```
#1. BATTISTINI Lisa
#2. GIBOUDEAU Coralie
#3. LEROY Antoine
#4. PAOLO Jules
```

L'ordre n'a pas d'importance, mais la personne en position 1 sera celle qui remettra le TP sur Campus. Merci de n'opérer qu'une seule remise par groupe!

Merci de nommer vos fichiers en faisant apparaître le premier nom, par exemple ici: TPXXX_GroupeDURAND.Rmd, TPXXX_GroupeDURAND.html

ATTENTION! seul les noms inscrits dans le documents seront utilisés, aucun ajout a posteriori ne sera accepté. Vérifiez bien la présence de tous les membres avant la remise. Merci de remplir le champs suivant: Nous certifions que notre groupe ne comportait que les personnes dont les noms sont indiqués ci-dessus, et qu'aucune personne n'a été oubliée, soit un total de:

```
# quatre personnes (4)
# Nous nous engageons à ne pas réclamer l'ajout d'une autre personne
après la remise du TP
```

Votre prédiction

Vous trouverez dans le répertoire de Campus un fichier intitulé "defi_observations.csv", il comporte des observations pour 7 variables X1, . . . , X7, et une variable à prédire Y.

Vous trouverez également dans le répertoire du cours un fichier intitulé "defi_apredire.csv" comportant des valeurs pour les 7 variables X1, . . . , X7, et où il faudra prédire Y.

- Remettre vos prédictions dans un fichier csv comportant les 7 variables X1, . . . , X7, et la colonne Y prédite. Votre fichier s'intitulera "DefiGroupeXXX" où vous remplacerez le suffixe XXX par le nom de la personne indiquée en position 1 à la question 1, celle qui remettra le TP sur Campus. Pour l'exemple ci-dessus le nom du fichier serait "DefiGroupeDURAND.csv". Merci de nommer les variables X1, . . . , X7 et Y pour la colonne prédite.
- Le programme à l'origine de vos prédictions sera initulé "programmeDefiGroupeXXX" (en remplaçant

XXX. . .). Il est attendu un programme sour format Rmd+ sortie Html ou pdf correspondante (bouton Knit, ou bien Python+Jupyter Notebook avec sortie html ou pdf).

La contrainte: la prédiction doit se faire au moyen du Krigeage (eh oui, c'est un TP de Krigeage), mais vous pouvez utiliser des éléments de régression aussi (cf. Krigeage universel).

ATTENTION! il vous faudra faire attention à bien utiliser une graine pour votre générateur aéatoire, si vous en utilisez un, p.ex set.seed(12345) de façon à ce que vos résultats soit reproductibles. D'une exécution à l'autre, votre programme doit proposer LA MEME prédiction!

Vérifiez bien que votre fichier "DefiGroupeXXX.csv" comporte bien le bon nombre de lignes, et des abscisses dans le bon ordre!

Veillez à remettre impérativement ces trois fichiers:

- Le fichier CSV DefiGroupeXXX.csv
- Un notebook Rmd (ou jupyter)
- la sortie html ou pdf correspondante (bouton knit) faisant tourner votre programme

Exemple import/export :

```
# ce qui est donné:
# lecture: Observations contient les X et les Y correspondants
Observations = read.csv("defi_observations.csv", header = TRUE)
# lecture: Apredire ne contient que des X, il faut prédire les Y
Apredire = read.csv("defi_apredire.csv", header = TRUE)
# Votre prédiction; faites mieux, hein ;-)
Y = Apredire$X1 + mean(Observations$Y)
# votre Y prédit ici X1 + la moyenne des Y, c'est très mauvais!
# Votre exportation
# On concatène d'abord les X avec le Y prédit à l'aide de cbind
MonFichierSoumis = cbind(Apredire, Y)
# puis on exporte
MonNomDeFichier = "DefiGroupeDURAND.csv" # nom de fichier à adapter
hein!!!
write.csv(MonFichierSoumis, MonNomDeFichier, row.names = FALSE)
#on vérifie que c'est bien lisible
LectureDeMonFichier = read.csv(MonNomDeFichier, header = TRUE)
#on fait des vérifications élémentaires, bon nombre de lignes, de
colonnes, etc.
#on ne doit voir apparaître que des TRUE, sinon ce n'est pas bon!
message(nrow(LectureDeMonFichier) == 150, ": bon nombre de lignes")
```

```
  File "<ipython-input-17-b2d6afd1e3b6>", line 7
    Y = Apredire$X1 + mean(Observations$Y)
                  ^
SyntaxError: invalid syntax
```

Voilà, c'est à vous, vous n'avez plus qu'à faire vos prédictions! en remplaçant la ligne Y = Apredire$X1+mean(Y) bien sûr!

ETAPE 1 : Analyse des données

```
# Importing necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Loading the observation and prediction datasets
observations_path = 'defi_observations.csv'
to_predict_path = 'defi_apredire.csv'

observations = pd.read_csv(observations_path)
to_predict = pd.read_csv(to_predict_path)
```

```
# Displaying the first few rows of each dataset
observations_head = observations.head()
to_predict_head = to_predict.head()

observations_head, to_predict_head
```

```
(         X1        X2        X3        X4        X5        X6
X7  \
 0  0.720904  0.615216  0.989378  0.214680  0.222268  0.917249
0.764709
 1  0.875773  0.356072  0.770103  0.289319  0.779996  0.053148
0.182393
 2  0.760982  0.465452  0.978355  0.136963  0.836258  0.905038
0.852514
 3  0.886125  0.298943  0.854093  0.316826  0.560331  0.871237
0.925344
 4  0.456481  0.071275  0.476923  0.678568  0.879769  0.402901
0.736208

            Y
 0  44.836548
 1  62.100114
 2  48.629370
 3  51.432870
 4  56.236593  ,
          X1        X2        X3        X4        X5        X6
X7
 0  0.411601  0.894667  0.763477  0.529625  0.766796  0.794756
0.115776
 1  0.703326  0.276269  0.639110  0.755133  0.525654  0.207659
0.662348
 2  0.428881  0.856445  0.494293  0.913113  0.516349  0.320523
0.578791
 3  0.020926  0.001308  0.795911  0.142316  0.337128  0.258195
0.170953
 4  0.959199  0.814551  0.441763  0.680215  0.606672  0.646589
0.415193)
```

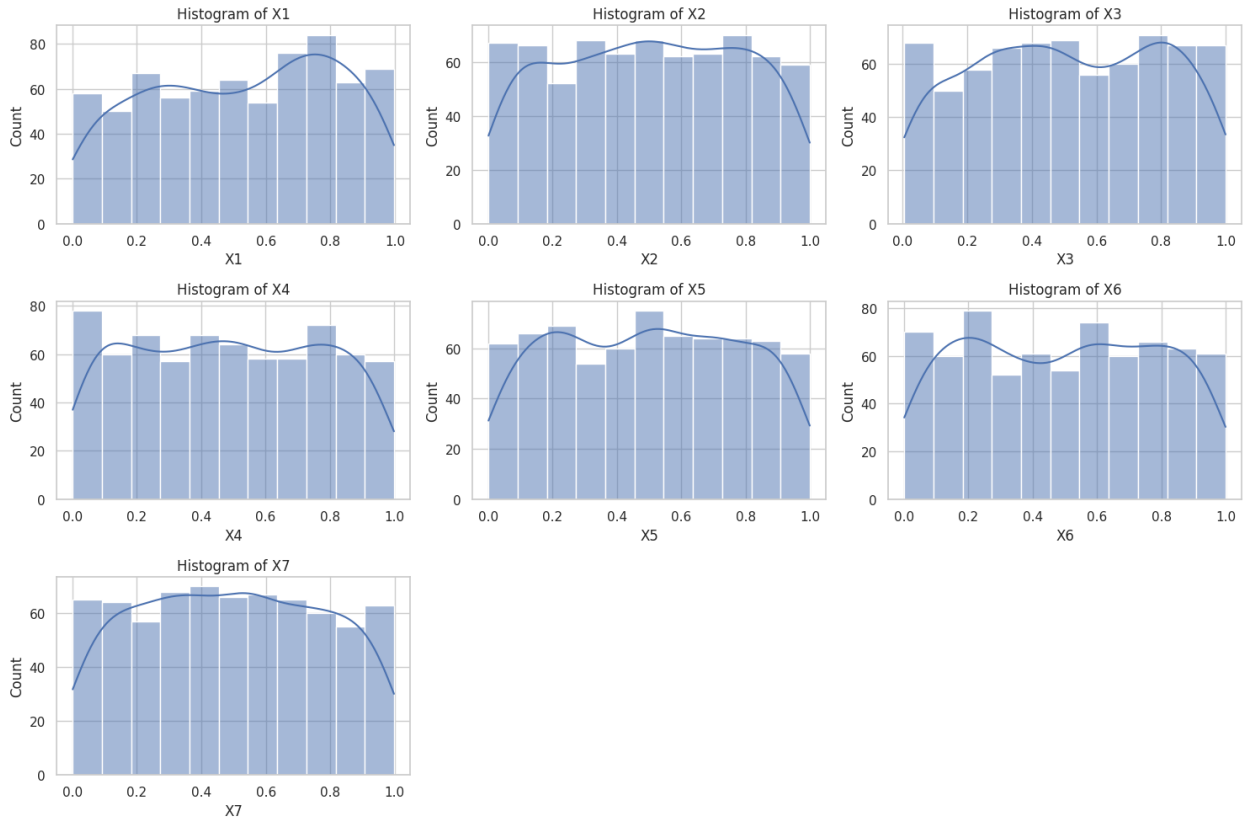
```
print(observations.describe())
```

```
               X1          X2          X3          X4          X5
X6  \
count  700.000000  700.000000  700.000000  700.000000  700.000000
700.000000
mean     0.525761    0.500472    0.510157    0.486307    0.496912
0.493766
std      0.287539    0.286069    0.288833    0.289895    0.285202
0.291538
min      0.001137    0.000516    0.005233    0.000427    0.001489
```

```
0.000886
25%      0.273345    0.262367    0.272647    0.230901    0.244350
0.232281
50%      0.538285    0.504640    0.502406    0.488062    0.507467
0.510308
75%      0.777550    0.741995    0.769832    0.742999    0.737310
0.750407
max      0.996997    0.999567    0.999012    0.999596    0.999660
0.999198
```

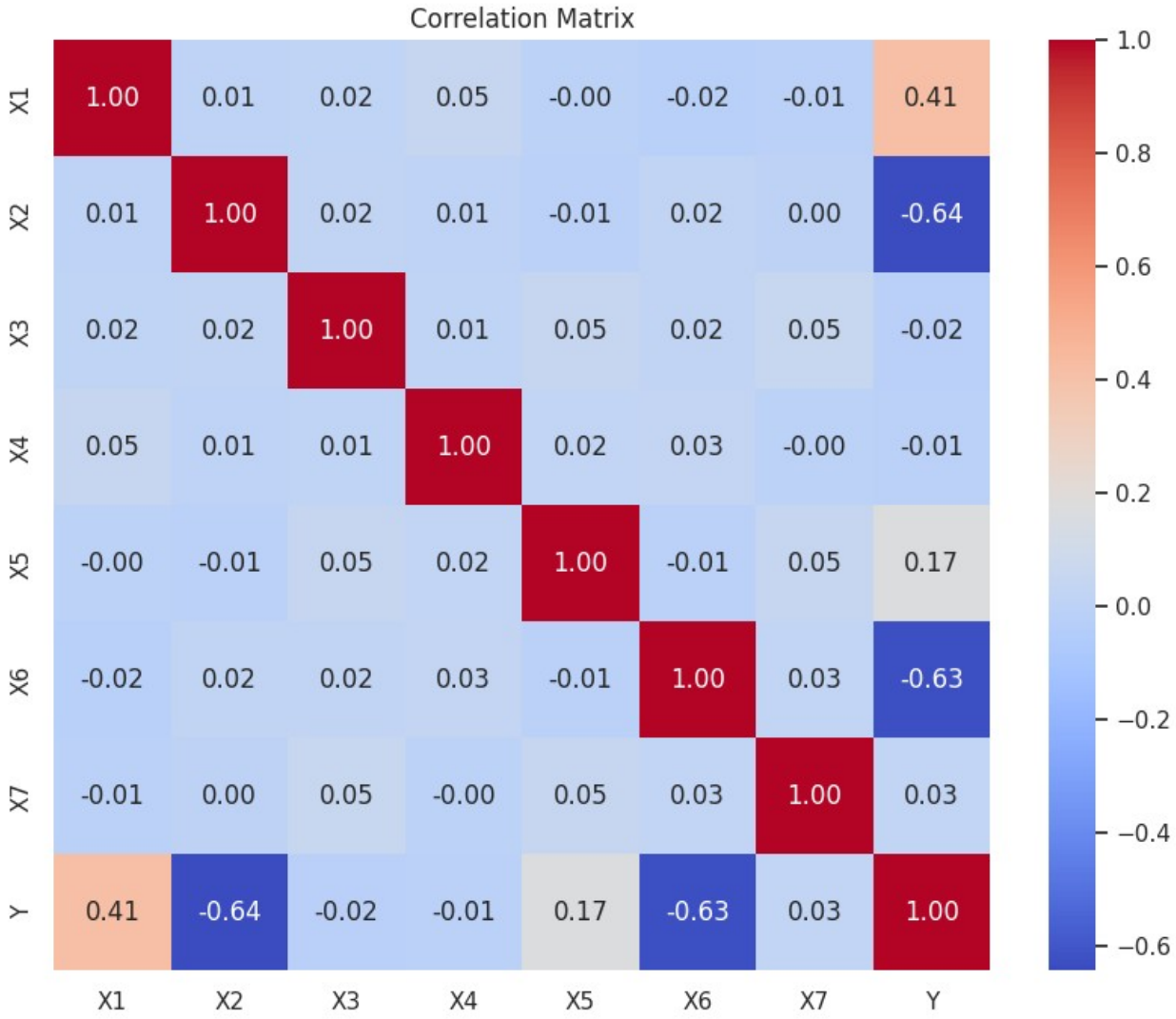
```
                X7           Y
count  700.000000  700.000000
mean     0.494135   48.932449
std      0.284071    6.304095
min      0.000840   34.905742
25%      0.259473   44.171695
50%      0.493886   49.085810
75%      0.727072   53.213642
max      0.997721   67.224834
```

```python
import seaborn as sns

# Réglage du style visuel de seaborn pour des graphiques plus esthétiques
sns.set(style="whitegrid")

# Histogrammes
plt.figure(figsize=(15, 10))
for i, col in enumerate(observations.columns[:-1]):  # Exclure la dernière colonne si c'est 'Y'
    plt.subplot(3, 3, i + 1)
    sns.histplot(observations[col], kde=True)
    plt.title(f'Histogram of {col}')
plt.tight_layout()
plt.show()
```



Distributions des Variables :
Chaque variable (X1 à X7) ainsi que la variable cible Y présentent des distributions variées. Certaines semblent être uniformément distribuées, tandis que d'autres présentent des distributions légèrement biaisées.

```
# Matrice de corrélation
plt.figure(figsize=(10, 8))
correlation_matrix = observations.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix')
plt.show()
```



Corrélation entre les Variables: La matrice de corrélation indique comment chaque variable est corrélée avec les autres, y compris la variable cible Y. On remarque rapidement que les variables explicatives sont ne sont quasiment pas corrélées entre elles si bien qu'on pourrait les considérées indépendantes.

ETAPE 2: Elaboration du modèle de Krigeage, Test et Performaneces

On décide premièrement de se concentrer sur une approche gaussienne afin d'élaborer un modèle de krigeage.

Nous allons tout d'abord diviser le jeu de données d'observation en ensembles d'entraînement et de test pour évaluer la performance du modèle.

Séparation des Données en deux jeux : test et train

```
from sklearn.model_selection import train_test_split

# Générer des données d'exemple
```

```
np.random.seed(42)

# Séparer les variables explicatives (X) et la variable cible (Y)
X = observations.drop('Y', axis=1)  # Supprime la colonne Y et garde
le reste
y = observations['Y']  # Garde uniquement la colonne Y

# Diviser les données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

Création, Entraînement et Évaluation du Modèle de Régression Linéaire

```
from sklearn.gaussian_process import GaussianProcessRegressor
from sklearn.gaussian_process.kernels import RBF, ConstantKernel as C

# Définir le modèle de krigeage avec un noyau RBF
kernel = C(1.0, (1e-3, 1e3)) * RBF(1.0, (1e-2, 1e2))
model = GaussianProcessRegressor(kernel=kernel,
n_restarts_optimizer=10, random_state=42)

# Entraîner le modèle sur les données d'entraînement
model.fit(X_train, y_train)

# Prédire les valeurs pour les données de test
y_pred, sigma = model.predict(X_test, return_std=True)

# Calculer le RMSE
MSE = 1/len(y_pred) * sum((y_pred-y_test)**2)
RMSE = np.sqrt(MSE)
print(MSE, RMSE)
```

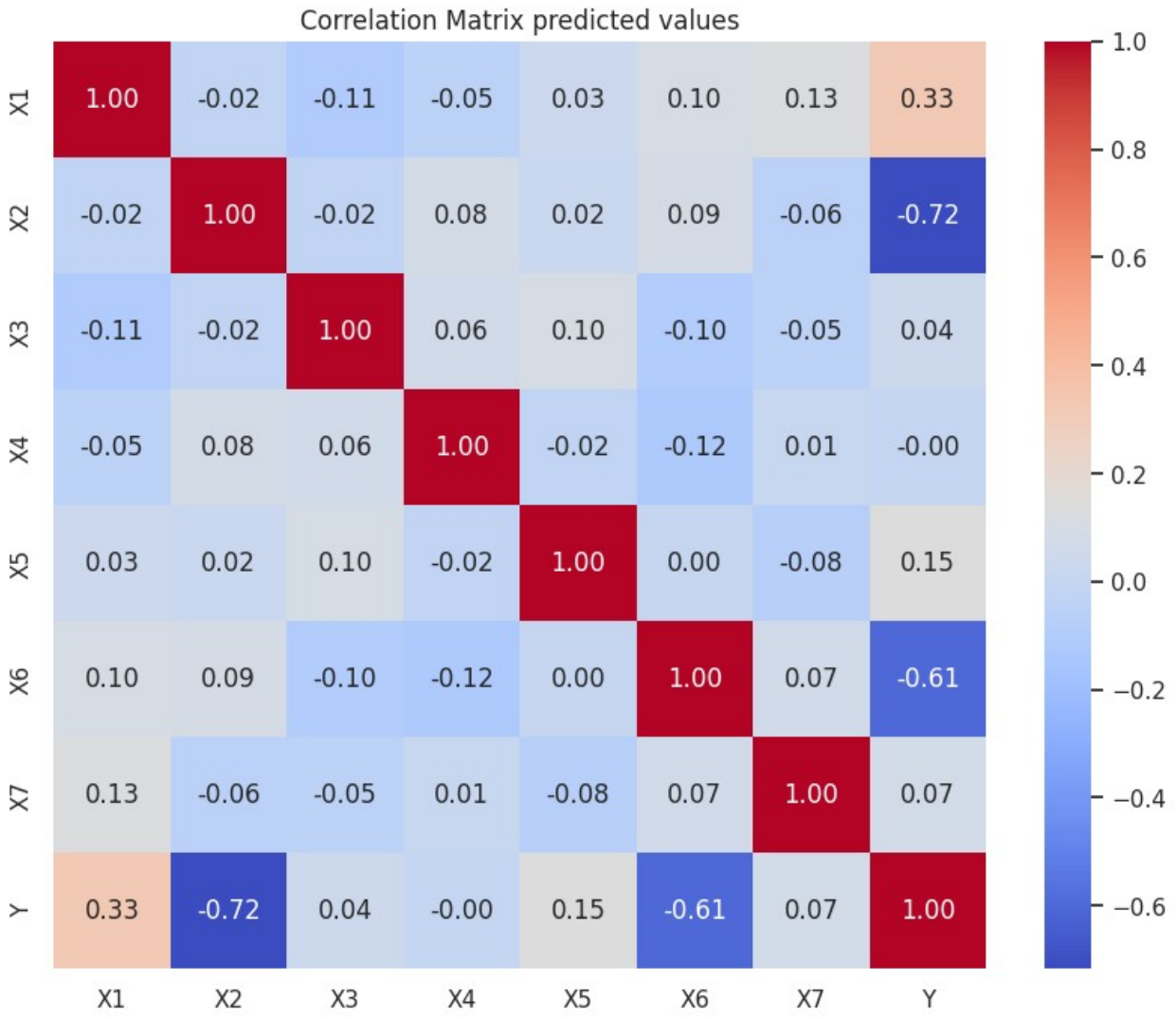
```
0.07347090750922354 0.2710551742897072
```

Prédiction : Appliquons le modèle crée précedemment sur le fichier defi_apredire.csv pour estimer la variable Y.

```
# Prédiction de la variable Y pour le jeu de données à prédire
y_pred_to_predict = model.predict(to_predict)

# Ajout des prédictions Y au jeu de données à prédire
to_predict['Y'] = y_pred_to_predict

# Sauvegarde des prédictions dans un nouveau fichier CSV
to_predict.to_csv('DefiGroupeBATTISTINI.csv', index=False)
```

On se propose de tracer la matrice de corrélation de nos variables de prédictions et de notre prédiction entre elles, afin de vérifier que l'on correspond toujours aux données d'apprentissage.

```
# Matrice de corrélation
plt.figure(figsize=(10, 8))
correlation_matrix = to_predict.corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm',
fmt=".2f")
plt.title('Correlation Matrix predicted values')
plt.show()
```



La matrice de corrélation est très proche de celle obtenue sur les données d'entraînement, ce qui nous conforte dans la prédiction de nos nouvelles données.