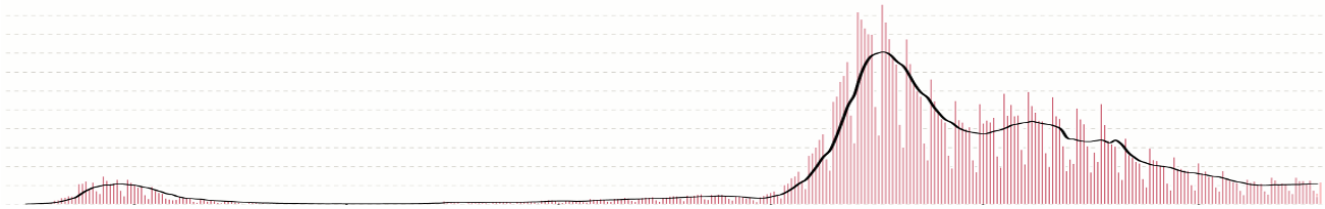
Data Wrangling (daw)
Cédric Huwyler

Fachhochschule Nordwestschweiz

Minichallenge: Covid-19 - Datenaufbereitung



1 Ausgangslage

Eine Kommission beim Bund möchte untersuchen, ob die Schweiz während der Corona-Pandemie richtig vorgegangen ist und was insbesondere das Verbesserungspotential ist. Dabei soll auch mit den Strategien anderer Länder verglichen werden und vor allem auch, wie sich die Strategien dann auf die Fallzahlen ausgewirkt haben. Du bist neu Data Scientist beim Bundesamt für Gesundheit und sollst die entsprechenden Länderdaten in einer aufbereiteten Form zur Verfügung stellen und dein Vorgehen in einem Jupyter Notebook dokumentieren.

Du entscheidest dich dazu, die international erfassten Fallzahlen der John Hopkins - Universität zu verwenden. Die zwischen dem 22.01.2020 und 09.03.2023 erfassten Daten sind entweder direkt als Zip-Datei hier oder auf Github (link) verfügbar.

2 Aufgabenstellung

Der Senior Data Scientist rät dir, folgendermassen vorzugehen:

1. **CSV-Dateien einlesen**

   Lies alle CSV-Dateien in ein gemeinsames Dataframe ein und definiere aus dem jeweiligen Dateinamen eine entsprechende Datumspalte. Achte ein wenig auf Effizienz, da du möglicherweise die CSVs noch ein paarmal einlesen musst, wenn du Fehler bemerkst. Zum besseren Verständnis des Einlesefortschritt kann es Sinn machen, diesen mit einer Progress-Bar darzustellen. Gib am Schluss die Grösse deines Dataframes in MB aus.

2. **Relevantes Subset aus Spalten extrahieren**

   Extrahiere die für deine Problemstellung relevanten Spaltennamen: Datum, Land, Provinz, Anzahl bestätigte Fälle, Anzahl Todesfälle. Gib am Schluss wiederum die Grösse des resultierenden Dataframes in MB aus.

   Hinweise:

   - Die Spalte mit den Genesenen lassen wir weg, die scheint nicht sehr verlässlich zu sein.
   - Achtung, anscheinend hat sich über die Monate eine Konvention geändert, die Spalten für Land und Provinz kommen in zwei verschiedenen Varianten vor. Lasse vorerst mal beide Varianten im Dataframe, du wirst sie dann weiter unten zusammenführen.

3. **Datumformat anpassen**

   Das Datum liegt nach dem Einlesen normalerweise als Zeichenkette vor. Das braucht einerseits viel Speicher und andererseits möchtest du später auf datumsspezifische Funktionen zugreifen können. Konvertiere die Datumsspalte in ein Datumsformat und sortiere das Dataframe nach der Datumsspalte. Prüfe kurz, ob für jeden Tag Werte für mindestens ein Land erfasst sind (also ob alle Tage zwischen Start und Ende der Erfassung vorkommen). Gib zum Schluss wieder die Grösse des resultierenden Dataframes in MB aus - wieviel Speichergewinn hat die Konvertierung in das Datumsformat gebracht?

4. **Spaltennamen für Land und Provinz zusammenführen**

   Nun möchtest du die verschiedenen Versionen für Land und Provinz zusammenführen. Deine Hypothese ist, dass sich einfach irgendwann einmal im Verlauf der Datenerfassung die Konvention geändert hat. Verifiziere diese Annahme, indem nach Fällen suchst, wo beide der Spaltenversionen für entweder Land oder Provinz nicht-fehlende Werte enthalten (falls ja musst du deine Hypothese hinterfragen).

   Entscheide dich nun für die Version mit Underscore und fülle alle fehlenden Werte dieser Spalte wo möglich mit der anderen Version auf. Welcher Anteil der Werte in den kombinierten Spalten für Land und für Provinz fehlt am Schluss noch?

   Hinweis: Die Provinz muss nicht unbedingt immer befüllt sein (Bsp. Schweiz).

   Gib auch hier zum Schluss wieder die Grösse des resultierenden Dataframes in MB aus

5. **Provinzen aufaggregieren**

   Das Provinz-Level interessiert dich aktuell nicht und du entscheidest dich darum dazu, die Erkrankungs- und Todesfallzahlen nur noch summarisch per Land anzugeben und die Spalte für die Provinz loszuwerden. Führe die entsprechende Aggregation aus und gib wiederum die Grösse des Dataframes in MB aus.

6. **Datenfehler identifizieren**

   Gib eine Liste aller im Dataframe enthaltenen Ländernamen aus und inspiziere diese Liste genau. Es fällt auf, dass einige Länder in verschiedenen Versionen vorkommen. Beispiele: Vietnam vs. Viet Nam, Taiwan vs Taiwan*, South Korea vs. Korea, South vs Republic of Korea, The Bahamas vs. Bahamas, The, und so weiter. Füge diese und weitere Länder zusammen, indem du dich für einen Ländernamen entscheidest und die Erkrankungs- und Todesfallzahlen pro Tag jeweils aufsummierst.

7. **Kategorien setzen**

   Für jedes Land, dessen Bezeichnung als Zeichenkette vorliegt, kommt pro Datum in deinem Data Frame eine Zeile vor. Definiere der Effizienz halber das Feld 'Country_Region' als kategorische Variable und prüfe, wieviel weniger Speicher dein Data Frame nun braucht.

8. **Tägliche neue Fallzahlen berechnen**

   Die Erkrankungs- und Todesfallzahlen liegen in kumulativer Form vor, du bist aber mehr in der Anzahl der neuen Fälle pro Tag interessiert. Erstelle zwei neue Spalten 'Confirmed_Reported' und 'Deaths_Reported', indem du jeweils die Differenz zum vorhergehenden Tag berechnest.

   Die Daten enthalten einige sprunghafte Änderungen (aufgrund von Konventionsänderungen, Fehlern, etc.), was teilweise zu stark negativen Werten in der Anzahl der neuen Fälle führt. Ersetze als einfache erste Strategie einfach mal alle negativen Werte mit Null.

   Hinweise:

   - Versuche die Differenz mit möglichst kurzem Code zu berechnen. Damit sich die einzelnen Länder nicht überlappen, brauchst du dazu ein Groupby.
   - Fülle allfällige fehlende Werte der neuen Felder mit Null auf (für den ersten Tag gibt es keinen vorhergehenden Tag).

9. **Anreicherung mit Zusatzinformationen**

   Nur mit absoluten Zahlen ist es schwierig, einzelne Länder miteinander zu vergleichen, da sie enorm verschiedene Bevölkerungsstärken haben. Du möchtest daher gerne im Anschluss die Anzahl Fälle pro 10'000 Personen berechnen. Dazu brauchst du aber zuerst einmal die Bevölkerungszahlen der einzelnen Länder. Ausserdem möchtest du für die Visualisierung im letzten Teil gleich noch zu jedem Land den üblichen dreistelligen Ländercode hinzufügen.

   Python bietet für beide Probleme jeweils ein brauchbares Modul:

   - `pycountry` für die Identifikation eines dreistelligen Ländercodes mit einem Land (das hier als String übergeben werden kann und nicht perfekt matchen muss)
   - `pypopulation` für die Bevölkerungsgrösse eines Landes (über den vorher bestimmten dreistelligen Ländercode) im Jahr 2020. Natürlich hat sich die Bevölkerungszahl in den Jahren 2021, 2022 und 2023 geändert, wir nehmen der Einfachheit halber aber einfach die aus 2020, da sie wohl ungefähr gleichgeblieben ist.

   In R gibt es sicher auch entsprechende Bibliotheken, die du ausfindig machen kannst. Falls du keine vernünftige Alternative findest, kannst du diesen Teil der Minichallenge auch in Python lösen und das resultierende Dataframe dann wieder in R einlesen und joinen.

   Erstelle konkret ein Dataframe mit den Spalten 'Country_Region', 'Country_Code' und 'Country_Population', das die gesuchten Informationen für möglichst jedes Land enthält und joine es an dein Dataframe aus den vorhergenden Teilaufgaben.

10. **Berechnung relativer täglicher Fallzahlen**

    Berechne nun die relativen täglichen Neuerkrankungen und neuen Todesfälle **pro 10'000 Personen** der Bevölkerung im Land.

11. **Graphische Darstellung und Glättung**

    Stelle die zeitliche Entwicklung der Neuerkrankungen pro 10'000 Personen der Bevölkerung für die **Schweiz**, **Italien** und **Deutschland** über die ganze Zeitperiode in einem gemeinsamen Plot graphisch dar.

    Es fällt auf, dass die Darstellung zwischen den Tagen extrem stark schwankt und nicht wirklich interpretierbar aussieht. Erstelle daher zwei neue Spalten 'Confirmed_Reported_per_10000_MA14' und 'Deaths_Reported_per_10000_MA14', die den Inhalt der ursprünglichen Spalten (Neuerkrankungen und Todefallzahlen) enthalten, aber über 14 Tage mit einem **Moving Average** geglättet sind.

    Visualisiere die zeitliche Entwicklung noch einmal mit diesen neuen Spalten. Sieht die graphische Darstellung nun interpretierbarer aus? Visualisiere auch nur die Entwicklung während der ersten Welle zwischem März und Juni 2020.

12. **Freiwillig: Graphische Darstellung und Animation auf Chloropleth-Karte**

    Die zeitliche Entwicklung der Fallzahlen kann auch geographisch visualisiert werden. Dazu bietet sich ein *Chloropleth*, eine Karte, wo jedem Land eine individuelle Farbe passend zu seinen aktuellen Fallzahlen zugeordnet ist. Versuche zum Beispiel mit *Plotly Express* oder *Plotly* eine solche Karte für ein fixes Datum zu erstellen und überlege dir, wie du die Karte animieren könntest (siehe Beispiel in Abbildung 1).

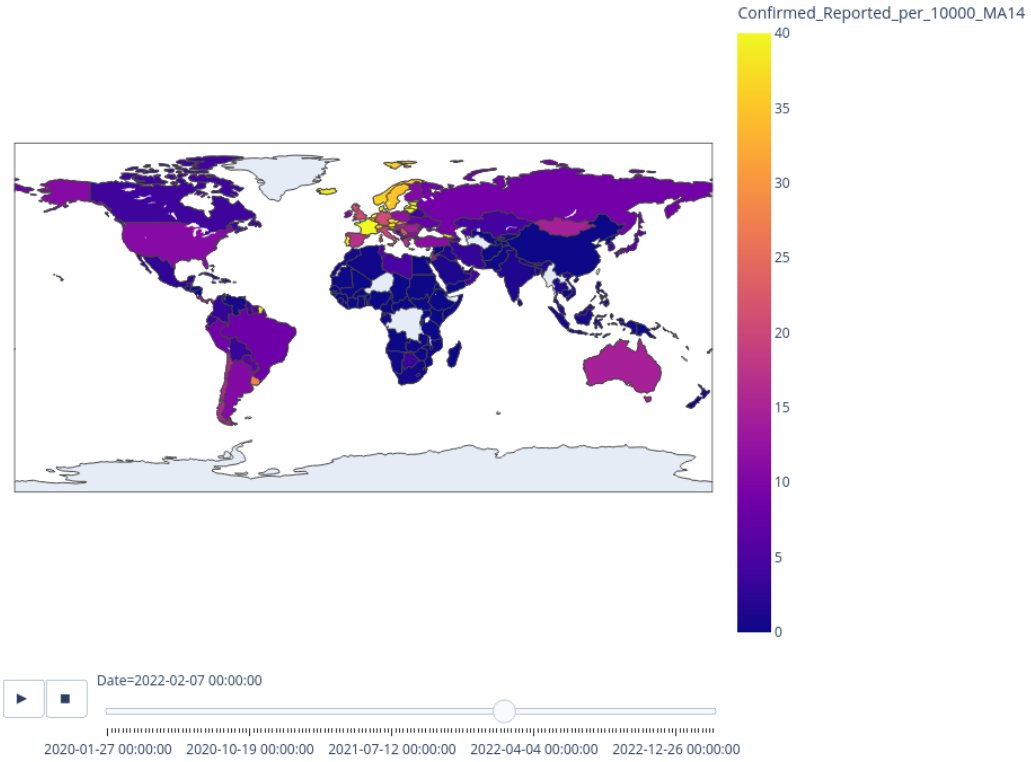


Abbildung 1: Beispiel einer Chloropleth-Animation mit Plotly-Express in Python