

Chapitre 3

Analyse en Composantes Principales

Nos allons procéder par traiter un exemple, puis nous nous intéresserons à l'exemple traité pendant la séance du cours et enfin nous étudierons une situation pratique.

1. Lancer R et modifier le répertoire de travail en allant dans **Fichier -> Changer le Répertoire Courant** et en choisissant le répertoire **Bureau/TP_M1MIASHS_SSD_AD** qui a été créé.
2. Ouvrir une fenêtre d'éditeur **Fichier -> Nouveau Script**.
3. Sauver le fichier dans le répertoire courant, par exemple, sous le nom **TP3.R** : **Fichier -> Sauver sous**
4. Penser à sauvegarder régulièrement le contenu du fichier **TP3.R** en appuyant sur les touches **Ctrl et S**.

3.1 Pratique de l'ACP

L'analyse en composantes principales (ACP), ou principal component analysis (PCA) en anglais, permet d'analyser et de visualiser un jeu de données contenant des individus décrits par plusieurs variables quantitatives. C'est une méthode statistique qui permet d'explorer des données dites multivariées (données avec plusieurs variables). Chaque variable pourrait être considérée comme une dimension différente. Si vous avez plus de 3 variables dans votre jeu de données, il pourrait être très difficile de visualiser les 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 originelles. 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 (i.e., 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.

Dans ce qui suit, nous décrivons les bases de l'ACP et démontrons comment calculer, visualiser et interpréter l'ACP en utilisant le logiciel R. De plus, nous montrerons comment révéler les variables les plus importantes qui expliquent les variations dans un jeu de données.

3.1.1 Notions de base

Comprendre les détails de l'ACP nécessite une connaissance de l'algèbre linéaire. Dans la figure 3.1, les données sont représentées dans le système de coordonnées $X - Y$. La *réduction de la dimension* est obtenue en identifiant les directions principales, appelées *composantes principales*, dans lesquelles les données varient. L'ACP suppose que les directions avec les plus grandes variances sont les plus

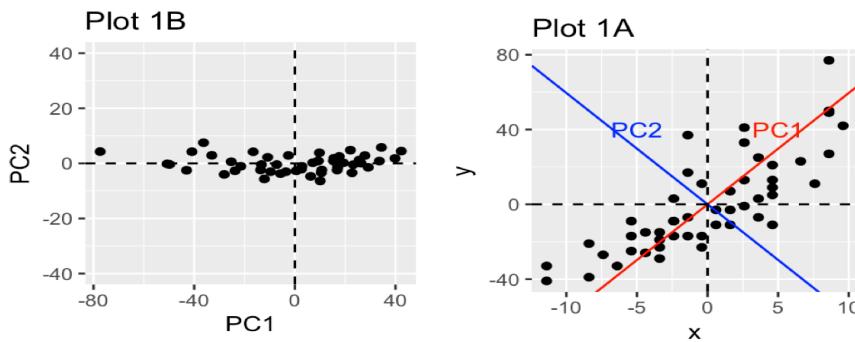


FIGURE 3.1 – Exemple

importantes (i.e., principales).

Dans la figure ci-dessous, l'axe PC1 est **le premier axe principal** le long duquel les échantillons présentent la plus grande variation. L'axe PC2 est **la seconde direction la plus importante et orthogonal à l'axe PC1**.

Les dimensions de notre jeu de données peuvent être réduites à une seule dimension en projetant chaque échantillon sur le premier axe principal (Plot 1B).

Techniquement parlant, la quantité de variance expliquée par chaque composante principale est mesurée par ce que l'on appelle valeur propre.

Notez que l'ACP est particulièrement utile lorsque les variables, dans le jeu de données, sont fortement corrélées. La corrélation indique qu'il existe une redondance dans les données. En raison de cette redondance, l'ACP peut être utilisée pour réduire les variables d'origine en un nombre plus petit de nouvelles variables (= **composantes principales**), ces dernières expliquant la plus grande

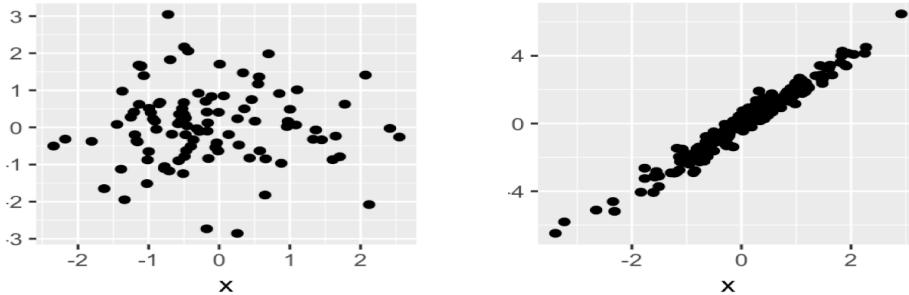


FIGURE 3.2 – Redondance faible et redondance forte

partie de la variance contenue dans les variables d'origine.

En résumé, l'analyse en composantes principales permet :

- d'identifier des “profils cachés” dans un jeu de données,
- de réduire les dimensions des données en enlevant la redondance des données,
- d'identifier les variables corrélées

3.1.2 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`).

Pour ceci, installer les deux packages comme suit :

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

Charger-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 3.3, les données utilisées ici décrivent la performance des athlètes lors de deux événements sportifs (Decastar et OlympicG). Elles contiennent 27 individus

(athlètes) décrits par 13 variables.

name	100m	Long.jump	//	Javeline	1500m	Rank	Points	Competition
SEBRLE	11.04	7.58		63.19	291.7	1	8217	Decastar
CLAY	10.76	7.4		60.15	301.5	2	8122	Decastar
Macey	10.89	7.47		58.46	265.42	4	8414	OlympicG
Warners	10.62	7.74		55.39	278.05	5	8343	OlympicG
\\								
Zsivoczky	10.91	7.14		63.45	269.54	6	8287	OlympicG
Hernu	10.97	7.19		57.76	264.35	7	8237	OlympicG
Pogorelov	10.95	7.31		53.45	287.63	11	8084	OlympicG
Schoenbeck	10.9	7.3		60.89	278.82	12	8077	OlympicG
Baras	11.14	6.99		64.55	267.09	13	8067	OlympicG
KARPOV	11.02	7.3		50.31	300.2	3	8099	Decastar
WARNERS	11.11	7.6		51.77	278.1	6	8030	Decastar
Nool	10.8	7.53		61.33	276.33	8	8235	OlympicG
Drews	10.87	7.38		51.53	274.21	19	7926	OlympicG

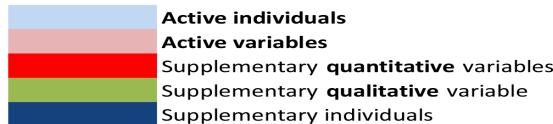


FIGURE 3.3 – Les données de Décathlon

Il faut noter que seulement certains de ces individus et variables seront utilisés pour effectuer l'analyse en composantes principales. Les coordonnées des individus et des variables restants seront prédites après l'ACP.

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 :

$$\frac{x_i - \text{mean}(x)}{\text{sd}(x)}$$

où `mean(x)` désigne la moyenne des valeurs de x , et `sd(x)` est l'écart type (SD).

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

Remarque 3.1.1. Par défaut, la fonction `PCA()` [dans `FactoMineR`], normalise les données automatiquement pendant l'ACP ; donc, on n'a 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.

3.1.3 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.
- On peut 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 (Jolliffe (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_screepplot()** [package **factoextra**].

```
fviz_eig(res.pca, addlabels = TRUE, ylim = c(0, 50))
```

De la figure 3.4, nous pourrions vouloir nous arrêter à la cinquième composante prin-

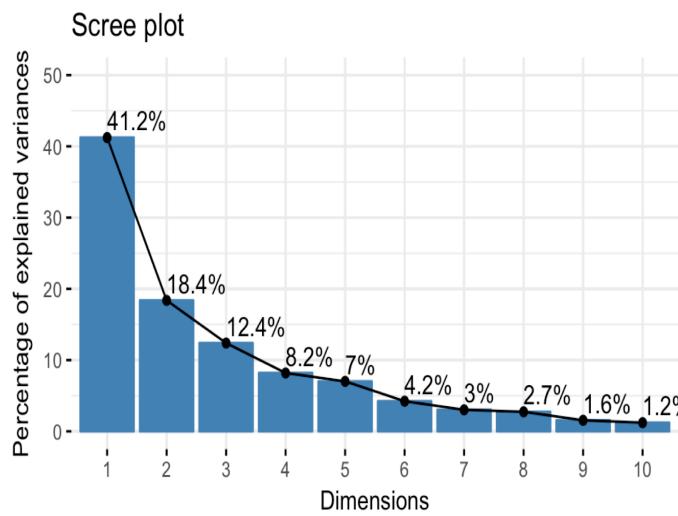


FIGURE 3.4 – Scree plot

pale. 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)`.

Il faut noter 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$coord)
# Cos2 : qualité de représentation
head(var$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

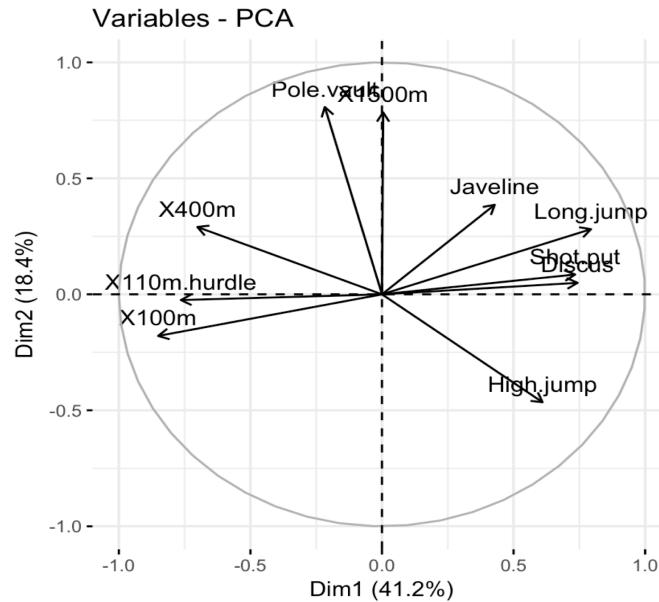


FIGURE 3.5 – Cercle des corrélations

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

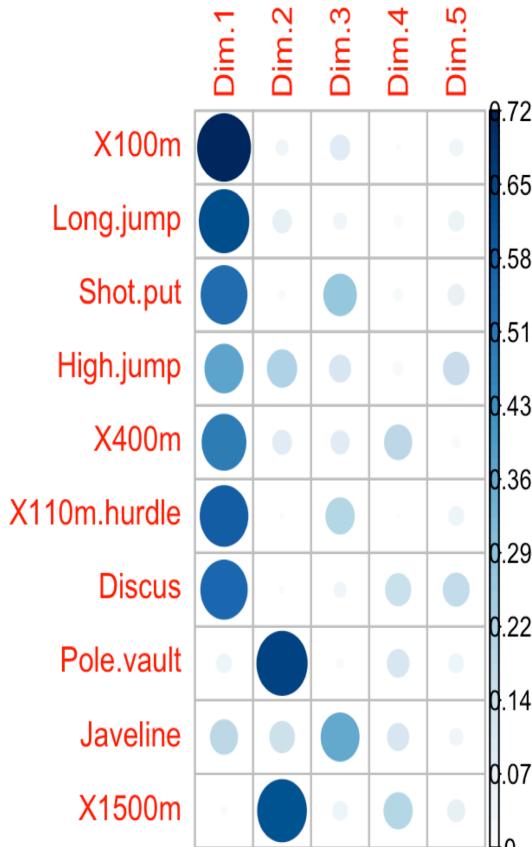


FIGURE 3.6 – qualité de représentation

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,

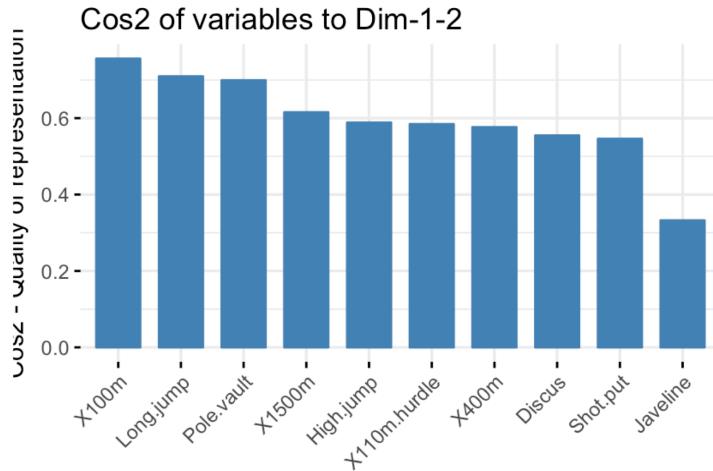


FIGURE 3.7 – Cos2, Coska

- 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 circonference 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 proches 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

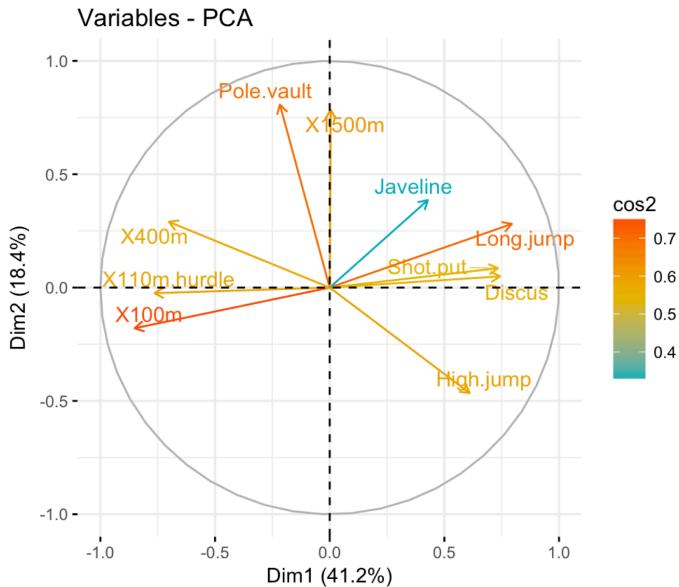


FIGURE 3.8 – Coloration en fonction de \cos^2

de leurs valeurs de \cos^2 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

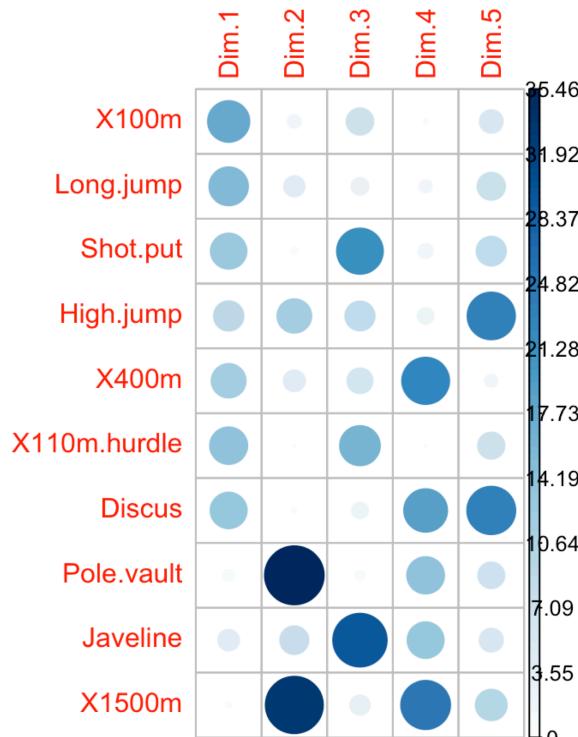


FIGURE 3.9 – contributions des variables

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 :
```

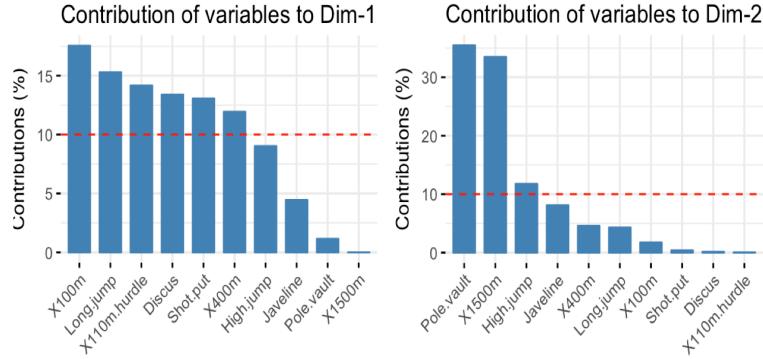


FIGURE 3.10 – Contributions des variables aux axes 1-2

```
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 $1/\text{length(variables)} = 1/10 = 10\%$. 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 : $[(10 * \text{Eig1}) + (10 * \text{Eig2})]/(\text{Eig1} + \text{Eig2})$

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") )
```

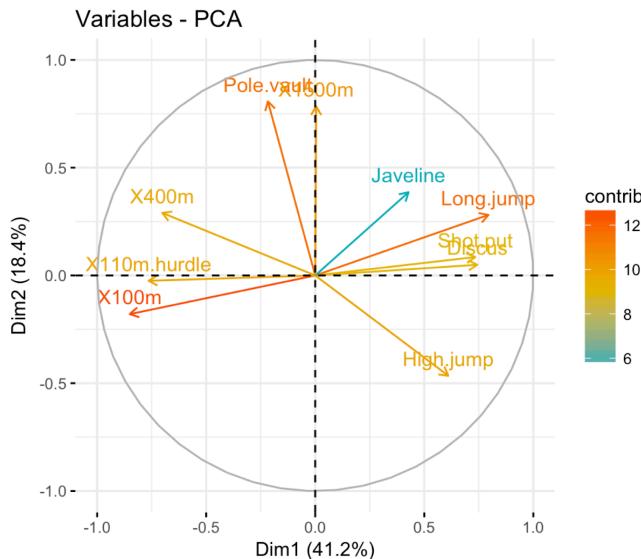


FIGURE 3.11 – Coloration des variables

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")
```

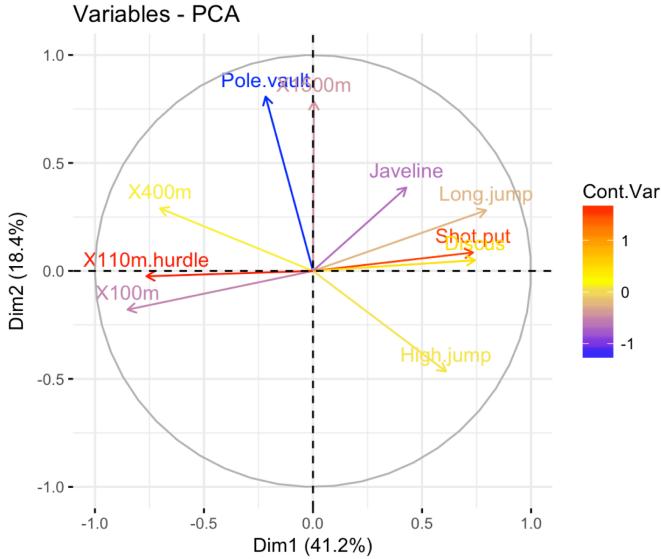


FIGURE 3.12 – Scree plot

- *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 R¹.

```
# 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")
```

1. <https://goo.gl/DmJ5y5>

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")
```

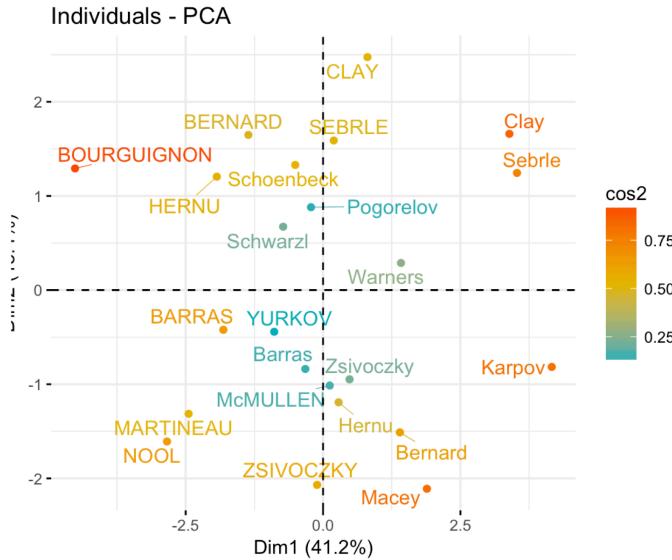


FIGURE 3.13 – Contribution des individus

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 longueur 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.

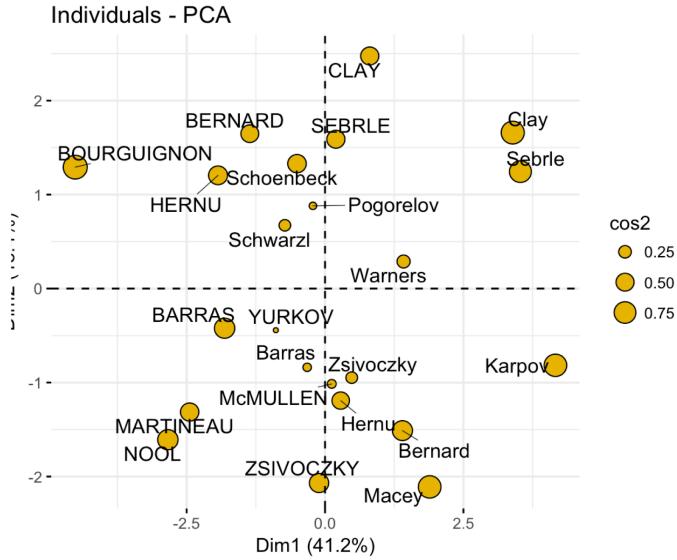


FIGURE 3.14 – Contributions des individus

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" )
```

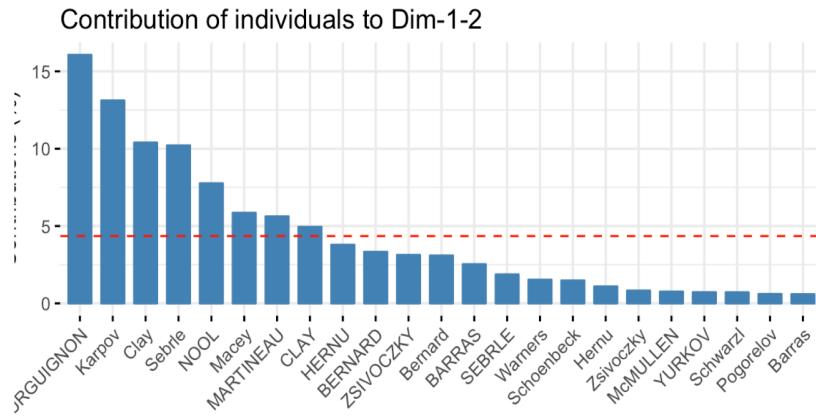


FIGURE 3.15 – Pourcentages de contributions des individus

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
```

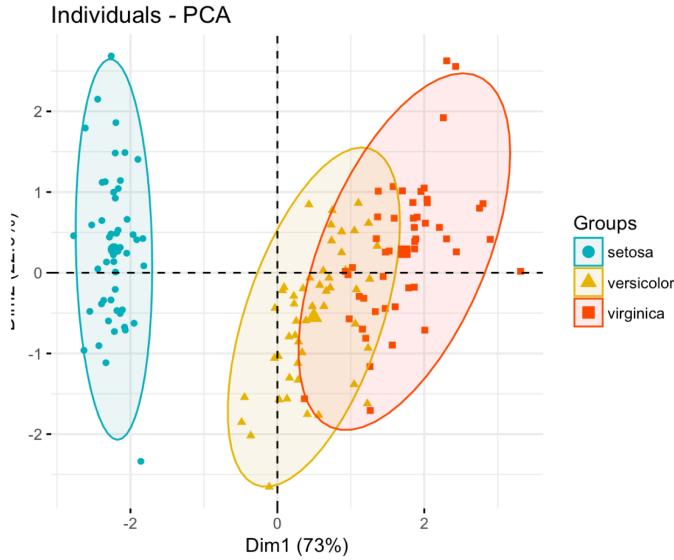


FIGURE 3.16 – Ellipses de concentration autour des groupes

```
palette = "jco" )
```

— **Personnalisation du graphique.** Il faut noter 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.

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"))
```

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", "text") 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

labelsize : taille du texte, par exemple : labelsize = 4.

pointsize : taille des points, par exemple : pointsize = 1.5.

arrowsize : taille des flèches. Contrôle l'épaisseur des flèches, par exemple : arrowsize = 0.5.

pointshape : forme des points, pointshape = 21. Tapez ggpibr ::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 appliquée.

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 ggpibr ::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() [package ggpibr]

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

- Le titre 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" )
```

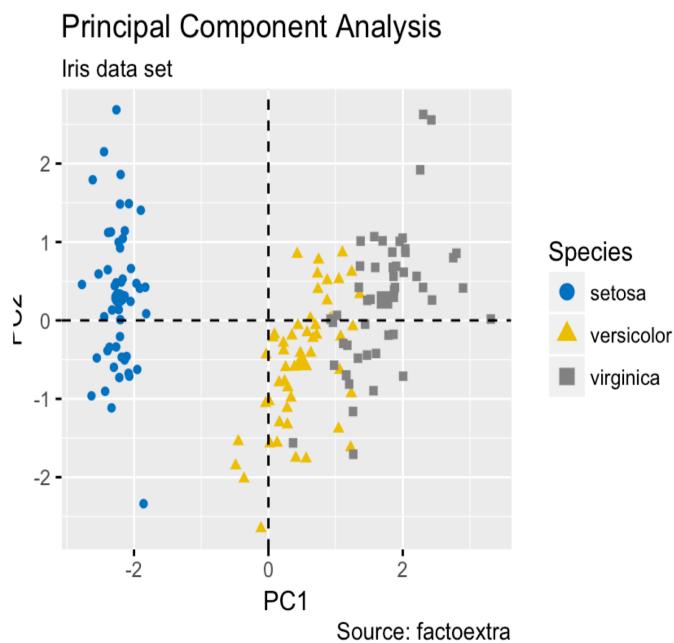


FIGURE 3.17 – Jeu de données : Iris

3.1.4 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 individus)`

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.

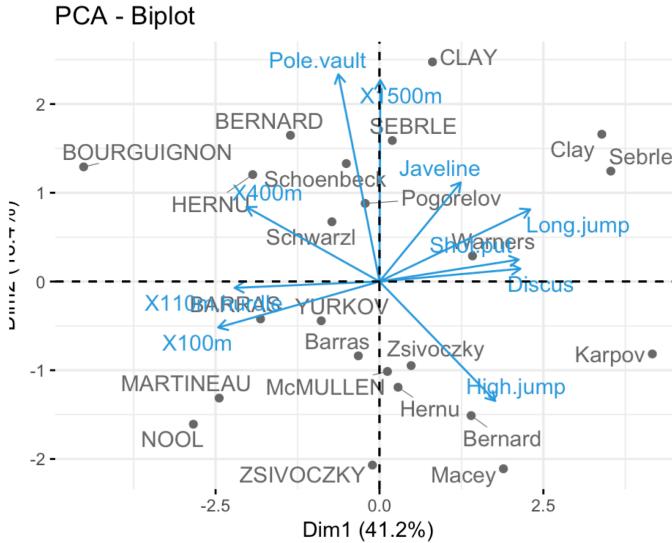


FIGURE 3.18 – Graphique individus-variables

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

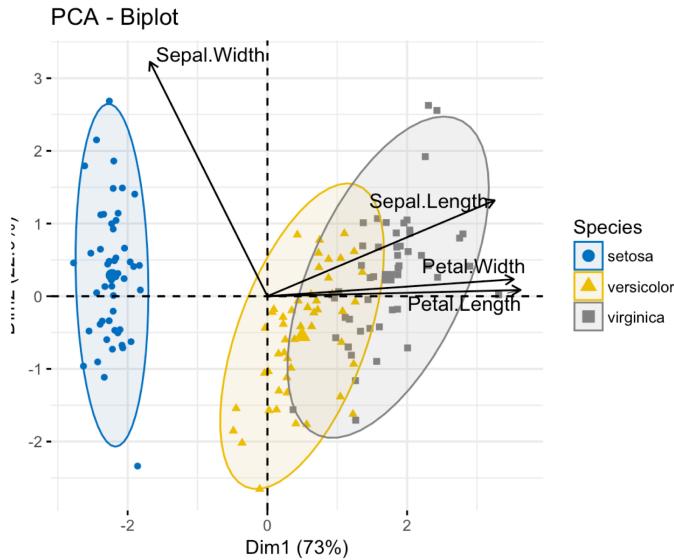


FIGURE 3.19 – Biplot individus-variables avec des couleurs

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 ggpibr].

```
fviz_pca_biplot(iris.pca,
Colleur de remplissage des individus 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 chevauchement 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",
```

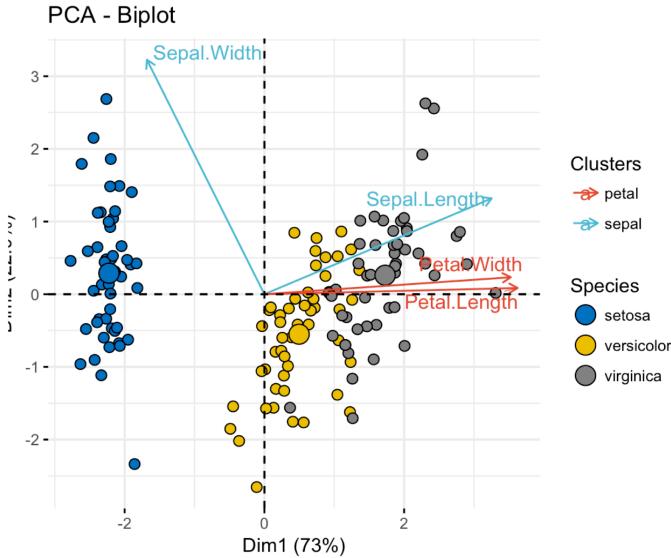


FIGURE 3.20 – Biplot individus-variables avec des couleurs personnalisées

```
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,
```

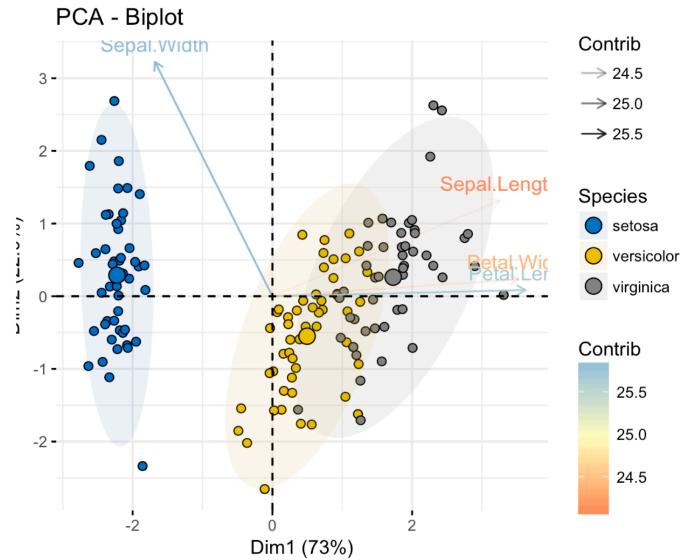


FIGURE 3.21 – Biplot individus-variables avec coloration des individus par groupes

```
quanti.sup = 11 :12, quali.sup = 13, graph=FALSE)
— Variables quantitatives. Résultats prédictes (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

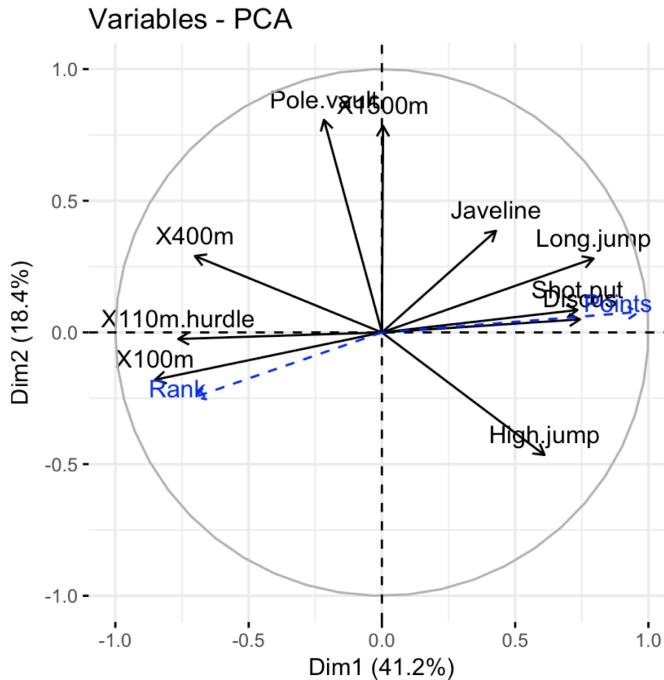


FIGURE 3.22 – Visualisation des variables actives et complémentaires

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 suppl.
)
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$quali.sup$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
```

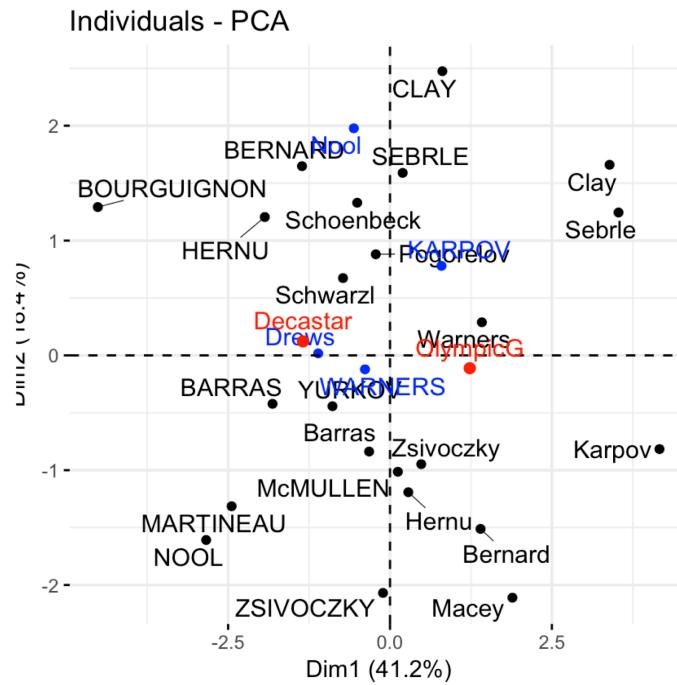


FIGURE 3.23 – Résultats prédites pour les variables supplémentaires

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)
```

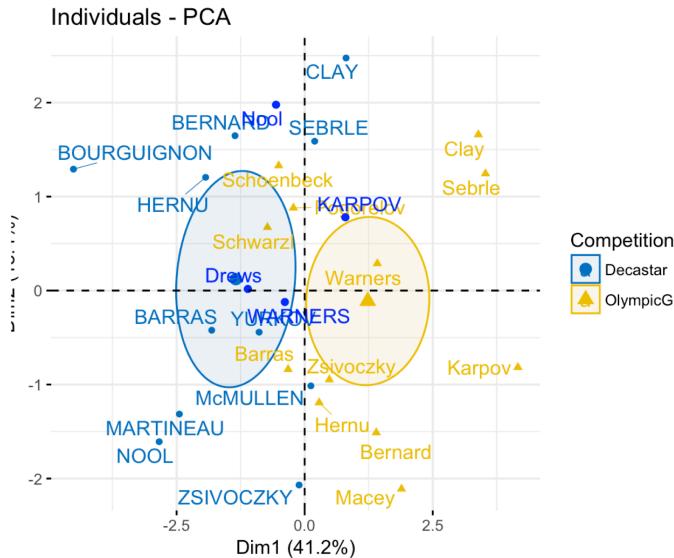


FIGURE 3.24 – Coloration des individus par une variable qualitative habillage

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 liste contenant 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  $\text{cos}2 > 0.6$  sont montrés. si  $\text{cos}2 > 1$ , 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 élevé
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 contributifs
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") # Crée 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 `ggbpubr`]. 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 (ggbpubr)
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 = "")

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 :

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

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

dudi.pca() [ade4]
library ("ade4") res.pca <- dudi.pca (iris [, -5], scannf = FALSE, nf = 5)

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) (Jolliffe 2002).

#### — Références

Abdi, Hervé, and Lynne J. Williams. 2010. Principal Component Analysis. John Wiley and Sons, Inc. WIREs Comp Stat 2 : 43359.  
<http://staff.ustc.edu.cn/zwp/teach/MVA/abdi-awPCA2010.pdf>.

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

Jolliffe, 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 : 12.

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 : 97497.

## 3.2 Exemple sur les variétés d'eaux minérales

On s'intéresse à  $n = 8$  variétés d'eaux minérales. Celles-ci sont décrites par  $p = 13$  variables.

1. Charger les données qui se trouvent dans le fichier `eaux.RData`

```
load("eaux.RData")
print(data[,1 :3])
X <- data # Matrice d'origine des données quantitatives
dim(X)
```

2. Description bivariée, distances et corrélations

- (a) Diagrammes de dispersion de toutes les paires de variables.

```
pairs(data[,1 :5])
library(GGally)
ggpairs(data[,1 :5])
```

- (b) Matrices des distances entre les individus et des corrélations entre les variables.

```
dist(data)
cor(data[,1 :5])
```

3. Données centrées et réduites/standardisées

- (a) Moyennes et écart-types des colonnes

```
m <- apply(X,2,mean) #mean
print(m,digits=3)
s <- apply(X,2,sd) #standard deviation
print(s,digits=3)
```

- (b) Données centrées et réduites.

```
Y <- sweep(X,2,m,"-")
apply(Y,2,mean)
apply(Y,2,sd)
```

```
Z <- sweep(Y,2,s,"/")
apply(Z,2,sd)
```

- (c) La matrice des covariances  $\frac{1}{n}Y^tY$ .

```
#Attention division par n-1=7
(t(as.matrix(Y))%*%as.matrix(Y)/7)[1,2]
cov(X)[1,2]
all.equal(cov(X),t(as.matrix(Y))%*%as.matrix(Y)/7)
```

- (d) La matrice des corrélations  $\frac{1}{n}Z^tZ$ .

```
#Attention division par n-1=7
(t(as.matrix(Z))%*%as.matrix(Z)/7)[1,2]
cor(X)[1,2]
all.equal(cor(X),t(as.matrix(Z))%*%as.matrix(Z)/7)
```

4. ACP de la matrice des covariances ou de la matrice des corrélations ? Nous utilisons la fonction PCA du package R FactoMineR.

```
library(FactoMineR) # package de R dédié à l'analyse des données multivariées
?PCA
```

- (a) ACP de la matrice des covariances : analyse des lignes et des colonnes de la matrice centrée  $Y$  i.e., la décomposition en spectrale de la matrice des covariances  $C = \frac{1}{n}Y^tT$ .

L'argument scale.unit=FALSE est utilisé.

```
res <- PCA(X,graph=FALSE,scale.unit=FALSE) # original data in input
plot(res,choix="ind",cex=1.5,title="")
plot(res,choix="var",cex=1.5,title "")
```

- (b) ACP de la matrice des corrélations : analyse des lignes et des colonnes de la matrice centrée-réduite  $Z$  i.e., la décomposition spectrale de la matrice des corrélations  $R = \frac{1}{n}Z^tZ$ .

```
res <- PCA(X,graph=FALSE) # original data in input
plot(res,choix="ind",cex=1.5,title="", select="cos2 6")
plot(res,choix="var",cex=1.5,title "") #plot des individus et des variables sur les dimensions 3-4.
```

- (c) Le graphique des individus est construit à partir de la matrice  $F$  des coordonnées principales des individus (composantes principales).

```
F <- resindcoord
print(F[1 :4],digits=2) #plot the individuals on the first principal component
map using F
axes <- c(1,2)
plot(F[,axes],pch=19,col=4,cex=1)
abline(h=0,lty=2)
abline(v=0,lty=2)
text(F[,axes],labels=rownames(Z),pos=3,col=4,cex=1)
```

- (d) Les variances des composantes principales (les colonnes de  $F$ ) sont les valeurs propres de la matrice des corrélations.

```
res$eig[,1] #calculate the variance of the columns of F
n <- nrow(X)
apply(F,2,var)*(n-1)/n
```

- (e) Le graphiques des variables est obtenu à partir de la matrice  $A$  des coordonnées principales des variables (te loadings).

```
A <- resvarcoord
print(A[1 :4,],digits=2)#plot the variables on the first principal component
map using A
axes <- c(1,2)
plot(A[,axes],pch=19,col=4,cex=1)
abline(h=0,lty=2)
abline(v=0,lty=2)
text(A[,axes],labels=colnames(Z),pos=3,col=4,cex=1)
```

- (f) The loadings sont les corrélations entre les variables et les composantes principales.

```
cor(X,F[,1])
```

### 3.2.1 ACP avec d'autres fonctions de R

Il existe deux fonctions qui permettent de faire une ACP dans le package `stats` qui sont `princomp` et `prcomp`. Est ce qu'elles performent l'ACP, par défaut, sur la matrice des covariances ou sur la matrice des corrélations ?

```
?princomp
#pr <- princomp(X,cor=T) # on covariance matrix
?prcomp
pr <- prcomp(X,scale=TRUE) # on covariance matrix
pr$sdev^2 # eigenvalues
```

### 3.2.2 ACP “à la main”

- (a) *Via une décomposition spectrale (an eigen decomposition) (de  $C$  ou  $R$ )* : Les vecteurs propres  $v_1, \dots, v_q$  ( $q \leq r$  où  $r$  est le rang de  $Z$ ) de la matrice des corrélations  $R = \frac{1}{n}Z^tZ$  donne les composantes principales dans  $F = ZV$ , où  $V$  est la matrice dont les colonnes sont les  $q$  vecteurs propres.

```
R <- t(as.matrix(Z))%*%as.matrix(Z)/7
e <- eigen(R)
```

```

names(e)
dim(e$vectors) #matrix with the q=3 first eigenvectors
q <- 3
V <- e$vectors[,1 :q]
Fbis <- as.matrix(Z)%*%V
head(Fbis)
head(F[,1 :q]) #with the PCA function

```

Les valeurs  $\lambda_1, \dots, \lambda_q$  sont la variance des composantes principales (colonnes de \$F).

```

e$values[1 :q]
res$eig[1 :q,1] #output of the PCA function

```

- (b) *Via la décomposition en SVD avec métrique* : Dans l'ACP la métrique dans  $\mathbb{R}^n$  est la matrice diagonale des poids des lignes (ici  $1/n$ ) et la métrique dans  $\mathbb{R}^p$  est la matrice identité.

```

#first compute Z with the non corrected standard deviation
s <- apply(Y,2,sd)*sqrt((n-1)/n)
Z <- sweep(Y,2,s,"/")
e_svd <- svd(Z)
names(e_svd)
(e_svd$d^2)[1 :q]/n #division by n because of the metric
res$eig[1 :q,1] #output of the PCA function
U <- e_svd$u[,1 :q] #matrix of the q first left singular vectors
Fbisbis <- U%*%diag(e_svd$d[1 :q]) #principal components
head(Fbisbis)
head(F[,1 :q]) #output of the PCA function

```

### 3.2.3 Interprétation des résultats

- (a) *Inertie (variance) des composantes principales* :

```

res$eig
barplot(res$eig[,1])

```

- (b) *Carte/Map des individus* : Les coska (cosinus carrés) et la carte/map des 3 eaux minérales les mieux projetées/représentées.

```

resindcos2[,1 :3]
plot(res, choix="ind", cex=1.5, title="", select="cos2 3")

```

Contributions et carte/map des 3 eaux minérales qui contribuent le plus.

```

resindcos2[,1 :3]
plot(res, choix="ind", cex=1.5, title="", select="contrib 3")

```

- (c) *Carte/Map des variables* : Le cercle des corrélations.

```
resvarcoord[,1 :2]
plot(res,choix="var",title="",cex=1)
```

Interprétation de la carte des individus à partir de la carte des variables.

```
plot(res,choix="ind",title="",cex=1)
```

### 3.3 Effet taille

Considérons l'exemples des poissons contaminés par le mercure.

```
load("poissons.RData")
datalog <- data
datalog[,5 :10] <- log(data[,5 :10])
res <- PCA(datalog[,-1],quanti.sup=2 :3,quali.sup=1,graph=FALSE)
plot(res,choix="var",title="")
plot(res,choix="ind",invisible = "quali",label = "none",habillage=1,title = "")
```

### 3.4 Etude des données sur les pays l'OCDE

Les données proviennent de l'Agence Internationale de l'Energie (IEA=International Energy Agency).

On dispose, pour les 30 pays de l'OCDE, des valeurs suivantes :

- popul : population (millions d'habitants, 2002)
- CO2 : émission de CO2 par combustion de fuel (Mt, 2002)
- TPES : Total Primary Energy supply (millions de tonnes équivalent pétrole, 2002)
- electr : consommation d'électricité (teraWatts heure, 2003, 1 TWh=1 millier de milliards de Watts/heure)
- import : importation d'électricité (teraWatts heure, 2003)
- export : exportation d'électricité (teraWatts heure, 2003)
- fuel : production d'électricité d'origine fossile (teraWatts heure, 2003)
- nuclear : production d'électricité d'origine nucléaire (teraWatts heure, 2003)
- hydro : production d'électricité d'origine hydrolique (teraWatts heure, 2003)
- geoth : production d'électricité d'origine géothermique ou autre (teraWatts heure, 2003)
- zone : zone géographique (pacasi=Pacific/Asie, europ=Europe, ameri=Amérique)
- climat : climat prédominant (aride, temp=tempéré, arct=arctique/tempéré, medi=méditerranéen, mari=maritime)

1. Charger et explorer les données du fichier OCDE.txt
2. Avec la commande `pairs()` afficher les nuages de points de l'ensemble des variables.
3. Variation des données (variables actives et illustratives) : A cause de l'effet de taille, on considère les variables quantitatives ramenées à un même nombre d'habitants, pour que les pays soient comparés suivant des critères plus homogènes.
4. Standardiser les données.

5. Donner la matrice des corrélations, puis les valeurs propres et les vecteurs propres.
6. Faire une représentation graphiques des valeurs propres.
7. Utiliser les règles du cours pour décider du nombre d'axes à retenir.
8. Déterminer les composantes principales et les facteurs principaux.
9. Tracer le nuage des individus selon les facteurs principaux. Interpréter.
10. Calculer les corrélations entre les variables, puis tracer le cercle des corrélations. Interpréter.
11. Faire une représentation simultanées des variables et des individus.
12. Calculer les aides à l'interprétation : contributions des variables, contributions des individus, coska ou  $\cos^2$ , ...
13. Refaire l'étude de ces données en utilisant PCA du package FactoMineR.
14. Faire un rapport de conclusion sur ces données.

### 3.5 Devoir

**Exercice 23.** On considère le tableau de données irréalistes, qui a été donné en exemple lors de la séance du cours, suivant :

| Individus | 1    | 2     | 3    | 4    | 5     |
|-----------|------|-------|------|------|-------|
| $Z_1$     | 1.00 | 2.00  | 3.00 | 4.00 | 9.00  |
| $Z_2$     | 5.00 | 10.00 | 8.00 | 8.00 | 12.00 |

1. Retrouver manuellement, avec R, tous les résultats de l'ACP, trouvés dans le cours, sur ce jeu de données.
2. Interpréter.
3. ~~Utiliser les commandes :~~
  - (a) ~~Standardiser les données avec la fonction scale.~~
  - (b) ~~Calculer avec la fonction gsvd les composantes principales de l'ACP et afficher avec la fonction head les résultats pour les premiers états.~~
  - (c) Comparer avec les résultats trouvés avec les fonctions princomp et prcomp.
  - (d) Décrire et utiliser les fonctions PCA du package FactoMineR, et comparer avec les résultats trouvés.

**Exercice 24.** Les données du tableaux ci-dessous concernent des stations de ski en Savoie. On dispose, pour 32 stations, des variables suivantes (données 1998).

- prixforf : prix du forfait 1 semaine (Euros)
- altmin : altitude minimum de la station (m)
- altmax : altitude maximum de la station (m)
- pistes : nombre de pistes de ski alpin
- kmfond : nombre de kilomètres de pistes de ski de fond remontee : nombre de remontées mécaniques

| Nom          | prixforf | altnmin | altnmax | pistes | kmfond | remontee |
|--------------|----------|---------|---------|--------|--------|----------|
| LesAillons   | 76       | 900     | 2000    | 45     | 50     | 22       |
| LesArcs      | 160      | 800     | 3226    | 117    | 30     | 69       |
| Arèches      | 85       | 750     | 2300    | 30     | 47     | 15       |
| Aussois      | 71       | 500     | 2750    | 21     | 10     | 11       |
| Bessans      | 54       | 1710    | 2200    | 4      | 80     | 4        |
| Bonneval     | 79       | 1850    | 3000    | 16     | 0      | 10       |
| LeCorbier    | 88       | 1550    | 1850    | 36     | 25     | 24       |
| Courchevel   | 140      | 1100    | 2707    | 100    | 66     | 67       |
| Crest-Voland | 82       | 1230    | 1650    | 26     | 7      | 17       |
| Flumet       | 42       | 1000    | 1600    | 20     | 12     | 40       |
| LesKarellis  | 84       | 1600    | 2550    | 28     | 30     | 17       |
| LesMenuires  | 140      | 1400    | 3200    | 61     | 28     | 45       |
| Méribel      | 140      | 1100    | 2950    | 74     | 33     | 60       |
| LaNorma      | 93       | 1350    | 2750    | 25     | 6      | 18       |
| Bellecombe   | 86       | 1150    | 2070    | 32     | 8      | 18       |
| LaPlagne     | 159      | 1250    | 3250    | 123    | 73     | 110      |
| Pralognan    | 87       | 1410    | 2355    | 22     | 25     | 14       |
| LaRosière    | 122      | 1150    | 2640    | 32     | 12     | 19       |
| LesSaisies   | 95       | 1150    | 1941    | 26     | 80     | 24       |
| StFrancois   | 96       | 1450    | 2550    | 28     | 50     | 17       |
| StMartin     | 140      | 1450    | 3200    | 61     | 28     | 48       |
| StSorlin     | 81       | 1500    | 2600    | 26     | 16     | 13       |
| LaTania      | 140      | 900     | 1900    | 100    | 8      | 67       |
| Tignes       | 160      | 1550    | 3450    | 129    | 52     | 96       |
| LaToussuire  | 98       | 1800    | 2400    | 27     | 12     | 19       |
| ValCenis     | 74       | 1400    | 2800    | 38     | 6      | 22       |
| Valfréjus    | 78       | 1550    | 2737    | 20     | 2      | 12       |
| Valdlsère    | 160      | 1850    | 2560    | 70     | 24     | 51       |
| Valloire     | 107      | 1430    | 2600    | 80     | 20     | 34       |
| Valmeinier   | 107      | 1450    | 2600    | 75     | 15     | 33       |
| Valmorel     | 123      | 1250    | 2550    | 56     | 20     | 36       |
| ValThorens   | 103      | 1800    | 3200    | 54     | 5      | 30       |

TABLE 3.1 – Les stations de ski en Savoie