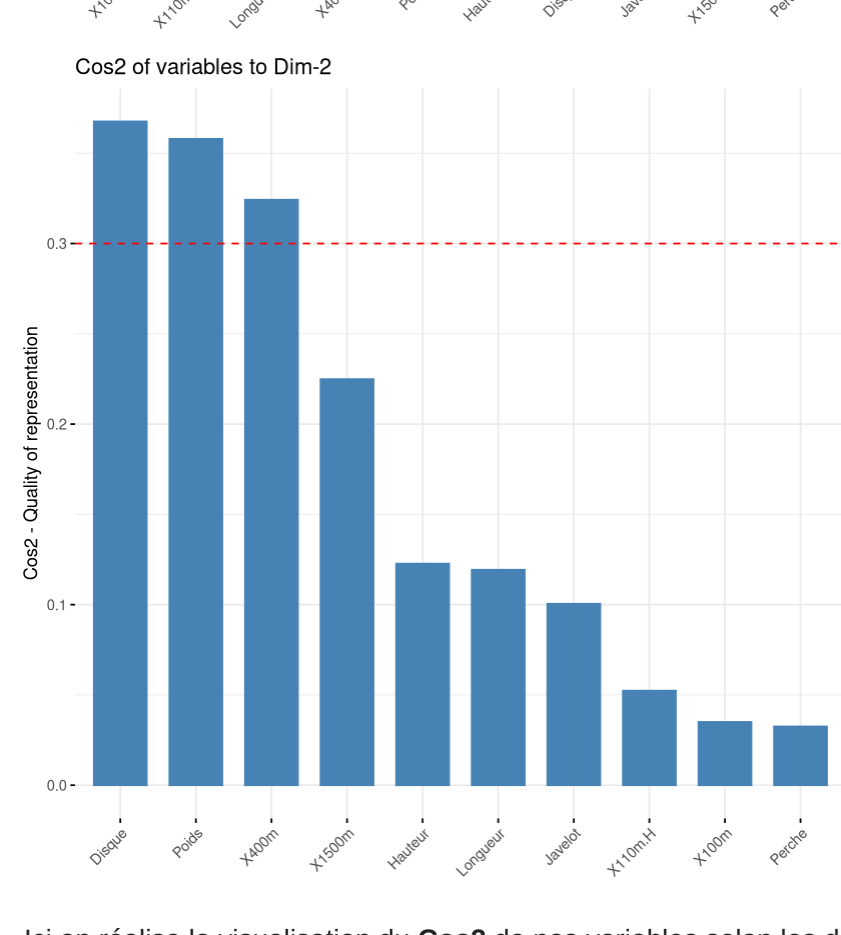
Ici on visualise les **valeurs propres**. Cela montre le pourcentage de variances expliquées par chaque axe principal.

Cette méthode nous permet de déterminer le nombre de **composantes principales** qui expliquent le maximum d'information. Le nombre d'axes est déterminé par le point, au-delà duquel les valeurs propres restantes sont toutes relativement petites et de tailles comparables.

Du graphique ci-dessus, nous pourrions vouloir nous arrêter à la cinquième composante principale. 81,5% des informations (variances) contenues dans les données sont conservées par les cinq premières composantes principales.

In [117…
```
fviz_cos2(res_acp, choice="var", axes=1) +
geom_hline(yintercept = 0.5, linetype = 2, color = "red")
fviz_cos2(res_acp, choice="var", axes=2)+
geom_hline(yintercept = 0.3, linetype = 2, color = "red")
```

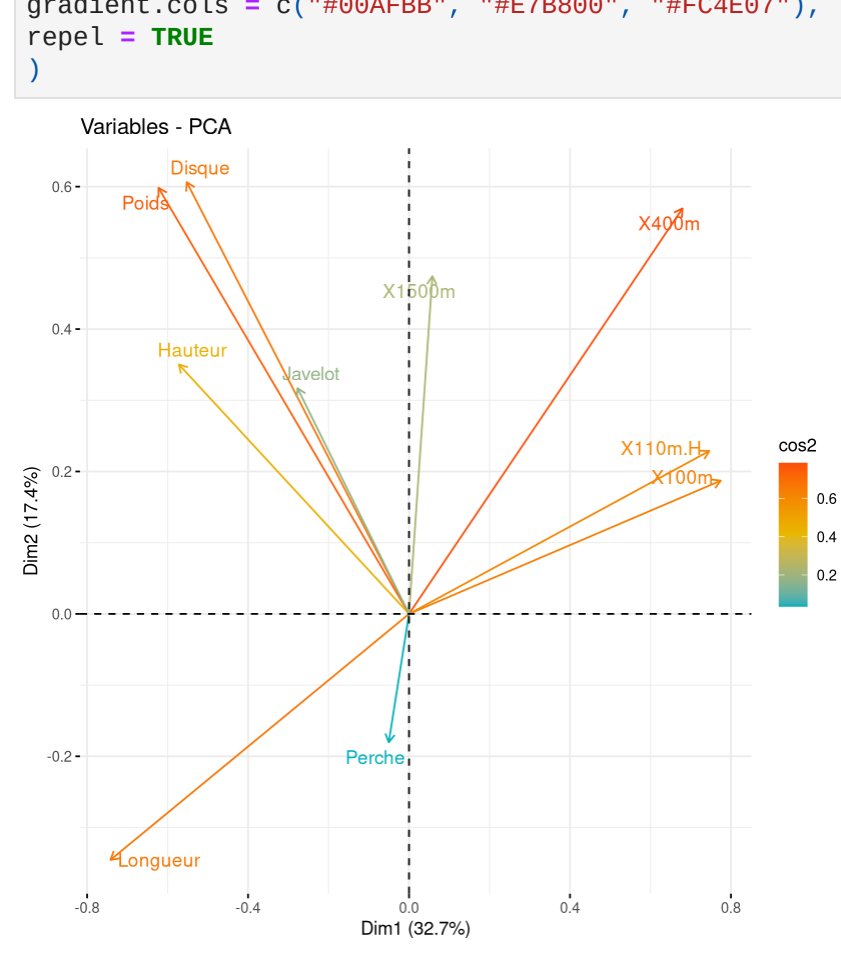




Ici on réalise la visualisation du **Cos2** de nos variables selon les dimensions 1 et 2.
Faire cette représentation nous permet de comparer la qualité de représentation de nos variables selon nos deux axes. Ici nous avons posé un seuil (le trait pointillé rouge) de 50% et 30% de qualité de représentation.

On peut donc voir que certaines variables sont bien représentées sur une dimension et beaucoup moins sur une autre, c'est par exemple le cas des épreuves du `X100m` ou `Longueur`.

In [118…
```
# Colorer en fonction du cos2: qualité de représentation
fviz_pca_var(res_acp, col.var = "cos2",
gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
repel = TRUE
)
```



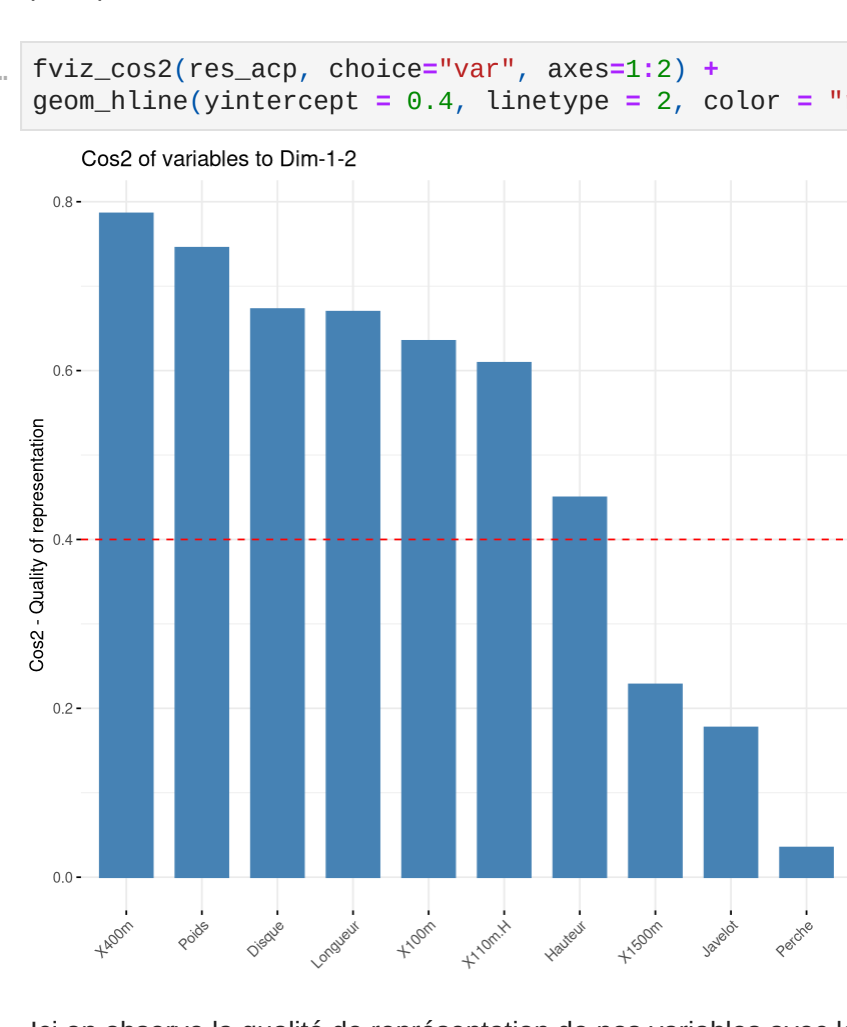
Cette représentation sur l'axe factoriel nous permet de bien remarquer la qualité de représentation de nos variables.
On doit ici interpréter les axes de facon indépendante.

On peut analyser plusieurs éléments :

- Les variables positivement corrélées sont regroupées.
- Les variables négativement corrélées sont positionnées sur les côtés opposés de l'origine du graphique (quadrants opposés).
- La distance entre les variables et l'origine mesure la qualité de représentation des variables. Les variables qui sont loin de l'origine sont bien représentées par l'ACP.

Par exemple, les épreuves de lancer, soit `Poids`, `Disques` et `Javelot`, sont fortement corrélées car pointent dans la meme direction. Cepandant, `Javelot` est moins bien représenté que `Poids` et `Disques` dans les deux premières composantes principales.

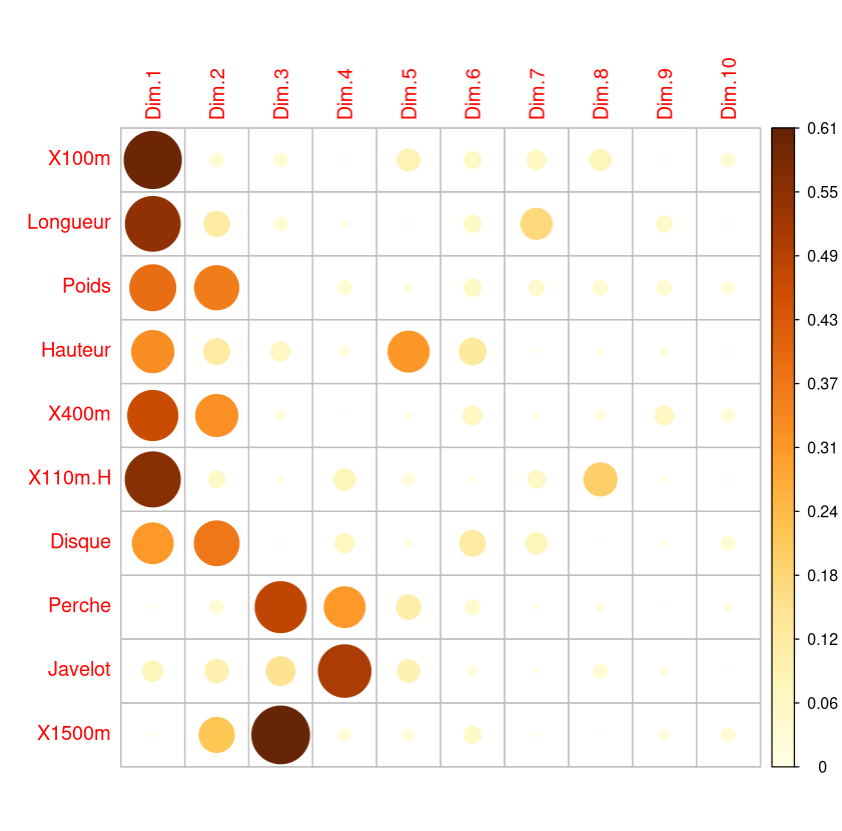
In [119…
```
fviz_cos2(res_acp, choice="var", axes=1:2) +
geom_hline(yintercept = 0.4, linetype = 2, color = "red")
```



Ici on observe la qualité de représentation de nos variables avec les deux dimensions en meme temps.

On remarque alors que `Javelot`, `X1500m` et `Perche` sont très mal représentés et ne nous permettront pas d'analyser correctement nos données.

In [120…
```
var <- get_pca_var(res_acp)
corrplot(var$cos2, is.corr=FALSE)
```

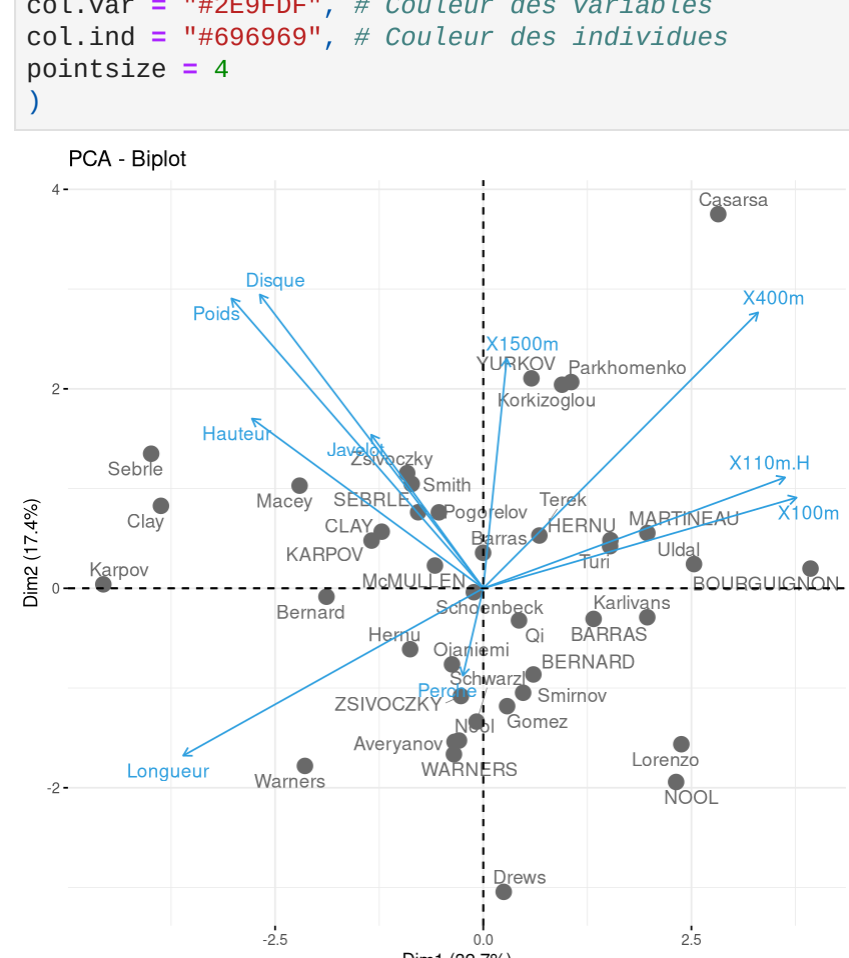


Le graphique ci-dessus indique la répartition de nos variables dans les différentes composantes principales.

On remarque par exemple que nos trois variables `Perche`, `Javelot` et `X1500m` qui sont mal expliquées dans nos deux premieres composantes, se retrouvent principalement dans les composantes 3 et 4.

Biplot

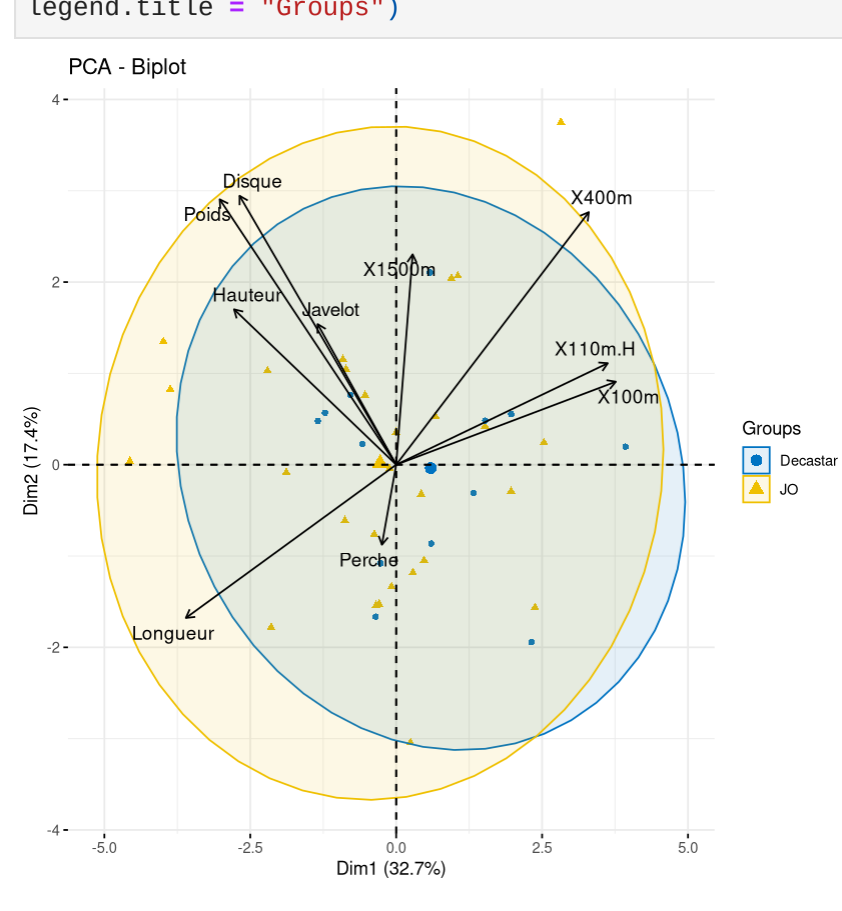
In [121…
```
fviz_pca_biplot(res_acp, repel = TRUE,
select.ind = list(name=rownames(df_scaled)),
col.var = "#2E9FDF", # Couleur des variables
col.ind = "#696969", # Couleur des individues
pointsize = 4
)
```



Grace au biplot, on peut voir la projection de nos individus dans la représentation sur l'axe factoriel de nos deux premieres composantes principales.

In [122…
```
df_scaled[1:28, 13] = "JO"
df_scaled[29:41, 13] = "Decastar"
df_scaled$Competition = as.character(df_scaled$Competition)

fviz_pca_biplot(res_acp,
col.ind = df_scaled$Competition, palette = "jco",
addEllipses = TRUE, label = "var",
col.var = "black", repel = TRUE,
legend.title = "Groups")
```



Ce nouveau biplot nous permet d'observer la répartition des individus en fonction de leur compétition.

On observe ici assez bien les différences entre les JO et le Decastar sur les scores :

- le Decastar comprend des candidats qui semblent plus performant sur les épreuves de courses (`X110m.H`, `X100m`)
- les Jeux Olympiques ont des candidats bon dans l'ensemble des épreuves, notamment sur le `Poids`, le `Disque` ou la `Longueur`.

Régression logistique

Quels sont les variables explicatives ?
Les variables explicatives seront les épreuves suivantes : `X400m`, `Poids`, `Disque`, `Longueur` et `X100m`.

Quelle est la variable que l'on cherche à expliquer ?
La variable que l'on cherche a expliquer ici est `Points`.

On applique donc la régression logistique (`Points` ~ `X400m`, `Poids`, `Disque`, `Longueur` et `X100m`).

In [123…
```
model = glm(df_scaled$Points ~ df_scaled$X400m + df_scaled$Poids + df_scaled$Disque + df_scaled$Longueur + df_s
```

In [124…
```
pander(summary(model))
```

```
|                   | Estimate | Std. Error | t value | Pr(>|t|)  |
|:----------------------:|:--------:|:----------:|:-------:|:---------:|
|    **(Intercept)**     |   8005   |   20.85    |   384   | 4.938e-65 |
|  **df_scaled$X400m**   |  -105.5  |   27.33    |  -3.86  | 0.0004676 |
|  **df_scaled$Poids**   |  140.7   |   28.13    |  5.002  | 1.597e-05 |
|  **df_scaled$Disque**  |  31.97   |     27     |  1.184  |  0.2444   |
| **df_scaled$Longueur** |  124.1   |   29.32    |  4.234  | 0.0001578 |
|  **df_scaled$X100m**   |  -47.85  |   28.73    | -1.665  |  0.1048   |


(Dispersion parameter for  gaussian  family taken to be  17816.38 )


|                    |                                  |
|:------------------:|:--------------------------------:|
|   Null deviance:   | 4689104  on 40  degrees of freedom |
| Residual deviance: | 623573  on 35  degrees of freedom  |
```

Ici la t value est la t-statistique, qui est le coefficient estimé (la colonne 2) divisé par l'erreur standard.
Pr(>|t|) est la **p-value** correspondante a la t-statistique, plus la p-value est petite, plus le coefficient estimé est important.

On remarque que la **p-value** est inférieur au seuil de 10% pour certaines des variables explicatives.

En effet,

- il existe une relation significative entre les `Points` et l'épreuve du `X400m` (0.0004676)
- il existe une relation significative entre les `Points` et l'épreuve du `Poids` (1.597e-05)
- il existe une relation significative entre les `Points` et l'épreuve du `Longueur` (0.0001578)
- on ne peut pas conclure qu'il existe une relation significative entre les `Points` et l'épreuve du `Disque` (0.2444) car le seuil de 10% est dépassé.
- on ne peut pas conclure qu'il existe une relation significative entre les `Points` et l'épreuve du `X100m` (0.1048) car le seuil de 10% est dépassé de peu.

Probabilité du `Classement` en fonction du nombre de `Points` :

In [125…
```
predict(model, type ="response") # prédiction du classement
```

1: 8737.67968841761 **2:** 8603.7256478761 **3:** 8885.66429923996 **4:** 8418.35089227936 **5:** 8407.84917908826 **6:** 8148.50617621855 **7:** 8097.30969370863 **8:** 8162.45912235028 **9:** 8251.31359589862 **10:** 7988.07621062211 **11:** 8035.30945284591 **12:** 8027.16924457413 **13:** 7971.1611974161 **14:** 8062.08874687084 **15:** 8060.36754978407 **16:** 8249.26148574078 **17:** 7877.31952477484 **18:** 7871.78395423618 **19:** 7896.84783468208 **20:** 7778.42918502854 **21:** 8026.28993512449 **22:** 8066.39556477783 **23:** 7476.52123728992 **24:** 7668.45029046145 **25:** 7650.28531149154 **26:** 7888.11434044162 **27:** 7561.19809007754 **28:** 7500.90488535754 **29:** 8160.34487119304 **30:** 8149.5905544036 **31:** 8224.80565774406 **32:** 7980.90418201601 **33:** 7942.97091112635 **34:** 8144.95751142744 **35:** 7930.7135924056 **36:** 7907.10555653465 **37:** 7711.02660010807 **38:** 7914.55393773567 **39:** 7756.55571810093 **40:** 7619.62646715238 **41:** 7408.01210337733

In [126…
```
residuals(model, type="deviance") # Résidus
```

1: 155.320311582387 **2:** 216.274352123903 **3:** -160.664299239963 **4:** -4.35089227935532 **5:** -64.8491790882599 **6:** 138.493823781454 **7:** 139.690306291369 **8:** 72.5408776497179 **9:** -26.3135958986222 **10:** 113.923789377889 **11:** 48.6905471540886 **12:** 49.8307554258654 **13:** 95.8388025839031 **14:** -39.0887468708424 **15:** -39.3675497840723 **16:** -243.261485740777 **17:** 115.680475225158 **18:** 62.2160457638211 **19:** 29.1521653179225 **20:** 139.570814971456 **21:** -133.28993512449 **22:** -201.395564777828 **23:** 231.478762710081 **24:** -76.4502904614483 **25:** -67.2853114915351 **26:** -315.114340441615 **27:** -66.1980900775407 **28:** -96.904885357540 5 **29:** 56.6551288069595 **30:** -27.5905544035968 **31:** -125.80565774406 **32:** 86.0958179839945 **33:** 93.029088873649 **34:** -114.957511427445 **35:** 73.2864075944017 **36:** 87.8944434653495 **37:** 90.9733998919328 **38:** -181.553937735674 **39:** -48.5557181009335 **40:** 31.3735328476232 **41:** -95.012103377333

Critique de l'analyse

Ce dataset nous a permis d'observer les différents comportements des classements de certains sportifs en fonction de leur performances sur les différentes épreuves des Jeux olympiques ou du Decastar.

Le dataset n'étant que de 41 lignes, soit 41 candidats, notre analyse ne permet pas d'en conclure que ces résultat s'appliquent à l'ensemble des Jeux olympiques ou Decastar.

Notre analyse fût simplifié par l'absence de valeur manquante ou aberrante, évitant de devoir éliminer d'autres informations ou de simuler les informations manquantes.

Les points que nous avons principalement pû observer sont:

- les sportifs bon dans un des sports de lancer (`Poids`, `Javelot`, `Disque`) ont plus de probabilité d'avoir de bons résultats dans l'ensemble de ces sports.
- les sportifs bon dans un des sports de courses courtes (`X100M`, `X110M.H`, `X400M`) ont plus de probabilité d'avoir de bons résultats dans l'ensemble de ces sports.
- l'épreuve `Perche` ne permet pas de conclure de corrélation avec les autres épreuves
- le `Classement` est fortement négativement corrélé avec les `Points`, indiquant que plus les points sur une épreuve sont élevés, plus le classement sera haut (1ère place, 2ème place...).

Le model de régression logistique nous permet de prédire un classement, mais celui-ci est très peu fiable dû au nombre de données utilisées pour son entrainement. On peut notamment observer que beaucoup de nos résidus de notre model ont des valeurs négatives, indiquant un taux d'erreur important sur cette prédiction.

Fin du projet.