Auswertung der Corona-Daten der Stadt Bielefeld

Matthias Westenfelder

Januar 2022

1 Einleitung

Die Covid Pandemie steht seit gut 2 Jahren vollkommen im Mittelpunkt aller unserer Leben und ist ein tägliches Thema unseres Alltags geworden. Die Bekämpfung der Pandemie und die Entwickelung dieser, lassen sich nur akkurat erfassen, mit statistischen und Data-zentrischen Methoden. Ob ein Gesetz Wirkung in der Pandemiebekämpfung zeigt, ist primär eine Frage der Datenlage. Methoden um strukturiert und gezielt Daten zu sammeln, Maschinenlesbar sowie Maschinen-verarbeitbar zu machen liefert zum Beispiel der XML-Stack. Mit XML Dokumenten zusammen mit XSD als defakto-Datenbank, kombiniert mit XPath als Query Sprache und XQuery als stärkeres Rechenwerkzeug basierend auf XPath liefert dieser Technologiestack die Möglichkeiten all diese Aufgaben zu stemmen. Dies versucht diese Arbeit zu demonstrieren, anhand einer Auswertung von Corona-Daten der Stadt Bielefeld.

2 Parsen der Daten und Konstruktion des XML-Schemas

2.1 XML-Schema Konstruktion

Die Konstruktion des XML-Schemas für die Auswertung war vor allem motiviert durch die Komplexität des Parsers und die Rohdaten im HTML der Webseite der Stadt Bielefeld. Wenn man die Unregelmäßigkeiten in der Formatierung des HTML betrachtet, ist klar dass sich eine klare Struktur ab dem 12.05.2021 - 09.12.2021 (Auslese Zeitpunkt) herauskristallisiert. Die Webseite liefert in zufriedenstellendem Maße für jeden Tag in diesem Zeitraum nur Rohdaten zu der Meldeinzidenz (Infektionen pro 100000), Neuinfektionen (Anzahl der positiv Getesteten), Genesenen und der Anzahl der geschätzten Infektions-Träger in Bielefeld. Zusammengefasst, in diesem Zeitfenster lassen sich die Daten praktikabel mit Regex auslesen und es gibt genügend interessante Rohdaten um dies überhaupt zu rechtfertigen. Deshalb wurden die "Blätter" der XML-Grammatik beschrieben durch den Typ *day_data*.

```
<xs:element name="day_data">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="positiv_getestet" type="xs:integer"/>
      <xs:element name="geschaetzt_infektioes" type="xs:integer"/>
      <xs:element name="inzidenz" type="xs:decimal"/>
      <xs:element name="genesen" type="xs:integer"/>
    </xs:sequence>
    <xs:attribute name="date" type="xs:date" use="required"/>
  </xs:complexType>
</xs:element>
```

Wobei hier das Attribut *date* verpflichtend ist, da es der eindeutige Key für jeden Eintrag ist und die vorgefertigten Date-Module von XPath/XQuery gute Werkezeuge für das querien und sortieren dieser Daten liefern.

Der darauf aufbauende Typ *month_data* ist die denkbar einfachste nicht triviale Extension von *day_data*. Die Idee ist hier die Tage zu Monaten zu kombinieren und Mittelwerte/Varianz über Monate zu berechnen. Somit kann man Monatsweise Änderungen im Infektionsgeschehen statistisch beobachten. Aber um Statistiken besser interpretieren zu können und mögliche Fehler schneller zu finden ist es wichtig zu sehen für welche, und wie viele, Tage Rohdaten vorhanden sind. Dies motivierte die notwendigen Attribute *first_date* und *last_date*. Diese sind das Datum des ersten und letzten Tages des Monats, welche Rohdaten enthalten. Damit lässt sich wenigstens grob einschätzen ob es überhaupt genügend valide Daten gibt damit statistische Ergebisse von Relevanz sind. Für unvollständige Monate zeigt es wie "valide" ein Monat ist (siehe beigefügter Datensatz Dezember 2021 in welchem ca. 1 Woche an Daten sind).

```
<xs:element name="month_data">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="day_data" minOccurs="1" maxOccurs="31"/>
        </xs:sequence>
        <xs:attribute name = "first_date" type="xs:date"/>
        <xs:attribute name = "last_date" type="xs:date"/>
    </xs:complexType>
</xs:element>
```

Hier wäre besser gewesen als Attribut zusätzlich die Anzahl der validen Tage zu haben, auch eine kleinschrittigere Aufschlüsselung auf die Wochen-Ebene hätte hier geholfen, um die verschiedenen Ebenen des Infektionsgeschehen besser trennen zu können. Ich habe mich bewusst gegen das Einführen eines Wochen-Tags entschieden, da dies Ingenieursaufwand im Parsing bedeutet hätte welchen ich nicht sinnvoll in einem 30 Stunden Workload Budget rechtfertigen kann.

2.2 Parsing

Der Parser ist vollständig in Python3 entwickelt und für schnelles Prototyping mit einem Jupyter Notebook umgesetz worden. In einem Setting in welchem man die Daten updaten würde, würde man dieses Prototypen-Notebook durch ein Python-Modul ersetzen um es gegebenenfalls auf einem Webserver/RaspberryPi

laufen lassen zu können. Das Parsing des eigentlichen HTML's wurde vollständig mit dem Paket BeautifulSoup4 gestemmt. Das Auslesen der Daten aus dem Rohtext hingegen wurde größtenteils mit Regulären Ausdrücken implementiert. Dies geht natürlich Hand in Hand mit einigem an preprocessing und edgecase Arbeit, da die Corona-Meldungen der Stadt Bielefeld ganz offensichtlich größtenteils händisch eingetippt werden.

3 XQuery Konstruktion und Output-Formatierung

3.1 XQuery Konstruktion

Das XQuery Skript hat vor allem eine Hauptaufgabe, diese ist die Varianz und das Arithmetische Mittel des Datensatzes zu ermitteln. (Hier geht automatisch eine implizite Annahme ein, dass alle Zufallsvariabeln die wir untersuchen gleichverteilt sind, diese ist jedoch nicht unüblich in Situationen in denen man die Wahrscheinlichkeiten nicht sinnvoll schätzen kann.) Die Varianz hat essenziell die selbe Information wie die Standard-Abweichung. Varianz und Erwartungswert sind die wichtigsten Kennwerte einer Wahrscheinlichkeits-Verteilung. Zusätzlich relevante Kennzahlen sind Minima wie Maxima der Datenpunkte, welche aber schon durch `fn:min` sowie `fn:max` implementiert sind.

Die Varianz ist für ein Sample $(X_i)_{i=1}^N$ einer gleichverteilten Zufallsvariable X mit $\hat{\mu} = \frac{1}{N}\sum_{i=1}^N X_i$ definiert als

$$Var[X] = E[(X - E[X])^2] \approx \frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{\mu})^2$$

wobei hier der letzte Term der üblich benutzte Schätzer für die Varianz ist. Da XQuery eine Funktionale Programmiersprache ist, habe ich in der Implementierung die Varianz in 3 Funktionen zerlegt. Der Quellcode für die Berechnung der Varianz ist:

```
declare function local:variance($seq as item()*)
{
  let $mean := fn:avg($seq)
  let $norm_seq := local:pointwise_sub($seq,$mean)
  return fn:avg(local:pointwise_square($norm_seq))
};
```

Da das bilden des arithmetischen Mittels mit `fn:avg` schon implementiert ist, fehlt nur das punktweise abziehen eines Skalar x von einer Sequenz

```
declare function local:pointwise_sub($seq1 as item()*,$x as xs:double)
{
  for $el at $p in $seq1
  return $seq1[$p] - $x
};
```

sowie das punktweise quadrieren einer Sequenz.

```
declare function local:pointwise_square($seq1 as item()*)
{
  for $el in $seq1
  return $el*$el
};
```

Womit ein Schätzer für die Varianz auf unseren Daten implementiert wäre. Nun fehlt nur noch zu bemerken, dass die Standardabweichung σ definiert wird durch die folgende Identität:

$$\sqrt{Var[X]} = \sigma_X$$

was mit der in XQuery implementierten Funktion `math:sqrt` erreicht werden kann.

Students t-Test braucht ebenfalls nicht mehr als Mittelwerte, Anzahl der Datenpunkte und einen erwartungstreuen Schätzer für die Varianz. Einen erwartungstreuen Schätzer für die Varianz s^2 kann man mit Hilfe der Bessel-Korrektur gewinnen. Sei $X = (x_1, \ldots, x_N)$ ein Sample einer Verteilung , mit $\hat{\mu}_X = \frac{1}{N}\sum_{i=1}^{N} x_i$, dann liefert:

$$s_X^2 = \frac{N}{N-1}\frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu}_X)^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \hat{\mu}_X)^2$$

einen erwartungstreuen Schätzer der Varianz. Seien also X_1,X_2 unabhängige Sample dann ist die Formel für das t-Value gegeben durch:

$$t = \frac{|\ \hat{\mu}_{X_1} - \hat{\mu}_{X_2}|}{\sqrt{\frac{s_{X_1}^2}{N_1} + \frac{s_{X_2}^2}{N_2}}}$$

Da wir im Prinzip nicht davon ausgehen dürfen, dass die Varianz der Population von Monat zu Monat sich im Infektionsgeschehen nicht ändert, müssen wir die angegebene Formel nutzen. Dies lässt sich in XQuery wie folgt implementieren:

```
declare function local:unbias_variance($seq as item()*)
{
  let $N-1 := count($seq) - 1
  let $N := count($seq)
  let $Bessel_Correction := $N div $N-1
  return $Bessel_Correction * local:variance($seq)
};

declare function local:Student_tTest($X1 as item()*,$X2 as item()*)
as xs:double
{
let $n1 := fn:count($X1)
let $n2 := fn:count($X2)
let $m1 := fn:avg($X1)
let $m2 := fn:avg($X2)
let $v1 := local:unbias_variance($X1)
```

```
let $v2 := local:unbias_variance($X2)

let $t := fn:abs($m1 - $m2) div math:sqrt($v1 div $n1 + $v2 div $n2)
return $t
};
```

3.2 Output-Formatierung

Um den Output übersichtlicher zu gestalten wurde ebenfalls eine Funktion `local:decimal_round` implementiert, welche Kommazahlen auf die 4te Kommastelle rundet.

```
declare function local:decimal_round($x as xs:double)
{
  let $round_x := fn:round($x * math:exp10(4)) div math:exp10(4)
  return $round_x
};
```

Der Output wird hier als HTML Table formatiert mit Spalten:

```
<tr>
    <th>Month:</th>
    <th>Count</th>

    <th>Average Number of Positive Tests</th>
    <th>Standard Deviation of Positive Tests</th>
    <th>Minimum Positive Tests</th>
    <th>Maximum Positive Tests</th>

    <th>Average estimated Infection Carrier</th>
    <th>Standard Deviation of estimated Infection Carriers</th>
    <th>Minimum estimated Infection Carrier</th>
    <th>Maximum estimated Infection Carrier</th>

    <th>Average Incidence</th>
    <th>Standard Deviation of Incidence</th>
    <th>Minimum Incidence</th>
    <th>Maximum Incidence</th>
  </tr>
```

Diese Form von Output gibt einem eine geordnete Übersicht über die wichtigesten Kennzahlen der gesammelten Daten und ist inspiriert durch die `df.describe()` Methode im Python Paket Pandas. Beim Output erzeugt das Skript jedoch leider kein valides HTML, da ich es nicht geschafft habe die Doc-Type Declaration in die erste Zeile des Outputs zu substituieren (essenziell habe ich keinen Weg gefunden die special-characters zu escapen). Da Browser jedoch kluge Compiler haben die auch viel interpretationsarbeit leisten rendered der Table trotz des invaliden HTML. Dieser Fehler fällt also nur auf wenn man wirklich in die Quell-Datei schaut und nicht die gerenderte Datei betrachtet. Es wäre trotzdem eine Verbesserung wenn der Output valides XML/HTML wäre um in etwaigen Prozessen mit den selben Werkzeugen weiterverarbeit werden zu können. Hier wäre also vermutlich das korrekte Werkzeug XSLT gewesen.

4 Auswertung

Die Bielefelder Daten zeichnen einige interessante Dynamiken der Pandemie ab. Einige Zeiträume die auffällig sind, sind die Monate:

- Juni, Juli, August in welchen die Lage in den Daten ziemlich gut aussah
- Oktober, der wie eine Art lokales Minimum wirkt

Die Analyse der Monate Mai bis August: Der Zeitraum von Mai bis August, sieht erst ein starkes Verlangsamen des Pandemie-Geschehens mit einem stabilen Monat gefolgt von einem rasanten Anstieg zum Schluss. Die Verlangsamung ist daran zu erkennen, dass die Anzahl der durschnittlichen Neuinfektionen von Mai auf Juni stark abnimmt von durchschnittlich 49 Neuinfektionen pro Tag im Mai zu nur durschnittlich 6 Neuinfektionen im Juni. Die Indikatoren für einen Umbruch im Infektionsgeschehen sind in diesen 4 Monaten gut zu erkennen, eine Erhöhung der Standard-Abweichung und eine größere Differenz zwischen Minima und Maxima weisen auf monatsinterne Volatilität hin. Besonders klar ist dieses Muster zu sehen wenn man die Daten von Juli und August betrachtet, im Juli war das Pandemie-Geschehen relativ konstant mit wenig Schwankungen. Niederige Standardabweichungen im Vergleich zu vorherigen Monaten und Maxima sowie Minima die nicht weit von ihren Mittelwerten entfernt sind. Der Juli ist sozusagen die ungestörte Fortsetzung des Trends welcher sich von Mai bis Juni einstellte. Dieser Trend war zu erwarten von Mai bis Juli lief die Impfkampange der Bundesregierung immer stärker an und die warmen Sommermonate bremsten das Pandemie-Geschehen, hier zeigen sich also die ersten Effekte der Impfungen und die kombinierten Auswirkungen des Sommers auf Grippeviren. Laut dem RKI ist die Delta Variante seit Ende April 2021 in Deutschland vertreten und ist seit Ende Juni 2021 die dominierende Virusvariante. Dies kann man im Wechsel zum August extrem gut in den Daten sehen. Eine Verzehnfachung der durchschnittlichen Neuinfektionen sowie eine Vervierfachung der Standardabweichung der Neuinfektionen in den Bielefelder-Daten zeigt die rasante Trendwende. Besonders spannend ist zu beobachten, dass der August der einzige Monat ist welcher sowohl Tage mit 0 Neuinfektionen als auch Tage mit 3 stelligen Neuinfektionen beinhaltet. Hier zeigt sich also klar, die Trendwende durch die Deltavariante ab.

Die Rasante Ausbreitung der Delta Variante und das vermeintliche Minimum Oktober: Die Monate September, Oktober, November, Dezember bestätigen klar die Trendwende des August. Mit der Außnahme des Oktobers welcher augenscheinlich nicht so ganz in den Datensatz passen will, da er eine klare Verbesserung nahelegt, wenn man eher eine Verschlechterung erwarten würde. Ich habe drei Hypothesen was diese Verbesserung angeht:

Hypothese 1: Das Infektionsschutz Gesetz des Landes NRW vom 01.10.2021 hat gut Wirkungen gezeigt und im Laufe des Oktobers die Zahlen gesenkt und so die niederigeren Durchschnitte verursacht.

Hypothese 2: Die ab dem 11.10.2021 kostenpflichtigen Bürgertests führen vermehrt zu späteren oder ganz ausbleibenden Meldungen von Covid Fällen.

Durch die allgemeine zeitliche Differenz zwischen dem Infektionsgeschehen an sich und den Daten führt dies dazu, dass Meldungen in den November umverteilt werden.

Hypothese 3: Die Sample sind nicht signifikant unterschiedlich.

Meine Analyse der Hypothesen: Ich halte Hypothese 2 oder 3 für wahrscheinlicher. Diese würden meines Erachtens nach besser zur Datenlage passen. Denn der Folge-Monat, welcher die defakto Nachmeldungen des Oktobers fangen müsste, hat eine vierfach so hohe Anzahl an durchschnittlichen Neuinfektionen mit der höchsten Standardabweichung der täglichen Neuinfektionen im ganzen Datensatz. Was zu einer systematischen Unterschätzung passen würde. Eine Anwendung von Students t-Test zeigt hier, dass die Populationen der Sample aus dem Oktober und der aus dem November statistisch signifikant anders sind, während die Unterschiede zu August und September nicht statistisch signifikant anders sind. Dies legt nahe, dass Hypothese 3 korrekt ist. Jedoch kann dies auch eine Anomalie der verfälschten Mittelwerte sein. Was nicht in den Daten ist, ist schwer quantifizierbar.

4.1 Fazit

Diese Auseinandersetzung zeigt deutlich die Fähigkeiten des XML Stacks zur Verarbeitung und Ordnung von Daten. Gerade das Modell der Sequenzen in Kombination mit XPath Ausdrücken macht es vergleichsweise einfach komplexe Baumstrukturen zu querien und zu verrechnen. An einigen Stellen hat die Entscheidung gegen ein Wochen-Tag die Interpretation schwerer gemacht als sie hätte sein müssen, was ich auch oft vermisst habe waren Meta-Information, wie die Einwohnerzahl Bielefelds oder die Temperatur/Sonnenstunden an einem entsprechenden Tag, sowie die durchschnittliche Differenz zwischen Meldung eines möglichen Coronafalles und eintreffen des PCR Ergebnisses. Auch wäre es in einem Produktionssetting sicherlich sinnvoll das Schema zu verallgemeinern damit es auch für Städte die nicht Bielefeld sind verwendet werden kann.