Fachprojekt Bioinformatik 17/18

Thomas Schlütter und Niclas Joswig

7.5.2018

1 Einleitung und Motivation

Data Science ist in der aktuellen Informatik eines der wichtigsten Themen. Die Wissenschaft der Datenanalyse befasst sich dabei mit verschiedenen Verfahren Daten zu sammeln, auszuwerten und Informationen zu generieren. In der heutigen Zeit treten Daten an allen möglichen Orten und Situationen im Alltag auf, im Zeitalter des Internets ist die globale Verfügbarkeit von Daten maximal. Ein Anwendungsfall, indem die Analyse von Daten eine wichtige Rolle spielt, ist die Bioinformatik. Subjekt der Bioinformatik ist die Lösung von medizinischen Herausforderungen durch computergestützte Methoden. Thema des Fachprojektes Bioinformatik war die Visualisierung eines Datensatzes von Genen. Dieser Datensatz wurde gewonnen, indem Proben von unterschiedlichen Tierzellen entnommen wurden und deren Genexpressionen gemessen wurden. Es wurden zwischen vier Zelltypen differenziert: embryonale Stammzellen (ESC), induzierte pluripotente Stammzellen (iPSC), Fibroblasten (Fib) und partiell induzierte pluripotente Stammzellen (piPSC). Die insgesamt 23 Proben sind wie folgt auf die vier Klassen verteilt: 13 ESC, 4 iPSC, 4 Fib und 3 piPSC. Innerhalb der Zellen wurden die Genexpression von 23232 einzelnen Genen gemessen. Da die Messungen der Expressionen Schwankungen unterliegen, wurden die Daten vor der weiteren Verwendung normalisiert.

Ziel der Forschung in diesem Bereich ist die Gewinnung von Zellen, welche ähnliche Eigenschaften der Stammzellen aufweisen. Diese Eigenschaften sind auf den Genen codiert, dementsprechend wird versucht durch Analyse der Genexpression die Unterschiede zwischen Stammzellen und anderen Zellen festzumachen. Aus diesen Ergebnissen können Schlüsse daraus gezogen werden, wie die zukünftige Forschung und Entwicklung von Zellen aussehen sollte.

2 Realisierung

Bevor auf die Inhalte und Funktionsweisen der Aufgaben eingegangen wird, muss die Realisierung und die Vorbereitung der Programmierung erläutert werden. Die Realisierung dieses Projekts geschieht mithilfe der *Bokeh* Bibliothek, basierend auf der Programmiersprache Python. Die Ausgabe erfolgt mithilfe eines Bokeh-Servers, welcher als *localhost* aufgerufen werden kann. Zunächst werden die notwendigen Bibliotheken *Numpy* und *Pandas* importiert. Numpy und Scipy stellen die Grundlage jeden Python Programmes zum Thema Data Science dar, da sie die essenziellen Bestandteile des Arrays und Dataframes beinhalten. Anschließend werden die zur Verwendung von Bokeh benötigten Bibliotheken importiert.

Die als CSV-Dateien vorliegenden Datensätze werden mithilfe von Pandas als Dataframes importiert. Diese Dataframes sind alphabetisch geordnet, wodurch die Notwendigkeit nach einer neuen Sortierung nach Klassen entsteht. Für die folgenden Implementierungen werden außerdem die Zelltypen als getrennte Liste definiert und die Farbpalette für die farbige Darstellung von Linien definiert.

Um die Daten korrekt einzulesen, sollten die CSV Dateien alle im selben Ordner, wie das Notebook sein. Die einzelnen Aufgaben sind bei Ausführen von oben nach unten die Aufgaben 1-3 unter den Ports $8001-8003$ unter den Adressen *localhost:*$8001-8003$. Bei mehrfachem Ausführen wird der Port auch unter der Zelle des Notebooks angegeben.

3 Aufgabe 1

3.1 Inhalt

Inhalt der ersten Aufgabe war die Darstellung der Verteilung der Genexpression in den verschiedenen Zellproben, welche im folgenden als Klassen bezeichnet werden. Diese Verteilung wird visualisiert, indem die einzelnen Gene auf der x-Achse und die Ausprägung auf der y-Achse dargestellt werden. Für jede der 23 Klassen wird eine Linie in den Graphen entsprechend ihrer Ausprägung der jeweiligen Gene eingezeichnet.

Bei Verwendung der nicht normalisierten Daten (Raw-Data) ist in der Abbildung zunächst nur eine flache Gerade und anschließend ein steiler Anstieg zu verzeichnen. Das gleich gilt für die Verwendung der normalisierten Daten.

Aufgrund dieser Problematik wird eine logarithmische Skala auf die y-Achse, also die Genexpressionswerte angewandt. Durch die große Differenz zwischen einzelnen Expressionwerten, hat die y-Achse eine sehr große Re-

ichweite, wodurch feine Unterschiede nicht sichtbar sind. Durch die Anwendung der logarithmischen Skala wirkt man dem Problem entgegen, durch die effektive Verkleinerung durch die Funktionseigenschaften des Logarithmus. Mit einer logarithmischen Skala werden die einzelnen Kurven deutlich feiner dargestellt. Um genauer auf Verläufe von spezifischen Kurven einzugehen, wird die Möglichkeit geboten einzelne Linien auszublenden, um beliebe Konfigurationen von Kurven darzustellen. Zusätzlich zu der logarithmischen Darstellung wird die Möglichkeit geboten die Werte abzüglich des Medians der jeweiligen Linie zu betrachten.

3.2 Bedienung

Beim Starten der Anwendung ist zunächst die normalisierten Daten auf einer linearen Skala dargestellt. Wie in Kapitel beschrieben ist auf der x-Achse die einzelnen Gene und auf der y-Achse die Expressionswerte abgebildet. Die Funktion zum Aus- und Einblenden der einzelnen Linien wird über die Legende in der linken oberen Ecke realisiert: Anklicken eines Eintrages blendet die Linie der passenden Klasse ein beziehungsweise aus. Die Verwendung der verschiedenen Darstellungsmöglichkeiten geschieht über die Knöpfe am linken Bildrand. Mithilfe der Knöpfe kann die Anwendung zwischen allen in Kapitel beschriebenen Darstellung beliebig wechseln. In Abbildung 1 sind die Daten exemplarisch in logarithmischer Darstellung abgebildet. In der unteren linken Ecke der Abbildung 1 ist erkennbar, dass die Nullen die Darstellung der Expressionswerte verschlechtern, jedoch war es uns nicht möglich die Nullen heraus zu filtern, da daraus unterschiedliche Intervalle auf der x-Achse entstanden sind, welche Inkonsistenzen innerhalb der Darstellung ergeben haben.

3.3 Code

Als Erstes wird die Liste *classes* als Liste von allen Klassennamen erstellt. Da der Datensatz 23323 Gene enthält, hat die x-Achse eine Distanz von 23323 Einheiten, welches in *x-data* spezifiziert ist. Die p1 genannte Figur ist das zentrale Element, welches von Bokeh dargestellt wird. Teil der Figur sind unter anderem die Achsen und die Linien, welche später gezeichnet werden. In dieser Aufgabe werden fünf verschiedene Dictionarys verwendet, um die fünf verschiedenen Darstellungsoptionen beschrieben in Kapitel zu speichern. Die Dictionarys werden mit Inhalt befüllt, indem die Spalten der einzelnen Klassen mit jeweils 23323 Einträgen als Liste dem Namen der Klasse zugeordnet werden. Dabei sind die einzelnen Dictionarys unterschiedlich: ein eines werden die normalisierten und in eines die nicht normalisierten

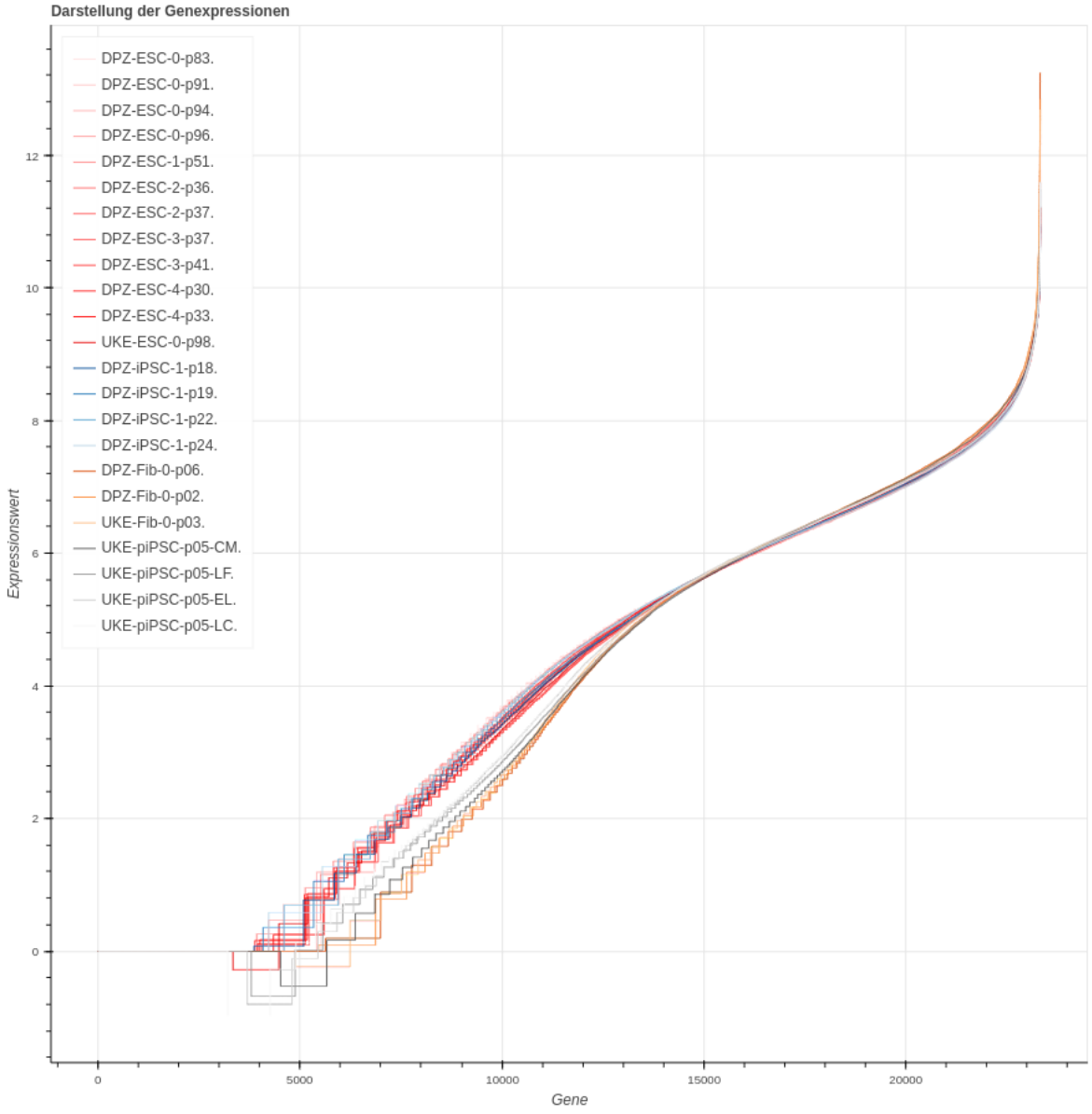

Figure 1: Logarithmische Darstellung der Daten

Daten gespeichert. Für die logarithmischen Darstellung müssen zunächst die Einträge nach Nullen durchsucht werden, da der Logarithmus von null nicht definiert ist. In diesem Prozess werden von allen Werten ungleich null der Logarithmische Wert gespeichert und für die Nullwerte wird weiterhin null gespeichert. Bei der Mediandarstellung haben wir uns dazu entschieden, die Mediandarstellung auf der logarithmischen Skala darzustellen, da auf der linearen Skala kein Unterschied festzumachen ist.

Die Linien werden mithilfe des *active*-Dictionarys gezeichnet. In diesem Dictionary befindet sich immer die Daten der aktuell ausgewählten Darstellung. Falls einer der Knöpfe gedrückt wird, wird der Inhalt des *active*-Dictionarys mit dem Inhalt der neuen Darstellungsweise überschrieben.

Um eine bessere Übersicht über die verschiedenen Linien zu bekommen, werden die Farben zwischen den Klassen variiert. Jede der vier Zellklassen hat eine Grundfarbe: ESC ist rot, iPSC ist blau, piPSC ist grün und Fib ist orange. Um zusätzlich zu den verschiedenen Zellklassen auch zwischen den einzelnen Zellproben innerhalb der vier Klassen zu unterscheiden, wird ausgehend von der Grundfarbe die Linienfarbe mithilfe einer *Bokeh Palette* variiert.

4 Aufgabe 2

4.1 Inhalt

In Aufgabe 2 wird mit Listen von interessanten Genes, welche als Text-Datei vorliegen gearbeitet. Ziel der Aufgabe ist das Finden von Genen, für welche die Expression in allen Klassen unterschiedlich ist. Ziel der Aufgabe ist eine Visualisierung zu entwickeln, welche die gesuchte Eigenschaft der Expressionen für interessante Gene leicht erkennbar macht.

4.2 Bedienung

Wenn die Anwendung gestartet wird, sind alle Gene, welche in der Datei **pluripotencygenes.txt** enthalten sind abgebildet. In der Abbildung 2 von Aufgabe 2 werden die eingelesenen Gene auf der x-Achse mit den zugehörigen Expressionen auf der y-Achse abgebildet. Durch die Darstellung übereinander in einer Linie ist es direkt möglich Gene zu erkennen, welche für die verschiedenen Klassen unterschiedliche Ausprägungen haben. In der linken unteren Ecke befindet sich ein Textfeld zum Eingeben von neuen Textdateien, welche in der Abbildung dargestellt werden sollen. Diese Option funktioniert leider nicht, genaueres dazu in Kapitel 4.3.

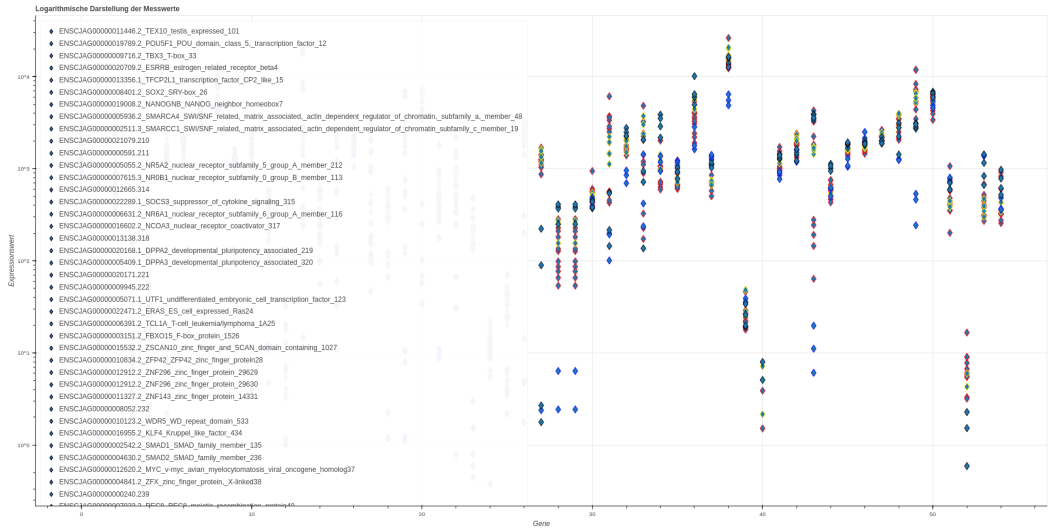

Figure 2: Darstellung der Pluripotency-Gene

4.3 Code

Die Funktion des Einlesens von Dateien wurde in zwei Funktionen *search* und *read_file* ausgelagert. Die Funktion *read_file* bereitet öffnet die Textdatei und wendet die Funktion *search* auf jeden Eintrag der Datei an. Die *search*-Funktion iteriert durch alle 23323 Gennamen und überprüft, ob der Eintrag ein gültiger Genname ist. Falls ein Genname gefunden ist, wird ein Tupel aus Genname und Indize des Gens zurückgegeben. Zurück in der *read_file*-Funktion werden alle diese Tupel zu einer Liste zusammengefügt und zurückgegeben. Basierend auf den beiden Methoden wird die **pluripotencygenes.txt** Datei eingelesen und die zurückgelieferten Gene aus den normalisierten Daten ausgeschnitten.

In dieser Aufgabe wird für jeder der vier Zellklassen ein eigenes Dictionary verwendet. Gefüllt werden die Dictionarys, indem eine Zeile der Daten in das Attribut *name* und die Liste *werte* aufgeteilt wird. In das Dictionary wird anschließend der extrahierte Genname auf die Expressionswerte abgebildet. Zusätzlich werden die x-Werte definiert, indem der Genindex so häufig wiederholt wird, wie es Proben innerhalb der vier Klassen gibt. Die x-Werte werden so gewählt, damit die vertikale Anordnung in einer Linie der Expressionen gewährleistet ist. Dieser Prozess wird für alle Gene, welche in der Datei aufgelistet sind, durchgeführt.

Die Linien werden anschließend mit dem *active*-Dictionary dargestellt, um die Möglichkeit zur Aktualisierung des Dictionarys zu haben. Falls der Benutzer einen Dateinamen in das Textfeld einträgt, wird die *text_import_handler*-

Funktion aufgerufen. Der Inhalt dieser ist der gleiche, welcher bereits verwendet wurde, um die **pluripotencygenes.txt** Datei darzustellen. Es wird die neue Datei eingelesen, die Dictionarys werden gefüllt und anschließend die *active*-Dictionarys aktualisiert.

Probleme gab es in dieser Aufgabe bei der Aktualisierung durch Eingabe einer neuen Datei. Obwohl das *active*-Dictionary korrekt aktualisiert wird, aktualisiert sich der Plot nicht. Durch Ausgeben der Dictionarys kann nachvollzogen werden, wie der Code funktioniert, jedoch kann keine Aktualisierung erreicht werden. Falls Sie uns Feedback zu diesem Problem geben können, wären wir sehr dankbar! Aufgrund dieser Problematik muss zum Testen von anderen Textdateien als die Datei **pluripotencygenes.txt** entweder der Inhalt der Datei **pluripotencygenes.txt** oder der Dateiname innerhalb des Notebooks geändert werden.

5 Aufgabe 3

5.1 Inhalt

Ziel von Aufgabe 3 ist das Finden von interessanten Genen. Ein Gen ist als interessant definiert, wenn das Minimum der Klasse mit dem größeren Mittelwert größer als das Maximum der Klasse mit dem niedrigeren Mittelwert ist.

Zunächst muss eine Klasse definiert werden: im Gegensatz zu vorherigen Aufgaben können in Aufgabe 3 die vier Zellklassen zu beliebigen Kombinationen zusammengefasst werden. Nach der Auswahl zweier Klassen, zwischen welchen die interessanten Gene berechnet werden sollen, muss der Mittelwert über alle Werte der Klasse für ein Gen berechnet werden. Anschließend wird das Minimum der größeren Klasse mit dem Maximum der kleineren Klasse verglichen.

Ist diese Bedingung erfüllt, ist das Gen als interessant markiert. Die Eigenschaft interessant sagt aus, dass die Ausprägungen zwischen den zu vergleichenden Klassen grundlegend unterschiedlich sind, deswegen ist das Gen ein wichtiger Faktor bei der Unterscheidung der verschiedenen Klassen. In Bezug auf die Erforschung von neuen Zellen können mithilfe des Wissens über interessante Gene Ansätze für eine Verbesserung der bisher entwickelten Zellen erreicht werden.

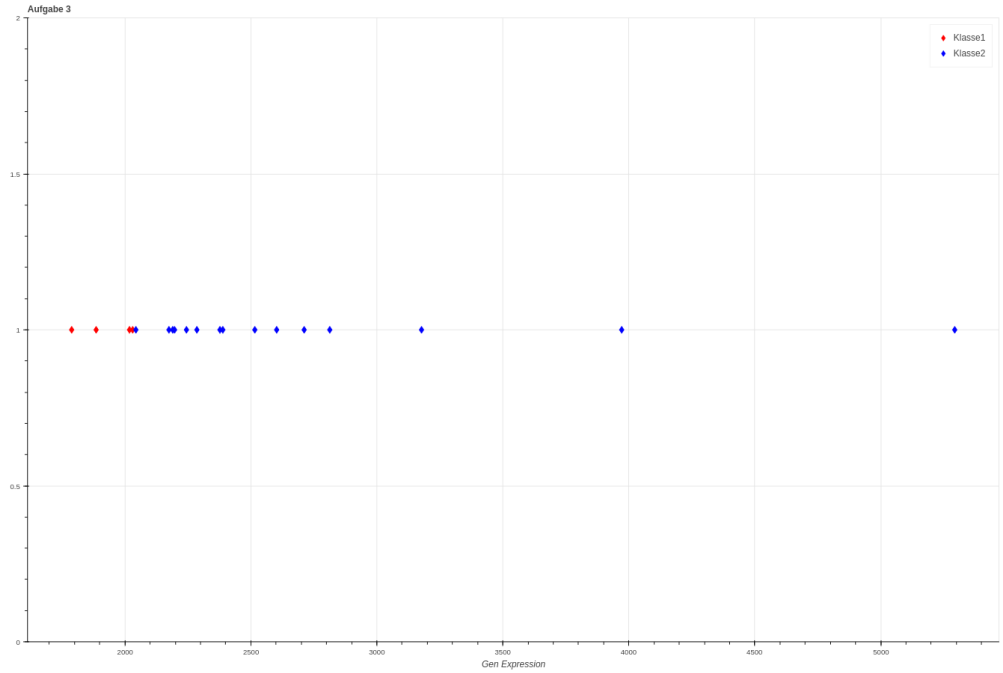

Figure 3: Darstellung eines interessanten Gens in Aufgabe 3

5.2 Bedienung

Der erste Teil von Aufgabe 3 ist die Auswahl der beiden Klassen. Über zwei Buttonleisten in der oberen linken Ecke können die beiden Klassen in beliebigen Kombinationen zusammengesetzt werden. Sind die richtigen Klassen ausgewählt, kann auf den "Gene anzeigen" Knopf gedrückt werden. Diese starten die in Kapitel 5.1 geschilderten Berechnung der interessanten Gene. Dieser Prozess kann je nach Systemleistung bis zu einigen Minuten dauern, welche abgewartet werden müssen. Sind die Berechnung vollendet, wird automatisch im *Select-Feld* darunter die Liste von interessanten Genen angezeigt. Der Benutzer kann anschließend ein beliebiges Gen aus der Liste auswählen, welches angezeigt werden soll. In Abbildung 3 ist exemplarisch dargestellt, wie ein ausgewähltes Gen in beiden Klassen dargestellt wird.

5.3 Code

Der wichtige Teil von Aufgabe 3 ist die Funktion *interessant*, welche die interessanten Gene berechnet. Als Eingabe bekommt sie die über die Knöpfe ausgewählten Klassen übergeben. Der erste Schritt ist die Umformung von den übergebenen Werten der Knöpfe, welche Zahlenwerte zwischen 0 und 3

sind, in die Klassennamen der ausgewählten Klassen. Mit diesen Klassennamen werden die ausgewählten Spalten aus den normalisierten Daten ausgeschnitten.

Basierend auf diesen beiden Dataframes, welche nur die jeweils ausgewählten Klassen enthalten, wird für jedes Gen einzeln entschieden, ob es interessant ist. Dies geschieht mit der in der Aufgabenstellung gegebenen Methode, den Mittelwert beider Klassen zu berechnen und das Minimum der größeren Klasse mit dem Maximum der kleineren Klasse zu vergleichen. Erfüllt das Gen die Eigenschaft, wird es zu der Liste der interessanten Gene hinzugefügt. Zuletzt wird für jedes interessante Gen der Name des Gens zusätzlich zu dem Index als Tupel gespeichert. Die Rückgabewerte sind die Liste der interessanten Gene und die Dataframes der beiden Klassen.

Innerhalb des Servers werden zu Beginn zwei Dictionarys, ein aktives und ein nicht-aktives, für jede Klasse erzeugt. Nach Auswahl der Klassen und dem Drücken auf den "Gene anzeigen"-Knopf wird die oben beschriebene *interessant*-Funktion ausgeführt und die Rückgabewerte gespeichert. Um die Auswahl durch ein *Select*-Feld zu ermöglichen, müssen die Gennamen als Optionen an das *Select*-Objekt übergeben werden. Dazu wird die *zip*-Funktion benutzt, um aus der Liste aus Tupeln mit Gennamen und Genindizes nur die Namen zu extrahieren.

Der nächste Schritt ist das Auswählen eines Gens innerhalb der *Select*-Box. Geschieht dies, wird der ausgewählte Eintrag innerhalb der Liste von interessanten Genen gesucht. Ist der passende Eintrag gefunden, wird mithilfe des Indexes die Datenreihen des Gens aus den Dataframes gewonnen.

Diese Werte werden zusätzlich zu einer Liste von Einsen in die Dictionarys eingetragen und in das *active*-Dictionary kopiert. Der letzte Schritt ist erneut das Zeichnen der Linien mithilfe der beiden Dictionarys.