**Compréhension des concepts de ACP et AFC et implémentation en cas réel :**

|  |
| --- |
| ACP – Analyse en composantes principales  AFC – Analyse Factorielle de correspondances  Analyse descriptive de « The World Happiness Report » |

**Table des matières**

1. [**Introduction : 4**](#_Toc123993494)

[**1-Les méthodes d’analyse de données : 4**](#_Toc123993495)

[**2-Les méthodes utilisées dans notre projet : 5**](#_Toc123993496)

[**Echantillon de données choisi : 6**](#_Toc123993497)

1. [**Besoins fonctionnels en outils et logiciels technologiques : 7**](#_Toc123993498)
2. [**ACP : Implémentation de l’Analyse en Composantes Principales 8**](#_Toc123993499)
3. [**AFC : Implémentation de l’Analyse Factorielle de Correspondances 19**](#_Toc123993500)
4. [**Relation entre ACP et AFC : 23**](#_Toc123993501)
5. [**Conclusion : 24**](#_Toc123993502)
6. [**Annexe 25**](#_Toc123993503)

[**1/ Code Python 25**](#_Toc123993504)

[**2/ Code R 46**](#_Toc123993505)

Table d’illustrations

[Figure 1 jeu de données en excel 5](#_Toc123991852)

[Figure 2 Chargement de données sous python 7](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991853)

[Figure 3 Chargement de données sous R 7](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991854)

[Figure 4 Choix des variables pour leur intégration à l'analyse sous Python 8](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991855)

[Figure 5 Choix des variables pour leur intégration à l'analyse sous R 9](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991856)

[Figure 6 Standardisation des données sous Python 10](#_Toc123991857)

[Figure 7 Standardisation des données sous R 10](#_Toc123991858)

[Figure 8 PCA sous Python 11](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991859)

[Figure 9 PCA sous R 11](#_Toc123991860)

[Figure 10 Projection des individus sur les axes factorielles 12](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991861)

[Figure 11 Détermination des coordonnées des variables sur les axes factoriels 12](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991862)

[Figure 12 Cercle de corrélation 13](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991863)

[Figure 13 Heatmap pour visualiser la corrélation entre variables 13](#_Toc123991864)

[Figure 14 Résultat de l'ACP 14](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991865)

[Figure 15 Variance expliquée cumulative sous python sous forme de line plot 15](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991866)

[Figure 16 Variance expliquée en pourcentage sous R 15](file:///C:\Users\WhiteLuce%20SIS\OneDrive\if5\analyse%20de%20données\République%20Tunisienne.docx#_Toc123991867)

[Figure 17 Détails sur les valeurs propres , les variances cumulatives et leurs pourcentages 16](#_Toc123991868)

[Figure 18 Pourcentage des variances cumulative 16](#_Toc123991869)

[Figure 19 Application de L'AFC sur nos données 17](#_Toc123991870)

[Figure 20 projection des variables sur les axes factoriels 18](#_Toc123991871)

[Figure 21 Distribution des individus sur les 2 axes factoriels 18](#_Toc123991872)

[Figure 22 Résultat e l'AFC 19](#_Toc123991873)

[Figure 23 Description statistique des résultats 20](#_Toc123991874)

[Figure 24 Pourcentage de la variance expliquée cumulative 21](#_Toc123991875)

[Figure 25 ACP & AFC 22](#_Toc123991876)

# Introduction :

Dana le cadre de projet d’analyse de données, nous choisissons un échantillon de données présenté sous un fichier csv qu’on a téléchargé du kaggle pour exercer nos acquis et tester les méthodes de l’ACP « Analyse en composantes principales » et AFC « l’analyse factorielle de correspondances » dont le but de comprendre la relation entre variables et identifier la contribution des variables dans l’étude de phénomène.

## Les méthodes d’analyse de données :

En effet ce projet déroule dans le cadre de la statistique descriptive de données ainsi que l’analyse descriptive de données. L'analyse descriptive est le terme donné à l'analyse des données permettant de décrire et de résumer des données historiques de manière significative afin que, par exemple, des insights en ressortent. L'analyse descriptive permet de répondre à la question « Que s’est-il passé ? ».

C’est ainsi qu’ayant plusieurs méthodes d’analyse descriptive de données qui se distingue selon deux critères :

1. Selon l’objectif de la méthode :

* Méthodes explicatives : qui consistent à expliquer une variable au moyen d’une ou plusieurs variable. On parle de variables à expliquer et les variables explicatives.

On trouve à titre d’exemple :

* Problème de régression et de corrélation
* Analyse de variance
* Analyse discriminatoire
* Régression logistique
* Corrélation canonique
* Méthodes descriptives : qui consistent à résumer, visualiser et synthétiser les informations.

On trouve à titre d’exemple :

* Analyse Factorielle des Correspondances
* Analyse en Composantes Principales
* Classification Ascendante Hiérarchique

1. Selon le type de mesure :

On a les données qualitatives qui se traduisent sous deux types de mesures :

* Mesure nominale : Des chiffres sans aucune relation d’ordre ni de distance ni d’origine.
* Méthode utilisée : Analyse Factorielle des Correspondances
* Mesure ordinale : Les chiffres qui identifient une relation d’ordre entre les propriétés d’objets.
* Méthode utilisée : Analyse Factorielle des Correspondances

On a les données quantitatives qui se traduisent sous deux types de mesures :

* Mesure discrète : des informations qui ne peuvent prendre que certaines valeurs.
* Mesure continues : renvoie au contraire à un ensemble dense : il existe toujours quelque chose entre deux éléments
* Pour ces deux types de mesures on utilise : Analyse en Composantes Principales

## Les méthodes utilisées dans notre projet :

Comme les données qu’on a extraites sont des données quantitatives et qualitatives évidement on utilisera l’ACP et l’AFC.

Définissons ces deux méthodes pour mieux savoir dans quel but ce projet nous dirige :

* Méthode de l’Analyse en Composantes principales (ACP) :

Il s’agit d’une méthode descriptive et un outil statistique qui a pour objectif l’analyse des tableaux de données qui ne comportent pas de structure préalable (aucune distinction entre variable et individu).

L'analyse en composantes principales (ACP ou PCA en anglais pour principal component analysis) est une méthode de la famille de l'analyse des données et plus généralement de la statistique multivariée, qui consiste à transformer des variables liées entre elles (dites « corrélées » en statistique) en nouvelles variables décorrélées les unes des autres. Ces nouvelles variables sont nommées « composantes principales » ou axes principaux. Elle permet au statisticien de résumer l'information en réduisant le nombre de variables.

* Le but principal : résumer l’information contenue dans un tableau composé d’un nombre élevé de lignes « n » et de colonnes « p ».
* Remplacer les p variables avec q nouvelles variables avec q << p vu que à partir d’un certain p >3 c’est rendu difficile de présenter le nuage de points.
* Méthode de l’Analyse Factorielle des Correspondances (AFC) :

Il s’agit d’une méthode exploratoire d’analyse multidimensionnelle des tableaux de contingence.

L'analyse factorielle des correspondances (AFC) est une méthode statistique d'analyse des données qui permet d'analyser et de hiérarchiser les informations contenues dans un tableau rectangulaire de données et qui est aujourd'hui particulièrement utilisée pour étudier le lien entre deux variables qualitatives (ou catégorielles).

Elle se rattache à la famille des analyses factorielles qui regroupe différentes méthodes d'analyses de grands tableaux rectangulaires de données, visant toutes à identifier et à hiérarchiser des facteurs corrélés aux données placées en colonnes.

* Le but principal : permet d’étudier la proximité qui caractérise les lignes (i) et les colonnes (j) sur la base des différences de poids affectés à (i) et (j).
* Permet aussi de répondre aux questions suivantes :
* Y-a-t-il un lien entre les deux caractères étudiés ?
* Si oui , comment se comporte un caractère par rapport à l’autre ?

Dans les sections suivantes nous décrirons de façon détaillée les différentes étapes qu’on a effectué pour aboutir à des résultats concrets et interprétables accompagnés de visualisations qui décriront au mieux le phénomène qu’on essaie de prouver.

# Echantillon de données choisi :

Au cours de notre projet on a choisi un échantillon de données contenant 149 lignes et 20 colonnes. Cet échantillon est une base de données intitulée « World Happiness Report » qui t une enquête historique sur l'état du bonheur dans le monde.

Le rapport continue de gagner une reconnaissance mondiale, les gouvernements, les organisations et la société civile utilisant de plus en plus les indicateurs de bonheur pour éclairer leurs décisions politiques. D'éminents experts dans divers domaines - économie, psychologie, analyse d'enquêtes, statistiques nationales, santé, politique publique et autres - décrivent comment les mesures du bien-être peuvent être utilisées efficacement pour évaluer les progrès des nations. Les rapports passent en revue l'état du bonheur dans le monde d'aujourd'hui et montrent comment la nouvelle science du bonheur explique les variations personnelles et nationales du bonheur.

* A propos des colonnes de cet échantillon :

Les colonnes qui suivent le score de bonheur estiment dans quelle mesure chacun des six facteurs - production économique, soutien social, espérance de vie, liberté, absence de corruption et générosité - contribue à rendre les évaluations de la vie plus élevées dans chaque pays qu'elles ne le sont dans Dystopia, un pays hypothétique dont les valeurs sont égales aux moyennes nationales les plus basses du monde pour chacun des six facteurs. Ils n'ont aucun impact sur le score total rapporté pour chaque pays, mais ils expliquent pourquoi certains pays sont mieux classés que d'autres.

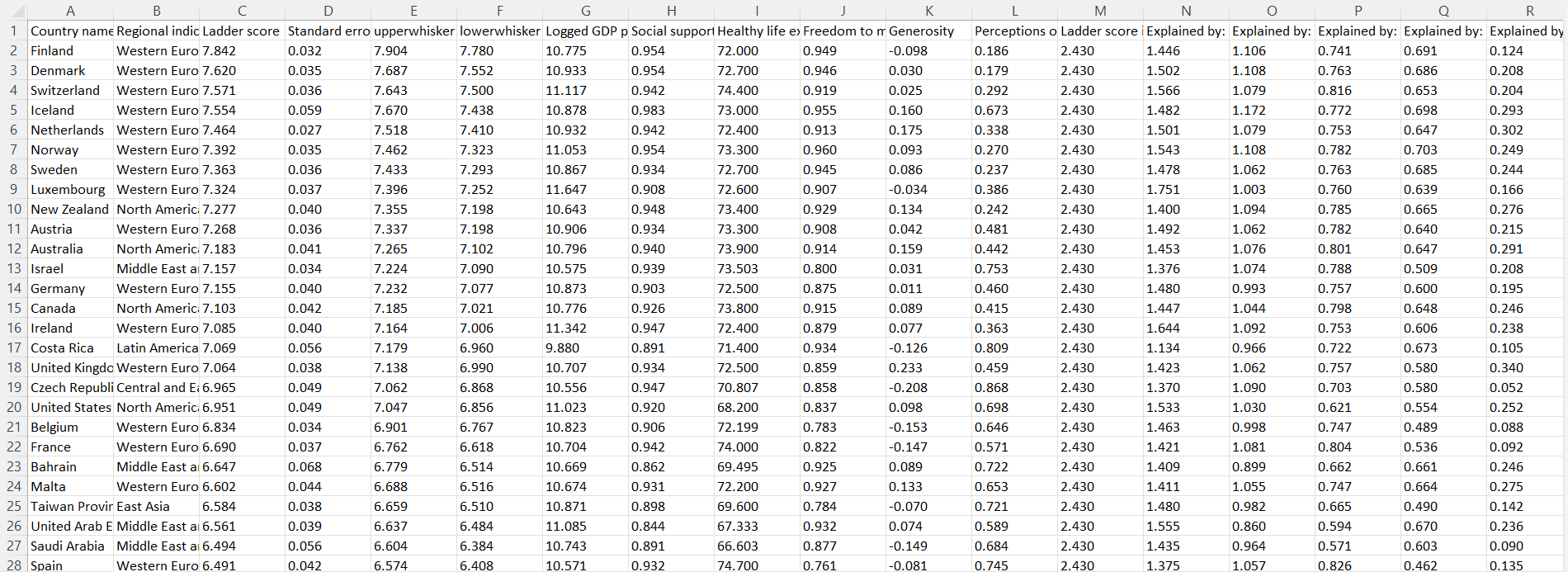


Figure 1 jeu de données en excel

# Besoins fonctionnels en outils et logiciels technologiques :

L’implémentation de ses méthodes d’analyse descriptives seront à la fois implémentées sous python et R :

1. Python :

Python est un langage de programmation interprété, multiparadigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet.



1. R :

R est un langage de programmation et un logiciel libre destiné aux statistiques et à la science des données.

Il permet de réaliser d'importants calculs statistiques sur une grande quantité de données et de générer des représentations graphiques rigoureuses.



Dans les sections suivantes on introduira notre travail en implémentant les deux méthodes descriptives qui sont l’ACP et l’AFC avec ces deux logiciels.

# ACP : Implémentation de l’Analyse en Composantes Principales

Les étapes de cette analyse sont définies comme suit :

1. Extraire les données et les charger
2. Choisir les variables à intégrer dans notre analyse
3. Standardiser nos données et applique l’ACP
4. Interprétation des résultats et des graphiques menés par l’application de cette méthode

* Chargement de données :
* Une image contenant texte

  Description générée automatiquementSous Python :

Figure Chargement de données sous python

* Une image contenant table

  Description générée automatiquementSous R :

Figure Chargement de données sous R

* Choix des variables à impliquer dans l’analyse :

Pour appliquer l’ACP on doit l’effectuer sur des mesures quantitatives pour cela nous avons omis quelques variables qui sont de nature qualitatives et qui ne seront pas utiles pour notre analyse.

* Sous python :

Une image contenant table

Description générée automatiquement

Figure Choix des variables pour leur intégration à l'analyse sous Python

* Une image contenant table

  Description générée automatiquementSous R :

Figure Choix des variables pour leur intégration à l'analyse sous R

* 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 qu’elles aient finalement :

* Un écart type égal à un
* Une moyenne égale à zéro.
* Sous Python : ppour la standardisation des données sous python on doit importer la bibliothèque sklearn.preprocessing auprés duquel on utilisera la méthode StandardScaler

**from** sklearn.preprocessing **import** StandardScaler

Une image contenant texte

Description générée automatiquement

Figure 6 Standardisation des données sous Python

* Sous R :

Une image contenant table

Description générée automatiquement

Figure 7 Standardisation des données sous R

* Application de l’ACP :
* Sous Python :
  + pour l’application de l’ACP on doit tout d’abord importer la bibliothèque sklearn.decomposition afin d’utiliser la méthode PCA
  + Une image contenant texte

    Description générée automatiquementaussi on doit l’utiliser sur nos données standardisées.

Figure PCA sous Python

* pca.n\_components\_ : nous retourne le nombre de composantes principales.

En théorique, le nombre de composantes principales est **inférieur ou égal au nombre de variables d'origine**.

* pca.explained\_variance \_ : L'ACP montre la variance qui peut être attribuée à chacune des composantes principales. C'est un tableau de valeurs où chaque valeur est égale à la variance de chaque composante principale et la longueur du tableau est égale au nombre de composantes défini par n\_composantes .
* Sous R :
  + Sous R on doit importer la bibliothèque FactoMiner pour pouvoir utiliser PCA .

Une image contenant table

Description générée automatiquement

Figure 9 PCA sous R

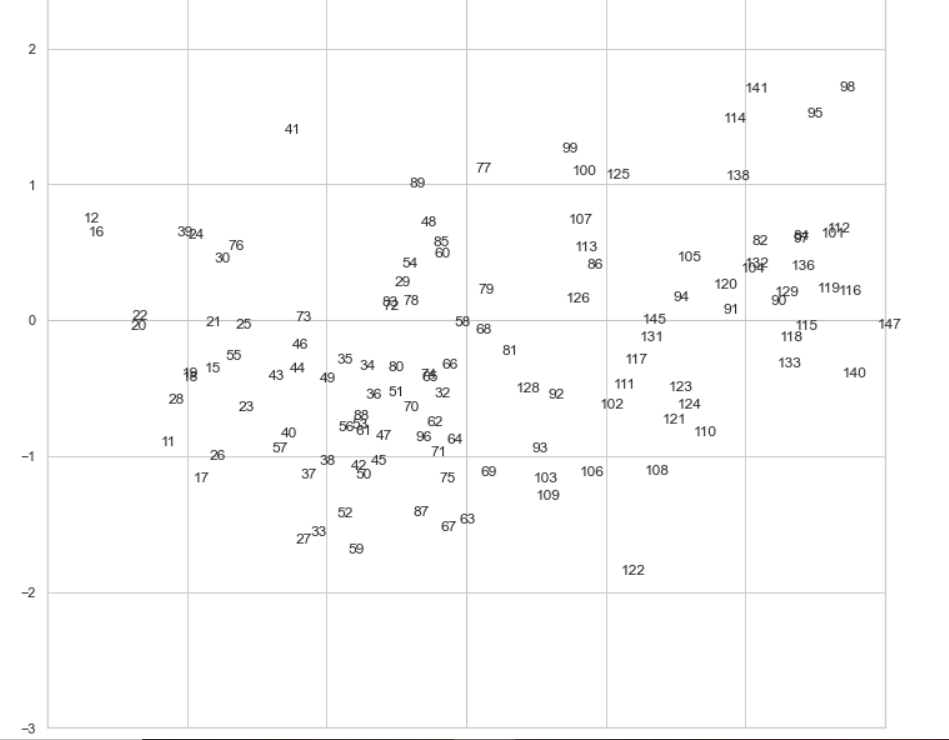
* Visualisation et interprétation :
* Sous Python : on présente tout d’abord la distribution des individus sur les axes factoriels :

Figure Projection des individus sur les axes factorielles

Maintenant traçons le cercle de corrélation : Dans un cercle de corrélation, toutes les variables initiales incluses dans un ACP sont représentées par rapport à deux des composantes principales, qui sont représentées par les axes x et y.

Une image contenant texte

Description générée automatiquement1/ On commence par la détermination de coordonnées des variables sur ces axes factorielles :

Figure Détermination des coordonnées des variables sur les axes factoriels

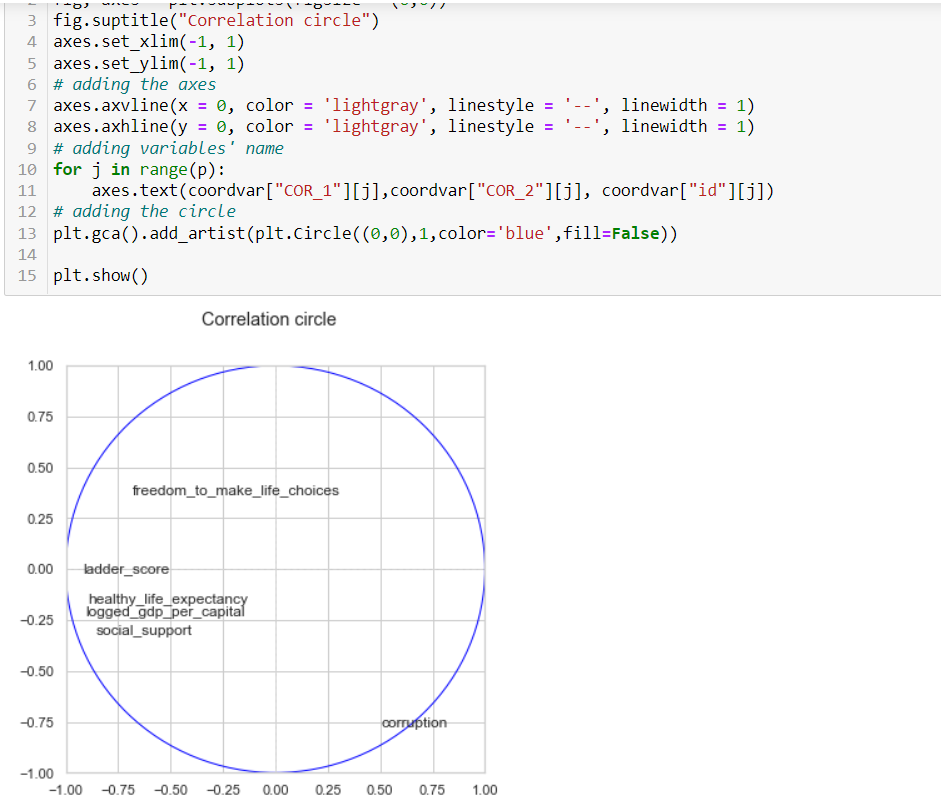
2/ Traçons le cercle de corrélation :

Figure Cercle de corrélation

Interprétation de cercle de corrélation : Il montre les relations entre toutes les variables. Elle peut être interprétée comme suit : Les variables positivement corrélées sont regroupées. Les variables corrélées négativement sont positionnées de part et d'autre de l'origine du graphique (quadrants opposés).

D’où on peut conclure que les variables healthy\_life\_expectancy et logged\_gdp\_percapit et social\_support sont corrélés positivement de même avec le ladder\_score par contre ils sont négativement corrélés avec la corruption ce qui est bien réaliste et actuel.

On peut justifier ceci par un heatmap correlation : Les cartes thermiques de corrélation sont un type de graphique qui permet de visualiser la force des relations entre les variables numériques. Les graphiques de corrélation sont utilisés pour comprendre quelles variables sont liées les unes aux autres et la force de cette relation.

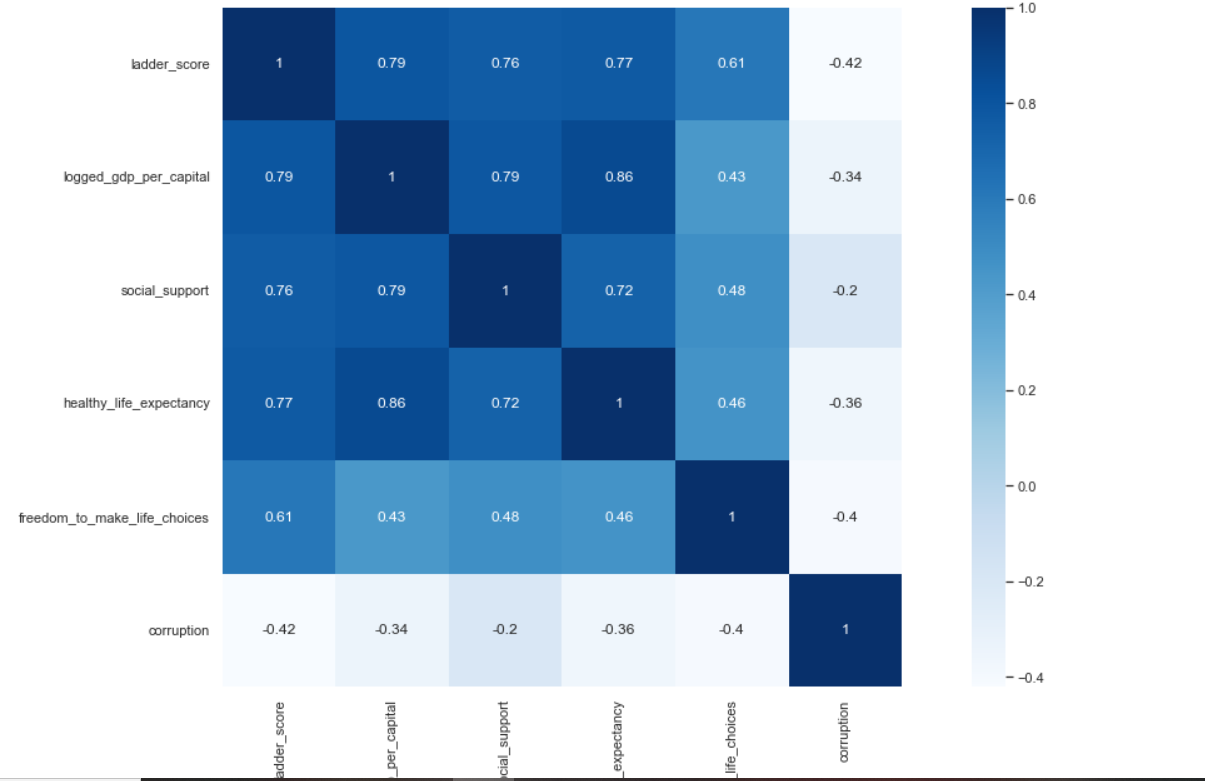


Figure 13 Heatmap pour visualiser la corrélation entre variables

* Sous R :

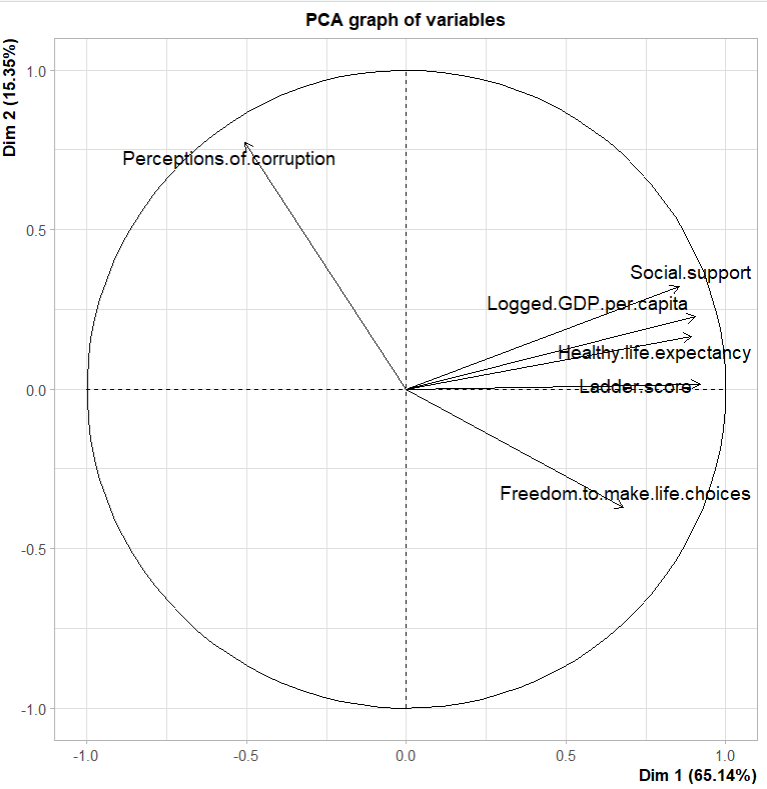
1/ On commence par visualiser le résultat d’application de PCA sur nos données

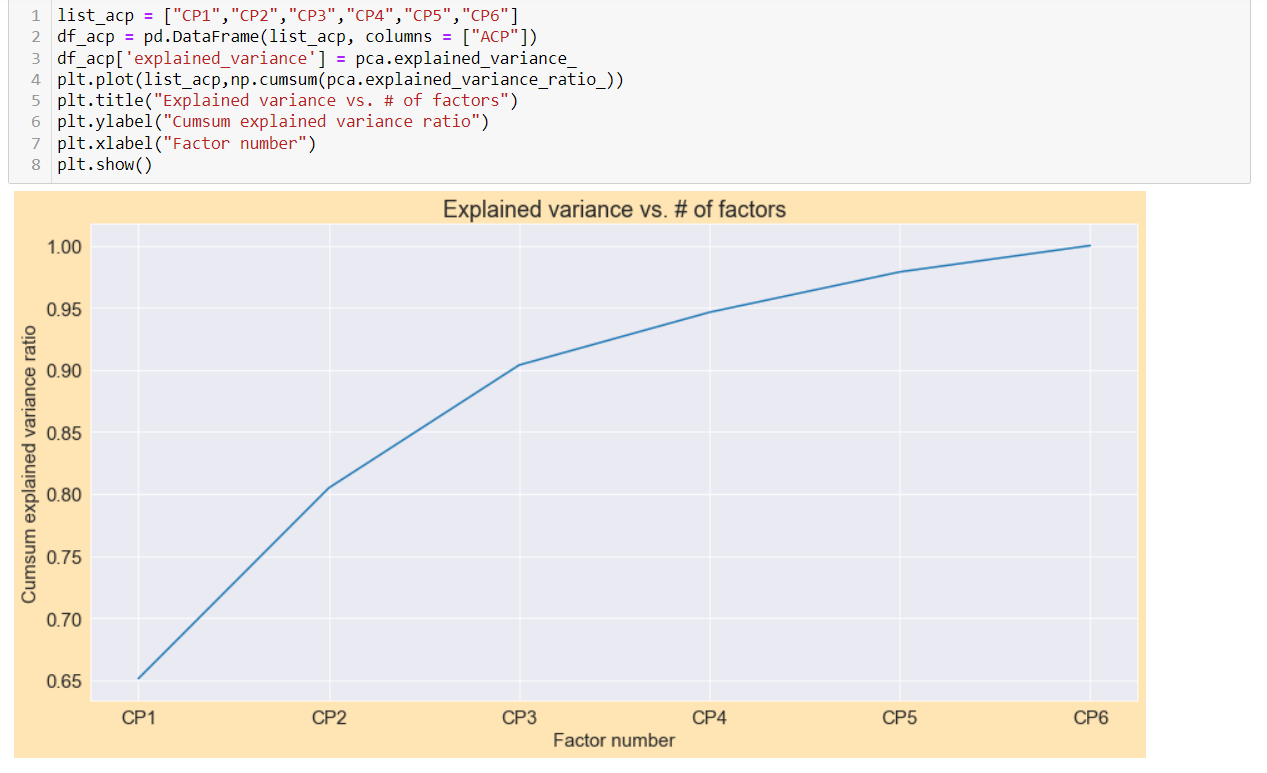
Figure Résultat de l'ACP

D’après cette visualisation qui présente le cercle de corrélation résultat de l’application de l’ACP on peut remarquer qu’on a abouti aux mêmes résultats par python seulement que R est plus rigoureux en accompagnant les axes par un pourcentage qui indique la quantité d’information qu’inclut cet axe.

On remarque que la 1 ère dimension présente 65,14% de l’information alors que le deuxième axe présente 15.35% de l’information.

* Présentons maintenant sous python et R les visualisations associées à la variation expliquée :

En statistique, la variation expliquée mesure la proportion dans laquelle un modèle mathématique rend compte de la variation (dispersion) d'un ensemble de données donné. Souvent, la variation est quantifiée en tant que variance ; ensuite, le terme plus spécifique de variance expliquée peut être utilisé.

* Sous Python :

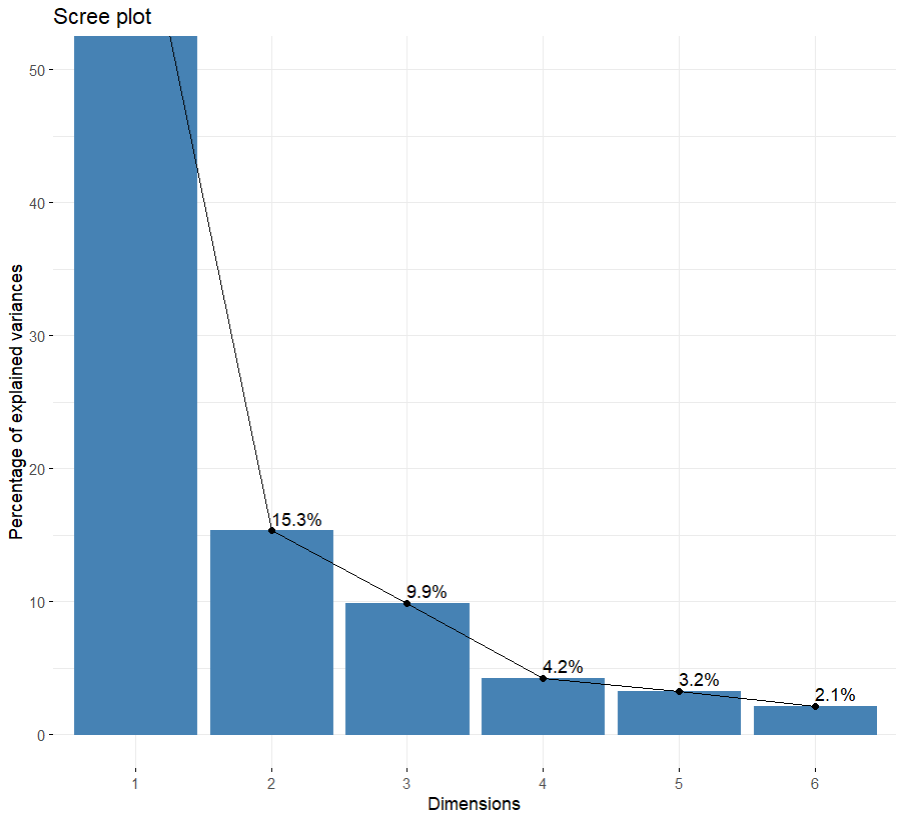


Figure Variance expliquée cumulative sous python sous forme de line plot

* Sous R :

Figure Variance expliquée en pourcentage sous R

* Sous R :

Le graphique ci-dessus montre la quantité d’information pour chaque composante , ainsi on pourra conclure que la 1 ère composante présente plus de 50% de l’information par contre la 6 ème présente uniquement 2.1%.

Pour plus de précision nous pouvons voir de façon détaillée les valeurs propres ainsi que les pourcentages de variances aussi les variances cumulatives qui ont abouti à ces résultats de visualisation.

Une image contenant texte

Description générée automatiquement

Figure 17 Détails sur les valeurs propres , les variances cumulatives et leurs pourcentages

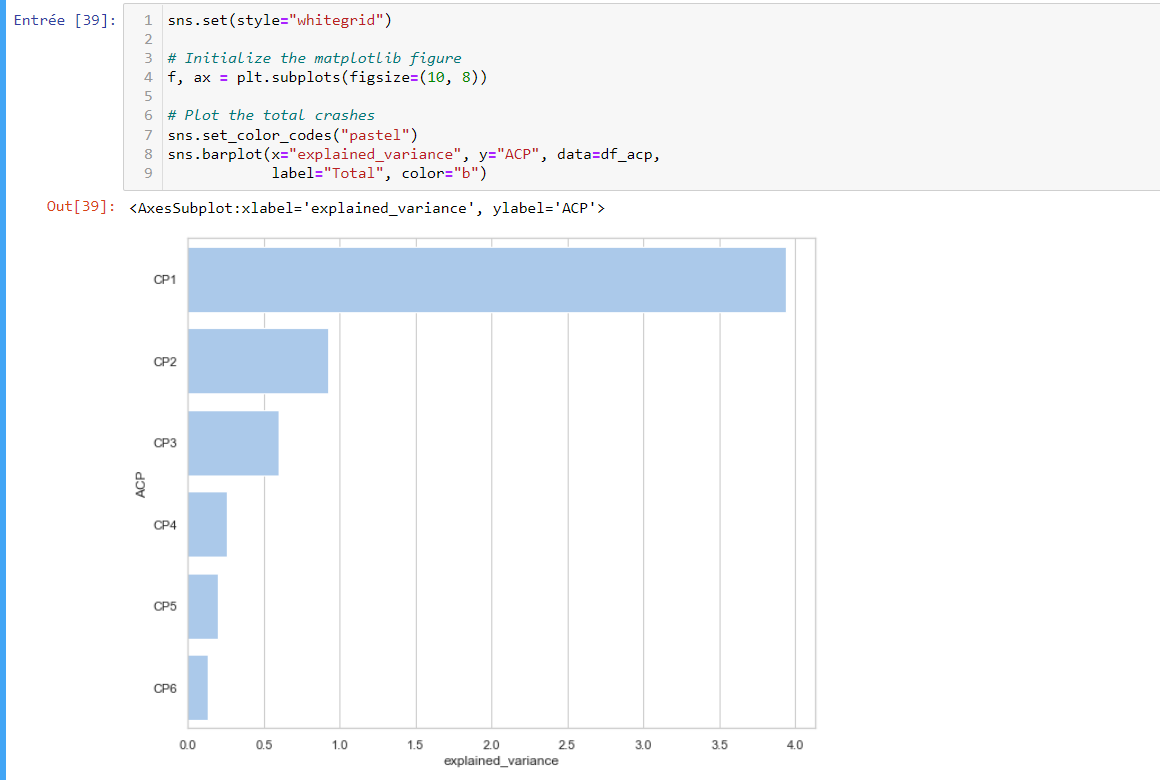


Figure 18 Pourcentage des variances cumulative

# AFC : Implémentation de l’Analyse Factorielle de Correspondances

Les étapes de l’analyse sont définies comme suit :

1. Extraction de données
2. Choix des variables à intégrer dans l’analyse
3. Application de l’AFC
4. Visualisation et interprétations de résultats

L’étape d’extraction et chargement de données est déjà effectuée auparavant ainsi qu’on travaillera sur les mêmes données et sur les mêmes variables qu’on a intégré lors de l’ACP.

* Application de l’AFC
* Sous Python :

Une image contenant texte

Description générée automatiquement

Figure 19 Application de L'AFC sur nos données

Afin de pouvoir implémenter l’AFC on doit tout d’abord importer de la bibliothèque sklear.decomposition la méthode FactorAnalysis. Cette méthode réalise une estimation du maximum de vraisemblance de la matrice dite de chargement, la transformation des variables latentes en combinaison linéaire de variables observées qui sont choisies de manière qu’elles ont aucune corrélation entre elles.

* Visualisons maintenant la projection des variables sur ces 2 axes factoriels :

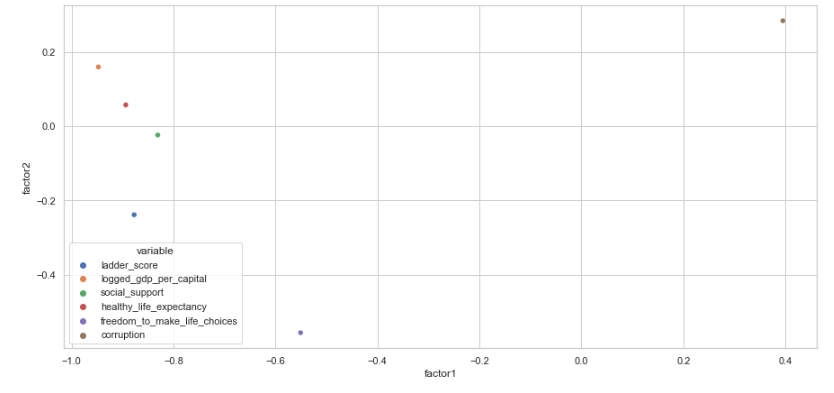


Figure 20 projection des variables sur les axes factoriels

Le graphique ci-dessus est un **scatter plot** [diagramme de dispersion pour déterminer si deux variables ont ou non une relation ou une corrélation. Le tracé d'un diagramme de dispersion avec vos points de données peut vous aider à déterminer s'il existe une relation potentielle entre eux.] nous a permis de voir la projection des variables sur les axes factoriels.

* Voyons maintenant la dispersion des individus sur ces deux axes factoriels :

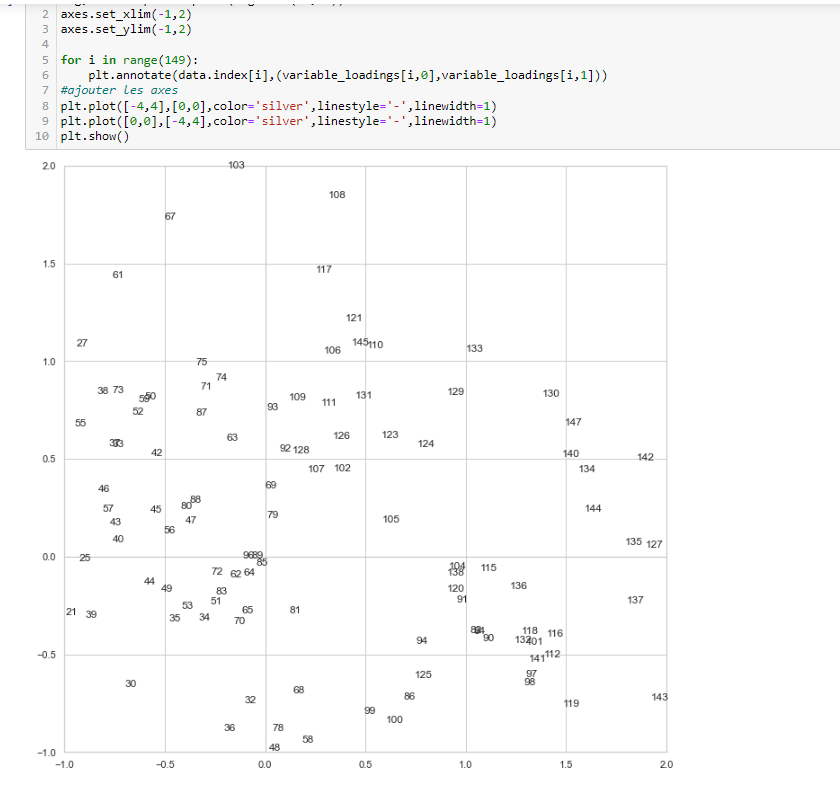


Figure 21 Distribution des individus sur les 2 axes factoriels

* Sous R :



Afin d’exécuter l’AFC on doit importer la bibliothèque FactoMiner afin d’utiliser la méthode CA qui effectuera l’AFC.

On peut voir ci-dessous le résultat de l’AFC on a deux axes factoriels dont l’un présente 23.3% de l’information alors que l’autre présente 54.52%.

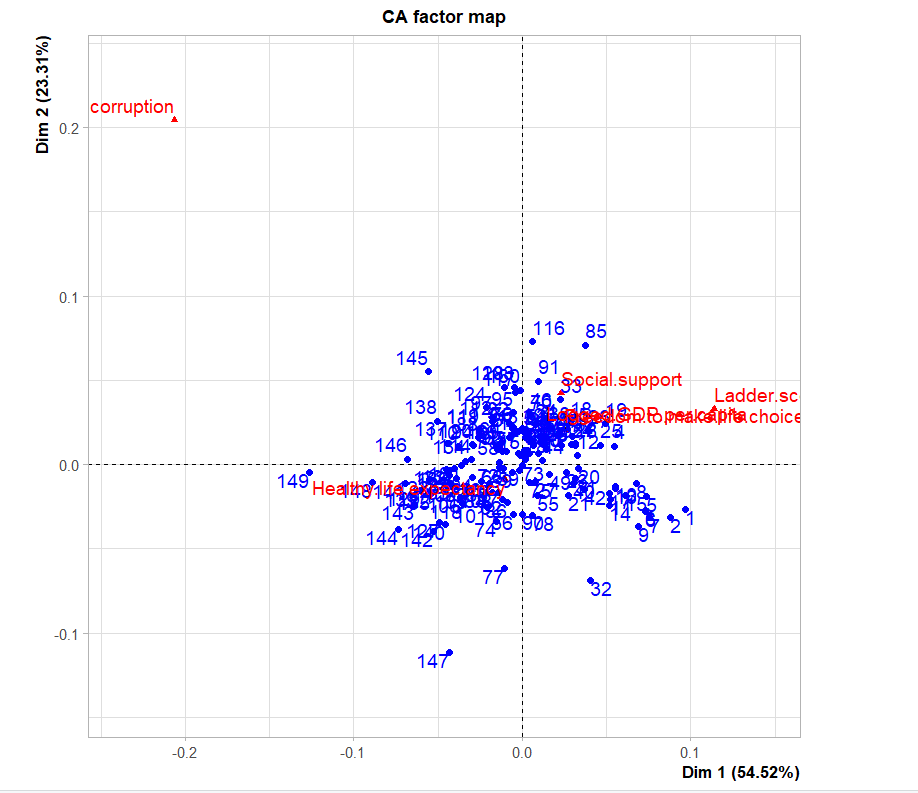


Figure 22 Résultat e l'AFC

Maintenant on s’intéresse à voir de façon détaillée notre résultat et les mesures de la variances ainsi que leurs pourcentages sur les différentes dimensions.

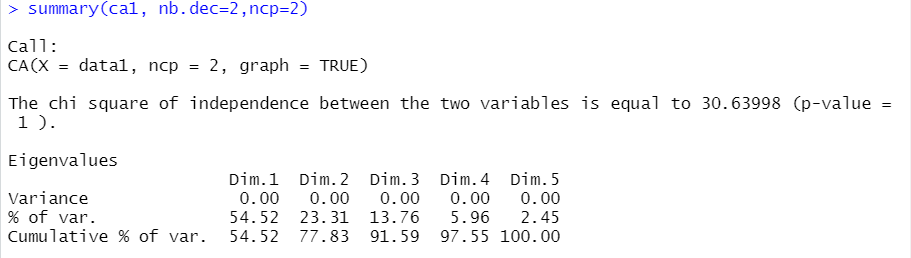


Figure 23 Description statistique des résultats

Le test du chi-deux (également test du chi-deux ou test du χ2) est un test d'hypothèse statistique utilisé dans l'analyse des tableaux de contingence lorsque la taille des échantillons est importante. En termes plus simples, ce test est principalement utilisé pour examiner si deux variables catégorielles (deux dimensions du tableau de contingence) sont indépendantes et influencent la statistique du test.

Ici on a le test egal à 30.63998 et on tolère p-value = 1 ainsi si les variables sont indépendantes alors ce test suit une loi de degré de liberté égal : r=(149-1)(6-1)=740.

D’où les variables sont vraiment indépendantes car le test de chi-deux a une valeur inférieure que le nombre de degré de liberté.



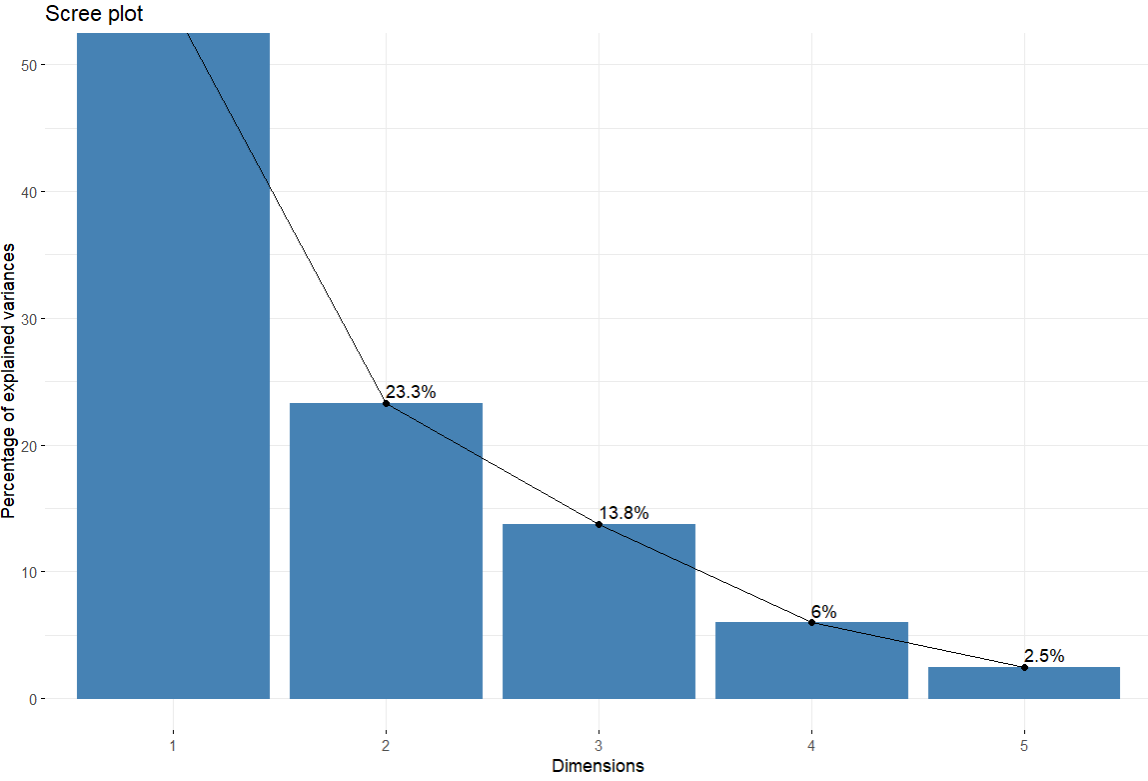


Figure 24 Pourcentage de la variance expliquée cumulative

La visualisation ci-dessus montre la variance expliquée cumulative en pourcentage par rapport aux dimensions.

# Relation entre ACP et AFC :

Dans ce projet nous avons entamé deux principales méthodes de statistique descriptives qui sont l’ACP et l’AFC.

En effet l’ACP s’execute quand les variables sont quantitatives alors que lorsque les individus sont décrits par deux variables qualitatives, on peut construire un tableau de contingence et réaliser une AFC (Analyse Factorielle des Correspondances).

En effet ACP est un type spécifique de l’AFC qu’on appelle fréquemment cette dernière par la double ACP, tel que la différence majeure entre ces deux approches peut être résumé par la figure ci-dessous :

Une image contenant texte

Description générée automatiquement

Figure 25 ACP & AFC

# Conclusion :

Les principales étapes du processus d'analyse consistent à cerner les sujets d'analyse, à déterminer la disponibilité de données appropriées, à décider des méthodes qu'il y a lieu d'utiliser pour répondre aux questions d'intérêt, à appliquer les méthodes et à évaluer, résumer et communiquer les résultats.

L'objectif de l'analyse des données est d'extraire une information statistique qui permet de cerner plus précisément le profil de la donnée. Les résultats obtenus permettent ensuite d'optimiser la stratégie de la société en question en ajustant certains points.

Mise à part de l’AFC et l’ACP on a d’autres méthodes qui nous aident pour l’analyse descriptive des données qui sont ACM (Analyse des correspondances Multiples) et HCPC (Classification Hiérarchique sur Composantes principales) permet de réaliser une classification non supervisée des individus. Cette fonction combine les facteurs principaux, la classification hiérarchique et le partitionnement pour mieux visualiser et mettre l'accent sur les similarités entre individus.

# Annexe

**1/ Code Python**: (on trouve aussi des visualisations pour mieux comprendre nos données avant de faire de L’ACP et de L’AFC)

1

**import** pandas **as** pd*# for data manipulation*

2

**import** numpy **as** np

3

**import** matplotlib.pyplot **as** plt*#to plot our data and create visualizations*

4

**import** seaborn **as** sns

5

**import** plotly.express **as** px

6

**from** plotly.subplots **import** make\_subplots

7

**from** datetime **import** datetime

8

**%**matplotlib inline

9

**from** sklearn.preprocessing **import** StandardScaler

10

**from** sklearn.decomposition **import** PCA

1

*#set the parameters that control the general style of the plot*

2

sns.set\_style('darkgrid')

3

*#runtime features pararmeters configuration*

4

plt.rcParams['font.size']**=**15

5

plt.rcParams['figure.figsize']**=**(10,7)

6

plt.rcParams['figure.facecolor']**=** '#FFE5B4'

1

*#load our dataset*

2

data**=**pd.read\_csv("world-happiness-report-2021.csv")

1

data.head()

Out[5]:

*#drop some columns that we don't need*

2

data\_columns**=**['Country name','Regional indicator','Ladder score','Logged GDP per capita','Social support','Healthy life expectancy','Freedom to make life choices','Generosity','Perceptions of corruption']

3

​

data**=**data[data\_columns].copy()

data

Out[8]:

|  | **Country name** | **Regional indicator** | **Ladder score** | **Logged GDP per capita** | **Social support** | **Healthy life expectancy** | **Freedom to make life choices** | **Generosity** | **Perceptions of corruption** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | Finland | Western Europe | 7.842 | 10.775 | 0.954 | 72.000 | 0.949 | -0.098 | 0.186 |
| **1** | Denmark | Western Europe | 7.620 | 10.933 | 0.954 | 72.700 | 0.946 | 0.030 | 0.179 |
| **2** | Switzerland | Western Europe | 7.571 | 11.117 | 0.942 | 74.400 | 0.919 | 0.025 | 0.292 |
| **3** | Iceland | Western Europe | 7.554 | 10.878 | 0.983 | 73.000 | 0.955 | 0.160 | 0.673 |
| **4** | Netherlands | Western Europe | 7.464 | 10.932 | 0.942 | 72.400 | 0.913 | 0.175 | 0.338 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **144** | Lesotho | Sub-Saharan Africa | 3.512 | 7.926 | 0.787 | 48.700 | 0.715 | -0.131 | 0.915 |
| **145** | Botswana | Sub-Saharan Africa | 3.467 | 9.782 | 0.784 | 59.269 | 0.824 | -0.246 | 0.801 |
| **146** | Rwanda | Sub-Saharan Africa | 3.415 | 7.676 | 0.552 | 61.400 | 0.897 | 0.061 | 0.167 |
| **147** | Zimbabwe | Sub-Saharan Africa | 3.145 | 7.943 | 0.750 | 56.201 | 0.677 | -0.047 | 0.821 |
| **148** | Afghanistan | South Asia | 2.523 | 7.695 | 0.463 | 52.493 | 0.382 | -0.102 | 0.924 |

149 rows × 9 columns

*#we will rename the columns to make it simpler*

2

happy\_df**=** data.rename(columns**=**{'Country name':'country\_name','Regional indicator':'regional\_indicator','Ladder score':'ladder\_score','Logged GDP per capita':'logged\_gdp\_per\_capital','Social support':'social\_support','Healthy life expectancy':'healthy\_life\_expectancy','Freedom to make life choices':'freedom\_to\_make\_life\_choices','Generosity':'generosity','Perceptions of corruption':'corruption'},inplace**=True**)

column\_headers **=** list(data.columns.values)

2

print(column\_headers)

['country\_name', 'regional\_indicator', 'ladder\_score', 'logged\_gdp\_per\_capital', 'social\_support', 'healthy\_life\_expectancy', 'freedom\_to\_make\_life\_choices', 'generosity', 'corruption']

*#check null values*

2

happy\_df**=**data.copy()

3

happy\_df.isnull().sum()

4

​

Out[11]:

country\_name 0

regional\_indicator 0

ladder\_score 0

logged\_gdp\_per\_capital 0

social\_support 0

healthy\_life\_expectancy 0

freedom\_to\_make\_life\_choices 0

generosity 0

corruption 0

dtype: int64

*#plot between happiness and gdp*

2

plt.rcParams['figure.figsize']**=**(15,7)

3

plt.title('Plot between Happiness Score and GDP')

4

sns.scatterplot(x**=**happy\_df.logged\_gdp\_per\_capital,y**=**happy\_df.ladder\_score,hue**=**happy\_df.regional\_indicator)

5

plt.legend(loc**=**'upper left', fontsize**=**'10')

6

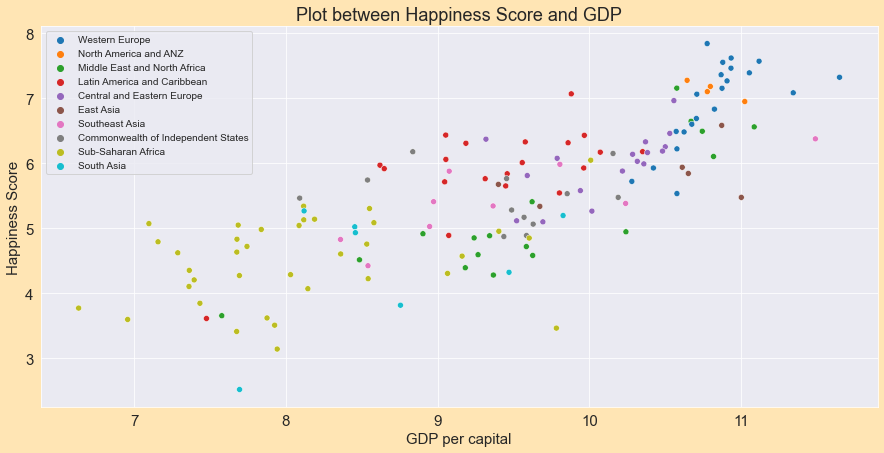
plt.xlabel('GDP per capital')

7

plt.ylabel('Happiness Score')

Out[12]:

Text(0, 0.5, 'Happiness Score')



gdp\_region**=**happy\_df.groupby('regional\_indicator')['logged\_gdp\_per\_capital'].sum()

2

gdp\_region

Out[13]:

regional\_indicator

Central and Eastern Europe 171.854

Commonwealth of Independent States 112.822

East Asia 62.206

Latin America and Caribbean 187.400

Middle East and North Africa 164.324

North America and ANZ 43.238

South Asia 60.778

Southeast Asia 84.793

Sub-Saharan Africa 290.707

Western Europe 227.277

Name: logged\_gdp\_per\_capital, dtype: float64

gdp\_region.plot.pie(autopct**=** '%1.1f%%')

2

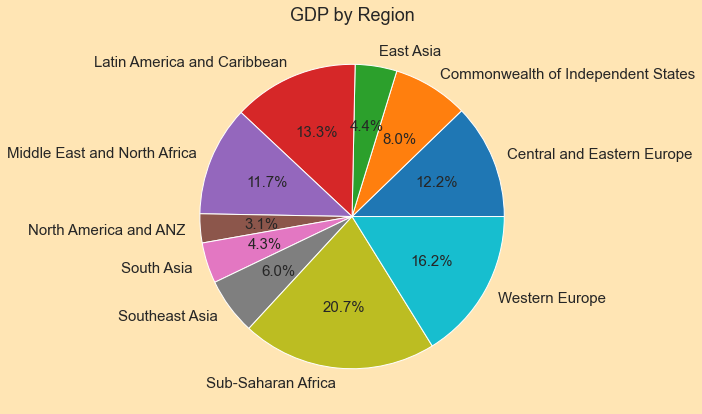
plt.title('GDP by Region')

3

plt.ylabel('')

Out[14]:

Text(0, 0.5, '')



*#Total countries*

2

total\_country**=** happy\_df.groupby('regional\_indicator')[['country\_name']].count()

3

print(total\_country)

country\_name

regional\_indicator

Central and Eastern Europe 17

Commonwealth of Independent States 12

East Asia 6

Latin America and Caribbean 20

Middle East and North Africa 17

North America and ANZ 4

South Asia 7

Southeast Asia 9

Sub-Saharan Africa 36

Western Europe 21

*#let's visualize the correlation between variables*

cor**=** happy\_df.corr(method**=**"pearson")

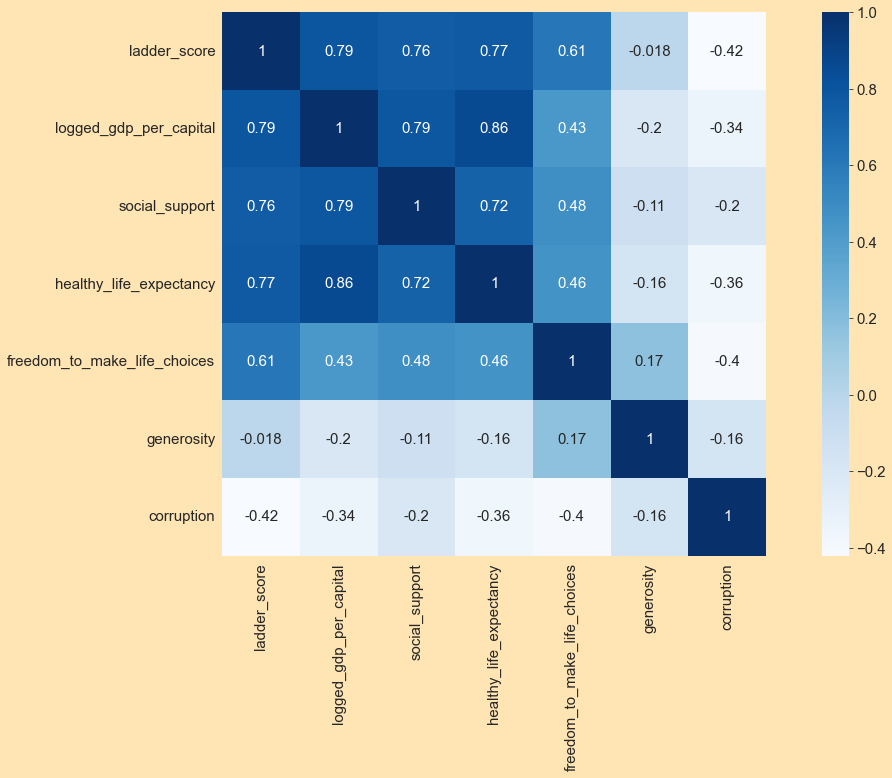
f,ax**=**plt.subplots(figsize**=**(20,10))

sns.heatmap(cor,mask**=**np.zeros\_like(cor,dtype**=**np.bool),cmap**=**"Blues",square**=True**,annot**=True**)

sns.heatmap(cor,mask=np.zeros\_like(cor,dtype=np.bool),cmap="Blues",square=True,annot=True)

Out[16]:

<AxesSubplot:>



*#corruption in regions*

corruption **=** happy\_df.groupby('regional\_indicator')[['corruption']].mean()

corruption

Out[17]:

|  | **corruption** |
| --- | --- |
| **regional\_indicator** |  |
| **Central and Eastern Europe** | 0.850529 |
| **Commonwealth of Independent States** | 0.725083 |
| **East Asia** | 0.683333 |
| **Latin America and Caribbean** | 0.792600 |
| **Middle East and North Africa** | 0.762235 |
| **North America and ANZ** | 0.449250 |
| **South Asia** | 0.797429 |
| **Southeast Asia** | 0.709111 |
| **Sub-Saharan Africa** | 0.765944 |
| **Western Europe** | 0.523095 |

plt.rcParams['figure.figsize']**=**(12,8)

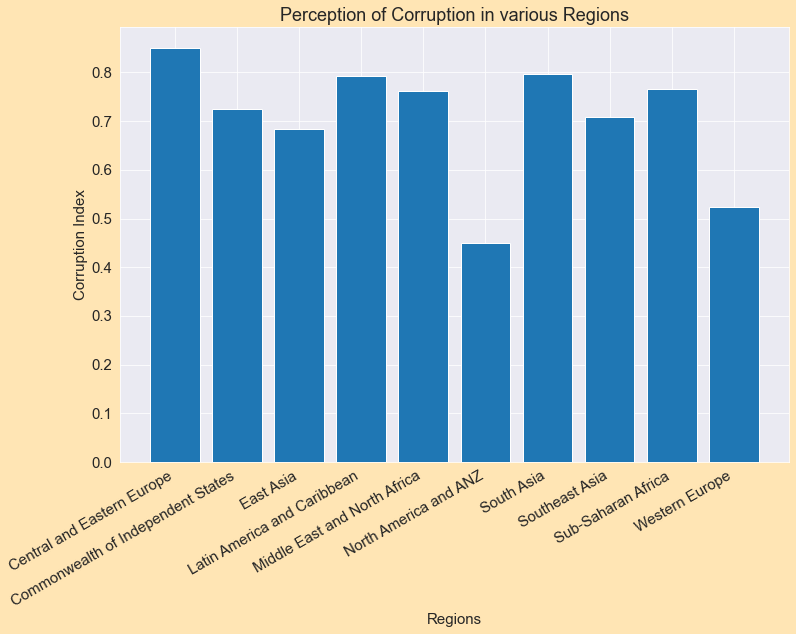
plt.title('Perception of Corruption in various Regions')

plt.xlabel('Regions',fontsize**=**15)

plt.ylabel('Corruption Index',fontsize**=**15)

plt.xticks(rotation**=**30,ha**=**'right')

plt.bar(corruption.index,corruption.corruption)



top\_10 **=** happy\_df.head(10)

bottom\_10**=**happy\_df.tail(10)

fig,axes**=** plt.subplots(1,2, figsize**=**(16,6))

plt.tight\_layout(pad**=**2)

xlabels**=**top\_10.country\_name

axes[0].set\_title('Top 10 happiest countries Life Expectancy')

axes[0].set\_xticklabels(xlabels,rotation**=**45,ha**=**'right')

axes[0].set\_xlabel('Country Name')

axes[0].set\_ylabel('Life Expectancy')

xlabels**=**bottom\_10.country\_name

axes[1].set\_title('Bottom 10 least happy countries Life Expectancy')

axes[1].set\_xticklabels(xlabels,rotation**=**45,ha**=**'right')

sns.barplot(x**=**bottom\_10.country\_name, y**=**bottom\_10.healthy\_life\_expectancy,ax**=**axes[1])

axes[1].set\_xlabel('Country Name')

axes[1].set\_ylabel('Life Expectancy')

​

C:\Users\WhiteLuce SIS\AppData\Local\Temp\ipykernel\_14392\164067708.py:5: UserWarning: FixedFormatter should only be used together with FixedLocator

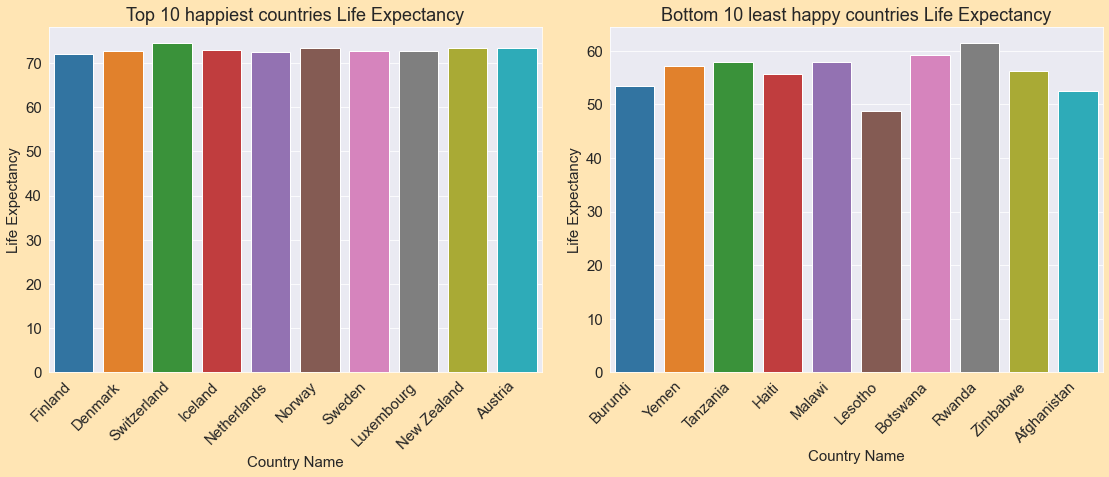
axes[0].set\_xticklabels(xlabels,rotation=45,ha='right')

C:\Users\WhiteLuce SIS\AppData\Local\Temp\ipykernel\_14392\164067708.py:12: UserWarning: FixedFormatter should only be used together with FixedLocator

axes[1].set\_xticklabels(xlabels,rotation=45,ha='right')

Out[20]:

Text(599.1068181818181, 0.5, 'Life Expectancy')



column\_headers **=** list(data.columns.values)

print(column\_headers)

['country\_name', 'regional\_indicator', 'ladder\_score', 'logged\_gdp\_per\_capital', 'social\_support', 'healthy\_life\_expectancy', 'freedom\_to\_make\_life\_choices', 'generosity', 'corruption']

*#plot between happiness and freedom to make life choices*

plt.rcParams['figure.figsize']**=**(15,9)

sns.scatterplot(x**=** happy\_df.freedom\_to\_make\_life\_choices,y **=**happy\_df.ladder\_score,hue**=**happy\_df.regional\_indicator)

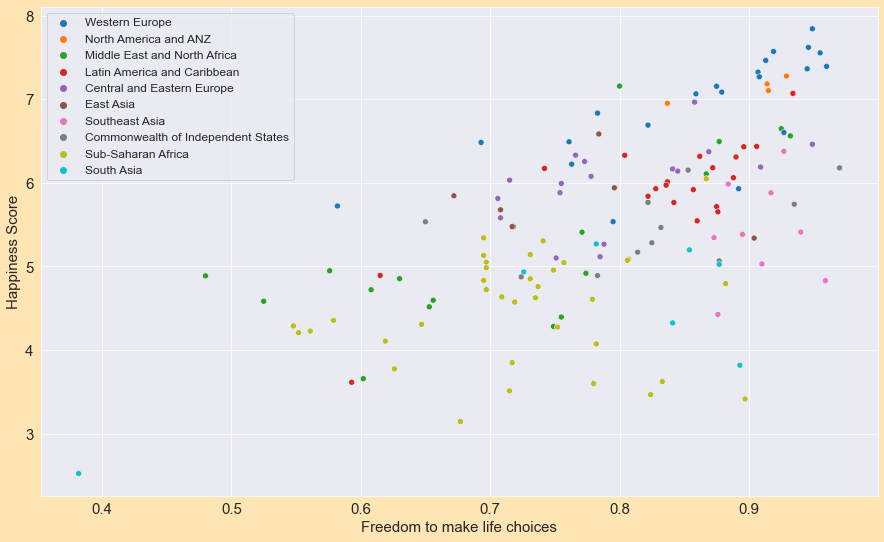
plt.legend(loc**=**"upper left", fontsize**=**'12')

plt.xlabel('Freedom to make life choices')

plt.ylabel('Happiness Score')

Out[22]:

Text(0, 0.5, 'Happiness Score')



country\_bottom**=**happy\_df.sort\_values(by**=**'corruption').tail(10)

2

country\_top**=**happy\_df.sort\_values(by**=**'corruption').head(10)

3

​

4

fig,axes**=** plt.subplots(1,2, figsize**=**(16,6))

5

plt.tight\_layout(pad**=**2)

6

xlabels**=**country\_bottom.country\_name

7

axes[0].set\_title('Most corrupted countries ')

8

axes[0].set\_xticklabels(xlabels,rotation**=**45,ha**=**'right')

9

sns.barplot(x**=**country\_bottom.country\_name, y**=**country\_bottom.corruption,ax**=**axes[0])

10

axes[0].set\_xlabel('Country Name')

11

axes[0].set\_ylabel('Corruption Index')

12

​

13

xlabels**=**country\_top.country\_name

14

axes[1].set\_title('Least corrupted countries')

15

axes[1].set\_xticklabels(xlabels,rotation**=**45,ha**=**'right')

16

sns.barplot(x**=**country\_top.country\_name, y**=**country\_top.corruption,ax**=**axes[1])

17

axes[1].set\_xlabel('Country Name')

18

axes[1].set\_ylabel('Corruption Index')

C:\Users\WhiteLuce SIS\AppData\Local\Temp\ipykernel\_14392\2744303341.py:8: UserWarning: FixedFormatter should only be used together with FixedLocator

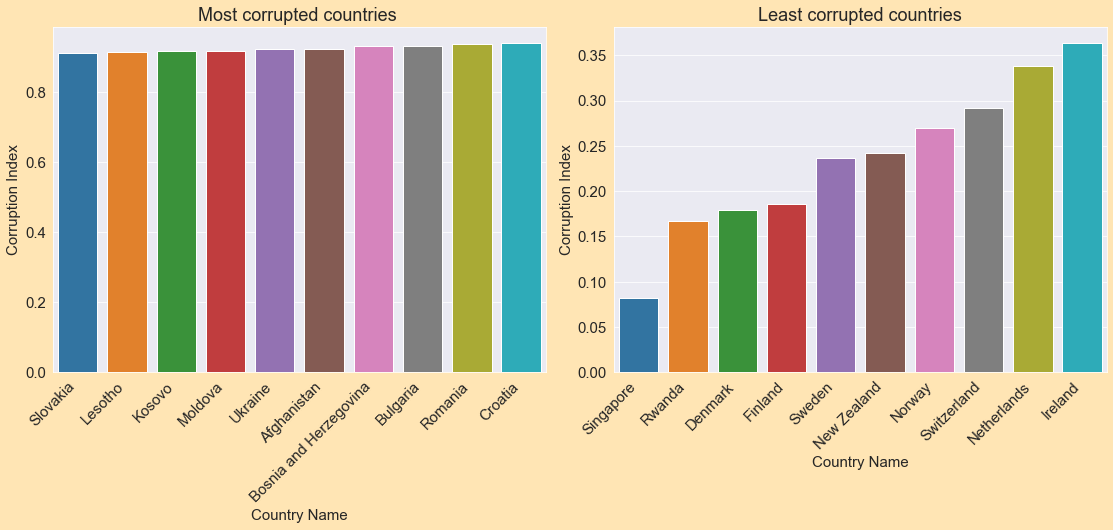
axes[0].set\_xticklabels(xlabels,rotation=45,ha='right')

C:\Users\WhiteLuce SIS\AppData\Local\Temp\ipykernel\_14392\2744303341.py:15: UserWarning: FixedFormatter should only be used together with FixedLocator

axes[1].set\_xticklabels(xlabels,rotation=45,ha='right')

Out[23]:

Text(599.1068181818181, 0.5, 'Corruption Index')



1

*#corruption vs happiness*

2

plt.rcParams['figure.figsize']**=**(15,7)

3

sns.scatterplot(x**=**happy\_df.ladder\_score,y**=**happy\_df.corruption,hue**=**happy\_df.regional\_indicator)

4

plt.legend(loc**=**'lower left',fontsize**=**'14')

5

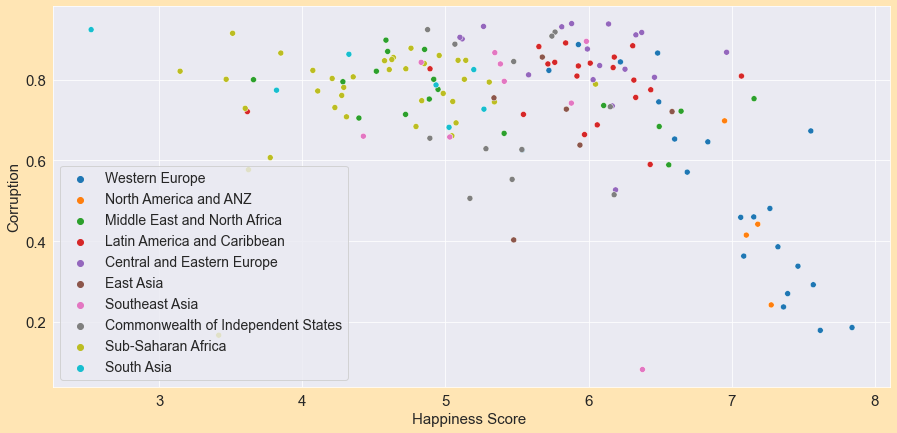
plt.xlabel('Happiness Score ')

6

plt.ylabel('Corruption')

Out[24]:

Text(0, 0.5, 'Corruption')



*#ACP using python*

2

*#let's chose the data we will work on*

3

data\_columns**=**['ladder\_score', 'logged\_gdp\_per\_capital', 'social\_support', 'healthy\_life\_expectancy', 'freedom\_to\_make\_life\_choices', 'corruption']

4

data\_columns

5

data**=**data[data\_columns].copy()

6

data

7

​

Out[34]:

|  | **ladder\_score** | **logged\_gdp\_per\_capital** | **social\_support** | **healthy\_life\_expectancy** | **freedom\_to\_make\_life\_choices** | **corruption** |
| --- | --- | --- | --- | --- | --- | --- |
| **0** | 7.842 | 10.775 | 0.954 | 72.000 | 0.949 | 0.186 |
| **1** | 7.620 | 10.933 | 0.954 | 72.700 | 0.946 | 0.179 |
| **2** | 7.571 | 11.117 | 0.942 | 74.400 | 0.919 | 0.292 |
| **3** | 7.554 | 10.878 | 0.983 | 73.000 | 0.955 | 0.673 |
| **4** | 7.464 | 10.932 | 0.942 | 72.400 | 0.913 | 0.338 |
| **...** | ... | ... | ... | ... | ... | ... |
| **144** | 3.512 | 7.926 | 0.787 | 48.700 | 0.715 | 0.915 |
| **145** | 3.467 | 9.782 | 0.784 | 59.269 | 0.824 | 0.801 |
| **146** | 3.415 | 7.676 | 0.552 | 61.400 | 0.897 | 0.167 |
| **147** | 3.145 | 7.943 | 0.750 | 56.201 | 0.677 | 0.821 |
| **148** | 2.523 | 7.695 | 0.463 | 52.493 | 0.382 | 0.924 |

149 rows × 6 columns

*#let's load the data into an array and standarize it*

2

X**=**np.array(data)

3

X**=** StandardScaler().fit\_transform(X)

4

X

Out[35]:

array([[ 2.15746188e+00, 1.16288545e+00, 1.21617095e+00,

1.03975002e+00, 1.39355041e+00, -3.03122783e+00],

[ 1.95004606e+00, 1.29971670e+00, 1.21617095e+00,

1.14361817e+00, 1.36699018e+00, -3.07041632e+00],

[ 1.90426509e+00, 1.45906449e+00, 1.11137006e+00,

1.39586938e+00, 1.12794812e+00, -2.43780212e+00],

[ 1.88838190e+00, 1.25208557e+00, 1.46943977e+00,

1.18813309e+00, 1.44667086e+00, -3.04828591e-01],

[ 1.80429441e+00, 1.29885068e+00, 1.11137006e+00,..]])

*# do the PCA by creating a pca project as a first step*

2

pca**=**PCA()

3

*#let's apply our APC on our data using the fit\_transform funtion*

4

pca\_components **=** pca.fit\_transform(X)

5

print(pca.n\_components\_)

6

print(pca.explained\_variance\_)

[3.93472876 0.92696777 0.5976831 0.25669291 0.19629055 0.12817745]

Visualisations

list\_acp **=** ["CP1","CP2","CP3","CP4","CP5","CP6"]

2

df\_acp **=** pd.DataFrame(list\_acp, columns **=** ["ACP"])

3

df\_acp['explained\_variance'] **=** pca.explained\_variance\_

4

plt.plot(list\_acp,np.cumsum(pca.explained\_variance\_ratio\_))

5

plt.title("Explained variance vs. # of factors")

6

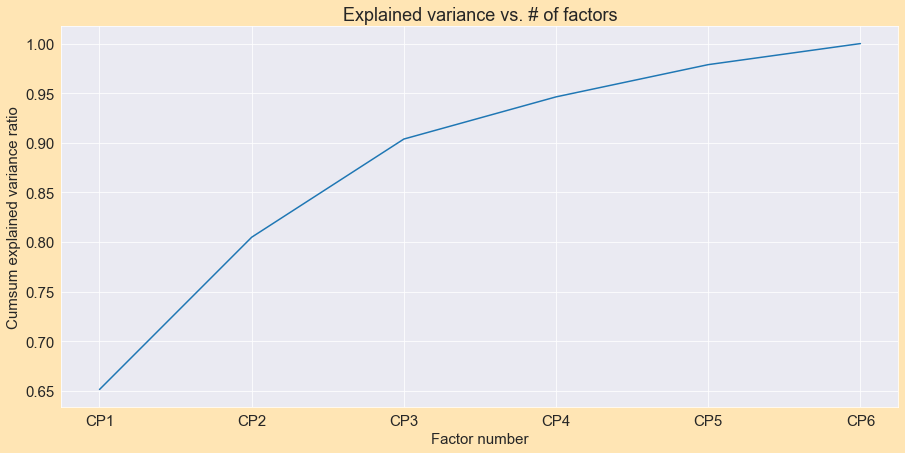
plt.ylabel("Cumsum explained variance ratio")

7

plt.xlabel("Factor number")

8

plt.show()



sns.set(style**=**"whitegrid")

2

​

3

*# Initialize the matplotlib figure*

4

f, ax **=** plt.subplots(figsize**=**(10, 8))

5

​

6

*# Plot the total crashes*

7

sns.set\_color\_codes("pastel")

8

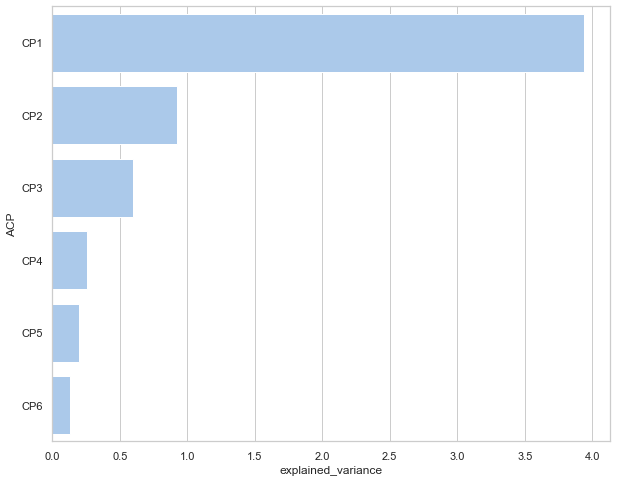
sns.barplot(x**=**"explained\_variance", y**=**"ACP", data**=**df\_acp,

9

label**=**"Total", color**=**"b")

Out[39]:

<AxesSubplot:xlabel='explained\_variance', ylabel='ACP'>



*#let's display our pca components*

2

pca\_components

Out[40]:

array([[-3.78412097e+00, 2.08608068e+00, 6.00297853e-01,

-6.92626617e-01, -4.26021846e-01, -6.73299517e-02],

[-3.79813770e+00, 2.06035524e+00, 7.13717165e-01,

-5.92255755e-01, -2.09555064e-01, -3.14658759e-03],

[-3.67472133e+00, 1.41154788e+00, 7.25352410e-01,

-2.46087521e-01, -2.66623425e-01, 2.86304718e-03],

[-3.19808473e+00, -2.19994836e-01, -6.69402366e-01,

-1.21641887e-01, -3.88057473e-01, -2.76855189e-02],

[-3.33672184e+00, 1.27437819e+00, 5.39054878e-01,

-3.92981747e-01, -2.98217407e-01, 5.31748887e-02],

[-3.69938174e+00, 1.66086648e+00, 4.38041341e-01,

-3.53229266e-01, -3.50753087e-02, 7.48861948e-02],

[-3.49890570e+00, 1.87103014e+00, 5.87173116e-01,

-3.76801849e-01, -1.16785305e-01, 1.96470622e-02],

[-3.35753004e+00, 9.86240025e-01, 6.61593495e-01,

8.84153620e-03, -2.58722724e-01, 6.03732936e-01],

[-3.41666811e+00, 1.78205417e+00, 6.55165827e-01,

-4.48553005e-01, -3.54754254e-02, -2.24510681e-01],

[-3.05172916e+00, 6.20016855e-01, 2.45215418e-01,

-7.68655211e-02, -2.51803421e-01, 1.09170050e-02],

[-3.10813120e+00, 8.07766980e-01, 3.01945778e-01,

-1.02892075e-01, -1.20543503e-01, -1.19428953e-01],

[-2.18698243e+00, -9.29209276e-01, 2.31502970e-01,

1.62264516e-03, -5.47287265e-01, -2.84422265e-01],

[-2.74853079e+00, 7.21302175e-01, 5.31988462e-01,

-3.58304592e-02, -3.29764592e-01, 9.05672098e-02],

[-3.04759503e+00, 9.82002037e-01, 3.74977526e-01,

-4.94011381e-02, -8.64269629e-02, -9.49063732e-02],

])

fig, axes **=** plt.subplots(figsize**=**(12,12))

2

axes.set\_xlim(**-**3,3) *#même limites en abscisse*

3

axes.set\_ylim(**-**3,3) *#et en ordonnée*

4

*#placement des étiquettes des observations*

5

**for** i **in** range(149):

6

plt.annotate(data.index[i],(pca\_components[i,0],pca\_components[i,1]))

7

*#ajouter les axes*

8

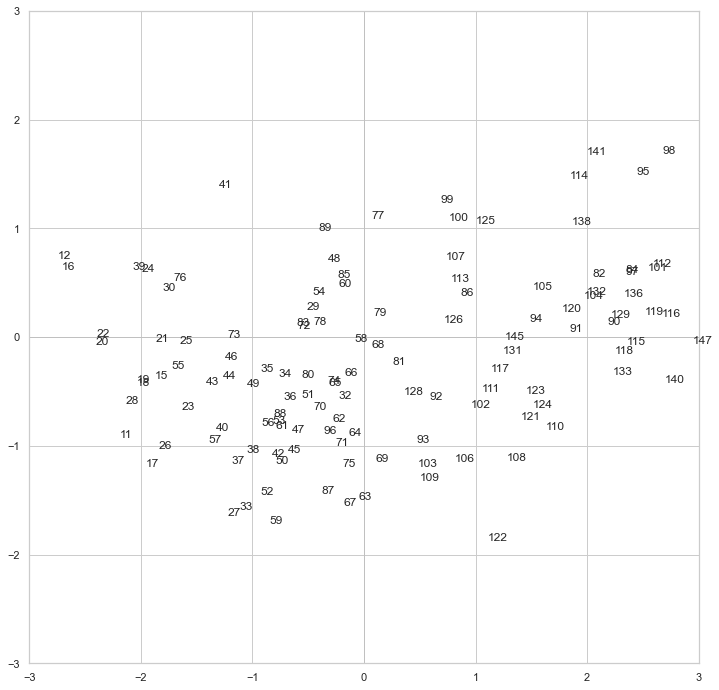
plt.plot([**-**4,4],[0,0],color**=**'silver',linestyle**=**'-',linewidth**=**1)

9

plt.plot([0,0],[**-**4,4],color**=**'silver',linestyle**=**'-',linewidth**=**1)

10

plt.show()



n **=** data.shape[0] *# nb of samples*

2

p **=** data.shape[1] *# nb of features*

3

eigval **=** (n**-**1) **/** n **\*** pca.explained\_variance\_ *# eigen values*

4

sqrt\_eigval **=** np.sqrt(eigval) *# square root of the eigen values*

5

corvar **=** np.zeros((p,p)) *# empty matrix to get the coordinates*

6

**for** k **in** range(p):

7

corvar[:,k] **=** pca.components\_[k,:] **\*** sqrt\_eigval[k]

8

*# the dataframe*

9

coordvar **=** pd.DataFrame({'id': data.columns, 'COR\_1': corvar[:,0], 'COR\_2': corvar[:,1]})

10

coordvar

Out[42]:

|  | **id** | **COR\_1** | **COR\_2** |
| --- | --- | --- | --- |
| **0** | ladder\_score | -0.919123 | -0.016101 |
| **1** | logged\_gdp\_per\_capital | -0.904747 | -0.228055 |
| **2** | social\_support | -0.853906 | -0.321100 |
| **3** | healthy\_life\_expectancy | -0.893620 | -0.166298 |
| **4** | freedom\_to\_make\_life\_choices | -0.679850 | 0.372095 |
| **5** | corruption | 0.505036 | -0.774121 |

1

*# creating an empty figure*

2

fig, axes **=** plt.subplots(figsize **=** (6,6))

3

fig.suptitle("Correlation circle")

4

axes.set\_xlim(**-**1, 1)

5

axes.set\_ylim(**-**1, 1)

6

*# adding the axes*

7

axes.axvline(x **=** 0, color **=** 'lightgray', linestyle **=** '--', linewidth **=** 1)

8

axes.axhline(y **=** 0, color **=** 'lightgray', linestyle **=** '--', linewidth **=** 1)

9

*# adding variables' name*

10

**for** j **in** range(p):

11

axes.text(coordvar["COR\_1"][j],coordvar["COR\_2"][j], coordvar["id"][j])

12

*# adding the circle*

13

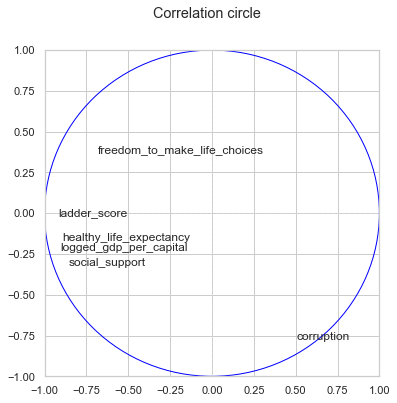
plt.gca().add\_artist(plt.Circle((0,0),1,color**=**'blue',fill**=False**))

14

​

15

plt.show()



AFC

**from** sklearn.decomposition **import** FactorAnalysis

2

model **=** FactorAnalysis(n\_components**=**2)

3

model.fit(X)

4

*# obtain the factors*

5

factors **=** model.components\_

6

*# obtain the contribution of the variables on each factor*

7

variable\_loadings **=** model.transform(X)

factors.shape

Out[52]:

(2, 6)

variable\_loadings.shape

Out[53]:

(149, 2)

*#Create a pandas dataframe that contains the factors and variables' name*

2

df **=** pd.DataFrame({'factor1': factors[0], 'factor2': factors[1], 'variable': data.columns})

3

​

4

*# Creation of scatter plot*

5

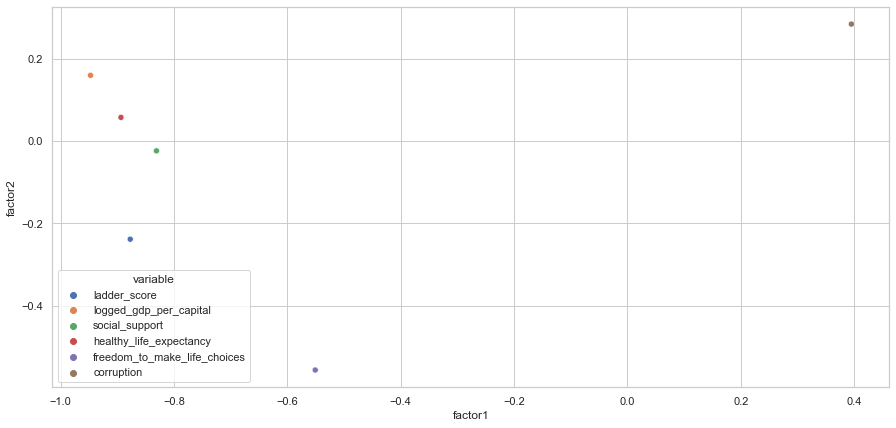
sns.scatterplot(x**=**'factor1', y**=**'factor2', hue**=**'variable', data**=**df)

6

​

7

plt.show()



fig, axes **=** plt.subplots(figsize**=**(12,12))

2

axes.set\_xlim(**-**1,2)

3

axes.set\_ylim(**-**1,2)

4

​

5

**for** i **in** range(149):

6

plt.annotate(data.index[i],(variable\_loadings[i,0],variable\_loadings[i,1]))

7

*#ajouter les axes*

8

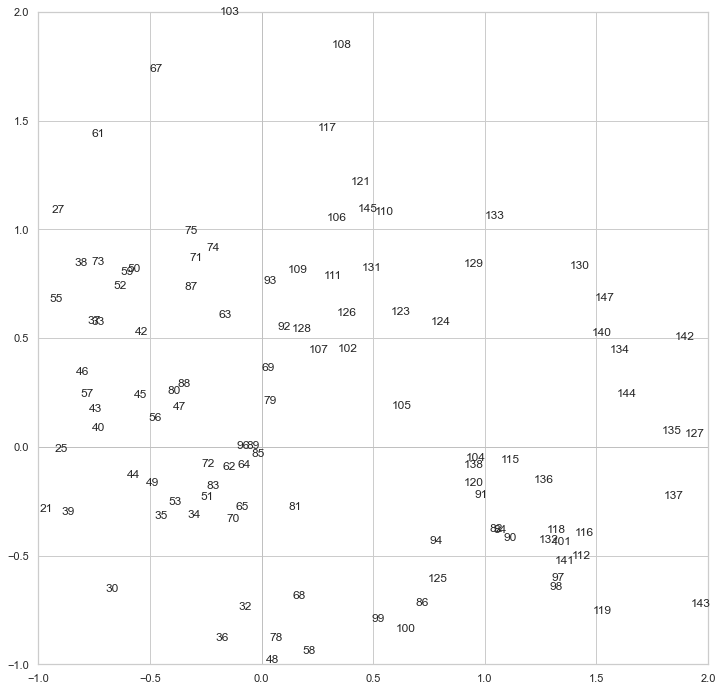
plt.plot([**-**4,4],[0,0],color**=**'silver',linestyle**=**'-',linewidth**=**1)

9

plt.plot([0,0],[**-**4,4],color**=**'silver',linestyle**=**'-',linewidth**=**1)

10

plt.show()



cor**=** data.corr(method**=**"pearson")

2

f,ax**=**plt.subplots(figsize**=**(20,10))

3

sns.heatmap(cor,mask**=**np.zeros\_like(cor,dtype**=**np.bool),cmap**=**"Blues",square**=True**,annot**=True**)

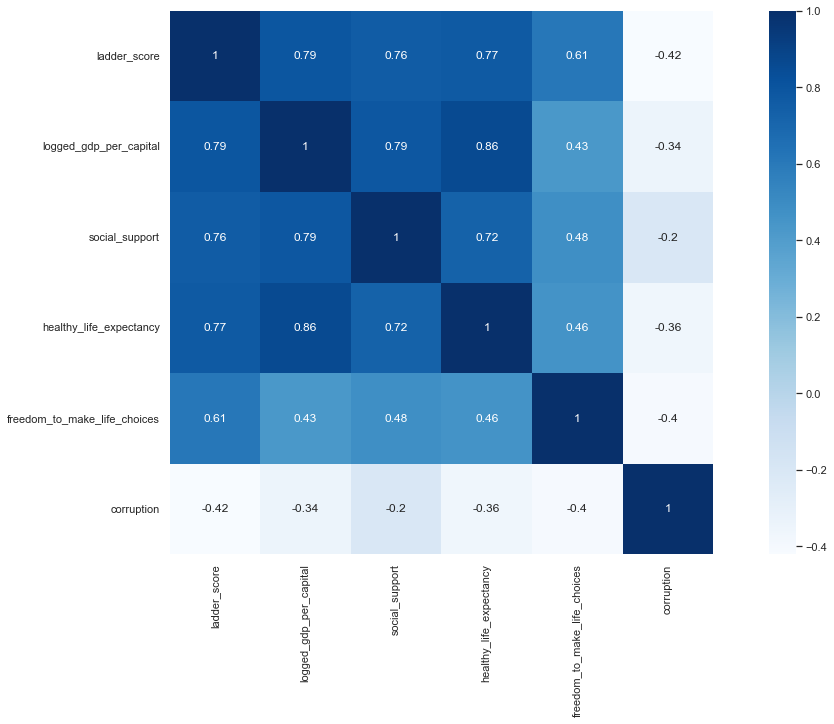
C:\Users\WhiteLuce SIS\AppData\Local\Temp\ipykernel\_14392\3860571064.py:3: DeprecationWarning: `np.bool` is a deprecated alias for the builtin `bool`. To silence this warning, use `bool` by itself. Doing this will not modify any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.bool\_` here.

Deprecated in NumPy 1.20; for more details and guidance: <https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations>

sns.heatmap(cor,mask=np.zeros\_like(cor,dtype=np.bool),cmap="Blues",square=True,annot=True)

Out[50]:

<AxesSubplot:>



**2/ Code R** :

> # chargement de données

> data <- read.csv("C:\\Users\\WhiteLuce SIS\\OneDrive\\if5\\analyse de données\\world-happiness-report-2021.csv")

> View(data)

> data

Country.name Regional.indicator Ladder.score

1 Finland Western Europe 7.842

2 Denmark Western Europe 7.620

3 Switzerland Western Europe 7.571

4 Iceland Western Europe 7.554

5 Netherlands Western Europe 7.464

Standard.error.of.ladder.score upperwhisker lowerwhisker Logged.GDP.per.capita

1 0.032 7.904 7.780 10.775

2 0.035 7.687 7.552 10.933

3 0.036 7.643 7.500 11.117

4 0.059 7.670 7.438 10.878

5 0.027 7.518 7.410 10.932

Social.support Healthy.life.expectancy Freedom.to.make.life.choices Generosity

1 0.954 72.000 0.949 -0.098

2 0.954 72.700 0.946 0.030

3 0.942 74.400 0.919 0.025

4 0.983 73.000 0.955 0.160

5 0.942 72.400 0.913 0.175

> colnames(data)

[1] "Country.name" "Regional.indicator"

[3] "Ladder.score" "Standard.error.of.ladder.score"

[5] "upperwhisker" "lowerwhisker"

[7] "Logged.GDP.per.capita" "Social.support"

[9] "Healthy.life.expectancy" "Freedom.to.make.life.choices"

[11] "Generosity" "Perceptions.of.corruption"

[13] "Ladder.score.in.Dystopia" "Explained.by..Log.GDP.per.capita"

[15] "Explained.by..Social.support" "Explained.by..Healthy.life.expectancy"

[17] "Explained.by..Freedom.to.make.life.choices" "Explained.by..Generosity"

[19] "Explained.by..Perceptions.of.corruption" "Dystopia...residual"

> data1 <- data[, c(

+ "Ladder.score",

+ "Logged.GDP.per.capita",

+ "Social.support",

+ "Healthy.life.expectancy",

+ "Freedom.to.make.life.choices", "Perceptions.of.corruption" )]

> head(data1)

Ladder.score Logged.GDP.per.capita Social.support Healthy.life.expectancy

1 7.842 10.775 0.954 72.0

2 7.620 10.933 0.954 72.7

3 7.571 11.117 0.942 74.4

4 7.554 10.878 0.983 73.0

5 7.464 10.932 0.942 72.4

6 7.392 11.053 0.954 73.3

Freedom.to.make.life.choices Perceptions.of.corruption

1 0.949 0.186

2 0.946 0.179

3 0.919 0.292

4 0.955 0.673

5 0.913 0.338

6 0.960 0.270

> #standariser les données

> data1\_scaled <- scale(data1)

> head(data1\_scaled)

Ladder.score Logged.GDP.per.capita Social.support Healthy.life.expectancy

[1,] 2.150210 1.158977 1.212083 1.036255

[2,] 1.943491 1.295348 1.212083 1.139774

[3,] 1.897864 1.454160 1.107634 1.391177

[4,] 1.882034 1.247877 1.464500 1.184139

[5,] 1.798230 1.294485 1.107634 1.095409

[6,] 1.731186 1.398921 1.212083 1.228505

Freedom.to.make.life.choices Perceptions.of.corruption

[1,] 1.388866 -3.021039

[2,] 1.362395 -3.060096

[3,] 1.124157 -2.429608

[4,] 1.441808 -0.303804

[5,] 1.071215 -2.172949

[6,] 1.485926 -2.552358

> library(FactoMineR)

> Result <-PCA(data1\_scaled)

> summary(Result)

Call:

PCA(X = data1\_scaled)

Eigenvalues

Dim.1 Dim.2 Dim.3 Dim.4 Dim.5 Dim.6

Variance 3.908 0.921 0.594 0.255 0.195 0.127

% of var. 65.139 15.346 9.895 4.250 3.250 2.122

Cumulative % of var. 65.139 80.484 90.379 94.628 97.878 100.000

Individuals (the 10 first)

Dist Dim.1 ctr cos2 Dim.2 ctr cos2 Dim.3

1 | 4.438 | 3.784 2.459 0.727 | -2.086 3.172 0.221 | -0.600

2 | 4.424 | 3.798 2.477 0.737 | -2.060 3.094 0.217 | -0.714

3 | 4.019 | 3.675 2.319 0.836 | -1.412 1.452 0.123 | -0.725

4 | 3.300 | 3.198 1.756 0.939 | 0.220 0.035 0.004 | 0.669

5 | 3.646 | 3.337 1.912 0.837 | -1.274 1.184 0.122 | -0.539

6 | 4.095 | 3.699 2.350 0.816 | -1.661 2.011 0.165 | -0.438

7 | 4.030 | 3.499 2.102 0.754 | -1.871 2.552 0.216 | -0.587

8 | 3.621 | 3.358 1.936 0.860 | -0.986 0.709 0.074 | -0.662

9 | 3.941 | 3.417 2.005 0.752 | -1.782 2.315 0.204 | -0.655

10 | 3.135 | 3.052 1.599 0.948 | -0.620 0.280 0.039 | -0.245

ctr cos2

1 0.407 0.018 |

2 0.576 0.026 |

3 0.595 0.033 |

4 0.507 0.041 |

5 0.328 0.022 |

6 0.217 0.011 |

7 0.390 0.021 |

8 0.495 0.033 |

9 0.485 0.028 |

10 0.068 0.006 |

Variables

Dim.1 ctr cos2 Dim.2 ctr cos2 Dim.3 ctr

Ladder.score | 0.919 21.615 0.845 | 0.016 0.028 0.000 | 0.048 0.392

Logged.GDP.per.capita | 0.905 20.944 0.819 | 0.228 5.649 0.052 | -0.199 6.640

Social.support | 0.854 18.656 0.729 | 0.321 11.198 0.103 | 0.080 1.084

Healthy.life.expectancy | 0.894 20.432 0.799 | 0.166 3.004 0.028 | -0.190 6.065

Freedom.to.make.life.choices | 0.680 11.826 0.462 | -0.372 15.037 0.138 | 0.616 63.871

Perceptions.of.corruption | -0.505 6.526 0.255 | 0.774 65.084 0.599 | 0.361 21.947

cos2

Ladder.score 0.002 |

Logged.GDP.per.capita 0.039 |

Social.support 0.006 |

Healthy.life.expectancy 0.036 |

Freedom.to.make.life.choices 0.379 |

Perceptions.of.corruption 0.130 |

> library(factoextra)

> eig.val<- get\_eigenvalue(Result)

> eig.val

eigenvalue variance.percent cumulative.variance.percent

Dim.1 3.9083212 65.138686 65.13869

Dim.2 0.9207465 15.345775 80.48446

Dim.3 0.5936718 9.894530 90.37899

Dim.4 0.2549701 4.249502 94.62849

Dim.5 0.1949732 3.249553 97.87805

Dim.6 0.1273172 2.121953 100.00000

> fviz\_eig(Result, addlabels = TRUE, ylim = c(0, 50))

> #AFC

> library(FactoMineR)

> ca1<-CA(data1, graph=TRUE, ncp=2)

> summary(data1)

Ladder.score Logged.GDP.per.capita Social.support Healthy.life.expectancy

Min. :2.523 Min. : 6.635 Min. :0.4630 Min. :48.48

1st Qu.:4.852 1st Qu.: 8.541 1st Qu.:0.7500 1st Qu.:59.80

Median :5.534 Median : 9.569 Median :0.8320 Median :66.60

Mean :5.533 Mean : 9.432 Mean :0.8147 Mean :64.99

3rd Qu.:6.255 3rd Qu.:10.421 3rd Qu.:0.9050 3rd Qu.:69.60

Max. :7.842 Max. :11.647 Max. :0.9830 Max. :76.95

Freedom.to.make.life.choices Perceptions.of.corruption

Min. :0.3820 Min. :0.0820

1st Qu.:0.7180 1st Qu.:0.6670

Median :0.8040 Median :0.7810

Mean :0.7916 Mean :0.7274

3rd Qu.:0.8770 3rd Qu.:0.8450

Max. :0.9700 Max. :0.9390

> summary(ca1, nb.dec=2,ncp=2)

Call:

CA(X = data1, ncp = 2, graph = TRUE)

The chi square of independence between the two variables is equal to 30.63998 (p-value = 1 ).

Eigenvalues

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

Variance 0.00 0.00 0.00 0.00 0.00

% of var. 54.52 23.31 13.76 5.96 2.45

Cumulative % of var. 54.52 77.83 91.59 97.55 100.00

Rows (the 10 first)

Iner\*1000 Dim.1 ctr cos2 Dim.2 ctr cos2

1 | 0.08 | 0.10 5.18 0.93 | -0.03 0.92 0.07 |

2 | 0.07 | 0.09 4.29 0.88 | -0.03 1.31 0.11 |

3 | 0.05 | 0.07 3.06 0.87 | -0.03 0.98 0.12 |

4 | 0.02 | 0.05 1.68 0.94 | 0.01 0.16 0.04 |

5 | 0.04 | 0.07 3.03 0.93 | -0.02 0.47 0.06 |

6 | 0.05 | 0.07 3.00 0.86 | -0.03 1.01 0.12 |

7 | 0.05 | 0.08 3.17 0.84 | -0.03 1.27 0.14 |

8 | 0.04 | 0.07 2.58 0.85 | -0.01 0.16 0.02 |

9 | 0.05 | 0.07 2.67 0.78 | -0.04 1.77 0.22 |

10 | 0.02 | 0.06 1.72 0.95 | -0.01 0.21 0.05 |

Columns

Iner\*1000 Dim.1 ctr cos2 Dim.2 ctr cos2

Ladder.score | 0.99 | 0.11 64.01 0.88 | 0.03 12.47 0.07 |

Logged.GDP.per.capita | 0.33 | 0.01 1.68 0.07 | 0.02 9.35 0.16 |

Social.support | 0.08 | 0.02 0.38 0.06 | 0.04 3.05 0.21 |

Healthy.life.expectancy | 0.16 | -0.01 5.83 0.51 | -0.01 11.05 0.41 |

Freedom.to.make.life.choices | 0.16 | 0.02 0.44 0.04 | 0.02 0.69 0.02 |

Perceptions.of.corruption | 0.77 | -0.21 27.66 0.49 | 0.20 63.39 0.48 |

> fviz\_screeplot (ca1, addlabels = TRUE, ylim = c(0, 50))