Semesterarbeit – 07.01.2020

Verwendung von Python zur Lösung zweier Problemstellungen in QGIS und Power BI

Im Rahmen des CAS Grundlagen Data Science der Fernfachhochschule Schweiz (FFHS)

Impressum

Autor: Matthias Setz
Titel: Verwendung von Python zur Lösung zweier Problemstellungen in QGIS und Power BI

Datum: 07.01.2020

Matthias Setz
Pappelweg 8
3013 Bern

Tel +41 79 798 27 00
matthias.setz@bluewin.ch

Inhaltsverzeichnis

1 Einleitung

Vorwort des Autors

Die vorliegende Arbeit wurde im Rahmen des CAS Grundlagen Data Science der Fernfachhochschule Schweiz (FFHS) erstellt. Dabei ging es darum, die erlernten Inhalte aus dem CAS an einem praktischen Beispiel anzuwenden. Da ich im letzten Jahr bereits das CAS Datenanalyse besucht hatte, konnte ich bereits auf ein gewisses Basiswissen in Python zurückgreifen.

Im Vordergrund der vorliegenden Arbeit stand für mich die Verwendung von Python-Skripts in verschiedenen Programmen, welche ich bei meiner täglichen Arbeit nutze. Dabei handelt es sich einerseits um die Geoinformationssystemssoftware **QGIS** und andererseits um die aus der Microsoft-Office-Umgebung stammende Business-Intelligence-Software **Power BI**.

Diese beiden Programme werden für die Lösung von zwei verschiedenen Problemstellungen verwendet, deren Hintergründe im nachfolgenden Abschnitt kurz erläutert werden. Die Arbeit ist im Anschluss grundsätzlich in zwei alleinstehende Kapitel gegliedert: In Kapitel 2 wird QGIS für die Lösung einer Problemstellung verwendet, in Kapitel 0 Power BI für die Ergänzung einer bereits bestehenden Auswertung. In beiden Fällen wird aber auf die Programmiersprache Python zurückgegriffen.

Hintergrund der Themenwahl

Im Rahmen eines Bundesmandats bin ich zurzeit daran, die siedlungsstrukturellen Veränderungen der letzten 10-20 Jahre auf der Ebene der Gemeinden zu analysieren. Hierzu mussten diverse Kartengrundlagen erstellt werden. Die vorliegende Arbeit beschäftigt sich in Kapitel 2 mit der Darstellung der Pendlerströme zwischen den Gemeinden sowie der Anzahl Arbeitsplätze und Einwohner/innen pro Gemeinde. Die beiden Strukturdatensätze (Arbeitsplätze und Bevölkerung) werden hierbei mittels eines Pythonskripts bezogen und manipuliert.

Im letzten Halbjahr durfte ich für das Monitoring von regiosuisse[1] drei «blogähnliche» digitale Storys erarbeiten, die sich mit der regionalwirtschaftlichen Entwicklung innerhalb der Schweiz auseinandersetzen.[2] In diesem Zusammenhang habe ich im Rahmen des CAS Visualisierung bereits einen ersten Entwurf einer interaktiven Indikatorauswertung auf Gemeindeebene erstellt.[3] Für die Erstellung der Visualisierung sowie der zugrundeliegenden Datenbank habe ich die Software Power BI verwendet, welche für das Datenbankmanagement sowie die Datenvisualisierung auch Python unterstützt. Die vorliegende Arbeit beschäftigt sich mit der Erstellung einer interaktiven Python-Visualisierung sowie dem skriptbasierten Einlesen von Daten innerhalb von Power BI.

---

[1] Mandats-Verhältnis im Auftrag des Staatssekretariats für Wirtschaft (seco).

[2] Vgl. regiosuisse-Monitoring im Internet: https://regiosuisse.ch/monitoring-der-regionalwirtschaftlichen-entwicklung.

[3] Vgl, Web-Applikation unter: https://app.powerbi.com/view?r=eyJrIjoiN2Q5MjYzZjEtMWYxMS00YmFhLTllZmQtMjMwMzQ2MmRmOTc5IiwidCI6ImZlNWRhNTQ0LTAyMmYtNDJiMy04YTVhLTExZjBhNTNjNGM2NCIsImMiOjI9

2 Analyse von siedlungsstrukturellen Veränderungen in QGIS

2.1 Problemstellung

Anhand einer Kartendarstellung auf Ebene Schweiz sollen die Siedlungsstrukturen bezüglich Bevölkerung und Arbeitsplätzen aufgezeigt werden. Diese Darstellung soll für lokale Analysen (Agglomeration oder Kanton) mit den visualisierten Pendlerströmen überlagert werden. Dadurch sollte ersichtlich werden, wo die Leute wohnen, wo sie Arbeiten und welche Wege täglich zurückgelegt werden. Idealerweise könnte die Karte zusätzlich um Informationen zum städtischen Charakter ergänzt werden.

2.2 Methodik

Für die Lösung der Problemstellung wird die Geoinformationssystemssoftware **QGIS**[4] verwendet, welche ein Open-Source-Projekt ist eine grosse Entwickler- und User-Community aufweist. Für den Aufbau der benötigten Layer und deren Darstellung in QGIS wird nicht nur die Programmiersprache Python verwendet, welche die Grundprogrammiersprache in QGIS darstellt, sondern auch SQL-ähnliche Abfragen, welche von QGIS unterstützt werden.

Während die Visualisierung der grössenabhängigen Kuchendiagramme absolut selbständig erarbeitet wurde, orientiert sich die Darstellung der Pendler-Pfeile am Blogartikel[5] «Flow maps in QGIS – no plugins needed!» von Anita Graser (2019).

2.3 Datengrundlagen

Als Datenquelle werden einerseits frei verfügbare Kartengrundlagen des Bundes verwendet:

- Generalisierte Gemeindegrenzen (inkl. See & CH-Grenze)[6]
- WMS-Layer (wie z.B. das Oberflächenmodell)[7]

Andererseits werden die Strukturdaten (Bevölkerung und Arbeitsplätze pro Gemeinde) über die Plattform STAT-Tab des BFS bezogen.[8] Zudem stammen auch die Pendlerdaten vom BFS (Sektion Mobilität).[9]

---

[4] Frei verfügbar unter: https://www.qgis.org/de/site/forusers/download.html [27.12.2019].

[5] Vgl. Anita Graser (2019), Flow maps in QGIS – no plugins needed!, Online im Internet: http://planet.qgis.org/planet/tag/flows/ [27.12.2019].

[6] Frei verfügbar unter: https://www.bfs.admin.ch/bfs/de/home/dienstleistungen/geostat/geodaten-bundesstatistik/administrative-grenzen/generalisierte-gemeindegrenzen.assetdetail.7566557.html [27.12.2019].

[7] Frei verfügbar unter: https://www.geo.admin.ch/de/geo-dienstleistungen/geodienste/darstellungsdienste-webmapping-webgis-anwendungen/web-map-services-wms.html [27.12.2019].

[8] Vgl. Online im Internet: https://www.pxweb.bfs.admin.ch/pxweb/de/?rxid=8335ce3a-6921-46c5-9a5e-882feddb024c [27.12.2019].

[9] Vgl. Online im Internet: https://www.bfs.admin.ch/bfs/de/home/statistiken/mobilitaet-verkehr/personenverkehr/pendlermobilitaet.assetdetail.8507273.html [03.01.2020].

2.4 Detaillierte Vorgehensweise

In einem ersten Schritt wurden allgemeine Kartengrundlagen zusammengestellt respektive Verschiedene Layer übereinandergelegt. Die untenstehende Abbildung listet die überlagerten Layer auf: Zuoberst befinden sich die Gemeindepolygone, welche zu 70% transparent gestaltet wurden, damit das darunterliegende Relief des BAFU-Oberflächenmodells[10] noch leicht durchschimmert und dadurch die topographischen Strukturen innerhalb der Schweiz veranschaulicht (vgl. Abbildung 2-2). Zudem wurde die Karte um die Seen innerhalb der Schweiz ergänzt und an der Schweizergrenze mittels eines invertierten Polygons «abgeschnitten».

Abbildung 2-1: Verwendete Grundlagelayer

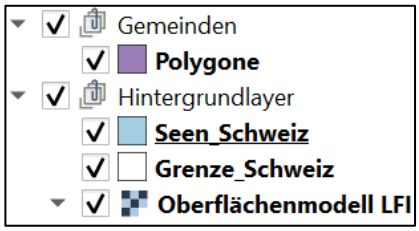
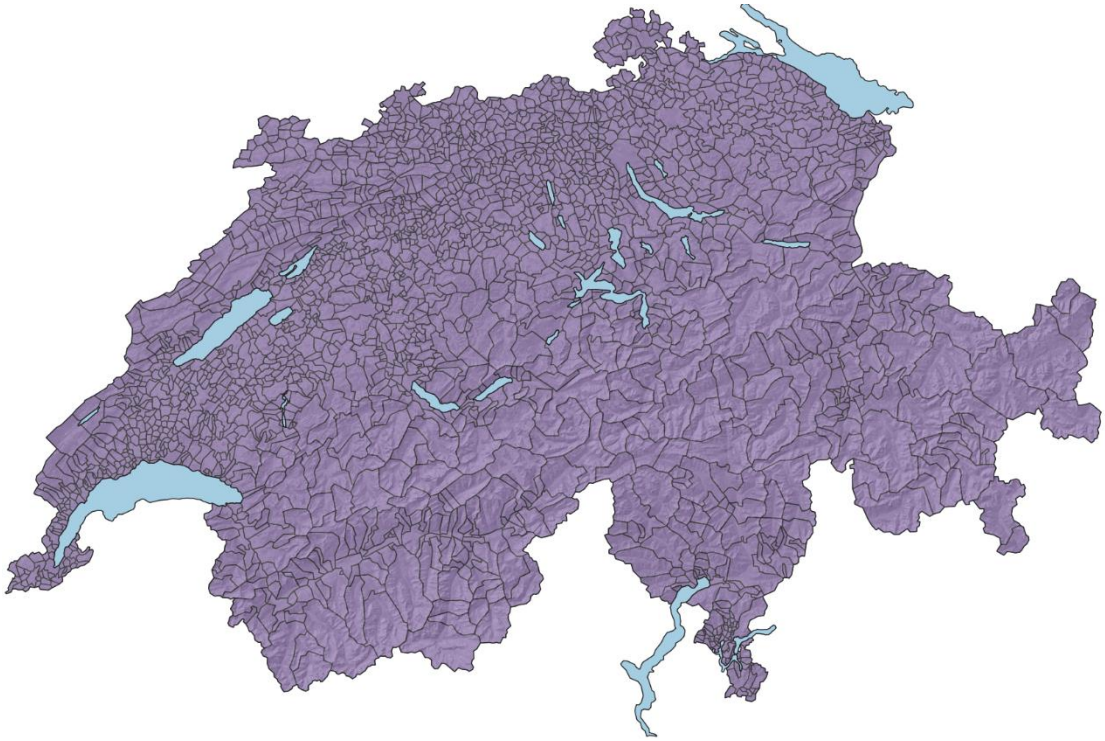


Abbildung 2-2: Karte mit Grundlagelayer als Ausgangspunkt der Analysen



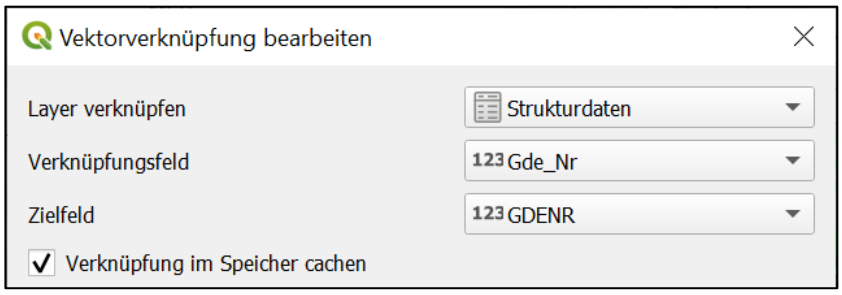
Quelle: Eigene Darstellung

[10] WMS-Layer: Oberflächenmodell LFI - ch.bafu.landesforstinventar-vegetationshoehenmodell_relief_visual

In einem nächsten Schritt werden die gewünschten Daten zu Bevölkerung und Arbeitsplätzen über eine Post-Request-Anfrage als CSV-Dateien bezogen, mittels der Pandas-Bibliothek auf Basis der Gemeindenummer zusammengeführt (merge) und als eine einzelne CSV-Datei abgelegt (vgl. Kommentiertes Jupyter-Notebook «Bezug_der_Strukturdaten**.ipynb**» im Ordner «Python-Skripte»). Das Python-Skript «Bezug_der_Strukturdaten**.py**» kann auch direkt in QGIS ausgeführt werden. Dazu muss über das Hauptmenu → *Erweiterungen* die Python-Konsole gestartet werden (Shortcut: Ctrl+Alt+P). Anschliessend kann über den Editor ein Python-Skript geöffnet und ausgeführt werden (Wichtiger Hinweis: Für die Ausführung des vorliegenden Skripts muss zuerst noch die Pandas-Bibliothek innerhalb der Python-QGIS-Umgebung installiert werden. Ansonsten ist das Skript nicht lauffähig)[11].

Nun, wo die darzustellenden Daten in der gewünschten Form vorhanden sind, kann mit der Visualisierung ebendieser Daten begonnen werden. Als erstes werden die grössenabhängigen Kuchendiagramme für Bevölkerung und Arbeitsplätze auf Ebene Schweiz erstellt. Hierzu wird als erstes der Gemeinde-Layer dupliziert und transparent eingefärbt. Danach werden die Strukturdaten, welche als Datensatz in QGIS geladen wurden[12], über die Gemeindenummer mit dem Gemeindelayer verknüpft (vgl. Abbildung 2-3).

Abbildung 2-3: Anknüpfung der Strukturdaten an den Gemeindelayer (Polygone)

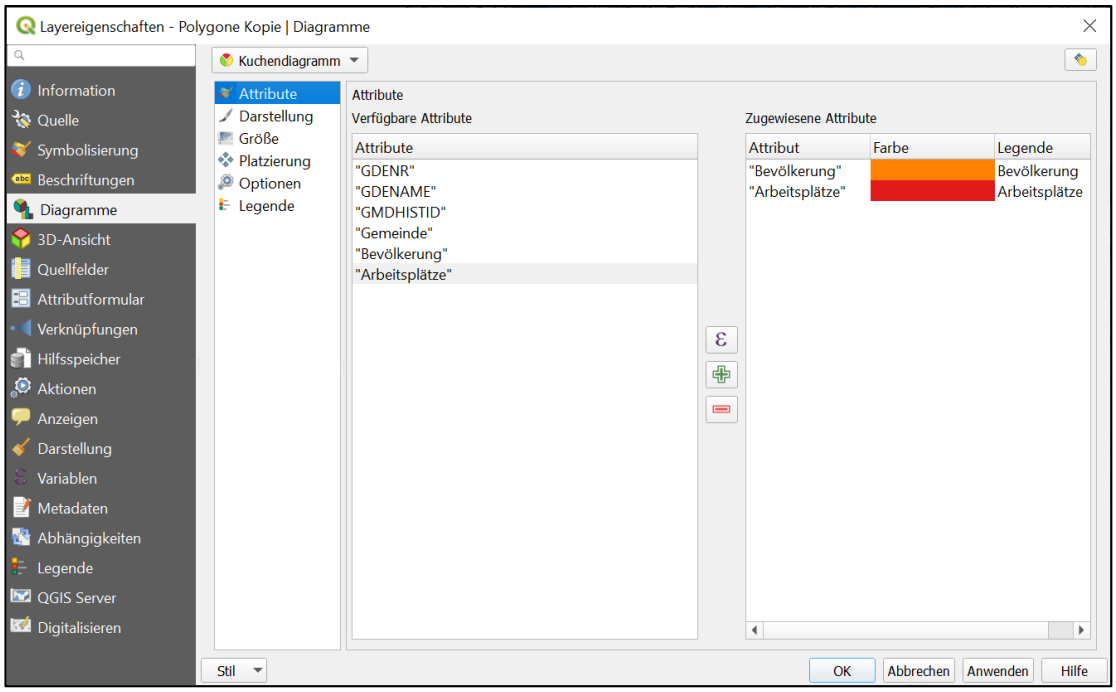


In einem weiteren Schritt werden auf der Symbolebene des Layers die Inputs für das Kreisdiagramm gewählt (vgl. **Fehler! Ungültiger Eigenverweis auf Textmarke.**).

[11] Alternativ kann auch das Python-Skript «Bezug_der_Strukturdaten_ohne Pandas.py» ohne Pandas ausgeführt werden. Dieses beinhaltet jedoch lediglich den Bezug der Daten der STAT-Tab-Website (Dateien «Arbeitsplätze.csv» und «Bevölkerung.csv». Die Daten werden jedoch weder umstrukturiert noch zusammengeführt.

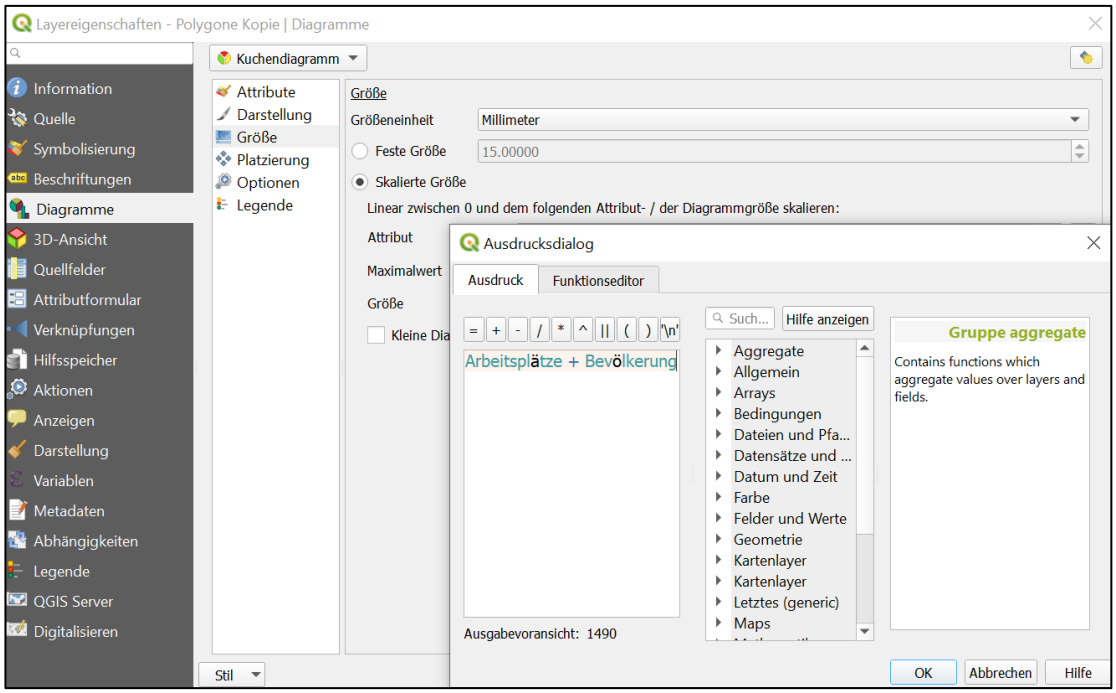
[12] Über das Hauptmenu → *Layer* → *Layer hinzufügen* → *Getrennte Textdatei als Layer hinzufügen*

Abbildung 2-4: Wahl der Kreisdiagrammelemente auf der Symbolebene des Gemeindelayers



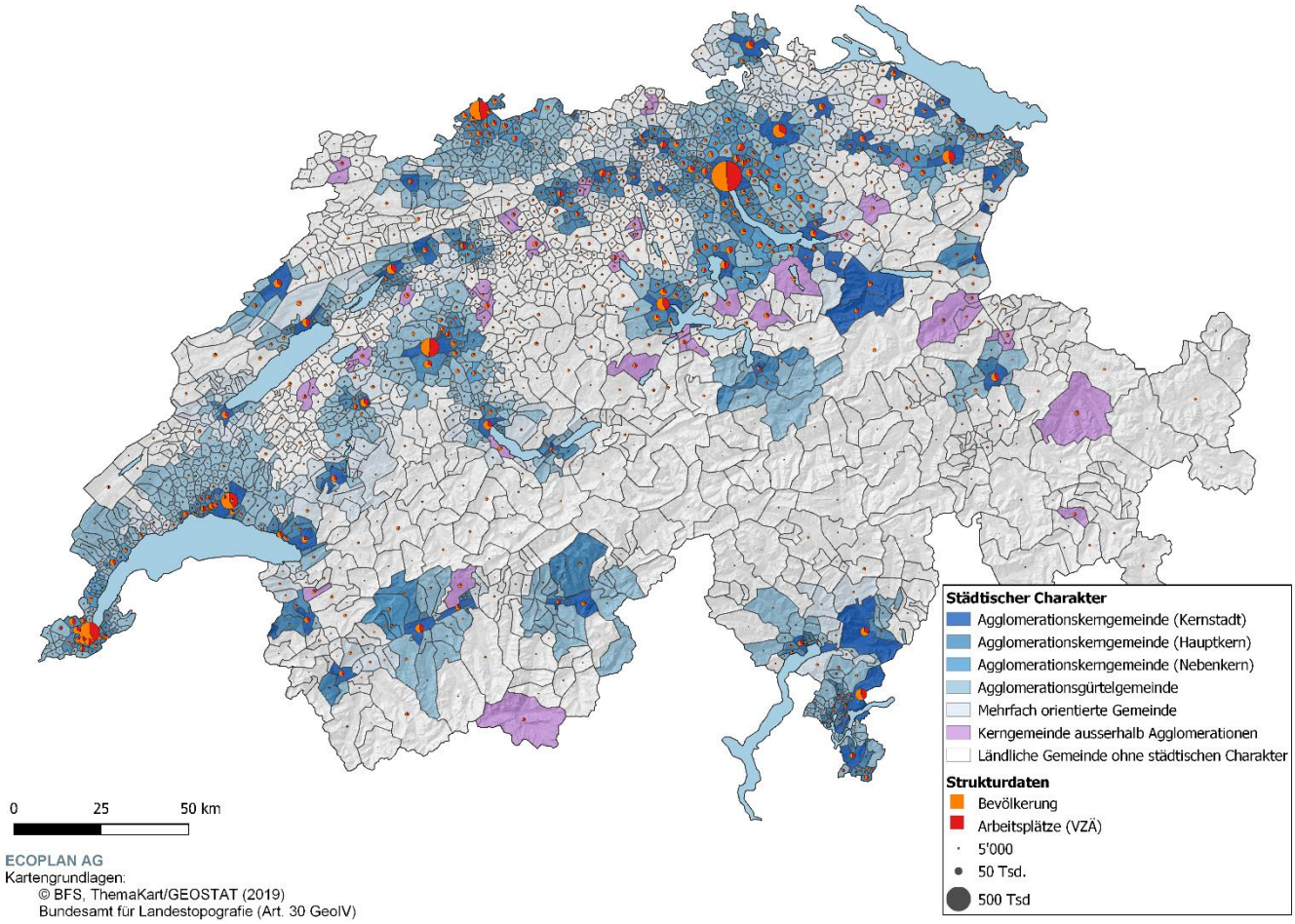
Um die Kreisdiagrammgrösse von der Gesamtmenge an Einwohner/innen und Arbeitsplätzen abhängig zu machen, kann die Grösse auf Basis einer Formel (=Arbeitsplätze + Bevölkerung) definiert werden (vgl. untenstehende Abbildung)

Abbildung 2-5: Definition der Skalierung des Kuchendiagramms



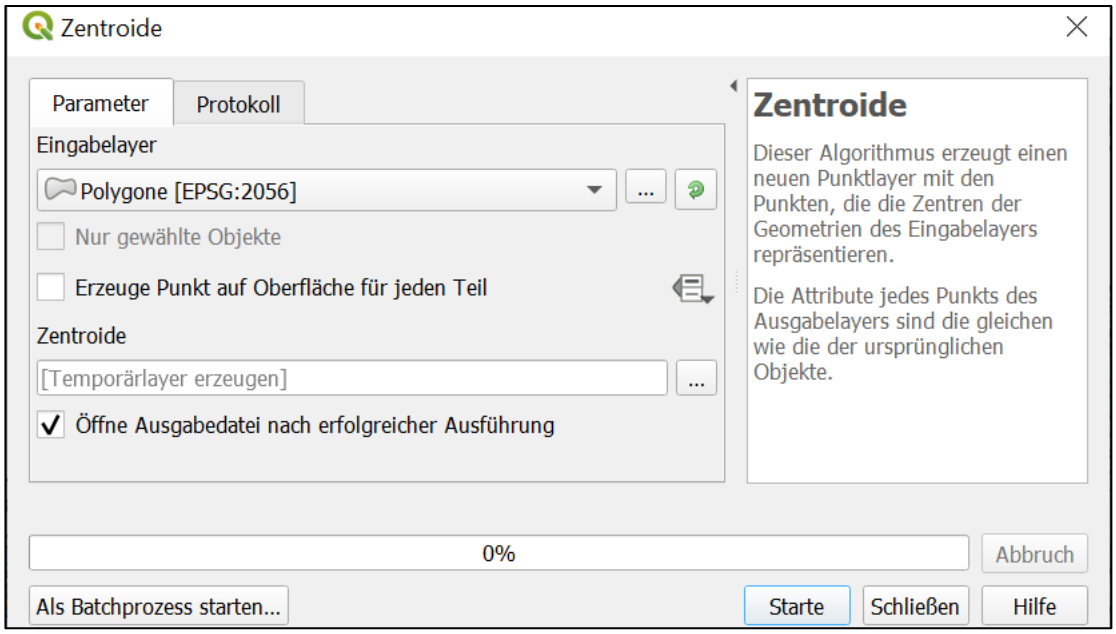
Untenstehend die resultierende Gemeindekarte auf der Ebene der Schweiz. Es zeigt sich, dass die urbanen Zentren (z.B. Zürich und Genf) eine deutlich höhere Siedlungsdichte aufweisen als die restlichen Landesteile und sich dadurch in gewissen Agglomerationen (vgl. Einfärbung der mit städtischem Charakter bzw. blaue Hinterlegung der Gemeindepolygone) «Ballungszentren» ergeben. Zudem ist ersichtlich, dass sich Bevölkerung und Arbeitsplätze (Vollzeitäquivalente (VZÄ) in den Grosszentren in etwa die Waage halten. Auf Ebene Schweiz sieht dies ein bisschen anders aus, denn dort kommen auf 1'000 Einwohner/innen gerade mal knapp 500 Arbeitsplätze (VZÄ).

Abbildung 2-6: Siedlungsstruktur (Arbeitsplätze und Bevölkerung) der Gemeinden, 2016



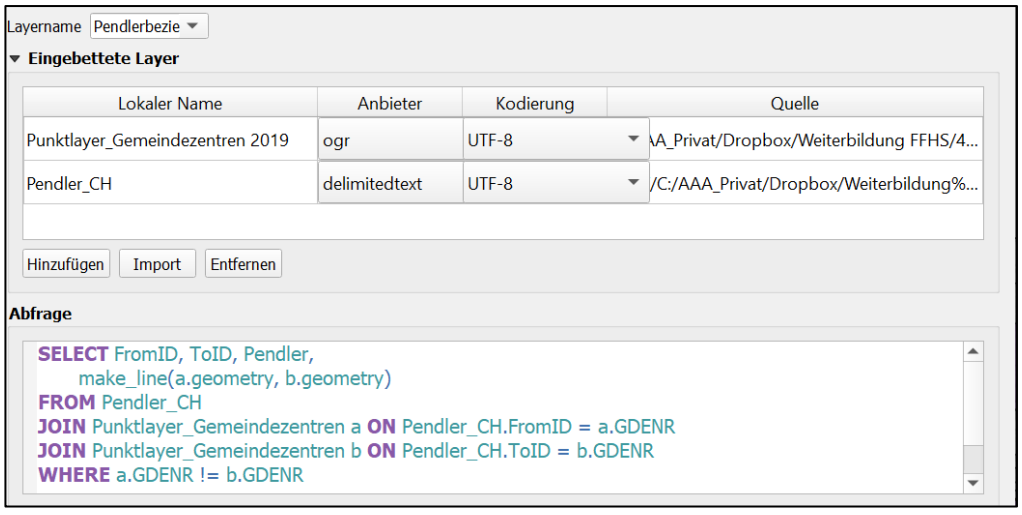
Soweit, so gut – Nun können wir uns an die Darstellung der Pendlerströme machen. Hierzu wird als erstes ein Punktlayer erstellet, welcher die einzelnen Gemeindemittelpunkte abbildet. Dazu kann das in QGIS integrierte Werkzeug «Zentroide» verwendet werden (vgl. Abbildung 2-7).

Abbildung 2-7: Anwendung des Zentroide-Werkezeugs



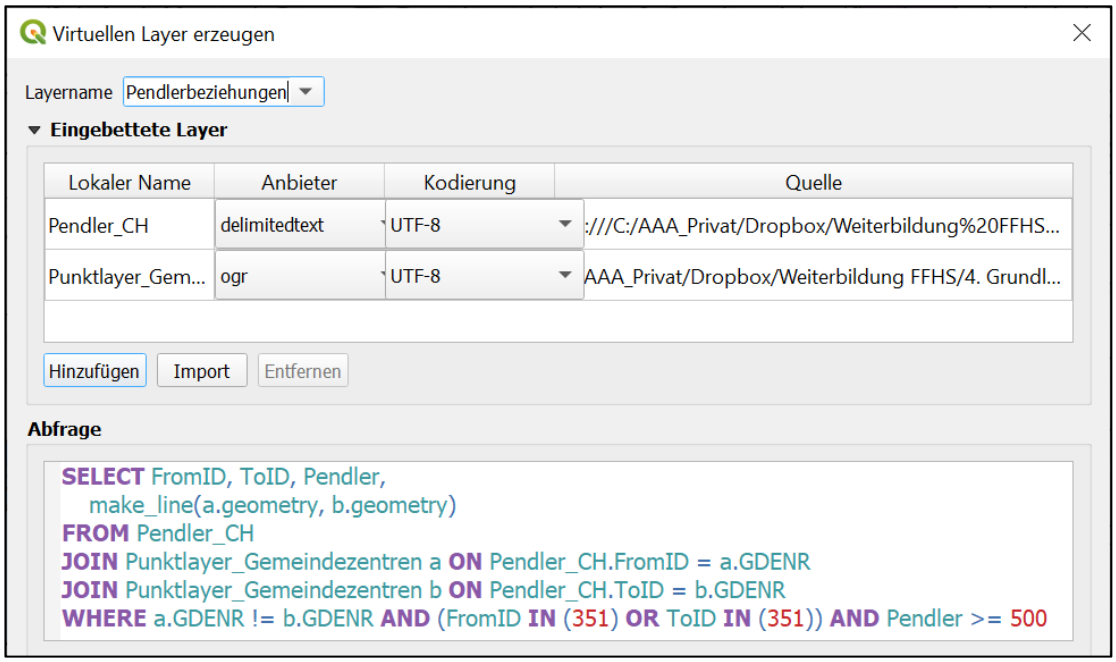
Der Punktlayer kann nun zur Erstellung eines virtuellen Linien-Layers verwendet werden, welcher jede einzelne Gemeinde mit allen anderen Gemeinden verbindet (d.h. gut 2'000 Verbindungen pro Gemeinde). Es werden also alle Gemeindemittelpunkte miteinander verbunden (jeweils in beide Richtungen, also 2x), was insgesamt rund 5'000'000 Beziehungen unter den Gemeinden ergibt. Zusätzlich kann an diese Beziehungen die Anzahl der Pendler angeknüpft werden. Dies geschieht über das Auslesen eines CSV-Files, in welchem für jede Von-Zu-Verbindung die entsprechende Pendlerzahl hinterlegt ist. Der Abfragecode für die Erstellung der Linien respektive Pendlerbeziehungen ist in Abbildung 2-8 festgehalten.

Abbildung 2-8: Generierung des virtuellen Linien-Layers für die Darstellung der Pendlerbeziehungen unter den Gemeinden



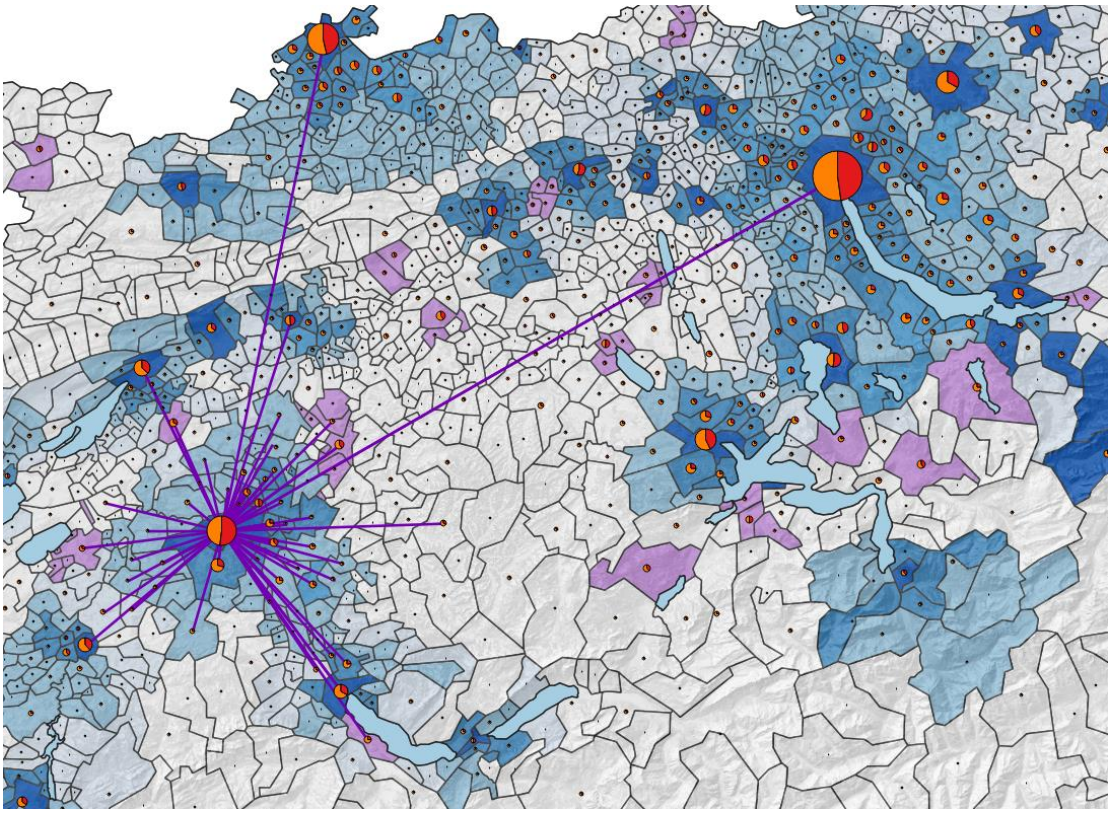
Da die Darstellung aller Beziehungen (in der Form eines virtuellen Layers) sehr viel Rechenzeit beanspruchen würde, werden nachfolgend nur «lokale» Analysen in Bezug auf eine Gemeinde, das Agglomerations- oder das Kantonsniveau vorgenommen. Zudem wird jeweils ein Schwellenwert für die Anzahl der einzublendenden Pendler definiert, damit die Darstellungen übersichtlich bleiben. Diese Einschränkung kann durch die Ergänzung des WHERE-Befehls in der Abfrage vorgenommen werden. Abbildung 2-9 zeigt beispielhaft ein solches Filter-Schema (nur Beziehungen, die von Bern ausgehen oder nach Bern gelangen (Bern hat die BFS-Gemeindenummer 351) ab einem Schwellenwert von mindestens 500 Pendler/innen).

Abbildung 2-9: Filterkriterien für Pendlerbeziehungen innerhalb des WHERE-Befehls



Diese führt bereits mal zu einer interessanten Darstellung der wichtigsten Pendlerbeziehungen (vgl. Abbildung 2-10 für Bern), zeigt aber die relativen Grössenverhältnisse bezüglich der Pendlerströme noch nicht auf. Deshalb sollen nun die Beziehungen anhand von Pfeilen dargestellt werden, deren Breite resp. Dicke von den Pendlerströmen abhängt. Dazu muss die Symbolsierungsebene des virtuellen Layers entsprechend angepasst werden.

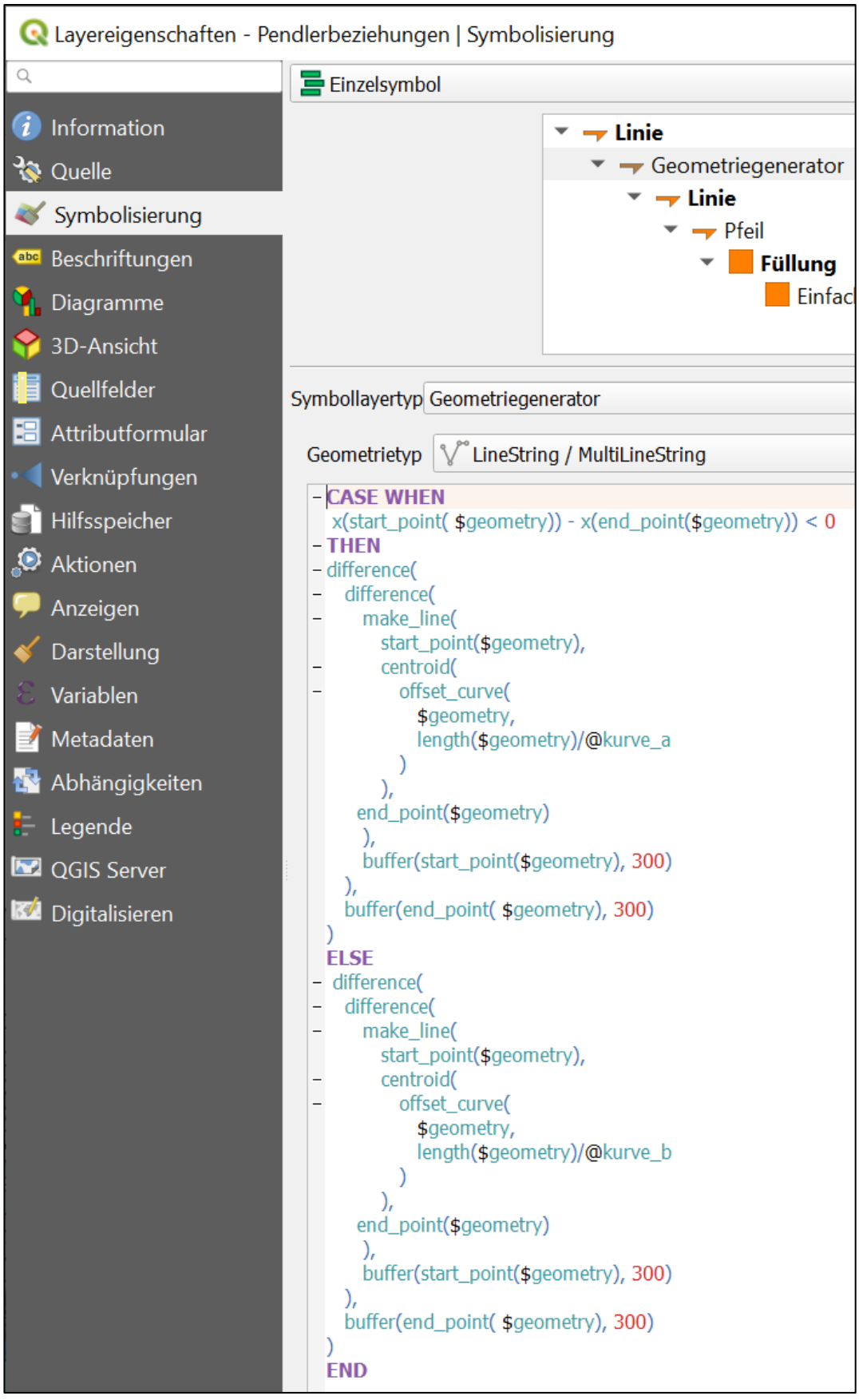
Abbildung 2-10: Pendlerbeziehungen (violette Linien) der Stadt Bern (Penderströme >= 500)



Als erstes definieren wir die Richtung der einzelnen Pfeile (von wo nach wo) sowie die Krümmung der Linie, damit die ganze Sache ein wenig dynamischer aussieht. In Abbildung 2-11 ist der Abfrage-Code für die Generierung der Linien ersichtlich. Ein Paar Details:

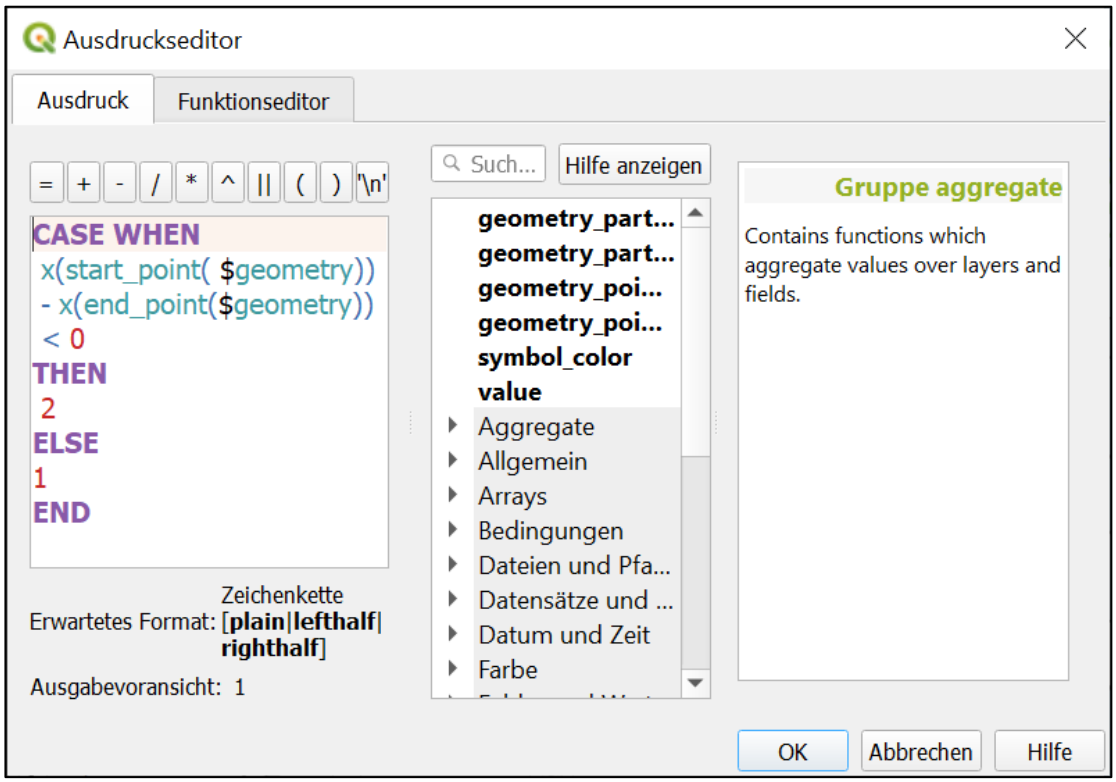
- Als erstes wird geprüft, ob der Startpunkt auf der X-Achse respektive im Koordinatensystem weiter westlich liegt als der Endpunkt.
- Auf dieser Basis kann danach entschieden werden, ob die Kurve des Pfeils nach oben oder nach unten gekrümmt werden soll, respektive in welche Richtung der Y-Achse eine Anpassung erfolgt.
- Die Krümmung wird über die globalen Projektvariablen «kurve_a» und «kurve_b» definiert, welche unter den Projekteigenschaften abgelegt sind (Reiter *Projekt → Eigenschaften*). Damit die beiden Kurven beieinanderliegen, wird kurve_a als minus kurve_b definiert.
- Zusätzlich wird ein Buffer zu den Gemeindezentren gelegt, damit die Pfeile mit einem gewissen Abstand zum jeweiligen Zentroid starten respektive enden.

Abbildung 2-11: Geometriegenerator-Abfrage für Darstellung der Pendler-Pfeile



Auch der Pfeiltyp wird aufgrund der Lokalisierung des Start- und Endpunkts definiert (über den Ausdruckseditor). Je nachdem wird Typ 1 (nur rechte Hälfte des Pfeils) oder Typ 2 (nur linke Hälfte des Pfeils) angezeigt (vgl. Abbildung 2-12). Dadurch zeigt die eine Hälfte des Pfeils die Zupendler nach Bern an, während die andere Hälfte die Wegpendler von Bern illustriert.

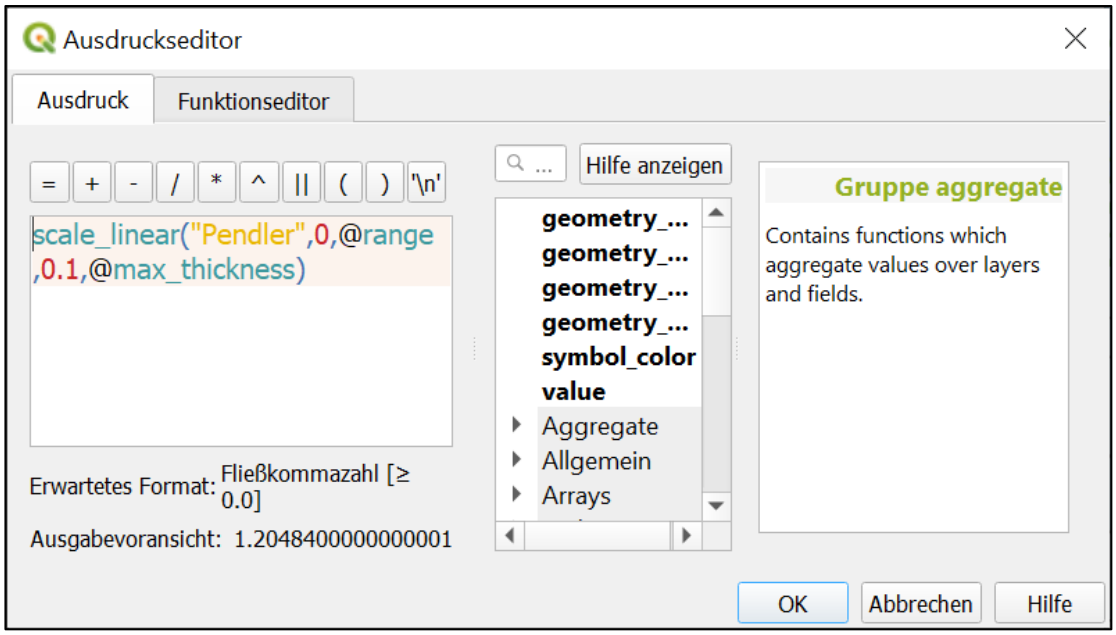
Abbildung 2-12: Definition der einzublendenden Pfeil-Hälfte



Nun kommen wir zur Definition der Pfeilbreite bzw. -dicke. Diese wird ebenfalls über eine Formel im Ausdruckseditor definiert (vgl. Abbildung 2-13): Die Pfeildicke wird anhand der Pendlerströme linear zwischen 0 und der maximal abzubildenden Pendlerströme (range) skaliert und dies auf die maximal definierte Dicke (max_thickness). Die beiden erwähnten Variablen sind wiederum global unter den Projekteigenschaften abgelegt.

Analog werden auch die Pfeilbreite am Anfang sowie die Dicke und Länge der Pfeilspitze definiert. Alternativ könnten diese auch um einen gewissen Faktor skaliert werden, damit sie besser oder eben weniger gut ersichtlich werden (bzw. kleiner erscheinen) in der Visualisierung.

Abbildung 2-13: Definition der Pfeildicke



Auch die Farbe der Pfeile kann über den Ausdrucksеditor bestimmte werden. In unserem Fall werden die Pfeile mit einer bestimmten Farbe eingefärbt, falls der Weg nach Bern führt (Zupendler) respektive mit einer anderen Farbe, falls der Weg von Bern weg führt (Wegpendler). Die Abfrage ist in Abbildung 2-14 zu finden.

Abbildung 2-14: Definition der Pfeilfarbe

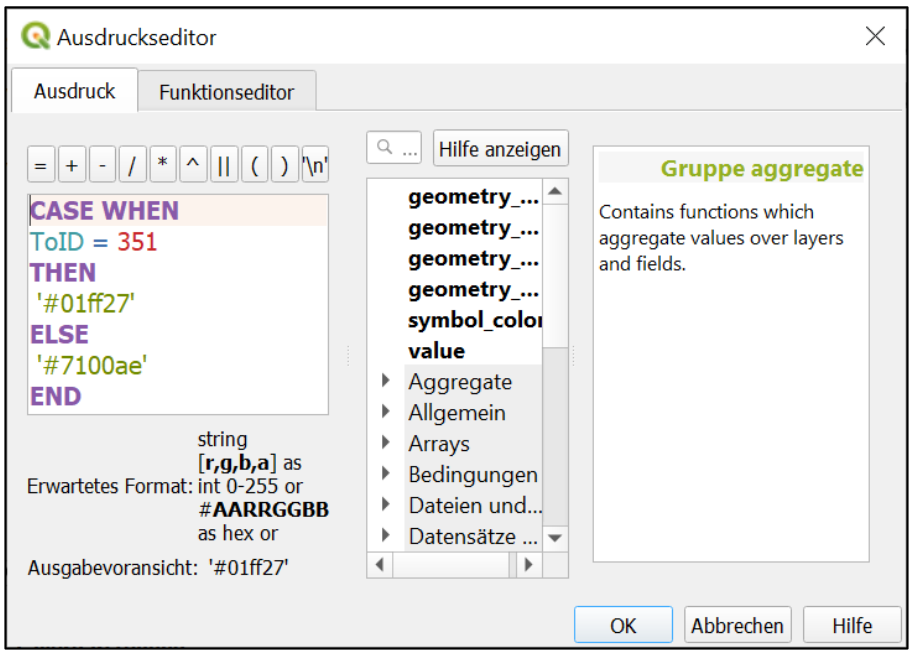


Abbildung 2-15 zeigt die resultierende Visualisierung für die Stadt Bern. Es werden nun insbesondere folgende Daten und Strukturen dargestellt:

- Woher die Zupendler/innen der Stadt Bern kommen (grüne Pfeile) und wohin die Stadtberner/innen pendeln (violette Pfeile).
- Wie die Siedlungsstrukturen in den jeweiligen Gemeinden aussehen (bezüglich Bevölkerung und Arbeitsplätzen).
- Städtischer Charkater der Gemeinden. Dadurch wird ersichtlich, dass die grössten Pendlerbeziehungen einerseits innerhalb der Agglomeration Bern bestehen und andererseits zu weiteren Kernstädten (insbesondere in der Region Bern-Mittelland).

Folgende Parameter können flexibel angepasst werden:

- Schwellenwert der anzuzeigenden Pendlerströme
- Grösse der Pfeile der Pendlerströme sowie die Grösse der Kuchendiagramme zur Darstellung der Siedlungsstrukturen.

Mögliche Ausbauoptionen der Visulisierung: Zusätzlich könnte nun beispielsweise noch das Nationalstrassen- und/oder Schienennetz eingeblendet werden, um eine visuelle Analyse zwischen Pendlerbeziehungen und Infrastrukturen zu ermöglichen.

Interpretation der Visualisierung respektive der dargestellten Daten und Zusammenhänge:

- Bern weist deutlich mehr Zupendler/innen (grüne Pfeile) als Wegbendler/innen (violette Pfeile) auf, was sich in einem stark positiven Pendlersaldo niederschlägt. Anders ausgedrückt: Am Morgen pendeln sehr viele Arbeitnehmer/innen in die Stadt Bern hinein und am Abend wieder hinaus.
- Auf Basis des dargestellten städtischen Charakters der Gemeinden und der Pendlerbeziehungen (Pfeile) wird ersichtlich, dass die meisten Zupendler/innen der Stadt Bern aus dem Agglomerationsgebiet stammen. Zudem bestehen einzelne grössere Wechselbeziehungen mit anderen Kernstädten (Zürich, Basel, Neuenburg, Fribourg und Thun), wobei hier die Pendlerströme etwas ausgeglichener sind (insbesondere nach Zürich).
- Bei näherer Betrachtung und unter Einbezug der Siedlungsstruktur (vgl. Abbildung 2-12) lässt sich gut erkennen, dass viele Zupendler/innen der Stadt Bern aus Gemeinden mit einer tiefen Arbeitsplatzdichte[13] respektive Schlafgemeinden[14] stammen, während Bern mit einer der schweizweit höchsten Arbeitsplatzdichten (mehr als 1 VZÄ pro Einwohner/in) eine Arbeitsgemeinde darstellt.

Wichtiger technischer Hinweis: Da sich der virtuelle Layer nicht richtig im QGIS-File speichern liess, respektive beim Laden des Files Probleme bereitet, wurde die Auswertung für Bern in einem separaten «fixen» Geometriedatenfile abgespeichert (unter dem Namen

[13] Definition: Verhältnis zwischen Einwohnern und Arbeitsplätzen.

[14] Der Name kommt daher, dass in solchen Gemeinden hauptsächlich gewohnt, nicht aber gearbeitet wird.

«Pendlerbeziehungen Bern»). In Anhang A wird das Vorgehen zur Fehlerbehebung beim Laden des QGIS-Files dargelegt.

Abbildung 2-15: Pendlerbeziehungen (2014) der Stadt Bern sowie Siedlungsstruktur (2016) der Gemeinden im Einzugsgebiet der Stadt Bern, (Pendlerbeziehungen > 500)

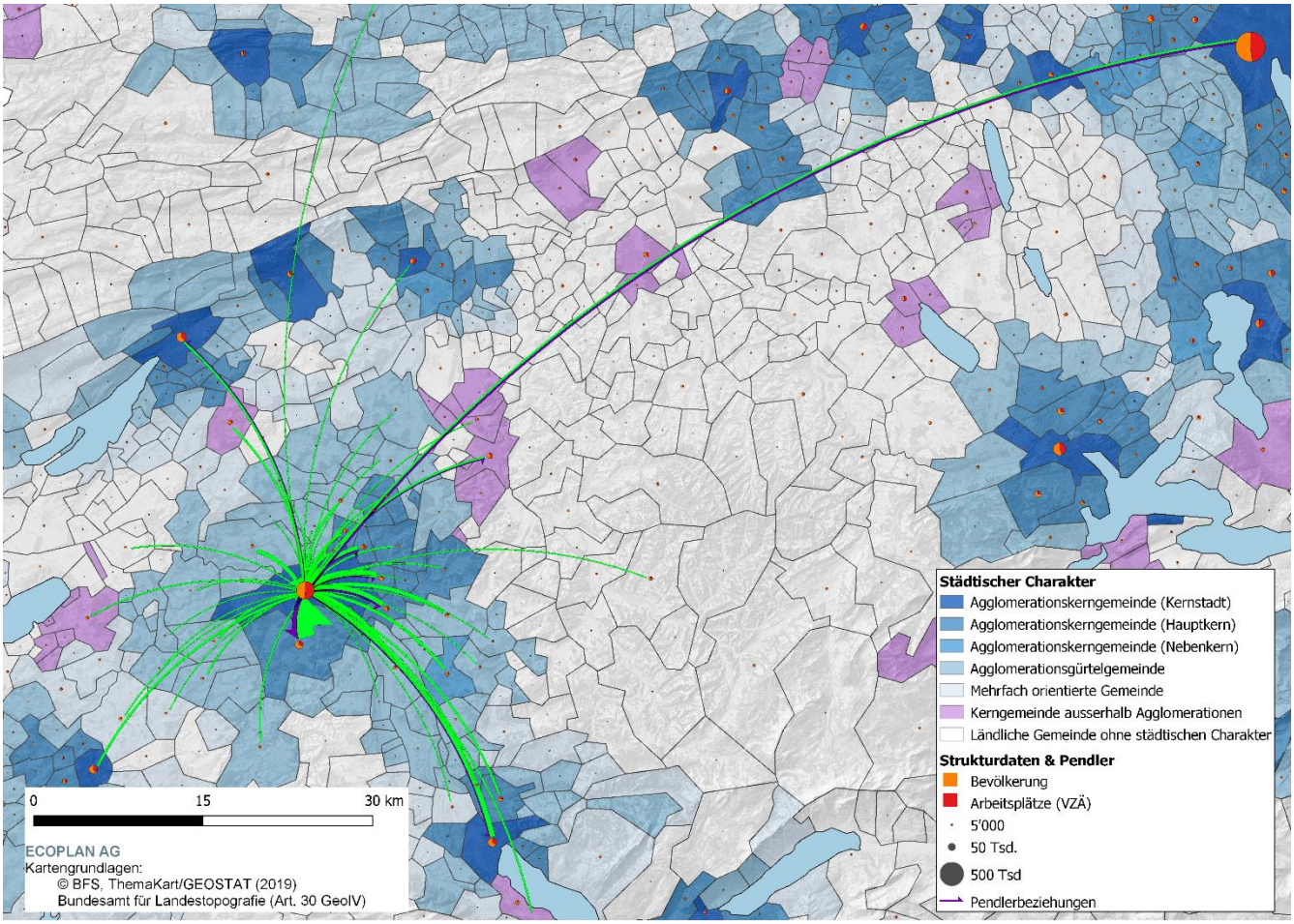
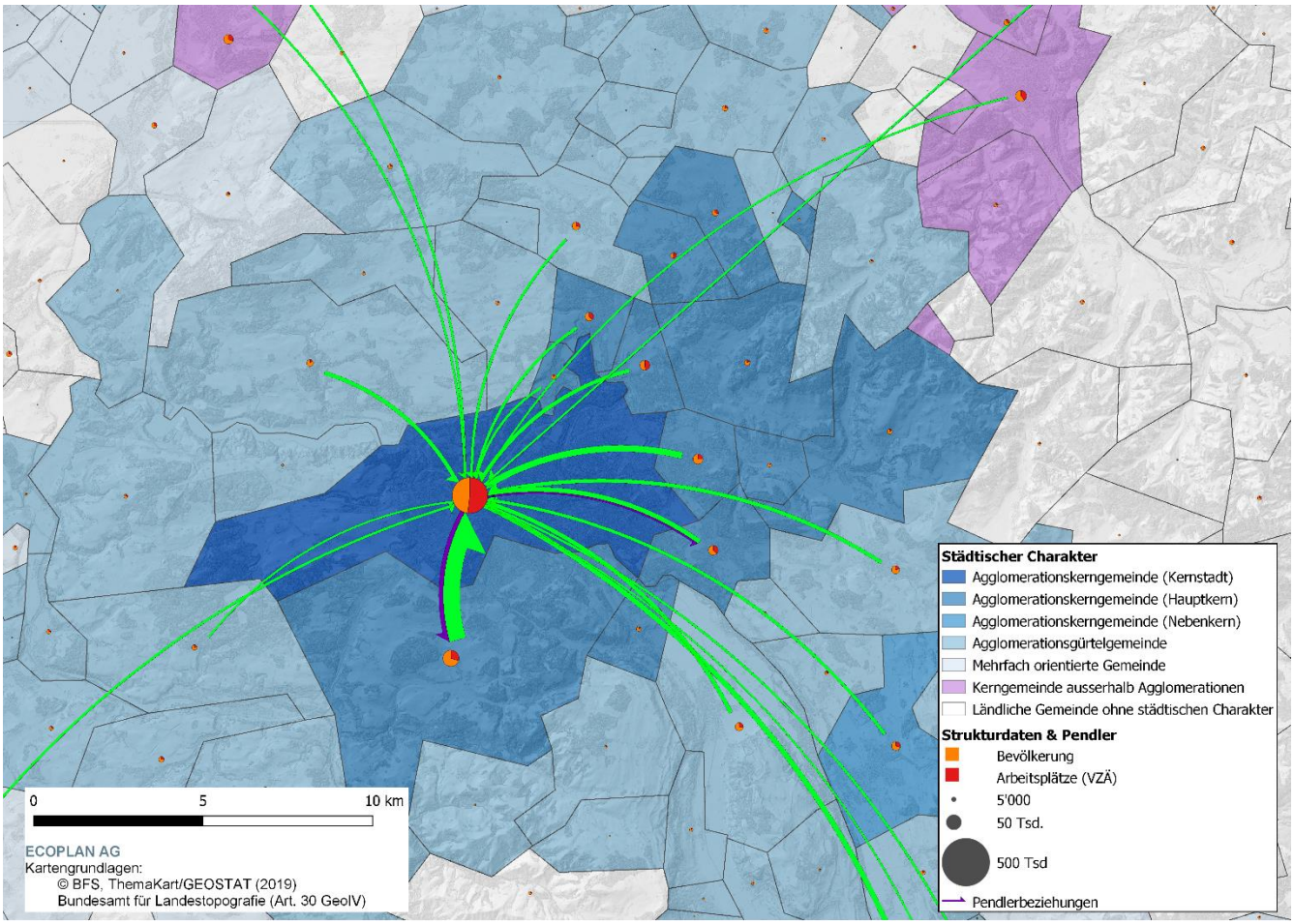


Abbildung 2-16: Pendlerbeziehungen (2014) der Stadt Bern sowie Siedlungsstruktur (2016) der Gemeinden im Einzugsgebiet der Stadt Bern, (Pendlerbeziehungen > 1'000)

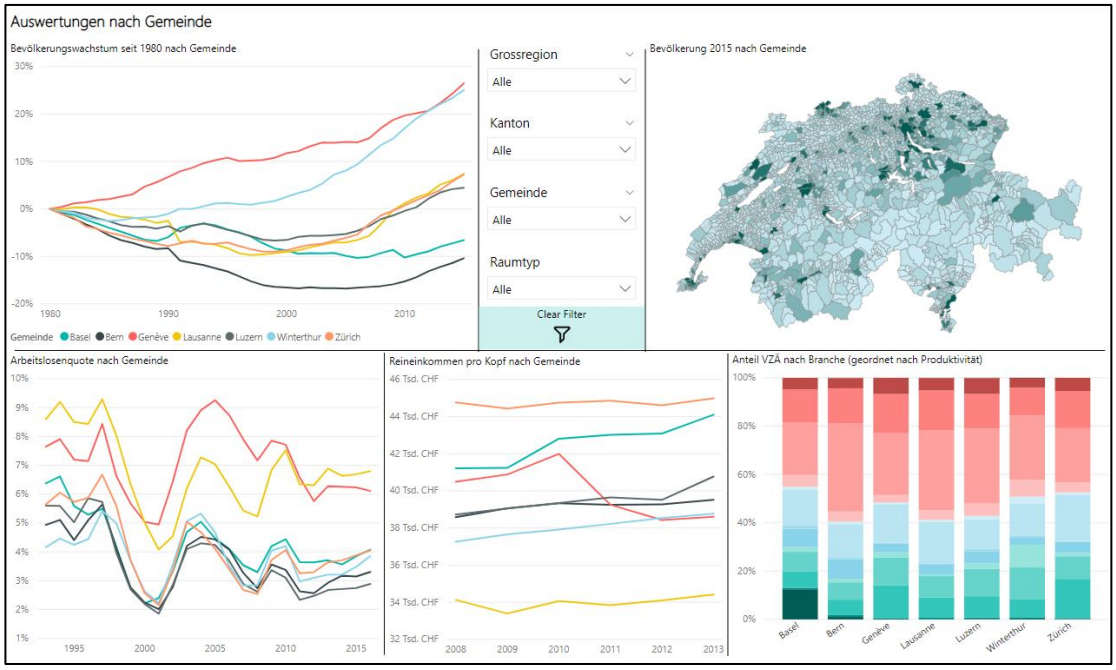


3 Verwendung von Python in Power BI

3.1 Problemstellung

Die interaktive Indikatorauswertung auf Gemeindeebene[15] (vgl. Abbildung 3-1), welche im Rahmen des CAS Visualisierung in Power BI erstellt wurde, soll auf Basis des Python-Plugins um eine Visualisierung (Visual) ergänzt werden. Zudem soll das Einlesen der Daten in die Gemeindedatenbank, welche der Visualisierung zugrunde liegt, für einzelne (erste) Indikatoren mithilfe eines Python-Skripts automatisiert werden.

Abbildung 3-1: Visualisierung in Power BI (Dashboard)



3.2 Methodik

Für die Lösung der Problemstellung wird die Business-Intelligence-Software **Power BI**[16] verwendet, welche in der Basisversion frei verfügbar ist. Aufgrund der «Offenheit» z.B. gegenüber von Plugins aus anderen Programmen (z.B. ArcGIS) resp. Programmiersprachen (z.B. Python) weist Power BI eine relativ umfangreiche Entwickler- und User-Community auf.

[15] Vgl, Web-Applikation unter: https://app.powerbi.com/view?r=eyJrIjoiN2Q5MjYzZjEtMWYxMS00YmFhLTllZ-mQtMjMwMzQ2MmRmOTc5IiwidCI6ImZlNWRhNTQ0LTAyMmYtNDJiMy04YTVhLTExZjBhNTNjNGM2NCIsImMi-OjI9

[16] Online im Internet: https://powerbi.microsoft.com/de-de/ [04.01.2020].

Die Erstellung der Python-Visualisierung erfolgt auf Basis der bereits bestehenden Gemeindedatenbank (vgl. Abbildung 3-2), welche in Power Query (Datenbanktool in Power BI und Excel) aufgebaut wurde und der aktuellen interaktiven Visualisierung (vgl. Abbildung 3-1) zugrunde liegt. Hierzu wird die Grafik zum «Reineinkommen pro Kopf» durch ein Python-Visual ersetzt.

Das automatisierte Einlesen der Daten (für die Daten zur Bevölkerung und zu den Arbeitsplätzen) in Power Query respektive der Gemeindedatenbank wird auf Basis eines Python-Skripts umgesetzt (Hinweis: Das Python-Skript entspricht einer gekürzten Version des Skripts, welches zur Generierung der Strukturdaten in Kapitel 2 verwendet wurde).

Abbildung 3-2: Verknüpfung der Daten in Power BI (Gemeindedatenbank)

3.3 Datengrundlagen

Als Datenquelle werden die verschiedenen Indikatoren des Monitorings von regiosuisse benutzt[17], wobei nur eine Auswahl davon tatsächlich verwendet wird. Konkret wird auf folgende Datensätze des BFS zurückgegriffen:

- Bevölkerung (BFS, STATPOP)
- Unternehmen und Arbeitsplätze (BFS, STATENT)
- Arbeitslosigkeit (SECO, AMSTAT)
- Reineinkommen (ESTV, Steuerstatistik)

3.4 Detaillierte Vorgehensweise

Damit Python in Power BI verwendet werden kann, muss Python zuerst in Power BI aktiviert und verknüpft werden. Zudem müssen die Bibliotheken bzw. Pakete[18] installiert werden, welche in den Skripten verwendet werden sollen. Microsoft (2019) bietet hierzu mit dem Artikel «Run Python scripts in Power BI Desktop»[19] eine gute Anleitung.

3.4.1 Automatisiertes Einlesen der Daten mittels Python-Skript

Als erstes sollen nun die Daten zur Bevölkerung und zu den Arbeitsplätzen automatisiert in Power Query respektive die Gemeindedatenbank geladen werden. Hierzu wird das leicht gekürzte[20] Python-Skript aus Kapitel 2 verwendet (vgl. Abbildung 3-3).[21] Innerhalb des Power-Query-Editors[22] kann das Python-Skript über das Hauptmenu *Home → Neue Quelle → Sonstige → Python-Skript* eingegeben und ausgeführt werden.

Die über das Python-Skript generierten Pandas-Dataframes können nun in Power Query ausgewählt und geladen werden (vgl. Abbildung 3-4). Wenn die Power-Query-Datenbank nun aktualisiert wird, so wird das Python-Skript jeweils mitausgeführt und dadurch die gewünschten Daten zur Bevölkerung und den Arbeitsplätzen aktualisiert.

Hinweis: Alternativ könnten die Daten auch über das durch das Python-Skript produzierte und abgelegte CSV-File in Power Query geladen werden. In diesem (manuellen) Fall wären die Daten jedoch statisch hinterlegt und würden entsprechend bei der Aktualisierung der Power-Query-Datenbank nicht neu von der STAT-Tab-Website geladen.

---

[17] Vgl. regiosuisse-Indikatorenset, Online im Internet: https://regiosuisse.ch/sites/default/files/2019-07/regiosuisse-Indikatorenset_2019.pdf [04.01.2020].

[18] Im vorliegenden Falle sind dies die Pakete: pandas, requests, matplotlib, seaborn, plotly, statsmodels.

[19] Online im Internet: https://docs.microsoft.com/en-us/power-bi/desktop-python-scripts [05.01.2020].

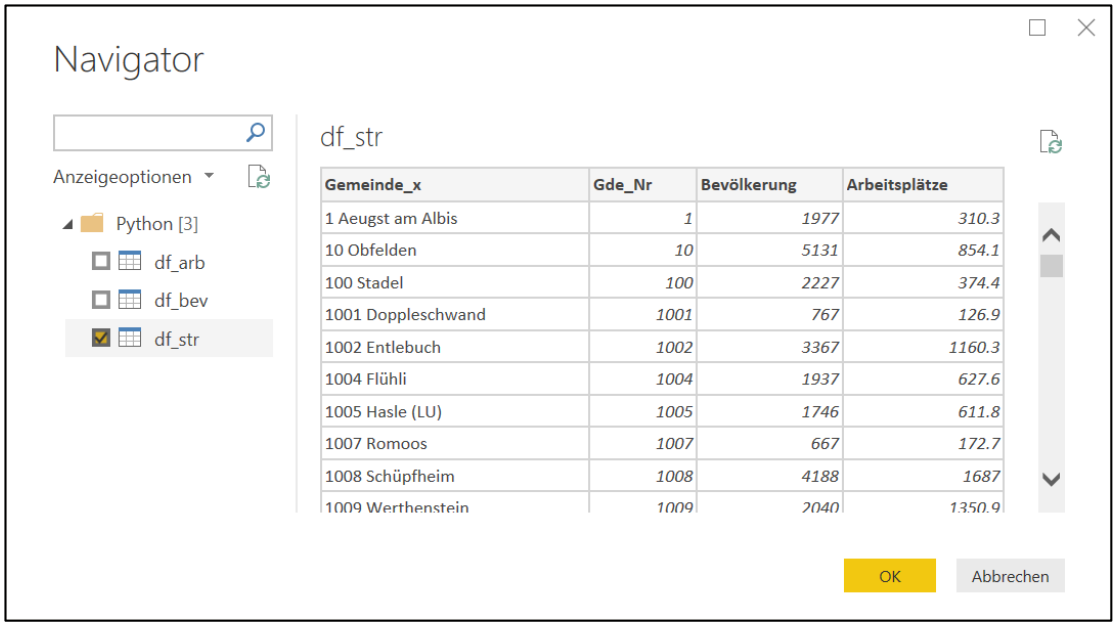
[20] Es wurden alle Code-Zeilen entfernt, welche eine visuelle Rückgabe (Prints) zum Ziel haben, da diese nicht dargestellt werden.

[21] Vgl. auch Python-Skript «Bezug_der_Strukturdaten» im Ordner «Semesterarbeit Power BI\Python-Skripte».

[22] Um dahin zu gelangen kann beispielsweise in Power BI im Hauptmenü unter *Home* auf *Abfragen bearbeiten* geklickt werden.

Abbildung 3-3: Laden der Strukturdaten mittels Python-Skript im Power-Query-Editor

Abbildung 3-4: Laden der Pandas-Dataframes in Power Query



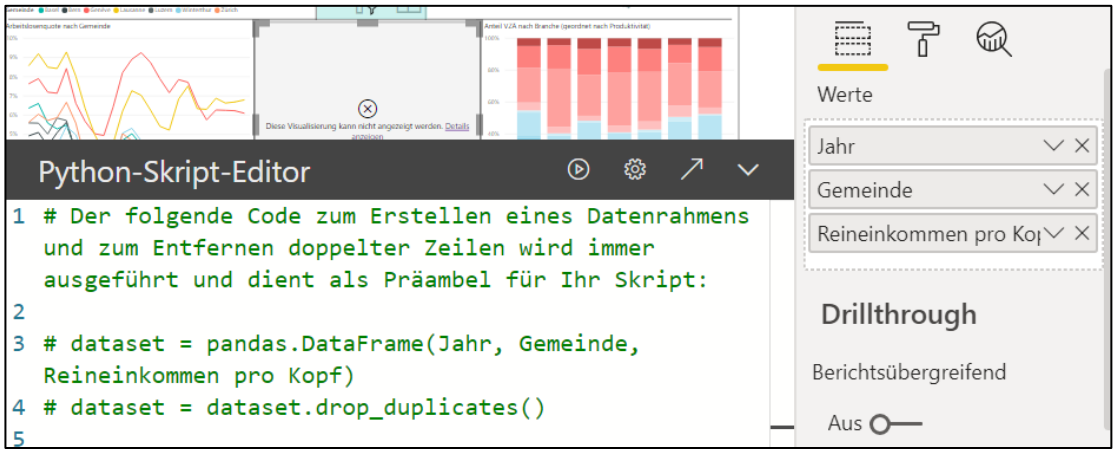
Legende: df_bev: Dataframe Bevölkerung
df_arb: Dataframe Arbeitsplätze
df_str: Dataframe Strukturdaten

3.4.2 Integration eines Python-Visuals in ein bestehendes Power-BI-Dashboard

Nun soll ein Python-Visual in das bestehende Dashboard (vgl. Abbildung 3-1) integriert werden. Hierzu wird als erstes ein Platzhalter für ein solches Visual hinzugefügt (vgl. Abbildung 3-5). Danach können die gewünschten Datenfelder aus der Gemeindedatenbank dem Wertebereich des Visuals hinzugefügt werden.[23] Diese Datenfelder werden automatisch einem Pandas-Dataframe hinzugefügt, welcher anschliessend über den Python-Skript-Editor manipuliert werden kann (vgl. Abbildung 3-6).

Abbildung 3-5: Hinzufügen eines Python-Visuals in Power BI

Abbildung 3-6: Generierung des Pandas-Dataframe auf Basis der Daten im Wertebereich des Python-Visuals



[23] Im vorliegenden Fall sind dies die Datenfelder: Jahr, Gemeinde, Reineinkommen pro Kopf.

Durch die wenigen in Abbildung 3-7 dargestellten Code-Zeilen kann nun auf Basis der Python-Bibliotheken «matplotlib» und «seaborn» ein Visual erstellt werden, welches die exakt gleichen Daten darstellt wie das ursprüngliche Visual (vgl. Abbildung 3-8, Visual unten-mitte)[24] – lediglich die Darstellung der Farben und Rahmen entsprechen nun der seaborn-Bibliothek (anstatt den Power-BI-Grundlagen). Die grafische Gestaltung des Visuals könnte nun weiter auf die restlichen Power-BI-Grafiken angepasst werden (oder umgekehrt). Dies ist jedoch nicht mehr das Ziel[25] der vorliegenden Arbeit, sondern wird allenfalls zu einem späteren Zeitpunkt erfolgen, wenn die Grafik tatsächlich verwendet werden soll.

Abbildung 3-7: Code zur Generierung des Python-Visuals mit der Darstellung des Reineinkommens pro Kopf, pro Gemeinde im Zeitverlauf[26]

Python-Skript-Editor

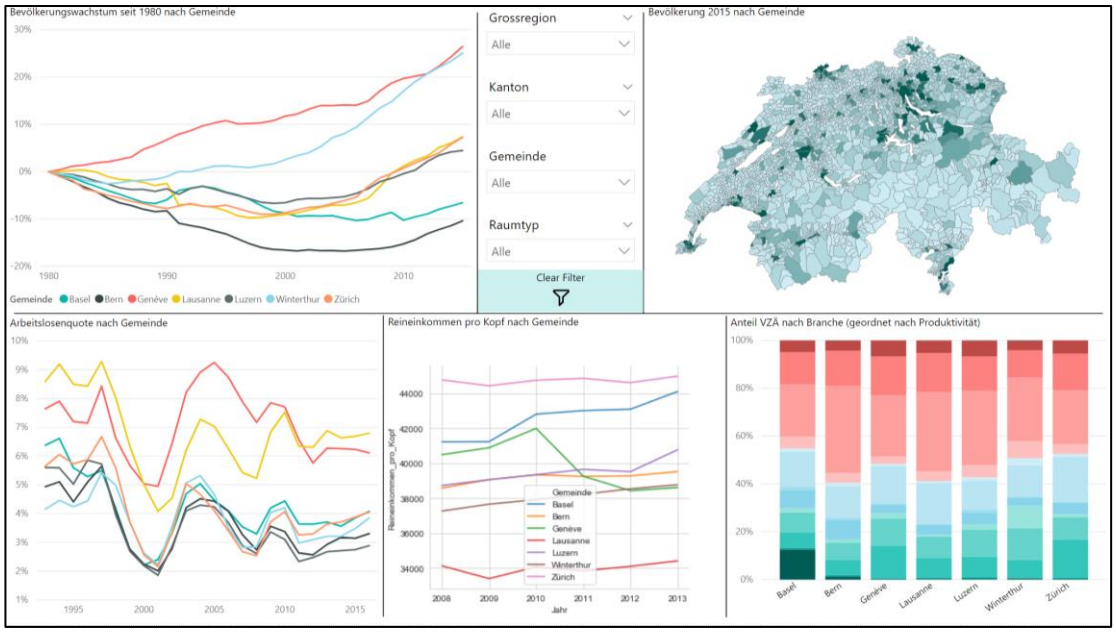
```
1  # Der folgende Code zum Erstellen eines Datenrahmens und zum Entfernen doppelter
2
3  # dataset = pandas.DataFrame(Jahr, Gemeinde, Reineinkommen_pro_Kopf)
4  # dataset = dataset.drop_duplicates()
5
6  # Fügen oder geben Sie hier Ihren Skriptcode ein:
7
8  import seaborn as sns; sns.set_style('whitegrid')
9  import matplotlib.pyplot as plt
10
11 sns.lineplot(x='Jahr', y='Reineinkommen_pro_Kopf', hue='Gemeinde', data=dataset)
12 sns.despine()
13 plt.show()
```

[24] Leider können Python-Visuals nur in der Power-BI-Pro-Version als Web-Applikation veröffentlicht werden. Um das Visual in Aktion (interaktiv) zu sehen, muss daher die Power-BI-Datei «Semesterarbeit_Power BI_Matthias Setz.pbix» verwendet und innerhalb der Power-BI-Entwicklerumgebung gestartet werden. Das Visual befindet sich auf der Seite «Gemeinde_v2»

[25] Das Ziel war, eine Grafik auf Basis eines Python-Skripts in Power BI zu erstellen.

[26] Vgl. auch Python-Skript «Python-Skript-Editor» im Ordner «Semesterarbeit Power BI\Python-Skripte».

Abbildung 3-8: Überarbeitete Visualisierung in Power BI (Dashboard)



Eine schöne Nebensache ist nun, dass in Power BI die Filterkriterien bzw. die Filter-Abhängigkeiten zwischen den Grafiken gesteuert und individuell angepasst werden können. So werden im Python-Visual nur die bevölkerungsmässig fünf grössten Gemeinden dargestellt (welche aufgrund der bereits gesetzten Filter übrigbleiben). Zudem reagiert die Grafik einerseits auf die Filter im Zentrum des Dashboards und andererseits auf angeklickte Gemeinden in der Kartenabbildung (rechts-oben im Dashboard).

Mögliche Weiterentwicklungen:

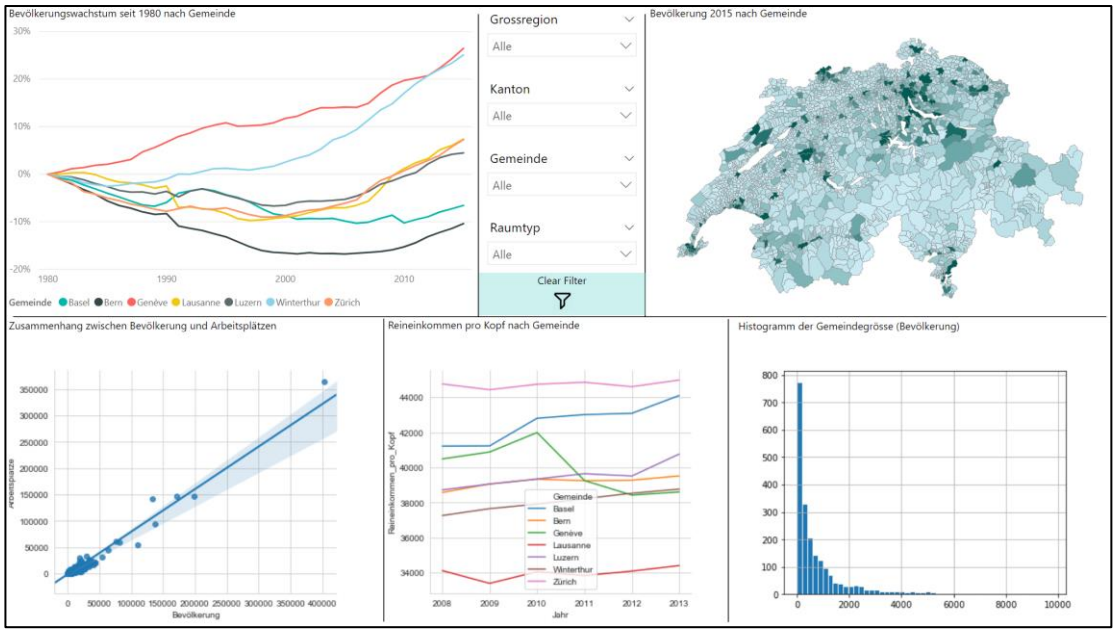
- In einem nächsten Schritt könnte nun (neben der visuellen Abstimmung) versucht werden, das Python-Visual nicht nur reaktiv (auf die anderen Darstellungen und Filter) sondern auch aktiv bzw. interaktiv zu gestalten. Hierzu könnte beispielsweise die Bibliothek «plotly» verwendet werden.[27]
- Zudem könnten auch statistische Modelle geschätzt (z.B. Machine-Learning-Algorithmen) und angezeigt werden. Die untenstehende Abbildung[28] zeigt beispielhaft einen ersten Versuch für ein solches Visual (unten-links): Scatterplot der Bevölkerung vs. Arbeitsplätze pro Gemeinde, inkl. Regressionslinie.[29]

[27] Online im Internet: https://plot.ly/python/ [05.01.2020].

[28] Leider können Python-Visuals nur in der Power-BI-Pro-Version als Web-Applikation veröffentlicht werden. Um das Visual in Aktion (interaktiv) zu sehen, muss daher die Power-BI-Datei «Semesterarbeit_Power BI_Matthias Setz.pbix» verwendet und innerhalb der Power-BI-Entwicklerumgebung gestartet werden. Das Visual befindet sich auf der Seite «Gemeinde_v3»

[29] Auch hier passt sich die Regression «automatisch» auf die gewählten Filter an, da das Visual entsprechend verknüpft wurde.

Abbildung 3-9: Visualisierung in Power BI (Dashboard) mit zusätzlichen Python-Visuals

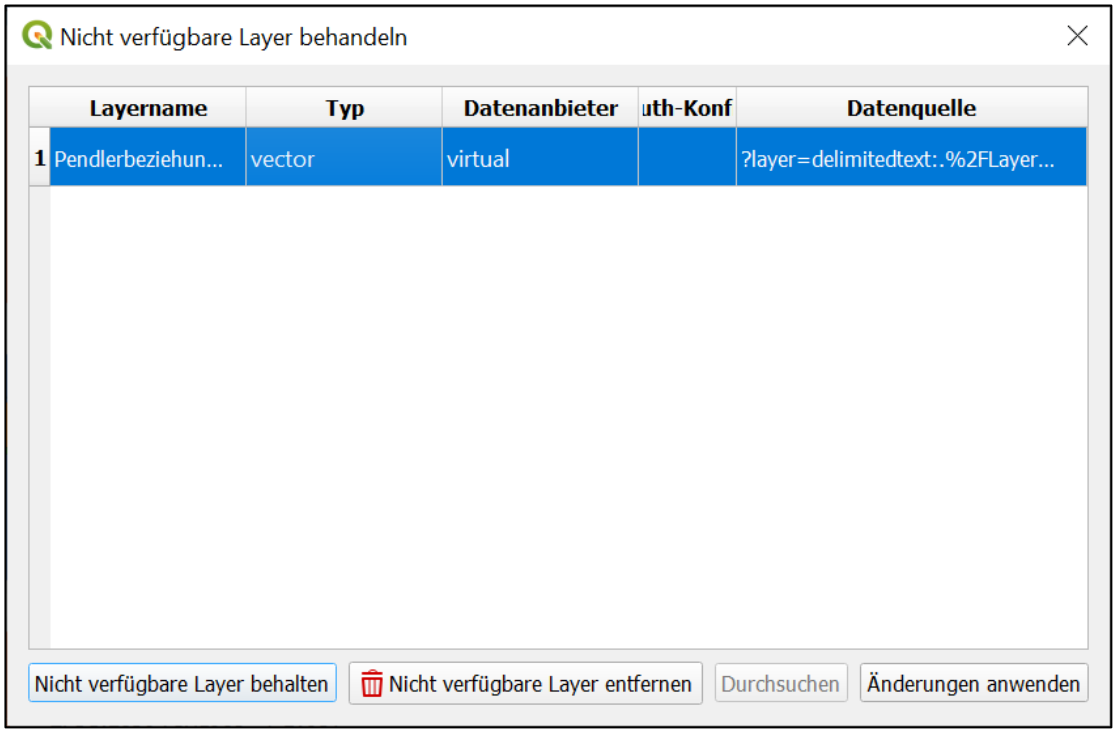


4 Anhang A: Vorgehen zur Korrektur der Fehlermeldung beim Laden von QGIS (Nicht verfügbare Layer behandeln)

Problem: Beim Laden des QGIS-Files erscheint jeweils die Fehlermeldung, dass der virtuelle Layer der Pendlerbeziehungen (Pfeile) nicht gefunden resp. erstellt werden konnte (vgl. Abbildung 4-1).

Lösung: Als erstes muss die «Fehlerbehandlung» abgebrochen werden. Hierzu wird das Fenster über das «X» (rechts-oben) geschlossen (vgl. Abbildung 4-1).[30] In einem zweiten Schritt wird der virtuelle Layer mit der rechten Maustaste angewählt und über «Virtuellen Layer bearbeiten…» (vgl. Abbildung 4-2) gelangt man zum ursprünglichen Eingabefenster für den virtuellen Layer (vgl. Abbildung 4-3). Nun müssen die beiden eingebetteten Layer gelöscht («Entfernen») und anschliessend wieder hinzugefügt («Import») werden. Abschliessend muss auf «Hinzufügen» (unten-rechts) gedrückt werden, damit der fehlerhafte Layer vom System mit einem funktionierenden Layer überschrieben wird.

Abbildung 4-1: Fehlermeldung beim Starten von QGIS



[30] Falls alternativ die Schaltfläche «Nicht verfügbare Layer behalten» angewählt wird, dann produziert QGIS einen Programmabsturz. Falls «Nicht verfügbare Layer entfernen» gedrückt wird, so wird der virtuelle Layer gelöscht und muss entsprechen komplett neu eingerichtet werden.

Abbildung 4-2: Virtuellen Layer bearbeiten

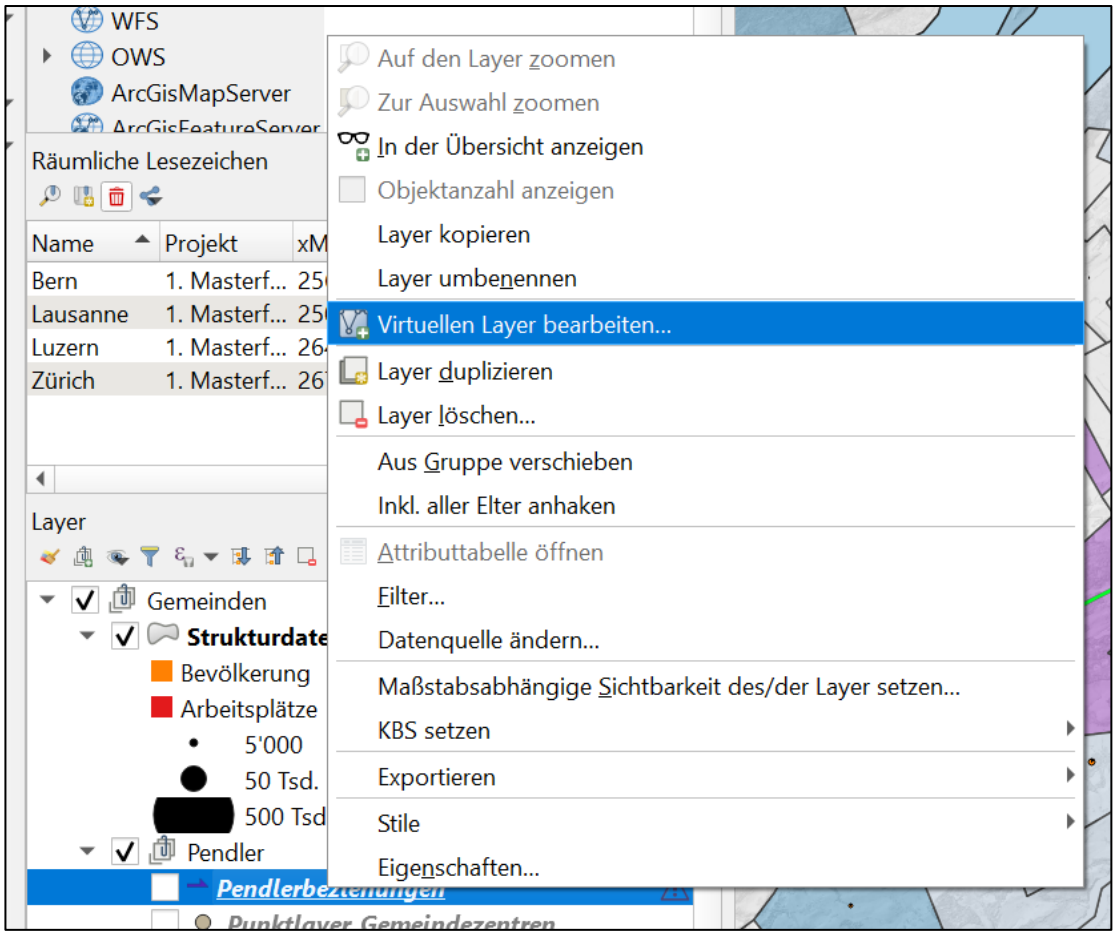
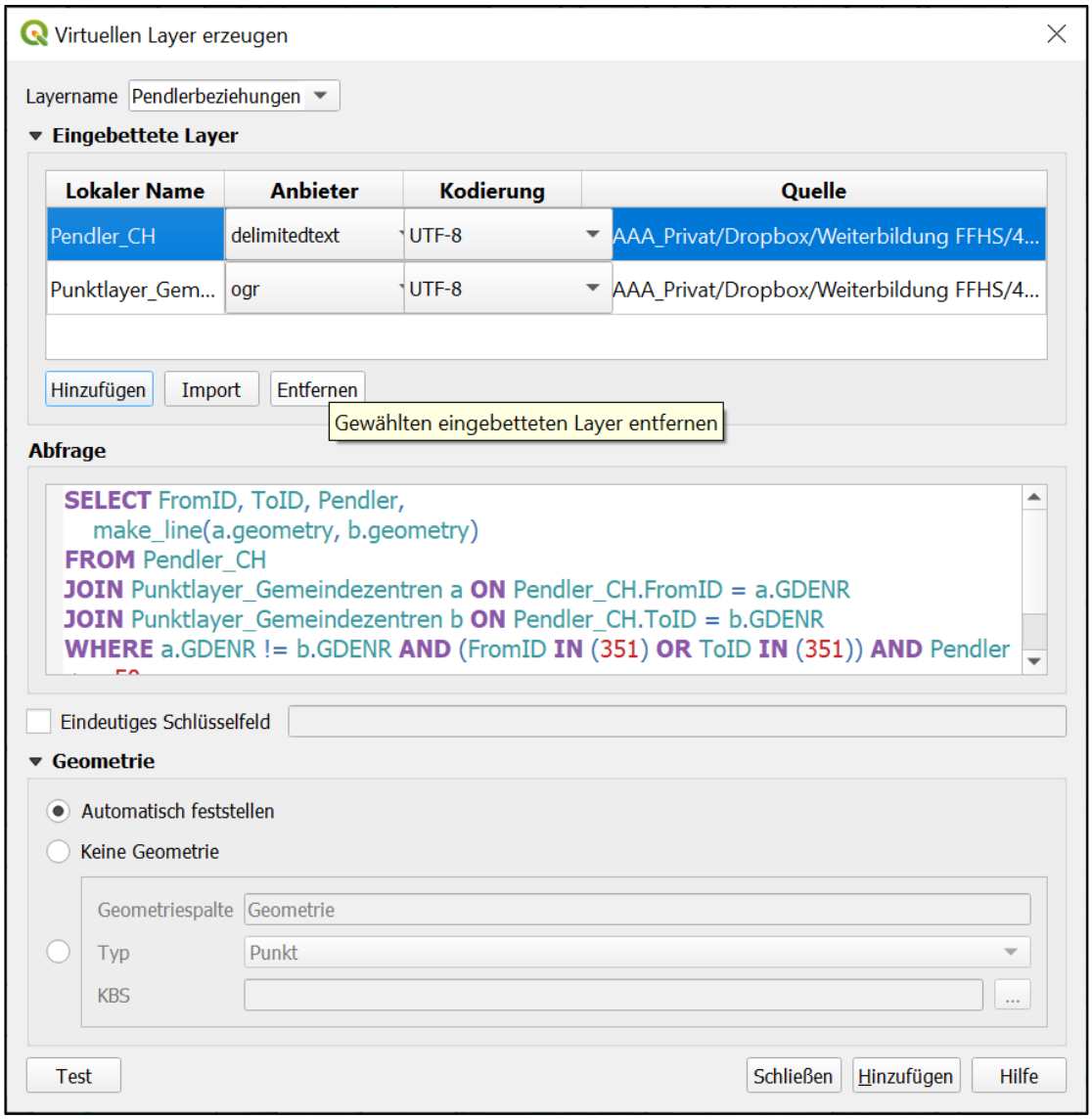


Abbildung 4-3: Eingabefenster für den virtuellen Layer



Literaturverzeichnis

Anita Graser (2019)
Blogartikel: Flow maps in QGIS – no plugins needed!, Online im Internet: http://planet.qgis.org/planet/tag/flows/ [27.12.2019].

Microsoft (2019)
Blogartikel: Run Python scripts in Power BI Desktop, Online im Internet: https://docs.microsoft.com/en-us/power-bi/desktop-python-scripts [05.10.2020].

regiosuisse (2019)
Regionenmonitoring: Die regionalwirtschaftliche Entwicklung in der Schweiz.Online im Internet: https://regiosuisse.ch/monitoring-der-regionalwirtschaftlichen-entwicklung [26.12.2019].