Groupe 1

Heyyy chat gpt tu pourrais pas me faire un programme en python déterminant la relation entre altitude et pression à partir du tableau statistique à 2 variables, fais en moi une représentation graphique sous forme de nuage de points, inclue y une droite d'ajustement affine permettant de déterminer le coefficient de détermination.

```
# Créé par liam.lavialle, le 22/11/2024 en Python 3.7
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Demander à l'utilisateur de saisir les valeurs d'altitude (en pieds)
altitudes_ft = input("Entrez les valeurs d'altitude en pieds (séparées par des espaces) : ")
altitudes_ft = np.array([float(x) for x in altitudes_ft.split()])  # Convertir en liste de floats

# Demander à l'utilisateur de saisir les valeurs de pression (en hPa)
pressions = input("Entrez les valeurs de pression en hPa (séparées par des espaces) : ")
pressions = np.array([float(x) for x in pressions.split()])  # Convertir en liste de floats

# Vérifier que le nombre d'altitudes et de pressions est le même
if len(altitudes_ft) != len(pressions):
    print("Erreur : le nombre de valeurs d'altitude et de pression doit être le même.")
else:
    # Calcul de la régression linéaire (ajustement affine)
    slope, intercept, r_value, p_value, std_err = stats.linregress(altitudes_ft, pressions)

    # Calcul du coefficient de détermination (R²)
    r_squared = r_value**2

    # Tracer le nuage de points
    plt.scatter(altitudes_ft, pressions, color='blue', label='Données')

    # Tracer la droite de régression
    plt.plot(altitudes_ft, slope * altitudes_ft + intercept, color='red', label=f'Droite d\'ajustement\nR² = {r_squared:.3f}')

    # Ajouter des labels et un titre
    plt.xlabel('Altitude (pieds)')
    plt.ylabel('Pression (hPa)')
    plt.title('Relation entre l\'altitude et la pression')
    plt.legend()

    # Afficher le graphique
    plt.grid(True)
    plt.show()

    # Afficher l'équation de la droite d'ajustement et le coefficient de détermination (R²)
    print(f"Equation de la droite d'ajustement: Pression = {slope:.2f} * Altitude + {intercept:.2f}")
    print(f"Coefficient de détermination (R²) = {r_squared:.3f}")
```

Groupe 4

Créer moi un programme python me permettant d'entrer deux variables dans un tableau statistique, afin de créer une représentation graphique du nuage de points, pour y déterminer la droite d'ajustement et le coefficient de détermination et l'équation de la droite

```
# Créé par Math-sciences, le 22/11/2024 en Python 3.7

import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Fonction pour entrer les données dans un tableau statistique
def entrer_donnees():
    print("Entrez les valeurs des deux variables sous forme de tableau.")
    x = input("Entrez les valeurs de la variable x (séparées par des espaces) : ").split()
    y = input("Entrez les valeurs de la variable y (séparées par des espaces) : ").split()

    # Conversion des valeurs en flottants
    x = [float(i) for i in x]
    y = [float(i) for i in y]

    # Vérifier que les deux listes ont la même longueur
    if len(x) != len(y):
        print("Les deux listes doivent avoir la même longueur !")
        return entrer_donnees()

    return np.array(x), np.array(y)

# Fonction pour afficher le nuage de points, la droite de régression et l'équation
def afficher_graphique(x, y):
    # Calcul de la droite de régression (pente, ordonnée à l'origine, etc.)
    slope, intercept, r_value, p_value, std_err = stats.linregress(x, y)

    # Calcul du coefficient de détermination R²
    r_squared = r_value**2
    print(f"Le coefficient de détermination R² est : {r_squared:.4f}")

    # Afficher l'équation de la droite de régression
    print(f"L'équation de la droite d'ajustement est : y = {slope:.2f}x + {intercept:.2f}")

    # Créer la droite d'ajustement
    y_pred = slope * x + intercept

    # Affichage du nuage de points
    plt.scatter(x, y, color='blue', label='Données observées')

    # Affichage de la droite de régression
    plt.plot(x, y_pred, color='red', label=f'Droite d\'ajustement: y = {slope:.2f}x + {intercept:.2f}')

    # Titre et labels
    plt.title("Nuage de points avec droite de régression")
    plt.xlabel("Variable x")
    plt.ylabel("Variable y")

    # Affichage de la légende
    plt.legend()

    # Affichage du graphique
    plt.show()

# Programme principal
def main():
    # Entrer les données
    x, y = entrer_donnees()

    # Afficher le graphique, l'équation et calculer la droite de régression
    afficher_graphique(x, y)

# Exécution du programme
if __name__ == "__main__":
    main()
```

Groupe 5
Fait initialement sous excel

Fait moi un tableau statistique à 2 variables. Fait une représentation graphique dans laquelle tu ajouteras le nuage de points correspondant au tableau statistique précédent, trace la droite d'ajustement affine puis dit-moi si le coefficient de détermination est pertinent ou non.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats

# Données
pression = np.array([999, 988.7, 980.4, 973.1, 966, 959, 950.1, 942.8, 935.8, 928.7, 920.8, 913.2, 908.2, 899.3, 891.6, 883.9])
altitude = np.array([0, 200, 400, 600, 800, 1000, 1200, 1400, 1600, 1800, 2000, 2200, 2400, 2600, 2800, 3000])

# Calcul de la droite d'ajustement affine (y = mx + b)
slope, intercept, r_value, p_value, std_err = stats.linregress(altitude, pression)

# Coefficient de détermination (R²)
r_squared = r_value**2

# Tracé du nuage de points
plt.scatter(altitude, pression, color='blue', label='Données')

# Tracé de la droite d'ajustement affine
plt.plot(altitude, slope * altitude + intercept, color='red', label=f'Droite d\'ajustement: y = {slope:.2f}x + {intercept:.2f}')

# Ajout des étiquettes et du titre
plt.xlabel('Altitude (ft)')
plt.ylabel('Pression (hPa)')
plt.title('Nuage de points et droite d\'ajustement affine')
plt.legend()

# Affichage du graphique
plt.grid(True)
plt.show()

# Affichage des résultats
print(f"Équation de la droite : y = {slope:.2f}x + {intercept:.2f}")
print(f"Coefficient de détermination (R²) : {r_squared:.4f}")

# Interprétation du coefficient de détermination
if r_squared > 0.9:
    print("Le coefficient de détermination est très élevé, ce qui signifie que la droite d'ajustement affine est un bon modèle pour les données.")
elif r_squared > 0.7:
    print("Le coefficient de détermination est modéré, ce qui indique une bonne tendance générale mais avec certaines variations.")
else:
    print("Le coefficient de détermination est faible, ce qui suggère que la droite d'ajustement affine n'est pas un bon modèle pour les données.")
```

Groupe 7

Donne-moi un programme python me permettant de rentrer deux variables (pression ; altitude) dans un tableau statistique pour ensuite tracer une représentation graphique en nuage de points puis tracer un ajustement affine s'il est pertinent déterminé à l'aide du coefficient de détermination qui devra être supérieur ou égal à 0.90. Grâce à cette représentation graphique donne-moi l'équation de la droite ce qui me permettra de trouver une relation entre la pression et l'altitude.

```
# Créé par math-sciences, le 22/11/2024 en Python 3.7

import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Fonction pour entrer les données de pression et d'altitude
def entrer_donnees():
    pression = []
    altitude = []

    # Demander à l'utilisateur combien de points il veut entrer
    n = int(input("Combien de points de données voulez-vous entrer ? "))

    for i in range(n):
        p = float(input(f"Entrez la pression pour le point {i+1} (en hPa) : "))
        a = float(input(f"Entrez l'altitude pour le point {i+1} (en mètre) : "))
        pression.append(p)
        altitude.append(a)

    return np.array(pression), np.array(altitude)

# Fonction pour effectuer un ajustement affine et calculer le coefficient de détermination
def ajuster_donnees(pression, altitude):
    # Effectuer l'ajustement affine (régression linéaire)
    slope, intercept, r_value, p_value, std_err = linregress(pression, altitude)

    # Calcul du coefficient de détermination (R^2)
    r2 = r_value ** 2

    # Afficher la droite d'ajustement et le coefficient de détermination
    print(f"Équation de la droite d'ajustement : altitude = {slope:.2f} * pression + {intercept:.2f}")
    print(f"Coefficient de détermination (R²) : {r2:.4f}")

    # Si le coefficient de détermination est supérieur ou égal à 0.90, afficher l'ajustement
    if r2 >= 0.90:
        return slope, intercept, r2
    else:
        print("Le coefficient de détermination est inférieur à 0.90, l'ajustement affine n'est pas pertinent.")
        return None, None, r2

# Fonction pour tracer le graphique
def tracer_graphique(pression, altitude, slope, intercept):
    # Tracer les points de données
    plt.scatter(pression, altitude, color='blue', label="Données observées")

    # Tracer la droite d'ajustement
    if slope is not None and intercept is not None:
        x_vals = np.linspace(min(pression), max(pression), 100)
        y_vals = slope * x_vals + intercept
        plt.plot(x_vals, y_vals, color='red', label=f"Ajustement affine")

    # Ajouter des titres et des labels
    plt.title("Relation entre la pression et l'altitude")
    plt.xlabel("Pression (hPa)")
    plt.ylabel("Altitude (m)")
```

```
    plt.legend()
    plt.grid(True)

    # Afficher le graphique
    plt.show()

# Fonction principale
def main():
    # Entrée des données
    pression, altitude = entrer_donnees()

    # Ajustement des données
    slope, intercept, r2 = ajuster_donnees(pression, altitude)

    # Si l'ajustement est pertinent, tracer le graphique
    if r2 >= 0.90:
        tracer_graphique(pression, altitude, slope, intercept)
    else:
        print("Aucun ajustement pertinent n'a été trouvé.")

# Lancer le programme
if __name__ == "__main__":
    main()
```

Groupe 8

Fait moi un programme python grâce a un tableau statistique a deux variables et représente la sous forme de représentation graphique contenant un nuage de points, qui contient aussi une droite qui donne le coefficient de détermination pour la relation pression/altitude.

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.stats import linregress

# Fonction pour saisir les données
def saisir_donnees():
    pression = []
    altitude = []
    print("Saisir les données de pression et d'altitude (saisir 'q' pour terminer) :")
    while True:
        # Saisie de la pression
        pression_str = input("Entrez la pression (en hPa) : ")
        if pression_str.lower() == 'q':
            break
        try:
            pression_val = float(pression_str)
        except ValueError:
            print("Veuillez entrer une valeur numérique valide pour la pression.")
            continue

        # Saisie de l'altitude
        altitude_str = input("Entrez l'altitude (en pieds) : ")
        try:
            altitude_val = float(altitude_str)
        except ValueError:
            print("Veuillez entrer une valeur numérique valide pour l'altitude.")
            continue

        pression.append(pression_val)
        altitude.append(altitude_val)

    return np.array(pression), np.array(altitude)

# Saisie des données
pression, altitude = saisir_donnees()

# Vérification qu'il y a des données à traiter
if len(pression) == 0 or len(altitude) == 0:
    print("Aucune donnée entrée, le programme va se terminer.")
    exit()

# Régression linéaire
slope, intercept, r_value, p_value, std_err = linregress(pression, altitude)

# Calcul du coefficient de détermination (R²)
```

	r_squared = r_value ** 2
	print(f"Le coefficient de détermination R² est : {r_squared:.4f}")
	
	# Vérification si l'ajustement est pertinent
	if r_squared >= 0.90:
	  print("L'ajustement affine est pertinent.")
	  print(f"Équation de la droite : Altitude = {slope:.4f} * Pression + {intercept:.4f}")
	else:
	  print("L'ajustement affine n'est pas pertinent (R² < 0.90).")
	
	# Tracer le nuage de points et la droite d'ajustement
	plt.scatter(pression, altitude, color='blue', label="Données")
	plt.plot(pression, slope * pression + intercept, color='red', label=f'Ajustement affine (R²={r_squared:.4f})')
	
	# Titres et labels
	plt.title("Relation entre la pression et l'altitude")
	plt.xlabel("Pression (hPa)")
	plt.ylabel("Altitude (pieds)")
	plt.legend()
	
	# Afficher le graphique
	plt.grid(True)
	plt.show()

Groupe 9

Fais-moi une représentation graphique qui détermine la relation entre les variables pression et altitudes, à l'aide d'un tableau de statistique. Il me faudra un nuage de points avec une droite représentant un coefficient de détermination (avec ordonné altitude et abscisse pression).

```
import numpy as np
from scipy import stats

# Fonction pour saisir les données
def saisir_donnees():
    pression = []
    altitude = []
    print("Entrez des valeurs de pression et d'altitude. Tapez 'fin' pour arrêter.")

    while True:
        try:
            pression_str = input("Entrez la pression (en hPa) : ")
            if pression_str.lower() == 'fin':
                break
            pression.append(float(pression_str))

            altitude_str = input("Entrez l'altitude (en mètres) : ")
            if altitude_str.lower() == 'fin':
                break
            altitude.append(float(altitude_str))
        except ValueError:
            print("Valeur incorrecte. Essayez à nouveau.")

    return np.array(pression), np.array(altitude)

# Fonction pour effectuer la régression linéaire et afficher l'équation
def ajuster_droite(pression, altitude):
    # Régression linéaire
    slope, intercept, r_value, _, _ = stats.linregress(pression, altitude)

    # Calcul du coefficient de détermination R^2
    r_squared = r_value**2
    return slope, intercept, r_squared

# Fonction principale
def main():
    # Saisie des données
    pression, altitude = saisir_donnees()

    if len(pression) < 2 or len(altitude) < 2:
        print("Il faut entrer au moins deux points de données.")
        return

    # Ajustement linéaire
    slope, intercept, r_squared = ajuster_droite(pression, altitude)
```

```
    # Vérification de la qualité de l'ajustement
    if r_squared >= 0.90:
        print(f"L'équation de la droite est : Altitude = {slope:.2f} * Pression + {intercept:.2f}")
    else:
        print(f"Le coefficient de détermination R^2 est trop faible ({r_squared:.2f}), ajustement non pertinent.")

if __name__ == "__main__":
    main()
```

Groupe 10

Fait moi un programme python avec les variables du tableau de statistique pour ensuite trouver la représentation graphique de nuage de point avec sa droite d'ajustement et trouve moi ensuite son point de détermination.

```
# Importation des bibliothèques nécessaires
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
# Fonction pour saisir les données
def saisir_donnees():
    print("Entrez les données de pression et d'altitude.")
    pression = []
    altitude = []
    n = int(input("Combien de points de données voulez-vous entrer ? "))
    for i in range(n):
        p = float(input(f"Entrez la pression (en hPa) pour le point {i+1} : "))
        a = float(input(f"Entrez l'altitude (en mètres) pour le point {i+1} : "))
        pression.append(p)
        altitude.append(a)
    # Conversion des listes en arrays NumPy
    X = np.array(altitude).reshape(-1, 1)  # Altitude est la variable indépendante
    Y = np.array(pression)  # Pression est la variable dépendante
    return X, Y
# Saisie des données
X, Y = saisir_donnees()

# Création du modèle de régression linéaire
model = LinearRegression()
model.fit(X, Y)

# Prédictions de Y en fonction de X
Y_pred = model.predict(X)

# Calcul du coefficient de détermination R²
r2 = r2_score(Y, Y_pred)

# Affichage du graphique
plt.scatter(X, Y, color='blue', label='Données réelles')  # Nuage de points
```

	plt.plot(X, Y_pred, color='red', label=f'Droite d\'ajustement: R²={r2:.2f}')
	# Droite d'ajustement
	plt.xlabel('Altitude (m)')
	plt.ylabel('Pression (hPa)')
	plt.title('Nuage de points avec droite d\'ajustement')
	plt.legend()
	plt.grid(True)
	plt.show()
	
	# Vérification si l'ajustement est pertinent (R² >= 0.90)
	if r2 >= 0.90:
	  # Affichage de l'équation de la droite
	  print(f"\nL'ajustement affine est pertinent (R² = {r2:.2f}).")
	  print(f"L'équation de la droite est : P = {model.coef_[0]:.2f} * Altitude + {model.intercept_:.2f}")
	else:
	  print(f"\nL'ajustement affine n'est pas pertinent (R² = {r2:.2f}). Il n'y a pas de relation linéaire forte entre pression et altitude.")

Groupe 11
 	
A l'aide d'un programme Python, crée un tableau statistique à deux variables, celles-ci sont la pression en fonction de l'altitude. En utilisant une représentation graphique, crée un nuage de points, puis, si le coefficient de détermination est pertinent, trace la droite.
	# Créé par math-sciences, le 22/11/2024 en Python 3.7
	import numpy as np
	import matplotlib.pyplot as plt
	from scipy.stats import linregress
	
	# Fonction pour entrer les données
	def entrer_donnees():
	  pression = []
	  altitude = []
	  while True:
	  try:
	  p = float(input("Entrez la pression (en hPa) : "))
	  a = float(input("Entrez l'altitude (en mètres) : "))
	  pression.append(p)
	  altitude.append(a)

```
            continuer = input("Voulez-vous entrer une autre paire de données ? (oui/non) : ").lower()
            if continuer != 'oui':
                break
        except ValueError:
            print("Entrée invalide, veuillez entrer des nombres valides.")
    return np.array(pression), np.array(altitude)

# Fonction pour afficher le graphique et effectuer l'ajustement affine
def graphique_et_ajustement(pression, altitude):
    # Calcul de l'ajustement affine
    slope, intercept, r_value, p_value, std_err = linregress(altitude, pression)
    r_squared = r_value**2

    # Vérifier si le coefficient de détermination est supérieur ou égal à 0.90
    if r_squared >= 0.90:
        print(f"L'ajustement affine est pertinent (R² = {r_squared:.3f})")
        print(f"L'équation de la droite de régression est : Pression = {slope:.2f} * Altitude + {intercept:.2f}")
    else:
        print(f"L'ajustement affine n'est pas pertinent (R² = {r_squared:.3f})")
        return

    # Affichage du nuage de points et de la droite de régression
    plt.scatter(altitude, pression, color='blue', label="Données")
    plt.plot(altitude, slope * altitude + intercept, color='red', label=f"Ajustement affine : P = {slope:.2f} * A + {intercept:.2f}")

    # Ajouter des labels et un titre
    plt.xlabel('Altitude (m)')
    plt.ylabel('Pression (hPa)')
    plt.title('Relation entre Pression et Altitude')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Retourner R² pour usage supplémentaire si nécessaire
    return r_squared

# Programme principal
def main():
    pression, altitude = entrer_donnees()  # Rentrer les données
    r_squared = graphique_et_ajustement(pression, altitude)  # Tracer et ajuster les données

    if r_squared >= 0.90:
        print(f"Coefficient de détermination (R²) : {r_squared:.3f}")
    else:
```

	print("Le coefficient de détermination est inférieur à 0.90, l'ajustement n'est pas pertinent.") if __name__ == "__main__": main()