DATAVISUALISATION – PROJET SEABORN & PLOTLY

2 ÉTAPES – 2 JEUX DE DONNÉES – VISUALISATIONS STATISTIQUES &INTERACTIVES

INTRODUCTION GÉNÉRALE

Ce projet vise à analyser deux jeux de données réels en utilisant deux approches complémentaires :

- Seaborn pour réaliser une analyse statistique descriptive sur le dataset Netflix.
- Plotly pour produire des visualisations interactives sur le dataset World Happiness Report.

L'objectif est de comprendre les tendances, identifier les patterns et présenter les résultats de manière visuelle et professionnelle.

PARTIE I
ANALYSE NETFLIX (SEABORN)

- Présentationdu dataset
- Préparation et nettoyage des données
- Visualisations et analyses détaillées
- Conclusion–AnalyseNetflix

```
df = pd.read_csv("netflix_titles.csv") # Charger le fichier CSV contenant les données Netflix
```

```
# Afficher les 5 premières lignes pour voir à quoi ressemble le dataset
df.head()
```

	show_id	type	title	director	cast	country	date_added	release_year	rating	duration	listed_in	description
0	s1	Movie	Dick Johnson Is Dead	Kirsten Johnson	NaN	United States	September 25, 2021	2020	PG-13	90 min	Documentaries	As her father nears the end of his life, filmm...
1	s2	TV Show	Blood & Water	NaN	Ama Qamata, Khosi Ngema, Gail Mabalane, Thaban...	South Africa	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	After crossing paths at a party, a Cape Town t...
2	s3	TV Show	Ganglands	Julien Leclercq	Sami Bouajila, Tracy Gotoas, Samuel Jouy, Nabi...	NaN	September 24, 2021	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	To protect his family from a powerful drug lor...
3	s4	TV Show	Jailbirds New Orleans	NaN	NaN	NaN	September 24, 2021	2021	TV-MA	1 Season	Docuseries, Reality TV	Feuds, flirtations and toilet talk go down amo...
4	s5	TV Show	Kota Factory	NaN	Mayur More, Jitendra Kumar, Ranjan Raj, Alam K...	India	September 24, 2021	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	In a city of coaching centers known to train I...

```
# Afficher des informations générales sur le DataFrame :
# - nombre de lignes et colonnes
# - type de chaque colonne (object, int, float, datetime...)
# - nombre de valeurs non nulles
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

```
# Voir combien de valeurs manquantes il y a dans chaque colonne
df.isna().sum()
```

```
show_id          0
type             0
title            0
director      2634
cast           825
country        831
date_added      10
release_year     0
rating           4
duration         3
listed_in        0
description      0
dtype: int64
```

Après chargement du fichier netflix_titles.csv, les premières lignes montrent que chaque entrée correspond à un film ou une série, avec les métadonnées associées :

- 8 807 lignes
- 12 colonnes
- Mélange de variables catégorielles et numériques

Colonnes disponibles :

- show_id — Identifiant interne
- type — Movie ou TV Show
- title — Titre du contenu
- director — Réalisateur
- cast — Acteurs principaux
- country — Pays d'origine
- date_added — Date d'ajout sur Netflix
- release_year — Année de sortie
- rating — Classification d'âge
- duration — Durée (minutes ou saisons)
- listed_in — Genres
- description

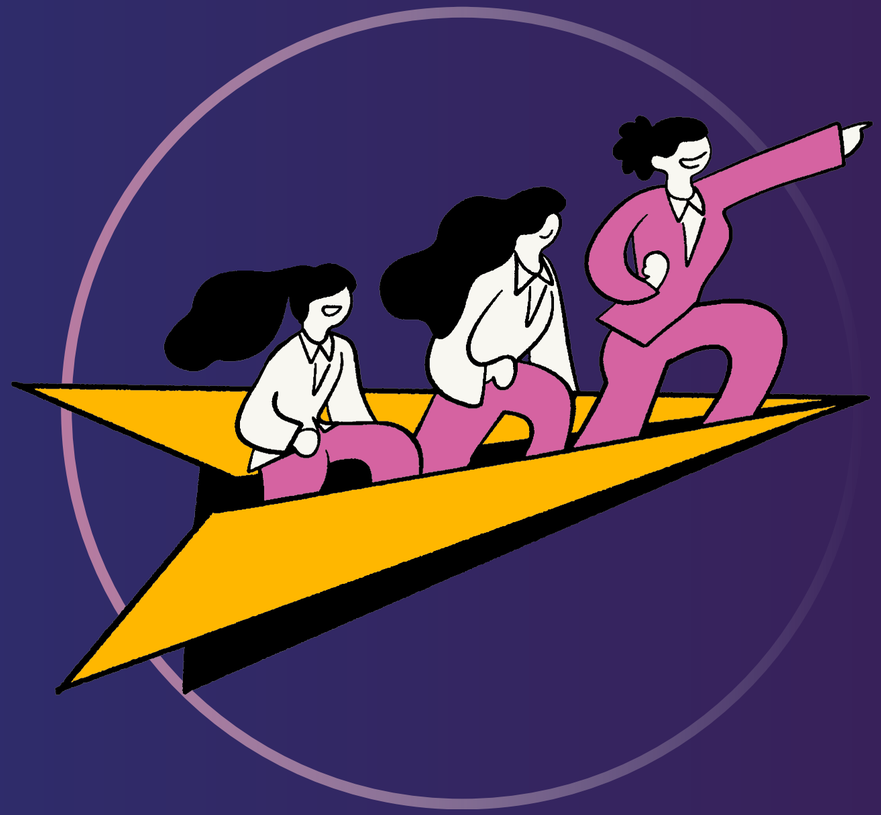
```
# Voir combien de valeurs manquantes il y a dans chaque colonne
df.isna().sum()
```

```
show_id          0
type             0
title            0
director      2634
cast           825
country        831
date_added      10
release_year     0
rating           4
duration         3
listed_in        0
description      0
dtype: int64
```

RÉSUMÉL'ANALYSE DES VALEURS MANQUANTES (DF.ISNA().SUM()) MONTRE QUE CERTAINES COLONNES NÉCESSITENT UN TRAITEMENT

Variable	Valeurs manquantes
director	2634
cast	825
country	831
date_added	10
rating	4
duration	3
listed_in / title / type / release_year / description	0

- LES COLONNES AVEC BEAUCOUP DE TEXTE (DIRECTOR, CAST, DESCRIPTION) NE SONT PAS UTILES POUR NOS VISUALISATIONS.
- CERTAINES COLONNES DOIVENT ÊTRE RÉPARÉES AVANT ANALYSE (DURATION, DATE_ADDED).



PRÉPARATION

ET NETTOYAGE DES DONNÉES

```
# Voir combien de valeurs manquantes il y a dans chaque colonne
df.isna().sum()
```

```
show_id          0
type             0
title            0
director      2634
cast           825
country        831
date_added      10
release_year     0
rating           4
duration         3
listed_in        0
description      0
dtype: int64
```

```
# Afficher des informations générales sur le DataFrame :
# - nombre de lignes et colonnes
# - type de chaque colonne (object, int, float, datetime...)
# - nombre de valeurs non nulles
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column        Non-Null Count  Dtype 
---  ------        --------------  ----- 
 0   show_id       8807 non-null   object
 1   type          8807 non-null   object
 2   title         8807 non-null   object
 3   director      6173 non-null   object
 4   cast          7982 non-null   object
 5   country       7976 non-null   object
 6   date_added    8797 non-null   object
 7   release_year  8807 non-null   int64 
 8   rating        8803 non-null   object
 9   duration      8804 non-null   object
 10  listed_in     8807 non-null   object
 11  description   8807 non-null   object
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Après chargement du fichier netflix_titles.csv, les premières lignes montrent que chaque entrée correspond à un film ou une série, avec les métadonnées associées :

- 8 807 lignes
- 12 colonnes
- Mélange de variables catégorielles et numériques

Colonnes disponibles :

- show_id — Identifiant interne
- type — Movie ou TV Show
- title — Titre du contenu
- director — Réalisateur
- cast — Acteurs principaux
- country — Pays d'origine
- date_added — Date d'ajout sur Netflix
- release_year — Année de sortie
- rating — Classification d'âge
- duration — Durée (minutes ou saisons)
- listed_in — Genres
- description

TRANSFORMATION DE LA DATE (DATE_ADDED → YEAR_ADDED)

LE FORMAT D'ORIGINE ÉTAIT : MONTH DAY, YEAR (EX : "SEPTEMBER 25, 2021").

CONVERSION EN DATETIME AVEC PD.TO_DATETIME().

EXTRACTION DE L'ANNÉE VIA .DT.YEAR.

OBJECTIF : FACILITER LES ANALYSES TEMPORELLES SUR L'ÉVOLUTION DU CATALOGUE NETFLIX.

DURATION_VALUE (EXTRACTION DE LA PARTIE NUMÉRIQUE)

FILMS : "90 MIN" → 90

SÉRIES : "2 SEASONS" → 2

DURATION_UNIT (TYPE D'UNITÉ)

FILMS → "MIN"

SÉRIES → "SEASON"

CES DEUX VARIABLES PERMETTENT UNE ANALYSE SÉPARÉE DES FILMS ET SÉRIES SANS MÉLANGER LES UNITÉS

```
# Extraire la partie numérique (minutes ou nombre de saisons)
df["duration_value"] = df["duration"].str.extract(r"(\d+)").astype(int)

# Définir l'unité : minutes pour films, seasons pour séries
df["duration_unit"] = df["duration"].apply(
    lambda x: "min" if "min" in x.lower() else "season"
)
```

```
df[["type", "duration", "duration_value", "duration_unit"]].head()
```

	type	duration	duration_value	duration_unit
0	Movie	90 min	90	min
1	TV Show	2 Seasons	2	season
2	TV Show	1 Season	1	season
3	TV Show	1 Season	1	season
4	TV Show	2 Seasons	2	season

CERTAINES COLONNES DU DATASET INITIAL NE PRÉSENTAIENT PAS D'INTÉRÊT ANALYTIQUE OU CONTENAIENT TROP DE VALEURS MANQUANTES POUR ÊTRE EXPLOITABLES CORRECTEMENT.
ELLES ONT ÉTÉ SUPPRIMÉES POUR SIMPLIFIER LE DATAFRAME ET ÉVITER LE BRUIT STATISTIQUE.

```python
cols_to_drop = ["show_id", "director", "cast", "description", "date_added"]
df = df.drop(columns=cols_to_drop)
#show_id : identifiant interne, n'apporte aucune information utile pour l'analyse.

#director : très nombreuses valeurs manquantes et non utilisé dans les visualisations.

#cast : texte long, difficile à exploiter et inutile pour les graphiques demandés.

#description : colonne textuelle non utilisée dans une analyse statistique.

#date_added : remplacée par year_added qui suffit pour l'analyse temporelle.
#« Nous avons conservé la colonne rating car elle permet d'analyser la classification d'âge des contenus Netflix.
#Cette variable n'est pas utilisée dans les visualisations obligatoires,
#mais elle est pertinente pour une analyse additionnelle portant sur le public ciblé par la plateforme. »


#Nous avons conservé la colonne year_added pour analyser l'évolution annuelle du catalogue Netflix.
#Cette variable permet d'étudier la dynamique d'ajout de contenus sur la plateforme,
#ce qui constitue une analyse additionnelle pertinente
```

```python
df.isna().sum()
```

```
type            0
title           0
country         0
release_year    0
rating          0
duration        0
listed_in       0
year_added     88
dtype: int64
```

COLONNES SUPPRIMÉES :
SHOW_ID : IDENTIFIANT INTERNE, AUCUNE VALEUR ANALYTIQUE.
DIRECTOR : PLUS DE 2 600 VALEURS MANQUANTES → VARIABLE INEXPLOITABLE.
CAST : CHAMP TEXTUEL TROP LONG ET NON UTILISABLE DANS LES VISUALISATIONS DEMANDÉES.
DESCRIPTION : COLONNE TEXTUELLE DESCRIPTIVE INUTILE POUR UNE ANALYSE STATISTIQUE.
DATE_ADDED : REMPLACÉE PAR UNE NOUVELLE VARIABLE PLUS PERTINENTE YEAR_ADDED.

LE DATASET NETFLIX CONTENAIT PLUSIEURS VALEURS NULLES.
LE TRAITEMENT DÉPENDAIT DE LA NATURE DE CHAQUE VARIABLE :
COUNTRY
VARIABLE ESSENTIELLE POUR L'ANALYSE GÉOGRAPHIQUE.
IMPOSSIBLE DE SUPPRIMER LES LIGNES (TROP DE PERTES).
REMPLACEMENT PAR "UNKNOWN".
RATING
TRÈS PEU DE VALEURS MANQUANTES.
VARIABLE CATÉGORIELLE → REMPLACEMENT PAR LA MODALITÉ LA PLUS FRÉQUENTE.
IMPUTATION PAR LE MODE.
DURATION
3 VALEURS MANQUANTES.
LA DURÉE EST INDISPENSABLE POUR LES BOXPLOTS FILM VS SÉRIE.
SUPPRESSION UNIQUEMENT DES LIGNES MANQUANTES.
DATE_ADDED
10 LIGNES MANQUANTES SUR 8 807.
CETTE VARIABLE EST NÉCESSAIRE POUR CRÉER YEAR_ADDED.
LIGNES SUPPRIMÉES (IMPACT NÉGLIGEABLE : 0,1%).
APRÈS TRAITEMENT, LE DATAFRAME NE CONTIENT PLUS AUCUNE VALEUR MANQUANTE.

```
df.isna().sum()
```

```
type            0
title           0
country         0
release_year    0
rating          0
duration        0
listed_in       0
year_added      0
dtype: int64
```

```
df.head()
```

	type	title	country	release_year	rating	duration	listed_in	year_added
0	Movie	Dick Johnson Is Dead	United States	2020	PG-13	90 min	Documentaries	2021.0
1	TV Show	Blood & Water	South Africa	2021	TV-MA	2 Seasons	International TV Shows, TV Dramas, TV Mysteries	2021.0
2	TV Show	Ganglands	Unknown	2021	TV-MA	1 Season	Crime TV Shows, International TV Shows, TV Act...	2021.0
3	TV Show	Jailbirds New Orleans	Unknown	2021	TV-MA	1 Season	Docuseries, Reality TV	2021.0
4	TV Show	Kota Factory	India	2021	TV-MA	2 Seasons	International TV Shows, Romantic TV Shows, TV ...	2021.0

```python
#country
#La colonne country est essentielle pour l'analyse géographique.
#On ne veut pas supprimer les lignes, mais on ne peut pas remplacer par le mode, car ça fausserait les stats.
#On remplace les NaN par "Unknown" (pays non spécifié).
df["country"] = df["country"].fillna("Unknown")
```

```python
#rating
#La variable rating présente très peu de valeurs manquantes.
#Elles ont été remplacées par la modalité la plus fréquente afin de conserver la cohérence de cette variable catégorielle
df["rating"] = df["rating"].fillna(df["rating"].mode()[0])
```

```python
#duration
#On a 3 valeurs manquantes dans duration.
#Cette colonne est essentielle pour comparer la durée des films vs séries (boxplot).
# Supprimer les lignes où la durée est manquante
df = df.dropna(subset=["duration"])
```

```python
#on veut une colonne propre pour créer l'année d'ajout.
#Supprimer 10 lignes sur >5 000 n'a presque aucun impact.#date_added
df = df.dropna(subset=["date_added"])
```

```python
# Conversion sécurisée des dates (pandas reconnaît parfaitement le format Month Day, Year)
df["date_added"] = pd.to_datetime(df["date_added"], errors="coerce")
```

```python
df["date_added"].head()
```

```
0   2021-09-25
1   2021-09-24
2   2021-09-24
3   2021-09-24
4   2021-09-24
Name: date_added, dtype: datetime64[ns]
```

```python
#La colonne date_added contient une date complète, qui n'est pas pratique à analyser telle quelle.
#Nous avons donc extrait uniquement l'année d'ajout dans une nouvelle colonne year_added,
#ce qui permet de visualiser facilement la progression du nombre de contenus ajoutés par Netflix d'année en année.
df["year_added"] = df["date_added"].dt.year
```

```python
df[["date_added", "year_added"]].head()
```

	date_added	year_added
0	2021-09-25	2021.0
1	2021-09-24	2021.0
2	2021-09-24	2021.0
3	2021-09-24	2021.0
4	2021-09-24	2021.0

L'ANALYSE VISE À RÉPONDRE AUX QUESTIONS SUIVANTES :

Structure du catalogue Netflix

Films vs séries : qui domine réellement la plateforme ?

Répartition géographique

Quels pays produisent le plus de contenus ?

Dimension temporelle

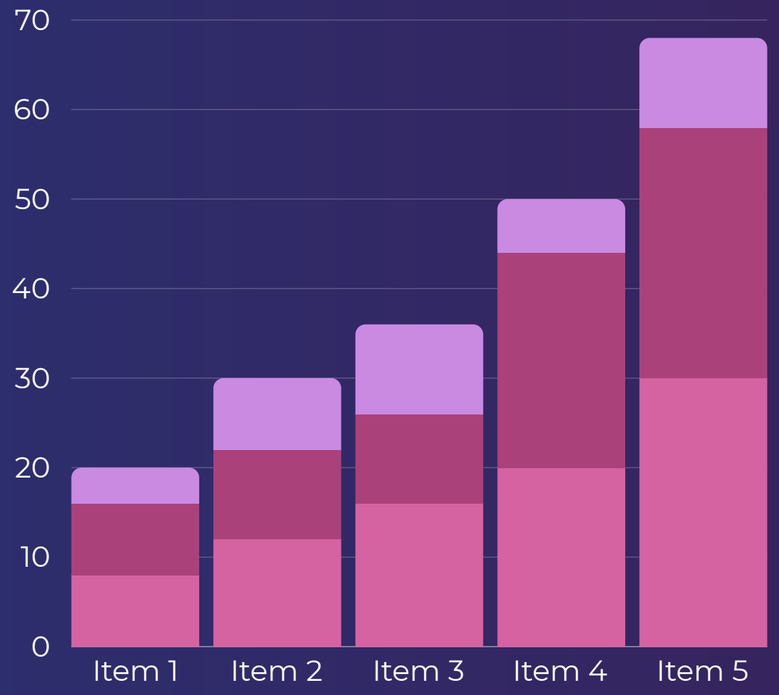
Comment les ajouts de contenus ont-ils évolué au fil des années ?

Caractéristiques des contenus

- Les films sont-ils plus longs que les séries ?
- Quels ratings sont les plus présents ?

Les années de sortie sont-elles concentrées sur une période ?

CES OBJECTIFS DÉTERMINERONT LES TYPES DE VISUALISATIONS UTILISÉES AVEC SEABORN (COUNTPLOT, BARPLOT, HISTPLOT, HEATMAP, BOXPLOT).

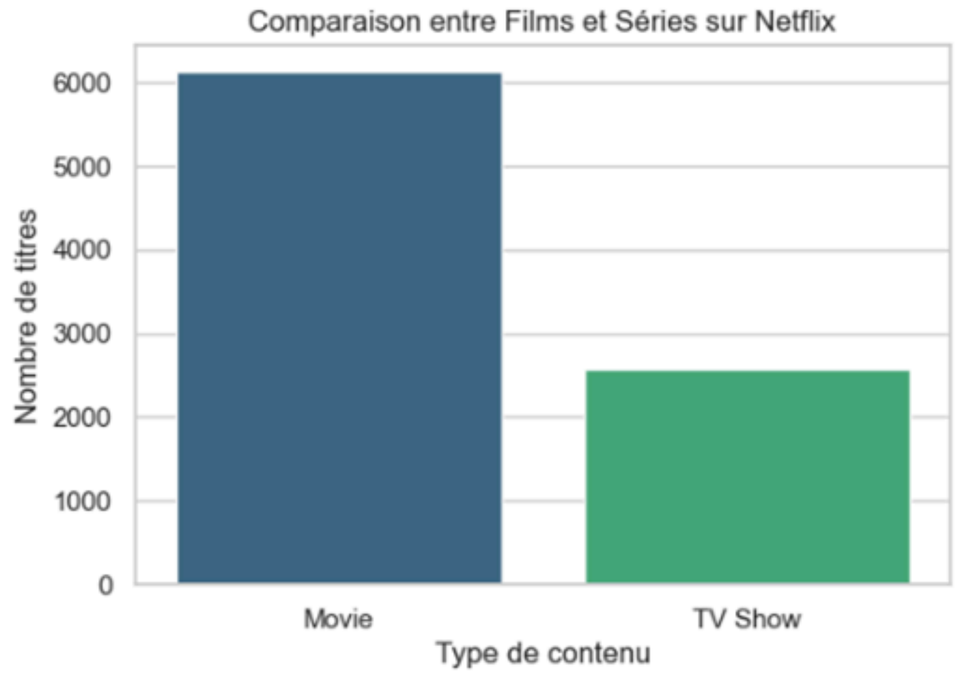


VISUALISATIONS
ET ANALYSES DÉTAILLÉES

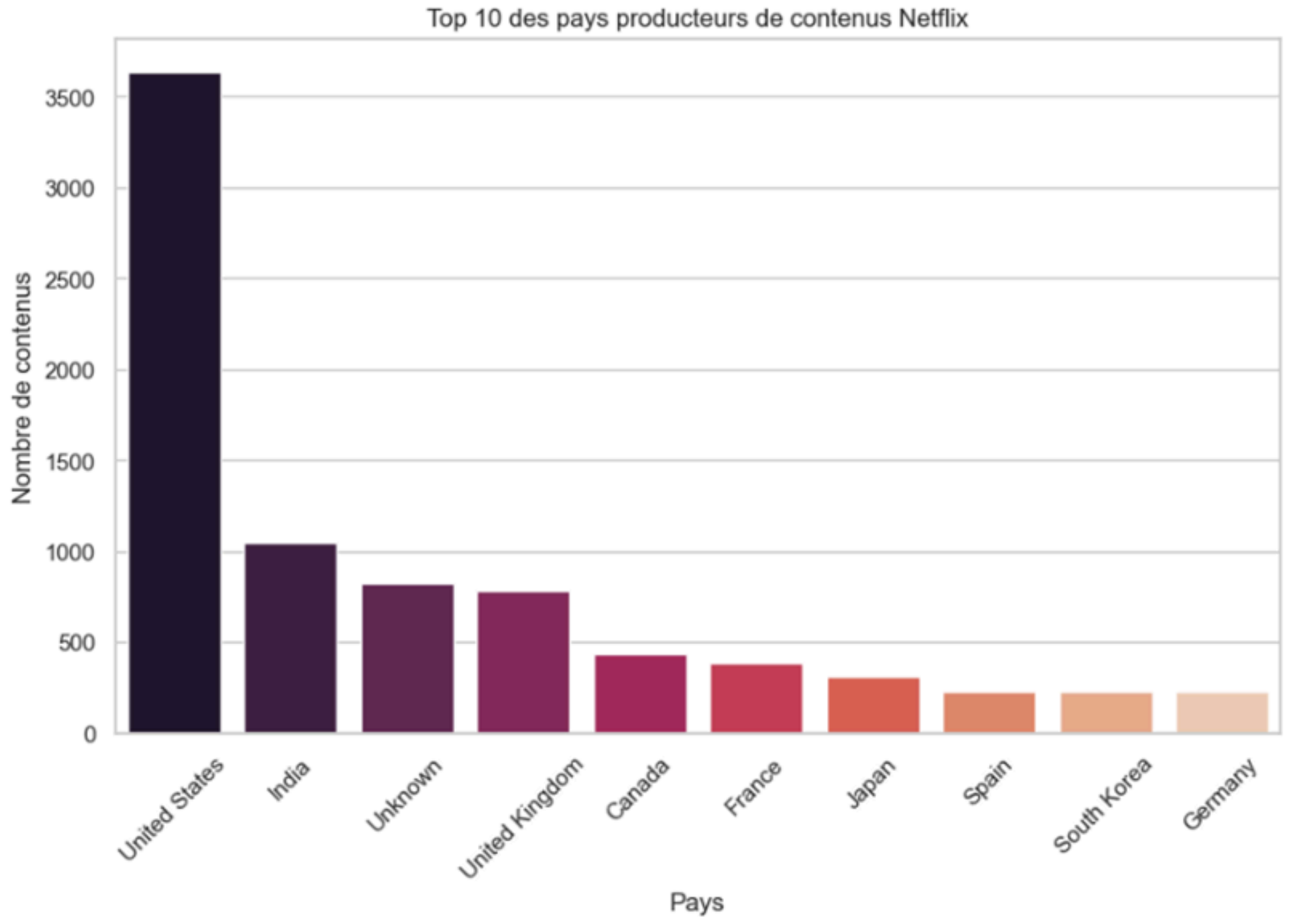
```
#countplot() - Films vs Séries
#barplot() - Top 10 Pays producteurs
#histplot() - Distribution des années de sortie
#heatmap() - Corrélation entre variables quantitatives
#boxplot() - Durée moyenne selon le type (Movie / TV Show)
#Analyse Additionnelle 1 : Répartition des contenus par Rating (classification d'âge)
#Analyse Additionnelle 2 : Évolution des contenus ajoutés par Année (year_added)
```

```
#Netflix propose nettement plus de films que de séries.
#Les films représentent la majorité du catalogue,
#ce qui montre une orientation plus forte vers le contenu cinématographique
```

```
#1 countplot – Films vs Séries / But : comparer la quantité de films et de séries.
fig_films_series = plt.figure(figsize=(6,4))
sns.countplot(data=df, x="type", palette="viridis")
plt.title("Comparaison entre Films et Séries sur Netflix")
plt.xlabel("Type de contenu")
plt.ylabel("Nombre de titres")
plt.show()
```



```python
#2 barplot – Top 10 pays producteurs
# Séparer les pays et compter les occurrences
top10 = df["country"].str.split(", ").explode().value_counts().head(10).reset_index()
top10.columns = ["country", "count"]
fig_top10 = plt.figure(figsize=(10,6))
sns.barplot(data=top10, x="country", y="count", palette="rocket")
plt.title("Top 10 des pays producteurs de contenus Netflix")
plt.xticks(rotation=45)
plt.xlabel("Pays")
plt.ylabel("Nombre de contenus")
plt.show()
```

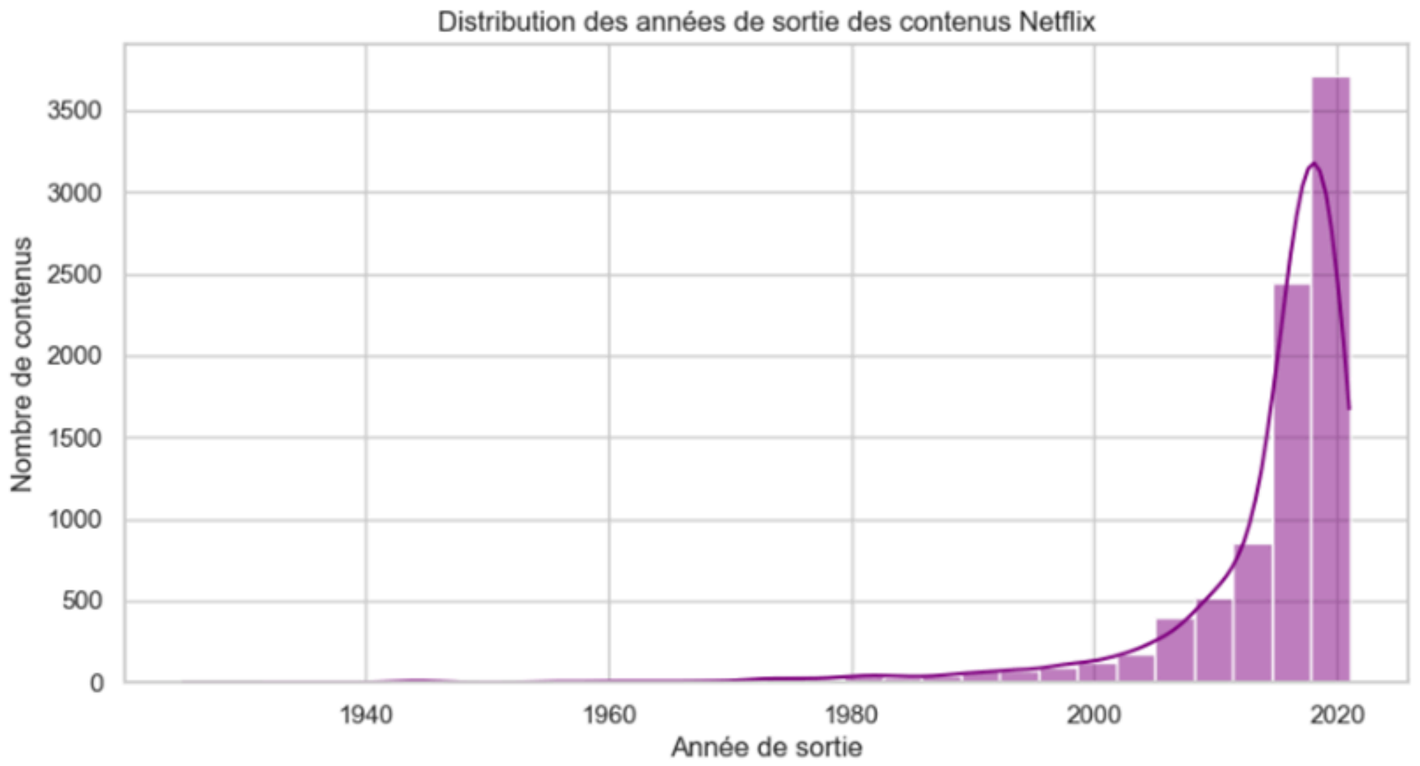



```
#Les États-Unis dominent largement la production de contenus présents sur Netflix, suivis par l'Inde,
#le Royaume-Uni et le Canada.
#Cela reflète le poids industriel du cinéma américain et l'importance des productions locales indiennes. »
```

```
#Les États-Unis dominent largement la production de contenus présents sur Netflix, suivis par l'Inde,
#le Royaume-Uni et le Canada.
#Cela reflète le poids industriel du cinéma américain et l'importance des productions locales indiennes. »
```

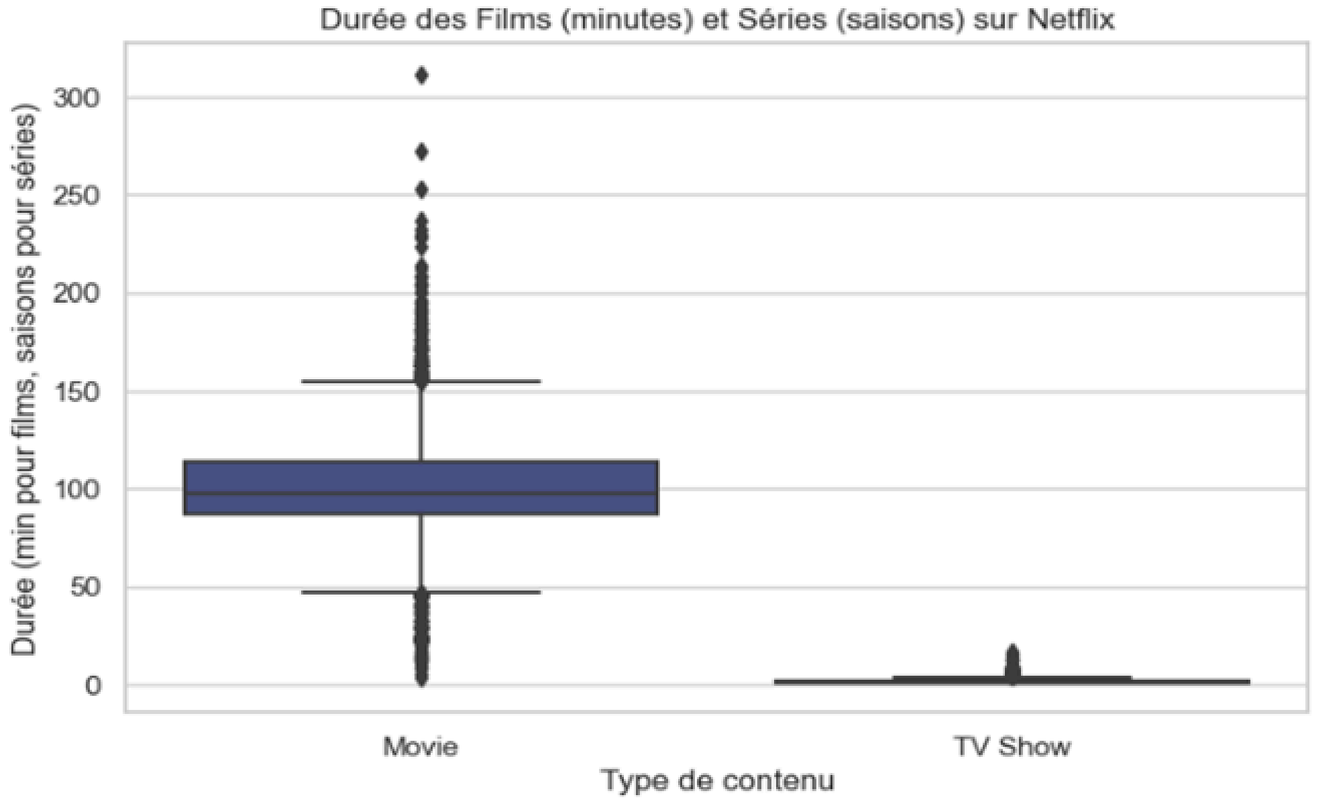
```
#histplot – Distribution des années de sortie (release_year)  / But : analyser l'ancienneté des contenus.
fig_hist_release_year = plt.figure(figsize=(10,5))
sns.histplot(data=df, x="release_year", bins=30, kde=True, color="purple")
plt.title("Distribution des années de sortie des contenus Netflix")
plt.xlabel("Année de sortie")
plt.ylabel("Nombre de contenus")
plt.show()
```

```
C:\Users\micro\anacondaa\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is depre
be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

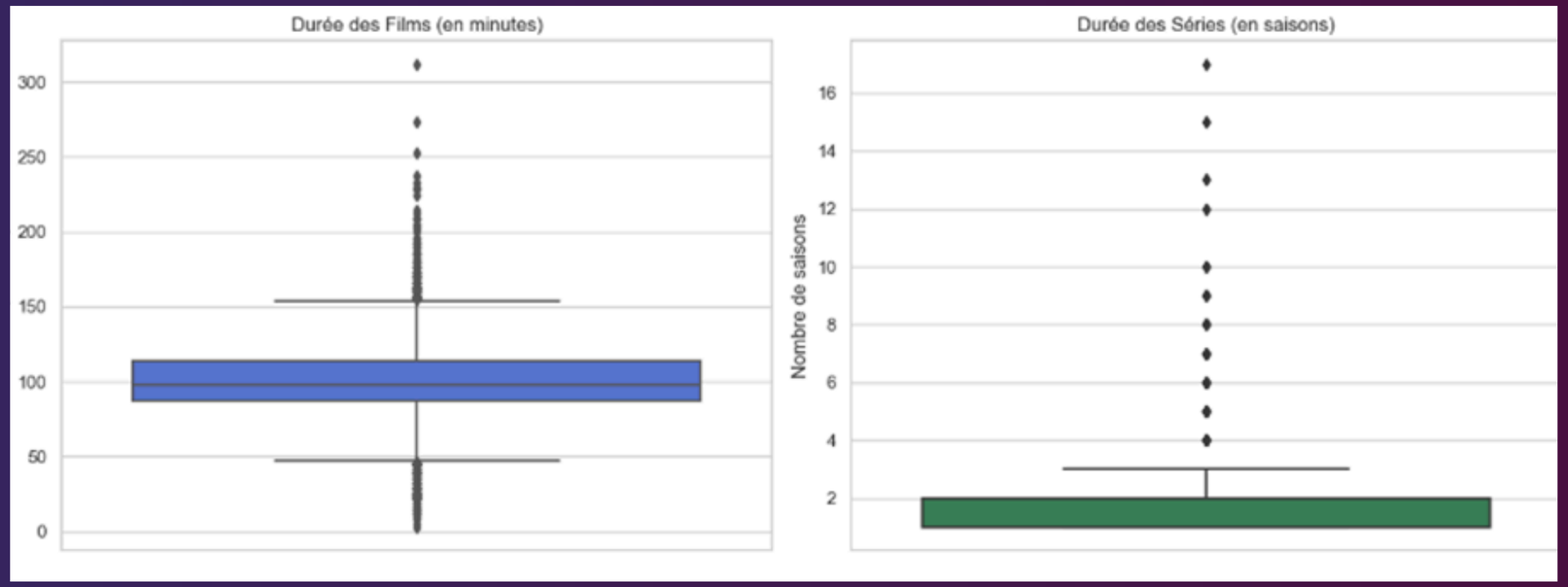



```
#« La majorité des contenus présents sur Netflix ont été produits entre 2000 et 2020.
#On observe une concentration importante dans les années récentes,
#ce qui montre que Netflix propose principalement des contenus modernes. »
```

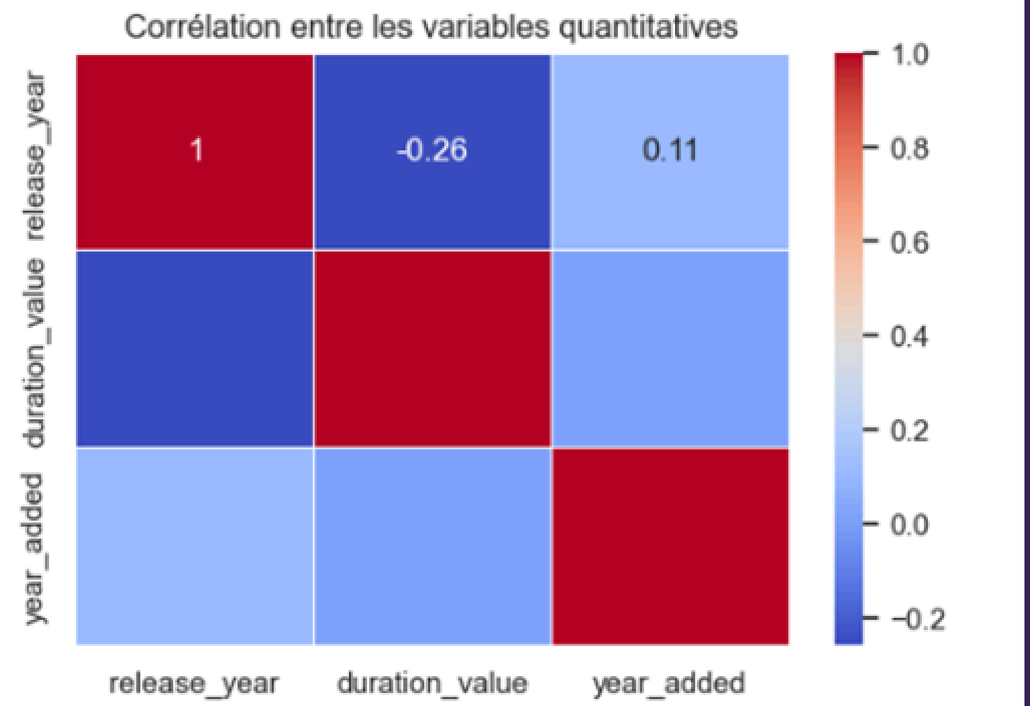
```
plt.figure(figsize=(8,5))

sns.boxplot(data=df, x="type", y="duration_value", palette="mako")

plt.title("Durée des Films (minutes) et Séries (saisons) sur Netflix")
plt.xlabel("Type de contenu")
plt.ylabel("Durée (min pour films, saisons pour séries)")
plt.show()
#1ere methode , difficile a lire puisque le différences entre nb minutes et nbsaisons est très grande
```



```python
#2ème méthode
# 4) Boxplots Films vs Séries
fig_box, axes = plt.subplots(1, 2, figsize=(14,5))
# Boxplot pour les films
sns.boxplot(
    data=df[df["type"]=="Movie"],
    y="duration_value",
    ax=axes[0],
    color="royalblue"
)
axes[0].set_title("Durée des Films (en minutes)")
axes[0].set_ylabel("Durée (minutes)")

# Boxplot pour les séries
sns.boxplot(
    data=df[df["type"]=="TV Show"],
    y="duration_value",
    ax=axes[1],
    color="seagreen"
)
axes[1].set_title("Durée des Séries (en saisons)")
axes[1].set_ylabel("Nombre de saisons")

plt.tight_layout()
plt.show()
```



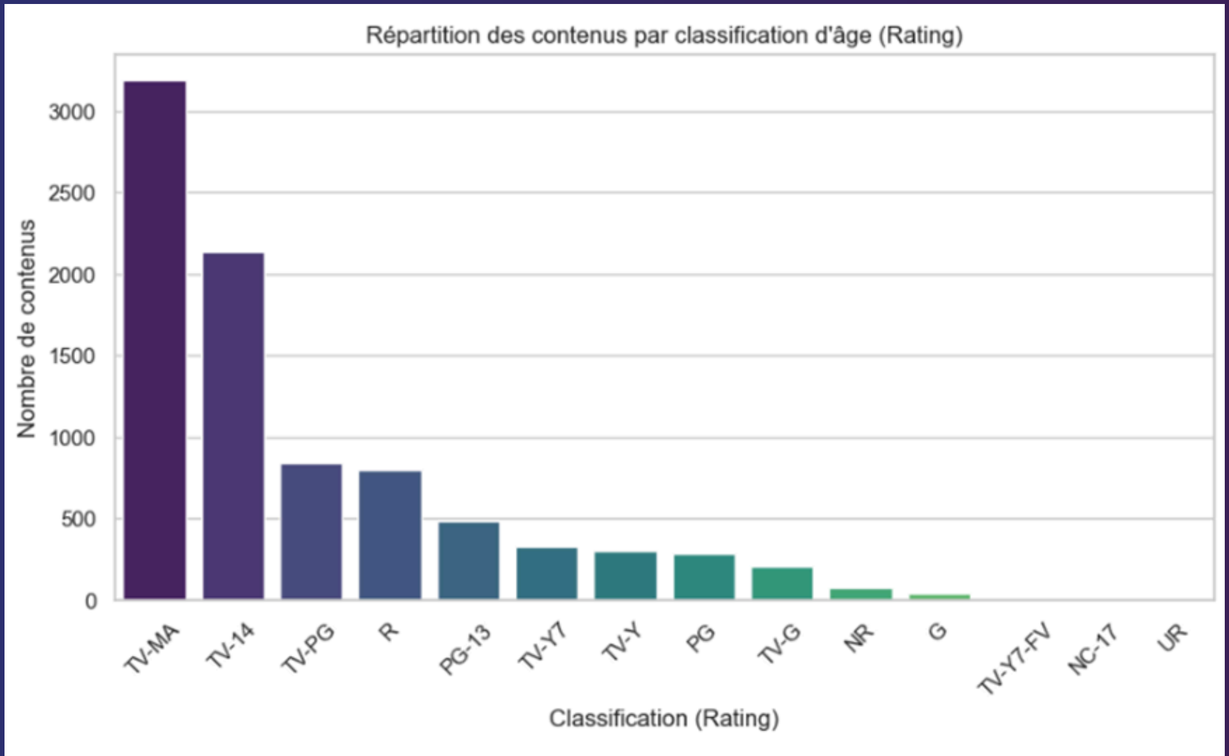
```python
#4 heatmap pour faire la Corrélation entre variables quantitatives
#release_year
#duration_value
#year_added
# Sélectionner uniquement les colonnes numériques
numeric_df = df[["release_year", "duration_value", "year_added"]]
# Calcul de la matrice de corrélation
corr_matrix = numeric_df.corr()
# Afficher la heatmap
fig_heatmap = plt.figure(figsize=(6,4))
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", linewidths=0.5)
plt.title("Corrélation entre les variables quantitatives")
plt.show()
```




```python
#release_year vs duration_value → presque 0

#release_year vs year_added → un peu plus corrélé

#duration_value vs year_added → presque 0
```

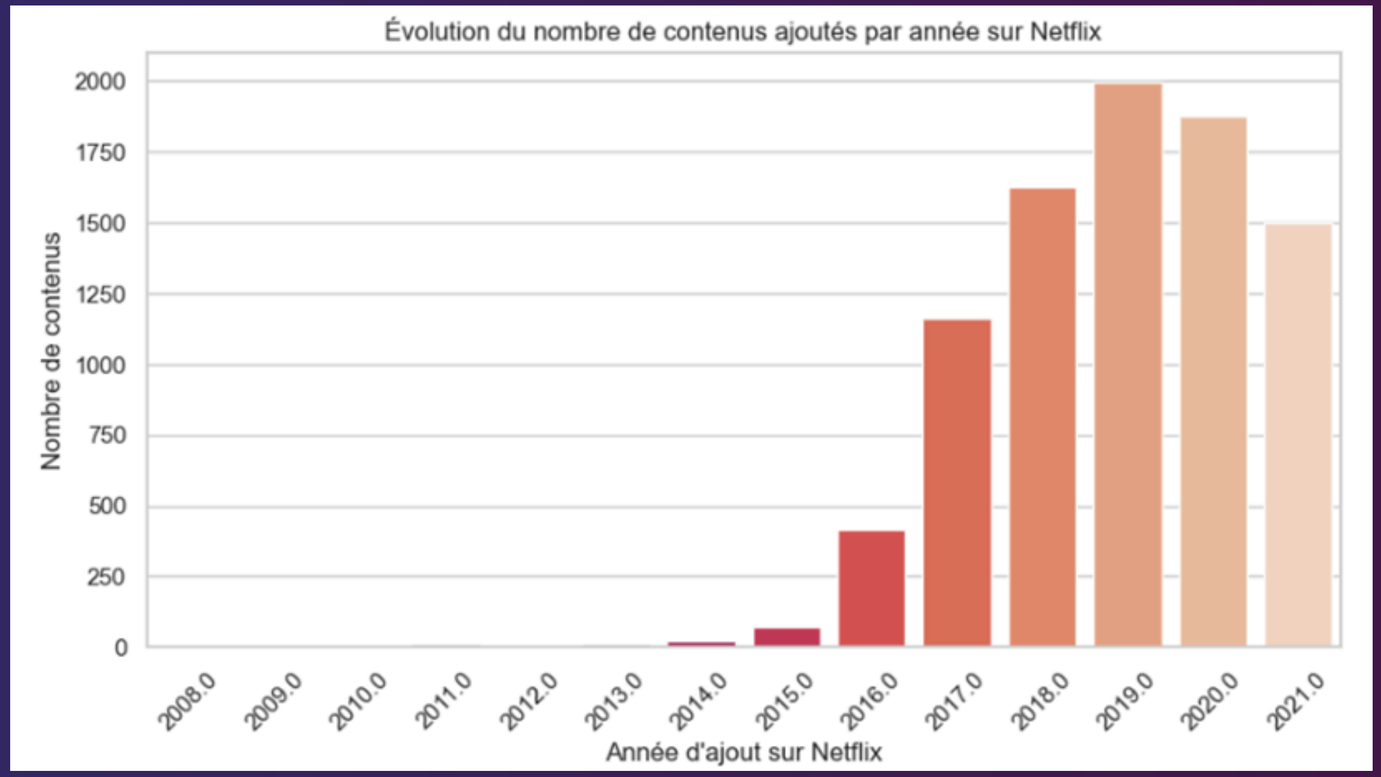
```
#Analyse Additionnelle 1 : Répartition des contenus par Rating (classification d'âge)
#Analyse Additionnelle 2 : Évolution des contenus ajoutés par Année (year_added)
```

```
#Analyse Additionnelle 1 : Répartition des contenus par Rating (classification d'âge)

fig_rating = plt.figure(figsize=(10,5))
sns.countplot(data=df, x="rating", order=df["rating"].value_counts().index, palette="viridis")

plt.title("Répartition des contenus par classification d'âge (Rating)")
plt.xlabel("Classification (Rating)")
plt.ylabel("Nombre de contenus")
plt.xticks(rotation=45)
plt.show()
```




```
#L'analyse de la répartition des ratings montre une forte dominance des classifications TV-MA et TV-14,
#ce qui indique que Netflix cible principalement un public adulte ou adolescent.
#Les classifications jeunes (TV-Y, TV-G) sont largement minoritaires, révélant que le contenu "famille"
#représente une part assez faible dans le catalogue.
#Cette distribution reflète la stratégie de Netflix d'investir davantage dans des séries et films matures,
#plus populaires auprès de son audience principale
#TV-MA Réservé aux adultes (17+)
#TV-14 Public adolescent (14+)
#TV-PG (Parental Guidance) Déconseillé sans supervision parentale
#R (Restricted) Interdit aux moins de 17 ans non accompagnés
#PG-13 Déconseillé aux moins de 13 ans
#TV-Y7 Pour enfants 7+
#TV-Y Pour jeunes enfants (presque tout public)
#TV-G Tout public
#PG Supervision parentale
#NR (Not Rated) Non classé
#NC-17 Strictement adultes
#TV-Y7-FV Pour enfants mais avec scènes de "Fantasy Violence"
#UR (Unrated)Non évalué officiellement
```

```python
#Analyse Additionnelle 2 : Évolution des contenus ajoutés par Année (year_added)
fig_year_added = plt.figure(figsize=(10,5))

sns.countplot(
    data=df,
    x="year_added",
    order=sorted(df["year_added"].dropna().unique()),
    palette="rocket"
)

plt.title("Évolution du nombre de contenus ajoutés par année sur Netflix")
plt.xlabel("Année d'ajout sur Netflix")
plt.ylabel("Nombre de contenus")
plt.xticks(rotation=45)
plt.show()
```

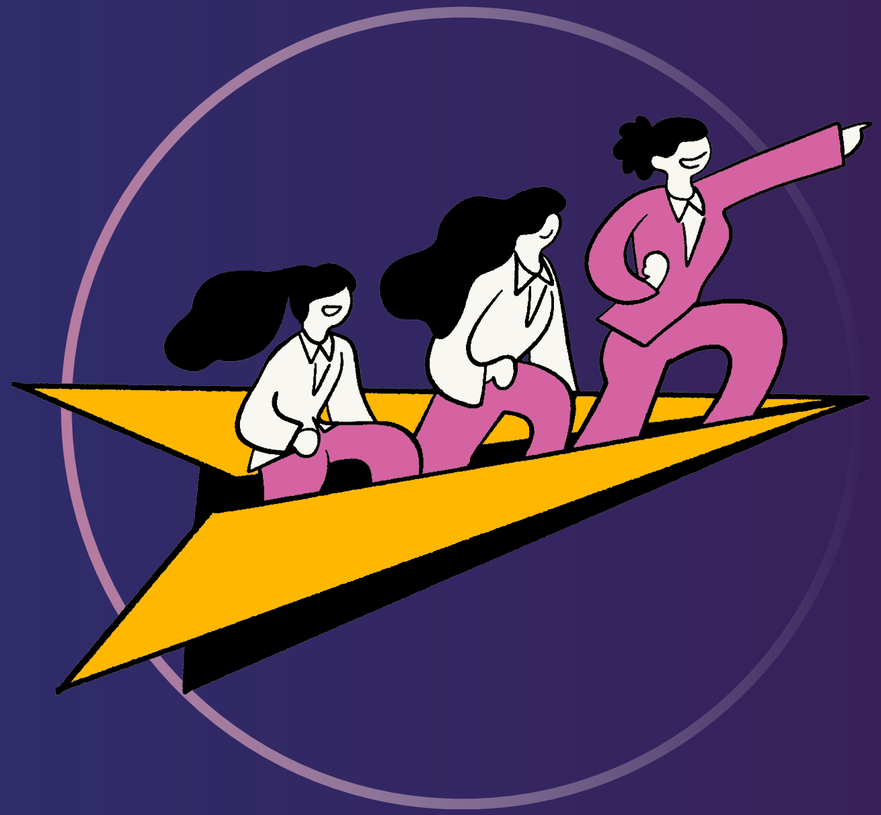


```
# L’évolution du catalogue Netflix montre une croissance spectaculaire à partir de 2016,
#correspondant à l’expansion internationale de la plateforme et à l’investissement massif dans les productions originales.
#Le nombre de contenus ajoutés atteint un pic entre 2019 et 2021, période durant laquelle Netflix a accéléré
#l’acquisition et la production de contenus en réponse à l’augmentation mondiale de la demande pour les services de streaming,
#notamment durant la pandémie.
#Cette tendance confirme la stratégie agressive de Netflix visant à élargir rapidement son catalogue pour attirer
#et fidéliser ses abonnés.
```

PARTIE II

WORLD HAPPINESS REPORT (PLOTLY)

- Présentation du dataset
- Préparation des données multi-annuelles
- Visualisations interactives
- Conclusion–World Happiness Report



PRÉSENTATION DU DATASET

LES DONNÉES PROVIENNENT DU WORLD HAPPINESS REPORT, UNE ÉTUDE INTERNATIONALE PUBLIÉE CHAQUE ANNÉE DEPUIS 2012.
ELLE MESURE LE NIVEAU DE BONHEUR DANS PLUS DE 150 PAYS EN SE BASANT SUR DES INDICATEURS SOCIO-ÉCONOMIQUES TELS QUE LE PIB, LA SANTÉ, LA LIBERTÉ, LA GÉNÉROSITÉ OU ENCORE LA PERCEPTION DE LA CORRUPTION.

STRUCTURE MULTI-ANNUELLE (2015–2019)

DANS CE PROJET, CINQ FICHIERS DISTINCTS ONT ÉTÉ UTILISÉS :

2015.CSV — 158 PAYS
2016.CSV — 157 PAYS
2017.CSV — 155 COUNTRIES
2018.CSV — 156 PAYS
2019.CSV — 156 PAYS

CHALLENGES

Chaque fichier représente une année complète du classement mondial du bonheur.
Cependant, la structure des colonnes change d'une année à l'autre, ce qui a nécessité un travail d'harmonisation

```
import pandas as pd
```

```
df2015 = pd.read_csv("2015.csv")
```

```
df2015.info()
df2015.describe(include="all")
df2015.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 158 entries, 0 to 157
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Country                        158 non-null    object
 1   Region                         158 non-null    object
 2   Happiness Rank                 158 non-null    int64
 3   Happiness Score                158 non-null    float64
 4   Standard Error                 158 non-null    float64
 5   Economy (GDP per Capita)       158 non-null    float64
 6   Family                         158 non-null    float64
 7   Health (Life Expectancy)       158 non-null    float64
 8   Freedom                        158 non-null    float64
 9   Trust (Government Corruption)  158 non-null    float64
 10  Generosity                     158 non-null    float64
 11  Dystopia Residual              158 non-null    float64
dtypes: float64(9), int64(1), object(2)
memory usage: 14.9+ KB

Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Standard Error', 'Economy (GDP per Capita)', 'Family',
       'Health (Life Expectancy)', 'Freedom', 'Trust (Government Corruption)',
       'Generosity', 'Dystopia Residual'],
      dtype='object')
```

```
df2015.head()
```

```
df2015.head()
```

	Country	Region	Happiness Rank	Happiness Score	Standard Error	Economy (GDP per Capita)	Family	Health (Life Expectancy)	Freedom	Trust (Government Corruption)	Generosity	Dystopia Residual
0	Switzerland	Western Europe	1	7.587	0.03411	1.39651	1.34951	0.94143	0.66557	0.41978	0.29678	2.51738
1	Iceland	Western Europe	2	7.561	0.04884	1.30232	1.40223	0.94784	0.62877	0.14145	0.43630	2.70201
2	Denmark	Western Europe	3	7.527	0.03328	1.32548	1.36058	0.87464	0.64938	0.48357	0.34139	2.49204
3	Norway	Western Europe	4	7.522	0.03880	1.45900	1.33095	0.88521	0.66973	0.36503	0.34699	2.46531
4	Canada	North America	5	7.427	0.03553	1.32629	1.32261	0.90563	0.63297	0.32957	0.45811	2.45176

```
df2016 = pd.read_csv("2016.csv")
```

```
df2016.info()
df2016.describe(include="all")
df2016.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 13 columns):
 #   Column                         Non-Null Count  Dtype
---  ------                         --------------  -----
 0   Country                        157 non-null    object
 1   Region                         157 non-null    object
 2   Happiness Rank                 157 non-null    int64
 3   Happiness Score                157 non-null    float64
 4   Lower Confidence Interval      157 non-null    float64
 5   Upper Confidence Interval      157 non-null    float64
 6   Economy (GDP per Capita)       157 non-null    float64
 7   Family                         157 non-null    float64
 8   Health (Life Expectancy)       157 non-null    float64
 9   Freedom                        157 non-null    float64
 10  Trust (Government Corruption)  157 non-null    float64
 11  Generosity                     157 non-null    float64
 12  Dystopia Residual              157 non-null    float64
dtypes: float64(10), int64(1), object(2)
memory usage: 16.1+ KB

Index(['Country', 'Region', 'Happiness Rank', 'Happiness Score',
       'Lower Confidence Interval', 'Upper Confidence Interval',
       'Economy (GDP per Capita)', 'Family', 'Health (Life Expectancy)',
       'Freedom', 'Trust (Government Corruption)', 'Generosity',
       'Dystopia Residual'],
      dtype='object')
```

```
df2017 = pd.read_csv("2017.csv")
```

```
df2017.info()
df2017.describe(include="all")
df2017.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 155 entries, 0 to 154
Data columns (total 12 columns):
 #   Column                         Non-Null Count  Dtype  
---  ------                         --------------  -----  
 0   Country                        155 non-null    object 
 1   Happiness.Rank                 155 non-null    int64  
 2   Happiness.Score                155 non-null    float64
 3   Whisker.high                   155 non-null    float64
 4   Whisker.low                    155 non-null    float64
 5   Economy..GDP.per.Capita.       155 non-null    float64
 6   Family                         155 non-null    float64
 7   Health..Life.Expectancy.       155 non-null    float64
 8   Freedom                        155 non-null    float64
 9   Generosity                     155 non-null    float64
 10  Trust..Government.Corruption.  155 non-null    float64
 11  Dystopia.Residual              155 non-null    float64
dtypes: float64(10), int64(1), object(1)
memory usage: 14.7+ KB

Index(['Country', 'Happiness.Rank', 'Happiness.Score', 'Whisker.high',
       'Whisker.low', 'Economy..GDP.per.Capita.', 'Family',
       'Health..Life.Expectancy.', 'Freedom', 'Generosity',
       'Trust..Government.Corruption.', 'Dystopia.Residual'],
      dtype='object')
```

```
df2018 = pd.read_csv("2018.csv")
```

```
df2018.info()
df2018.describe(include="all")
df2018.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Overall rank                  156 non-null    int64  
 1   Country or region             156 non-null    object 
 2   Score                         156 non-null    float64
 3   GDP per capita                156 non-null    float64
 4   Social support                156 non-null    float64
 5   Healthy life expectancy       156 non-null    float64
 6   Freedom to make life choices  156 non-null    float64
 7   Generosity                    156 non-null    float64
 8   Perceptions of corruption     155 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 11.1+ KB

Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

LES FICHIERS 2015–2017 CONTIENNENT :
HAPPINESS RANK, STANDARD ERROR, CONFIDENCE INTERVALS, DYSTOPIA RESIDUAL
QUI NE SONT PAS NÉCESSAIRES POUR UNE ANALYSE VISUELLE.

```
df2019 = pd.read_csv("2019.csv")
```

```
df2019.info()
df2019.describe(include="all")
df2019.columns
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 156 entries, 0 to 155
Data columns (total 9 columns):
 #   Column                        Non-Null Count  Dtype  
---  ------                        --------------  -----  
 0   Overall rank                  156 non-null    int64  
 1   Country or region             156 non-null    object 
 2   Score                         156 non-null    float64
 3   GDP per capita                156 non-null    float64
 4   Social support                156 non-null    float64
 5   Healthy life expectancy       156 non-null    float64
 6   Freedom to make life choices  156 non-null    float64
 7   Generosity                    156 non-null    float64
 8   Perceptions of corruption     156 non-null    float64
dtypes: float64(7), int64(1), object(1)
memory usage: 11.1+ KB

Index(['Overall rank', 'Country or region', 'Score', 'GDP per capita',
       'Social support', 'Healthy life expectancy',
       'Freedom to make life choices', 'Generosity',
       'Perceptions of corruption'],
      dtype='object')
```

LES FICHIERS 2018–2019 ONT UNE STRUCTURE MODERNISÉE, AVEC DES NOMS DIFFÉRENTS COMME :
GDP PER CAPITA
SOCIAL SUPPORT
HEALTHY LIFE EXPECTANCY
PERCEPTIONS OF CORRUPTION

HARMONISATION DES COLONNES

POUR GARANTIR UNE ANALYSE COHÉRENTE, CHAQUE DATASET A ÉTÉ :

- **FILTRÉ** POUR NE GARDER QUE LES INDICATEURS UTILES
- **RENOMMÉ** POUR STANDARDISER LES NOMS
- **NETTOYÉ** (SUPPRESSION DES COLONNES TECHNIQUES INUTILES)
- **ÉTIQUETÉ** AVEC UNE COLONNE **YEAR**
- **FUSIONNÉ EN UN SEUL** DATAFRAME FINAL

```python
#Harmonisation 2015
df2015_clean = df2015[[
    "Country",
    "Region",
    "Happiness Score",
    "Economy (GDP per Capita)",
    "Family",
    "Health (Life Expectancy)",
    "Freedom",
    "Generosity",
    "Trust (Government Corruption)"
]].copy()
```

```python
df2015_clean["year"] = 2015

df2015_clean = df2015_clean.rename(columns={
    "Country": "country",
    "Region": "region",
    "Happiness Score": "happiness_score",
    "Economy (GDP per Capita)": "gdp_per_capita",
    "Family": "social_support",
    "Health (Life Expectancy)": "life_expectancy",
    "Freedom": "freedom",
    "Generosity": "generosity",
    "Trust (Government Corruption)": "corruption"
})
```

```python
#Harmonisation 2016
df2016_clean = df2016[[
    "Country",
    "Region",
    "Happiness Score",
    "Economy (GDP per Capita)",
    "Family",
    "Health (Life Expectancy)",
    "Freedom",
    "Generosity",
    "Trust (Government Corruption)"
]].copy()

df2016_clean["year"] = 2016

df2016_clean = df2016_clean.rename(columns={
    "Country": "country",
    "Region": "region",
    "Happiness Score": "happiness_score",
    "Economy (GDP per Capita)": "gdp_per_capita",
    "Family": "social_support",
    "Health (Life Expectancy)": "life_expectancy",
    "Freedom": "freedom",
    "Generosity": "generosity",
    "Trust (Government Corruption)": "corruption"
})
```

```python
df2017_clean = df2017[[
    "Country",
    "Happiness.Score",
    "Economy..GDP.per.Capita.",
    "Family",
    "Health..Life.Expectancy.",
    "Freedom",
    "Generosity",
    "Trust..Government.Corruption."
]].copy()

df2017_clean["year"] = 2017
# On ajoute la colonne region (absente en 2017, donc laissée à None)
df2017_clean["region"] = None

df2017_clean = df2017_clean.rename(columns={
    "Country": "country",
    "Happiness.Score": "happiness_score",
    "Economy..GDP.per.Capita.": "gdp_per_capita",
    "Family": "social_support",
    "Health..Life.Expectancy.": "life_expectancy",
    "Freedom": "freedom",
    "Generosity": "generosity",
    "Trust..Government.Corruption.": "corruption"
})

df2017_clean = df2017_clean[[
    "country", "region", "happiness_score", "gdp_per_capita",
    "social_support", "life_expectancy", "freedom",
    "generosity", "corruption", "year"
]]
```

POUR GARANTIR UNE ANALYSE COHÉRENTE, CHAQUE DATASET A ÉTÉ :

- **FILTRÉ** POUR NE GARDER QUE LES INDICATEURS UTILES
- **RENOMMÉ** POUR STANDARDISER LES NOMS
- **NETTOYÉ** (SUPPRESSION DES COLONNES TECHNIQUES INUTILES)
- **ÉTIQUETÉ** AVEC UNE COLONNE **YEAR**
- **FUSIONNÉ EN UN SEUL** DATAFRAME FINAL

```python
df2018_clean = df2018[[
    "Country or region",
    "Score",
    "GDP per capita",
    "Social support",
    "Healthy life expectancy",
    "Freedom to make life choices",
    "Generosity",
    "Perceptions of corruption"
]].copy()

df2018_clean["year"] = 2018
df2018_clean["region"] = None

df2018_clean = df2018_clean.rename(columns={
    "Country or region": "country",
    "Score": "happiness_score",
    "GDP per capita": "gdp_per_capita",
    "Social support": "social_support",
    "Healthy life expectancy": "life_expectancy",
    "Freedom to make life choices": "freedom",
    "Generosity": "generosity",
    "Perceptions of corruption": "corruption"
})
```

```python
df2019_clean = df2019.copy()

df2019_clean["year"] = 2019
df2019_clean["region"] = None

df2019_clean = df2019_clean.rename(columns={
    "Country or region": "country",
    "Score": "happiness_score",
    "GDP per capita": "gdp_per_capita",
    "Social support": "social_support",
    "Healthy life expectancy": "life_expectancy",
    "Freedom to make life choices": "freedom",
    "Generosity": "generosity",
    "Perceptions of corruption": "corruption"
})

df2019_clean = df2019_clean[[
    "country", "region", "happiness_score", "gdp_per_capita",
    "social_support", "life_expectancy", "freedom",
    "generosity", "corruption", "year"
]]
```

```
df_final = pd.concat([
    df2015_clean,
    df2016_clean,
    df2017_clean,
    df2018_clean,
    df2019_clean
], ignore_index=True)
```

```
df_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 782 entries, 0 to 781
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   country          782 non-null    object 
 1   region           315 non-null    object 
 2   happiness_score  782 non-null    float64
 3   gdp_per_capita   782 non-null    float64
 4   social_support   782 non-null    float64
 5   life_expectancy  782 non-null    float64
 6   freedom          782 non-null    float64
 7   generosity       782 non-null    float64
 8   corruption       781 non-null    float64
 9   year             782 non-null    int64  
dtypes: float64(7), int64(1), object(2)
memory usage: 61.2+ KB
```

```
df_final.head()
```

	country	region	happiness_score	gdp_per_capita	social_support	life_expectancy	freedom	generosity	corruption	year
0	Switzerland	Western Europe	7.587	1.39651	1.34951	0.94143	0.66557	0.29678	0.41978	2015
1	Iceland	Western Europe	7.561	1.30232	1.40223	0.94784	0.62877	0.43630	0.14145	2015
2	Denmark	Western Europe	7.527	1.32548	1.36058	0.87464	0.64938	0.34139	0.48357	2015
3	Norway	Western Europe	7.522	1.45900	1.33095	0.88521	0.66973	0.34699	0.36503	2015
4	Canada	North America	7.427	1.32629	1.32261	0.90563	0.63297	0.45811	0.32957	2015

```
df_final["year"].unique()
```

```
array([2015, 2016, 2017, 2018, 2019], dtype=int64)
```

```
df_final.isna().sum()
```

```
country              0
region             467
happiness_score      0
gdp_per_capita       0
social_support       0
life_expectancy      0
freedom              0
generosity           0
corruption           1
year                 0
dtype: int64
```

```
#des NaN dans region pour 2017/2018/2019 : normal, on a choisi de les laisser
```

CE SONT LES VARIABLES LES PLUS ADAPTÉES POUR RÉALISER :

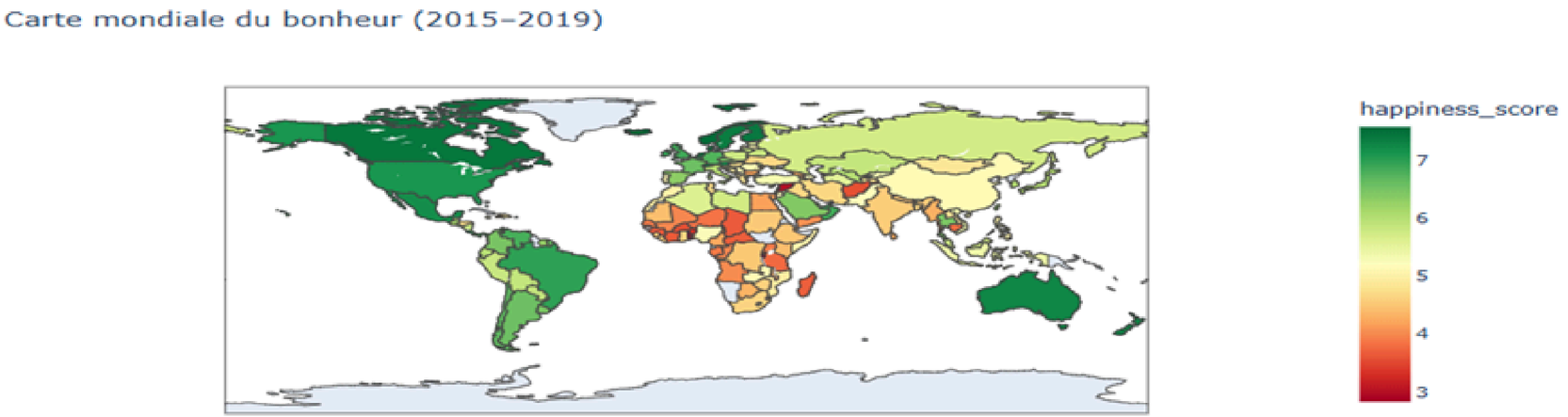
- **UNE CARTE CHOROPLÈTHE DU BONHEUR,**
- **UN SCATTER INTERACTIF PIB VS SCORE,**
- **UNE HEATMAP DE CORRÉLATION,**
- **UNE COURBE D'ÉVOLUTION DU BONHEUR DANS LE TEMPS.**

LES VARIABLES INUTILES OU TECHNIQUES ONT ÉTÉ SUPPRIMÉES (EX. : HAPPINESS RANK, CONFIDENCE INTERVALS, WHISKERS, DYSTOPIA RESIDUAL).

VISUALISATIONS INTERACTIVES

```
#Carte mondiale du bonheur (choropleth)
#Objectif : Répondre à la question « Qui sont les pays les plus heureux ? »
```

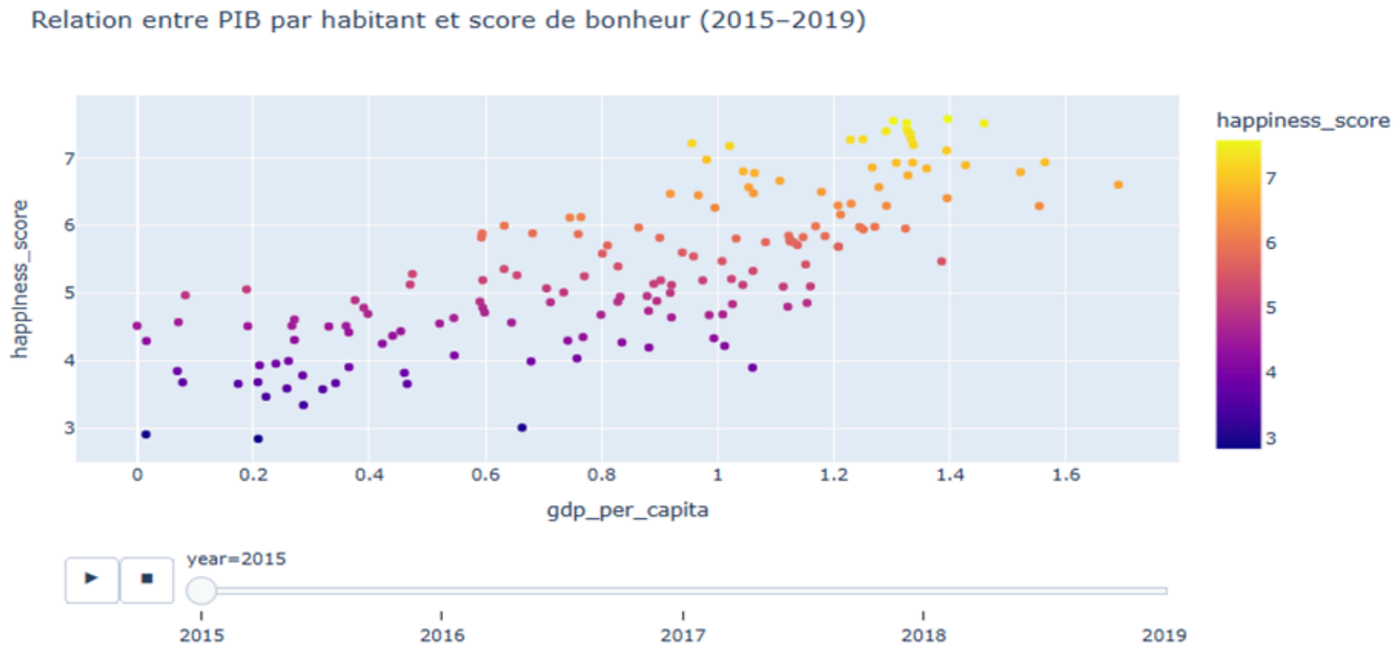
```
import plotly.express as px

fig_choropleth = px.choropleth(
    df_final,
    locations="country",
    locationmode="country names",
    color="happiness_score",
    hover_name="country",
    animation_frame="year",    #slider interactif suivant l'année
    color_continuous_scale="RdYlGn",  #vert=plus heureux,rouge=moins heureux
    title="Carte mondiale du bonheur (2015-2019)"
)

fig_choropleth.show()
```



LA CARTE ANIMÉE MET EN ÉVIDENCE DE FORTES DISPARITÉS GÉOGRAPHIQUES DU NIVEAU DE BONHEUR ENTRE 2015 ET 2019.
LES PAYS NORDIQUES ET D'EUROPE DE L'OUEST, AINSI QUE QUELQUES PAYS D'AMÉRIQUE DU NORD ET D'OCÉANIE, AFFICHENT LES SCORES LES PLUS ÉLEVÉS. À L'INVERSE, UNE GRANDE PARTIE DE L'AFRIQUE SUBSAHARIENNE ET CERTAINS PAYS EN CONFLIT RESTENT PARMI LES MOINS BIEN CLASSÉS. GLOBALEMENT, LA CARTE MONTRE UNE STABILITÉ DES ZONES "HEUREUSES" ET "MOINS HEUREUSES" AU FIL DES ANNÉES.

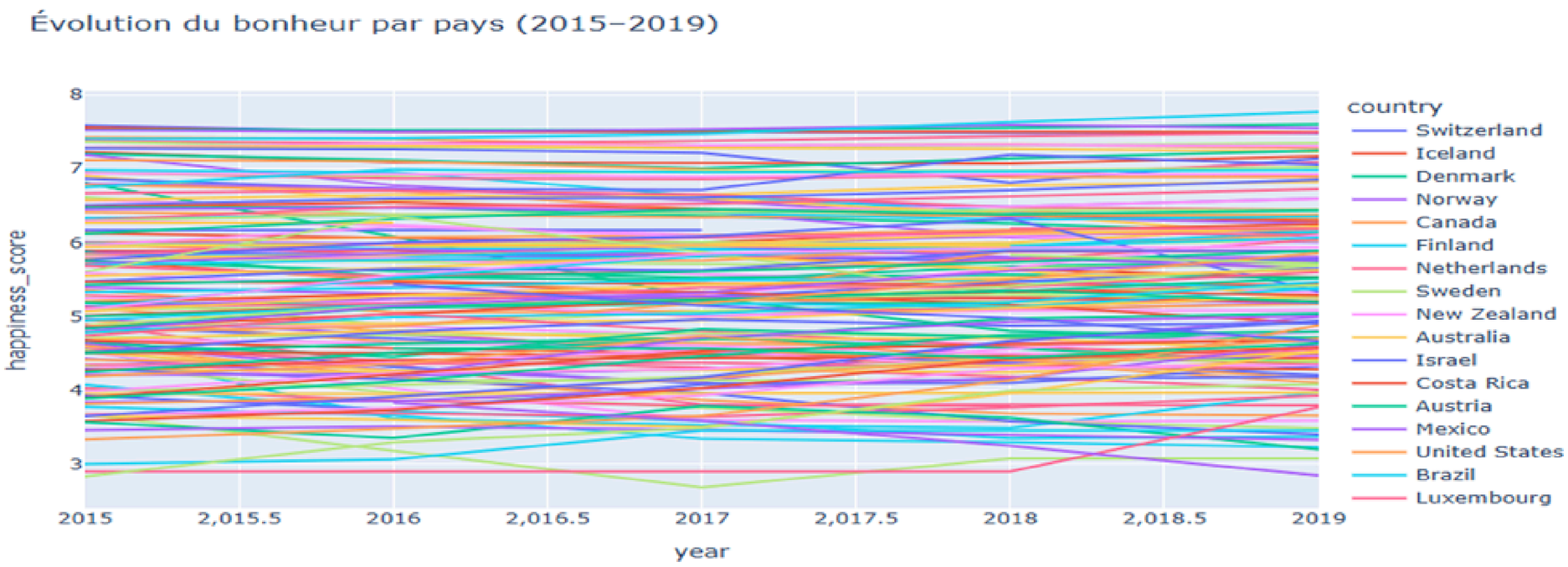
```
#Scatter GDP vs Happiness (relation PIB-bonheur)
#Objectif:Lien entre richesse économique et bien-être.
```

```
df_scatter = df_final.dropna(subset=["gdp_per_capita", "happiness_score"])

fig_scatter = px.scatter(
    df_scatter,
    x="gdp_per_capita",          # PIB par habitant
    y="happiness_score",         # score de bonheur
    color="happiness_score",     # couleur = niveau de bonheur
    hover_name="country",
    animation_frame="year",      # voir l'évolution dans le temps
    title="Relation entre PIB par habitant et score de bonheur (2015-2019)"
)

fig_scatter.show()
```



CEPENDANT, LA RELATION N'EST PAS LINÉAIRE : AU-DELÀ D'UN CERTAIN NIVEAU DE PIB, LE GAIN DE BONHEUR DEVIENT PLUS LIMITÉ, CE QUI SUGGÈRE UN EFFET DE SATURATION. ON OBSERVE AUSSI QUE CERTAINS PAYS, MALGRÉ UN PIB MODESTE, OBTIENNENT UN SCORE DE BONHEUR CORRECT, CE QUI SOULIGNE LE RÔLE D'AUTRES FACTEURS SOCIAUX.

```python
#Courbe temporelle - line()
#Évolution temporelle
fig_line_all = px.line(
    df_final,
    x="year",
    y="happiness_score",
    color="country",            # une courbe par pays
    line_group="country",
    title="Évolution du bonheur par pays (2015-2019)"
)

fig_line_all.show()
```

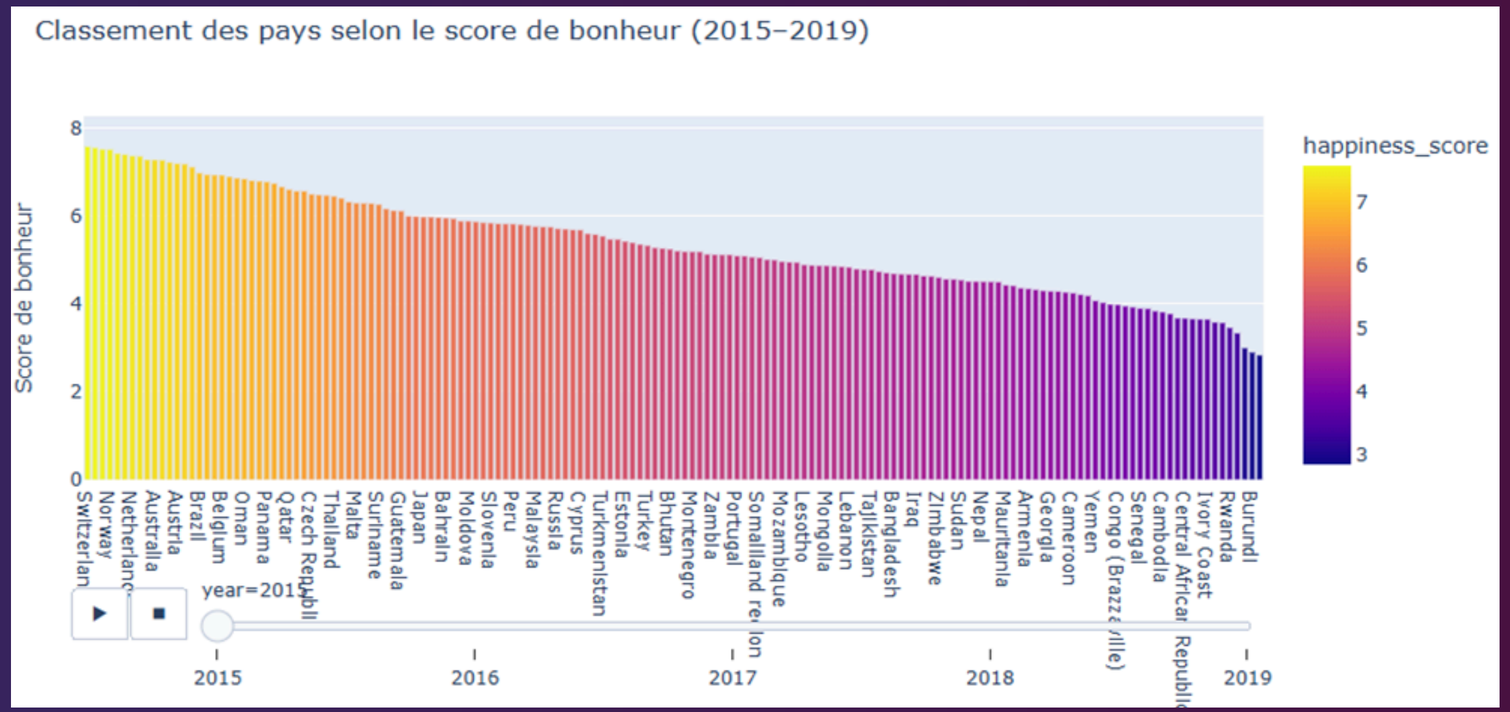


LE GRAPHIQUE EN LIGNES, AVEC UNE COURBE PAR PAYS, MONTRE QUE LA MAJORITÉ DES PAYS PRÉSENTENT DES SCORES RELATIVEMENT STABLES ENTRE 2015 ET 2019.

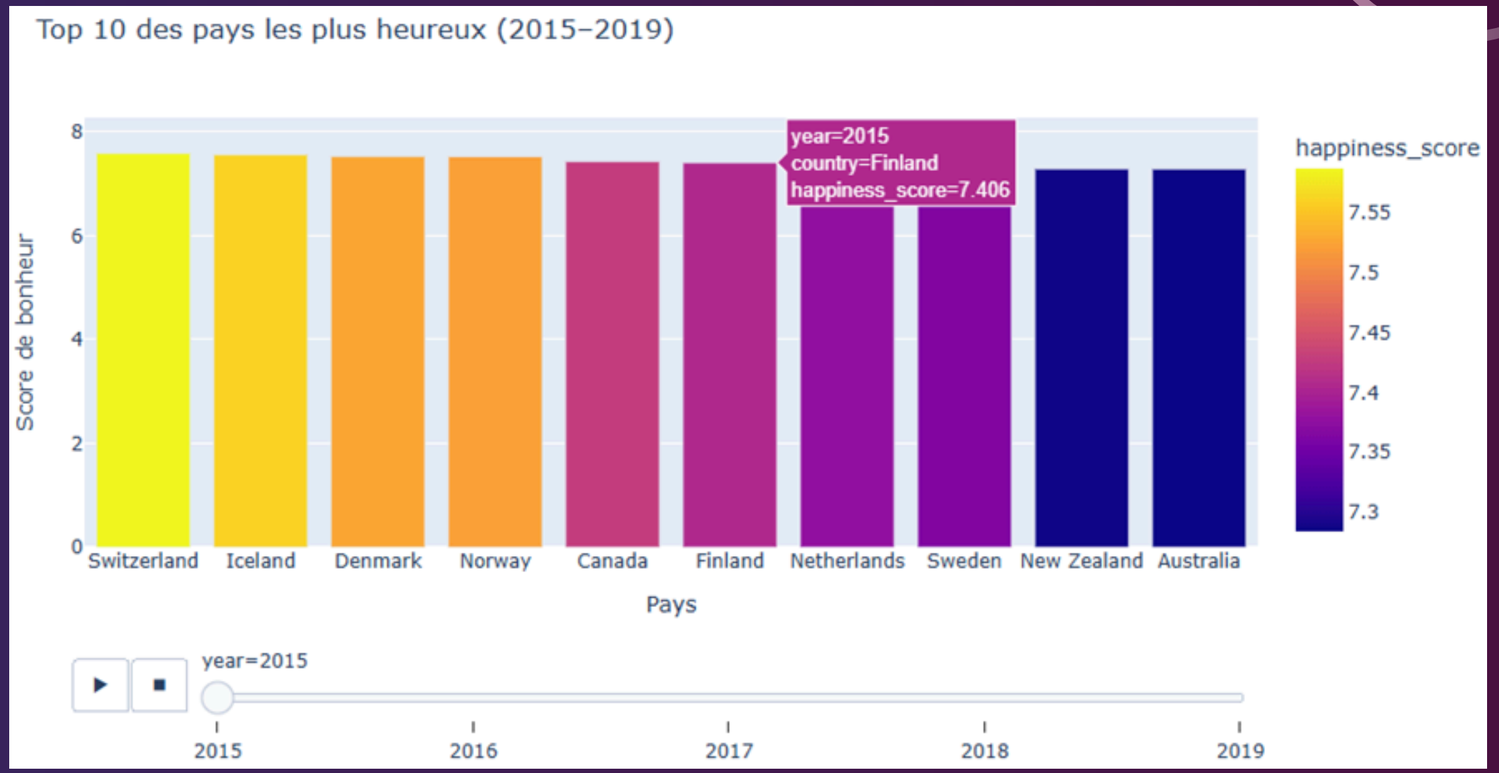
QUELQUES PAYS CONNAISSENT DE LÉGÈRES AMÉLIORATIONS OU DÉGRADATIONS, MAIS IL N'Y A PAS DE BOULEVERSEMENT MASSIF DU CLASSEMENT MONDIAL. CELA CONFIRME QUE LE BONHEUR NATIONAL CHANGE LENTEMENT, ET DÉPEND DE FACTEURS STRUCTURELS (NIVEAU DE VIE, SANTÉ, INSTITUTIONS, ETC.) PLUTÔT QUE DE VARIATIONS PONCTUELLES.

```python
# On sélectionne une seule année à la fois automatiquement grâce à animation_frame
df_bar = df_final.copy()

# Optionnel : trier dans chaque frame
df_bar = df_bar.sort_values(["year", "happiness_score"], ascending=[True, False])

fig_bar_animated = px.bar(
    df_bar,
    x="country",
    y="happiness_score",
    color="happiness_score",
    animation_frame="year",
    title="Classement des pays selon le score de bonheur (2015-2019)",
    range_y=[0, df_final["happiness_score"].max() + 0.5]
)

# Amélioration esthétique
fig_bar_animated.update_layout(
    xaxis_title="Pays",
    yaxis_title="Score de bonheur",
    xaxis={'categoryorder':'total descending'}  # garde l'ordre dans chaque frame
)

fig_bar_animated.show()
```

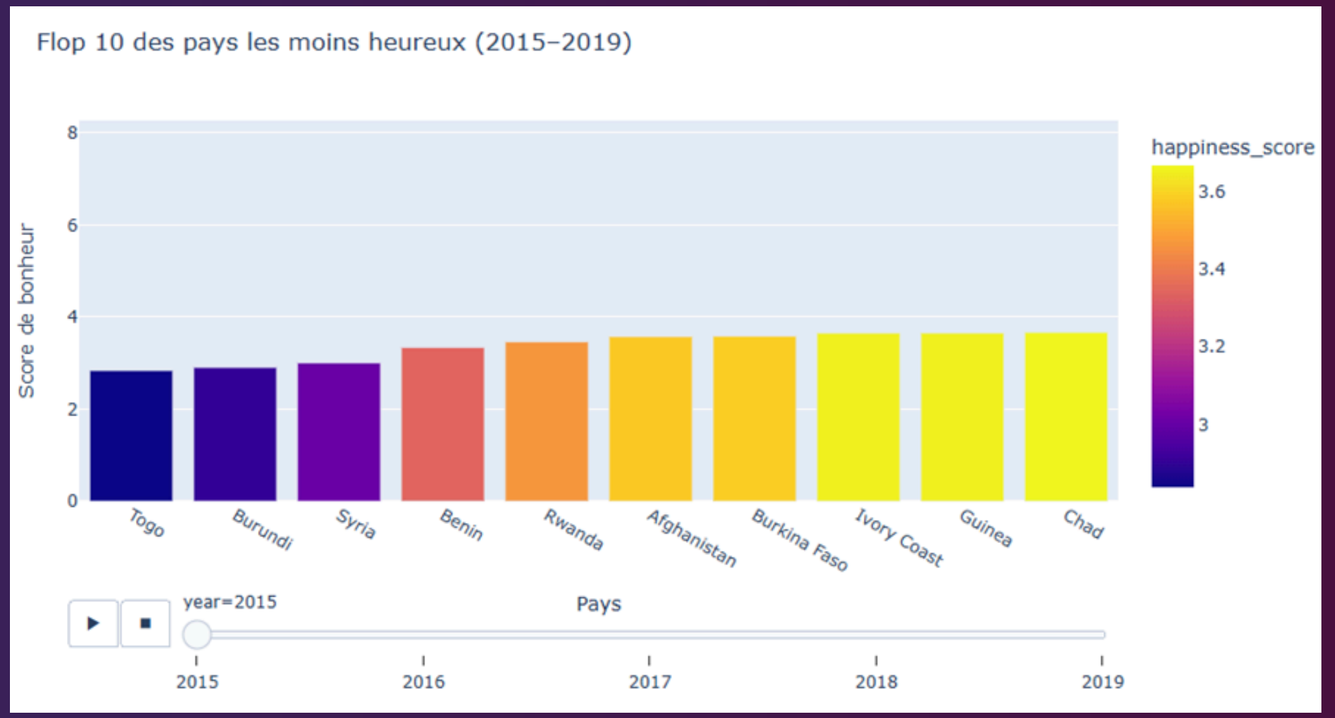
L'HISTOGRAMME ANIMÉ QUI CLASSE LES PAYS SELON LEUR SCORE DE BONHEUR CONFIRME LA DOMINATION RÉCURRENTE D'UN NOYAU DE PAYS TRÈS BIEN CLASSÉS SUR TOUTE LA PÉRIODE.
ON RETROUVE RÉGULIÈREMENT LES MÊMES PAYS EN HAUT DU CLASSEMENT, CE QUI TRADUIT UNE STABILITÉ DES "PAYS LES PLUS HEUREUX". LE GRAPHIQUE PERMET ÉGALEMENT DE VISUALISER LA LONGUE "TRAÎNE" DE PAYS AVEC DES SCORES PLUS FAIBLES, ILLUSTRANT UN ÉCART IMPORTANT ENTRE LE HAUT ET LE BAS DU CLASSEMENT MONDIAL.

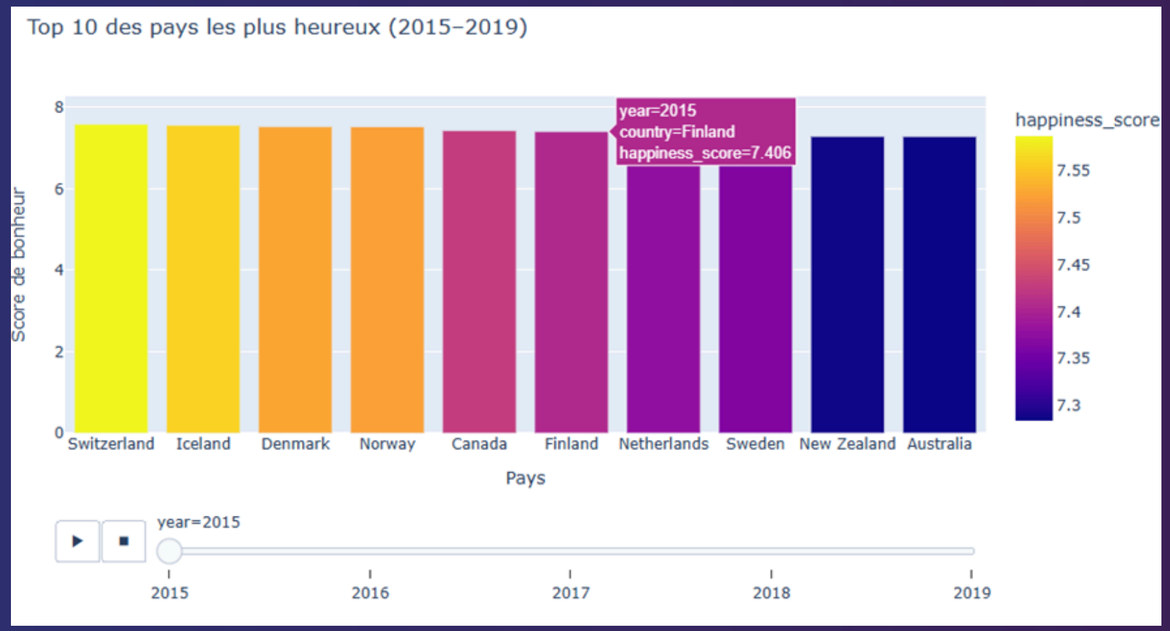


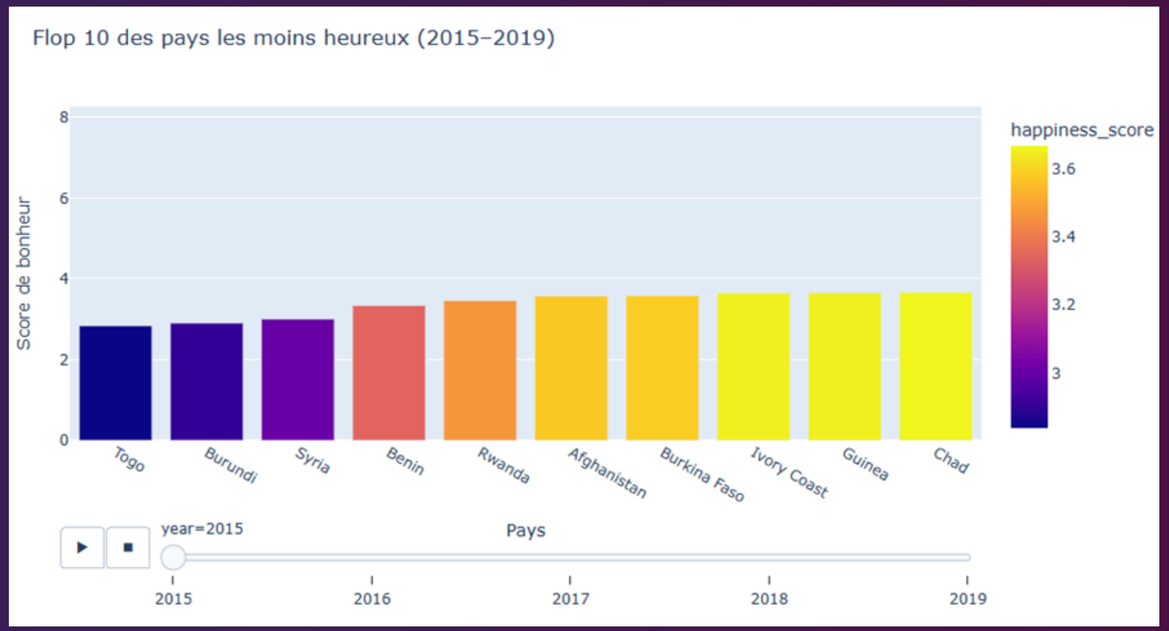
```python
import plotly.express as px

# Top 10 par année (une seule ligne de logique)
df_top = (
    df_final
    .sort_values(["year", "happiness_score"], ascending=[True, False])
    .groupby("year")
    .head(10)          # garde les 10 premiers par année
)

fig_top = px.bar(
    df_top,
    x="country",
    y="happiness_score",
    color="happiness_score",
    animation_frame="year",
    title="Top 10 des pays les plus heureux (2015-2019)",
    range_y=[0, df_final["happiness_score"].max() + 0.5]
)

fig_top.update_layout(
    xaxis_title="Pays",
    yaxis_title="Score de bonheur",
    xaxis={"categoryorder": "total descending"}
)

fig_top.show()
```



```python
# Flop 10 par année
df_flop = (
    df_final
    .sort_values(["year", "happiness_score"], ascending=[True, True])
    .groupby("year")
    .head(10)      # ici, comme on a trié du plus petit au plus grand, head(10) = 10 moins heureux
)

fig_flop = px.bar(
    df_flop,
    x="country",
    y="happiness_score",
    color="happiness_score",
    animation_frame="year",
    title="Flop 10 des pays les moins heureux (2015-2019)",
    range_y=[0, df_final["happiness_score"].max() + 0.5]
)

fig_flop.update_layout(
    xaxis_title="Pays",
    yaxis_title="Score de bonheur",
    xaxis={"categoryorder": "total ascending"}
)

fig_flop.show()
```

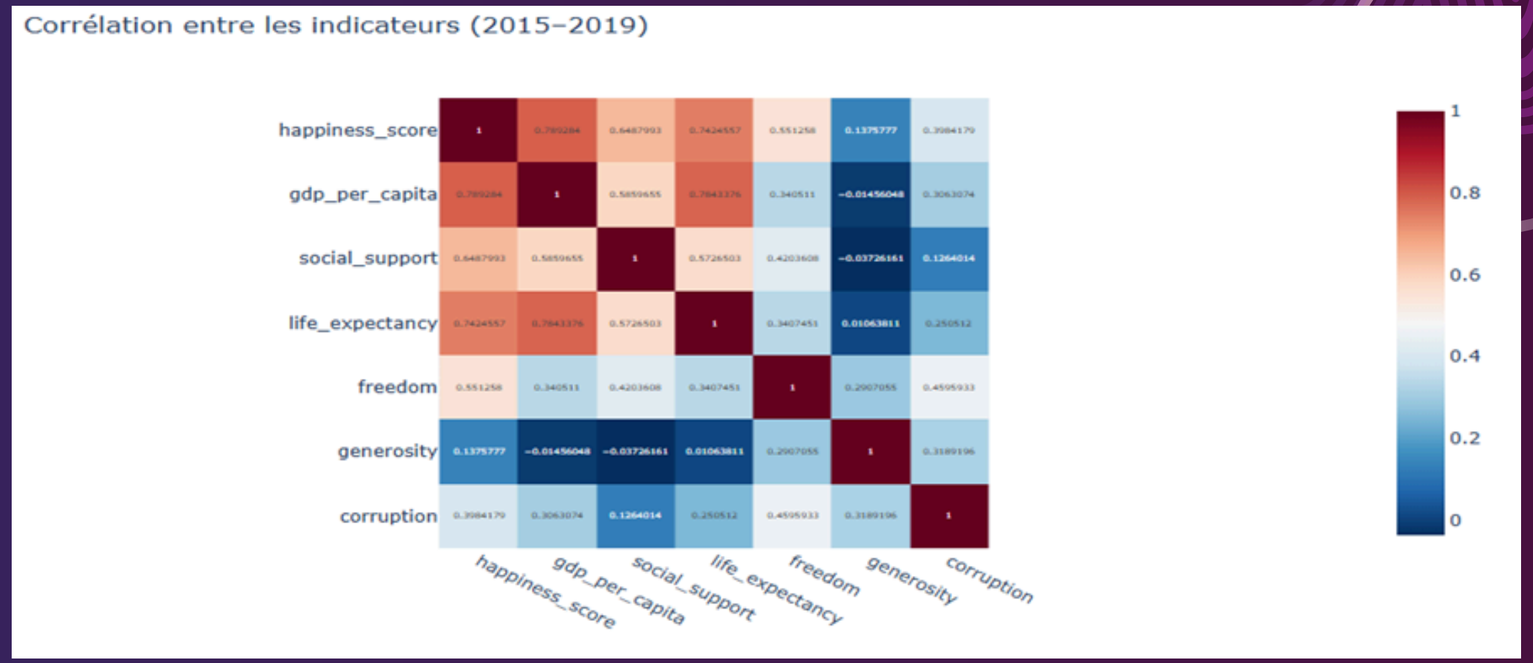






LES GRAPHIQUES “TOP 10” ET “FLOP 10” ISOLENT LES PAYS LES PLUS HEUREUX ET LES MOINS HEUREUX CHAQUE ANNÉE.
LE TOP 10 EST DOMINÉ PAR LES PAYS NORDIQUES ET QUELQUES PAYS DÉVELOPPÉS (EUROPE DE L’OUEST, AMÉRIQUE DU NORD, OCÉANIE), AVEC DES SCORES PROCHES DE 7,5–8.
LE FLOP 10 REGROUPE SURTOUT DES PAYS À FAIBLE REVENU ET/OU EN SITUATION DE CONFLITS OU D’INSTABILITÉ POLITIQUE, AVEC DES SCORES AUTOUR DE 3. CES VISUALISATIONS RENFORCENT L’IDÉE D’UN ÉCART MARQUÉ ET PERSISTANT ENTRE LE HAUT ET LE BAS DU CLASSEMENT MONDIAL.

```python
num_cols = [
    "happiness_score",
    "gdp_per_capita",
    "social_support",
    "life_expectancy",
    "freedom",
    "generosity",
    "corruption"
]

corr_all = df_final[num_cols].corr()

fig_heat_all = px.imshow(
    corr_all,
    text_auto=True,
    color_continuous_scale="RdBu_r",
    title="Corrélation entre les indicateurs (2015–2019)"
)
fig_heat_all.show()
```



- **PIB PAR HABITANT (GDP_PER_CAPITA) :** CORRÉLATION ÉLEVÉE → LA RICHESSE ÉCONOMIQUE EST UN DÉTERMINANT MAJEUR DU BIEN-ÊTRE MOYEN.
- **SOUTIEN SOCIAL (SOCIAL_SUPPORT) ET ESPÉRANCE DE VIE EN BONNE SANTÉ (LIFE_EXPECTANCY) :** CORRÉLATIONS FORTES → LES PAYS OÙ LES HABITANTS SONT BIEN ENTOURÉS ET EN BONNE SANTÉ SONT NETTEMENT PLUS HEUREUX.
- **LIBERTÉ DE FAIRE DES CHOIX DE VIE (FREEDOM) :** CORRÉLATION POSITIVE SIGNIFICATIVE → LA LIBERTÉ PERÇUE CONTRIBUE DIRECTEMENT AU BONHEUR.
- **GÉNÉROSITÉ (GENEROSITY) :** CORRÉLATION PLUS FAIBLE, MAIS POSITIVE → LA GÉNÉROSITÉ JOUE UN RÔLE, MAIS PLUS MARGINAL QUE LE PIB OU LA SANTÉ.
- **PERCEPTION DE CORRUPTION (CORRUPTION) :** CORRÉLATION POSITIVE (PLUS LE SCORE EST ÉLEVÉ, MOINS LA CORRUPTION EST PERÇUE) → LES PAYS OÙ LES INSTITUTIONS SONT JUGÉES MOINS CORROMPUES TENDENT À ÊTRE PLUS HEUREUX.

MINI-DASHBOARD HTML

```
fig_films_series.savefig("netflix_1_films_series.png", bbox_inches="tight")
fig_top10.savefig("netflix_2_top10_pays.png", bbox_inches="tight")
fig_hist_release_year.savefig("netflix_3_hist_release_year.png", bbox_inches="tight")
fig_box.savefig("netflix_4_boxplots_duree.png", bbox_inches="tight")
fig_heatmap.savefig("netflix_5_heatmap_corr.png", bbox_inches="tight")
fig_rating.savefig("netflix_6_rating.png", bbox_inches="tight")
fig_year_added.savefig("netflix_7_year_added.png", bbox_inches="tight")
```

```
# 2) Créer un dashboard HTML simple pour Netflix

images = [
    ("Comparaison Films vs Séries", "netflix_1_films_series.png"),
    ("Top 10 des pays producteurs", "netflix_2_top10_pays.png"),
    ("Distribution des années de sortie", "netflix_3_hist_release_year.png"),
    ("Durée des films et des séries (boxplots)", "netflix_4_boxplots_duree.png"),
    ("Corrélation entre variables quantitatives", "netflix_5_heatmap_corr.png"),
    ("Répartition des contenus par Rating", "netflix_6_rating.png"),
    ("Évolution des contenus ajoutés par année", "netflix_7_year_added.png"),
]

html_parts = []

for title, filename in images:
    html_parts.append(f"<h2>{title}</h2>")
    html_parts.append(f"<img src='{filename}' style='max-width:100%; height:auto;'>")
    html_parts.append("<hr>")   # séparateur
content = "\n".join(html_parts)

full_html = f"""
<html>
<head>
    <meta charset="utf-8">
    <title>Dashboard Netflix - Analyse Exploratoire</title>
</head>
<body style="font-family:Arial, sans-serif; margin:20px;">
    <h1>Dashboard Netflix - Analyse Exploratoire</h1>
    <p>Ce rapport regroupe les principales visualisations réalisées sur le dataset Netflix :
    comparaison films/séries, pays producteurs, distribution temporelle, durées, corrélations
    et analyses additionnelles.</p>

    {content}
</body>
</html>
"""

with open("dashboard_netflix.html", "w", encoding="utf-8") as f:
    f.write(full_html)
```

MINI-DASHBOARD HTML

```python
import plotly.io as pio

# Liste des figures existantes dans l'ordre que tu veux afficher
figs = [
    ("Carte mondiale du bonheur", fig_choropleth),
    ("PIB vs Bonheur", fig_scatter),
    ("Évolution du bonheur par pays", fig_line_all),
    ("Top 10 des pays les plus heureux", fig_top),
    ("Flop 10 des pays les moins heureux", fig_flop),
    ("Corrélation entre indicateurs", fig_heat_all),
]

html_parts = []

# La première figure inclut plotly.js, les autres non
for i, (title, fig) in enumerate(figs):
    html_parts.append(f"<h2>{title}</h2>")
    html_parts.append(
        fig.to_html(
            full_html=False,
            include_plotlyjs="cdn" if i == 0 else False
        )
    )

full_html = """
<html>
<head>
    <meta charset="utf-8">
    <title>Dashboard World Happiness Report</title>
</head>
<body>
    <h1>Dashboard Interactif - World Happiness Report (2015-2019)</h1>
    {}
</body>
</html>
""".format("\n<hr>\n".join(html_parts))

with open("dashboard_world_happiness.html", "w", encoding="utf-8") as f:
    f.write(full_html)
```

BIEN QUE CERTAINES VERSIONS DU WORLD HAPPINESS REPORT UTILISENT L'INTITULÉ "COUNTRY OR REGION", LES VALEURS CONTENUES DANS CETTE COLONNE CORRESPONDENT UNIQUEMENT À DES PAYS, ET NON À DES ZONES RÉGIONALES. AINSI, APRÈS HARMONISATION ET RENOMMAGE EN COUNTRY, LA VARIABLE RESTE PARFAITEMENT EXPLOITABLE POUR LES VISUALISATIONS GÉOGRAPHIQUES.
DANS LE CAS OÙ DES VALEURS RÉGIONALES SERAIENT PRÉSENTES, PLOTLY LES IGNORERAIT SIMPLEMENT LORS DU TRACÉ DE LA CARTE (AUCUNE ZONE CORRESPONDANTE), SANS PROVOQUER D'ERREUR.

CONCLUSION

CE PROJET A PERMIS DE TRANSFORMER DEUX JEUX DE DONNÉES — NETFLIX ET WORLD HAPPINESS REPORT — EN VISUALISATIONS CLAIRES ET EXPLOITABLES.
GRÂCE À SEABORN ET PLOTLY, NOUS AVONS MIS EN ÉVIDENCE LES TENDANCES CLÉS : RÉPARTITION DES CONTENUS, ÉVOLUTION TEMPORELLE, FACTEURS INFLUENÇANT LE BONHEUR ET DIFFÉRENCES ENTRE PAYS.
LES DASHBOARDS CRÉÉS OFFRENT UNE LECTURE SIMPLE ET INTERACTIVE, DÉMONTRANT L'IMPORTANCE DE LA DATAVISUALISATION POUR COMPRENDRE RAPIDEMENT DES DONNÉES COMPLEXES.

MERCI