**Calcul**

**Packages R**

Plusieurs fonctions, de différents packages, sont disponibles dans le logiciel *R* pour le calcul de l’ACP:

* prcomp() et princomp() [fonction de base, package stats],
* PCA() [package FactoMineR],
* dudi.pca() [package ade4],
* et epPCA() [package ExPosition]

Peu importe la fonction que vous décidez d’utiliser, vous pouvez facilement extraire et visualiser les résultats de l’ACP en utilisant les fonctions R fournies dans le package factoextra.

Ici, nous utiliserons les deux packages FactoMineR (pour l’analyse) et factoextra (pour la visualisation, des données, basée sur ggplot2).

Installez les deux packages comme suit:

install.packages(c("FactoMineR", "factoextra"))

Chargez-les dans R, en tapant ceci:

**library**("FactoMineR")

**library**("factoextra")

**Format des données**

Nous utiliserons les jeux de données de démonstration decathlon2 du package factoextra:

data(decathlon2)

# head(decathlon2)

Comme l’illustre la figure 3.1, les données utilisées ici décrivent la performance des athlètes lors de deux évènements sportifs (Decastar et OlympicG). Elles contiennent 27 individus (athlètes) décrits par 13 variables.

Selon la terminologie ACP, nos données contiennent des:

* *Individus actifs* (en bleu clair, lignes 1:23): individus qui sont utilisés lors de l’analyse en composantes principales.
* *Individus supplémentaires* (en bleu foncé, lignes 24:27): les coordonnées de ces individus seront prédites en utilisant l’information et les paramètres de l’ACP obtenue avec les individus/variables actifs.
* *Variables actives* (en rose, colonnes 1:10): variables utilisées pour l’ACP.
* *Variables supplémentaires*: comme les individus supplémentaires, les coordonnées de ces variables seront également prédites. On distingue des:
  + *Variables quantitatives supplémentaires* (rouge): les colonnes 11 et 12 correspondent respectivement au rang et aux points des athlètes.
  + *Variables qualitatives supplémentaires* (vert): Colonne 13 correspondant aux deux rencontres sportives (Jeux olympiques de 2004 ou Décastar 2004). Il s’agit d’une variable catégorielle. Elle peut être utilisée pour colorer les individus par groupes.

Nous commençons par extraire les individus actifs et les variables actives pour l’ACP:

decathlon2.active <- decathlon2[1:23, 1:10]

**head**(decathlon2.active[, 1:6], 4)

## X100m Long.jump Shot.put High.jump X400m X110m.hurdle

## SEBRLE 11.0 7.58 14.8 2.07 49.8 14.7

## CLAY 10.8 7.40 14.3 1.86 49.4 14.1

## BERNARD 11.0 7.23 14.2 1.92 48.9 15.0

## YURKOV 11.3 7.09 15.2 2.10 50.4 15.3

**Standardisation des données**

Dans l’analyse en composantes principales, les variables sont souvent normalisées. Ceci est particulièrement recommandé lorsque les variables sont mesurées dans différentes unités (par exemple: kilogrammes, kilomètres, centimètres, …); sinon, le résultat de l’ACP obtenue sera fortement affecté.

L’objectif est de rendre les variables comparables. Généralement, les variables sont normalisées de manière à ce qu’elles aient au final i) un écart type égal à un et ii) une moyenne égale à zéro.

Techniquement, l’approche consiste à transformer les données en soustrayant à chaque valeur une valeur de référence (la moyenne de la variable) et en la divisant par l’écart type. A l’issue de cette transformation les données obtenues sont dites *données centrées-réduites*. L’ACP appliquée à ces données transformées est appelée *ACP normée*.

La standardisation des données est une approche beaucoup utilisée dans le contexte de l’analyse des données d’expression de gènes avant les analyses de type PCA et de clustering.

Lors de la normalisation des variables, les données peuvent être transformées comme suit:

xi−mean(x)sd(x)xi−mean(x)sd(x)

Où mean(x)mean(x) est la moyenne des valeurs de x, et sd(x)sd(x) est l’écart type (SD).

La fonction scale() peut être utilisée pour normaliser les données.

Notez que, par défaut, la fonction **PCA**() [dans FactoMineR], normalise les données automatiquement pendant l’ACP; donc, vous n’avez pas besoin de faire cette transformation avant l’ACP.

### code R

Fonction R: PCA() [FactoMineR].

Format simplifié:

PCA(X, scale.unit = TRUE, ncp = 5, graph = TRUE)

* X: jeu de données de type data frame. Les lignes sont des individus et les colonnes sont des variables numériques
* scale.unit: une valeur logique. Si TRUE, les données sont standardisées/normalisées avant l’analyse.
* ncp: nombre de dimensions conservées dans les résultats finaux.
* graph: une valeur logique. Si TRUE un graphique est affiché.

Calculer l’ACP sur les individus/variables actifs:

**library**("FactoMineR")

res.pca <- PCA(decathlon2.active, graph = FALSE)

Le résultat de la fonction PCA() est une liste, contenant les éléments suivants:

print(res.pca)

## \*\*Results for the Principal Component Analysis (PCA)\*\*

## The analysis was performed on 23 individuals, described by 10 variables

## \*The results are available in the following objects:

##

## name description

## 1 "$eig" "eigenvalues"

## 2 "$var" "results for the variables"

## 3 "$var$coord" "coord. for the variables"

## 4 "$var$cor" "correlations variables - dimensions"

## 5 "$var$cos2" "cos2 for the variables"

## 6 "$var$contrib" "contributions of the variables"

## 7 "$ind" "results for the individuals"

## 8 "$ind$coord" "coord. for the individuals"

## 9 "$ind$cos2" "cos2 for the individuals"

## 10 "$ind$contrib" "contributions of the individuals"

## 11 "$call" "summary statistics"

## 12 "$call$centre" "mean of the variables"

## 13 "$call$ecart.type" "standard error of the variables"

## 14 "$call$row.w" "weights for the individuals"

## 15 "$call$col.w" "weights for the variables"

L’objet créé avec la fonction PCA() contient de nombreuses informations trouvées dans de nombreuses listes et matrices différentes. Ces valeurs sont décrites dans la section suivante.

## Visualisation et interprétation

Les fonctions suivantes, de factoextra, seront utilisées:

* get\_eigenvalue(res.pca): Extraction des valeurs propres / variances des composantes principales
* fviz\_eig(res.pca): Visualisation des valeurs propres
* get\_pca\_ind(res.pca), get\_pca\_var(res.pca): Extraction des résultats pour les individus et les variables, respectivement.
* fviz\_pca\_ind(res.pca), fviz\_pca\_var(res.pca): visualisez les résultats des individus et des variables, respectivement.
* fviz\_pca\_biplot(res.pca): Création d’un biplot des individus et des variables.

Dans les sections suivantes, nous allons illustrer chacune de ces fonctions.

### Valeurs propres / Variances

Comme décrit dans les sections précédentes, les valeurs propres (eigenvalues en anglais) mesurent la quantité de variance expliquée par chaque axe principal. Les valeurs propres sont grandes pour les premiers axes et petits pour les axes suivants. Autrement dit, les premiers axes correspondent aux directions portant la quantité maximale de variation contenue dans le jeu de données.

Nous examinons les valeurs propres pour déterminer le nombre de composantes principales à prendre en considération. Les valeurs propres et la proportion de variances (i.e. information) retenues par les composantes principales peuvent être extraites à l’aide de la fonction get\_eigenvalue() [package factoextra].

**library**("factoextra")

eig.val <- get\_eigenvalue(res.pca)

eig.val

## eigenvalue variance.percent cumulative.variance.percent

## Dim.1 4.124 41.24 41.2

## Dim.2 1.839 18.39 59.6

## Dim.3 1.239 12.39 72.0

## Dim.4 0.819 8.19 80.2

## Dim.5 0.702 7.02 87.2

## Dim.6 0.423 4.23 91.5

## Dim.7 0.303 3.03 94.5

## Dim.8 0.274 2.74 97.2

## Dim.9 0.155 1.55 98.8

## Dim.10 0.122 1.22 100.0

La somme de toutes les valeurs propres donne une variance totale de 10.

La proportion de variance expliquée par chaque valeur propre est donnée dans la deuxième colonne. Par exemple, 4.124 divisé par 10 est égal à 0.4124, ou, environ 41.24% de la variation est expliquée par cette première valeur propre. Le pourcentage cumulé expliqué est obtenu en ajoutant les proportions successives de variances expliquées. Par exemple, 41.242% plus 18.385% sont égaux à 59.627%, et ainsi de suite. Par conséquent, environ 59.627% de la variance totale est expliquée par les deux premières valeurs propres.

Les valeurs propres peuvent être utilisées pour déterminer le nombre d’axes principaux à conserver après l’ACP (Kaiser 1961):

* Une valeur propre > 1 indique que la composante principale (PC) concernée représente plus de variance par rapport à une seule variable d’origine, lorsque les données sont standardisées. Ceci est généralement utilisé comme seuil à partir duquel les PC sont conservés. A noter que cela ne s’applique que lorsque les données sont normalisées.
* Vous pouvez également limiter le nombre d’axes à un nombre qui représente une certaine fraction de la variance totale. Par exemple, si vous êtes satisfaits avec 70% de la variance totale expliquée, utilisez le nombre d’axes pour y parvenir.

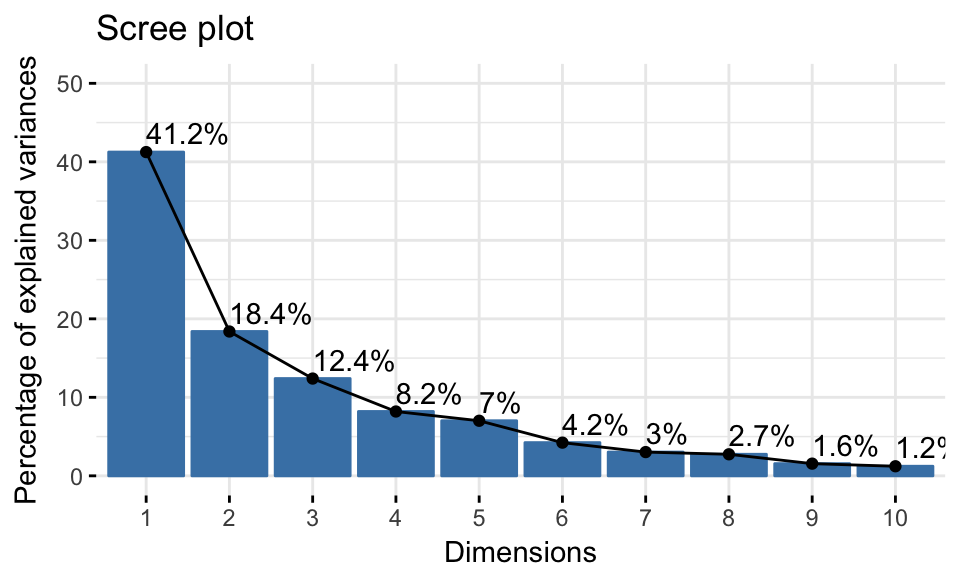
Malheureusement, il n’existe pas de méthode objective bien acceptée pour décider du nombre d’axes principaux qui suffisent. Cela dépendra du domaine d’application spécifique et du jeu de données spécifiques. Dans la pratique, on a tendance à regarder les premiers axes principaux afin de trouver des profils intéressants dans les données.

Dans notre analyse, les trois premières composantes principales expliquent 72% de la variation. C’est un pourcentage acceptable.

Une autre méthode pour déterminer le nombre de composantes principales est de regarder le graphique des valeurs propres (appelé scree plot). Le nombre d’axes est déterminé par le point, au-delà duquel les valeurs propres restantes sont toutes relativement petites et de tailles comparables (Jollife 2002, Peres-Neto, Jackson, and Somers (2005)).

Le graphique des valeurs propres peut être généré à l’aide de la fonction fviz\_eig () ou fviz\_screeplot () [package factoextra].

fviz\_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))



Du graphique ci-dessus, nous pourrions vouloir nous arrêter à la cinquième composante principale. 87% des informations (variances) contenues dans les données sont conservées par les cinq premières composantes principales.

### Graphique des variables

#### Résultats

Une méthode simple pour extraire les résultats, pour les variables, à partir de l’ACP est d’utiliser la fonction get\_pca\_var() [package factoextra]. Cette fonction retourne une liste d’éléments contenant tous les résultats pour les variables actives (coordonnées, corrélation entre variables et les axes, cosinus-carré et contributions)

var <- get\_pca\_var(res.pca)

var

## 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"

Les composants de get\_pca\_var() peuvent être utilisés dans le graphique des variables comme suit:

* var$coord: coordonnées des variables pour créer un nuage de points.
* var$cos2: cosinus carré des variables. Représente la qualité de représentation des variables sur le graphique de l’ACP. Il est calculé comme étant les coordonnées au carré: var.cos2 = var.coord \* var.coord.
* var$contrib: contient les contributions (en pourcentage), des variables, aux composantes principales. La contribution d’une variable (var) à une composante principale donnée: (var.cos2 \* 100) / (total cos2 du composant).

Notez qu’il est possible de visualiser les variables et de les colorer en fonction de i) leurs qualités de représentation (cos2) ou ii) leurs contributions aux composantes principales (contrib).

Les différents éléments peuvent être consultés comme suit:

# Coordonnées

**head**(var$coord)

# Cos2: qualité de répresentation

**head**(var$cos2)

# Contributions aux composantes principales

**head**(var$contrib)

Dans cette section, nous décrivons comment visualiser les variables et tirer des conclusions concernant leurs corrélations. Ensuite, nous mettons en évidence les variables selon i) leurs qualités de représentation sur le graphique ou ii) leurs contributions aux composantes principales.

#### Cercle de corrélation

La corrélation entre une variable et une composante principale (PC) est utilisée comme coordonnées de la variable sur la composante principale. La représentation des variables diffère de celle des observations: les observations sont représentées par leurs projections, mais les variables sont représentées par leurs corrélations (Abdi and Williams 2010).

# Coordonnées des variables

**head**(var$coord, 4)

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## X100m -0.851 -0.1794 0.302 0.0336 -0.194

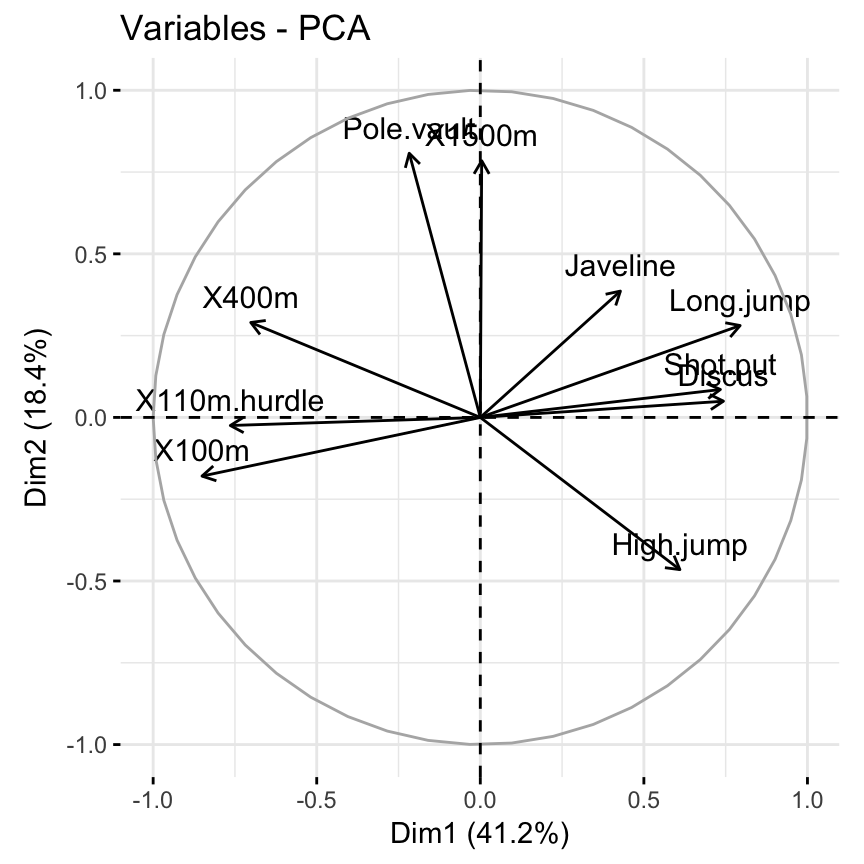
## Long.jump 0.794 0.2809 -0.191 -0.1154 0.233

## Shot.put 0.734 0.0854 0.518 0.1285 -0.249

## High.jump 0.610 -0.4652 0.330 0.1446 0.403

Pour visualiser les variables, tapez ceci:

fviz\_pca\_var(res.pca, col.var = "black")



Le graphique ci-dessus est également connu sous le nom de graphique de corrélation des variables. Il montre les relations entre toutes les variables. Il peut être interprété comme suit:

* 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.

#### Qualité de représentation

La qualité de représentation des variables sur la carte de l’ACP s’appelle cos2 (cosinus carré) . Vous pouvez accéder au cos2 comme suit:

**head**(var$cos2, 4)

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## X100m 0.724 0.03218 0.0909 0.00113 0.0378

## Long.jump 0.631 0.07888 0.0363 0.01331 0.0544

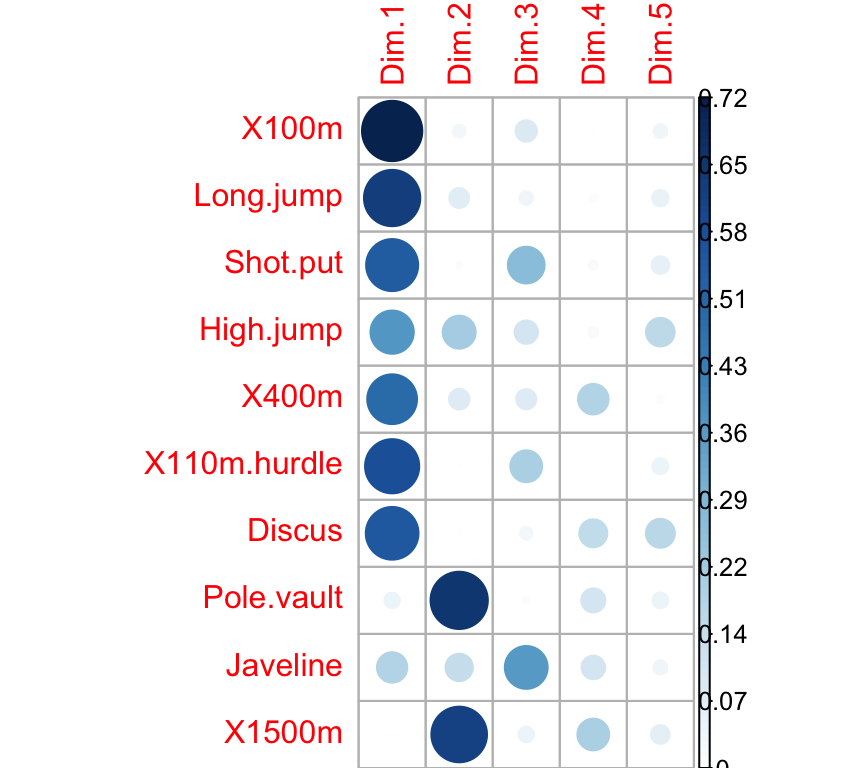
## Shot.put 0.539 0.00729 0.2679 0.01650 0.0619

## High.jump 0.372 0.21642 0.1090 0.02089 0.1622

Vous pouvez visualiser le cos2 des variables sur toutes les dimensions en utilisant le package corrplot:

**library**("corrplot")

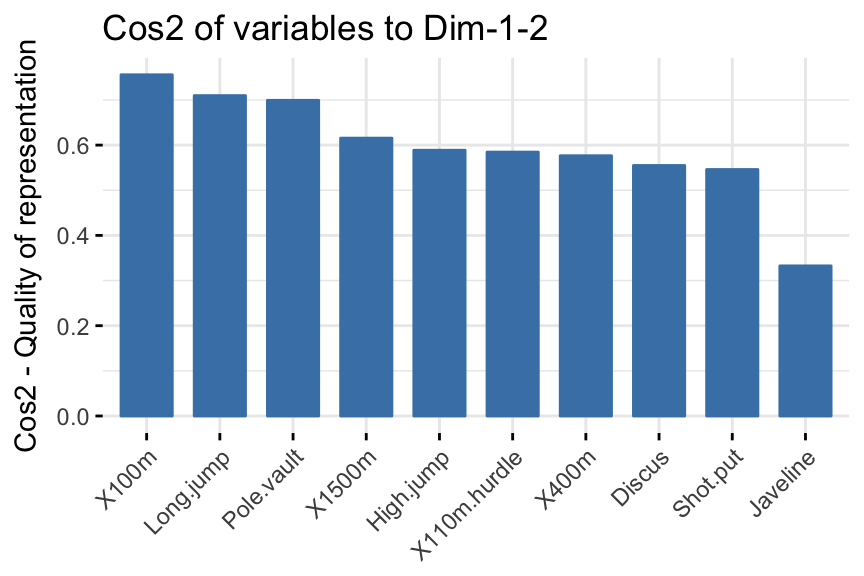
corrplot(var$cos2, is.corr=FALSE)



Il est également possible de créer un bar plot du cosinus carré des variables en utilisant la fonction fviz\_cos2() [dans factoextra]:

# Cos2 total des variables sur Dim.1 et Dim.2

fviz\_cos2(res.pca, choice = "var", axes = 1:2)



Notez que,

* Un cos2 élevé indique une bonne représentation de la variable sur les axes principaux en considération. Dans ce cas, la variable est positionnée à proximité de la circonférence du cercle de corrélation.
* Un faible cos2 indique que la variable n’est pas parfaitement représentée par les axes principaux. Dans ce cas, la variable est proche du centre du cercle.

Pour une variable donnée, la somme des cos2 sur toutes les composantes principales est égale à 1.

Si une variable est parfaitement représentée par seulement deux composantes principales (Dim.1 & Dim.2), la somme des cos2 sur ces deux axes est égale à 1. Dans ce cas, les variables seront positionnées sur le cercle de corrélation.

Pour certaines des variables, plus de 2 axes peuvent être nécessaires pour représenter parfaitement les données. Dans ce cas, les variables sont positionnées à l’intérieur du cercle de corrélation.

En résumé:

* Les valeurs de cos2 sont utilisées pour estimer la qualité de la représentation
* Plus une variable est proche du cercle de corrélation, meilleure est sa représentation sur la carte de l’ACP (et elle est plus importante pour interpréter les composantes principales en considération)
* Les variables qui sont proche du centre du graphique sont moins importantes pour les premières composantes.

Il est possible de colorer les variables en fonction de la valeur de leurs cos2 à l’aide de l’argument col.var = "cos2". Cela produit un gradient de couleurs. Dans ce cas, l’argument gradient.cols peut être utilisé pour spécifier une palette de couleur personnalisée. Par exemple, gradient.cols = c("white", "blue", "red") signifie que:

* les variables à faible valeur de cos2 seront colorées en “white” (blanc)
* les variables avec les valeurs moyennes de cos2 seront colorées en “blue” (bleu)
* les variables avec des valeurs élevées de cos2 seront colorées en “red” (rouge)

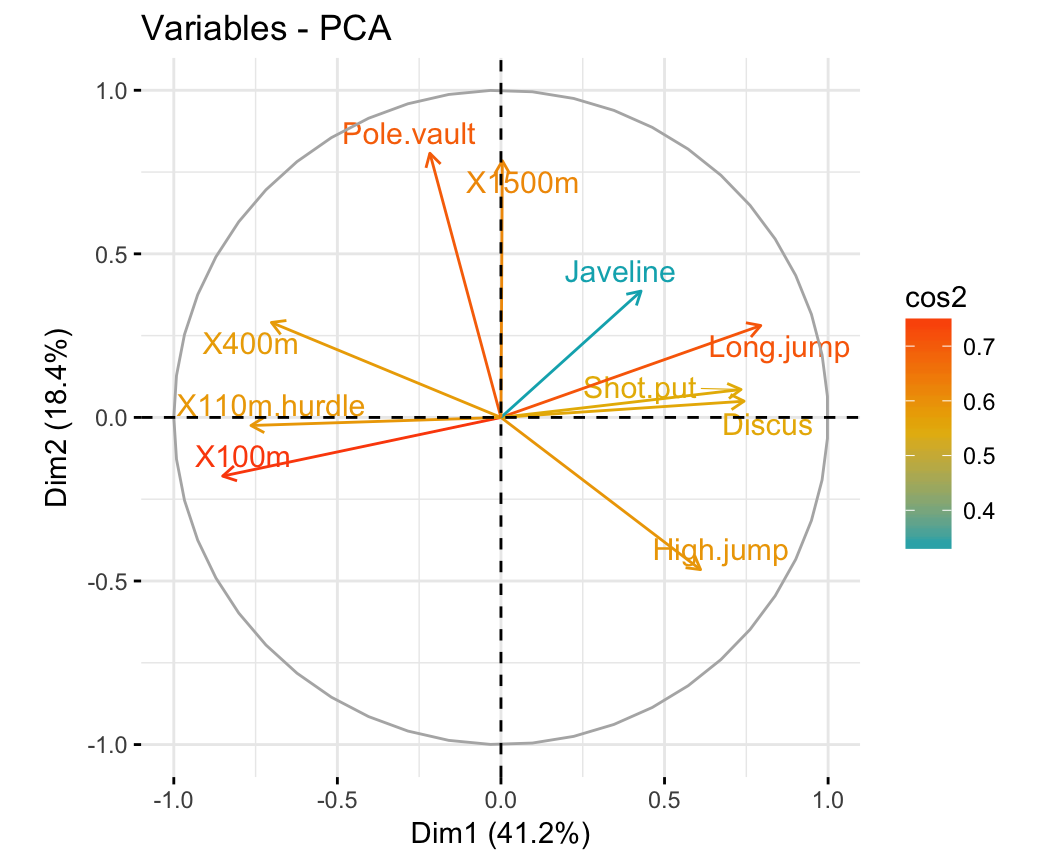
# Colorer en fonction du cos2: qualité de représentation

fviz\_pca\_var(res.pca, col.var = "cos2",

gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),

repel = TRUE # Évite le chevauchement de texte

)



Notez qu’il est également possible de modifier la transparence des variables en fonction de leurs valeurs de cos2 à l’aide de l’option alpha.var = "cos2". Par exemple, tapez ceci:

# Changer la transparence en fonction du cos2

fviz\_pca\_var(res.pca, alpha.var = "cos2")

#### Contributions des variables aux axes principaux

Les contributions des variables dans la définition d’un axe principal donné, sont exprimées en pourcentage.

* Les variables corrélées avec PC1 (i.e., Dim.1) et PC2 (i.e., Dim.2) sont les plus importantes pour expliquer la variabilité dans le jeu de données.
* Les variables qui ne sont pas en corrélation avec un axe ou qui sont corrélées avec les derniers axes sont des variables à faible apport et peuvent être supprimées pour simplifier l’analyse globale.

La contribution des variables peut être extraite comme suit :

**head**(var$contrib, 4)

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## X100m 17.54 1.751 7.34 0.138 5.39

## Long.jump 15.29 4.290 2.93 1.625 7.75

## Shot.put 13.06 0.397 21.62 2.014 8.82

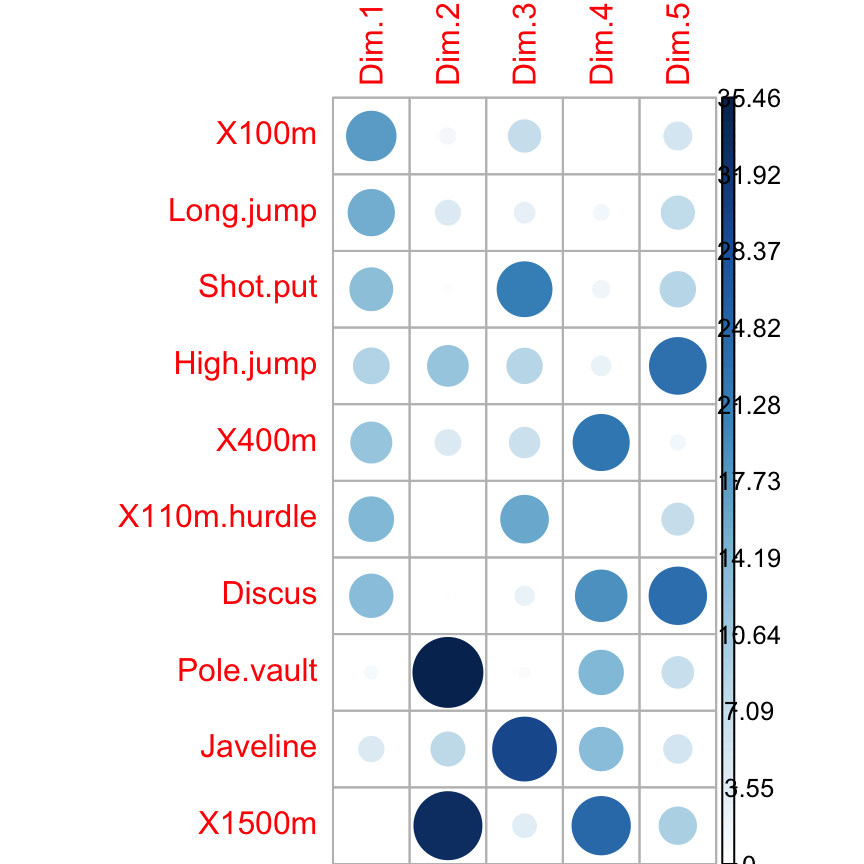
## High.jump 9.02 11.772 8.79 2.550 23.12

Plus la valeur de la contribution est importante, plus la variable contribue à la composante principale en question.

Il est possible d’utiliser la fonction corrplot() [package corrplot] pour mettre en évidence les variables les plus contributives pour chaque dimension:

**library**("corrplot")

corrplot(var$contrib, is.corr=FALSE)



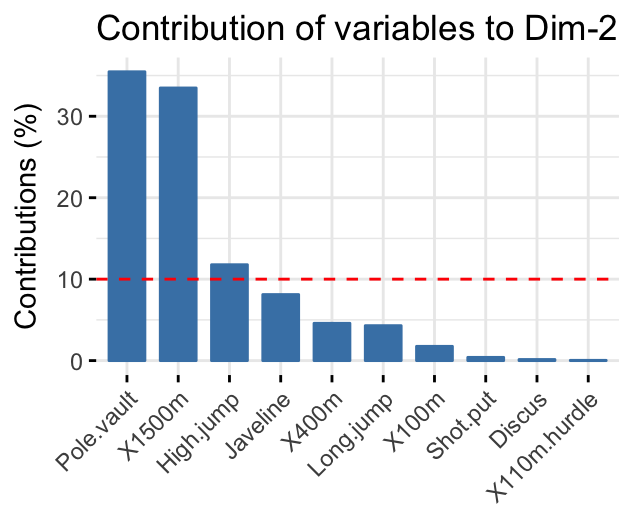
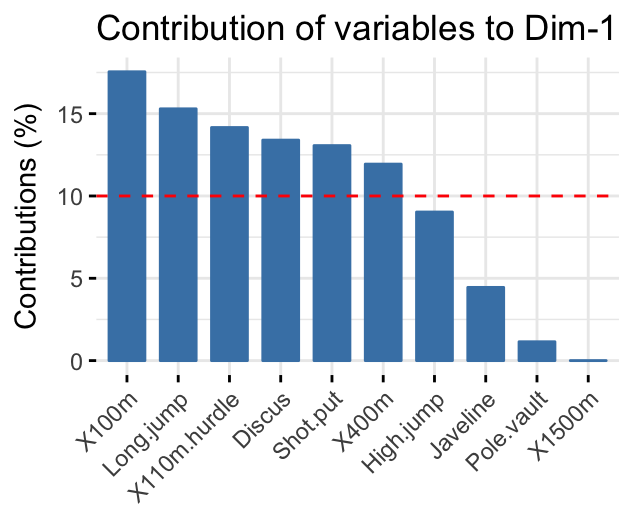
La fonction fviz\_contrib() [package factoextra] peut être utilisée pour créer un bar plot de la contribution des variables. Si vos données contiennent de nombreuses variables, vous pouvez décider de ne montrer que les principales variables contributives. Le code R ci-dessous montre le top 10 des variables contribuant le plus aux composantes principales:

# Contributions des variables à PC1

fviz\_contrib(res.pca, choice = "var", axes = 1, top = 10)

# Contributions des variables à PC2

fviz\_contrib(res.pca, choice = "var", axes = 2, top = 10)



La contribution totale à PC1 et PC2 est obtenue avec le code R suivant:

fviz\_contrib(res.pca, choice = "var", axes = 1:2, top = 10)

La ligne en pointillé rouge, sur le graphique ci-dessus, indique la contribution moyenne attendue. Si la contribution des variables était uniforme, la valeur attendue serait 1/length(variables) = 1/10 = 10%. Pour une composante donnée, une variable avec une contribution supérieure à ce seuil pourrait être considérée comme importante pour contribuer à la composante.

Notez que la contribution totale d’une variable donnée, pour expliquer la variance retenue par deux composantes principales, disons PC1 et PC2, est calculée comme contrib = [(C1 \* Eig1) + (C2 \* Eig2)]/(Eig1 + Eig2) , où

* C1 et C2 sont les contributions de la variable aux axes PC1 et PC2, respectivement
* Eig1 et Eig2 sont les valeurs propres de PC1 et PC2, respectivement. Rappelons que les valeurs propres mesurent la quantité de variation retenue par chaque PC.

Dans ce cas, la contribution moyenne attendue (seuil) est calculée comme suit:

Comme mentionné ci-dessus, si les contributions des 10 variables étaient uniformes, la contribution moyenne attendue pour une PC donnée serait 1/10 = 10%. La contribution moyenne attendue d’une variable pour PC1 et PC2 est: [(10 \* Eig1) + (10 \* Eig2)]/ (Eig1 + Eig2)

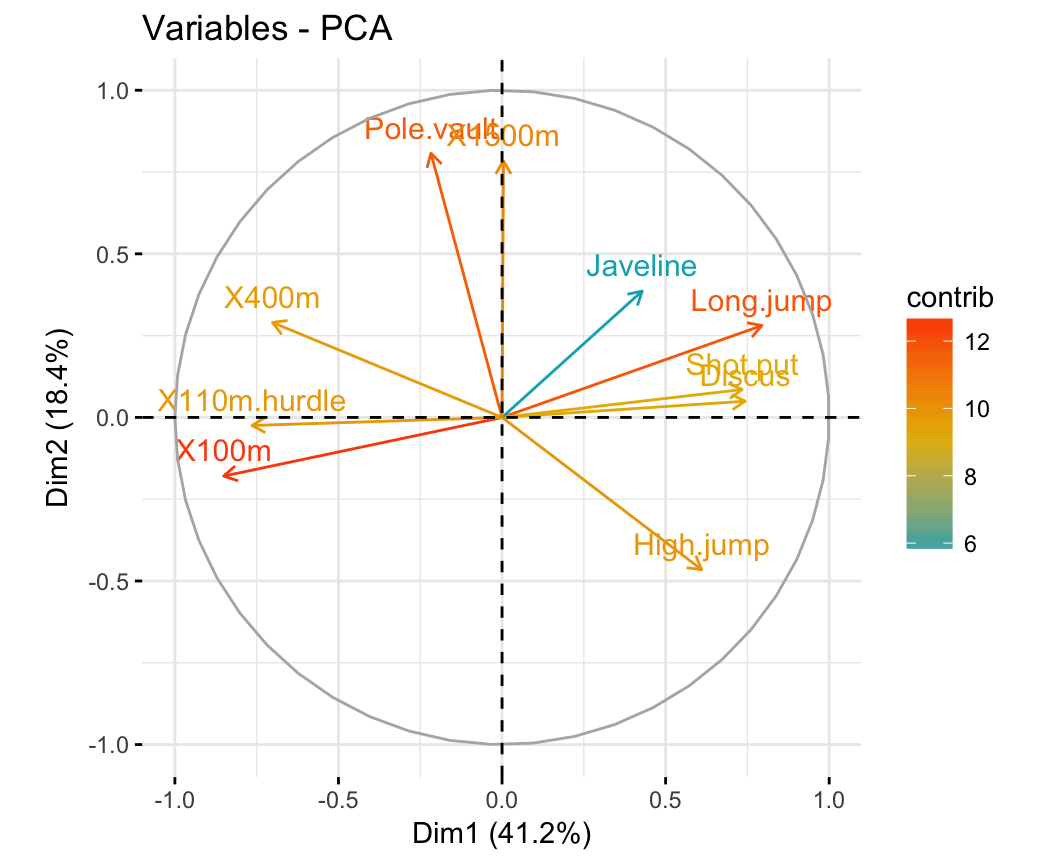
On peut voir que les variables - X100m, Long.jump et Pole.vault - contribuent le plus aux dimensions 1 et 2.

Les variables les plus importantes (ou, contributives) peuvent être mises en évidence sur le graphe de corrélation comme suit:

fviz\_pca\_var(res.pca, col.var = "contrib",

gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07")

)



Notez qu’il est aussi possible de modifier la transparence des variables en fonction de leurs contributions en utilisant l’option alpha.var = "contrib". Par exemple, tapez ceci:

# Changez la transparence en fonction de contrib

fviz\_pca\_var(res.pca, alpha.var = "contrib")

#### Colorer en fonction d’une variable continue quelconque

Dans les sections précédentes, nous avons montré comment colorer les variables par leurs contributions et leurs cos2. Notez qu’il est possible de colorer les variables par n’importe quelle variable continue personnalisée. La variable de coloration doit avoir la même longueur que le nombre de variables actives dans l’ACP (ici n = 10).

Par exemple, tapez ceci:

# Créer une variable aléatoire continue de longueur 10

set.seed (123)

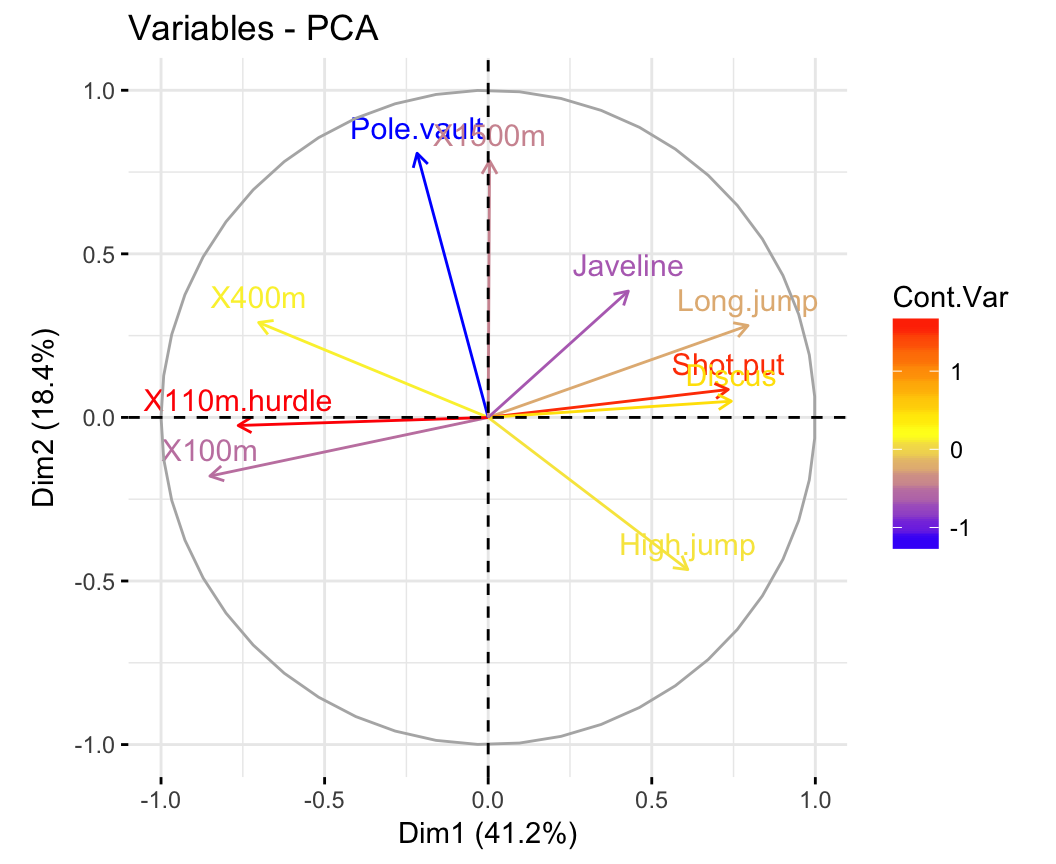
my.cont.var <- rnorm (10)

# Colorer les variables en fonction de la variable continue

fviz\_pca\_var(res.pca, col.var = my.cont.var,

gradient.cols = c("blue", "yellow", "red"),

legend.title = "Cont.Var")



#### Colorer par groupes

Il est également possible de modifier la couleur des variables par groupes définis par une variable qualitative / catégorielle, également appelée factor dans la terminologie R.

Comme nous n’avons pas de variable de regroupement/classification dans notre jeu de données pour classer les variables, nous la créerons.

Dans l’exemple suivant, nous commençons par classer les variables en 3 groupes en utilisant l’algorithme de classification k-means. Ensuite, nous utilisons les clusters retournés par l’algorithme k-means pour colorer les variables.

Notez que, si vous êtes intéressés par apprendre le clustering, nous avons précédemment publié un livre intitulé “Practical Guide To Cluster Analysis in R” (<https://goo.gl/DmJ5y5>).

# Créez une variable de regroupement en utilisant kmeans

# Créez 3 groupes de variables (centers = 3)

set.seed(123)

res.km <- kmeans(var$coord, centers = 3, nstart = 25)

grp <- as.factor(res.km$cluster)

# Colorer les variables par groupes

fviz\_pca\_var(res.pca, col.var = grp,

palette = c("#0073C2FF", "#EFC000FF", "#868686FF"),

legend.title = "Cluster")

Notez que, pour changer la couleur des groupes, l’argument “palette” peut être utilisé. Pour modifier le gradient de couleurs, l’argument “gradient.cols” peut être utilisé.

### Description des dimensions

Dans les sections précédentes, nous avons décrit comment mettre en évidence les variables en fonction de leurs contributions aux composantes principales.

Notez également que la fonction dimdesc() [dans FactoMineR], pour dimension description (en anglais), peut être utilisée pour identifier les variables les plus significativement associées avec une composante principale donnée . Elle peut être utilisée comme suit:

res.desc <- dimdesc(res.pca, axes = c(1,2), proba = 0.05)

# Description de la dimension 1

res.desc$Dim.1

## $quanti

## correlation p.value

## Long.jump 0.794 6.06e-06

## Discus 0.743 4.84e-05

## Shot.put 0.734 6.72e-05

## High.jump 0.610 1.99e-03

## Javeline 0.428 4.15e-02

## X400m -0.702 1.91e-04

## X110m.hurdle -0.764 2.20e-05

## X100m -0.851 2.73e-07

# Description de la dimension 2

res.desc$Dim.2

## $quanti

## correlation p.value

## Pole.vault 0.807 3.21e-06

## X1500m 0.784 9.38e-06

## High.jump -0.465 2.53e-02

Dans le résultat ci-dessus, $quanti représente les résultats pour les variables quantitatives. Notez que les variables sont triées en fonction de la p-value de la corrélation.

### Graphique des individus

#### Résultats

Les résultats, pour les individus, peuvent être extraits à l’aide de la fonction get\_pca\_ind() [package factoextra]. Comme get\_pca\_var(), la fonction get\_pca\_ind() retourne une liste de matrices contenant tous les résultats pour les individus (coordonnées, corrélation entre individus et axes, cosinus-carré et contributions)

ind <- get\_pca\_ind(res.pca)

ind

## Principal Component Analysis Results for individuals

## ===================================================

## Name Description

## 1 "$coord" "Coordinates for the individuals"

## 2 "$cos2" "Cos2 for the individuals"

## 3 "$contrib" "contributions of the individuals"

Pour accéder aux différents éléments, utilisez ceci:

# Coordonnées des individus

**head**(ind$coord)

# Qualité des individus

**head**(ind$cos2)

# Contributions des individus

**head**(ind$contrib)

#### Graphique: qualité et contribution

La fonction fviz\_pca\_ind() est utilisée pour produire le graphique des individus. Pour créer un graphique simple, tapez ceci:

fviz\_pca\_ind (res.pca)

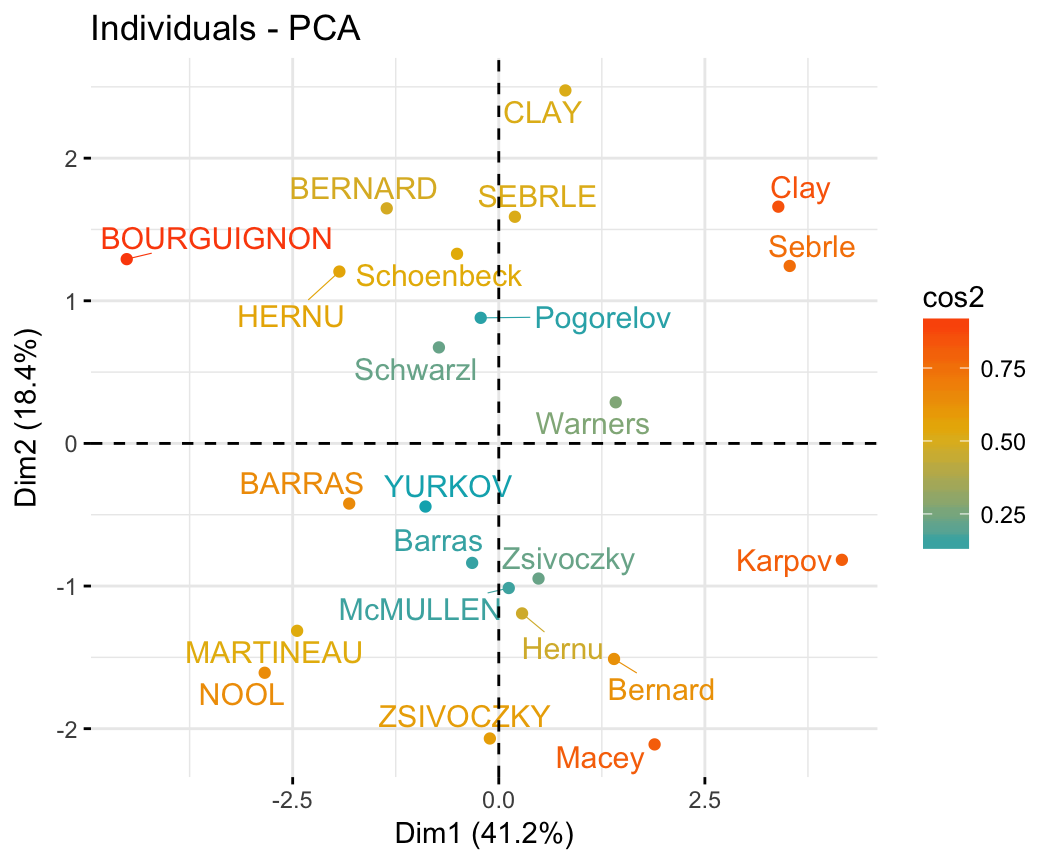
Comme les variables, il est également possible de colorer les individus en fonction de leurs valeurs de cos2:

fviz\_pca\_ind (res.pca, col.ind = "cos2",

gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),

repel = TRUE # Évite le chevauchement de texte

)



Notez que les individus qui sont similaires sont regroupés sur le graphique.

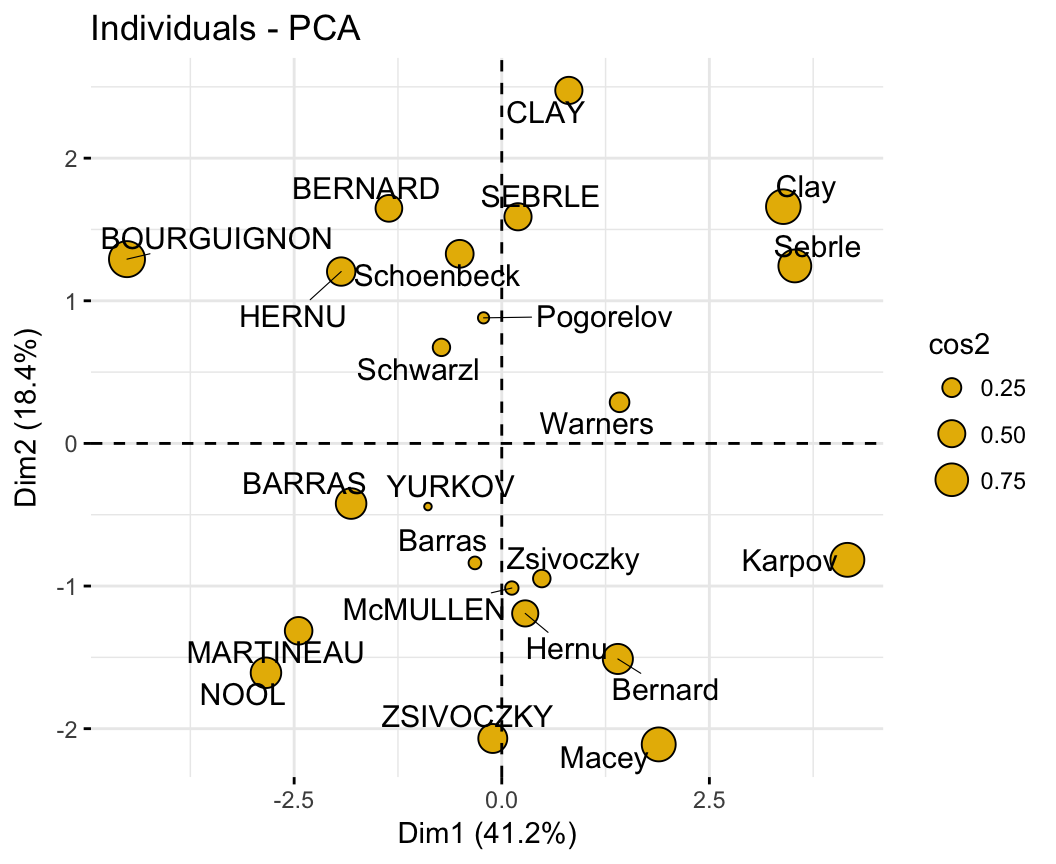
Vous pouvez également modifier la taille des points en fonction du cos2 des individus correspondants:

fviz\_pca\_ind (res.pca, pointsize = "cos2",

pointshape = 21, fill = "#E7B800",

repel = TRUE # Évite le chevauchement de texte

)



Pour modifier la taille et la couleur des points en fonction du cos2, essayez ceci:

fviz\_pca\_ind(res.pca, col.ind = "cos2", pointsize = "cos2",

gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),

repel = TRUE

)

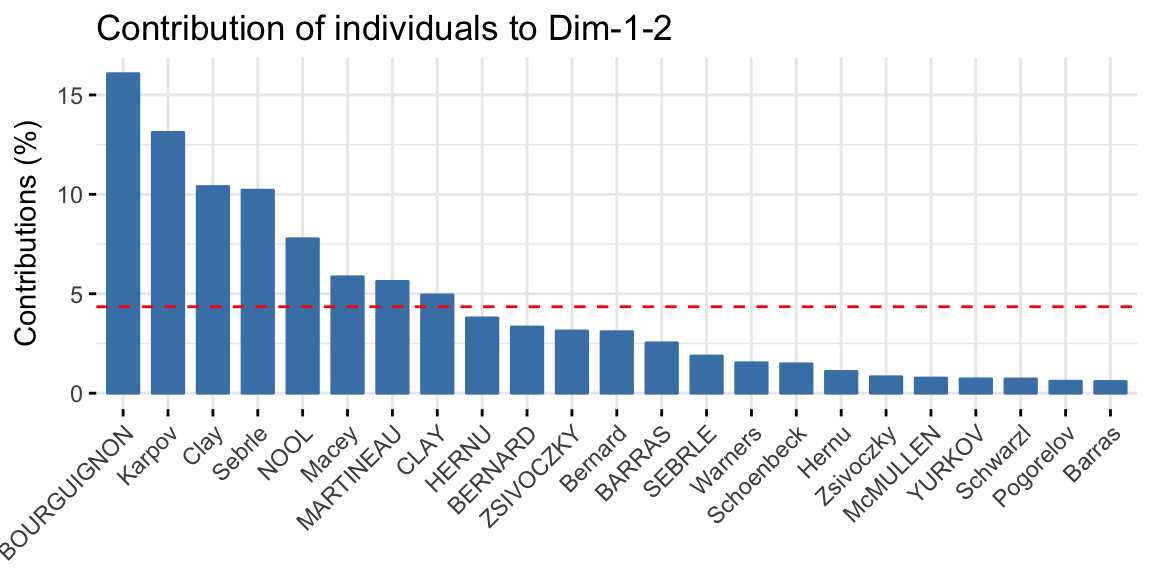
Pour créer un bar plot de la qualité de représentation (cos2) des individus, vous pouvez utiliser la fonction fviz\_cos2()comme décrit précédemment pour les variables:

fviz\_cos2(res.pca, choice = "ind")

Pour visualiser la contribution des individus aux deux premières composantes principales, tapez ceci:

# Contribution totale sur PC1 et PC2

fviz\_contrib(res.pca, choice = "ind", axes = 1:2)



#### Colorer en fonction d’une variable continue quelconque

Comme pour les variables, les individus peuvent être colorés par n’importe quelle variable continue personnalisée en spécifiant l’argument col.ind.

Par exemple, tapez ceci:

# Créez une variable continue aléatoire de longueur 23,

# Même longeur que le nombre d'individus actifs dans l'ACP

set.seed (123)

my.cont.var <- rnorm(23)

# Colorer les individus par la variable continue

fviz\_pca\_ind(res.pca, col.ind = my.cont.var,

gradient.cols = c("blue", "yellow", "red"),

legend.title = "Cont.Var")

#### Colorer par groupes

Ici, nous décrivons comment colorer les individus par groupes. En plus, nous montrons comment ajouter des ellipses de concentration et des ellipses de confiance par groupes. Pour cela, nous utiliserons le jeu de données iris.

Le jeu de données iris ressemblent à ceci:

**head**(iris, 3)

## Sepal.Length Sepal.Width Petal.Length Petal.Width Species

## 1 5.1 3.5 1.4 0.2 setosa

## 2 4.9 3.0 1.4 0.2 setosa

## 3 4.7 3.2 1.3 0.2 setosa

La colonne “Species” (espèces) sera utilisée comme variable de regroupement. Nous commençons par calculer l’analyse en composantes principales comme suit:

# La variable Species (index = 5) est supprimée

# avant l'ACP

iris.pca <- PCA(iris [, - 5], graph = FALSE)

Dans le code R ci-dessous: l’argument habillage ou col.ind peut être utilisé pour spécifier la variable à utiliser pour colorer les individus par groupes.

Pour ajouter une ellipse de concentration autour de chaque groupe, spécifiez l’argument addEllipses = TRUE. L’argument palette peut être utilisé pour changer les couleurs du groupe.

fviz\_pca\_ind(iris.pca,

geom.ind = "point", # Montre les points seulement (mais pas le "text")

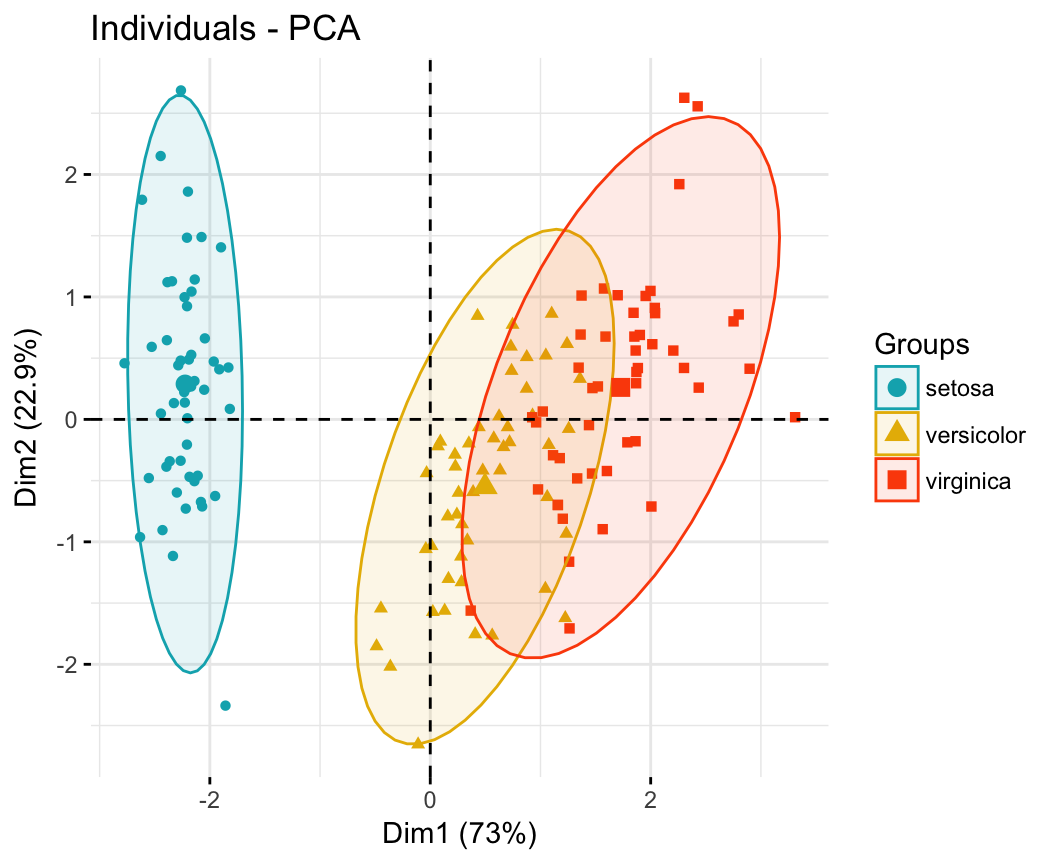
col.ind = iris$Species, # colorer by groups

palette = c("#00AFBB", "#E7B800", "#FC4E07"),

addEllipses = TRUE, # Ellipses de concentration

legend.title = "Groups"

)



Pour supprimer le point moyen des groupes (centre de gravité), spécifiez l’argument mean.point = FALSE.

Si vous voulez des ellipses de confiance au lieu des ellipses de concentration, utilisez ellipse.type = “confidence”.

# Ajoutez des ellipses de confiance

fviz\_pca\_ind(iris.pca, geom.ind = "point", col.ind = iris$Species,

palette = c("#00AFBB", "#E7B800", "#FC4E07"),

addEllipses = TRUE, ellipse.type = "confidence",

legend.title = "Groups"

)

Notez que les valeurs autorisées pour l’argument palette incluent:

* “grey” pour les palettes de couleurs grises;
* les palettes brewer, par exemple “RdBu”, “Blues”, …; Pour afficher tout, tapez ceci dans R: RColorBrewer::display.brewer.all().
* palette de couleurs personnalisées, par ex. c(“blue”, “red”);
* et des palettes de journaux scientifiques du package ggsci, par exemple: “npg”, “aaas”, “lancet”, “jco”, “ucscgb”, “uchicago”, “simpsons” et “rickandmorty”.

Par exemple, pour utiliser la palette de couleurs jco (journal of clinical oncology), tapez ceci:

fviz\_pca\_ind(iris.pca,

label = "none", # Caché le texte des individus

habillage = iris$Species, # colorer par groupes

addEllipses = TRUE, # Ellipses de concentration

palette = "jco"

)

### Personnalisation du graphique

Notez que fviz\_pca\_ind() et fviz\_pca\_var() héritent de la fonction principale fviz() [in factoextra]. fviz() englobe la fonction ggscatter() [dans ggpubr]. Par conséquent, d’autres arguments, à passer à la fonction fviz() et ggscatter(), peuvent être spécifiés dans fviz\_pca\_ind() et fviz\_pca\_var().

Ici, nous présentons certains de ces arguments supplémentaires pour personnaliser les graphiques de l’ACP.

#### Dimensions

Par défaut, les variables/individus sont représentés en fonction des dimensions 1 et 2. Si vous souhaitez les visualiser sur les dimensions 2 et 3, par exemple, vous devez spécifier l’argument axes = c(2, 3).

# Variables sur les dimensions 2 et 3

fviz\_pca\_var(res.pca, axes = c(2, 3))

# Individus sur les dimensions 2 et 3

fviz\_pca\_ind(res.pca, axes = c(2, 3))

#### Eléments graphiques: point, texte, flèche

L’argument geom (pour géométrie) et les dérivées sont utilisés pour spécifier les éléments géométriques ou les éléments graphiques à utiliser.

1. geom.var: un texte spécifiant la géométrie à utiliser pour tracer les variables. Les valeurs autorisées sont la combinaison de c(“point”, “arrow”, “text”).

* Utilisez geom.var = "point", pour afficher uniquement les points;
* Utilisez geom.var = "text" pour afficher uniquement le texte d’annotation des points;
* Utilisez geom.var = c("point", "text") pour afficher à la fois les points et le texte d’annotation;
* Utilisez geom.var = c("arrow", "text") pour afficher les flèches et les annotations (par défaut).

Par exemple, tapez ceci:

# Afficher les points et l'annotation des variables

fviz\_pca\_var(res.pca, geom.var = c("point", "text"))

1. geom.ind: un texte spécifiant la géométrie à utiliser pour les individus. Les valeurs autorisées sont la combinaison de c(“point”, “text”).

* Utilisez geom.ind = "point", pour afficher uniquement les points;
* Utilisez geom.ind = "text" pour afficher uniquement le texte d’annotation des individus;
* Utilisez geom.ind = c("point", "texte") pour afficher à la fois les points et le texte d’annotation (valeur par défaut)

Par exemple, tapez ceci:

# Afficher uniquement l'annotation des individus

fviz\_pca\_ind(res.pca, geom.ind = "text")

#### Taille et forme des éléments graphiques

1. labelsize: taille du texte, par exemple: labelsize = 4.
2. pointsize: taille des points, par exemple: pointsize = 1.5.
3. arrowsize: taille des flèches. Contrôle l’épaisseur des flèches, par exemple: arrowsize = 0.5.
4. pointshape: forme des points, pointshape = 21. Tapez ggpubr::show\_point\_shapes() pour voir les formes de points disponibles.

# Changez la taille des flèches et du texte

fviz\_pca\_var(res.pca, arrowsize = 1, labelsize = 5,

repel = TRUE)

# Modification de la taille, de la forme

# et de la couleur de remplissage des points

# Modifier la taille du texte

fviz\_pca\_ind (res.pca,

pointsize = 3, pointshape = 21, fill = "lightblue",

labelsize = 5, repel = TRUE)

#### Ellipses

Comme nous l’avons décrit dans la section précédente @ref(color-ind-by-groups), lors de la coloration des individus par groupes, vous pouvez ajouter des ellipses de concentration des points à l’aide de l’argument addEllipses = TRUE.

Notez que l’argument ellipse.type peut être utilisé pour modifier le type d’ellipses. Les valeurs possibles sont:

* "convex": trace une coque convexe autour de chaque groupe de points.
* "confidence": trace les ellipses de confiance autour des points moyens des groupes, comme la fonction coord.ellipse() [dans FactoMineR].
* "t": suppose une distribution t multivariée.
* "norm": suppose une distribution normale multivariée.
* "euclid": dessine un cercle avec le rayon égal au niveau, représentant la distance euclidienne du centre. Cette ellipse ne sera probablement pas circulaire sauf si coord\_fixed() est appliqué.

L’argument ellipse.level est également disponible pour modifier la taille de l’ellipse de concentration en probabilité normale. Par exemple, spécifiez ellipse.level = 0.95 ou ellipse.level = 0.66.

# Add confidence ellipses

fviz\_pca\_ind(iris.pca, geom.ind = "point",

col.ind = iris$Species, # color by groups

palette = c("#00AFBB", "#E7B800", "#FC4E07"),

addEllipses = TRUE, ellipse.type = "confidence",

legend.title = "Groups"

)

# Convex hull

fviz\_pca\_ind(iris.pca, geom.ind = "point",

col.ind = iris$Species, # color by groups

palette = c("#00AFBB", "#E7B800", "#FC4E07"),

addEllipses = TRUE, ellipse.type = "convex",

legend.title = "Groups"

)

#### Centre de gravité: Le point moyen des groupes

Lors de la coloration des individus par groupes (section @ref(color-ind-by-groups)), les points moyens des groupes (barycentre ou centre de gravité) sont également affichés par défaut.

Pour supprimer les points moyens, utilisez l’argument mean.point = FALSE.

fviz\_pca\_ind (iris.pca,

geom.ind = "point", # afficher les points seulement (pas de "texte")

col.ind = iris$Species, # Couleur par groupes

legend.title = "Groupes",

mean.point = FALSE)

#### Axes

L’argument axes.linetype peut être utilisé pour spécifier le type de trait des axes. La valeur par défaut est “dashed” (pointillé). Les valeurs autorisées incluent “blank”, “solid”, “dotted”, etc. Pour voir toutes les valeurs possibles, tapez ggpubr::show\_line\_types() dans R.

Pour supprimer le trait des axes, utilisez axes.linetype = “blank”:

fviz\_pca\_var (res.pca, axes.linetype = "blank")

#### Paramètres graphiques

Pour changer facilement les paramètres graphiques de n’importe quels ggplots, vous pouvez utiliser la fonction [ggpar()](http://www.sthda.com/english/rpkgs/ggpubr/reference/ggpar.html) [package ggpubr]

Les paramètres graphiques qui peuvent être modifiés à l’aide de ggpar() incluent:

* Le tritre principal, le titre des axes et des légendes.
* Position de la légende. Valeurs possibles: “top”, “bottom”, “left”, “right”, “none”.
* Palette de couleurs.
* Thèmes. Les valeurs autorisées incluent: theme\_gray(), theme\_bw(), theme\_minimal(), theme\_classic(), theme\_void().

ind.p <- fviz\_pca\_ind(iris.pca, geom = "point", col.ind = iris$Species)

ggpubr::ggpar(ind.p,

title = "Principal Component Analysis",

subtitle = "Iris data set",

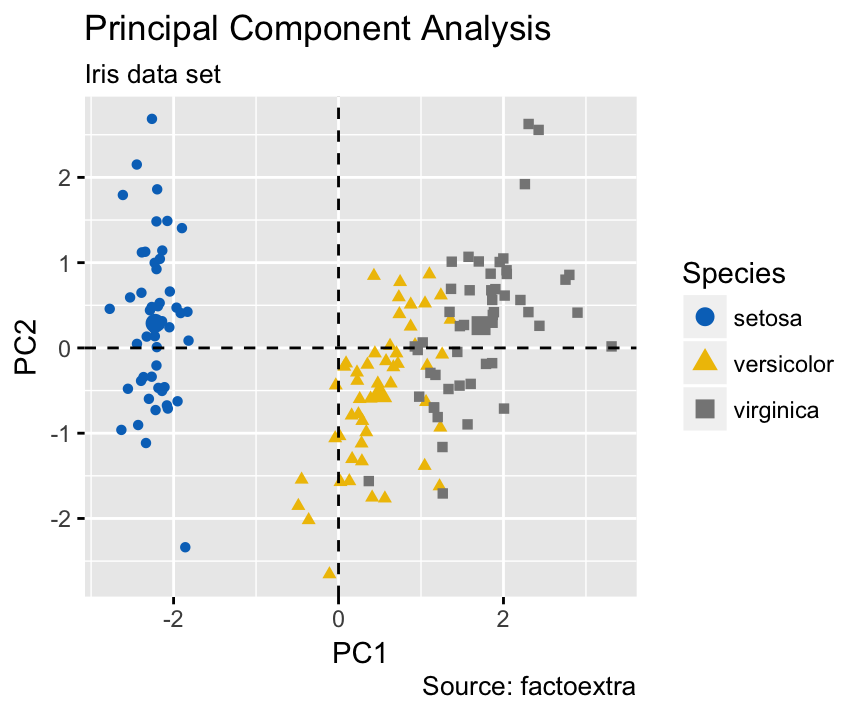
caption = "Source: factoextra",

xlab = "PC1", ylab = "PC2",

legend.title = "Species", legend.position = "top",

ggtheme = theme\_gray(), palette = "jco"

)



# Biplot

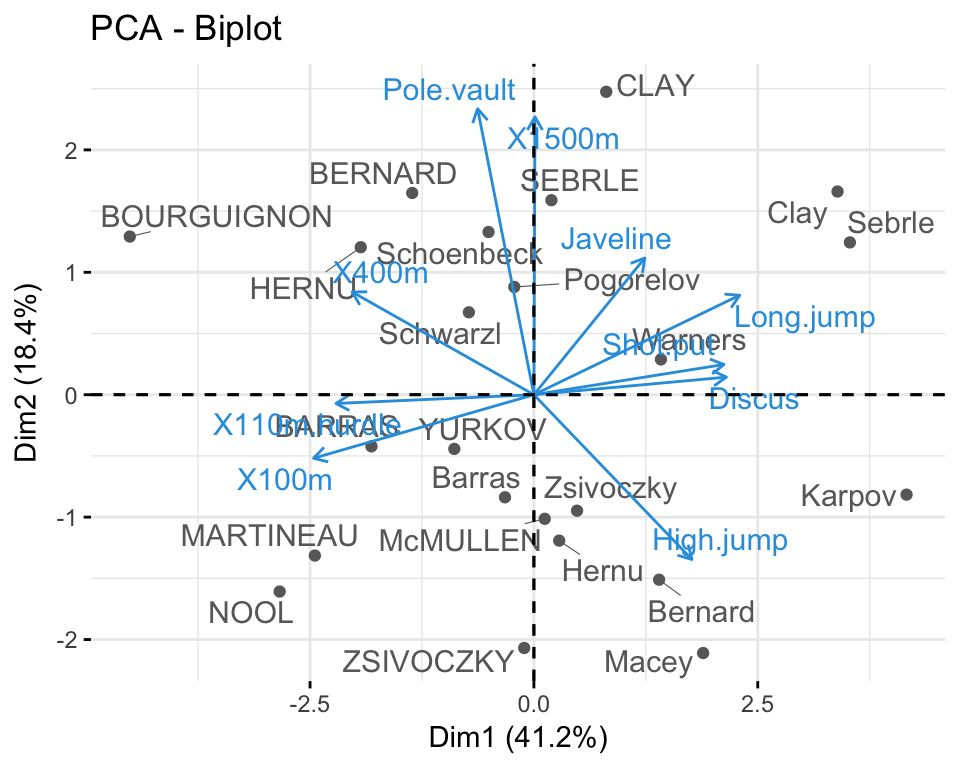
Pour créer un biplot simple des individus et des variables, tapez ceci:

fviz\_pca\_biplot(res.pca, repel = TRUE,

col.var = "#2E9FDF", # Couleur des variables

col.ind = "#696969" # Couleur des individues

)



Notez que le biplot n’est utile que s’il existe un faible nombre de variables et d’individus dans le jeu de données; sinon le graphique final serait illisible.

Notez également que les coordonnées des individus et des variables ne sont pas construites dans le même espace. Par conséquent, dans le biplot, vous devriez vous concentrer principalement sur la direction des variables mais pas sur leurs positions absolues sur le graphique.

Globalement, un biplot peut être interprété comme suit:

* un individu qui se trouve du même côté d’une variable donnée a une valeur élevée pour cette variable;
* un individu qui se trouve sur le côté opposé d’une variable donnée a une faible valeur pour cette variable.

Maintenant, en utilisant le résultat iris.pca, nous allons:

* faire un biplot des individus et des variables
* changer la couleur des individus par groupes: col.ind = iris$Species
* afficher uniquement l’annotation des variables: label = "var" ou utilisez geom.ind = "point"

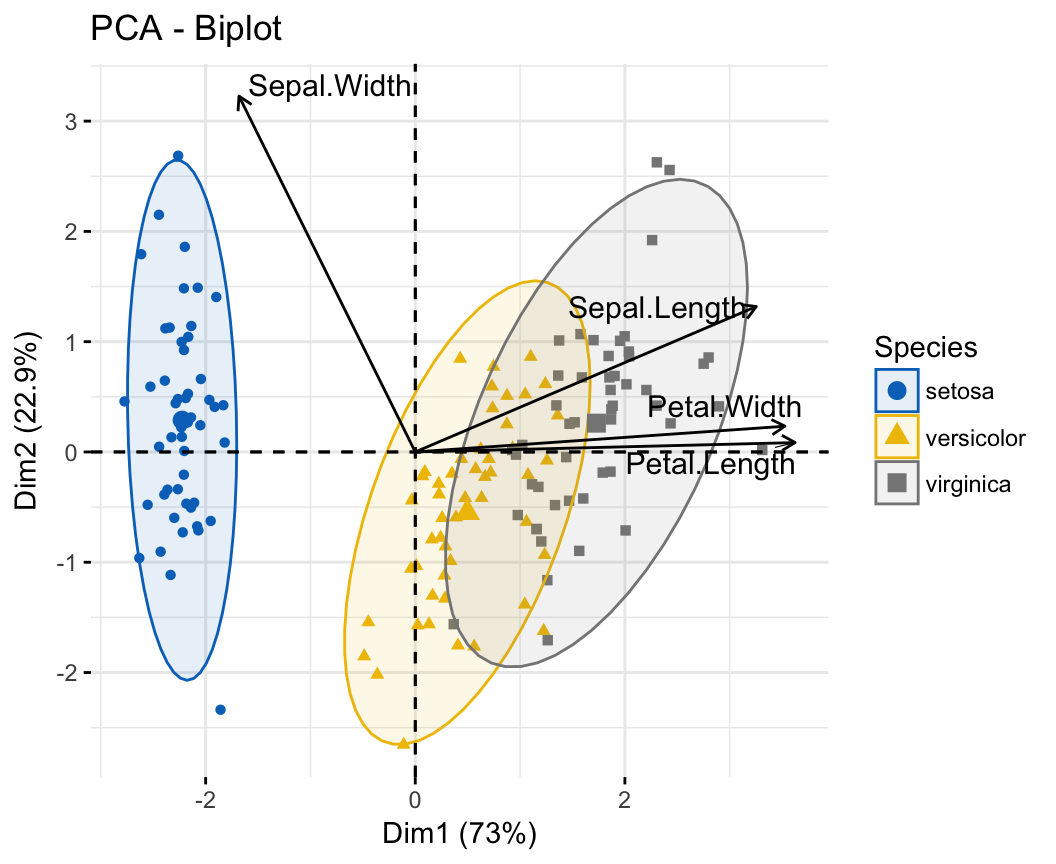
fviz\_pca\_biplot (iris.pca,

col.ind = iris$Species, palette = "jco",

addEllipses = TRUE, label = "var",

col.var = "black", repel = TRUE,

legend.title = "Species")



Dans l’exemple suivant, nous voulons colorer les individus et les variables par groupes. L’astuce consiste à utiliser pointshape = 21 pour les points des individus. Cette forme de point particulière peut être remplie par une couleur en utilisant l’argument fill.ind. La couleur de la bordure des points des individus est définie en “black” en utilisant col.ind. Pour colorer les variables par groupes, l’argument col.var sera utilisé.

Pour personnaliser les couleurs des individus et des variables, nous utilisons les fonctions helper fill\_palette() et color\_palette() [package ggpubr].

fviz\_pca\_biplot(iris.pca,

# Colueur de remplissage des individdus par groupes

geom.ind = "point",

pointshape = 21,

pointsize = 2.5,

fill.ind = iris$Species,

col.ind = "black",

# Colorer les variables par groupes

col.var = factor(c("sepal", "sepal", "petal", "petal")),

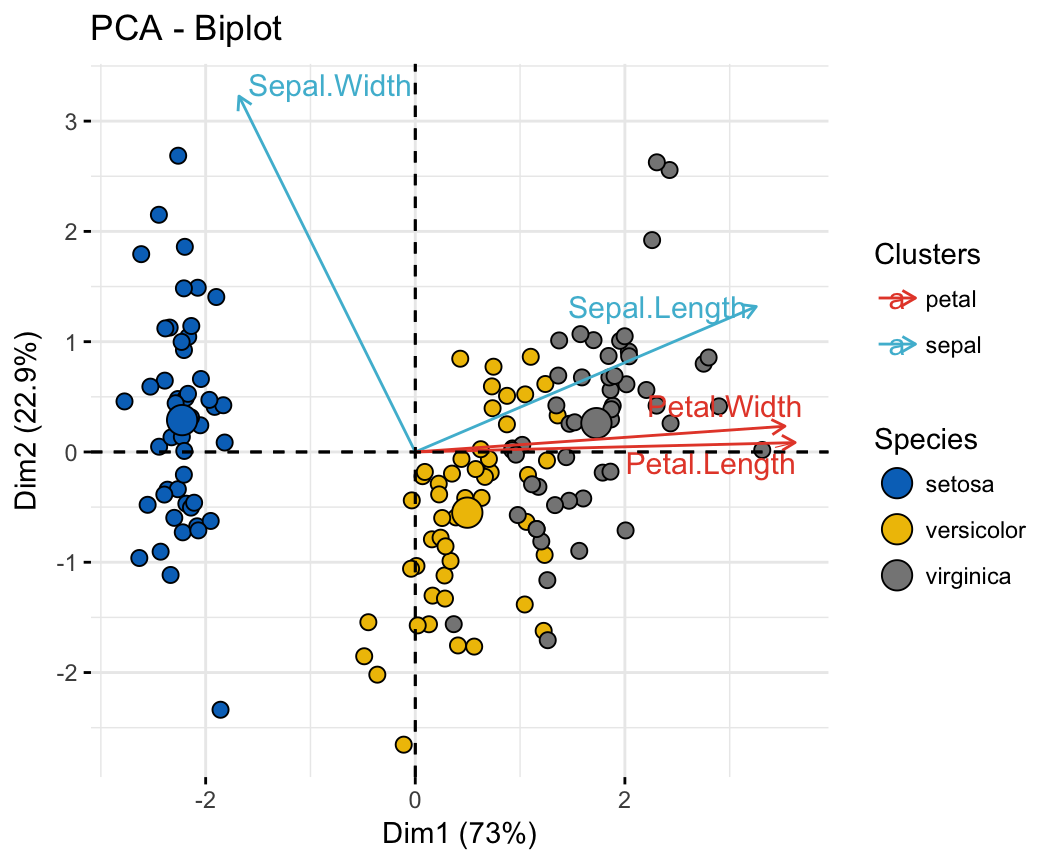
legend.title = list(fill = "Species", color = "Clusters"),

repel = TRUE # Evite le chévauchement du texte

)+

ggpubr::fill\_palette("jco")+ # Couleur des individus

ggpubr::color\_palette("npg") # Couleur des variables



Un autre exemple complexe est de colorer les individus par groupes (couleurs discrètes) et les variables par leurs contributions aux composantes principales (gradient de couleurs). En plus, nous modifierons la transparence des variables par leurs contributions en utilisant l’argument alpha.var.

fviz\_pca\_biplot(iris.pca,

# Individus

geom.ind = "point",

fill.ind = iris$Species, col.ind = "black",

pointshape = 21, pointsize = 2,

palette = "jco",

addEllipses = TRUE,

# Variables

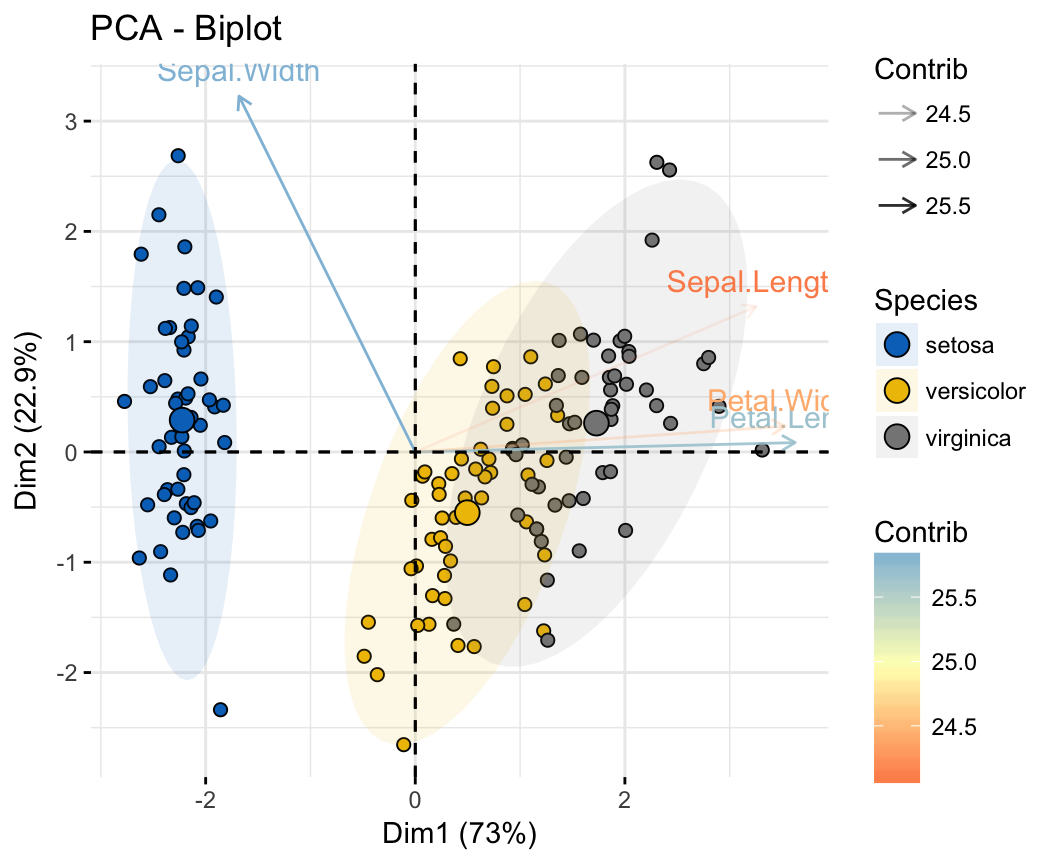
alpha.var ="contrib", col.var = "contrib",

gradient.cols = "RdYlBu",

legend.title = list(fill = "Species", color = "Contrib",

alpha = "Contrib")

)



## Eléments supplémentaires

### Définition et types

Comme décrit ci-dessus (section @ref(pca-data-format)), le jeu de données decathlon2 contient des variables continues supplémentaires (quanti.sup, colonnes 11:12), des variables qualitatives supplémentaires ( quali.sup, colonne 13) et des individus supplémentaires (ind.sup, lignes 24:27).

Les variables et individus supplémentaires ne sont pas utilisés pour la détermination des composantes principales. Leurs coordonnées sont prédites en utilisant uniquement les informations fournies par l’analyse en composantes principales effectuée sur les variables/individus actifs.

### Spécification dans l’ACP

Pour spécifier des individus et des variables supplémentaires, la fonction PCA() peut être utilisée comme suit:

PCA(X, ind.sup = NULL,

quanti.sup = NULL, quali.sup = NULL, graph = TRUE)

* X: un data frame. Les lignes sont des individus et les colonnes sont des variables numériques.
* ind.sup: un vecteur numérique spécifiant les positions des individus supplémentaires
* quanti.sup, quali.sup: un vecteur numérique spécifiant, respectivement, les positions des variables quantitatives et qualitatives
* graph: une valeur logique. Si TRUE, un graphique est affiché.

Par exemple, tapez ceci:

res.pca <- PCA(decathlon2, ind.sup = 24:27,

quanti.sup = 11:12, quali.sup = 13, graph=FALSE)

### Variables quantitatives

* Résultats prédites (coordonnées, corrélation et cos2) pour les variables quantitatives supplémentaires:

res.pca$quanti.sup

## $coord

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## Rank -0.701 -0.2452 -0.183 0.0558 -0.0738

## Points 0.964 0.0777 0.158 -0.1662 -0.0311

##

## $cor

## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## Rank -0.701 -0.2452 -0.183 0.0558 -0.0738

## Points 0.964 0.0777 0.158 -0.1662 -0.0311

##

## $cos2

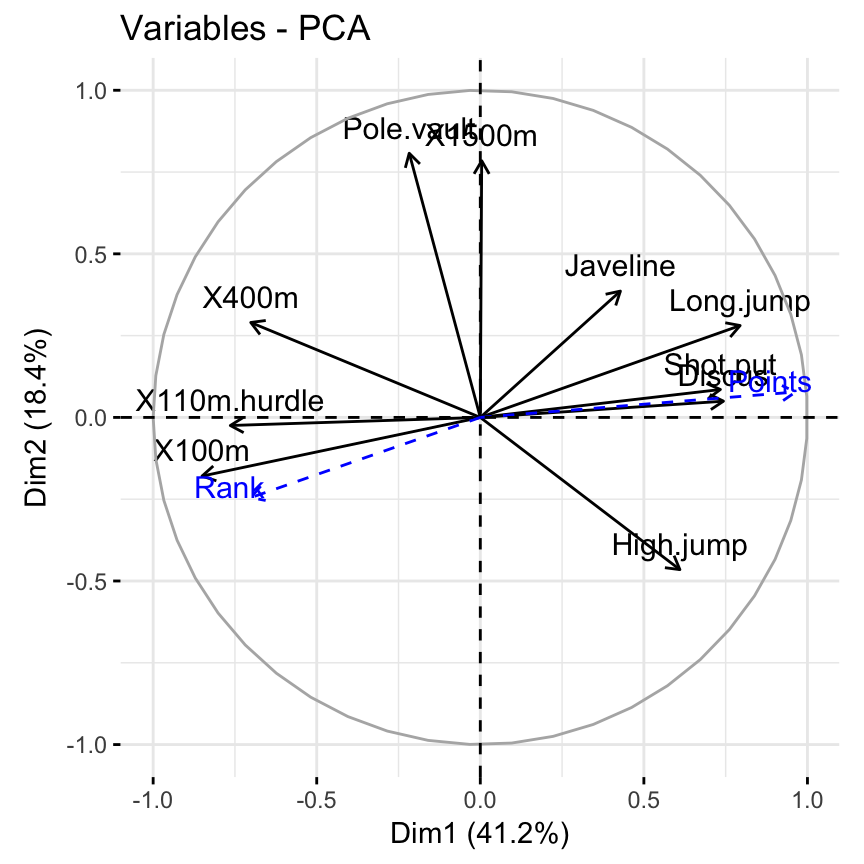
## Dim.1 Dim.2 Dim.3 Dim.4 Dim.5

## Rank 0.492 0.06012 0.0336 0.00311 0.00545

## Points 0.929 0.00603 0.0250 0.02763 0.00097

* Visualiser toutes les variables (actives et complémentaires):

fviz\_pca\_var(res.pca)



Notez que, par défaut, les variables quantitatives supplémentaires sont indiquées en couleur bleue et en pointillé.

Autres arguments pour personnaliser le graphique:

# Changer la couleur des variables

fviz\_pca\_var(res.pca,

col.var = "black", # Variables actives

col.quanti = "red" # variables quantitatives supl.

)

# Cacher les variables actives sur le graphique,

# ne montrent que des variables supplémentaires

fviz\_pca\_var(res.pca, invisible = "var")

# Cacher les variables supplémentaires

fviz\_pca\_var(res.pca, invisible = "quanti.sup")

En utilisant fviz\_pca\_var(), les variables supplémentaires quantitatives sont affichées automatiquement sur le graphique du cercle de corrélation. Notez que vous pouvez ajouter les variables quanti.sup manuellement, en utilisant la fonction fviz\_add(), pour plus de personnalisation. Un exemple est illustré ci-dessous.

# Graphique des variables actives

p <- fviz\_pca\_var(res.pca, invisible = "quanti.sup")

# Ajouter des variables actives supplémentaires

fviz\_add(p, res.pca$quanti.sup$coord,

geom = c("arrow", "text"),

color = "red")

### Indivius

* Résultats prédites pour les individus supplémentaires:

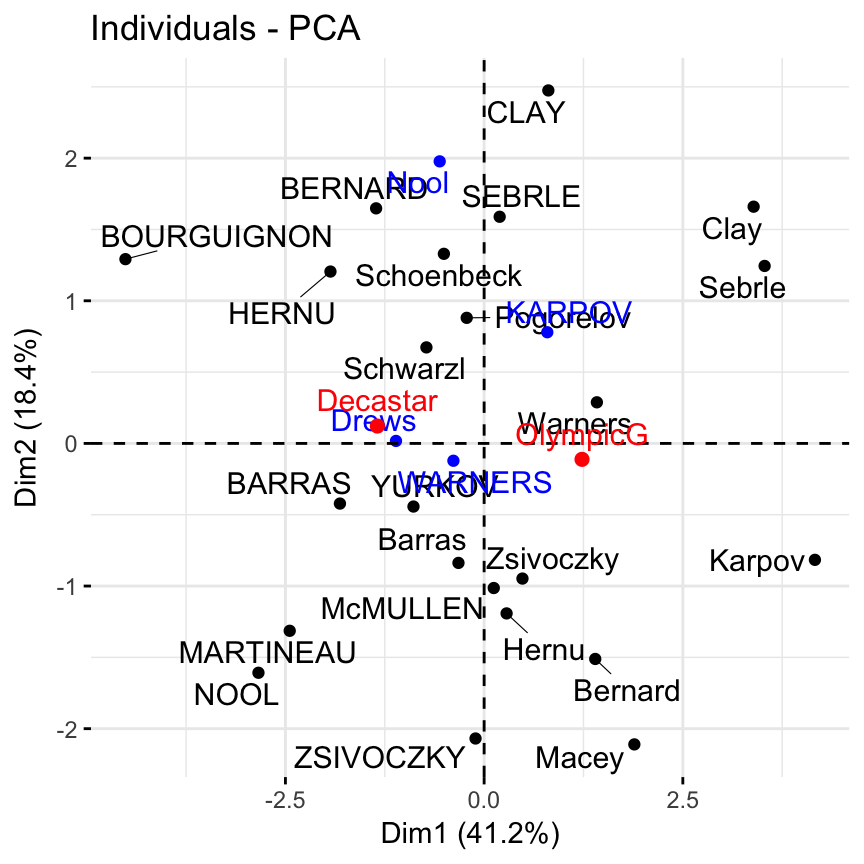
res.pca$ind.sup

* Visualiser tous les individus (actifs et supplémentaires). Sur le graphique, vous pouvez ajouter aussi les variables qualitatives supplémentaires (quali.sup), dont les coordonnées sont accessibles à l’aide de la fonction res.pca\$quali.supp\$coord.

p <- fviz\_pca\_ind(res.pca, col.ind.sup = "blue", repel = TRUE)

p <- fviz\_add(p, res.pca$quali.sup$coord, color = "red")

p



Les individus supplémentaires sont représentés en bleu. Les variables qualitatives supplémentaires sont indiquées en rouge.

### Variables qualitatives

Dans la section précédente, nous avons montré que vous pouvez ajouter les variables qualitatives supplémentaires sur les individus en utilisant fviz\_add().

Notez que les variables qualitatives supplémentaires peuvent également être utilisées pour colorer les individus par groupes. Cela peut aider à interpréter les données. Le jeu de données decathlon2 contient une variable qualitative supplémentaire dans la colonne 13 correspondant aux types de compétitions.

Les résultats concernant la variable qualitative supplémentaire sont les suivants:

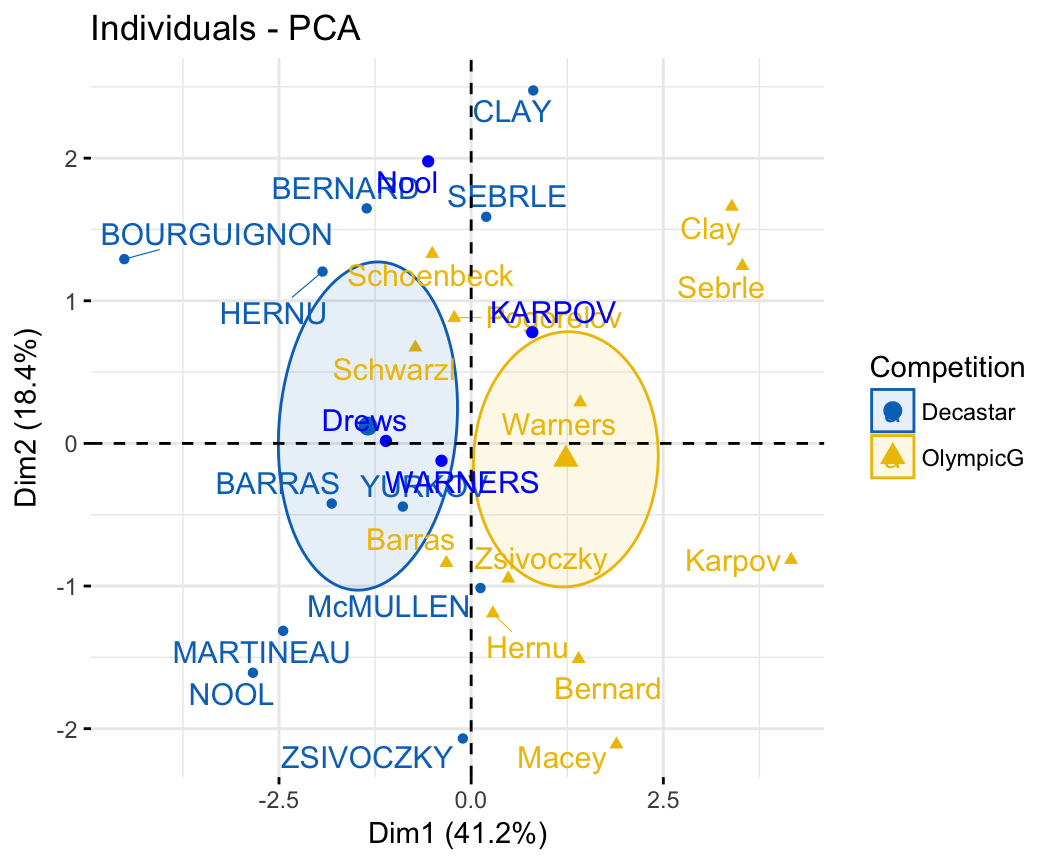
res.pca$quali

Pour colorer les individus par une variable qualitative supplémentaire, l’argument habillage sert à spécifier la position de la variable qualitative supplémentaire. Historiquement, ce nom d’argument provient du package FactoMineR. Pour garder la cohérence entre FactoMineR et factoextra, nous avons décidé de garder le même nom.

fviz\_pca\_ind(res.pca, habillage = 13,

addEllipses =TRUE, ellipse.type = "confidence",

palette = "jco", repel = TRUE)



Rappelons que, pour supprimer les points moyens des groupes, spécifiez l’argument mean.point = FALSE.

## Filtrer les résultats

Si vous avez un nombre élevé d’individus / variables, il est possible de visualiser seulement certains d’entre eux en utilisant les arguments select.ind et select.var.

select.ind, select.var: une sélection d’individus / variables à visualiser. Les valeurs autorisées sont NULL ou une listcontenant le nom des arguments, cos2 ou contrib:

* name: est un vecteur de caractères contenant les noms des individus / variables à visualiser
* cos2: si cos2 est dans [0, 1], ex: 0.6, alors les individus / variables avec un cos2> 0.6 sont montrés. si cos2> 1, ex: 5, les top 5 individus / variables actifs ainsi que les top 5 individus/ variables supplémentaires avec le cos2 le plus élevé sont montrés
* contrib: si contrib> 1, ex: 5, alors les top 5 individus / variables avec les contributions les plus importantes sont montrés

# Visualiser les variables avec cos2> = 0.6

fviz\_pca\_var (res.pca, select.var = list(cos2 = 0.6))

# Top 5 variables actives avec le cos2 le plus elevé

fviz\_pca\_var (res.pca, select.var = list(cos2 = 5))

# Sélectionnez par noms

name <- list (name = c ("Long.jump", "High.jump", "X100m"))

fviz\_pca\_var (res.pca, select.var = name)

# Top 5 des individus/variables les plus contibutifs

fviz\_pca\_biplot (res.pca, select.ind = list (contrib = 5),

select.var = list (contrib = 5),

ggtheme = theme\_minimal())

Lorsque la sélection se fait selon les valeurs de contribution, les individus / variables supplémentaires ne sont pas représentés parce qu’ils ne contribuent pas à la construction des axes.

## Exportation des résultats

### Exporter les graphiques en PDF / PNG

Le package factoextra produit des graphiques de type ggplot2. Pour enregistrer un ggplot, le code R standard est comme suit:

# Enregistré au format pdf

pdf ("myplot.pdf")

print (myplot)

dev.off ()

Dans les exemples suivants, nous vous montrerons comment enregistrer les différents graphiques en fichiers pdf ou png.

La première étape consiste à créer les graphiques que vous voulez en tant qu’objets R:

# Scree plot

scree.plot <- fviz\_eig (res.pca)

# Graphique des d'individus

ind.plot <- fviz\_pca\_ind (res.pca)

# Graphique des variables

var.plot <- fviz\_pca\_var (res.pca)

Ensuite, les graphiques peuvent être exportés dans un seul fichier pdf comme suit:

pdf ("PCA.pdf") # Créer un nouveau périphérique pdf

print (scree.plot)

print (ind.plot)

print (var.plot)

dev.off () # Ferme le périphérique pdf

Notez que, en utilisant le code R ci-dessus, vous créerez le fichier PDF dans votre répertoire de travail actuel. Pour voir le chemin d’accès de votre répertoire de travail actuel, tapez getwd() dans la console R.

Pour enregistrer chaque graphique vers un fichier png spécifique, le code R ressemble à ceci:

# Enregistrer sur un fichier png

png ("pca-scree-plot.png")

print(scree.plot)

dev.off ()

# Enregistrer les individus dans un fichier png

png ("pca-variables.png")

print(var.plot)

dev.off ()

# Enregistrer les variables dans un fichier png

png ("pca-individuals.png")

print(ind.plot)

dev.off ()

Une autre alternative, pour l’exportation de ggplots, est d’utiliser la fonction ggexport() [dans ggpubr]. Nous préférons ggexport(), car c’est très simple. En une seule ligne de code R, il nous permet d’exporter des graphiques individuels vers un fichier (pdf, eps ou png) (un graphique par page). Il peut également organiser les graphiques (2 par page, par exemple) avant de les exporter. Les exemples ci-dessous montrent comment exporter des ggplots en utilisant ggexport().

Exportez des graphiques individuels dans un fichier pdf (un graphique par page):

**library** (ggpubr)

ggexport (plotlist = list(scree.plot, ind.plot, var.plot),

filename = "PCA.pdf")

Organiser et exporter. Spécifiez nrow et ncol pour afficher plusieurs graphiques sur la même page:

ggexport (plotlist = list(scree.plot, ind.plot, var.plot),

nrow = 2, ncol = 2,

filename = "PCA.pdf")

Exporter des graphiques vers des fichiers png. Si vous spécifiez une liste de graphiques, plusieurs fichiers png seront automatiquement créés pour contenir chaque graphique.

ggexport (plotlist = list(scree.plot, ind.plot, var.plot),

filename = "PCA.png")

### Exporter les résultats vers les fichiers txt / csv

Les résultats de l’ACP (coordonnées et contributions des individus / variables) peuvent être exportés, dans un fichier TXT / CSV, en utilisant la fonction write.infile() [FactoMineR]:

# Exporter vers un fichier TXT

write.infile (res.pca, "pca.txt", sep = "\t")

# Exporter vers un fichier CSV

write.infile (res.pca, "pca.csv", sep = ";")

## Résumé

En conclusion, nous avons décrit comment calculer et interpréter l’analyse en composantes principales (ACP). Nous avons calculé l’ACP en utilisant la fonction PCA() [FactoMineR]. Ensuite, nous avons utilisé le package R factoextra pour produire une visualisation ggplot2 des résultats de l’ACP.

Il existe d’autres fonctions [packages] pour calculer l’ACP dans R:

1. prcomp() [stats]

res.pca <- prcomp (iris [, -5], scale. = TRUE)

1. princomp() [stats]

res.pca <- princomp (iris [, -5], cor = TRUE)

1. dudi.pca() [ade4]

**library** ("ade4")

res.pca <- dudi.pca (iris [, -5], scannf = FALSE, nf = 5)

1. epPCA() [ExPosition]

**library** ("ExPosition")

res.pca <- epPCA (iris [, -5], graph = FALSE)

Peu importe les fonctions que vous décidez d’utiliser, dans la liste ci-dessus, le package factoextra peut gérer les résultats pour créer de beaux graphiques similaires à ceux décrits dans les sections précédentes pour FactoMineR:

fviz\_eig (res.pca) # Scree plot

fviz\_pca\_ind (res.pca) # Graphique des individus

fviz\_pca\_var (res.pca) # Graphique des variables

## Autres lectures

Pour les bases mathématiques de l’ACP, reportez-vous aux cours, articles et livres suivants (en anglais):

* Principal component analysis (article) (Abdi and Williams 2010). <https://goo.gl/1Vtwq1>.
* Exploratory Multivariate Analysis by Example Using R (book) (Husson, Le, and Pagès 2017).
* Principal Component Analysis (book) (Jollife 2002).

## References

Abdi, Hervé, and Lynne J. Williams. 2010. “Principal Component Analysis.” John Wiley and Sons, Inc. WIREs Comp Stat 2: 433–59. <http://staff.ustc.edu.cn/~zwp/teach/MVA/abdi-awPCA2010.pdf>.

Husson, Francois, Sebastien Le, and Jérôme Pagès. 2017. Exploratory Multivariate Analysis by Example Using R. 2nd ed. Boca Raton, Florida: Chapman; Hall/CRC. <http://factominer.free.fr/bookV2/index.html>.

Jollife, I.T. 2002. Principal Component Analysis. 2nd ed. New York: Springer-Verlag. <https://goo.gl/SB86SR>.

Kaiser, Henry F. 1961. “A Note on Guttman’s Lower Bound for the Number of Common Factors.” British Journal of Statistical Psychology 14: 1–2.

Peres-Neto, Pedro R., Donald A. Jackson, and Keith M. Somers. 2005. “How Many Principal Components? Stopping Rules for Determining the Number of Non-Trivial Axes Revisited.” British Journal of Statistical Psychology 49: 974–97.