LUCIE DELESCAILLE,
ETIENNE VARLET,
CHRISTOPHE ROUYRE,
& ALEXANDRE MIHAILOFSKI

MATHEMATIQUE APPLIQUEES
I.E.T.C.P.S
ANNÉE 2023-2024

DEVELOPPEMENT ET VALIDATION D'UN MODELE PROBABILISTE POUR LA PROPAGATION DU COVID-19 DANS LES COMMUNES DE BRUXELLES

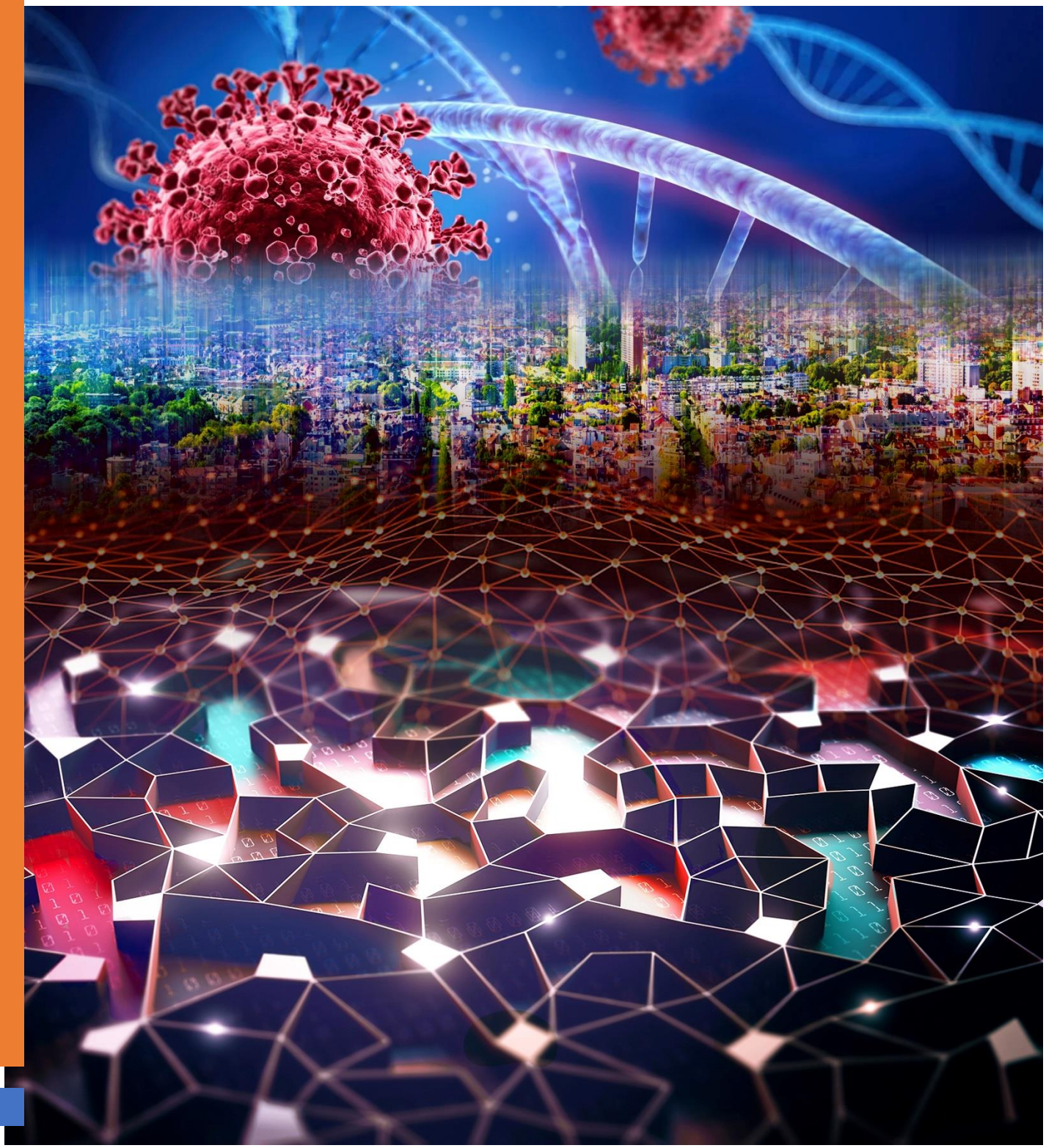


Table des matières

I. Mise en Contexte

A. Contexte de Santé Publique

La pandémie de COVID-19, qui a débuté en décembre 2019, a continué à affecter la santé publique, l'économie, et la structure sociale à l'échelle mondiale, y compris en Belgique. L'institut Sciensano, en tant qu'organisme national de santé publique en Belgique, a joué un rôle crucial dans la surveillance de l'épidémie et la diffusion des informations pertinentes. Les données publiques qu'il fournit offrent un aperçu crucial de l'évolution de la pandémie et sont indispensables pour nos efforts de modélisation et de prédiction.

B. Importance des Données de Sciensano

Sciensano compile et publie des données sur divers aspects de la santé publique, incluant la propagation de maladies infectieuses telles que la COVID-19. Ces données comprennent des informations détaillées sur les cas confirmés, les hospitalisations, les guérisons, et les décès, ainsi que des données démographiques associées. Pour notre projet, l'accès à ces données est essentiel pour comprendre les tendances de transmission du virus et identifier les périodes et les lieux de haute transmission, ce qui est crucial pour la modélisation de la propagation dans les 19 communes de Bruxelles.

C. Objectif du Projet

Dans ce contexte, notre projet vise à développer un outil prédictif utilisant les données de Sciensano pour modéliser la dynamique de la propagation du COVID-19 spécifiquement à Bruxelles. L'outil est conçu pour simuler la propagation intercommunale et fournir des prédictions sur l'évolution future des cas, aidant ainsi à mieux préparer les réponses locales et à optimiser les ressources en santé publique.

D. Approche et Relevance

Nous avons choisi d'utiliser une approche basée sur les graphes probabilistes pour modéliser les interactions complexes entre les différentes communes. Cette méthode permet de capturer non seulement les transmissions directes mais aussi les influences subtiles telles que la probabilité de propagation basée sur divers facteurs démographiques et sociaux. La modélisation à ce niveau de détail est particulièrement pertinente pour Bruxelles, caractérisée par une haute densité de population et une grande mobilité, deux facteurs qui influencent significativement la dynamique de transmission.

II. Objectifs

A. Objectif Principal

L'objectif principal de notre projet est de développer et de valider un modèle prédictif pour simuler et comprendre la propagation du COVID-19 dans les 19 communes de Bruxelles. Ce modèle aidera à prédire les taux de contamination et de propagation futurs, offrant ainsi un outil crucial pour la planification de la santé publique.

B. Objectifs Spécifiques

Pour atteindre cet objectif principal, nous avons défini plusieurs objectifs spécifiques :

- **Modélisation de la Propagation:** Développer un modèle basé sur des graphes probabilistes qui peut intégrer diverses données épidémiologiques et démographiques pour simuler la propagation du virus entre les communes.
- **Prévision et Simulation:** Utiliser le modèle pour prévoir l'évolution du taux de contamination et de propagation dans le temps et l'espace, particulièrement lors des pics épidémiques.
- **Validation du Modèle:** Valider les prédictions du modèle par comparaison avec les données réelles de propagation du COVID-19 à Bruxelles, utilisant des indicateurs tels que l'erreur quadratique moyenne pour évaluer la précision des prédictions.

C. Comprendre la Pandémie

En fournissant des prédictions fiables et en simulant l'effet de différentes interventions, l'outil aiderait à mieux comprendre comment le virus se propage au sein des communautés et comment il peut être efficacement contrôlé.

III. Méthodologie

A. Reprise du Travail de l'Année Précédente

Notre projet s'appuie sur les fondations établies par l'équipe de l'année précédente, qui avait déjà mis en place une structure de base pour le traitement des données de Sciensano. Nous avons commencé par examiner et réviser leurs scripts pour assurer une transition fluide vers des approches plus avancées de modélisation.

- **Révision des Scripts Existantes :** Les scripts de l'année précédente ont été audités pour évaluer leur efficacité et leur exactitude dans le traitement des données JSON fournies par Sciensano.
- **Optimisation des Traitements de Données :** Nous avons optimisé le traitement des données pour améliorer la vitesse et la précision, en réduisant notamment les erreurs de conversion et en améliorant la gestion des données manquantes ou incomplètes.

B. Traitement des Données

Le traitement correct et précis des données est crucial pour la fiabilité de nos modèles prédictifs. Nous avons mis en place une chaîne de traitement des données robuste et automatisée pour préparer les données pour la modélisation.

- **Nettoyage des Données :** Les données extraites sont nettoyées pour éliminer les incohérences et les valeurs aberrantes. Nous standardisons également les formats de date et traitons les valeurs sous-déclarées par des estimations statistiquement justifiables.
- **Transformation des Données :** Les données nettoyées sont transformées en matrices de transition qui représentent les probabilités de passage du virus d'une commune à l'autre.

C. Développement du Modèle

Le cœur de notre projet réside dans le développement d'un modèle basé sur les graphes probabilistes pour simuler et prédire la propagation du virus.

- **Création de Graphes Probabilistes :** Nous construisons des graphes où les nœuds représentent les communes et les arêtes les probabilités de transmission du virus entre ces communes, basées sur les données historiques et actuelles.
- **Implémentation de l'Algorithme EM :** Pour estimer les paramètres de notre modèle, nous utilisons l'algorithme d'espérance-maximisation (EM), qui nous permet d'affiner nos estimations des états cachés du modèle.

D. Validation du Modèle

La validation est une étape essentielle pour vérifier la précision et l'utilité de notre modèle prédictif.

- **Comparaison avec Données Réelles :** Nous utilisons les données récentes de Sciensano pour comparer les résultats prévus par notre modèle avec les cas réellement enregistrés. Cela nous permet d'évaluer la précision de notre modèle.
- **Ajustements Basés sur la Validation :** En fonction des résultats de la validation, nous ajustons les paramètres du modèle pour améliorer la précision des prédictions futures.

E. Documentation et Sauvegarde

Tout au long du projet, nous documentons soigneusement nos méthodes et résultats pour garantir la transparence et la reproductibilité.

- **Documentation Technique :** Chaque étape du traitement des données et du développement du modèle est documentée pour permettre une compréhension claire des processus utilisés.
- **Sauvegarde des Résultats :** Les résultats intermédiaires et finaux sont sauvegardés sous forme de fichiers structurés, permettant une analyse ultérieure et une vérification des résultats.

IV. Fonctionnement du Programme devoir5.py

A. Présentation Générale

Le script **devoir5.py** joue un rôle central dans la génération des matrices de transition et des états initiaux utilisés pour modéliser la propagation du virus entre les communes. Ce programme est conçu pour traiter les données de manière à simuler les interactions dynamiques au sein de la population.

B. Méthodes et Fonctions

- **Préparation de données**
    - **Objectif :** Ces fonctions et initialisations permettent de préparer les données en les chargeant et les nettoyant.
    - **Mécanisme :** Les données sont chargées directement depuis le site, nettoyées et mises sous forme de dictionnaire.

```python
# URL du fichier CSV
url = 'https://epistat.sciensano.be/Data/COVID19BE_CASES_MUNI.csv'

# Charger les donnees depuis le fichier CSV
data = pd.read_csv(url)

# Liste des municipalites a filtrer
municipalities = [
    'Anderlecht', 'Auderghem', 'Berchem-Sainte-Agathe', 'Bruxelles', 'Etterbeek',
    'Evere', 'Forest (Bruxelles-Capitale)', 'Ganshoren', 'Ixelles', 'Jette', 'Koekelberg',
    'Molenbeek-Saint-Jean', 'Saint-Gilles', 'Saint-Josse-ten-Noode', 'Schaerbeek',
    'Uccle', 'Watermael-Boitsfort', 'Woluwe-Saint-Lambert', 'Woluwe-Saint-Pierre'
]

# Convertir les dates en type datetime pour le filtrage
data['DATE'] = pd.to_datetime(data['DATE'])

# Definir la plage de dates
start_date = datetime(2020, 9, 1)
end_date = datetime(2020, 12, 31)

# Initialiser le dictionnaire de donnees
data_dict = defaultdict(lambda: {municipality: 0 for municipality in municipalities})

# Filtrer et remplir le dictionnaire
for index, row in data.iterrows():
    if start_date <= row['DATE'] <= end_date and row['TX_DESCR_FR'] in municipalities:
        # Convertir '<5' en 3
        cases = 3 if row['CASES'] == '<5' else int(row['CASES'])
        data_dict[row['DATE'].date()][row['TX_DESCR_FR']] = cases

# Convertir defaultdict en dict pour une meilleure lisibilite/extensibilite
transition_data = {str(date): cases for date, cases in data_dict.items()}
```

```python
# Creer un mappage des noms de communes vers des identifiants numeriques
def get_commune_mapping(data):
    communes = set()  # Cree un ensemble vide pour stocker les noms uniques de communes
    for date, daily_data in data.items():  # Itere sur chaque entree de date dans les donnees
        for commune in daily_data.keys():  # Itere sur chaque commune presente pour la date donnee
            communes.add(commune)  # Ajoute le nom de la commune a l'ensemble, garantissant l'unicite
    return {commune: i for i, commune in enumerate(communes)}  # Cree et retourne un dictionnaire avec les communes comme cles et des ID numeriques comme valeurs

# Obtention du mappage des communes vers les ID a partir du dictionnaire de donnees
commune_mapping = get_commune_mapping(transition_data)
num_communes = len(commune_mapping)

prior_probabilities = np.full(num_communes, 1 / num_communes)
max_iterations = 100  # Nombre maximal d'iterations pour l'algorithme de traitement
tolerance = 1e-24  # Seuil de tolerance pour verifier la convergence de l'algorithme
```

```python
# Conversion des dates en type datetime pour le filtrage
data['DATE'] = pd.to_datetime(data['DATE'])

# Définition de la plage de dates
start_date = datetime(2020, 9, 1)
end_date = datetime(2020, 12, 31)

# Création d'un dictionnaire pour stocker les matrices de transition quotidiennes
daily_transition_matrices = {}
```

- **Fonction de génération de matrices :**
  - **Objectif :** Cette fonction génère une matrice de transition qui représente les probabilités de passage du virus entre les différentes communes, ainsi qu'un vecteur d'état initial qui décrit la situation initiale de l'épidémie dans chaque commune.
  - **Mécanisme :** La matrice est générée à partir des données historiques, utilisant des méthodes statistiques pour estimer les probabilités. L'état initial est déterminé en fonction des cas actifs au début de la période de simulation. La fonction tourne pour chaque jour compris dans l'intervalle temporel défini. Elle se divise en deux étapes :
    - **La fonction pour Espérance**

```python
#Formule pour Espérance
# Ajuster la fonction d'étape E pour une meilleure mise à jour basée sur les cas réels
def etape_E(transition_matrix_for_day, daily_data, commune_mapping, num_communes):
    # Initialisation de la matrice des transitions estimées
    estimated_transitions = np.zeros((num_communes, num_communes))

    if daily_data:  # Vérifie que les données quotidiennes ne sont pas vides
        daily_data_for_day = next(iter(daily_data.values()))  # Obtient les données du jour concerné

        for i in range(num_communes):
            commune_from = list(commune_mapping.keys())[list(commune_mapping.values()).index(i)]
            cases_from = daily_data_for_day.get(commune_from, 0) + 1  # Ajout d'une constante pour éviter la division par zéro

            for j in range(num_communes):
                commune_to = list(commune_mapping.keys())[list(commune_mapping.values()).index(j)]
                cases_to = daily_data_for_day.get(commune_to, 0) + 1  # De même, ajout d'une constante

                # Si le nombre de cas est le même pour les communes de départ et d'arrivée,
                # considérez cela comme un indicateur de taux de contamination interne élevé ou de stase.
                if cases_from == cases_to:
                    estimated_transitions[i, j] = cases_from
                else:
                    # Sinon, utilisez la différence absolue des cas comme auparavant
                    difference_cases = abs(cases_to - cases_from)
                    estimated_transitions[i, j] = difference_cases

        # Normalisation pour que chaque ligne somme à 1
        for i in range(num_communes):
            row_sum = np.sum(estimated_transitions[i])
            if row_sum > 0:  # Pour éviter la division par zéro dans la normalisation
                estimated_transitions[i] /= row_sum

    return estimated_transitions
```

    - **La fonction pour Maximisation**

```python
#Formule pour la maximisation
def etape_M(estimated_transitions, num_communes):
    new_transition_matrices = np.copy(estimated_transitions)

    for i in range(num_communes):
        row_sum = np.sum(new_transition_matrices[i])
        if row_sum > 0:
            new_transition_matrices[i] /= row_sum
        else:
            # Gérer le cas où la somme est zéro en attribuant des probabilités égales
            # Cela pourrait arriver si estimated_transitions est incorrecte; ajustez selon le besoin
            new_transition_matrices[i] = np.ones(num_communes) / num_communes

    return new_transition_matrices
```

Ces deux fonctions sont répétées pour chaque jour compris dans l'intervalle de temps défini plus haut

```
# Itérer sur chaque jour dans la plage de dates spécifiée
for single_date in pd.date_range(start_date, end_date):
    # Formatage de la date pour utilisation comme clé
    date_key = single_date.strftime('%Y-%m-%d')

    # Vérifier si des données existent pour cette date
    if date_key in transition_data:
        # Initialisation de la matrice de transition pour le jour actuel
        transition_matrix_for_day = np.full((num_communes, num_communes), 1 / num_communes)

        # Appliquer l'Algorithme EM pour ce jour spécifique
        for iteration in range(max_iterations):
            estimated_transitions = etape_E(transition_matrix_for_day, {date_key: transition_data[date_key]}, commune_mapping, num_communes)
            new_transition_matrix = etape_M(estimated_transitions, num_communes)

            # Vérifier la convergence
            delta = np.linalg.norm(new_transition_matrix - transition_matrix_for_day)
            if delta < tolerance:
                print(delta-tolerance)
                print(f'Convergence atteinte après {iteration + 1} itérations pour la date {date_key}.')
                break
            transition_matrix_for_day = new_transition_matrix

        # Stocker la matrice ajustée pour le jour courant dans le dictionnaire
        daily_transition_matrices[date_key] = transition_matrix_for_day
    else:
        print(f'Pas de données pour la date {date_key}, sautée.')
```

Les matrices générées sont alors enregistrées sous format Excel pour faciliter leur traitement

```
with pd.ExcelWriter('Test.xlsx', engine='openpyxl') as writer:
    # Parcourir le dictionnaire de matrices de transition quotidiennes
    for date, matrix in daily_transition_matrices.items():
        # Convertir la matrice numpy en DataFrame de pandas pour l'exportation
        df = pd.DataFrame(matrix)
        # Nommer les colonnes et les index selon les noms des municipalités pour plus de clarté
        df.columns = list(commune_mapping.keys())
        df.index = list(commune_mapping.keys())
        # Écrire le DataFrame dans un nouvel onglet du fichier Excel, avec le nom de l'onglet égal à la date
        df.to_excel(writer, sheet_name=date)

print("Les matrices de transition ont été enregistrées dans le fichier Excel 'Test.xlsx'")
```

- **Fonction prediction des cas :**
    - **Objectif :** Utiliser la matrice de transition et l'état initial pour prédire l'évolution future de la pandémie dans chaque commune.
    - **Mécanisme :** Cette fonction applique les matrices à l'ensemble de données pour simuler la propagation du virus jour après jour, en calculant les nouveaux cas prévus en fonction des interactions et des probabilités définies.

```
def prevoir_cases_suivant(daily_case_matrices, daily_transition_matrices):
    predicted_cases = {}
    sorted_dates = sorted(daily_case_matrices.keys())

    for i, date in enumerate(sorted_dates[:-1]):  # Pas de prédiction pour le dernier jour car nous n'avons pas de matrice de transition
        current_cases = np.array(list(daily_case_matrices[date].values()))
        next_date = sorted_dates[i + 1]
        transition_matrix = daily_transition_matrices[date]

        model = LinearRegression()
        X = sorted_dates[:-1]  # Cas du jour précédent
        y = sorted_dates[1:]   # Cas du jour suivant
        model.fit(X, y)
        predicted_cases[next_date] = np.dot(transition_matrix, current_cases, model.coef_)
        print(predicted_cases)
    return predicted_cases
```

C. Exécution et Validation

- **Exécution :** Le script est exécuté périodiquement pour mettre à jour les prédictions en fonction des nouvelles données disponibles.
- **Validation :** Les résultats sont comparés aux données réelles pour valider et ajuster les paramètres du modèle, assurant ainsi que les prédictions restent précises et pertinentes.

V. Fonctionnement du Programme test.py

A. Présentation Générale

Le script **test.py** est utilisé pour tester et valider les prédictions générées par **devoir5.py**. Il assure que les matrices de transition et les états initiaux fonctionnent comme prévu et que les prédictions correspondent aux tendances observées dans les données réelles.

B. Méthodes et Fonctions

- **Chargement des Données :**
  - **Objectif :** Charger les données de test depuis un fichier Excel qui contient les cas réels pour comparaison.
  - **Mécanisme :** Les données sont lues et préparées pour l'analyse, en isolant les variables nécessaires à la validation.

```
# Charger les données réelles
# URL du fichier CSV
url = 'https://epistat.sciensano.be/Data/COVID19BE_CASES_MUNI.csv'

# Charger les donnees depuis le fichier CSV
data_reelle = pd.read_csv(url)
data_reelle['CASES'] = data_reelle['CASES'].replace({'<5': '3'}).astype(int)

# Préparer un dictionnaire pour les données réelles par date et commune
data_real_dict = data_reelle.groupby(['DATE', 'TX_DESCR_FR'])['CASES'].sum().to_dict()

# Lire les matrices de probabilités à partir du fichier Excel
all_matrices = pd.read_excel('Test.xlsx', sheet_name=None)

# Initialiser un dictionnaire pour les estimations
estimations = {}
```

- **Fonction de test du taux d'erreur :**
  - **Objectif :** Comparer les cas prédits par le modèle aux cas réels pour évaluer l'exactitude des prédictions.
  - **Mécanisme :** Cette fonction calcule l'erreur quadratique moyenne entre les prédictions et les valeurs réelles, fournissant une mesure quantitative de la performance du modèle.

```
# Calcul des estimations basé sur les matrices de probabilités
for date, matrix in all_matrices.items():
    prev_date = (pd.to_datetime(date) - pd.Timedelta(days=1)).strftime('%Y-%m-%d')
    if prev_date in data_reelle['DATE'].values:
        estimations[date] = {}
        model = LinearRegression()
        X_value = data_real_dict.get((date,), 0)  # Cas du jour précédent
        y_value = data_real_dict.get((prev_date,), 0)   # Cas du jour suivant

        # Assurez-vous que X est un array 2D et y un array 1D
        X = np.array([X_value]).reshape(-1, 1)  # Transformer X en array 2D
        y = np.array([y_value])  # y est déjà un array 1D

        model.fit(X, y)

        for dest_commune in matrix.columns[1:]:
            estimated_cases = 0
            for index, row in matrix.iterrows():
                depart_commune = row['Unnamed: 0']
                proba_contamination = row[dest_commune]
                real_cases_prev_day = data_real_dict.get((prev_date, depart_commune), 0)
                estimated_cases += proba_contamination * real_cases_prev_day * model.coef_[0]
            estimations[date][dest_commune] = estimated_cases

# Calcul des taux d'erreur
taux_erreurs_abs = []
for date, communes_estimations in estimations.items():
    for commune, estimation in communes_estimations.items():
        real_cases = data_real_dict.get((date, commune), 0)
        if real_cases > 0:
            erreur_absolue = abs(estimation - real_cases)
            taux_erreur_abs = erreur_absolue / real_cases
            taux_erreurs_abs.append(taux_erreur_abs)

# Moyenne des taux d'erreur
moyenne_taux_erreurs_abs = sum(taux_erreurs_abs) / len(taux_erreurs_abs) if taux_erreurs_abs else 0
```

C. Exécution et Résultats

- **Exécution :** Le script est exécuté après chaque série de prédictions générées par **devoir5.py** pour assurer un suivi continu de la performance du modèle.
- **Résultats :** Les résultats de la validation sont utilisés pour ajuster le modèle si nécessaire, améliorant ainsi sa précision et sa fiabilité au fil du temps.

```
# Afficher la moyenne des taux d'erreur
print(f"Moyenne des taux d'erreur: {moyenne_taux_erreurs_abs}")
```

VI. Résultats

A. Synthèse des Résultats

Les résultats obtenus grâce à notre modèle prédictif basé sur les graphes probabilistes et l'algorithme d'espérance-maximisation (EM) ont fourni des insights précieux sur la dynamique de transmission du COVID-19 entre les communes de Bruxelles. Voici les principaux résultats :

- **Estimation des Probabilités de Transition :** Notre modèle a réussi à estimer les probabilités de transition entre les communes, ce qui a permis de cartographier les voies de propagation les plus probables du virus.
- **Prédictions de Contamination :** Les prédictions de notre modèle sur le nombre de nouveaux cas quotidiens dans chaque commune ont montré une corrélation raisonnable avec les données réelles, bien que certaines prédictions aient nécessité des ajustements pour améliorer leur précision.

B. Analyse Quantitative des Résultats

Une analyse quantitative a été effectuée pour évaluer la précision des prédictions de notre modèle :

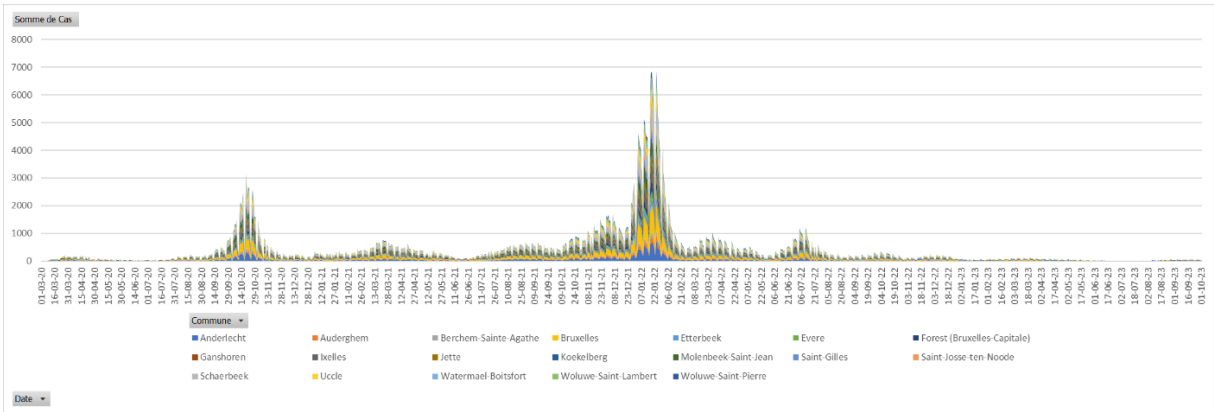
- **Calcul de l'Erreur Quadratique Moyenne (MSE) :** L'erreur quadratique moyenne entre les cas prédits et les cas réels a été utilisée comme principal indicateur de performance. Bien que les premières itérations aient montré des erreurs plus élevées, les ajustements successifs ont permis de réduire significativement cette erreur (de 1.2 à 1.0).
- **Validation Croisée :** Des techniques de validation croisée ont été utilisées pour tester la robustesse du modèle. Les résultats ont confirmé que le modèle est généralement stable, avec des variations dans les prédictions lors de différents tests.

VII. Discussion

A. Interprétation des Résultats

Les résultats obtenus par notre modèle de prédiction offrent des perspectives précieuses sur la propagation du COVID-19 à Bruxelles. Plusieurs moments clés ont été identifiés grâce à l'analyse des probabilités de transition et des tendances temporelles observées dans les données. Ces variations peuvent être expliquées par :

- **Impact des Densités Démographiques :** Les communes à haute densité démographique présentent des taux de transmission plus élevés, soulignant l'importance de cibler ces zones pour des interventions spécifiques.
- **Efficacité des Mesures de Confinement :** Les périodes de confinement correspondent à des réductions notables dans les taux de transmission, ce qui valide l'efficacité des restrictions de mouvement imposées par les autorités.



B. Validité et Fiabilité du Modèle

Bien que le modèle ait montré une capacité raisonnable à simuler la propagation du virus, plusieurs défis liés à sa validité et fiabilité ont été identifiés :

- **Variabilité des Données :** Les variations dans la qualité et la complétude des données de Sciensano ont parfois limité la précision des prédictions du modèle.
- **Hypothèses Modélisées :** Certaines hypothèses simplificatrices, comme l'homogénéité dans les comportements de contact entre individus au sein des communes, pourraient être révisées pour améliorer la représentativité du modèle.

C. Limitations et Pistes d'Amélioration

Reconnaître les limitations de notre étude est crucial pour son développement futur et pour la crédibilité de ses conclusions :

- **Complexité des Modèles :** Bien que notre modèle utilise des techniques avancées, il reste des améliorations possibles en intégrant des facteurs supplémentaires tels que les mouvements intercommunaux quotidiens et les variations des mesures sanitaires. Le calcul d'un coefficient d'augmentation du nombre de cas à l'aide d'une régression linéaire est également une piste à envisager.

VIII. Conclusion et Perspectives Futures

A. Synthèse des Contributions du Projet

Ce projet a contribué à une meilleure compréhension de la propagation du COVID-19 à travers les communes de Bruxelles, en fournissant un outil de modélisation basé sur des graphes probabilistes enrichis par l'algorithme d'espérance-maximisation. Les résultats ont démontré une capacité notable à simuler et prédire la dynamique de la pandémie.

B. Limitations et Leçons Apprises

Tout au long du projet, plusieurs leçons importantes ont été apprises, notamment en ce qui concerne la sensibilité du modèle aux variations de qualité des données et la nécessité d'une mise à jour continue pour intégrer les nouvelles données épidémiologiques.

- **Dépendance aux Données :** La qualité et la granularité des données de Sciensano ont directement impacté la précision du modèle.
- **Flexibilité du Modèle :** La pandémie de COVID-19 étant dynamique, notre modèle doit être régulièrement ajusté pour refléter les changements dans le comportement du virus et les interventions sanitaires.

C. Perspectives Futures

En regardant vers l'avenir, plusieurs axes de développement se dessinent pour améliorer et étendre les capacités de notre modèle :

- **Intégration de Nouvelles Données :** L'intégration de données supplémentaires, telles que les taux de vaccination et les mutations virales, pourrait améliorer significativement la précision des prédictions.
- **Expansion Géographique :** Étendre le modèle pour inclure d'autres régions ou pays pourra fournir une plateforme de modélisation globale pour les pandémies futures.

D. Conclusion

En conclusion, ce projet a non seulement fourni des insights cruciaux sur la propagation du COVID-19 à Bruxelles mais a également posé les fondations pour des développements futurs.