1. Une analyse réplicable, c'est quoi ? 1
2. Étude de cas : l'incidence de syndromes grippaux 2
3. Importer les données 3
3.A. Importer les données (Jupyter) 3
3.B. Importer les données (RStudio) 4
3.C. Importer les données (Org-mode) 6
4. Vérification et inspection 8
4.A. Vérification et inspection (Jupyter) 8
4.B. Vérification et inspection (RStudio) 9
4.C. Vérification et inspection (Org-mode) 11
5. Questions et réponses 13
5.A. Questions et réponses (Jupyter) 13
5.B. Questions et réponses (RStudio) 15
5.C. Questions et réponses (Org-mode) 17

1. Une analyse réplicable, c'est quoi ?

Dans ce module, nous mettons la main à la pâte et vous montrons pas à pas comment réaliser une analyse de données réplicable sous forme d'un document computationnel.

D'abord, nous vous expliquons ce qu'est une analyse de données réplicable. Dans une analyse de données traditionnelle, on se concentre sur les résultats et on montre un résumé méthodologique des calculs ayant permis de les obtenir. On termine avec une discussion, notamment pour exposer les conclusions tirées de l'analyse. Dans une analyse de données réplicable, on remplace le résumé méthodologique par la totalité des codes ayant permis de faire les calculs, accompagnés d'une explication détaillée, notamment des choix faits à chaque étape.

Pourquoi faire une analyse de données réplicable qui demande plus d'efforts ? Premièrement, l'analyse est plus facile à refaire si des données changent. Deuxièmement,

l'analyse elle-même est plus facile à modifier si d'autres questions auxquelles on aimerait répondre se posent. Troisièmement, l'analyse est plus facile à inspecter et donc à vérifier. On fait plus facilement confiance à une analyse dont tous les éléments peuvent être vérifiés en détail.

Dans les séquences qui suivront, je vous montrerai comment faire pas à pas. Comment récupérer les données, les regarder, les inspecter, faire une analyse et montrer les résultats.

2. Étude de cas : l'incidence de syndromes grippaux

Dans cette séquence, nous allons analyser les données provenant du réseau "Sentinelles" regroupant des médecins généralistes surveillant certaines maladies toute l'année.

Sur le site du réseau "Sentinelles", nous allons dans "Bases de données", dans les "Tableaux de données", où nous allons demander les données sur les syndromes grippaux que nous souhaitons extraire sous forme d'un fichier .csv. Nous commençons en 1985, première semaine, et nous allons terminer aujourd'hui. Nous demandons les données pour la France métropolitaine et nous lançons le téléchargement.

Regardons les données que nous venons de télécharger. Sur la première ligne, il n'y a qu'un commentaire sans importance pour l'analyse. La deuxième ligne donne les étiquettes pour les colonnes des données. Après, chaque ligne correspond à une semaine de l'année. Dans la première colonne, nous avons l'année et la semaine en format ISO. Le format ISO précise que l'année commence par la semaine contenant le 4 janvier. Nous avons donc sur la première ligne les données pour la semaine 39 de l'année 2017, dernière semaine pour laquelle les données sont disponibles. Dans la troisième colonne, nous avons les données qui nous intéressent. L'incidence des syndromes grippaux. C'est un entier, un chiffre par semaine. Nous allons maintenant parcourir le fichier pour voir visuellement s'il y a des anomalies. Là, il y a une semaine pour laquelle il n'y a pas de données, juste un tiret. Il s'agit de la semaine 19 de l'année 1989. C'est un phénomène assez fréquent. Suite à des erreurs de prises de données, de gestion des données, il n'y a, à un moment donné, simplement pas de chiffre. Dans notre cas, ce n'est pas particulièrement grave. Ce n'est qu'une semaine sur de longues années. Nous pouvons a priori simplement ignorer ce point et le supprimer. Beaucoup de gens seraient tentés de prendre l'éditeur de texte, marquer cette ligne, la supprimer, sauvegarder le fichier et analyser le fichier modifié. Pourtant, une telle approche ne serait pas réplicable. Il n'y aurait aucune trace de la suppression de cette ligne. Un lecteur de notre analyse de données

comparant avec les données d'origine pourrait se demander pourquoi cette ligne a été supprimée. Ou pire, il pourrait penser à une tentative de fraude.

Voici les points clés à retenir. Il ne faut jamais faire une modification des données "à la main". Tout doit se faire dans du code. Par la suite nous montrerons comment faire cela.

3. Importer les données

3.A. Importer les données (Jupyter)

Dans cette séquence, nous allons montrer comment importer des données dans un notebook Jupyter.

Les choix techniques pour cette séquence et les suivantes sont : le document computationnel est réalisé sous forme d'un notebook Jupyter. Le langage de programmation sera Python 3. Et nous utiliserons quelques bibliothèques pour nous faciliter le travail, notamment la bibliothèque Pandas pour la gestion des données, Matplotlib pour faire des plots, des figures, et Isoweek pour gérer les numéros de semaines au format ISO, parce que Pandas ne comprend pas ce format.

Sur la page d'accueil de Jupyter, nous allons créer un nouveau notebook en langage Python 3. Nous allons lui donner un nom : "analyse-syndromes-grippaux". Pour gagner un peu de place sur mon écran, je vais faire disparaître cette première ligne. On commence typiquement un notebook avec une préface où on donne les bibliothèques à importer pour qu'un lecteur sache immédiatement sur quoi on se base. La toute première chose, c'est souvent cette ligne magique qui dit à Jupyter qu'on veut avoir les plots directement dans le notebook plutôt que dans une fenêtre séparée. Maintenant, nous allons importer les trois bibliothèques mentionnées précédemment. Là, nous avons malheureusement oublié de donner un titre à notre document. Il n'est jamais trop tard pour se rattraper. Nous créons une cellule Markdown, qui est donc du texte et nous la déplaçons en haut de notre document. La prochaine étape, c'est l'importation des données. Pandas sait lire le format .csv que nous venons de montrer. Mais nous avons besoin de l'adresse, de l'URL pour télécharger ces données. Comment la récupérer ? Le site web du Réseau Sentinelle ne le montre pas directement. Pour ça, il y a une astuce relativement simple. Nous allons demander à notre navigateur qui a fait le téléchargement et qui a gardé l'adresse dans la liste, de nous donner l'URL. Voilà. C'est un peu long, mais c'est comme ça. Nous devons encore ajouter un petit argument, "skiprows=1", pour expliquer à Pandas que la première ligne, qui est un commentaire, doit être ignorée dans

la lecture de ce fichier. Nous pouvons maintenant jeter un premier coup d'œil sur nos données. Là, Pandas nous affiche ce qu'il vient de lire. Les 30 premières et les 30 dernières lignes. Tout se passe bien apparemment. La prochaine étape est de nous occuper des données manquantes. Nous venons de voir que dans ce fichier il y a une ligne, une semaine, qui n'a pas de données. Comme nous nous sommes interdits de la supprimer dans un éditeur de texte, nous devons maintenant, dans notre notebook, trouver, identifier cette ligne et la supprimer. D'abord, pour identifier, nous demandons à Pandas où il n'y a pas de données. Malheureusement, ça donne une réponse assez volumineuse. Mais on s'adresse seulement aux lignes. On veut savoir dans quelle ligne il manque quelque chose. Maintenant, nous avons 1 700 lignes avec "False" et "True". Ce qui nous intéresse réellement, c'est de voir les lignes, on sait qu'il n'y en a qu'une pour lesquelles il y a des données manquantes. Et voilà. C'est la ligne de la semaine 19 de l'année 1989. Nous allons ajouter un petit commentaire, une cellule en format Markdown, pour expliquer à nos lecteurs que nous supprimons une ligne pour une bonne raison. Comment faire ? Pandas sait supprimer les données manquantes car c'est une opération fréquente. Le petit détail à savoir, c'est qu'après il faut faire une copie des données. Ça irait trop loin de vous expliquer pourquoi. Le manuel de Pandas l'explique. C'est un phénomène lié à la gestion interne de la mémoire. C'est un problème technique qu'on aimerait ne pas avoir. Mais on ne peut s'en passer. Regardons les données qu'il nous reste. C'est à peu près le même résultat qu'avant, sauf qu'il y a une ligne en moins.

Voici les points clés à retenir de cette séquence : il faut lire les données directement de la source dans le document computationnel. Il faut s'occuper des données manquantes même si, comme dans notre cas, on peut simplement les supprimer. Par la suite, nous montrerons comment vérifier et inspecter les données.

3.B. Importer les données (RStudio)

Dans cette séquence, nous montrons comment créer un document computationnel et comment importer nos données sur l'incidence du syndrome grippal.

Mais d'abord, il faudra faire quelques choix techniques. Nous réalisons notre document computationnel dans RStudio, notre langage de programmation sera R et nous utilisons la bibliothèque "parsedate" pour la gestion des dates en format ISO.

Dans RStudio, nous créons un nouveau document du type R Markdown et nous lui donnons un titre : "Analyse de l'incidence du syndrome grippal". RStudio a prérempli des informations, notamment les métadonnées, qui sont tout à fait convenables. Nous allons

supprimer le reste de ce document, qui sert comme exemple en présentant le langage R Markdown, mais nous allons à la place mettre notre propre contenu. Nous devons d'abord télécharger nos données du réseau Sentinelles. Pour cela, nous devons connaître l'URL de téléchargement. Nous la récupérons le plus facilement de notre navigateur qui nous a servi pour le téléchargement. Nous allons dans la liste des téléchargements. Il ne reste plus qu'à insérer l'URL dans le document et exécuter ce petit bloc de code R. Maintenant, nous pouvons télécharger les données. Mais nous devons encore indiquer à R d'ignorer la première ligne qui contient un commentaire. Et nous allons regarder le début des données. Donc, nous voyons que les données sont en ordre chronologique inverse. La première ligne correspond au dernier point du jeu de données. Nous avons les colonnes que nous avons vues précédemment. Ce qui nous intéresse est la colonne "week", qui contient le numéro de semaine, et la colonne "inc", "incidence", qui donne nos valeurs à analyser. Regardons, pour être sûrs, l'autre bout du jeu de données. Là, nous voyons l'autre bout, donc les premières données de 1985. Maintenant, nous allons regarder s'il y a des points manquants et comment les gérer. Pour cela, nous demandons d'abord à R de nous dire dans quelles lignes il y a des données manquantes. Donc, "manquantes", en anglais, c'est "na" pour "not available", donc on va utiliser ça. Nous allons traverser notre jeu de données ligne par ligne. C'est le numéro "1" qui indique ça. Et nous allons appliquer à chaque ligne la fonction suivante, qui, pour sa part, traverse la ligne colonne par colonne et revient "true" s'il y a au moins un champ qui a une valeur non disponible. Mais nous ne voulons pas voir les numéros de ligne, nous voulons voir le contenu des lignes. Donc, nous affichons les lignes qui correspondent à ce critère et nous retrouvons la semaine 19 de l'année 1989, que nous avions déjà repérée comme données manquantes précédemment. Un dernier test qui est à effectuer pour être sûr que tout s'est bien passé, c'est d'inspecter le type de données. Nous allons demander à R quel est le type de données de la colonne "week". Il s'agit d'entiers. Nous devons nous en occuper, mais effectivement, cette combinaison d'années et semaines, ça ressemble à un entier. Et l'autre colonne, "inc", là, il va falloir faire quelque chose, parce que R a déclaré qu'il s'agit du type "factor". Ceci est dû aux données manquantes, parce que si vous regardez bien, dans cette colonne, il y a un tiret à la place d'une valeur numérique, et R prend ça comme une valeur potentiellement réelle, possible, et du coup, ce n'est plus un entier, mais quelque chose de plus vaste. Je pourrais tenter de réparer cette erreur et reconstituer un jeu de données avec des entiers, mais le plus facile, c'est de relire les données, un peu comme avant, avec l'URL et le "skip=1" pour sauter la première ligne, mais en rajoutant un argument qui dit à R que le tiret est à considérer

comme une donnée manquante. Et après cela, nous allons vérifier que ça a bien réglé le problème. Ce qui est bien le cas, nous avons maintenant des entiers.

Voici les points clés à retenir de cette séquence : nous devons lire les données directement de la source et nous devons faire attention aux données manquantes, notamment lors de la lecture.

3.C. Importer les données (Org-mode)

Dans cette séquence, nous montrons comment réaliser un document computationnel et y importer nos données sur l'incidence du syndrome grippal.

Mais d'abord, il y a des choix techniques à faire. Nous utilisons l'environnement Emacs avec Org-mode et nous utilisons deux langages de programmation : Python 3, pour la phase de préparation des données, et le langage R, pour son analyse.

Voici notre document computationnel, où, pour l'instant, il n'y a que le titre. La première partie de notre travail, c'est le téléchargement des données à partir du site du réseau Sentinelles. Pour cela, nous avons besoin de l'URL de téléchargement. Nous la récupérons dans notre navigateur que nous avons utilisé pour télécharger ces fichiers précédemment. Nous retrouvons, dans l'historique de notre navigateur, l'URL, que nous copions. Et maintenant, nous pouvons coller cette URL dans notre document computationnel. Nous allons lui donner un nom : "data-url". Et par ce nom, Org-mode pourra identifier cette valeur et la passer à du code, comme nous verrons tout de suite. Voici notre premier bout de code, qui fait le téléchargement en langage Python. Nous voyons ici comment Org-mode passe l'information, dans notre cas, l'URL, à ce bloc de code. Donc, en Org mode, nous avons donné l'étiquette "data-url" à notre URL et, dans le code Python, le nom sera "data_url" pour respecter la syntaxe des noms de variables Python. Sinon, nous avons dit ici qu'il s'agit du code en langage Python et que nous ne voulons pas afficher les résultats, parce qu'en fait, ce code n'a pas de résultats. Nous l'exécutons avec deux fois Ctrl+C à l'intérieur du code. Rien ne se passe, mais c'est normal. Le deuxième bout de code sert à afficher les premières lignes de notre tableau, pour qu'on puisse l'inspecter. Je l'exécute, voici les résultats. La différence par rapport au premier bloc de code, c'est qu'ici, nous avons indiqué "results value", ce qui veut dire que nous voulons afficher comme résultat de notre code celui qui est donné par l'expression dans le code. C'est une expression Python, "table" avec un index, et cette valeur est affichée par la suite. Org-mode fait la traduction entre une liste de valeurs Python et un tableau joliment affiché, comme vous voyez ici. Donc, nous voyons ici que les cinq premières

lignes sont l'en-tête, avec les noms de colonnes, suivi par les quatre premières lignes du tableau, les dernières valeurs chronologiques de notre suite d'observations. Le prochain bout de code s'occupe de l'extraction des deux colonnes qui nous intéressent à partir de ce tableau. La première colonne est la colonne "week", qui contient, comme nous savons, l'année et le numéro de semaine en format ISO, et nous irons plus tard décoder cette information. Pour l'instant, nous faisons l'extraction de la première colonne de chaque ligne, et après, nous vérifions que l'en-tête est réellement ce qu'il faut, que c'est vraiment "week", et donc, nous le supprimons. Nous faisons pareil pour la colonne "inc", qui contient les valeurs d'incidence, et nous composons un nouveau tableau qui ne contient que ces deux colonnes. Voyons ce que ça donne. Nous n'avons pas de résultat, ce qui est normal. Ça veut dire que les messages d'erreur qu'auraient pu faire les "assert" ne se sont pas produits. Passons à la prochaine étape, où nous affichons un peu plus joliment un résumé de l'information que nous avons. Ici, nous avons les deux colonnes qui nous intéressent. Le code Python n'a rien d'exceptionnel, mais il illustre comment on passe l'information de Python vers Org-mode pour faire un joli affichage. Le résultat est une liste, chaque élément de la liste correspondant à une ligne du tableau qui sera affiché. Je commence avec l'en-tête, qui sont les valeurs "week" et "inc". Après, il y a le "None", qui est l'indicateur pour la ligne horizontale. Après, j'ai les cinq premières lignes de données, encore une ligne horizontale, et les cinq dernières lignes de données. Donc, nous voyons ici deux parties du tableau qui montrent le début et la fin de notre suite chronologique, qui est en ordre chronologique inverse, car elles commencent avec les dernières données et se terminent avec les premières. Continuons. En fait, nous savons que les semaines sont indiquées par ces chiffres, donc il faut bien qu'il y ait six caractères et que ce soit des chiffres. Nous le vérifions ici. Si ce n'est pas le cas, nous affichons la valeur suspecte pour qu'on puisse regarder ce qui s'est passé. Pour la colonne "inc", nous n'avons rien d'autre à vérifier que le fait qu'il s'agit de chiffres. Donc, pareil, nous affichons un message si jamais il y a un problème. Exécutons ce petit bout de code. Il y a effectivement une valeur suspecte. Nous le savions déjà, il y a une valeur manquante pour la semaine 19 de l'année 1989, où la valeur manquante de la colonne "inc" est remplacée par un tiret. Continuons notre analyse en éliminant ce point inutile, gênant. Nous faisons simplement un filtrage des données et nous créons un nouveau tableau qui ne contient plus la ligne manquante.

Voici les points clés à retenir de cette séquence. Premièrement, nous devons lire les données directement de la source, sans intervention manuelle. Deuxièmement, nous devons nous occuper de la gestion des données manquantes.

4. Vérification et inspection

4.A. Vérification et inspection (Jupyter)

Dans cette séquence, nous commençons par vérifier et inspecter nos données pour être sûrs qu'il n'y ait pas d'anomalies dues aux données ou à notre importation.

Avant de pouvoir travailler avec nos données avec Pandas, nous devons adapter la représentation des semaines. Dans les données que nous avons récupérées, il y a, pour chaque semaine, un entier à six chiffres. Par exemple, ici à la fin, 198501, ce qu'il faut lire comme 1985, semaine 01. Nous avons donc quelque chose comme ça que nous devons décomposer. La première étape, c'est de convertir cet entier en chaîne de caractères. Maintenant, nous pouvons décomposer en quatre chiffres et deux chiffres. Et convertir la chaîne de caractères de nouveau en entier. La même chose pour la semaine. Regardons si c'est bon. Ça en a l'air, mais ce n'est pas encore terminé. Ici, nous demandons à la bibliothèque ISOweek d'interpréter cette combinaison en date. Et nous récupérons la première journée de cette semaine. Ici, c'est la semaine qui commence le 31 décembre 1984. Pandas s'attend à trouver quelque chose de cette nature, un objet de type période construit à partir d'une date et d'une longueur. La longueur W veut dire une semaine qui commence par un dimanche. Cette formule "pd.Period" etc. veut dire : "Nous voulons la période qui correspond à la journée qui est contenue dans la première semaine de 1985 et d'une longueur d'une semaine." Ça donne ça. Maintenant, il faut faire tout ça pour chaque ligne du jeu de données. Pour ça, nous allons en faire une fonction Python. Maintenant, nous devons l'appliquer à chaque ligne de notre jeu de données. Et le résultat devient une nouvelle colonne du jeu de données. Avec cette information en plus, Pandas peut travailler sur nos données. Mais il est pratique de faire d'abord une version triée, car les données sont dans l'ordre inverse. Nous demandons à Pandas d'utiliser la période fraîchement créée comme index et de faire un tri en utilisant cet index. Regardons le résultat. Maintenant, nos données commencent en 1985, avec la première semaine, et se terminent en 2017. Nous pouvons maintenant commencer la vérification. Il est toujours bon de vérifier si les données sont cohérentes. Ici, nous pourrions vérifier que chaque période du jeu de données commence juste après la fin de la période précédente. Faisons ça dans le code. Récupérons d'abord la liste des périodes. Et nous faisons une boucle sur les paires adjacentes aux périodes. À l'intérieur de cette boucle, nous devons vérifier si la période "p2" commence juste après la fin de la période "p1". Ceci est la différence temporelle entre les deux périodes. Maintenant, nous vérifions si cette différence n'est pas trop grande, par exemple, n'excède pas un seuil

d'une seconde. Si elle dépasse une seconde, nous affichons les deux périodes pour inspection. Nous retrouvons notre semaine de 1989 dont nous savons qu'elle manque dans le jeu de données. Mais il n'y a pas d'autres problèmes, donc on suppose que tout va bien. On laisse un petit commentaire pour le lecteur que nous sommes au courant du problème. Ensuite, on fait un plot pour regarder nos données. Nous nous intéressons à la colonne "inc" pour l'incidence du syndrome grippal. Nous demandons un plot. Ah, ça se passe mal... Il semble que nos données ne soient pas numériques. Qu'est-ce qui se passe ? Voyons. En effet, ce n'est pas un entier, mais une chaîne de caractères. Peut-on l'expliquer ? Oui. Retournons à notre fichier de départ. Il y avait ici une ligne avec un trait à la place d'un chiffre. Pandas a lu toute cette colonne et dit: "Il y a un trait, je ne peux pas représenter ça par un entier, donc cette colonne sera représentée en chaîne de caractères." Maintenant que nous connaissons l'origine du problème, nous allons la documenter pour le lecteur. Du coup, nous pouvons maintenant faire la conversion de chaînes de caractères en entiers car le trait a disparu. Et nous pouvons tenter une autre fois notre plot. Voici enfin nos données. Nous voyons une oscillation avec des hauts et des bas, beaucoup de pics. On voit plus facilement ce qu'il se passe avec un zoom, donc nous allons regarder simplement les dernières années. Ici, nous voyons mieux qu'il y a des pics en hiver, autour de février, et une incidence de syndrome grippal très faible le reste de l'année. Nous le savons, c'est le phénomène de l'épidémie de grippe en hiver.

Voici les points clés à retenir de cette séquence. Nous avons fait un prétraitement des données pour les adapter aux conventions du logiciel. Ici, il fallait traduire la représentation des semaines du Réseau Sentinelles en quelque chose que Pandas comprenne. Ce prétraitement facilite aussi notre analyse. Après, nous avons fait une seule vérification. On peut, dans un cas plus réaliste, en faire d'autres. Cette vérification comprend l'inspection visuelle, un outil très puissant, mais aussi du code de validation. Par la suite, nous allons voir comment tirer des informations utiles de ces données.

4.B. Vérification et inspection (RStudio)

Dans cette séquence, nous préparons nos données pour l'analyse. Celle-ci implique une inspection et une vérification des données.

La tâche principale de la préparation des données, c'est la transformation des dates que nous avons récupérées du réseau Sentinelles dans un format que le langage R peut traiter plus facilement. Nous utilisons pour ceci une bibliothèque supplémentaire qui s'appelle "parsedate" et que nous devons d'abord charger. Nous avons, dans nos données, des dates de la forme

suivante : il s'agit d'entiers de six chiffres dont les quatre premiers chiffres représentent l'année et les deux derniers, la semaine, au format ISO, qui est définie telle que la première semaine d'une année est celle qui contient le 4 janvier. Je ne montrerai pas à pas comment on fait cette conversion. La première chose à faire est de convertir cet entier en chaînes de caractères. Donc, pour construire notre nouvelle chaîne de caractères, nous allons commencer avec le morceau représentant l'année. Après, nous allons coller le "-W" et pour finir, les deux chiffres qui correspondent à l'année. Nous avons le format demandé. Maintenant, nous pouvons faire appel à la fonction "parse_iso_8601" qui est le numéro de la norme ISO, pour obtenir une date. Si vous regardez bien, vous voyez l'année suivie par "01-02", c'est le 2 janvier. Donc, dans la conversion, la bibliothèque parsedate a silencieusement supposé un jour, car ce n'est pas une semaine, mais un jour précis. Le jour choisi par défaut est le lundi. Nous obtenons la date du lundi qui correspond à la semaine indiquée dans notre jeu de données. Maintenant, ce qu'il reste à faire, c'est appliquer cette procédure à chaque ligne du jeu de données. Pour cela, nous allons faire une fonction. Il ne nous reste plus qu'à appliquer cette fonction à chaque ligne de notre jeu de données et mettre le résultat dans une nouvelle colonne de notre jeu de données, que nous allons appeler "date". C'est ce qu'on aimerait bien faire idéalement. Ceci ne fonctionne pas si simplement à cause d'une particularité du langage R. Si nous appliquons notre fonction à chaque élément du jeu de données avec la fonction "sapply", le retour sera automatiquement converti en entier. Cela ne nous convient pas du tout. Pour contourner ceci, je transforme d'abord les résultats en chaînes de caractères dans la fonction, puis, ici, je convertis le résultat en date. Je vais bien vérifier que maintenant, j'ai bien des dates. Ça s'est bien passé. La deuxième étape dans la préparation des données, c'est de trier les données pour qu'elles soient dans l'ordre chronologique. Classées par la date. Vérifions. Les données commencent bien en 1985. Maintenant, nous pouvons continuer avec quelques vérifications. Si tout va bien, la distance entre deux dates voisines du jeu de données est exactement une semaine, donc sept jours. Voyons. Ça s'est bien passé. Nous pouvons maintenant jeter un coup d'œil sur nos données en affichant un plot. Sur l'axe x, la date, sur l'axe y, l'incidence. Tout ça affiché comme ligne. Ce n'est pas si mal que ça. Nous voyons bien qu'il y a des creux et des pics. C'est un peu serré, donc nous allons refaire la même chose avec un petit zoom. Nous allons copier et appliquer seulement aux 200 derniers points, soit à peu près quatre ans. Ici, nous voyons bien le comportement des incidences du syndrome grippal. Comme on le sait par expérience, les pics de l'épidémie se situent en hiver, donc

plutôt janvier, février. Les étiquettes des années correspondent au 1er janvier. Nous avons exactement le phénomène attendu.

Voici les points clés à retenir de cette séquence. Nous avons fait un pré-traitement des données dont l'objectif est de les adapter aux conventions utilisées pas les logiciels pour faciliter l'analyse qui va suivre. Nous avons également fait quelques vérifications par l'inspection visuelle, qui est un outil très puissant, mais aussi en écrivant du code de validation.

4.C. Vérification et inspection (Org-mode)

Dans cette séquence, nous continuons la préparation de nos données, notamment par la vérification et l'inspection.

La prochaine étape de la préparation des données est la conversion des dates. Nous avions déjà évoqué la difficulté qui résulte du fait que le réseau Sentinelles utilise le format ISO pour indiquer des semaines, ce qui est relativement inhabituel et notamment utilisé presque exclusivement en Europe. Très peu de logiciels savent traiter ce format, malgré le fait qu'il soit standardisé. Si nous avons choisi le langage Python 3 pour démarrer la préparation de nos données, c'est parce que depuis la version 3.6, la bibliothèque standard de Python peut gérer ces formats de données. C'est ce que nous faisons ici. Nous faisons une simple conversion de dates en langage Python, comme c'est toujours fait dans ce langage. Ce qu'il faut regarder dans la documentation, c'est que c'est bien le "%G", qui veut dire année, format ISO, et le "%V", qui veut dire semaine, format ISO. Comme il faut indiquer une date et non pas une semaine, il faut ajouter le jour. Nous ajoutons le 1, ici, pour indiquer le lundi. Donc, nous remplaçons chaque semaine par la date qui correspond au lundi de cette semaine. Par la même occasion, nous faisons un tri par ordre chronologique pour que nos données ne soient plus dans l'ordre inverse. Comme d'habitude, nous allons exécuter ces petits bouts de code. Comme nous avons indiqué, là, les résultats silencieux, nous ne voyons a priori rien. Nous passons donc à la prochaine étape. Il s'agit de préparer nos données juste pour les afficher partiellement dans le document Org-mode. Je vais d'abord l'exécuter pour voir le résultat. Le principe est le même qu'avant. Vous reconnaissez probablement la structure. On retrouve une liste avec des colonnes. Ici, les en-têtes, la ligne horizontale, la première partie, cinq lignes du début, une ligne horizontale et encore cinq lignes de la fin. Donc, c'est le même système qu'avant. Maintenant, nous voyons que nos données commencent fin 1984, car la première semaine de 1985 est celle qui inclut le 4 janvier, par définition. Donc, le lundi tombe sur le 31

décembre 1984. Nos données se terminent à la fin, en 2017. Ça a l'air tout à fait correct. Nous pouvons continuer notre analyse. Ici, nous passons par une autre vérification. Comme il s'agit de données hebdomadaires, a priori, nous avons une suite de dates de lundis. La différence entre chaque date et la date précédente doit être exactement une semaine, donc sept jours. C'est ce que vérifie ce bout de code en Python. La partie critique est ici. Il y a la comparaison "date1" et "date2" qui est comparée avec la distance d'une semaine. Voyons ce que ça donne. Il y a un cas où ce n'est pas le cas, un point où ce n'est pas le cas. Il y a 14 jours entre deux dates. C'est exactement le point manquant en 1989. C'est un problème dont nous sommes déjà au courant. Il n'y a pas de mesure supplémentaire à prendre. Pour la suite de notre analyse, nous passons au langage R. Nous vous montrons comment faire ceci en Org-mode. Il faut d'abord, pour le code qui fait l'exportation en Python vers Org-mode, donner un nom. Nous l'appelons data-for-R. Puis, nous donnons ce bloc de codes. Nous verrons tout de suite comment ce sera utilisé. Le seul but de ces bouts de code, c'est d'exporter les données Python dans un format qu'Org-mode comprend et qui peut, après, passer en langage R. C'est un peu comme nous avons fait pour afficher les tableaux plus tôt. Nous construisons une liste où chaque ligne correspond à une ligne de notre tableau de données. C'est pareil qu'avant. Nous avons l'en-tête, la ligne horizontale, etc. Nous formatons la date en "string" pour que ça passe en chaînes de caractères. La seule différence avec avant, c'est qu'on demande que les résultats soient silencieux. Sinon, on aura un tableau de 1700 lignes affiché dans le document. Ce n'est pas très utile. Alors, allons exécuter notre petit bout de code. La deuxième partie du transfert, c'est le code R qui fait l'importation de ces données. Ici, nous avons un bloc de code en langage R qui prend une variable qui s'appelle, en R, "data" et en Org-mode, "data-for-R". Là, vous reconnaissez le nom que nous avons donné à ce bloc, ici. C'est donc à l'exécution de ces bouts de code R qu'Org-mode va chercher le résultat du bloc précédent et le fournir à R sous le nom de "data". Donc, ici, R fait appel à ce "data". Le code R fait comme première opération la conversion, encore, de la date, d'une chaîne de caractères en date R. Après, on affiche un petit résumé des données pour montrer que le transfert s'est bien passé. Allons-y. Voilà. Les informations statistiques ne sont pas très utiles, mais nous donnent l'information que, globalement, ce qu'il s'est passé est tout à fait crédible. Là, j'ai montré comment faire le passage d'informations entre deux langages par Org-mode, donc par des structures de données en Emacs Lisp, en fin de compte. Une autre façon de faire est possible dans tout autre environnement, c'est d'écrire en Python un fichier Texte, format csv ou proche, et de le relire en langage R. À vous de choisir ce qui vous convient le mieux. Nous allons maintenant

afficher un "plot" de nos données en langage R, ce qui est bien plus simple qu'en langage Python. Voilà le code, qui n'a rien d'exceptionnel. Ce qui peut être intéressant à regarder, c'est qu'ici, nous fournissons un nom de fichier. Donc, R va produire le "plot" sous forme de fichier graphique dans le fichier de ce nom, et Org-mode va nous l'afficher après, ici, directement. Nous devons aussi, pour que ça marche, indiquer ici qu'on veut le résultat sous forme graphique. Alors, allons l'exécuter. Voilà notre "plot". Nous voyons bien que nos données ont une structure saisonnière. Ça a l'air d'être un pic par an. C'est un peu serré. On ne peut pas bien le voir. Nous allons faire un zoom. C'est le même type de "plot", mais seulement sur les 200 dernières données, donc à peu près quatre ans. Là, nous voyons bien qu'il y a des pics en début d'année, plutôt janvier, février, ce qui correspond tout à fait à ce qu'on sait du syndrome grippal de notre expérience de la vie quotidienne.

Voici les points clés à retenir de cette séquence. Nous avons effectué un prétraitement des données, notamment pour adapter les données aux conventions qu'aiment bien nos logiciels, spécifiquement concernant la représentation des dates. Ceci facilite évidemment l'analyse qui va suivre. Nous avons profité de l'occasion pour faire des vérifications. Il est toujours bon de vérifier autant que possible. Ceci implique l'inspection visuelle par les "plots" que nous avons faits, mais aussi du code de validation, comme la vérification de la distance entre deux dates, ce qui vérifie qu'il s'agit bien des distances d'une semaine.

5. Questions et réponses

5.A. Questions et réponses (Jupyter)

Dans cette section, nous allons enfin poser des questions à nos données et obtenir des réponses par des calculs simples.

La première question que nous allons poser est : Dans quelles années y avait-il les épidémies les plus fortes ? Après, nous allons élargir cette question à la fréquence des épidémies : faible, moyenne, forte. Pour voir comment ça se comporte au cours des années. Nous souhaitons analyser la variation des épidémies d'une année à l'autre, plutôt qu'une variation hebdomadaire détaillée. Nous devons additionner les incidences hebdomadaires année par année. Ceci n'est malheureusement pas si facile parce qu'une année ne contient pas un nombre entier de semaines.

Voici notre plan. Nous trouvons d'abord pour chaque année la semaine contenant le 1er août. Puis, nous utilisons ces semaines comme début d'année dans notre sommation des

incidences. En choisissant les limites des sommations dans le creux de l'été, nous minimisons les biais introduits par notre analyse. Le 1er août en langage Pandas s'écrit comme ça : "pd.Timestamp(2000, 8, 1)". Ici, pour l'année 2000. La semaine qui contient le 1er août et la période qui contient cette date, ayant la longueur d'une semaine. Maintenant, nous devons appliquer ce calcul à toutes les années de notre jeu de données. Nous introduisons une variable et nous faisons une boucle sur un intervalle d'année, dont le premier est l'année de la première ligne de notre jeu de données et la dernière correspond à la dernière année. Maintenant, nous devons faire une boucle sur toutes les paires de valeurs adjacentes dans cette liste de semaines qui correspondent au 1er août de chaque année. À l'intérieur de cette boucle, nous pouvons maintenant extraire les données correspondant à l'année entre les deux semaines du 1er août. Nous commençons avec la première semaine et nous nous arrêtons avant la deuxième. Une petite remarque pour ceux qui connaissent bien le langage Python. On s'attendrait ici à pouvoir écrire "Week 2" sans le "-1" derrière. Malheureusement, Pandas ne respecte pas cette convention très répandue dans le langage Python. Voici une bonne occasion de faire une autre vérification. Nous savons que notre année du 1er août au 1er août ne contient pas exactement 52 semaines mais entre 51 et 53. Vérifions. Nous insérons une commande qui crée un message d'erreur si la condition qui suit n'est pas satisfaite. Là, nous calculons la déviation de la longueur de notre année de 52. Si cette valeur est inférieure à 2, donc 0 ou 1, tout va bien. Dans l'autre cas, cela nous crée une erreur. Ce qu'il nous reste à faire, c'est de récupérer nos incidences annuelles. Nous créons une liste à laquelle nous rajoutons, dans chaque itération de notre boucle, la somme de nos données d'incidence. Il reste une chose à faire, c'est de garder pour chaque valeur annuelle une étiquette avec le nombre, le numéro d'année qu'il faut associer à ces données. Et là aussi, nous devons commencer avec une liste vide. Maintenant, nous avons deux listes mais il est plus pratique de les convertir en objets série de temps Pandas, pour pouvoir travailler avec. Nous pouvons de nouveau afficher un plot où nous allons mettre une petite étoile pour chaque année. Sur ce plot, nous voyons déjà où sont les épidémies les plus fortes. Nous avons trois étoiles ici, dans ce coin. Pour une meilleure précision, nous souhaitons obtenir des valeurs numériques, et la façon la plus facile d'y arriver est d'afficher les valeurs triées. Nous voyons donc que les épidémies les plus fortes correspondent aux années 1986, 1990 et 1989. Il nous reste à répondre à notre deuxième question qui élargit la question des épidémies fortes à une distribution générale de l'ampleur des épidémies. Nous allons faire ça en affichant un histogramme. Nous retrouvons ici les épidémies les plus fortes, il y en a trois, dont la valeur

oscille autour de 5 millions, 5,5 millions. Nous voyons aussi que la plupart sont beaucoup moins fortes que ça. Le pic de notre histogramme est plutôt à la moitié. Donc, nous avons eu affaire à des épidémies très fortes dans ces années. Pour présenter une telle analyse à un lecteur, il faut ajouter plus d'explications et de commentaires. Nous ne le faisons pas ici mais nous vous proposons le document nettoyé, rédigé, amélioré, comme cas d'étude dans notre MOOC.

Voici les points clés à retenir de ce module. Premièrement, une analyse réplicable doit contenir toutes les étapes de traitement des données sous une forme exécutable. Il est important d'expliquer tous les choix pouvant influencer les résultats. Ceci nécessite d'exposer beaucoup de détails techniques, comme, dans notre cas, la conversion des dates, ou la vérification des données, mais c'est à ce niveau qu'on fait le plus d'erreurs. C'est pour ça qu'il est important, pour la traçabilité et la transparence des calculs, de ne pas cacher ces informations au lecteur, même si, pour la compréhension du résultat, ceci n'est pas important.

5.B. Questions et réponses (RStudio)

Dans cette séquence, nous allons poser des questions à nos données et obtenir des réponses. La première question est : dans quelles années y avait-il les épidémies les plus fortes ? Nous élargirons cette question et regarderons la distribution de l'ampleur des épidémies au cours des années.

Ceci nécessite de regarder une incidence annuelle plutôt qu'hebdomadaire, qui nous livre nos données. Nous devons donc faire une conversion, une sommation sur l'année. Ce n'est malheureusement pas si facile, car une année ne comporte pas un nombre entier de semaines. Nous devons donc décider comment faire exactement cette sommation. Nous devons, à cette occasion, aussi tenir compte du fait que les pics de l'épidémie se trouvent en hiver, les creux en été. Il n'est donc pas une bonne idée de faire la sommation sur une année civile. On commencerait de janvier à janvier, ce qui commencerait la sommation au milieu d'un pic. Nous allons donc définir notre année comme allant du 1er août au 1er août. Concrètement, nous définissons les pics annuels pour une année donnée. Disons, pour l'exemple, 1990. La date de début de notre sommation est donc construite à partir de l'année moins 1, à laquelle on attache le 1er août. La date de la fin de sommation, c'est presque pareil, sauf qu'on prend l'année sans soustraire 1. Les semaines sur lesquelles il faut réaliser la sommation sont données par les dates qui sont supérieures au début et inférieures à la fin. Nous pouvons enfin faire notre sommation sur la colonne des incidences pour les semaines que nous venons de

sélectionner. Nous devons encore préciser un paramètre magique, qui dit à R de simplement ignorer les valeurs pour lesquelles il n'y a pas de données. Donc concrètement, la semaine de 1999 numéro 19 pour laquelle il n'y a pas de données. Nous allons sous-estimer le pic de l'année 1989. Par la suite, nous devons l'appliquer à toutes les années de notre jeu de données. Mais il faut encore réfléchir sur les années qu'on peut intégrer dans notre calcul. Car pour la première année, en 1985, nos données commencent le 1er janvier ou la première semaine de janvier. C'est donc déjà au début du pic et comme nous n'avons pas les données de 1984, ça risque de donner un faux résultat. Il vaut mieux ne pas tenir compte de l'année 1985 et commencer notre calcul en 1986. À l'autre bout, en 2017, nos données se terminent en été, près du creux. Là, il n'y a aucun problème. Nous pouvons tout à fait calculer jusqu'à 2017. Nous allons créer un nouveau tableau, "incidence annuelle", avec la colonne "année" et la colonne "incidence" qui va être calculée en appliquant notre nouvelle fonction. Voilà. Nous allons regarder ce que ça donne. Nous avons une valeur 1986, pour 1987. Ça a l'air tout à fait raisonnable. Faisons encore un "plot" avec des points, pour changer. Nous voyons déjà très bien ce qu'il se passe. Nous avons ici trois valeurs très élevées et les autres sont bien moins importantes. Pour obtenir ces valeurs numériques, je vais appliquer un tri sur la colonne "incidence". Nous voyons que le pic le plus important est en 1989, le suivant en 1990 et le troisième en 1986, ce qui donne une réponse à notre première question. Pour terminer, il nous reste à examiner la distribution des pics annuels, ce qui est relativement simple, car le langage R a tout ce qu'il faut. Nous allons simplement afficher un histogramme et nous allons demander dix catégories. Voilà, tout à droite, nous avons de nouveau les trois pics les plus importants, qui sont autour de cinq millions, environ 10% de la population française, pour donner un ordre de grandeur. Mais ces épidémies fortes sont relativement rares. Vous voyez que la plupart des épidémies sont bien plus faibles. Le pic se situe plutôt autour de la moitié de ce maximum. Avec ça, nous avons terminé notre analyse. Nous donnons, comme matériel supplémentaire, sur FUN, une version annotée, nettoyée, avec des plus jolies étiquettes. Mais l'essentiel du travail est fait.

Voici les points clés à retenir de ce module. Une analyse réplicable doit contenir toutes les étapes de traitement des données sous une forme exécutable. Il est important d'expliquer tous les choix qui peuvent influencer les résultats. Par exemple, nous avons expliqué pourquoi nous commençons en 1986 pour la sommation sur les années. Ceci nécessite d'exposer un grand nombre de détails techniques et ça peut paraître superflu ou même gênant, mais c'est à ce niveau qu'on peut faire le plus d'erreurs. Si le lecteur doit être capable de vérifier notre

calcul, il doit avoir accès à tous ces détails. Par exemple, notre gestion de dates, qui a pris un certain nombre d'étapes de calcul, est tout à fait un endroit où on peut facilement se tromper et ce doit être vérifiable par les lecteurs.

5.C. Questions et réponses (Org-mode)

Dans cette séquence, nous posons enfin des questions à des données pour obtenir des réponses.

Concrètement, la première question que nous posons, c'est : dans quelles années y avait-il les épidémies les plus fortes ? Puis, nous élargirons cette question à la distribution de l'ampleur des épidémies. Il y a des épidémies faibles, moyennes et fortes. Comment se distribuent-elles dans les années ?

La tâche principale qu'il reste à faire est la conversion des données hebdomadaires en données annuelles. Nous devons calculer l'ampleur du pic que nous voyons dans notre plot pour chaque année. Ceci n'est pas difficile en principe, mais il y a quelques subtilités, notamment dues au fait qu'une année ne contient pas un nombre entier de semaines. Un autre problème est qu'il n'est pas judicieux de choisir l'année civile comme période de sommation. Car le pic est tout au début de l'année. Donc, il vaut mieux définir un pic entre deux creux. Les creux se trouvent en été. Donc, nous décidons de découper notre suite chronologique en années aux dates du 1er août, qui correspond à peu près au creux. Pour ceci, nous définissons une fonction "r" qui fait ce travail pour une année donnée. On lui fournit une année, comme 1990. On définit comme début de cette période le 1er août de l'année précédente. C'est l'année moins 1. Comme fin de la période, le 1er août de l'année concernée. Par la suite, on applique un filtre pour extraire les semaines qui sont entre le début et la fin. Enfin, nous faisons la somme sur la colonne "incidence" de nos données. Pour les semaines que nous avons choisies par notre filtre, nous exécutons ce bloc qui ne fait, a priori, rien. Il fait juste la définition d'une fonction. Ce qui est plus intéressant, c'est l'utilisation. Mais avant de pouvoir l'utiliser, nous devons définir les années pour lesquelles nous faisons notre calcul. On pourrait penser que c'est simplement la première année jusqu'à la dernière. Mais nous devons faire attention à cause des pics situés en hiver et des creux situés en été. Nos données commencent le 1er janvier 1985. On est déjà dans le pic de 1985 et on risque de le sous-estimer sérieusement. Il vaut mieux ne pas le faire, donc on commence en 1986. Nous n'avons pas le même problème à la fin, car nos données se terminent en juillet 2017. Donc, nous continuons jusqu'en 2017. Ceci est l'occasion d'insister sur le fait que ce genre de choix doit être documenté pour le

lecteur. Ici, je ferai court, mais la version améliorée, rédigée, que vous trouverez dans le matériel de cours donnera des commentaires plus détaillés. Voilà. Ça, c'est juste un pense-bête pour la suite. Donc, nous avons défini ici les années pour lesquelles nous allons faire le calcul. Il reste à le faire. Voici le bout de code qui le fait. Nous construisons un nouveau tableau en R avec une colonne "année", où on a simplement le numéro des années, et une deuxième colonne "incidence" qui est calculée comme application de la fonction "pic_annuel" à chaque valeur de notre tableau d'années. Nous affichons tout de suite le début de ce fichier pour voir à quoi ça ressemble. Nous avons la première valeur pour 1986, après 1987. Ça a l'air tout à fait crédible, ce qui veut dire que nous pouvons continuer avec un autre "plot" qui montre l'incidence annuelle. Nous avons choisi des points plutôt que des lignes pour varier. Nous voyons en haut à gauche les valeurs qui sortent un peu de l'ordinaire. Trois épidémies ont été plus fortes que les autres. On voit sur le "plot" que c'est à peu près au début des jeux de données, mais on aimerait bien voir ça un peu plus précisément. Nous allons faire une extraction un peu plus numérique. Nous trions nos données par ordre décroissant de l'incidence annuelle et nous l'affichons ici. Donc, la valeur la plus élevée correspond à l'année 1989, suivie par 1990 et 1986. Ce sont les trois épidémies les plus fortes, identifiées sur notre "plot". Après, on tombe sur des valeurs clairement plus inférieures. Nous pouvons encore, pour répondre à la deuxième question, regarder la distribution des données, ce que le langage R fait pour nous. Nous affichons un histogramme de l'incidence annuelle calculée et nous voyons ici, qu'à droite, il y a un pic de hauteur 3 qui regroupe les trois épidémies les plus fortes touchant cinq millions de personnes, environ 10% de la population française. On voit aussi que l'épidémie moyenne est bien plus faible, plutôt la moitié. Mais il y a aussi quelques épidémies qui touchent entre 3,5 millions et 4 millions de personnes en France, chaque année. Avec ça, nous pouvons conclure notre analyse très simple. Nous avons insisté ici plus sur le document computationnel que sur les calculs. Il reste à mettre en forme ce document, ajouter des commentaires, une structuration, ce qui est très facile avec Org-mode. Nous pouvons commencer à dire, par exemple "Préparation des données", puis, nous pouvons passer à "Vérification". Par exemple, ici, on fait une première vérification. On peut aussi marquer... Ici, c'est la partie très intéressante de la conversion des dates, et ainsi de suite. Nous vous fournirons une version commentée dans le matériel de notre cours.

Voici les points clés à retenir de cette séquence. Une analyse réplicable doit contenir toutes les étapes du traitement des données sous une forme exécutable. Il est important d'expliquer tous les choix qui peuvent influencer les résultats. Ceci nécessite d'exposer beaucoup de

détails techniques. Dans notre cas, notamment, la gestion des dates et des points manquants. Mais il est important de les exposer, car c'est à ce niveau qu'on fait le plus d'erreurs. Si on supprime ces informations, le lecteur ne peut plus identifier ces erreurs.