# ETL_Influence_factors

December 22, 2022

# APPENDIX 1 - ETL Process

## 0.1 ETL process for establishing input table for modelling influence factors on the share of public transport subscriptions {-}

The script is used to go through all the necessary steps in order to process the data according to the Master's thesis ***Modelling of factors influencing the share of public transport tickets in Swiss municipalities including cluster analysis*** from Gabriel Peier to establish a working database for the modelling of possible influence factors on Public Transport in Switzerland.

**Important notes:**

**Data sources:** All data are can be accessible free of charge and are found here (with name according to chapter 4)

- ga_hta_list: opentrasportdata.swiss
- verbundabo_list: opentrasportdata.swiss
- STATPOP2020_GMDE: Federal Statistical Office
- population: Federal Statistical Office
- stations_list_bav: opentrasportdata.swiss
- stop_count: opentrasportdata.swiss
- town_directory: cadastre
- cars_per_municipality: Federal Statistical Office
- inbound_comm: Federal Statistical Office
- outbound_comm: Federal Statistical Office
- dist: NPVM data source (2 different zip files to download; "OeV_Reisezeit_Distanz" and "Strasse_Reisezeit_Distanz" with each 2 corresponding mtx files)

**Storage:**

- The personal Google Drive account from Gabriel Peier was used to store all data, scripts, outputs and visualizations.
- Due to storage limitations, the data could not be stored in the GitHub Repository

- Access can be granted to the whole Master's Thesis Drive storage via: **gabrielpeier@gmail.com** (this can make the process easier)
- If used in your own Drive Storage: Adapt all pathes accordingly in the script: All Data must be placed in the Data folder with the sub-pathes as described in the different chapters of this script, otherwise adapt it.

**GitHub Repository (freely available):** https://github.com/Icelander169/MasterThesis

# 1 Set connection to Google Drive

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Change present working directory

[2]:
```
%cd /content/drive/MyDrive/MasterThesis
```

```
/content/drive/MyDrive/MasterThesis
```

# 2 Git Handling

These fields have to be adapted when used from someone else!

[3]:
```
!git config --global user.email "gabriel.peier@stud.hslu.ch"
```

[323]:
```
username = "Icelander169"
git_token = "***"  # never save file with Key token visible!
repository = "MasterThesis"
```

[ ]:
```
# !git init Scripts
```

[5]:
```
%cd Scripts
```

```
/content/drive/MyDrive/MasterThesis/Scripts
```

[324]:
```
!git add . # adding changes for commitment
```

[325]:
```
!git status
```

```
On branch test
Changes to be committed:
  (use "git reset HEAD <file>…" to unstage)
```

```
        modified:   ETL_Influence_factors.ipynb
```

[ ]:
```
!git remote add origin1 https://{git_token}@github.com/{username}/{repository}.
↪git
!git remote -v
# delete output afterwards!!
```

[327]:
```
!git commit -m "Final cleaning"
```

```
[test 74296d8] Final cleaning
 1 file changed, 1 insertion(+), 1 deletion(-)
 rewrite ETL_Influence_factors.ipynb (73%)
```

[328]:
```
!git push origin1 test
```

```
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 25.71 KiB | 774.00 KiB/s, done.
Total 3 (delta 2), reused 0 (delta 0)
remote: Resolving deltas: 100% (2/2), completed with 2 local objects.
To https://github.com/Icelander169/MasterThesis.git
   b0ccf5f..74296d8  test -> test
```

[329]:
```
!git remote remove origin1
!git remote -v
```

[ ]:
```
# !git checkout -b test
```

```
M       01_Reading_Data.ipynb
Switched to a new branch 'test'
```

## 3 Importing packages

[12]:
```
import pandas as pd
import numpy as np
# import scanpy as sc
from scipy.io import mminfo,mmread # handlings sparse matrices
import copy
import re # for regular expressions
!pip install mysql-connector-python # to install mysql connector!
import mysql.connector
from sqlalchemy import create_engine
import csv
```

```python
import sqlite3
from functools import reduce  # for multiple merging
import requests # for downloading
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting mysql-connector-python
  Downloading mysql_connector_python-8.0.31-cp38-cp38-manylinux1_x86_64.whl
(23.5 MB)
     |                  | 23.5 MB 1.1 MB/s
Requirement already satisfied: protobuf<=3.20.1,>=3.11.0 in
/usr/local/lib/python3.8/dist-packages (from mysql-connector-python) (3.19.6)
Installing collected packages: mysql-connector-python
Successfully installed mysql-connector-python-8.0.31
```

# 4 Loading Data

In this section, all previously downloaded data is loaded into the Colab environment.

[13]:
```python
ga_hta = pd.read_excel("../Data/0_Raw/ga_hta_list.xlsx")
ga_hta
```

[13]:
```
       Jahr_An_Anno  PLZ_NPA        GA_AG  GA_AG_flag  HTA_ADT_meta-prezzo  \
0              2012     1000    72.000000         NaN                976.0
1              2012     1003   744.000000         NaN               3195.0
2              2012     1004  1919.000000         NaN               8167.0
3              2012     1005   860.000000         NaN               4021.0
4              2012     1006  1279.000000         NaN               5366.0
…                 …        …            …          …                    …
31854          2021     9652    56.000000         NaN                286.0
31855          2021     9655    11.795455         1.0                107.0
31856          2021     9656    22.000000         NaN                194.0
31857          2021     9657    33.000000         NaN                246.0
31858          2021     9658    63.000000         NaN                399.0

       HTA_ADT_meta-prezzo_flag
0                           NaN
1                           NaN
2                           NaN
3                           NaN
4                           NaN
…                             …
31854                       NaN
31855                       NaN
31856                       NaN
31857                       NaN
```

```
31858                        NaN

[31859 rows x 6 columns]
```

[14]:
```
fn_tck = pd.read_excel("../Data/0_Raw/verbundabo_list.xlsx") #regional fare␣
↪network ticket
fn_tck
```

[14]:
```
       Jahr_An_Anno  PLZ_NPA Verbund_Communaute_Comunita  \
0              2017     1001                         ZVV
1              2017     1003                         ZVV
2              2017     1004                         ZVV
3              2017     1005                         ZVV
4              2017     1006                         ZVV
…                 …      …                           …
28862          2021     9656                    OSTWIND
28863          2021     9657                    OSTWIND
28864          2021     9658                    OSTWIND
28865          2021     9658                         ZVV
28866          2021     9721                 Arcobaleno

       Anzahl_Nombre_Quantita  Flag
0                    2.985232   3.0
1                    2.985232   3.0
2                    2.985232   3.0
3                    2.985232   3.0
4                    2.985232   3.0
…                         …    …
28862               29.000000   NaN
28863               42.000000   NaN
28864               38.000000   NaN
28865                3.010000   3.0
28866                4.850000   3.0

[28867 rows x 5 columns]
```

In the first population list, we get the data about age segments, country of origin, gender and marital status:

[15]:
```
population_1 = pd.read_excel('../Data/0_Raw/population.xlsx', header=[2]) #␣
↪header = 2 due to unneccessary rows at beginning
population_1
```

```
/usr/local/lib/python3.8/dist-packages/openpyxl/worksheet/header_footer.py:48:
UserWarning: Cannot parse header or footer so it will be ignored
  warn("""Cannot parse header or footer so it will be ignored""")
```

[15]:

```
                                          Unnamed: 0      Total    Schweiz  \
0                                            Schweiz  8670300.0  6459512.0
1                                               1000     3991.0     2379.0
2                                               1003     6528.0     3555.0
3                                               1004    31084.0    17927.0
4                                               1005    12465.0     7213.0
…                                                …         …         …
3183         1 Serienbruch ab 2014: Exkl. „ohne Angabe"        NaN        NaN
3184                                 Quelle: STATPOP        NaN        NaN
3185                                           © BFS        NaN        NaN
3186                                             NaN        NaN        NaN
3187  Auskunft: Bundesamt für Statistik (BFS), Sekti…        NaN        NaN

        Ausland       Mann       Frau      0-4      5-9    10-14    15-19  \
0     2210788.0  4302599.0  4367701.0  437118.0  439685.0  429468.0  420030.0
1        1612.0     1957.0     2034.0    208.0    179.0    219.0    559.0
2        2973.0     3290.0     3238.0    265.0    187.0    190.0    206.0
3       13157.0    15075.0    16009.0   1464.0   1230.0   1164.0   1252.0
4        5252.0     6006.0     6459.0    643.0    501.0    483.0    506.0
…          …        …        …        …        …        …        …
3183        NaN        NaN        NaN      NaN      NaN      NaN      NaN
3184        NaN        NaN        NaN      NaN      NaN      NaN      NaN
3185        NaN        NaN        NaN      NaN      NaN      NaN      NaN
3186        NaN        NaN        NaN      NaN      NaN      NaN      NaN
3187        NaN        NaN        NaN      NaN      NaN      NaN      NaN

      …     80-84     85-89  90 und mehr      Ledig  Verheiratet  Verwitwet  \
0     …  227086.0  147174.0      84029.0  3903333.0    3588894.0   403471.0
1     …      50.0      27.0         14.0     2378.0       1307.0       81.0
2     …      85.0      55.0         56.0     4101.0       1628.0      178.0
3     …     751.0     533.0        363.0    17350.0       9284.0     1261.0
4     …     243.0     206.0        119.0     7395.0       3496.0      397.0
…   …        …         …          …          …           …          …
3183  …       NaN       NaN          NaN        NaN          NaN        NaN
3184  …       NaN       NaN          NaN        NaN          NaN        NaN
3185  …       NaN       NaN          NaN        NaN          NaN        NaN
3186  …       NaN       NaN          NaN        NaN          NaN        NaN
3187  …       NaN       NaN          NaN        NaN          NaN        NaN

      Geschieden  Unverheiratet  In eingetrage-ner Partner-schaft  \
0       751735.0          617.0                           19022.0
1          217.0            0.0                               7.0
2          556.0            1.0                              59.0
3         3028.0            7.0                             127.0
4         1109.0            2.0                              53.0
…           …             …                              …
3183         NaN            NaN                               NaN
```

```
3184         NaN            NaN                                 NaN
3185         NaN            NaN                                 NaN
3186         NaN            NaN                                 NaN
3187         NaN            NaN                                 NaN

      Aufgelöste Partnerschaft
0                       2981.0
1                          1.0
2                          5.0
3                         25.0
4                         12.0
…                       …
3183                       NaN
3184                       NaN
3185                       NaN
3186                       NaN
3187                       NaN

[3188 rows x 32 columns]
```

[16]:
```python
population_2 = pd.read_csv('../Data/0_Raw/STATPOP2020_GMDE.csv', sep=";")
population_2
```

[16]:
```
      GDENR  B20BTOT  B20B11  B20B12  B20B13  B20B14  B20B15  B20B16  B20B21  \
0         1     2014    1724     290     218      42      30       0    1565
1         2    12289    8725    3564    2083     947     533       1    8135
2         3     5610    4639     971     708     109     154       0    4297
3         4     3801    3199     602     401     100     101       0    3016
4         5     3795    3136     659     390     159     110       0    2957
…     …     …       …       …       …       …       …       …       …
2193   6806      560     520      40      25       6       9       0     468
2194   6807     1241    1125     116      96       6      14       0    1018
2195   6808     1263    1170      93      84       1       8       0    1120
2196   6809     1096    1011      85      70       9       6       0     960
2197   6810     1135    1067      68      64       3       1       0     994

      B20B22  …  B20B55  B20B56  H20PTOT  H20P01  H20P02  H20P03  H20P04  \
0         13  …      12       1      877     269     324     111     132
1       2515  …     145      10     5512    1993    1881     650     685
2         17  …      63       2     2357     683     797     344     410
3         33  …      29       2     1580     461     546     219     248
4         17  …      26       1     1584     478     532     212     259
…       …  …     …       …      …       …      …       …      …
2193      20  …       5       2      248      83      96      23      29
2194      59  …       9       2      545     200     176      60      65
2195      63  …      11       1      597     241     206      61      51
2196     117  …      11       0      510     181     190      66      48
```

```
2197       65  …       13        0       506       188       156       63       64

      H20P05  H20P06  H20PI
0         32       9      2
1        233      70      2
2        107      16      1
3         79      27      1
4         73      30      2
…      …      …     …
2193      10       7      1
2194      28      16      1
2195      31       7      1
2196      18       7      1
2197      23      12      1

[2198 rows x 78 columns]
```

[17]:

```python
cars = pd.read_csv('../Data/0_Raw/cars_per_municipality.csv', sep = ";",␣
↪encoding = 'latin-1')
cars
```

[17]:

```
                 Gemeinde Fahrzeuggruppe                          Treibstoff  \
0       1 Aeugst am Albis  Personenwagen                              Benzin
1       1 Aeugst am Albis  Personenwagen                              Diesel
2       1 Aeugst am Albis  Personenwagen   Benzin-elektrisch: Normal-Hybrid
3       1 Aeugst am Albis  Personenwagen  Benzin-elektrisch: Plug-in-Hybrid
4       1 Aeugst am Albis  Personenwagen   Diesel-elektrisch: Normal-Hybrid
…                     …            …                                 …
151405    6810 La Baroche       Anhänger  Diesel-elektrisch: Plug-in-Hybrid
151406    6810 La Baroche       Anhänger                          Elektrisch
151407    6810 La Baroche       Anhänger                         Wasserstoff
151408    6810 La Baroche       Anhänger           Gas (mono- und bivalent)
151409    6810 La Baroche       Anhänger                             Anderer

        2015  2016  2017  2018  2019  2020  2021
0        845   822   815   816   809   804   792
1        288   306   316   318   326   329   320
2         13    18    16    20    22    30    43
3          0     1     2     7     7    12    20
4          0     0     2     2     3     2     5
…      …    …    …    …    …    …    …
151405     0     0     0     0     0     0     0
151406     0     0     0     0     0     0     0
151407     0     0     0     0     0     0     0
151408     0     0     0     0     0     0     0
151409   180   190   202   202   209   216   227
```

```
[151410 rows x 10 columns]
```

[18]:
```python
stations = pd.read_excel('../Data/0_Raw/stations_list_bav.xlsx')
stations
```

[18]:
```
                                Dst-Nr85          Ld  \
0                               N° sv.85          py
1      Dienststellen-\nNummer siebenstellig  Ländercode
2                                    NaN         NaN
3                                8506013          85
4                                8573363          85
…                                      …           …
49998                                NaN         NaN
49999                                NaN         NaN
50000                                NaN         NaN
50001                                NaN         NaN
50002                                NaN         NaN

                          Dst-Nr                     KZ  \
0                         N° sv.                     Cc
1      Dienststellen-\nNummer (85…)  Kontrollziffer (o.Ld)
2                            NaN                    NaN
3                           6013                      7
4                          73363                      4
…                              …                      …
49998                        NaN                    NaN
49999                        NaN                    NaN
50000                        NaN                    NaN
50001                        NaN                    NaN
50002                        NaN                    NaN

                                              Name          Länge  \
0                              Nom (ordre alphab.)       Longueur
1                          Name \n(Dst-Bezeichnung)   Länge (Name)
2      Datenstand am 24.02.2022, Auszug für 24.02.2022           NaN
3                                           Aadorf              6
4                                  Aadorf, Bahnhof             15
…                                                …              …
49998                                          NaN            NaN
49999                                          NaN            NaN
50000                                          NaN            NaN
50001                                          NaN            NaN
50002                                          NaN            NaN

                    Name lang                  Dst-Abk  \
0                    Nom long                Sigle sv.
1      Name lang \n(50 Zeichen)  Dienststellen-\nAbkürzung
```

```
2                        NaN                        NaN
3                        NaN                         AD
4                        NaN                        NaN
…                          …                          …
49998                    NaN                        NaN
49999                    NaN                        NaN
50000                    NaN                        NaN
50001                    NaN                        NaN
50002                    NaN                        NaN

                            BP           VP  …  Ortschaft  \
0                           PE           PT  …   Localité
1      Betriebspunkt des Fahrplans  Haltestelle  …  Ortschaft
2                          NaN          NaN  …        NaN
3                            *           Ho  …     Aadorf
4                            *           Ho  …     Aadorf
…                            …            …  …          …
49998                      NaN          NaN  …        NaN
49999                      NaN          NaN  …        NaN
50000                      NaN          NaN  …        NaN
50001                      NaN          NaN  …        NaN
50002                      NaN          NaN  …        NaN

                     Gde-Nr  Gemeinde     Kt.      E-Koord.      N-Koord.  \
0                N° commune   Commune     Ct.      Coord. E      Coord. N
1      Gemeinde-\nNummer BFS  Gemeinde  Kanton  E-Koordinate  N-Koordinate
2                       NaN       NaN     NaN           NaN           NaN
3                      4551    Aadorf      TG       2710378       1260736
4                      4551    Aadorf      TG       2710335       1260768
…                         …         …       …             …             …
49998                   NaN       NaN     NaN           NaN           NaN
49999                   NaN       NaN     NaN           NaN           NaN
50000                   NaN       NaN     NaN           NaN           NaN
50001                   NaN       NaN     NaN           NaN           NaN
50002                   NaN       NaN     NaN           NaN           NaN

               Höhe  Bemerkungen                          Karte  \
0          Altitude     Remarque                          Carte
1      Höhe m ü.M.  Bemerkungen  Hyperlink auf \nmapsearch.ch
2               NaN          NaN
3               528          NaN
4               528          NaN
…                 …            …                              …
49998           NaN          NaN
49999           NaN          NaN
50000           NaN          NaN
50001           NaN          NaN
```

```
50002          NaN          NaN

                               Karte.1
0                                Carte
1       Hyperlink auf \nmap.geo.admin.ch
2
3
4
…                                  …
49998
49999
50000
50001
50002

[50003 rows x 29 columns]
```

[19]:
```
stop_count = pd.read_csv('../Data/0_Raw/stop_count.csv', sep = ",",␣
↪encoding="latin-1")
stop_count
```

```
/usr/local/lib/python3.8/dist-packages/IPython/core/interactiveshell.py:3326:
DtypeWarning: Columns (7,14,16) have mixed types.Specify dtype option on import
or set low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

[19]:
```
         FP_ID  TU_CODE        TU_BEZEICHNUNG TU_ABKUERZUNG  FARTNUMMER  \
0         2022      101  Verkehrsbetriebe Biel         VB-be       23000
1         2022      101  Verkehrsbetriebe Biel         VB-be       23000
2         2022      101  Verkehrsbetriebe Biel         VB-be       23000
3         2022      101  Verkehrsbetriebe Biel         VB-be       23001
4         2022      101  Verkehrsbetriebe Biel         VB-be       23001
…         …      …                       …          …        …
4584247   2022     9999          Diverse INFO       DIVINFO         906
4584248   2022     9999          Diverse INFO       DIVINFO         906
4584249   2022     9999          Diverse INFO       DIVINFO         906
4584250   2022     9999          Diverse INFO       DIVINFO         906
4584251   2022     9999          Diverse INFO       DIVINFO         906

           BPUIC                 BP_BEZEICHNUNG BP_ABKUERZUNG KANTON  \
0        8504351           Biel/Bienne Beaumont           NaN     BE
1        8504350  Biel/Bienne Leubringenb.(Funi)           NaN     BE
2        8504352             Evilard/Leubringen           NaN     BE
3        8504351           Biel/Bienne Beaumont           NaN     BE
4        8504350  Biel/Bienne Leubringenb.(Funi)           NaN     BE
…          …                          …            …     …
4584247  8509195                        Filisur          FILI     GR
```

```
4584248  8509251                         Samedan          SAME     GR
4584249  8509253                      St. Moritz          SMOR     GR
4584250  8509189                          Thusis           THS     GR
4584251  8509192                    Tiefencastel          TICA     GR

                    SLOID VM_ART  FAHRTAGE           AB_ZEIT_KB  \
0        ch:1:sloid:4351    FUN       359  01.01.1970 05:58:00
1        ch:1:sloid:4350    FUN       359  01.01.1970 05:55:00
2        ch:1:sloid:4352    FUN       359                  NaN
3        ch:1:sloid:4351    FUN       359  01.01.1970 05:58:00
4        ch:1:sloid:4350    FUN       359                  NaN
…                     …    …        …                   …
4584247  ch:1:sloid:9195     PE       163  01.01.1970 20:01:00
4584248  ch:1:sloid:9251     PE       163  01.01.1970 20:49:00
4584249  ch:1:sloid:9253     PE       163                  NaN
4584250  ch:1:sloid:9189     PE       163  01.01.1970 19:29:00
4584251  ch:1:sloid:9192     PE       163  01.01.1970 19:46:00

                  AN_ZEIT_KB RICHTUNG_TEXT_AGGREGIERT  \
0        01.01.1970 05:58:00                      NaN
1                        NaN                      NaN
2        01.01.1970 06:02:00                      NaN
3        01.01.1970 05:58:00                      NaN
4        01.01.1970 06:02:00                      NaN
…                        …                      …
4584247  01.01.1970 20:00:00                      NaN
4584248  01.01.1970 20:45:00                      NaN
4584249  01.01.1970 21:00:00                      NaN
4584250  01.01.1970 19:27:00                      NaN
4584251  01.01.1970 19:44:00                      NaN

                      END_BP_BEZEICHNUNG LINIE   BP_ID
0                    Evilard/Leubringen  23.0  138747
1                    Evilard/Leubringen  23.0  123038
2                    Evilard/Leubringen  23.0  163638
3        Biel/Bienne Leubringenb.(Funi)  23.0  138747
4        Biel/Bienne Leubringenb.(Funi)  23.0  123038
…                                   …    …     …
4584247                      St. Moritz   NaN  119134
4584248                      St. Moritz   NaN  119158
4584249                      St. Moritz   NaN  119160
4584250                      St. Moritz   NaN  119128
4584251                      St. Moritz   NaN  119131

[4584252 rows x 18 columns]
```

[20]:
```
town_directory = pd.read_csv('../Data/0_Raw/town_directory.csv')
town_directory
```

[20]:
```
      Ortschaftsname   PLZ  Zusatzziffer             Gemeindename  BFS-Nr  \
0        Lausanne 25  1000            25                 Lausanne    5586
1        Lausanne 26  1000            26                 Lausanne    5586
2        Lausanne 27  1000            27                 Lausanne    5586
3           Lausanne  1003             0                 Lausanne    5586
4           Lausanne  1004             0                 Lausanne    5586
…                ...   …           …                       …      …
4123     Unterwasser  9657             0  Wildhaus-Alt St. Johann    3359
4124        Wildhaus  9658             0  Wildhaus-Alt St. Johann    3359
4125       Thunersee  9999             1                Thunersee    9073
4126     Brienzersee  9999             2              Brienzersee    9089
4127       Bielersee  9999             0           Bielersee (BE)    9149

     Kantonskürzel            E            N Sprache
0               VD  542094.8938  157051.9666      fr
1               VD  543068.1153  156403.0412      fr
2               VD  541921.1403  154775.3096      fr
3               VD  537956.7751  152398.2869      fr
4               VD  537089.8121  153349.5648      fr
…              …          …           …      …
4123            SG  741690.2129  229037.4686      de
4124            SG  744861.3314  229854.4341      de
4125            BE  621181.5226  170794.5768      de
4126            BE  640930.6820  175395.8963      de
4127            BE  580261.9545  215168.9479      de

[4128 rows x 9 columns]
```

[21]:
```
inbound_comm = pd.read_excel('../Data/0_Raw/inbound_comm.xlsx')
inbound_comm
```

[21]:
```
                                      Zupendlerquote 2000   Unnamed: 1  \
0                                                     NaN          NaN
1                                                     NaN          NaN
2                                              Regions-ID  Regionsname
3                                                     NaN          NaN
4                                                     NaN      Schweiz
…                                                       …            …
2907  11 - Mobilität, Verkehr > Pendlermobilität >  …              NaN
2908           Schweiz / Politische Gemeinden / 5.12.2000          NaN
2909                                                  NaN          NaN
2910                       Kontakt: statatlas@bfs.admin.ch          NaN
2911  © Bundesamt für Statistik, ThemaKart, Neuchâte…             NaN
```

```
                                                   3561
0                                                   NaN
1      Anteil der zupendelnden Erwerbstätigen an den …
2                                                   NaN
3                                                   NaN
4                                             58.882161
…                                                 …
2907                                                NaN
2908                                                NaN
2909                                                NaN
2910                                                NaN
2911                                                NaN

[2912 rows x 3 columns]
```

[22]:
```python
outbound_comm = pd.read_excel('../Data/0_Raw/outbound_comm.xlsx')
outbound_comm
```

[22]:
```
                                   Wegpendlerquote 2000   Unnamed: 1  \
0                                                   NaN          NaN
1                                                   NaN          NaN
2                                            Regions-ID  Regionsname
3                                                   NaN          NaN
4                                                   NaN      Schweiz
…                                                 …            …
2907  11 - Mobilität, Verkehr > Pendlermobilität >  …          NaN
2908         Schweiz / Politische Gemeinden / 5.12.2000          NaN
2909                                                NaN          NaN
2910                     Kontakt: statatlas@bfs.admin.ch          NaN
2911  © Bundesamt für Statistik, ThemaKart, Neuchâte…          NaN

                                                   3581
0                                                   NaN
1      Anteil der wegpendelnden Erwerbstätigen an den…
2                                                   NaN
3                                                   NaN
4                                             57.259905
…                                                 …
2907                                                NaN
2908                                                NaN
2909                                                NaN
2910                                                NaN
2911                                                NaN

[2912 rows x 3 columns]
```

[23]:
```
dist_st = pd.read_table("../Data/0_Raw/DWV_2017_Strasse_Distanz_CH_2337.mtx",␣
↪encoding="latin-1")
dist_st
```

[23]:
```
               $O;D3
0          * Von Bis
1          0000 0000
2           * Faktor
3               1.00
4                 *
…                …
5463910    7009 ""
5463911    7010 ""
5463912    7011 ""
5463913    7101 ""
5463914    7301 ""

[5463915 rows x 1 columns]
```

[24]:
```
time_st = pd.read_table("../Data/0_Raw/DWV_2017_Strasse_Reisezeit_CH_2337.mtx",␣
↪encoding="latin-1")
time_st
```

[24]:
```
               $O;D3
0          * Von Bis
1          0000 0000
2           * Faktor
3               1.00
4                 *
…                …
5463910    7009 ""
5463911    7010 ""
5463912    7011 ""
5463913    7101 ""
5463914    7301 ""

[5463915 rows x 1 columns]
```

[25]:
```
dist_pt = pd.read_table("../Data/0_Raw/DWV_2017_ÖV_Distanz_CH_2337.mtx",␣
↪encoding="latin-1")
dist_pt
```

[25]:
```
               $O;D3
0          * Von Bis
1          0000 0000
2           * Faktor
3               1.00
```

```
4                *
…              …
5463910    7009 ""
5463911    7010 ""
5463912    7011 ""
5463913    7101 ""
5463914    7301 ""

[5463915 rows x 1 columns]
```

[26]:
```
time_pt = pd.read_table("../Data/0_Raw/DWV_2017_ÖV_Reisezeit_CH_2337.mtx",␣
↪encoding="latin-1")
time_pt
```

[26]:
```
              $O;D3
0         * Von Bis
1         0000 0000
2          * Faktor
3              1.00
4                *
…              …
5463910    7009 ""
5463911    7010 ""
5463912    7011 ""
5463913    7101 ""
5463914    7301 ""

[5463915 rows x 1 columns]
```

# 5 Cleaning Data

In this section, all data is cleaned to reach proper data without noise and unnecessary columns.

## 5.1 Distance + time matrices

### 5.1.1 Street distance

[33]:
```
dist_st.iloc[0:8]
```

[33]:
```
                                            $O;D3
0                                       * Von Bis
1                                       0000 0000
2                                        * Faktor
3                                            1.00
```

```
4                                          *
5  * Bundesamt für Raumentwicklung ARE Ittigen
6                                   * 18.03.20
7                          1          1  5.326
```

delete the leading 7 header rows

[34]:
```
dist_st.drop(dist_st.index[0:7], inplace=True)
dist_st
```

[34]:
```
                                 $O;D3
7                  1          1  5.326
8                  1          2  5.948
9                  1          3  9.613
10                 1          4  8.669
11                 1          5  8.191
…                                 …
5463910                        7009 ""
5463911                        7010 ""
5463912                        7011 ""
5463913                        7101 ""
5463914                        7301 ""

[5463908 rows x 1 columns]
```

Split the second column, which has the information "from", "to" and "distance" in it!

[35]:
```
dist_st = dist_st["$O;D3"].str.split(expand=True)
```

there is still one undesirable row left at the end:

[36]:
```
dist_st.loc[dist_st[1] == "Netzobjektnamen"]
```

[36]:
```
         0                1     2
5461576  *  Netzobjektnamen  None
```

delete this!

[37]:
```
dist_st = dist_st.iloc[:5461569, : ]
```

[38]:
```
dist_st.rename(columns = {0: "from", 1: "to", 2: "dist_street"}, inplace = True)
```

[39]:
```
dist_st
```

[39]:
```
         from    to dist_street
7           1     1       5.326
8           1     2       5.948
9           1     3       9.613
10          1     4       8.669
```

```
11          1     5        8.191
…         …     …          …
5461571  7301  7009      200.735
5461572  7301  7010      204.878
5461573  7301  7011      205.223
5461574  7301  7101      278.181
5461575  7301  7301        2.754

[5461569 rows x 3 columns]
```

### 5.1.2 Street time

Same approach as in 5.1.1

[40]:
```python
time_st.iloc[0:8]
```

[40]:
```
                                         $0;D3
0                                     * Von Bis
1                                     0000 0000
2                                      * Faktor
3                                          1.00
4                                             *
5  * Bundesamt für Raumentwicklung ARE Ittigen
6                                    * 18.03.20
7                          1          1 14.342
```

delete the leading 6 header rows

[41]:
```python
time_st.drop(time_st.index[0:7], inplace=True)
time_st
```

[41]:
```
                                 $0;D3
7                 1          1 14.342
8                 1          2 15.830
9                 1          3 20.440
10                1          4 20.096
11                1          5 20.371
…                                 …
5463910                        7009 ""
5463911                        7010 ""
5463912                        7011 ""
5463913                        7101 ""
5463914                        7301 ""

[5463908 rows x 1 columns]
```

[42]:
```
time_st = time_st["$O;D3"].str.split(expand=True)
time_st
```

[42]:
```
             0   1       2
7            1   1  14.342
8            1   2  15.830
9            1   3  20.440
10           1   4  20.096
11           1   5  20.371
…          …   ..     …
5463910  7009  ""    None
5463911  7010  ""    None
5463912  7011  ""    None
5463913  7101  ""    None
5463914  7301  ""    None

[5463908 rows x 3 columns]
```

[43]:
```
time_st.loc[time_st[1] == "Netzobjektnamen"]
```

[43]:
```
         0                1     2
5461576  *  Netzobjektnamen  None
```

[44]:
```
time_st = time_st.iloc[:5461569, : ]
```

[45]:
```
time_st.rename(columns = {0: "from", 1: "to", 2: "time_street"}, inplace = True)
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:5039:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

[46]:
```
time_st
```

[46]:
```
         from    to time_street
7           1     1      14.342
8           1     2      15.830
9           1     3      20.440
10          1     4      20.096
11          1     5      20.371
…         …    …          …
5461571  7301  7009     133.474
5461572  7301  7010     135.730
5461573  7301  7011     140.941
```

```
5461574  7301  7101     218.801
5461575  7301  7301       7.805

[5461569 rows x 3 columns]
```

### 5.1.3 public transport time

Same approach as in 5.1.1

[47]:
```
time_pt.iloc[0:8]
```

[47]:
```
                                          $0;D3
0                                      * Von Bis
1                                      0000 0000
2                                       * Faktor
3                                           1.00
4                                             *
5  * Bundesamt für Raumentwicklung ARE Ittigen
6                                     * 18.03.20
7                          1          1 20.483
```

delete the leading 6 header rows

[48]:
```
time_pt.drop(time_pt.index[0:7], inplace=True)
time_pt
```

[48]:
```
                                  $0;D3
7                  1          1 20.483
8                  1          2 24.290
9                  1          3 42.945
10                 1          4 36.187
11                 1          5 37.729
…                                   …
5463910                      7009 ""
5463911                      7010 ""
5463912                      7011 ""
5463913                      7101 ""
5463914                      7301 ""

[5463908 rows x 1 columns]
```

[49]:
```
time_pt = time_pt["$0;D3"].str.split(expand=True)
time_pt
```

[49]:
```
            0   1      2
7           1   1   20.483
8           1   2   24.290
```

```
9           1   3  42.945
10          1   4  36.187
11          1   5  37.729
…         …  ..     …
5463910  7009  ""    None
5463911  7010  ""    None
5463912  7011  ""    None
5463913  7101  ""    None
5463914  7301  ""    None

[5463908 rows x 3 columns]
```

[50]:
```
time_pt.loc[time_pt[1] == "Netzobjektnamen"]
```

[50]:
```
         0                1     2
5461576  *  Netzobjektnamen  None
```

[51]:
```
time_pt = time_pt.iloc[:5461569, : ]
```

[52]:
```
time_pt.rename(columns = {0: "from", 1: "to", 2: "time_pt"}, inplace = True)
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:5039:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

[53]:
```
time_pt
```

[53]:
```
         from    to  time_pt
7           1     1   20.483
8           1     2   24.290
9           1     3   42.945
10          1     4   36.187
11          1     5   37.729
…         …    …       …
5461571  7301  7009  302.505
5461572  7301  7010  310.881
5461573  7301  7011  319.149
5461574  7301  7101  275.675
5461575  7301  7301   14.917

[5461569 rows x 3 columns]
```

### 5.1.4 public transport distance

Same approach as in 5.1.1

[54]:
```
dist_pt.iloc[0:8]
```

[54]:
```
                                              $0;D3
0                                         * Von Bis
1                                         0000 0000
2                                          * Faktor
3                                              1.00
4                                                 *
5  * Bundesamt für Raumentwicklung ARE Ittigen
6                                        * 18.03.20
7                          1           1   4.183
```

delete the leading 6 header rows

[55]:
```
dist_pt.drop(dist_pt.index[0:7], inplace=True)
dist_pt
```

[55]:
```
                                      $0;D3
7                  1           1   4.183
8                  1           2   6.062
9                  1           3  11.986
10                 1           4   9.970
11                 1           5   8.778
…                                      …
5463910                        7009 ""
5463911                        7010 ""
5463912                        7011 ""
5463913                        7101 ""
5463914                        7301 ""

[5463908 rows x 1 columns]
```

[56]:
```
dist_pt = dist_pt["$0;D3"].str.split(expand=True)
dist_pt
```

[56]:
```
             0   1       2
7            1   1   4.183
8            1   2   6.062
9            1   3  11.986
10           1   4   9.970
11           1   5   8.778
…         …   ..      …
5463910   7009  ""    None
5463911   7010  ""    None
```

```
5463912  7011  ""    None
5463913  7101  ""    None
5463914  7301  ""    None

[5463908 rows x 3 columns]
```

[57]:
```
dist_pt.loc[dist_pt[1] == "Netzobjektnamen"]
```

[57]:
```
         0                1     2
5461576  *  Netzobjektnamen  None
```

[58]:
```
dist_pt = dist_pt.iloc[:5461569, : ]
```

[59]:
```
dist_pt.rename(columns = {0: "from", 1: "to", 2: "dist_pt"}, inplace = True)
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/frame.py:5039:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  return super().rename(
```

[60]:
```
dist_pt
```

[60]:
```
         from    to  dist_pt
7           1     1    4.183
8           1     2    6.062
9           1     3   11.986
10          1     4    9.970
11          1     5    8.778
…         …   …       …
5461571  7301  7009  257.160
5461572  7301  7010  253.233
5461573  7301  7011  256.255
5461574  7301  7101  242.097
5461575  7301  7301    0.229

[5461569 rows x 3 columns]
```

[60]:
```

```

### 5.1.5 Joining distance tables

Now, all 4 tables should be joined together here

[61]:
```
dist = dist_st
```

[62]:
```
dist_st["dist_pt"] = dist_pt["dist_pt"]
```

[63]:
```
dist_st["time_st"] = time_st["time_street"]
```

[64]:
```
dist_st["time_pt"] = time_pt["time_pt"]
```

[65]:
```
dist
```

[65]:
```
         from    to dist_street  dist_pt  time_st  time_pt
7           1     1       5.326    4.183   14.342   20.483
8           1     2       5.948    6.062   15.830   24.290
9           1     3       9.613   11.986   20.440   42.945
10          1     4       8.669    9.970   20.096   36.187
11          1     5       8.191    8.778   20.371   37.729
…         …     …         …        …        …        …
5461571  7301  7009     200.735  257.160  133.474  302.505
5461572  7301  7010     204.878  253.233  135.730  310.881
5461573  7301  7011     205.223  256.255  140.941  319.149
5461574  7301  7101     278.181  242.097  218.801  275.675
5461575  7301  7301       2.754    0.229    7.805   14.917

[5461569 rows x 6 columns]
```

### 5.1.6 Write distance csv

[66]:
```
dist.to_csv("../Data/1_Cleaned/distances.csv", index=False)
```

## 5.2 Stations + stops data

### 5.2.1 Stop count data

[68]:
```
stop_count[:2]
```

[68]:
```
   FP_ID  TU_CODE         TU_BEZEICHNUNG TU_ABKUERZUNG  FARTNUMMER    BPUIC  \
0   2022      101   Verkehrsbetriebe Biel         VB-be       23000  8504351
1   2022      101   Verkehrsbetriebe Biel         VB-be       23000  8504350

                  BP_BEZEICHNUNG BP_ABKUERZUNG KANTON            SLOID  \
0           Biel/Bienne Beaumont           NaN     BE  ch:1:sloid:4351
1  Biel/Bienne Leubringenb.(Funi)          NaN     BE  ch:1:sloid:4350

  VM_ART  FAHRTAGE          AB_ZEIT_KB          AN_ZEIT_KB  \
```

```
0     FUN        359   01.01.1970 05:58:00  01.01.1970 05:58:00
1     FUN        359   01.01.1970 05:55:00                  NaN

  RICHTUNG_TEXT_AGGREGIERT  END_BP_BEZEICHNUNG LINIE   BP_ID
0                      NaN  Evilard/Leubringen  23.0  138747
1                      NaN  Evilard/Leubringen  23.0  123038
```

This table looks pretty good. Some columns will not be needed afterwards:

**Deleting unneccessary columns** There are different ID's here. To specify one primary key, we only need the combination of "ride ID" and "stop ID". The combination of both occurs only once in a table. The SLOID and BP ID can be ignored and therefore deleted. Further, we don't need the "BP_ABKUERZUNG" and the field "RICHTUNG_TEXT_AGGREGIERT" is somehow not very useful.

These 4 attributes can therefore be deleted in the next step.

[69]:
```
stop_count.drop(["BP_ABKUERZUNG", "SLOID", "BP_ID",␣
↪"RICHTUNG_TEXT_AGGREGIERT"], axis=1, inplace=True)
```

[70]:
```
stop_count[0:2]
```

[70]:
```
   FP_ID  TU_CODE        TU_BEZEICHNUNG TU_ABKUERZUNG  FARTNUMMER    BPUIC  \
0   2022      101  Verkehrsbetriebe Biel         VB-be       23000  8504351
1   2022      101  Verkehrsbetriebe Biel         VB-be       23000  8504350

                BP_BEZEICHNUNG KANTON VM_ART  FAHRTAGE  \
0         Biel/Bienne Beaumont     BE    FUN       359
1  Biel/Bienne Leubringenb.(Funi)     BE    FUN       359

            AB_ZEIT_KB           AN_ZEIT_KB  END_BP_BEZEICHNUNG LINIE
0  01.01.1970 05:58:00  01.01.1970 05:58:00  Evilard/Leubringen  23.0
1  01.01.1970 05:55:00                  NaN  Evilard/Leubringen  23.0
```

**Minimizing to relevant attributes and renaming columns** At the end only a reduced table, containing the attributes "FARTNUMMER", "BPUIC" and "FAHRTAGE" is needed. The "FARTNUMMER" reflects the ID of the ride, the "BPUIC" stands for the stop ID and the "FAHRTAGE" shows the number of days in a year, when this stop occurs.

To make it more understandable, I will rename these 3 columns into "ride ID", "station ID" and "nr_days". The other columns can be deleted here.

[71]:
```
stop_count.rename(columns={"FARTNUMMER":"ride_id", "BPUIC":"stop_id",␣
↪"FAHRTAGE":"nr_days"}, inplace=True)
```

[72]:
```
stop_count_reduced = stop_count[["ride_id", "stop_id", "nr_days"]]
stop_count_reduced
```

[72]:
```
         ride_id  stop_id  nr_days
0          23000  8504351      359
1          23000  8504350      359
2          23000  8504352      359
3          23001  8504351      359
4          23001  8504350      359
…            …        …      …
4584247      906  8509195      163
4584248      906  8509251      163
4584249      906  8509253      163
4584250      906  8509189      163
4584251      906  8509192      163

[4584252 rows x 3 columns]
```

Now the table seems to be ok and can be written into a csv.

**Writing stop_count csv** Write table now to google drive.

[73]:
```python
stop_count_reduced.to_csv("../Data/1_Cleaned/stop_count.csv", index=False)
```

### 5.2.2 Public stations list

[74]:
```python
stations.columns
```

[74]:
```
Index(['Dst-Nr85', 'Ld', 'Dst-Nr', 'KZ', 'Name', 'Länge', 'Name lang',
       'Dst-Abk', 'BP', 'VP', 'VG', 'RB', 'TH', 'Status', 'Verkehrsmittel',
       'TU-Nr', 'TU-Abk', 'GO-Nr', 'GO-Abk', 'Ortschaft', 'Gde-Nr', 'Gemeinde',
       'Kt.', 'E-Koord.', 'N-Koord.', 'Höhe', 'Bemerkungen', 'Karte',
       'Karte.1'],
      dtype='object')
```

[75]:
```python
stations[:7]
```

[75]:
```
                              Dst-Nr85          Ld  \
0                             N° sv.85          py
1  Dienststellen-\nNummer siebenstellig  Ländercode
2                                  NaN         NaN
3                              8506013          85
4                              8573363          85
5                              8576958          85
6                              8506853          85

                        Dst-Nr                    KZ  \
0                       N° sv.                    Cc
1  Dienststellen-\nNummer (85…)  Kontrollziffer (o.Ld)
```

```
2                             NaN                    NaN
3                            6013                      7
4                           73363                      4
5                           76958                      8
6                            6853                      6

                                       Name         Länge  \
0                         Nom (ordre alphab.)      Longueur
1                    Name \n(Dst-Bezeichnung)  Länge (Name)
2  Datenstand am 24.02.2022, Auszug für 24.02.2022          NaN
3                                      Aadorf             6
4                             Aadorf, Bahnhof            15
5                      Aadorf, Matthofstrasse            22
6                           Aadorf, Morgental            17

              Name lang                  Dst-Abk  \
0              Nom long                Sigle sv.
1  Name lang \n(50 Zeichen)  Dienststellen-\nAbkürzung
2                   NaN                      NaN
3                   NaN                       AD
4                   NaN                      NaN
5                   NaN                      NaN
6                   NaN                      NaN

                        BP           VP  …  Ortschaft  \
0                       PE           PT  …   Localité
1  Betriebspunkt des Fahrplans  Haltestelle  …  Ortschaft
2                      NaN          NaN  …        NaN
3                        *           Ho  …     Aadorf
4                        *           Ho  …     Aadorf
5                        *           Ho  …     Aadorf
6                        *           Ho  …     Aadorf

              Gde-Nr  Gemeinde     Kt.      E-Koord.      N-Koord.  \
0           N° commune   Commune     Ct.      Coord. E      Coord. N
1  Gemeinde-\nNummer BFS  Gemeinde  Kanton  E-Koordinate  N-Koordinate
2                  NaN       NaN     NaN           NaN           NaN
3                 4551    Aadorf      TG       2710378       1260736
4                 4551    Aadorf      TG       2710335       1260768
5                 4551    Aadorf      TG       2710483       1260407
6                 4551    Aadorf      TG       2709827       1261373

         Höhe  Bemerkungen                     Karte  \
0    Altitude      Remarque                     Carte
1  Höhe m ü.M.  Bemerkungen  Hyperlink auf \nmapsearch.ch
2         NaN          NaN
3         528          NaN
```

```
4          528          NaN
5          531          NaN
6          517          NaN

                           Karte.1
0                            Carte
1  Hyperlink auf \nmap.geo.admin.ch
2
3
4
5
6

[7 rows x 29 columns]
```

**Remove header** The first three rows are not usable, therefore I can delete them:

[76]:
```
stations.drop([0, 1, 2], axis=0, inplace=True)
```

[77]:
```
stations[:2]
```

[77]:
```
  Dst-Nr85  Ld Dst-Nr KZ             Name Länge Name lang Dst-Abk BP  VP  …  \
3  8506013  85   6013  7           Aadorf     6       NaN      AD  *  Ho  …
4  8573363  85  73363  4  Aadorf, Bahnhof    15       NaN     NaN  *  Ho  …

  Ortschaft Gde-Nr Gemeinde Kt. E-Koord. N-Koord. Höhe Bemerkungen Karte  \
3    Aadorf   4551   Aadorf  TG  2710378  1260736  528         NaN
4    Aadorf   4551   Aadorf  TG  2710335  1260768  528         NaN

  Karte.1
3
4

[2 rows x 29 columns]
```

[78]:
```
stations[-5:]
```

[78]:
```
      Dst-Nr85   Ld Dst-Nr   KZ Name Länge Name lang Dst-Abk   BP   VP  …  \
49998      NaN  NaN    NaN  NaN  NaN   NaN       NaN     NaN  NaN  NaN  …
49999      NaN  NaN    NaN  NaN  NaN   NaN       NaN     NaN  NaN  NaN  …
50000      NaN  NaN    NaN  NaN  NaN   NaN       NaN     NaN  NaN  NaN  …
50001      NaN  NaN    NaN  NaN  NaN   NaN       NaN     NaN  NaN  NaN  …
50002      NaN  NaN    NaN  NaN  NaN   NaN       NaN     NaN  NaN  NaN  …

      Ortschaft Gde-Nr Gemeinde  Kt. E-Koord. N-Koord. Höhe Bemerkungen Karte  \
49998       NaN    NaN      NaN  NaN      NaN      NaN  NaN         NaN
```

```
49999        NaN     NaN       NaN   NaN       NaN       NaN   NaN          NaN
50000        NaN     NaN       NaN   NaN       NaN       NaN   NaN          NaN
50001        NaN     NaN       NaN   NaN       NaN       NaN   NaN          NaN
50002        NaN     NaN       NaN   NaN       NaN       NaN   NaN          NaN

      Karte.1
49998
49999
50000
50001
50002

[5 rows x 29 columns]
```

**Remove undesired columns and NA rows** Many rows seem to have "NA" values and the last two columns are not usable. Lets delete first the two columns and afterwards the rows with only NaN:

[79]:
```
stations.drop(["Karte", "Karte.1"], axis=1, inplace=True)
```

[80]:
```
stations.dropna(axis=0, how='all', inplace=True) # drop rows with all NA
```

[81]:
```
len(stations) # number of rows!
```

[81]:
```
28388
```

Now more than 20000 rows have been deleted which is good!

[82]:
```
stations.describe()
```

[82]:
```
        Dst-Nr85      Ld  Dst-Nr     KZ     Name  Länge              Name lang  \
count      28388   28388   28388  28388    28388  28388                    401
unique     28388       1   28388     10    28388     29                    401
top      8506013      85    6013      8   Aadorf     18  Abtwil SG, Dufourpark
freq           1   28388       1   2867        1   2072                      1

       Dst-Abk     BP     VP  …  GO-Nr GO-Abk Ortschaft  Gde-Nr Gemeinde  \
count     4303  28388  26785  …  28388  28388     28035   28014    28014
unique    4303      1      6  …    494    494      3769    2121     2123
top         AD      *     Ho  …    801    PAG    Zürich     261   Zürich
freq         1  28388  25775  …  10014  10014       560     561      561

          Kt. E-Koord.  N-Koord.   Höhe Bemerkungen
count   27834    28388     28388  28380        3670
unique     26    26631     25950   2158        1916
top        BE  2500400   1259400    435       (Zug)
freq     3865        4         6    230         811
```

```
[4 rows x 27 columns]
```

**Deleting unneccessary columns** According to the ER model, only Station ID, name & status; canton, BFS Nr. and locality; transport type & company as well as coordinates are needed. Therefore, the other columns will be deleted:

[83]:
```
stations_reduced = stations[["Dst-Nr85", "Name", "Status", "Kt.", "Gde-Nr",␣
↪"Ortschaft", "Verkehrsmittel", "TU-Abk", "E-Koord.", "N-Koord."]]
```

[84]:
```
stations_reduced
```

[84]:
```
       Dst-Nr85                            Name Status  Kt. Gde-Nr  Ortschaft  \
3       8506013                          Aadorf       3   TG   4551     Aadorf
4       8573363                 Aadorf, Bahnhof       3   TG   4551     Aadorf
5       8576958          Aadorf, Matthofstrasse       3   TG   4551     Aadorf
6       8506853               Aadorf, Morgental       3   TG   4551     Aadorf
7       8573362                 Aadorf, Zentrum       3   TG   4551     Aadorf
…         …                              …   …    …        …
28386   8591218  Zürich,Kalkbreite/Bhf.Wiedikon       3   ZH    261     Zürich
28387   8503653                 Zürichhorn (See)       3   ZH    261     Zürich
28388   8530528                           Älpli       3   GR   3954  Malans GR
28389   8518708                        Äuli (B)       3   GR   3861    Fideris
28390   8518838                      Überlingen       3  NaN    NaN        NaN

       Verkehrsmittel TU-Abk E-Koord. N-Koord.
3                 Zug    SBB  2710378  1260736
4                 Bus    PAG  2710335  1260768
5                 Bus    PAG  2710483  1260407
6                 Bus    PAG  2709827  1261373
7                 Bus    PAG  2710079  1261060
…                   …     …       …        …
28386        Bus_Tram    VBZ  2681770  1247629
28387          Schiff    ZSG  2684205  1245239
28388     Kabinenbahn    AMG  2763452  1209076
28389             NaN    RhB  2776150  1199237
28390             Zug     DB  2729242  1292368

[28388 rows x 10 columns]
```

**Writing stations csv** Write table now to google drive.

[85]:
```
stations_reduced.to_csv("../Data/1_Cleaned/stations.csv", index=False)
```

## 5.3 Population data

### 5.3.1 Population 1 list

In the first population list, we get the data about marital status.

[86]:
```
population_1[:4]
```

[86]:
```
  Unnamed: 0      Total    Schweiz    Ausland       Mann       Frau       0-4  \
0     Schweiz  8670300.0  6459512.0  2210788.0  4302599.0  4367701.0  437118.0
1        1000     3991.0     2379.0     1612.0     1957.0     2034.0     208.0
2        1003     6528.0     3555.0     2973.0     3290.0     3238.0     265.0
3        1004    31084.0    17927.0    13157.0    15075.0    16009.0    1464.0

        5-9     10-14     15-19  …     80-84     85-89  90 und mehr  \
0  439685.0  429468.0  420030.0  …  227086.0  147174.0      84029.0
1     179.0     219.0     559.0  …      50.0      27.0         14.0
2     187.0     190.0     206.0  …      85.0      55.0         56.0
3    1230.0    1164.0    1252.0  …     751.0     533.0        363.0

       Ledig  Verheiratet  Verwitwet  Geschieden  Unverheiratet  \
0  3903333.0    3588894.0   403471.0    751735.0          617.0
1     2378.0       1307.0       81.0       217.0            0.0
2     4101.0       1628.0      178.0       556.0            1.0
3    17350.0       9284.0     1261.0      3028.0            7.0

   In eingetrage-ner Partner-schaft  Aufgelöste Partnerschaft
0                           19022.0                    2981.0
1                               7.0                       1.0
2                              59.0                       5.0
3                             127.0                      25.0

[4 rows x 32 columns]
```

Sum row on top is not necessary => dropping

[87]:
```
population_1.drop(0, inplace = True)
```

[88]:
```
population_1[3180:3187]
```

[88]:
```
                                           Unnamed: 0   Total  Schweiz  \
3181                                             9657   714.0    641.0
3182                                             9658  1272.0   1101.0
3183         1 Serienbruch ab 2014: Exkl. „ohne Angabe"     NaN      NaN
3184                                     Quelle: STATPOP     NaN      NaN
3185                                               © BFS     NaN      NaN
3186                                                 NaN     NaN      NaN
3187  Auskunft: Bundesamt für Statistik (BFS), Sekti…     NaN      NaN
```

```
      Ausland   Mann   Frau   0-4   5-9  10-14  15-19  ...  80-84  85-89  \
3181     73.0  358.0  356.0  30.0  44.0   37.0   35.0  ...   19.0   10.0
3182    171.0  654.0  618.0  60.0  67.0   51.0   60.0  ...   44.0   27.0
3183      NaN    NaN    NaN   NaN   NaN    NaN    NaN  ...    NaN    NaN
3184      NaN    NaN    NaN   NaN   NaN    NaN    NaN  ...    NaN    NaN
3185      NaN    NaN    NaN   NaN   NaN    NaN    NaN  ...    NaN    NaN
3186      NaN    NaN    NaN   NaN   NaN    NaN    NaN  ...    NaN    NaN
3187      NaN    NaN    NaN   NaN   NaN    NaN    NaN  ...    NaN    NaN

      90 und mehr  Ledig  Verheiratet  Verwitwet  Geschieden  Unverheiratet  \
3181          8.0  293.0        313.0       40.0        68.0            0.0
3182         15.0  522.0        549.0       88.0       112.0            0.0
3183          NaN    NaN          NaN        NaN         NaN            NaN
3184          NaN    NaN          NaN        NaN         NaN            NaN
3185          NaN    NaN          NaN        NaN         NaN            NaN
3186          NaN    NaN          NaN        NaN         NaN            NaN
3187          NaN    NaN          NaN        NaN         NaN            NaN

      In eingetrage-ner Partner-schaft  Aufgelöste Partnerschaft
3181                               0.0                       0.0
3182                               1.0                       0.0
3183                               NaN                       NaN
3184                               NaN                       NaN
3185                               NaN                       NaN
3186                               NaN                       NaN
3187                               NaN                       NaN

[7 rows x 32 columns]
```

Last 5 rows are of no value => dropping

[89]:
```python
population_1.drop(population_1.tail(5).index, inplace = True) # deleting last 5
 ↪rows
population_1
```

[89]:
```
     Unnamed: 0    Total  Schweiz  Ausland     Mann     Frau     0-4     5-9  \
1          1000   3991.0   2379.0   1612.0   1957.0   2034.0   208.0   179.0
2          1003   6528.0   3555.0   2973.0   3290.0   3238.0   265.0   187.0
3          1004  31084.0  17927.0  13157.0  15075.0  16009.0  1464.0  1230.0
4          1005  12465.0   7213.0   5252.0   6006.0   6459.0   643.0   501.0
5          1006  15520.0   9390.0   6130.0   7409.0   8111.0   816.0   664.0
...         ...      ...      ...      ...      ...      ...     ...     ...
3178       9652    699.0    613.0     86.0    349.0    350.0    34.0    21.0
3179       9655    342.0    325.0     17.0    176.0    166.0    17.0    30.0
3180       9656    638.0    553.0     85.0    325.0    313.0    36.0    47.0
3181       9657    714.0    641.0     73.0    358.0    356.0    30.0    44.0
```

```
3182       9658   1272.0   1101.0    171.0    654.0    618.0    60.0    67.0

       10-14   15-19  …  80-84  85-89  90 und mehr    Ledig  Verheiratet  \
1      219.0   559.0  …   50.0   27.0         14.0   2378.0       1307.0
2      190.0   206.0  …   85.0   55.0         56.0   4101.0       1628.0
3     1164.0  1252.0  …  751.0  533.0        363.0  17350.0       9284.0
4      483.0   506.0  …  243.0  206.0        119.0   7395.0       3496.0
5      646.0   607.0  …  353.0  279.0        203.0   8723.0       4642.0
…        …      …  …    …     …              …       …            …
3178    38.0    29.0  …   24.0    9.0          4.0    293.0        318.0
3179    17.0    17.0  …    4.0    4.0          1.0    144.0        147.0
3180    41.0    36.0  …   17.0   11.0          6.0    286.0        270.0
3181    37.0    35.0  …   19.0   10.0          8.0    293.0        313.0
3182    51.0    60.0  …   44.0   27.0         15.0    522.0        549.0

      Verwitwet  Geschieden  Unverheiratet  In eingetrage-ner Partner-schaft  \
1          81.0       217.0            0.0                               7.0
2         178.0       556.0            1.0                              59.0
3        1261.0      3028.0            7.0                             127.0
4         397.0      1109.0            2.0                              53.0
5         616.0      1464.0            2.0                              58.0
…          …          …             …                                …
3178       36.0        50.0            0.0                               2.0
3179       21.0        28.0            0.0                               2.0
3180       33.0        49.0            0.0                               0.0
3181       40.0        68.0            0.0                               0.0
3182       88.0       112.0            0.0                               1.0

      Aufgelöste Partnerschaft
1                          1.0
2                          5.0
3                         25.0
4                         12.0
5                         15.0
…                          …
3178                       0.0
3179                       0.0
3180                       0.0
3181                       0.0
3182                       0.0

[3182 rows x 32 columns]
```

From this table, only popluation count and marital status are taken, the other columns can be deleted, because the information is also available in the second population table.

Therefore, many columns can be deleted here:

```
[90]: population_1.drop(["0-4", "5-9","10-14", "15-19", "20-24", "25-29", "30-34",␣
 ↪"35-39", "40-44", "45-49",
                  "50-54", "55-59", "60-64", "65-69", "70-74", "75-79", "80-84",␣
 ↪"85-89","90 und mehr",
                  "Schweiz", "Ausland", "Mann", "Frau"], axis=1, inplace=True)
```

Let's have a look at the occurrences of the different categories:

```
[91]: print("Ledige Personen in CH:                              " + str(round(np.
 ↪sum(population_1["Ledig"]))) + "  /  " + str(round(np.
 ↪sum(population_1["Ledig"])/np.sum(population_1["Total"])*100, 2))+'%')
print("Verheiratete Personen in CH:                        " + str(round(np.
 ↪sum(population_1["Verheiratet"]))) + "  /  " + str(round(np.
 ↪sum(population_1["Verheiratet"])/np.sum(population_1["Total"])*100, 2))+'%')
print("Verwitwete Personen in CH:                          " + str(round(np.
 ↪sum(population_1["Verwitwet"]))) + "   /  " + str(round(np.
 ↪sum(population_1["Verwitwet"])/np.sum(population_1["Total"])*100, 2))+'%')
print("Geschiedene Personen in CH:                         " + str(round(np.
 ↪sum(population_1["Geschieden"]))) + "   /  " + str(round(np.
 ↪sum(population_1["Geschieden"])/np.sum(population_1["Total"])*100, 2))+'%')
print("Unverheiratete Personen in CH:                      " + str(round(np.
 ↪sum(population_1["Unverheiratet"]))) + "      /  " + str(round(np.
 ↪sum(population_1["Unverheiratet"])/np.sum(population_1["Total"])*100,␣
 ↪2))+'%')
print("Personen mit eingetragener Partnerschaft in CH:   " + str(round(np.
 ↪sum(population_1["In eingetrage-ner Partner-schaft"]))) + "    /  " +␣
 ↪str(round(np.sum(population_1["In eingetrage-ner Partner-schaft"])/np.
 ↪sum(population_1["Total"])*100, 2))+'%')
print("Personen mit aufgelöster Partnerschaft in CH:     " + str(round(np.
 ↪sum(population_1["Aufgelöste Partnerschaft"]))) + "     /  " + str(round(np.
 ↪sum(population_1["Aufgelöste Partnerschaft"])/np.
 ↪sum(population_1["Total"])*100, 2))+'%')
```

```
Ledige Personen in CH:                              3903333  /  45.02%
Verheiratete Personen in CH:                        3588894  /  41.39%
Verwitwete Personen in CH:                          403471   /  4.65%
Geschiedene Personen in CH:                         751735   /  8.67%
Unverheiratete Personen in CH:                      617      /  0.01%
Personen mit eingetragener Partnerschaft in CH:   19022    /  0.22%
Personen mit aufgelöster Partnerschaft in CH:     2981     /  0.03%
```

The marital state "Unverheiratet" means that the marriage has been cancelled somehow. I will categorize this as "ledig" to avoid too many categories. Furthermore, the state "eingetragene Partnerschaft" reflects somehow marriage for relationships between people of the same gender, therefore this will be categorized as "verheiratet", the same principle is valid for the "aufgelöste Partnerschaft".

The desribed categorization will be handled in the next code section:

[92]:
```
population_1["Ledig"] = population_1["Ledig"] + population_1["Unverheiratet"]
population_1["Verheiratet"] = population_1["Verheiratet"] + population_1["In␣
↪eingetrage-ner Partner-schaft"]
population_1["Geschieden"] = population_1["Geschieden"] +␣
↪population_1["Aufgelöste Partnerschaft"]
```

The original columns can therefore be removed:

[93]:
```
population_1.drop(["Unverheiratet", "In eingetrage-ner␣
↪Partner-schaft","Aufgelöste Partnerschaft"], axis=1, inplace=True)
```

[94]:
```
population_1
```

[94]:
```
      Unnamed: 0    Total    Ledig  Verheiratet  Verwitwet  Geschieden
1           1000   3991.0   2378.0       1314.0       81.0       218.0
2           1003   6528.0   4102.0       1687.0      178.0       561.0
3           1004  31084.0  17357.0       9411.0     1261.0      3053.0
4           1005  12465.0   7397.0       3549.0      397.0      1121.0
5           1006  15520.0   8725.0       4700.0      616.0      1479.0
…          …       …        …            …        …           …
3178        9652    699.0    293.0        320.0       36.0        50.0
3179        9655    342.0    144.0        149.0       21.0        28.0
3180        9656    638.0    286.0        270.0       33.0        49.0
3181        9657    714.0    293.0        313.0       40.0        68.0
3182        9658   1272.0    522.0        550.0       88.0       112.0

[3182 rows x 6 columns]
```

Now the columns should be renamed to match the defined ER model (English words):

[95]:
```
population_1.rename(columns={"Unnamed: 0": "PLZ", "Total": "pop_count",
                             "Ledig": "single_count", "Verheiratet":␣
↪"married_count",
                             "Verwitwet": "widowed_count", "Geschieden":␣
↪"divorced_count"}, inplace=True)
```

[96]:
```
population_1
```

[96]:
```
       PLZ  pop_count  single_count  married_count  widowed_count  \
1     1000     3991.0        2378.0         1314.0           81.0
2     1003     6528.0        4102.0         1687.0          178.0
3     1004    31084.0       17357.0         9411.0         1261.0
4     1005    12465.0        7397.0         3549.0          397.0
5     1006    15520.0        8725.0         4700.0          616.0
…     …         …            …             …              …
3178  9652      699.0         293.0          320.0           36.0
3179  9655      342.0         144.0          149.0           21.0
```

```
3180  9656      638.0          286.0          270.0           33.0
3181  9657      714.0          293.0          313.0           40.0
3182  9658     1272.0          522.0          550.0           88.0

      divorced_count
1              218.0
2              561.0
3             3053.0
4             1121.0
5             1479.0
…               …
3178            50.0
3179            28.0
3180            49.0
3181            68.0
3182           112.0

[3182 rows x 6 columns]
```

The table is prepared and can be written into a csv. No shares will be calculated now, because this has to be done on the level of the municipalities (BFS-Nr.) and not the PLZ. After the first joining step, the shares will be calculated.

[97]:
```
population_1.to_csv("../Data/1_Cleaned/population_marital.csv", index=False)
```

### 5.3.2 Population 2 list

In the first population list, we get the data about age segments, country of origin, gender, residence duration and household size.

[98]:
```
population_2[:2]
```

[98]:
```
   GDENR  B20BTOT  B20B11  B20B12  B20B13  B20B14  B20B15  B20B16  B20B21  \
0      1     2014    1724     290     218      42      30       0    1565
1      2    12289    8725    3564    2083     947     533       1    8135

   B20B22  …  B20B55  B20B56  H20PTOT  H20P01  H20P02  H20P03  H20P04  \
0      13  …      12       1      877     269     324     111     132
1    2515  …     145      10     5512    1993    1881     650     685

   H20P05  H20P06  H20PI
0      32       9      2
1     233      70      2

[2 rows x 78 columns]
```

There are 78 columns, which have to been described. Out of the column title, it is not visible, what this means.

[99]:
```
population_2.columns
```

[99]:
```
Index(['GDENR', 'B20BTOT', 'B20B11', 'B20B12', 'B20B13', 'B20B14', 'B20B15',
       'B20B16', 'B20B21', 'B20B22', 'B20B23', 'B20B24', 'B20B25', 'B20B26',
       'B20B27', 'B20B28', 'B20B29', 'B20B30', 'B20BMTOT', 'B20BM01',
       'B20BM02', 'B20BM03', 'B20BM04', 'B20BM05', 'B20BM06', 'B20BM07',
       'B20BM08', 'B20BM09', 'B20BM10', 'B20BM11', 'B20BM12', 'B20BM13',
       'B20BM14', 'B20BM15', 'B20BM16', 'B20BM17', 'B20BM18', 'B20BM19',
       'B20BWTOT', 'B20BW01', 'B20BW02', 'B20BW03', 'B20BW04', 'B20BW05',
       'B20BW06', 'B20BW07', 'B20BW08', 'B20BW09', 'B20BW10', 'B20BW11',
       'B20BW12', 'B20BW13', 'B20BW14', 'B20BW15', 'B20BW16', 'B20BW17',
       'B20BW18', 'B20BW19', 'B20B41', 'B20B42', 'B20B43', 'B20B44', 'B20B45',
       'B20B46', 'B20B51', 'B20B52', 'B20B53', 'B20B54', 'B20B55', 'B20B56',
       'H20PTOT', 'H20P01', 'H20P02', 'H20P03', 'H20P04', 'H20P05', 'H20P06',
       'H20PI'],
      dtype='object')
```

In the explanation document, the abbreviations are explained: "B20" stands for "population 2020", the available year. "H20" for "household 2020".

Looking at the last 3 or 4 characters, B11 to B16 belongs to "Permanent resident population by nationality", B21 to B30 to "Permanent resident population by birthplace".

BM means male population, BW female population. The numbers 01 to 19 reflects age segments in 5-years-groups (0-4, 5-9, 10-14, ... >90).

B41 to B46 show different resident durations within the municipality (>1 year to since birth).

B51 to B56 stands for the residence 1 year before ("same municipality", "same canton", ... , "foreign country").

P01 to P06 is the household size, from 1 to 6+ people. PI is a classification of plausibility, which will not be used.

According to the description in the preliminary study, only the age groups, the population by birthplace, the gender, resident duration and household size will be used. Therefore, the "population by nationality"-categories and the residence 1 year before can be removed:

[100]:
```
population_2.drop(['B20B11', 'B20B12', 'B20B13', 'B20B14', 'B20B15',
       'B20B16', 'B20B51', 'B20B52', 'B20B53', 'B20B54', 'B20B55', 'B20B56']
       ,axis=1, inplace=True)
population_2[:2]
```

[100]:
```
   GDENR  B20BTOT  B20B21  B20B22  B20B23  B20B24  B20B25  B20B26  B20B27  \
0      1     2014    1565      13    1071     481       0     449     293
1      2    12289    8135    2515    2933    2680       7    4154    1895

   B20B28  …  B20B45  B20B46  H20PTOT  H20P01  H20P02  H20P03  H20P04  \
0      54  …     254       0      877     269     324     111     132
```

```
1     1217  …     2053        3      5512     1993     1881      650      685

   H20P05  H20P06  H20PI
0      32       9      2
1     233      70      2

[2 rows x 66 columns]
```

**Age segments** Out of the available date, the age segments should be build as described in the preliminary study: >20, 20-40, 40-60, >60.

As the data is shown PER gender, the age groups have to be summed up and calculated together by the total population number.

[101]:
```
population_2["age0_20"] = (population_2["B20BM01"] + population_2["B20BM02"] +␣
↪population_2["B20BM03"] + population_2["B20BM04"] +
                          population_2["B20BW01"] + population_2["B20BW02"] +␣
↪population_2["B20BW03"] + population_2["B20BW04"]) / population_2["B20BTOT"]

population_2["age20_40"] = (population_2["B20BM05"] + population_2["B20BM06"] +␣
↪population_2["B20BM07"] + population_2["B20BM08"] +
                          population_2["B20BW05"] + population_2["B20BW06"] +␣
↪population_2["B20BW07"] + population_2["B20BW08"]) / population_2["B20BTOT"]

population_2["age40_60"] = (population_2["B20BM09"] + population_2["B20BM10"] +␣
↪population_2["B20BM11"] + population_2["B20BM12"] +
                          population_2["B20BW09"] + population_2["B20BW10"] +␣
↪population_2["B20BW11"] + population_2["B20BW12"]) / population_2["B20BTOT"]

population_2["age60+"] = (population_2["B20BM13"] + population_2["B20BM14"] +␣
↪population_2["B20BM15"] + population_2["B20BM16"] +
                          population_2["B20BM17"] + population_2["B20BM18"] +␣
↪population_2["B20BM19"] +
                          population_2["B20BW13"] + population_2["B20BW14"] +␣
↪population_2["B20BW15"] + population_2["B20BW16"] +
                          population_2["B20BW17"] + population_2["B20BW18"] +␣
↪population_2["B20BW19"]) / population_2["B20BTOT"]

population_2["age0_20cnt"] = (population_2["B20BM01"] + population_2["B20BM02"]␣
↪+ population_2["B20BM03"] + population_2["B20BM04"] +
                          population_2["B20BW01"] + population_2["B20BW02"] +␣
↪population_2["B20BW03"] + population_2["B20BW04"])

population_2["age20_40cnt"] = (population_2["B20BM05"] +␣
↪population_2["B20BM06"] + population_2["B20BM07"] + population_2["B20BM08"] +
```

```
                           population_2["B20BW05"] + population_2["B20BW06"] +␣
 ↪population_2["B20BW07"] + population_2["B20BW08"])

population_2["age40_60cnt"] = (population_2["B20BM09"] +␣
 ↪population_2["B20BM10"] + population_2["B20BM11"] + population_2["B20BM12"] +
                           population_2["B20BW09"] + population_2["B20BW10"] +␣
 ↪population_2["B20BW11"] + population_2["B20BW12"])

population_2["age60+cnt"] = (population_2["B20BM13"] + population_2["B20BM14"]␣
 ↪+ population_2["B20BM15"] + population_2["B20BM16"] +
                           population_2["B20BM17"] + population_2["B20BM18"] +␣
 ↪population_2["B20BM19"] +
                           population_2["B20BW13"] + population_2["B20BW14"] +␣
 ↪population_2["B20BW15"] + population_2["B20BW16"] +
                           population_2["B20BW17"] + population_2["B20BW18"] +␣
 ↪population_2["B20BW19"])
```

**Birthplace** Second, the categorization to birthplace will be done, according to the defined groups: - Birth within municipality (birth_munic) => "B20B22" - Birth within canton (birth_cant) => "B20B23" - Birth within Switzerland (birth_CH) => "B20B24" (other canton) + => "B20B25" (CH, but not assignable) - Birth outside of Switzerland (birth_notCH) => "B20B26"

The fields "B20B27" to "B20B30" specify the country of origin, which I will not consider. Therefore, this columns can be deleted afterwards.

[102]:
```
population_2["birth_munic"] = population_2["B20B22"] / population_2["B20BTOT"]
population_2["birth_cant"] = population_2["B20B23"] / population_2["B20BTOT"]
population_2["birth_CH"] = (population_2["B20B24"] + population_2["B20B25"]) /␣
 ↪population_2["B20BTOT"]
population_2["birth_notCH"] = population_2["B20B26"] / population_2["B20BTOT"]

population_2["birth_munic_cnt"] = population_2["B20B22"]
population_2["birth_cant_cnt"] = population_2["B20B23"]
population_2["birth_CH_cnt"] = (population_2["B20B24"] + population_2["B20B25"])
population_2["birth_notCH_cnt"] = population_2["B20B26"]
```

[103]:
```
(population_2["birth_munic"] + population_2["birth_CH"] +␣
 ↪population_2["birth_cant"] + population_2["birth_notCH"])[:5]
```

[103]:
```
0    1.0
1    1.0
2    1.0
3    1.0
4    1.0
dtype: float64
```

Control shows that the sum is always 100%, which is good.

**Gender** The gender categorization is a simple differentiation between "male" and "female". The share can directly be calculated each.

[104]:
```
population_2["male"] = population_2["B20BMTOT"] / population_2["B20BTOT"]
population_2["female"] = population_2["B20BWTOT"] / population_2["B20BTOT"]

population_2["male_cnt"] = population_2["B20BMTOT"]
population_2["female_cnt"] = population_2["B20BWTOT"]
```

[105]:
```
population_2["male"] + population_2["female"]
```

[105]:
```
0       1.0
1       1.0
2       1.0
3       1.0
4       1.0
       …
2193    1.0
2194    1.0
2195    1.0
2196    1.0
2197    1.0
Length: 2198, dtype: float64
```

Control shows that the sum is always 100%, which is good.

**Residence duration** The fourth category is the length of the residenceship, divided into the defined categories: - 0-1 year ("resid <1y") => "B20B41" - 1-5 years ("resid 1-5y") => "B20B42" - 6-10 years ("resid 6-10y") => "B20B43" - 10+ years ("resid >10 y", including "since birth", even if this could also be less than 10 years.) => "B20B44" + "B20B45"

The last category "B20B46" (not known) will be ignored, as it cannot be matched. So the sum of all categories will not be equal to 1 as a consequence

[106]:
```
# shares data
population_2["resid_0_1y"] = population_2["B20B41"] / population_2["B20BTOT"]
population_2["resid_1_5y"] = population_2["B20B42"] / population_2["B20BTOT"]
population_2["resid_6_10y"] = population_2["B20B43"] / population_2["B20BTOT"]
population_2["resid_10+y"] = (population_2["B20B44"] + population_2["B20B45"]
                              ) / population_2["B20BTOT"]


# count data
population_2["resid_0_1y_cnt"] = population_2["B20B41"]
population_2["resid_1_5y_cnt"] = population_2["B20B42"]
population_2["resid_6_10y_cnt"] = population_2["B20B43"]
population_2["resid_10+y_cnt"] = population_2["B20B44"] + population_2["B20B45"]
```

[107]:
```
population_2["resid_0_1y"] + population_2["resid_1_5y"] +␣
↪population_2["resid_6_10y"] + population_2["resid_10+y"]
```

[107]:
```
0       1.000000
1       0.999756
2       0.999465
3       1.000000
4       1.000000
          …
2193    1.000000
2194    1.000000
2195    1.000000
2196    1.000000
2197    1.000000
Length: 2198, dtype: float64
```

As expected, the sum is not always 1, but this should not be a big deal.

**Household size** The last category build the household size, which will be classified as the following: - 1 person ("hh_1") => "H20P01" - 2 persons ("hh_2") => "H20P02" - 3-5 persons ("hh_3-5") => "H20P03" + "H20P04" + "H20P05" - 6+ persons ("hh_>6") => "H20P06"

[108]:
```
# shares data
population_2["hh_1"] = population_2["H20P01"] / population_2["H20PTOT"]
population_2["hh_2"] = population_2["H20P02"] / population_2["H20PTOT"]
population_2["hh_3_5"] = (population_2["H20P03"] + population_2["H20P04"] +
                          population_2["H20P05"]) / population_2["H20PTOT"]
population_2["hh_6+"] = population_2["H20P06"] / population_2["H20PTOT"]

# count data
population_2["hh_1_cnt"] = population_2["H20P01"]
population_2["hh_2_cnt"] = population_2["H20P02"]
population_2["hh_3_5_cnt"] = (population_2["H20P03"] + population_2["H20P04"] +
                          population_2["H20P05"])
population_2["hh_6+_cnt"] = population_2["H20P06"]
```

[109]:
```
population_2["hh_1"] + population_2["hh_2"] + population_2["hh_3_5"] +␣
↪population_2["hh_6+"]
```

[109]:
```
0       1.0
1       1.0
2       1.0
3       1.0
4       1.0
      …
2193    1.0
2194    1.0
```

```
2195    1.0
2196    1.0
2197    1.0
Length: 2198, dtype: float64
```

Control shows that the sum is always 100%, which is good.

**Renaming + Deleting of unneccessary rows**

[110]:
```python
list(population_2.columns)
```

[110]:
```
['GDENR',
 'B20BTOT',
 'B20B21',
 'B20B22',
 'B20B23',
 'B20B24',
 'B20B25',
 'B20B26',
 'B20B27',
 'B20B28',
 'B20B29',
 'B20B30',
 'B20BMTOT',
 'B20BM01',
 'B20BM02',
 'B20BM03',
 'B20BM04',
 'B20BM05',
 'B20BM06',
 'B20BM07',
 'B20BM08',
 'B20BM09',
 'B20BM10',
 'B20BM11',
 'B20BM12',
 'B20BM13',
 'B20BM14',
 'B20BM15',
 'B20BM16',
 'B20BM17',
 'B20BM18',
 'B20BM19',
 'B20BWTOT',
 'B20BW01',
 'B20BW02',
 'B20BW03',
```

```
'B20BW04',
'B20BW05',
'B20BW06',
'B20BW07',
'B20BW08',
'B20BW09',
'B20BW10',
'B20BW11',
'B20BW12',
'B20BW13',
'B20BW14',
'B20BW15',
'B20BW16',
'B20BW17',
'B20BW18',
'B20BW19',
'B20B41',
'B20B42',
'B20B43',
'B20B44',
'B20B45',
'B20B46',
'H20PTOT',
'H20P01',
'H20P02',
'H20P03',
'H20P04',
'H20P05',
'H20P06',
'H20PI',
'age0_20',
'age20_40',
'age40_60',
'age60+',
'age0_20cnt',
'age20_40cnt',
'age40_60cnt',
'age60+cnt',
'birth_munic',
'birth_cant',
'birth_CH',
'birth_notCH',
'birth_munic_cnt',
'birth_cant_cnt',
'birth_CH_cnt',
'birth_notCH_cnt',
'male',
```

```
 'female',
 'male_cnt',
 'female_cnt',
 'resid_0_1y',
 'resid_1_5y',
 'resid_6_10y',
 'resid_10+y',
 'resid_0_1y_cnt',
 'resid_1_5y_cnt',
 'resid_6_10y_cnt',
 'resid_10+y_cnt',
 'hh_1',
 'hh_2',
 'hh_3_5',
 'hh_6+',
 'hh_1_cnt',
 'hh_2_cnt',
 'hh_3_5_cnt',
 'hh_6+_cnt']
```

Out of all the columns, we only need the newly created columns at the end as well as the "GDENR" and the total population ("B20BTOT"). These two columns should be renamed to "BFS-Nr" and "pop_count" first.

```
[111]: # Renaming columns

population_2.rename({"GDENR":"BFS_Nr", "B20BTOT":"pop_count"}, axis=1,␣
 ↪inplace=True)
```

Now all columns with "B20" or "H20" at the beginning should be removed. These are column numbers 2 to 66:

```
[112]: population_2.columns[2:66]
```

```
[112]: Index(['B20B21', 'B20B22', 'B20B23', 'B20B24', 'B20B25', 'B20B26', 'B20B27',
       'B20B28', 'B20B29', 'B20B30', 'B20BMTOT', 'B20BM01', 'B20BM02',
       'B20BM03', 'B20BM04', 'B20BM05', 'B20BM06', 'B20BM07', 'B20BM08',
       'B20BM09', 'B20BM10', 'B20BM11', 'B20BM12', 'B20BM13', 'B20BM14',
       'B20BM15', 'B20BM16', 'B20BM17', 'B20BM18', 'B20BM19', 'B20BWTOT',
       'B20BW01', 'B20BW02', 'B20BW03', 'B20BW04', 'B20BW05', 'B20BW06',
       'B20BW07', 'B20BW08', 'B20BW09', 'B20BW10', 'B20BW11', 'B20BW12',
       'B20BW13', 'B20BW14', 'B20BW15', 'B20BW16', 'B20BW17', 'B20BW18',
       'B20BW19', 'B20B41', 'B20B42', 'B20B43', 'B20B44', 'B20B45', 'B20B46',
       'H20PTOT', 'H20P01', 'H20P02', 'H20P03', 'H20P04', 'H20P05', 'H20P06',
       'H20PI'],
      dtype='object')
```

```
[113]: population_2.drop(population_2.columns[2:66], axis=1, inplace=True)
```

[114]:
```
population_2
```

[114]:
```
      BFS_Nr  pop_count   age0_20   age20_40   age40_60     age60+  age0_20cnt  \
0          1       2014  0.189672  0.187190  0.350050  0.273088         382
1          2      12289  0.201969  0.278298  0.275856  0.243877        2482
2          3       5610  0.240642  0.225312  0.308734  0.225312        1350
3          4       3801  0.220994  0.189687  0.337543  0.251776         840
4          5       3795  0.216074  0.220553  0.327009  0.236364         820
…        …          …         …         …         …         …         …
2193    6806        560  0.173214  0.228571  0.262500  0.335714          97
2194    6807       1241  0.216761  0.189363  0.275584  0.318292         269
2195    6808       1263  0.182106  0.229612  0.250990  0.337292         230
2196    6809       1096  0.170620  0.208029  0.250912  0.370438         187
2197    6810       1135  0.207048  0.200881  0.279295  0.312775         235

      age20_40cnt  age40_60cnt  age60+cnt  …  resid_6_10y_cnt  \
0             377          705        550  …              324
1            3420         3390       2997  …             1598
2            1264         1732       1264  …              759
3             721         1283        957  …              470
4             837         1241        897  …              533
…             …            …          …  …              …
2193          128          147        188  …               68
2194          235          342        395  …              102
2195          290          317        426  …              121
2196          228          275        406  …              100
2197          228          317        355  …              103

      resid_10+y_cnt      hh_1      hh_2    hh_3_5     hh_6+  hh_1_cnt  \
0               1076  0.306727  0.369441  0.313569  0.010262       269
1               6827  0.361575  0.341255  0.284470  0.012700      1993
2               3295  0.289775  0.338142  0.365295  0.006788       683
3               2218  0.291772  0.345570  0.345570  0.017089       461
4               2236  0.301768  0.335859  0.343434  0.018939       478
…                …         …         …         …         …       …
2193             365  0.334677  0.387097  0.250000  0.028226        83
2194             836  0.366972  0.322936  0.280734  0.029358       200
2195             879  0.403685  0.345059  0.239531  0.011725       241
2196             745  0.354902  0.372549  0.258824  0.013725       181
2197             772  0.371542  0.308300  0.296443  0.023715       188

      hh_2_cnt  hh_3_5_cnt  hh_6+_cnt
0          324         275          9
1         1881        1568         70
2          797         861         16
3          546         546         27
4          532         544         30
```

```
…        …          …          …
2193        96          62          7
2194       176         153         16
2195       206         143          7
2196       190         132          7
2197       156         150         12

[2198 rows x 38 columns]
```

**Writing csv** This table now reflects exactly the desired table from the preliminary study and can therefore be stored as csv:

[115]:
```
population_2.to_csv("../Data/1_Cleaned/population_shares.csv", index=False)
```

## 5.4 Commuter share list

### 5.4.1 Prepare Inbound Data

[116]:
```
inbound_comm[:7] ## First 4 rows can be deleted
```

[116]:
```
  Zupendlerquote 2000          Unnamed: 1  \
0                 NaN                 NaN
1                 NaN                 NaN
2          Regions-ID          Regionsname
3                 NaN                 NaN
4                 NaN             Schweiz
5                   1      Aeugst am Albis
6                   2   Affoltern am Albis

                                              3561
0                                              NaN
1  Anteil der zupendelnden Erwerbstätigen an den …
2                                              NaN
3                                              NaN
4                                        58.882161
5                                        47.699758
6                                        59.777951
```

[117]:
```
inbound_comm.drop([0, 1, 2, 3, 4], axis=0, inplace=True)
```

[118]:
```
inbound_comm[-15:] # after line 2900, no more value is generated => Dropping␣
 ↪last lines!
```

[118]:
```
                                    Zupendlerquote 2000     Unnamed: 1  \
2897                                               6803        Rocourt
```

```
2898                                              6804  Saint-Ursanne
2899                                              6805        Seleute
2900                                              6806   Vendlincourt
2901                                               NaN            NaN
2902                        Erhebungszeitpunkte/ -zeiträume:       NaN
2903                                        Quelle(n):            NaN
2904                                               NaN            NaN
2905                                               NaN            NaN
2906                         Statistischer Atlas der Schweiz       NaN
2907  11 - Mobilität, Verkehr > Pendlermobilität >  …          NaN
2908        Schweiz / Politische Gemeinden / 5.12.2000            NaN
2909                                               NaN            NaN
2910                       Kontakt: statatlas@bfs.admin.ch        NaN
2911  © Bundesamt für Statistik, ThemaKart, Neuchâte…          NaN

                                                  3561
2897                                          6.451613
2898                                         68.100358
2899                                                20
2900                                         44.978166
2901                                               NaN
2902                                         5.12.2000
2903  BFS - Eidgenössische Volkszählung, 1850-2000 (VZ)
2904                                               NaN
2905                                               NaN
2906                                               NaN
2907                                               NaN
2908                                               NaN
2909                                               NaN
2910                                               NaN
2911                                               NaN
```

[119]:
```
inbound_comm.drop(list(range(2901,2912)), axis=0, inplace=True)
```

Columns must be renamed

[120]:
```
inbound_comm.rename({"Zupendlerquote 2000":"BFS_Nr", "Unnamed: 1":
↪"municipality", 3561:"inbound share %"}, axis=1, inplace=True)
```

[121]:
```
inbound_comm
```

[121]:
```
     BFS_Nr         municipality  inbound share %
5         1      Aeugst am Albis        47.699758
6         2   Affoltern am Albis        59.777951
7         3           Bonstetten        48.221344
8         4      Hausen am Albis        42.020666
9         5             Hedingen        69.798658
```

```
...     ...              ...          ...
2896    6802     Roche-d'Or            0
2897    6803        Rocourt     6.451613
2898    6804  Saint-Ursanne    68.100358
2899    6805        Seleute           20
2900    6806   Vendlincourt    44.978166

[2896 rows x 3 columns]
```

In the next step, the values have to be transformed to a share between 0 and 1 (=> / 100)

```
[122]: inbound_comm["inbound share %"] = inbound_comm["inbound share %"] / 100
inbound_comm.rename({"inbound share %":"inbound_share"}, axis = 1, inplace=True)
```

```
[123]: inbound_comm[:3]
```

```
[123]:   BFS_Nr        municipality inbound_share
5       1      Aeugst am Albis      0.476998
6       2   Affoltern am Albis       0.59778
7       3           Bonstetten      0.482213
```

### 5.4.2 Add Outbound Data

The outbound table does have the exact same structure and can therefore be treated the same way

```
[124]: outbound_comm.drop([0, 1, 2, 3, 4], axis=0, inplace=True)
outbound_comm.drop(list(range(2901,2912)), axis=0, inplace=True)
outbound_comm.rename({"Wegpendlerquote 2000":"BFS_Nr", "Unnamed: 1":
 ↪"municipality", 3581:"outbound share %"}, axis=1, inplace=True)
```

```
[125]: outbound_comm
```

```
[125]:      BFS_Nr        municipality outbound share %
5          1      Aeugst am Albis        75.757576
6          2   Affoltern am Albis        62.358731
7          3           Bonstetten        82.860881
8          4      Hausen am Albis        70.467836
9          5             Hedingen         75.34997
...      ...                  ...              ...
2896    6802           Roche-d'Or        13.333333
2897    6803              Rocourt        61.842105
2898    6804        Saint-Ursanne        49.142857
2899    6805              Seleute        33.333333
2900    6806         Vendlincourt               55

[2896 rows x 3 columns]
```

In the next step, the values have to be transformed to a share between 0 and 1 (=> / 100)

[126]:
```
outbound_comm["outbound share %"] = outbound_comm["outbound share %"] / 100
outbound_comm.rename({"outbound share %":"outbound_share"}, axis = 1,␣
 ↪inplace=True)
```

[127]:
```
outbound_comm[:3]
```

[127]:
```
   BFS_Nr        municipality outbound_share
5       1      Aeugst am Albis       0.757576
6       2   Affoltern am Albis       0.623587
7       3           Bonstetten       0.828609
```

Now the outbound share can easily be added to the inbound_commuters table

[128]:
```
commuters = inbound_comm
```

[129]:
```
commuters["outbound_share"] = outbound_comm["outbound_share"]
commuters
```

[129]:
```
      BFS_Nr         municipality inbound_share outbound_share
5          1      Aeugst am Albis      0.476998       0.757576
6          2   Affoltern am Albis       0.59778       0.623587
7          3           Bonstetten      0.482213       0.828609
8          4      Hausen am Albis      0.420207       0.704678
9          5             Hedingen      0.697987         0.7535
…      …                    …            …              …
2896   6802           Roche-d'Or           0.0       0.133333
2897   6803              Rocourt      0.064516       0.618421
2898   6804        Saint-Ursanne      0.681004       0.491429
2899   6805              Seleute           0.2       0.333333
2900   6806          Vendlincourt      0.449782           0.55

[2896 rows x 4 columns]
```

The name of the municipality is not necessary here, as it will be provided from other tables after joining. Therefore, it can be deleted:

[130]:
```
commuters.drop(["municipality"], axis=1, inplace=True)
commuters
```

[130]:
```
      BFS_Nr inbound_share outbound_share
5          1      0.476998       0.757576
6          2       0.59778       0.623587
7          3      0.482213       0.828609
8          4      0.420207       0.704678
9          5      0.697987         0.7535
…      …            …              …
```

```
2896    6802          0.0       0.133333
2897    6803     0.064516       0.618421
2898    6804     0.681004       0.491429
2899    6805          0.2       0.333333
2900    6806     0.449782           0.55

[2896 rows x 3 columns]
```

#### 5.4.3 Write Commuters Table

[131]:
```
commuters.to_csv("../Data/1_Cleaned/commuters.csv", index=False)
```

First, a foreigner quote and a female quote are calculated out of the data and integrated.

### 5.5 Cars table

From the cars table, we need the count of private cars per municipality and fuel type.

[132]:
```
cars[:2]
```

[132]:
```
            Gemeinde Fahrzeuggruppe Treibstoff  2015  2016  2017  2018  2019  \
0  1 Aeugst am Albis  Personenwagen     Benzin   845   822   815   816   809
1  1 Aeugst am Albis  Personenwagen     Diesel   288   306   316   318   326

   2020  2021
0   804   792
1   329   320
```

As some data are only available for the year 2020, I will reduce the data to the year 2020 first:

[133]:
```
cars.drop(["2015", "2016", "2017", "2018", "2019", "2021"], axis=1,␣
 ↪inplace=True)
```

#### 5.5.1 Categorizing Fuel type

[134]:
```
cars["Treibstoff"].unique()
```

[134]:
```
array(['Benzin', 'Diesel', 'Benzin-elektrisch: Normal-Hybrid',
       'Benzin-elektrisch: Plug-in-Hybrid',
       'Diesel-elektrisch: Normal-Hybrid',
       'Diesel-elektrisch: Plug-in-Hybrid', 'Elektrisch', 'Wasserstoff',
       'Gas (mono- und bivalent)', 'Anderer'], dtype=object)
```

The number of categories should be reduced to the following: - Combustion: (Benzin + Diesel) - Hybrid (Electric and all all 4 possible hybrid categories) - Other (all the rest)

[135]:
```python
cars["Treibstoff"]
```

[135]:
```
0                                   Benzin
1                                   Diesel
2           Benzin-elektrisch: Normal-Hybrid
3          Benzin-elektrisch: Plug-in-Hybrid
4           Diesel-elektrisch: Normal-Hybrid
                         …
151405     Diesel-elektrisch: Plug-in-Hybrid
151406                          Elektrisch
151407                         Wasserstoff
151408             Gas (mono- und bivalent)
151409                             Anderer
Name: Treibstoff, Length: 151410, dtype: object
```

[137]:
```python
Combustion = ["Benzin", "Diesel"]
Electric = ['Benzin-elektrisch: Normal-Hybrid',
       'Benzin-elektrisch: Plug-in-Hybrid',
       'Diesel-elektrisch: Normal-Hybrid',
       'Diesel-elektrisch: Plug-in-Hybrid',
       'Elektrisch']
Other = ['Wasserstoff',
       'Gas (mono- und bivalent)', 'Anderer']
```

[138]:
```python
7 % 2
```

[138]:
```
1
```

[139]:
```python
len(cars)
```

[139]:
```
151410
```

[140]:
```python
cars["fueltp"] = "Other"

for i in range(len(cars)):
  if cars["Treibstoff"][i] in Combustion:
    cars["fueltp"][i] = "Combustion"
  elif cars["Treibstoff"][i] in Electric:
    cars["fueltp"][i] = "Electric"
```

```
<ipython-input-140-61c6f2a4aa0a>:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cars["fueltp"][i] = "Combustion"
<ipython-input-140-61c6f2a4aa0a>:7: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  cars["fueltp"][i] = "Electric"
```

Now I don't need the column "Treibstoff" anymore:

[141]:
```
cars.drop(["Treibstoff"], axis=1, inplace=True)
cars[:2]
```

[141]:
```
            Gemeinde Fahrzeuggruppe  2020      fueltp
0  1 Aeugst am Albis  Personenwagen   804  Combustion
1  1 Aeugst am Albis  Personenwagen   329  Combustion
```

### 5.5.2 Categorizing Car type

Only the car types, which are used for individual transport, should be used, because other cars are not considered as relevant for public transport tickets.

[142]:
```
cars["Fahrzeuggruppe"].unique()
```

[142]:
```
array(['Personenwagen', 'Personentransportfahrzeuge',
       'Sachentransportfahrzeuge', 'Landwirtschaftsfahrzeuge',
       'Industriefahrzeuge', 'Motorräder', 'Anhänger'], dtype=object)
```

From these categories, I only want to take "Personenwagen" and "Motorräder", the rest can be ignored.

[143]:
```
individual = [] # individual transport like personenwagen and motorräder

for i in range(len(cars)):
  individual.append(cars["Fahrzeuggruppe"][i] == "Personenwagen" or
        cars["Fahrzeuggruppe"][i] == "Motorräder")

print(individual[:20])
```

```
[True, True, True, True, True, True, True, True, True, True, False, False,
False, False, False, False, False, False, False, False]
```

[144]:
```
cars_reduced = copy.deepcopy(cars[individual]) # only take defined categories␣
 ↪above!
cars_reduced
```

[144]:
```
                  Gemeinde Fahrzeuggruppe  2020      fueltp
0        1 Aeugst am Albis  Personenwagen   804  Combustion
1        1 Aeugst am Albis  Personenwagen   329  Combustion
2        1 Aeugst am Albis  Personenwagen    30    Electric
```

```
3         1 Aeugst am Albis  Personenwagen    12    Electric
4         1 Aeugst am Albis  Personenwagen     2    Electric
…                    …             …    …          …
151395     6810 La Baroche      Motorräder     0    Electric
151396     6810 La Baroche      Motorräder     2    Electric
151397     6810 La Baroche      Motorräder     0       Other
151398     6810 La Baroche      Motorräder     0       Other
151399     6810 La Baroche      Motorräder     0       Other

[43260 rows x 4 columns]
```

Now, the "Fahrzeuggruppe" is not needed anymore:

[145]:
```
cars_reduced.drop(["Fahrzeuggruppe"], axis=1, inplace=True)
cars_reduced[:2]
```

[145]:
```
            Gemeinde  2020      fueltp
0  1 Aeugst am Albis   804  Combustion
1  1 Aeugst am Albis   329  Combustion
```

### 5.5.3 Grouping and re-arranging

[146]:
```
cars_group = cars_reduced.groupby(["Gemeinde","fueltp"], group_keys=False).sum()
cars_group
```

[146]:
```
                              2020
Gemeinde          fueltp
1 Aeugst am Albis Combustion  1400
                  Electric      78
                  Other          3
10 Obfelden       Combustion  3525
                  Electric     124
…                              …
993 Wangenried    Electric       5
                  Other          1
995 Wiedlisbach   Combustion  1616
                  Electric      54
                  Other          2

[6489 rows x 1 columns]
```

Remove multi-indexing!

[147]:
```
cars_group = cars_group.reset_index(level=[0,1])
cars_group
```

[147]:
```
                 Gemeinde      fueltp  2020
0      1 Aeugst am Albis  Combustion  1400
1      1 Aeugst am Albis    Electric    78
2      1 Aeugst am Albis       Other     3
3            10 Obfelden  Combustion  3525
4            10 Obfelden    Electric   124
…                    …         …   …
6484      993 Wangenried    Electric     5
6485      993 Wangenried       Other     1
6486     995 Wiedlisbach  Combustion  1616
6487     995 Wiedlisbach    Electric    54
6488     995 Wiedlisbach       Other     2

[6489 rows x 3 columns]
```

The next step is to pivot the table into a more wide format to have different enginges separately.

[148]:
```python
cars_pivot = cars_group.pivot(index=["Gemeinde"], columns="fueltp",␣
↪values="2020")
cars_pivot
```

[148]:
```
fueltp                         Combustion  Electric  Other
Gemeinde
1 Aeugst am Albis                    1400        78      3
10 Obfelden                          3525       124     12
100 Stadel                           1654        57      3
1001 Doppleschwand                    557        10      0
1002 Entlebuch                       2129        32      1
…                                       …       …    …
990 Walliswil bei Niederbipp          205         8      0
991 Walliswil bei Wangen              463        10      2
992 Wangen an der Aare               1683        23      4
993 Wangenried                        298         5      1
995 Wiedlisbach                      1616        54      2

[2163 rows x 3 columns]
```

[149]:
```python
print(f"Share of car type 'other': {cars_pivot['Other'].aggregate(sum) /␣
↪(cars_pivot['Combustion'].aggregate(sum) + cars_pivot['Electric'].
↪aggregate(sum) + cars_pivot['Other'].aggregate(sum))*100} %")
```

```
Share of car type 'other': 0.2730794794252071 %
```

only 0.27% of all vehicles are classified as "other". Therefore, this category can be removed:

[150]:
```python
cars_pivot.drop(["Other"], axis=1, inplace=True)
cars_pivot
```

[150]:
```
fueltp                         Combustion  Electric
Gemeinde
1 Aeugst am Albis                    1400        78
10 Obfelden                          3525       124
100 Stadel                           1654        57
1001 Doppleschwand                    557        10
1002 Entlebuch                       2129        32
…                                       …         …
990 Walliswil bei Niederbipp          205         8
991 Walliswil bei Wangen              463        10
992 Wangen an der Aare               1683        23
993 Wangenried                        298         5
995 Wiedlisbach                      1616        54

[2163 rows x 2 columns]
```

### 5.5.4 Write BFS Number into table

Now the only thing missing is the BFS-Nr. It is included in the "Gemeinde", but I only need the number, not the name of the municipality. So the number can be taken, while the rest will be deleted in the next step.

[151]:
```
Gemeinde = cars_pivot.index.tolist()
```

[152]:
```
Gemeinde[0]
```

[152]:
```
'1 Aeugst am Albis'
```

[153]:
```
bfs = []
for i in range(len(Gemeinde)):
  bfs.append([int(s) for s in Gemeinde[i].split() if s.isdigit()]) # write␣
↪numbers ouf of string in list
```

[154]:
```
bfs[:5]
```

[154]:
```
[[1], [10], [100], [1001], [1002]]
```

This is a nested list, which has to be corrected:

[155]:
```
from itertools import chain # to unnest the list

Gemeinde_list = list(chain(*bfs))
```

[156]:
```
Gemeinde_list[:5]
```

[156]:
```
[1, 10, 100, 1001, 1002]
```

Write BFS Number into table:

[157]:
```
cars_pivot["BFS-Nr"] = Gemeinde_list
cars_pivot[:5]
```

[157]:
```
fueltp               Combustion  Electric  BFS-Nr
Gemeinde
1 Aeugst am Albis          1400        78       1
10 Obfelden                3525       124      10
100 Stadel                 1654        57     100
1001 Doppleschwand          557        10    1001
1002 Entlebuch             2129        32    1002
```

Now the BFS-Nr. can be used as index instead, the "Gemeinde" is not used anymore:

[158]:
```
cars_pivot.set_index('BFS-Nr', inplace=True)
```

[159]:
```
cars_pivot[:5]
```

[159]:
```
fueltp  Combustion  Electric
BFS-Nr
1             1400        78
10            3525       124
100           1654        57
1001           557        10
1002          2129        32
```

#### 5.5.5 Write csv

This table can now be exported!

[160]:
```
cars_pivot.to_csv("../Data/1_Cleaned/cars_cleaned.csv", index=True)
```

### 5.6 Travelcards

2 Datasets for the Travelcards are present: 1. List of GA's and Half fare tickets (ga_hta) 2. List of Regional Fare Network tickets (fn_tck)

The two of them have to be combined together at the end

#### 5.6.1 List of GA's and Half fare tickets

[161]:
```
ga_hta[:3]
```

[161]:
```
   Jahr_An_Anno  PLZ_NPA   GA_AG  GA_AG_flag  HTA_ADT_meta-prezzo  \
0          2012     1000    72.0         NaN                976.0
```

```
1          2012     1003    744.0          NaN                 3195.0
2          2012     1004   1919.0          NaN                 8167.0

   HTA_ADT_meta-prezzo_flag
0                       NaN
1                       NaN
2                       NaN
```

From the Travelcards table, we need the number of GA's per PLZ for the year 2020 (as for the other tables)

**Reducing to year 2020**

[162]:
```
ga_hta_2020 = copy.deepcopy(ga_hta[ga_hta["Jahr_An_Anno"]==2020])
```

[163]:
```
ga_hta_2020[:2]
```

[163]:
```
       Jahr_An_Anno  PLZ_NPA  GA_AG  GA_AG_flag  HTA_ADT_meta-prezzo  \
25506          2020     1000   75.0         NaN               1258.0
25507          2020     1003  677.0         NaN               3449.0

       HTA_ADT_meta-prezzo_flag
25506                       NaN
25507                       NaN
```

**Removing unneccessary columns** The "flags" columns reflect PLZ with only few people living there. The number there is a mean value of all PLZ with a low population and with the same digit at the first place. This reflects therefore not the true value, but can still be used, as the value is a good estimate.

[164]:
```
ga_hta_2020.drop(["Jahr_An_Anno", "GA_AG_flag","HTA_ADT_meta-prezzo_flag"],␣
 ↪axis=1, inplace=True)
ga_hta_2020[:2]
```

[164]:
```
       PLZ_NPA  GA_AG  HTA_ADT_meta-prezzo
25506     1000   75.0               1258.0
25507     1003  677.0               3449.0
```

**Renaming columns**

[165]:
```
ga_hta_2020.rename(columns={"PLZ_NPA":"PLZ", "GA_AG":"GA",
                            "HTA_ADT_meta-prezzo":"HTA"}, inplace=True)
ga_hta_2020[:3]
```

[165]:
```
        PLZ     GA     HTA
25506  1000   75.0  1258.0
```

```
25507  1003   677.0   3449.0
25508  1004  1653.0  10657.0
```

### 5.6.2 Regional fare network tickets

[166]:
```
fn_tck[:2]
```

[166]:
```
   Jahr_An_Anno  PLZ_NPA Verbund_Communaute_Comunita  Anzahl_Nombre_Quantita  \
0          2017     1001                         ZVV                2.985232
1          2017     1003                         ZVV                2.985232

   Flag
0   3.0
1   3.0
```

**Reducing to year 2020**

[167]:
```
fn_tck_2020 = copy.deepcopy(fn_tck[fn_tck["Jahr_An_Anno"]==2020])
fn_tck_2020[:3]
```

[167]:
```
       Jahr_An_Anno  PLZ_NPA Verbund_Communaute_Comunita  \
16225          2020     1000                  Onde Verte
16226          2020     1000                     unireso
16227          2020     1000                     mobilis

       Anzahl_Nombre_Quantita  Flag
16225                    2.10   3.0
16226                    2.48   3.0
16227                  711.00   NaN
```

**Removing unneccessary columns** The "flag" column reflect PLZ with only few people living there. The number there is a mean value of all PLZ with a low population and with the same digit at the first place. This reflects therefore not the true value, but can still be used, as the value is a good estimate.

[168]:
```
fn_tck_2020.drop(["Flag", "Jahr_An_Anno"], axis=1, inplace=True)
fn_tck_2020[:2]
```

[168]:
```
       PLZ_NPA Verbund_Communaute_Comunita  Anzahl_Nombre_Quantita
16225     1000                  Onde Verte                    2.10
16226     1000                     unireso                    2.48
```

**Group by PLZ** Some PLZ show more than just one fare network systems in it. Assuming that one person only possesses one card of one system, the different numbers can be summed up to get the amount of regional fare tickets per PLZ:

```
[169]: fn_tck_2020_group = fn_tck_2020.groupby(["PLZ_NPA"], group_keys=False).sum().
       ↪round(0).astype(int)
       fn_tck_2020_group[:10]
```

```
[169]:          Anzahl_Nombre_Quantita
       PLZ_NPA
       1000                        716
       1001                          4
       1002                          4
       1003                        772
       1004                       4383
       1005                       1796
       1006                       2355
       1007                       3434
       1008                       1177
       1009                       2265
```

The PLZ should not be the index here, therefore I reset the index:

```
[170]: fn_tck_2020_group.reset_index(level=0, inplace=True)
       fn_tck_2020_group[:2]
```

```
[170]:    PLZ_NPA  Anzahl_Nombre_Quantita
       0     1000                     716
       1     1001                       4
```

**Renaming Columns**

```
[171]: fn_tck_2020_group.rename(columns={"PLZ_NPA":"PLZ", "Anzahl_Nombre_Quantita":
       ↪"fn_tck"}, inplace=True)
       fn_tck_2020_group[:2]
```

```
[171]:     PLZ  fn_tck
       0  1000     716
       1  1001       4
```

### 5.6.3 Joining GA + regional fare network tickets

**Preparation** First, ensure that both PLZ are saved as type "integer":

```
[172]: ga_hta_2020["PLZ"] = ga_hta_2020["PLZ"].astype(int)
```

```
[173]: fn_tck_2020_group["PLZ"] = fn_tck_2020_group["PLZ"].astype(int)
```

**Joining** Now, the joining can be done

[174]:
```
travelcards = ga_hta_2020.merge(fn_tck_2020_group, on="PLZ", how = "outer")
travelcards
```

[174]:
```
       PLZ      GA      HTA  fn_tck
0     1000    75.0   1258.0   716.0
1     1003   677.0   3449.0   772.0
2     1004  1653.0  10657.0  4383.0
3     1005   825.0   5237.0  1796.0
4     1006  1217.0   6811.0  2355.0
…      …       …        …       …
3286  9495     NaN      NaN    20.0
3287  9496     NaN      NaN    16.0
3288  9497     NaN      NaN     5.0
3289  9572     NaN      NaN     5.0
3290  9721     NaN      NaN     5.0

[3291 rows x 4 columns]
```

**Fill NA values** To end, the NaN values should be filled up with 0, as there are no such tickets present.

[175]:
```
travelcards.fillna(0, inplace=True)
```

**Writing csv**

[176]:
```
travelcards.to_csv("../Data/1_Cleaned/travelcards.csv", index=False)
```

## 5.7 Town directory

The town directory forms the base to join all other entities together. From this table, we need the PLZ, BFS_Nr, canton, coordinates, language and municipality name.

[177]:
```
town_directory[:11]
```

[177]:
```
    Ortschaftsname   PLZ  Zusatzziffer     Gemeindename  BFS-Nr  \
0      Lausanne 25  1000            25         Lausanne    5586
1      Lausanne 26  1000            26         Lausanne    5586
2      Lausanne 27  1000            27         Lausanne    5586
3         Lausanne  1003             0         Lausanne    5586
4         Lausanne  1004             0         Lausanne    5586
5         Lausanne  1005             0         Lausanne    5586
6         Lausanne  1006             0         Lausanne    5586
7         Lausanne  1007             0         Lausanne    5586
8   Jouxtens-Mézery  1008             2  Jouxtens-Mézery    5585
```

```
9            Prilly  1008             0          Prilly    5589
10            Pully  1009             0           Pully    5590

   Kantonskürzel            E            N Sprache
0             VD  542094.8938  157051.9666      fr
1             VD  543068.1153  156403.0412      fr
2             VD  541921.1403  154775.3096      fr
3             VD  537956.7751  152398.2869      fr
4             VD  537089.8121  153349.5648      fr
5             VD  538907.7414  152372.3783      fr
6             VD  538483.9524  148573.8617      fr
7             VD  536344.2571  149061.8207      fr
8             VD  535509.0763  156070.6429      fr
9             VD  536259.1817  154013.5757      fr
10            VD  540390.0563  151170.8879      fr
```

### 5.7.1 Delete unneccessary columns

The localities ("Ortschaftsname") are not needed here, the same also for the "Zusatzziffer", which can differentiate between several localities within the same PLZ. The coordinates can be used to visualize at the end, but not needed here in the analysis.

[178]:
```
town_directory.drop(columns=["Ortschaftsname", "Zusatzziffer", "E", "N"],␣
 ↪inplace=True)
town_directory[:3]
```

[178]:
```
    PLZ Gemeindename  BFS-Nr Kantonskürzel Sprache
0  1000     Lausanne    5586            VD      fr
1  1000     Lausanne    5586            VD      fr
2  1000     Lausanne    5586            VD      fr
```

### 5.7.2 Re-group on level BFS

[179]:
```
town_dir_PLZ = town_directory.groupby(["PLZ", "BFS-Nr"]).first().reset_index()
town_dir_PLZ[5:12]
```

[179]:
```
     PLZ  BFS-Nr     Gemeindename Kantonskürzel Sprache
5   1007    5586         Lausanne            VD      fr
6   1008    5585  Jouxtens-Mézery            VD      fr
7   1008    5589           Prilly            VD      fr
8   1009    5590            Pully            VD      fr
9   1010    5586         Lausanne            VD      fr
10  1011    5586         Lausanne            VD      fr
11  1012    5586         Lausanne            VD      fr
```

### 5.7.3 Writing csv

[180]:
```
town_dir_PLZ.to_csv("../Data/1_Cleaned/town_directory_cleaned.csv", index=False)
```

# 6 Joining temporary entities

In the case of the stop list cleaned and the distances, the desired goal entities for the database can only be reached if several original tables are joined together.

This will be done in this chapter, using already cleaned data from chapter 5.

## 6.1 Stop list cleaned

For the stop list, the nr_days attribute of the stop_count list has to be added to the stations table.

Let's have a look first at the different tables:

### 6.1.1 Overview + Preparation

[181]:
```
stations = pd.read_csv("../Data/1_Cleaned/stations.csv")
stations[:3]
```

[181]:
```
   Dst-Nr85                   Name  Status Kt.  Gde-Nr Ortschaft  \
0   8506013                 Aadorf       3  TG  4551.0    Aadorf
1   8573363        Aadorf, Bahnhof       3  TG  4551.0    Aadorf
2   8576958  Aadorf, Matthofstrasse      3  TG  4551.0    Aadorf

  Verkehrsmittel TU-Abk  E-Koord.  N-Koord.
0            Zug    SBB   2710378   1260736
1            Bus    PAG   2710335   1260768
2            Bus    PAG   2710483   1260407
```

[182]:
```
stations_stops = pd.read_csv("../Data/1_Cleaned/stop_count.csv")
stations_stops
```

[182]:
```
         ride_id  stop_id  nr_days
0          23000  8504351      359
1          23000  8504350      359
2          23000  8504352      359
3          23001  8504351      359
4          23001  8504350      359
…            …        …        …
4584247      906  8509195      163
4584248      906  8509251      163
4584249      906  8509253      163
```

```
4584250      906  8509189      163
4584251      906  8509192      163

[4584252 rows x 3 columns]
```

[183]:
```
stations_stops.loc[stations_stops["stop_id"] == 8503530]
```

[183]:
```
         ride_id  stop_id  nr_days
1416458    15601  8503530      101
1416470    15602  8503530      101
2905287      901  8503530      255
2905297      902  8503530      255
2905307      903  8503530      255
2905317      905  8503530      255
2905327      907  8503530      255
2905337      908  8503530      255
2905347      909  8503530      255
2905357      910  8503530      255
2905367      911  8503530      255
2905377      912  8503530      255
2905387      913  8503530      255
2905397      914  8503530      255
2905407      915  8503530      255
2905417      916  8503530      255
2905625     1115  8503530      255
2905700     1123  8503530      255
2905747     1128  8503530      255
2905764     1129  8503530      255
2905798     1131  8503530      255
2905869     1138  8503530      255
2906023     1162  8503530      255
2906054     1164  8503530      255
2906125     1267  8503530      109
2906126     1267  8503530      109
2906136     1273  8503530      109
2906137     1273  8503530      109
```

For the joining, the name of the column should be identically:

[184]:
```
stations_stops.rename(columns={"stop_id":"Dst-Nr85"}, inplace=True)
```

[185]:
```
stations_stops[["Dst-Nr85"]].describe()
```

[185]:
```
          Dst-Nr85
count  4.584252e+06
mean   8.572817e+06
std    3.079674e+04
```

```
min      8.500010e+06
25%      8.574257e+06
50%      8.587907e+06
75%      8.592060e+06
max      8.596125e+06
```

### 6.1.2 Overview station types

[186]:
```python
stations[["Verkehrsmittel"]].value_counts()
```

[186]:
```
Verkehrsmittel
Bus                                       22756
Zug                                        1786
Sesselbahn                                  596
Kabinenbahn                                 498
Bus_Tram                                    392
Schiff                                      369
Tram                                        133
Standseilbahn                               121
Zahnradbahn                                  59
Bus_Metro                                    17
Metro                                        10
Kabinenbahn_Standseilbahn                     8
Zug_Bus                                       7
Bus_Standseilbahn                             6
Aufzug                                        4
Bus_Kabinenbahn                               3
Zug_Bus_Tram                                  3
Zug_Tram                                      3
Kabinenbahn_Sesselbahn                        2
Zug_Kabinenbahn                               2
Zug_Standseilbahn                             2
Kabinenbahn_Zahnradbahn                       1
Schiff_Standseilbahn                          1
Schiff_Zahnradbahn                            1
Kabinenbahn_Sesselbahn_Zahnradbahn            1
Kabinenbahn_Sesselbahn_Standseilbahn          1
Bus_Tram_Zahnradbahn                          1
Bus_Tram_Standseilbahn                        1
Zug_Metro                                     1
dtype: int64
```

Obviously, there are many different transport types possible! Also many combinations are possible. At the end I only want to have 3 categories: train, bus and rest. So I will set the following categories as "train": - "Zug", "Standseilbahn", "Zahnradbahn", "Zug_... "(different categories), "Metro" Additionnally, I will then match the following categories to "bus": - "Bus", "Tram" (similar to bus than to train), "Bus_... "(different categories)

All the rest will go into the "rest" category.

Now I will perform this classification in the next step:

[187]:
```
Zug = ["Zug", "Standseilbahn", "Zahnradbahn", "Zug_Metro",
       "Zug_Standseilbahn", "Zug_Kabinenbahn", "Zug_Tram",
       "Zug_Bus_Tram", "Zug_Bus", "Metro"]

Bus = ["Bus", "Tram", "Bus_Tram_Standseilbahn", "Bus_Tram_Zahnradbahn",
       "Bus_Kabinenbahn", "Bus_Standseilbahn", "Bus_Metro", "Bus_Tram"]
```

[188]:
```
for i, row in stations.iterrows():
  ifor_val = "Other"
  if row["Verkehrsmittel"] in Zug:
    ifor_val = "Zug"
  if row["Verkehrsmittel"] in Bus:
    ifor_val = "Bus"

  stations.at[i, 'tp_means'] = ifor_val
```

[189]:
```
stations
```

[189]:
```
       Dst-Nr85                        Name  Status  Kt.  Gde-Nr  \
0       8506013                      Aadorf       3   TG  4551.0
1       8573363             Aadorf, Bahnhof       3   TG  4551.0
2       8576958      Aadorf, Matthofstrasse       3   TG  4551.0
3       8506853           Aadorf, Morgental       3   TG  4551.0
4       8573362             Aadorf, Zentrum       3   TG  4551.0
…         …                           …     …   …     …
28383   8591218  Zürich,Kalkbreite/Bhf.Wiedikon       3   ZH   261.0
28384   8503653               Zürichhorn (See)       3   ZH   261.0
28385   8530528                       Älpli       3   GR  3954.0
28386   8518708                     Äuli (B)       3   GR  3861.0
28387   8518838                  Überlingen       3  NaN     NaN

       Ortschaft Verkehrsmittel TU-Abk  E-Koord.  N-Koord. tp_means
0         Aadorf            Zug    SBB   2710378   1260736      Zug
1         Aadorf            Bus    PAG   2710335   1260768      Bus
2         Aadorf            Bus    PAG   2710483   1260407      Bus
3         Aadorf            Bus    PAG   2709827   1261373      Bus
4         Aadorf            Bus    PAG   2710079   1261060      Bus
…           …           …     …       …         …        …
28383     Zürich       Bus_Tram    VBZ   2681770   1247629      Bus
28384     Zürich         Schiff    ZSG   2684205   1245239    Other
28385  Malans GR    Kabinenbahn    AMG   2763452   1209076    Other
28386    Fideris            NaN    RhB   2776150   1199237    Other
```

```
28387        NaN            Zug     DB   2729242   1292368      Zug

[28388 rows x 11 columns]
```

[190]:
```
stations[stations["Gde-Nr"]==572]
```

[190]:
```
      Dst-Nr85                       Name  Status Kt.  Gde-Nr  \
4205   8508371                    Bönigen       3  BE   572.0
4206   8518394          Bönigen Gleisende       3  BE   572.0
4207   8518393     Bönigen Werkstätte BLS       3  BE   572.0
4208   8507490             Bönigen, Dorf       3  BE   572.0
4209   8576388        Bönigen, Erschwanden       3  BE   572.0
4210   8576386        Bönigen, Hauetenbach       3  BE   572.0
4211   8507390  Bönigen, Lütschinenbrücke       3  BE   572.0
4212   8579113             Bönigen, Sand       3  BE   572.0
4213   8576385          Bönigen, Schlössli       3  BE   572.0
4214   8576378              Bönigen, See       3  BE   572.0
4215   8576387           Bönigen, Wäldli       3  BE   572.0

                Ortschaft Verkehrsmittel TU-Abk  E-Koord.  N-Koord. tp_means
4205  Bönigen b. Interlaken         Schiff  BLSSF   2635144   1171011    Other
4206  Bönigen b. Interlaken            NaN    BLS   2634620   1170876    Other
4207  Bönigen b. Interlaken            NaN    BLS   2634422   1170905    Other
4208  Bönigen b. Interlaken            Bus    PAG   2634864   1170645      Bus
4209  Bönigen b. Interlaken            Bus    PAG   2637354   1171450      Bus
4210  Bönigen b. Interlaken            Bus    PAG   2635842   1170930      Bus
4211  Bönigen b. Interlaken            Bus    PAG   2634496   1170885      Bus
4212  Bönigen b. Interlaken            Bus    PAG   2634625   1170425      Bus
4213  Bönigen b. Interlaken            Bus    PAG   2635385   1170916      Bus
4214  Bönigen b. Interlaken            Bus    PAG   2635166   1170861      Bus
4215  Bönigen b. Interlaken            Bus    PAG   2636592   1170890      Bus
```

### 6.1.3 Join "nr_days" to stations

[191]:
```
stop_list_cleaned = stations.merge(stations_stops, on="Dst-Nr85")
```

[192]:
```
stop_list_cleaned
```

[192]:
```
         Dst-Nr85             Name  Status Kt.  Gde-Nr  Ortschaft  \
0         8506013           Aadorf       3  TG  4551.0     Aadorf
1         8506013           Aadorf       3  TG  4551.0     Aadorf
2         8506013           Aadorf       3  TG  4551.0     Aadorf
3         8506013           Aadorf       3  TG  4551.0     Aadorf
4         8506013           Aadorf       3  TG  4551.0     Aadorf
…             …                …      …  ..     …          …
4581752   8503653  Zürichhorn (See)       3  ZH   261.0     Zürich
```

```
4581753   8503653  Zürichhorn (See)       3  ZH   261.0     Zürich
4581754   8503653  Zürichhorn (See)       3  ZH   261.0     Zürich
4581755   8530528             Älpli       3  GR  3954.0  Malans GR
4581756   8530528             Älpli       3  GR  3954.0  Malans GR

        Verkehrsmittel TU-Abk  E-Koord.  N-Koord. tp_means  ride_id  nr_days
0                  Zug    SBB   2710378   1260736      Zug    19215      255
1                  Zug    SBB   2710378   1260736      Zug    19219      255
2                  Zug    SBB   2710378   1260736      Zug    19220      255
3                  Zug    SBB   2710378   1260736      Zug    19223      255
4                  Zug    SBB   2710378   1260736      Zug    19224      255
…                    …    …        …         …        …        …        …
4581752          Schiff   ZSG   2684205   1245239    Other     3037       62
4581753          Schiff   ZSG   2684205   1245239    Other     3040       62
4581754          Schiff   ZSG   2684205   1245239    Other     3043       62
4581755      Kabinenbahn   AMG   2763452   1209076    Other        1      184
4581756      Kabinenbahn   AMG   2763452   1209076    Other       51      184

[4581757 rows x 13 columns]
```

### 6.1.4 Calculate Stations per means of transport an BFS (without stops)

[193]:
```python
stations_list_reduced = stop_list_cleaned[["Gde-Nr", "tp_means","Dst-Nr85"]]
stations_list_reduced
```

[193]:
```
         Gde-Nr tp_means  Dst-Nr85
0        4551.0      Zug   8506013
1        4551.0      Zug   8506013
2        4551.0      Zug   8506013
3        4551.0      Zug   8506013
4        4551.0      Zug   8506013
…           …       …        …
4581752   261.0    Other   8503653
4581753   261.0    Other   8503653
4581754   261.0    Other   8503653
4581755  3954.0    Other   8530528
4581756  3954.0    Other   8530528

[4581757 rows x 3 columns]
```

[194]:
```python
stations_list_grouped = stations_list_reduced.groupby(["Gde-Nr","tp_means"],␣
↪group_keys=False).nunique()
stations_list_grouped[190:200]
```

```
[194]:                      Dst-Nr85
        Gde-Nr tp_means
        225.0  Bus              3
               Zug              1
        226.0  Bus              5
        227.0  Bus              8
               Zug              1
        228.0  Bus             18
               Zug              1
        230.0  Bus             18
               Zug             10
        231.0  Bus              5
```

The multiindex leads later to problems and should therefore be removed here:

```
[195]: stations_list_indexed = stations_list_grouped.reset_index(level=[0,1])
       stations_list_indexed[190:200]
```

```
[195]:      Gde-Nr tp_means  Dst-Nr85
        190  225.0      Bus         3
        191  225.0      Zug         1
        192  226.0      Bus         5
        193  227.0      Bus         8
        194  227.0      Zug         1
        195  228.0      Bus        18
        196  228.0      Zug         1
        197  230.0      Bus        18
        198  230.0      Zug        10
        199  231.0      Bus         5
```

This has to be unformed into a wide pivot!

```
[196]: stations_list_pivot = stations_list_indexed.pivot(index="Gde-Nr",␣
        ↪columns="tp_means", values="Dst-Nr85").reset_index()
       stations_list_pivot.fillna(0, inplace=True)
       stations_list_pivot
```

```
[196]: tp_means  Gde-Nr   Bus  Other  Zug
       0            1.0   6.0    0.0  0.0
       1            2.0  13.0    0.0  1.0
       2            3.0   7.0    0.0  1.0
       3            4.0  10.0    0.0  0.0
       4            5.0   2.0    0.0  1.0
       …              …     …      …    …
       2087      6806.0   0.0    0.0  1.0
       2088      6807.0   6.0    0.0  3.0
       2089      6808.0  22.0    0.0  1.0
       2090      6809.0   8.0    0.0  0.0
```

```
2091      6810.0  12.0    0.0  0.0

[2092 rows x 4 columns]
```

[197]:
```python
stations_list_pivot.rename(columns={"Gde-Nr":"BFS_Nr", "Bus":"bus_stat", "Zug":
 ↪"train_stat", "Other":"other_stat"}, inplace=True)
```

[198]:
```python
stations_list_pivot[:5]
```

[198]:
```
tp_means  BFS_Nr  bus_stat  other_stat  train_stat
0            1.0       6.0         0.0         0.0
1            2.0      13.0         0.0         1.0
2            3.0       7.0         0.0         1.0
3            4.0      10.0         0.0         0.0
4            5.0       2.0         0.0         1.0
```

### 6.1.5 Calculate train, bus and other stop count

[199]:
```python
stop_list_cleaned[:123]
```

[199]:
```
     Dst-Nr85             Name  Status Kt.  Gde-Nr Ortschaft Verkehrsmittel  \
0     8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
1     8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
2     8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
3     8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
4     8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
..        …                …      …  ..      …         …             …
118   8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
119   8506013           Aadorf       3  TG  4551.0    Aadorf            Zug
120   8573363  Aadorf, Bahnhof       3  TG  4551.0    Aadorf            Bus
121   8573363  Aadorf, Bahnhof       3  TG  4551.0    Aadorf            Bus
122   8573363  Aadorf, Bahnhof       3  TG  4551.0    Aadorf            Bus

    TU-Abk  E-Koord.  N-Koord. tp_means  ride_id  nr_days
0      SBB   2710378   1260736      Zug    19215      255
1      SBB   2710378   1260736      Zug    19219      255
2      SBB   2710378   1260736      Zug    19220      255
3      SBB   2710378   1260736      Zug    19223      255
4      SBB   2710378   1260736      Zug    19224      255
..     …       …         …        …      …        …
118    SBB   2710378   1260736      Zug    13615      108
119    SBB   2710378   1260736      Zug    13617      109
120    PAG   2710335   1260768      Bus    83401      255
121    PAG   2710335   1260768      Bus    83402      255
122    PAG   2710335   1260768      Bus    83403      364
```

```
[123 rows x 13 columns]
```

[200]:
```python
stop_list_cleaned_reduced = stop_list_cleaned[["Gde-Nr", "tp_means", "nr_days"]]
stop_list_cleaned_reduced
```

[200]:
```
         Gde-Nr tp_means  nr_days
0        4551.0      Zug      255
1        4551.0      Zug      255
2        4551.0      Zug      255
3        4551.0      Zug      255
4        4551.0      Zug      255
…          …        …       …
4581752   261.0    Other       62
4581753   261.0    Other       62
4581754   261.0    Other       62
4581755  3954.0    Other      184
4581756  3954.0    Other      184

[4581757 rows x 3 columns]
```

[201]:
```python
stop_list_group = stop_list_cleaned_reduced.groupby(["Gde-Nr","tp_means"],␣
 ↪group_keys=False).sum()
stop_list_group[190:200]
```

[201]:
```
                 nr_days
Gde-Nr tp_means
225.0  Bus         64140
       Zug         29509
226.0  Bus         83881
227.0  Bus        135492
       Zug         41132
228.0  Bus        157371
       Zug         33237
230.0  Bus        292003
       Zug        734564
231.0  Bus         18746
```

The multiindex leads later to problems and should therefore be removed here:

[202]:
```python
stop_list_indexed = stop_list_group.reset_index(level=[0,1])
stop_list_indexed[190:200]
```

[202]:
```
     Gde-Nr tp_means  nr_days
190   225.0      Bus    64140
191   225.0      Zug    29509
192   226.0      Bus    83881
193   227.0      Bus   135492
```

```
194    227.0       Zug    41132
195    228.0       Bus   157371
196    228.0       Zug    33237
197    230.0       Bus   292003
198    230.0       Zug   734564
199    231.0       Bus    18746
```

This has to be unformed into a wide pivot!

[203]:
```
stoplist_pivot = stop_list_indexed.pivot(index="Gde-Nr", columns="tp_means",␣
↪values="nr_days").reset_index()
```

[204]:
```
stoplist_pivot.fillna(0, inplace=True)
```

[205]:
```
stoplist_pivot[1000:1005]
```

[205]:
```
tp_means  Gde-Nr       Bus   Other       Zug
1000      3233.0  193275.0     0.0       0.0
1001      3234.0  184766.0     0.0       0.0
1002      3235.0  142642.0   870.0   59454.0
1003      3236.0  176524.0     0.0  101392.0
1004      3237.0  311220.0  1450.0   53356.0
```

[206]:
```
stoplist_pivot.rename(columns={"Gde-Nr":"BFS_Nr", "Bus":"bus_count", "Zug":
↪"train_count", "Other":"other_count"}, inplace=True)
```

[207]:
```
print(f"Bus stops in CH: {stoplist_pivot['bus_count'].sum()}")
print(f"Train stops in CH: {stoplist_pivot['train_count'].sum()}")
print(f"Other PT stops in CH: {stoplist_pivot['other_count'].sum()}")
```

```
Bus stops in CH: 635792977.0
Train stops in CH: 61040241.0
Other PT stops in CH: 2258945.0
```

### 6.1.6 Join Stations count to stoplist

[208]:
```
stat_stop = stoplist_pivot.merge(stations_list_pivot, on="BFS_Nr")
```

Now the stops by population has to be calculated:

### 6.1.7 Join population data

[209]:
```
pop_shares = pd.read_csv("../Data/1_Cleaned/population_shares.csv")
pop_shares[:3]
```

```
[209]:    BFS_Nr  pop_count   age0_20  age20_40  age40_60    age60+  age0_20cnt  \
0       1       2014  0.189672  0.187190  0.350050  0.273088         382
1       2      12289  0.201969  0.278298  0.275856  0.243877        2482
2       3       5610  0.240642  0.225312  0.308734  0.225312        1350

   age20_40cnt  age40_60cnt  age60+cnt  …  resid_6_10y_cnt  resid_10+y_cnt  \
0          377          705        550  …              324            1076
1         3420         3390       2997  …             1598            6827
2         1264         1732       1264  …              759            3295

       hh_1      hh_2    hh_3_5     hh_6+  hh_1_cnt  hh_2_cnt  hh_3_5_cnt  \
0  0.306727  0.369441  0.313569  0.010262       269       324         275
1  0.361575  0.341255  0.284470  0.012700      1993      1881        1568
2  0.289775  0.338142  0.365295  0.006788       683       797         861

   hh_6+_cnt
0          9
1         70
2         16

[3 rows x 38 columns]
```

I only need the pop_count column here

```
[210]: bfs_pop = pop_shares[["BFS_Nr", "pop_count"]]
```

```
[211]: stop_pop = stat_stop.merge(bfs_pop, on="BFS_Nr")
stop_pop
```

```
[211]:       BFS_Nr  bus_count  other_count  train_count  bus_stat  other_stat  \
0        1.0   210319.0          0.0          0.0       6.0         0.0
1        2.0   488680.0          0.0      51616.0      13.0         0.0
2        3.0   249494.0          0.0      51616.0       7.0         0.0
3        4.0   234267.0          0.0          0.0      10.0         0.0
4        5.0    43000.0          0.0      51616.0       2.0         0.0
…        …          …            …            …         …
2077  6806.0        0.0          0.0      15420.0       0.0         0.0
2078  6807.0    64218.0          0.0      34654.0       6.0         0.0
2079  6808.0   162731.0          0.0      29848.0      22.0         0.0
2080  6809.0    82398.0          0.0          0.0       8.0         0.0
2081  6810.0   225457.0          0.0          0.0      12.0         0.0

      train_stat  pop_count
0            0.0       2014
1            1.0      12289
2            1.0       5610
3            0.0       3801
```

```
4              1.0       3795
…           …        …
2077           1.0        560
2078           3.0       1241
2079           1.0       1263
2080           0.0       1096
2081           0.0       1135

[2082 rows x 8 columns]
```

#### 6.1.8 Calculate stations and stops per population

[212]:
```python
stop_pop["bus_stops_per_pop"] = stop_pop["bus_count"] / stop_pop["pop_count"]
stop_pop["train_stops_per_pop"] = stop_pop["train_count"] /␣
 ↪stop_pop["pop_count"]
stop_pop["other_stops_per_pop"] = stop_pop["other_count"] /␣
 ↪stop_pop["pop_count"]

stop_pop["bus_stat_per_1000"] = stop_pop["bus_stat"] / stop_pop["pop_count"] *␣
 ↪1000
stop_pop["train_stat_per_1000"] = stop_pop["train_stat"] /␣
 ↪stop_pop["pop_count"] * 1000
stop_pop["other_stat_per_1000"] = stop_pop["other_stat"] /␣
 ↪stop_pop["pop_count"] * 1000
```

[213]:
```python
stop_pop[:3]
```

[213]:
```
   BFS_Nr  bus_count  other_count  train_count  bus_stat  other_stat  \
0     1.0   210319.0          0.0          0.0       6.0         0.0
1     2.0   488680.0          0.0      51616.0      13.0         0.0
2     3.0   249494.0          0.0      51616.0       7.0         0.0

   train_stat  pop_count  bus_stops_per_pop  train_stops_per_pop  \
0         0.0       2014         104.428500             0.000000
1         1.0      12289          39.765644             4.200179
2         1.0       5610          44.473084             9.200713

   other_stops_per_pop  bus_stat_per_1000  train_stat_per_1000  \
0                  0.0           2.979146             0.000000
1                  0.0           1.057857             0.081374
2                  0.0           1.247772             0.178253

   other_stat_per_1000
0                  0.0
1                  0.0
```

```
2                    0.0
```

### 6.1.9 Writing csv

[214]:
```
stop_pop.to_csv("../Data/2_Joined_entities/stop_list_final.csv", index=False)
```

## 6.2 City distances

### 6.2.1 Read table

[215]:
```
distances = pd.read_csv("../Data/1_Cleaned/distances.csv")
distances[:3]
```

[215]:

| | from | to | dist_street | dist_pt | time_st | time_pt |
|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 5.326 | 4.183 | 14.342 | 20.483 |
| 1 | 1 | 2 | 5.948 | 6.062 | 15.830 | 24.290 |
| 2 | 1 | 3 | 9.613 | 11.986 | 20.440 | 42.945 |

### 6.2.2 Join population data

[216]:
```
population = pd.read_csv("../Data/1_Cleaned/population_shares.csv")
population[:3]
```

[216]:

| | BFS_Nr | pop_count | age0_20 | age20_40 | age40_60 | age60+ | age0_20cnt | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2014 | 0.189672 | 0.187190 | 0.350050 | 0.273088 | 382 | |
| 1 | 2 | 12289 | 0.201969 | 0.278298 | 0.275856 | 0.243877 | 2482 | |
| 2 | 3 | 5610 | 0.240642 | 0.225312 | 0.308734 | 0.225312 | 1350 | |

| | age20_40cnt | age40_60cnt | age60+cnt | … | resid_6_10y_cnt | resid_10+y_cnt | \ |
|---|---|---|---|---|---|---|---|
| 0 | 377 | 705 | 550 | … | 324 | 1076 | |
| 1 | 3420 | 3390 | 2997 | … | 1598 | 6827 | |
| 2 | 1264 | 1732 | 1264 | … | 759 | 3295 | |

| | hh_1 | hh_2 | hh_3_5 | hh_6+ | hh_1_cnt | hh_2_cnt | hh_3_5_cnt | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.306727 | 0.369441 | 0.313569 | 0.010262 | 269 | 324 | 275 | |
| 1 | 0.361575 | 0.341255 | 0.284470 | 0.012700 | 1993 | 1881 | 1568 | |
| 2 | 0.289775 | 0.338142 | 0.365295 | 0.006788 | 683 | 797 | 861 | |

| | hh_6+_cnt |
|---|---|
| 0 | 9 |
| 1 | 70 |
| 2 | 16 |

```
[3 rows x 38 columns]
```

Only the BFS-Nr and pop_count for merging are necessary here

[217]:
```
population.columns[1:3]
```

[217]:
```
Index(['pop_count', 'age0_20'], dtype='object')
```

[218]:
```
population_reduced = copy.deepcopy(population[["BFS_Nr", "pop_count"]])
population_reduced[:3]
```

[218]:

| | BFS_Nr | pop_count |
|---|---|---|
| 0 | 1 | 2014 |
| 1 | 2 | 12289 |
| 2 | 3 | 5610 |

Now the population should once be joined according to the ”from” population and once according to the ”to” population. So I rename the column therefore two times.

[219]:
```
population_reduced.rename(columns={"BFS_Nr":"from", "pop_count":"pop_from"},␣
 ↪inplace=True)
```

[220]:
```
dist_pop1 = distances.merge(population_reduced, on ="from")
```

[221]:
```
population_reduced.rename(columns={"from":"to", "pop_from":"pop_to"},␣
 ↪inplace=True)
```

[222]:
```
dist_pop2 = dist_pop1.merge(population_reduced, on ="to")
dist_pop2[:3]
```

[222]:

| | from | to | dist_street | dist_pt | time_st | time_pt | pop_from | pop_to |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 5.326 | 4.183 | 14.342 | 20.483 | 2014 | 2014 |
| 1 | 2 | 1 | 5.948 | 6.062 | 15.830 | 24.290 | 12289 | 2014 |
| 2 | 3 | 1 | 9.613 | 11.986 | 20.440 | 42.945 | 5610 | 2014 |

### 6.2.3 Classify ”pop_from” and ”pop_to”

In the next step, the different municipalities should be classified, according to the description in the preliminary study: - Big city: > 100’000 people - Medium city: 30’000 - 100’000 people - Rest: < 30’000 people

This should be applied to the ”pop_from” and the ”pop_to” field:

[223]:
```
# classification table
classification = [{"low": 0, "high": 30000, "name": "-"},
        {"low": 30000, "high": 100000, "name": "medium"},
        {"low": 100000, "high": 1000000, "name": "big"}]
```

```
class_df = pd.DataFrame(classification)

#create bins from original data
bins = list(class_df["high"])
bins.insert(0,0)

dist_pop2["from_cat"] = pd.cut(dist_pop2["pop_from"], bins, labels =␣
 ↪class_df["name"])
dist_pop2["to_cat"] = pd.cut(dist_pop2["pop_to"], bins, labels =␣
 ↪class_df["name"])
dist_pop2[21870:21875]
```

[223]:

```
        from  to  dist_street  dist_pt  time_st  time_pt  pop_from  pop_to  \
21870    195  11       33.061   43.348   48.642   97.310     10780    2704
21871    196  11       42.928   54.624   55.805   96.343      4082    2704
21872    197  11       43.127   44.861   52.084   69.662      5193    2704
21873    198  11       50.088   50.271   53.187   80.168     35337    2704
21874    199  11       45.284   46.876   49.522   81.918     18865    2704

      from_cat to_cat
21870        -      -
21871        -      -
21872        -      -
21873   medium      -
21874        -      -
```

[224]:

```
#for i in range(len(dist_pop2)):
#for i in range(20):
```

### 6.2.4 Create final city_distances table

A new table is needed with all BFS_Nr only occurring once. This has to be filled later with the minimal distances and time amount needed for PT and streets, both for medium and big cities.

[225]:

```
city_distances = pd.DataFrame({"BFS_Nr":dist_pop2["from"].unique(),␣
 ↪"PT_dist_medium":0,
                          "PT_time_medium":0, "PT_dist_big":0, "PT_time_big":0,
                         "str_dist_medium":0, "str_time_medium":0,
                         "str_dist_big":0, "str_time_big":0})
city_distances[:3]
```

[225]:

```
   BFS_Nr  PT_dist_medium  PT_time_medium  PT_dist_big  PT_time_big  \
0       1               0               0            0            0
1       2               0               0            0            0
2       3               0               0            0            0
```

```
   str_dist_medium  str_time_medium  str_dist_big  str_time_big
0                0                0             0             0
1                0                0             0             0
2                0                0             0             0
```

The index should be the BFS_Nr here, which makes it easier to iterate afterwards:

[226]:
```
city_distances.set_index("BFS_Nr", inplace=True)
```

### 6.2.5 Find minimal distances/time and fill table

[227]:
```
for i in dist_pop2["from"].unique():

  # make a cut of the dataset with alle "to"-distances of category "medium"
  dist_temp = dist_pop2[dist_pop2["from"]==i] # all distances with the same␣
↪"from" municipality
  dist_temp = dist_temp[dist_temp["to_cat"]=="medium"] # within, all distances␣
↪with a "to_cat" of medium

  # write now the minimal distances and time in the city distances table
  city_distances["str_dist_medium"].loc[i] = min(dist_temp["dist_street"]) #␣
↪minimal dist_street
  city_distances["PT_dist_medium"].loc[i] = min(dist_temp["dist_pt"]) # minimal␣
↪dist_pt
  city_distances["str_time_medium"].loc[i] = min(dist_temp["time_st"]) #␣
↪minimal time_street
  city_distances["PT_time_medium"].loc[i] = min(dist_temp["time_pt"]) # minimal␣
↪time_pt


  # now make another cut of the dataset with all "to"-distances of category␣
↪"big"
  dist_temp = dist_pop2[dist_pop2["from"]==i] # all distances with the same␣
↪"from" municipality
  dist_temp = dist_temp[dist_temp["to_cat"]=="big"] # within, all distances␣
↪with a "to_cat" of medium

  # write now the minimal distances and time in the city distances table
  city_distances["str_dist_big"].loc[i] = min(dist_temp["dist_street"]) #␣
↪minimal dist_street
  city_distances["PT_dist_big"].loc[i] = min(dist_temp["dist_pt"]) # minimal␣
↪dist_pt
  city_distances["str_time_big"].loc[i] = min(dist_temp["time_st"]) # minimal␣
↪time_street
```

```
 city_distances["PT_time_big"].loc[i] = min(dist_temp["time_pt"]) # minimal␣
↪time_pt
```

```
/usr/local/lib/python3.8/dist-packages/pandas/core/indexing.py:1732:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  self._setitem_single_block(indexer, value, name)
```

### 6.2.6 Calculate Comparison factors PT / Street

At the end, it is probably mostly relevant for the decision of the transport, what is faster. Therefore, a factor is calculated both for street and Public Transport, which compares the time.

[228]:
```
city_distances["PT_fact_big"] = city_distances["PT_time_big"] /␣
↪city_distances["str_time_big"]
city_distances["PT_fact_medium"] = city_distances["PT_time_medium"] /␣
↪city_distances["str_time_medium"]
```

[229]:
```
city_distances
```

[229]:
```
        PT_dist_medium  PT_time_medium  PT_dist_big  PT_time_big  \
BFS_Nr
1               21.327          51.392       25.793       61.008
2               15.384          33.779       25.355       45.628
3               22.463          43.891       18.120       37.031
4               15.902          44.969       30.128       63.564
5               17.715          36.447       22.436       39.591
…                  …               …            …            …
6806            74.164          97.112       77.084      110.411
6807            72.741          93.558       75.660      110.916
6808            55.915          74.479       58.536       89.818
6809            77.381         117.050       79.608      126.877
6810            72.658          96.194       75.406      113.674

        str_dist_medium  str_time_medium  str_dist_big  str_time_big  \
BFS_Nr
1                22.158           32.677        22.288        35.522
2                17.267           22.651        21.131        27.870
3                27.129           28.739        14.706        23.281
4                11.590           23.337        23.171        37.718
5                20.315           29.129        17.598        26.014
…                   …                …             …             …
6806             64.815           62.915        46.482        72.767
6807             74.429           66.274        65.179        83.016
```

```
6808                51.676            53.887         64.969         72.605
6809                68.819            62.720         75.191         81.352
6810                58.764            59.812         44.557         66.530

        PT_fact_big  PT_fact_medium
BFS_Nr
1          1.717471        1.572727
2          1.637173        1.491281
3          1.590610        1.527228
4          1.685243        1.926940
5          1.521911        1.251227
…             …             …
6806       1.517322        1.543543
6807       1.336080        1.411685
6808       1.237077        1.382133
6809       1.559605        1.866231
6810       1.708613        1.608273

[2175 rows x 10 columns]
```

This table looks quite good! This can be written into a csv now:

### 6.2.7 Writing csv

[230]:
```
city_distances.to_csv("../Data/2_Joined_entities/city_distances.csv",␣
↪index=True)
```

## 6.3 Cars

For the cars, the cars by 1000 people has to be calculated.

### 6.3.1 Loading datasets

[231]:
```
cars = pd.read_csv("../Data/1_Cleaned/cars_cleaned.csv")
cars[:3]
```

[231]:
```
   BFS-Nr  Combustion  Electric
0       1        1400        78
1      10        3525       124
2     100        1654        57
```

[232]:
```
pop_shares = pd.read_csv("../Data/1_Cleaned/population_shares.csv")
pop_shares[:3]
```

[232]:
```
   BFS_Nr  pop_count   age0_20  age20_40  age40_60    age60+  age0_20cnt  \
0       1       2014  0.189672  0.187190  0.350050  0.273088         382
1       2      12289  0.201969  0.278298  0.275856  0.243877        2482
2       3       5610  0.240642  0.225312  0.308734  0.225312        1350

   age20_40cnt  age40_60cnt  age60+cnt  …  resid_6_10y_cnt  resid_10+y_cnt  \
0          377          705        550  …              324            1076
1         3420         3390       2997  …             1598            6827
2         1264         1732       1264  …              759            3295

       hh_1      hh_2    hh_3_5      hh_6+  hh_1_cnt  hh_2_cnt  hh_3_5_cnt  \
0  0.306727  0.369441  0.313569  0.010262       269       324         275
1  0.361575  0.341255  0.284470  0.012700      1993      1881        1568
2  0.289775  0.338142  0.365295  0.006788       683       797         861

   hh_6+_cnt
0          9
1         70
2         16

[3 rows x 38 columns]
```

Only pop_count column is needed here

[233]:
```python
bfs_pop = pop_shares[["BFS_Nr", "pop_count"]]
```

### 6.3.2 Joining population data to cars

[234]:
```python
cars_pop = cars.merge(bfs_pop, left_on = "BFS-Nr", right_on="BFS_Nr")
cars_pop
```

[234]:
```
      BFS-Nr  Combustion  Electric  BFS_Nr  pop_count
0          1        1400        78       1       2014
1         10        3525       124      10       5779
2        100        1654        57     100       2336
3       1001         557        10    1001        816
4       1002        2129        32    1002       3230
…        …           …        …       …          …
2151     990         205         8     990        227
2152     991         463        10     991        604
2153     992        1683        23     992       2377
2154     993         298         5     993        407
2155     995        1616        54     995       2382

[2156 rows x 5 columns]
```

[235]:
```
cars_pop.drop(columns=["BFS_Nr"], inplace=True)
```

### 6.3.3 Calculating cars per 1000 inhabitants

[236]:
```
cars_pop["comb_car_1000"] = cars_pop["Combustion"] / cars_pop["pop_count"] *␣
↪1000
cars_pop["el_car_1000"] = cars_pop["Electric"] / cars_pop["pop_count"] * 1000
```

The population is not not needed anymore

[237]:
```
cars_pop.drop(columns=["pop_count"], inplace=True)
cars_pop.rename(columns={"BFS-Nr":"BFS_Nr"}, inplace=True)
cars_pop[:3]
```

[237]:
```
   BFS_Nr  Combustion  Electric  comb_car_1000  el_car_1000
0       1        1400        78     695.134062    38.728898
1      10        3525       124     609.967122    21.456999
2     100        1654        57     708.047945    24.400685
```

[238]:
```
cars_pop[cars_pop["BFS_Nr"] == 261]
```

[238]:
```
     BFS_Nr  Combustion  Electric  comb_car_1000  el_car_1000
473     261      156483      8093     370.920029    19.183271
```

### 6.3.4 Writing csv

[239]:
```
cars_pop.to_csv("../Data/2_Joined_entities/cars_final.csv", index=False)
```

## 6.4 Town directory

[240]:
```
town_dir_PLZ = pd.read_csv("../Data/1_Cleaned/town_directory_cleaned.csv")
```

Unfortunately, some duplicate PLZ are present, which belong to different municipalities. This generates problems in aggregating on the BFS-Nr. and when using the PLZ as a primary key of the table in the relational database. To get an idea about the number of such cases, I have to check for duplicates:

[241]:
```
len(town_dir_PLZ[town_dir_PLZ["PLZ"].duplicated(keep=False)]) # keep=False =>␣
↪show all duplicates!
```

[241]: 481

481 cases where the same PLZ occurs in different BFS-Nr. Let's look at one example

[242]:
```
town_dir_PLZ[town_dir_PLZ["PLZ"].duplicated(keep=False)][200:207]
```

[242]:

| | PLZ | BFS-Nr | Gemeindename | Kantonskürzel | Sprache |
|---|---|---|---|---|---|
| 949 | 2933 | 6793 | Lugnez | JU | fr |
| 991 | 3053 | 310 | Rapperswil (BE) | BE | de |
| 992 | 3053 | 535 | Deisswil bei Münchenbuchsee | BE | de |
| 993 | 3053 | 536 | Diemerswil | BE | de |
| 994 | 3053 | 546 | Münchenbuchsee | BE | de |
| 995 | 3053 | 553 | Wiggiswil | BE | de |
| 1032 | 3126 | 869 | Kaufdorf | BE | de |

The PLZ ”3053” belongs to 5 different municipalities with different BFS-Nr!

The question now is, how to bring the data on the level of the PLZ together with the data on the level of the BFS_Nr.

One possibility is based on the population numbers. Data for the PLZ 3053 should be distributed to the different BFS_Nr looking at the specific population of the municipalities. Therefore, the population data on the level of PLZ (”population_marital”) as well as the population of the BFS_Nr (”population_shares”) should be joined here. With this, factors can be calculated for the different entries of PLZ.

### 6.4.1 Joining population data

[243]:
```
pop_plz = pd.read_csv("../Data/1_Cleaned/population_marital.csv")
pop_plz[:2]
```

[243]:

| | PLZ | pop_count | single_count | married_count | widowed_count | divorced_count |
|---|---|---|---|---|---|---|
| 0 | 1000 | 3991.0 | 2378.0 | 1314.0 | 81.0 | 218.0 |
| 1 | 1003 | 6528.0 | 4102.0 | 1687.0 | 178.0 | 561.0 |

[244]:
```
pop_bfs = pd.read_csv("../Data/1_Cleaned/population_shares.csv")
pop_bfs[:2]
```

[244]:

| | BFS_Nr | pop_count | age0_20 | age20_40 | age40_60 | age60+ | age0_20cnt | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2014 | 0.189672 | 0.187190 | 0.350050 | 0.273088 | 382 | |
| 1 | 2 | 12289 | 0.201969 | 0.278298 | 0.275856 | 0.243877 | 2482 | |

| | age20_40cnt | age40_60cnt | age60+cnt | … | resid_6_10y_cnt | resid_10+y_cnt | \ |
|---|---|---|---|---|---|---|---|
| 0 | 377 | 705 | 550 | … | 324 | 1076 | |
| 1 | 3420 | 3390 | 2997 | … | 1598 | 6827 | |

| | hh_1 | hh_2 | hh_3_5 | hh_6+ | hh_1_cnt | hh_2_cnt | hh_3_5_cnt | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.306727 | 0.369441 | 0.313569 | 0.010262 | 269 | 324 | 275 | |
| 1 | 0.361575 | 0.341255 | 0.284470 | 0.012700 | 1993 | 1881 | 1568 | |

hh_6+_cnt

```
0           9
1          70

[2 rows x 38 columns]
```

From the two tables, I need the "pop_count" columns as well as the "BFS_Nr" or the "PLZ" column. The rest can be dropped.

```
[245]: pop_plz.drop(["single_count", "married_count", "widowed_count",␣
        ↪"divorced_count"], axis=1, inplace=True)
```

```
[246]: pop_plz.rename(columns={"pop_count":"pop_PLZ"}, inplace=True)
```

```
[247]: pop_bfs_red = copy.deepcopy(pop_bfs[["BFS_Nr", "pop_count"]])
```

```
[248]: pop_bfs_red.rename(columns={"pop_count":"pop_BFS"}, inplace=True)
```

```
[249]: town_PLZ = town_dir_PLZ.merge(pop_plz, on="PLZ", how="left")
```

```
[250]: town_pop_PLZ_BFS = town_PLZ.merge(pop_bfs_red, left_on="BFS-Nr",␣
        ↪right_on="BFS_Nr", how="left")
       town_pop_PLZ_BFS[:10]
```

```
[250]:     PLZ  BFS-Nr     Gemeindename Kantonskürzel Sprache  pop_PLZ  BFS_Nr  \
       0  1000    5586         Lausanne            VD      fr   3991.0  5586.0
       1  1003    5586         Lausanne            VD      fr   6528.0  5586.0
       2  1004    5586         Lausanne            VD      fr  31084.0  5586.0
       3  1005    5586         Lausanne            VD      fr  12465.0  5586.0
       4  1006    5586         Lausanne            VD      fr  15520.0  5586.0
       5  1007    5586         Lausanne            VD      fr  22299.0  5586.0
       6  1008    5585  Jouxtens-Mézery            VD      fr  13755.0  5585.0
       7  1008    5589           Prilly            VD      fr  13755.0  5589.0
       8  1009    5590            Pully            VD      fr  18568.0  5590.0
       9  1010    5586         Lausanne            VD      fr  15216.0  5586.0

           pop_BFS
       0  140202.0
       1  140202.0
       2  140202.0
       3  140202.0
       4  140202.0
       5  140202.0
       6    1412.0
       7   12360.0
       8   18694.0
       9  140202.0
```

[251]:
```
town_pop_PLZ_BFS.drop(["BFS_Nr"], axis=1, inplace=True)
```

### 6.4.2 Calculating Factor to join data to the level of BFS

A littel example first:

[252]:
```
town_pop_PLZ_BFS[town_pop_PLZ_BFS["PLZ"]==3303]
```

[252]:

| | PLZ | BFS-Nr | Gemeindename | Kantonskürzel | Sprache | pop_PLZ | pop_BFS |
|---|---|---|---|---|---|---|---|
| 1129 | 3303 | 540 | Jegenstorf | BE | de | 6227.0 | 5738.0 |
| 1130 | 3303 | 557 | Zuzwil (BE) | BE | de | 6227.0 | 563.0 |

[253]:
```
town_pop_PLZ_BFS[town_pop_PLZ_BFS["PLZ"]==3305]
```

[253]:

| | PLZ | BFS-Nr | Gemeindename | Kantonskürzel | Sprache | pop_PLZ | pop_BFS |
|---|---|---|---|---|---|---|---|
| 1131 | 3305 | 540 | Jegenstorf | BE | de | 502.0 | 5738.0 |
| 1132 | 3305 | 541 | Iffwil | BE | de | 502.0 | 428.0 |

These two tables show the complexity of the situation: There are two PLZ (3303, 3305) and 3 municipalities (540, 541, 557). If data on the PLZ level should be aggregated to the municipality level, in this case it is not possible.

Jegenstorf has 2 different PLZ, while these 2 PLZ are used from other municipalities at the same time. Therefore, all these situations have to be looked at very carefully. This is done in the following coding sequence:

First, a copy of the table is made and enlarged with 3 new columns, which are filled later on:

[254]:
```
town_pop_corr = copy.deepcopy(town_pop_PLZ_BFS)
town_pop_corr["PLZ_check"] = False # check if PLZ is unique in example
town_pop_corr["pop_BFS_real"] = 0 # corrected population number
town_pop_corr["PLZ_to_BFS_factor"] = 0 # factor to calculate
```

Now, a huge loop is created afterwards to find the calculation factors for PLZ having more than one BFS-Nr present. The steps are described directly in the code:

[ ]:
```
for i in (town_pop_PLZ_BFS["PLZ"].unique()):
# for i in range(2882, 2883):

  # STEP 1
  # write all entries for one specific PLZ in a separate file
  PLZ_town_1 = town_pop_PLZ_BFS[town_pop_PLZ_BFS["PLZ"]==i] # write all entries␣
↪for one specific PLZ in a separate file


  # STEP 2
  # iterate through all different BFS-Nr's belonging to this PLZ
```

```
# it can be that further PLZ are appearing afterwards, belonging to the same␣
↪"cluster"
# these "new" entries should be added to the PLZ-town table
for j in town_pop_PLZ_BFS[town_pop_PLZ_BFS["PLZ"]==i]["BFS-Nr"]:
  PLZ_town_1 = PLZ_town_1.
↪append(town_pop_PLZ_BFS[town_pop_PLZ_BFS["BFS-Nr"]==j])
PLZ_town_2 = PLZ_town_1
PLZ_town_2.drop_duplicates(inplace=True) # delete duplicate rows!


# STEP 3
# now, possible new PLZ can appear, repeat step 1 and 2 to identify all␣
↪connecting PLZ / BFS
# for this, 2 new loops are necessary
for k in PLZ_town_2["PLZ"].unique():
  for l in town_pop_PLZ_BFS[town_pop_PLZ_BFS["PLZ"]==k]["BFS-Nr"]: # iterate␣
↪through these entries:
    PLZ_town_2 = PLZ_town_2.
↪append(town_pop_PLZ_BFS[town_pop_PLZ_BFS["BFS-Nr"]==l]) # for each entry,␣
↪search for all possible BFS-Nr. and append file
PLZ_town_3 = PLZ_town_2
PLZ_town_3.drop_duplicates(inplace=True) # delete duplicate rows
# print(PLZ_town_3)


# STEP 4
# in theory, this process can continue more and more, as new PLZ's and BFS␣
↪can be added to the cluster
# It is assumed, that after step 3, most clusters are found completely.
# instead of continuing the same process over and over, a check function␣
↪comes to play
# if a further PLZ is found with the newly added BFS, then print an error␣
↪message

PLZ_check = PLZ_town_3["PLZ"].to_list()
for m in PLZ_town_3["BFS-Nr"].unique():
  for n in town_pop_PLZ_BFS[town_pop_PLZ_BFS["BFS-Nr"]==m]["PLZ"]:
    PLZ_check.append(n)
PLZ_town_3["PLZ_check"] = len(set(PLZ_check)) == len(set(PLZ_town_3["PLZ"]))
# print(PLZ_town_3)

 # STEP 5
 # if PLZ check is true, then cluster is complete
 # If BFS is unique in cluster, then population number is equal to PLZ number
 # Create unique (single BFS-Nr's) and duplicate (more BFS-Nr's present) index
```

```
uniq_ind = PLZ_town_3["BFS-Nr"].duplicated(keep=False)==False # unique index
dupl_ind = PLZ_town_3["BFS-Nr"].duplicated(keep=False) # duplicate index

# create new column "pop_BFS_real" for the distributed population number per␣
↪bfs
PLZ_town_3["pop_BFS_real"]=0
# All entries with single BFS-Nr have the same pop_BFS_real value as the␣
↪pop_BFS value

PLZ_town_3["pop_BFS_real"][uniq_ind] = PLZ_town_3["pop_BFS"][uniq_ind]
# print(PLZ_town_3)


# # STEP 6
# # Per PLZ, the rest of the pop_PLZ must be distributed to the remaining␣
↪municipalities

# for PLZ in PLZ_town_3["PLZ"].unique():
#   PLZ_town_4 = copy.deepcopy(PLZ_town_3[PLZ_town_3["PLZ"]==PLZ]) # first␣
↪make small copy
#   pop_rest = (np.max(PLZ_town_4["pop_PLZ"]) -       # subtract from PLZ␣
↪population ... (all equal, instead of max could also mean or min be used)
#               np.max(PLZ_town_4["pop_BFS_real"]))   # ... the already␣
↪distributed population numbers
#   index_0 = (PLZ_town_4["pop_BFS_real"]==0)         # index, where␣
↪population is still 0
#   if sum(index_0) != 0:                             # if there is an entry␣
↪without population:
#     PLZ_town_4["pop_BFS_real"][index_0] = pop_rest / sum(index_0) # fill␣
↪zero values. If more than one empty value present, rest population must be␣
↪divided
#   # print(PLZ_town_4)
#   PLZ_town_3 = PLZ_town_3.append(PLZ_town_4)        # write back to␣
↪PLZ_town_3 table
# # print(PLZ_town_3)

  # STEP 6
# Per PLZ, the unique occurrences can be filled up with the rest population

for PLZ in PLZ_town_3["PLZ"].unique():
  PLZ_town_4 = copy.deepcopy(PLZ_town_3[PLZ_town_3["PLZ"]==PLZ]) # first make␣
↪small copy
  if len(PLZ_town_4)==1:                              # IF PLZ IS ONLY␣
↪OCCURRING ONCE! (Otherwise, division could lead to some fatal errors.)
    pop_rest = (np.max(PLZ_town_4["pop_PLZ"]) -       # subtract from PLZ␣
↪population ... (all equal, instead of max could also mean or min be used)
```

```
                np.max(PLZ_town_4["pop_BFS_real"]))   # ... the already␣
↪distributed population numbers
        index_0 = (PLZ_town_4["pop_BFS_real"]==0)          # index, where␣
↪population is still 0
        if sum(index_0) != 0:                             # if there is an entry␣
↪without population:
            PLZ_town_4["pop_BFS_real"][index_0] = pop_rest / sum(index_0) # fill␣
↪zero values. If more than one empty value present, rest population must be␣
↪divided
        # print(PLZ_town_4)
    PLZ_town_3 = PLZ_town_3.append(PLZ_town_4)         # write back to␣
↪PLZ_town_3 table
# print(PLZ_town_3)

# # STEP 7
# # Duplicates must be removed here

PLZ_town_5 = copy.deepcopy(PLZ_town_3.
↪drop_duplicates(subset=["PLZ","BFS-Nr"], keep='last'))
PLZ_town_5
# print(PLZ_town_5)

# STEP 8
# Now, fill all the remaining 0 values in the population for multiple␣
↪occurrences of PLZ in cluster
for BFS in PLZ_town_5["BFS-Nr"].unique():
    PLZ_town_6 = copy.deepcopy(PLZ_town_5[PLZ_town_5["BFS-Nr"]==BFS]) # first␣
↪make small copy of all with same BFS-Nr
    pop_sum = np.sum(PLZ_town_6["pop_BFS_real"])        # sum up all population␣
↪numbers which are already calculated
    index_0_2 = (PLZ_town_6["pop_BFS_real"]==0)          # index, where␣
↪population is still 0
    if sum(index_0_2) != 0:                               # if there is an entry␣
↪without population:
        # print(PLZ_town_6[index_0_2])
        PLZ_town_6["pop_BFS_real"][index_0_2] = (PLZ_town_6["pop_BFS"][index_0_2]␣
↪- pop_sum) / sum(index_0_2) # fill zero values. If more than one empty value␣
↪present, rest population must be divided
        # print(PLZ_town_6)
    PLZ_town_5 = PLZ_town_5.append(PLZ_town_6)         # write back to␣
↪PLZ_town_5 table
 # print(PLZ_town_5)

# # STEP 9
# # Duplicates must be removed once more
```

```
 PLZ_town_7 = copy.deepcopy(PLZ_town_5.
↪drop_duplicates(subset=["PLZ","BFS-Nr"], keep='last'))
 # print(PLZ_town_7)

 # STEP 8
 # Create distribution factor from PLZ to BFS!

 PLZ_town_7["PLZ_to_BFS_factor"] = PLZ_town_7["pop_BFS_real"] /␣
↪PLZ_town_7["pop_PLZ"]
 # print(PLZ_town_7)


 # STEP 9
 # Append this list to the created copy and remove duplicate afterwards:
 town_pop_corr = town_pop_corr.append(PLZ_town_7)
 town_pop_corr.drop_duplicates(subset=["PLZ","BFS-Nr"], keep='last',␣
↪inplace=True)
 # print(PLZ_town_7)
town_pop_corr.reset_index(inplace=True, drop=True)
```

### 6.4.3 NA handling

[256]:
```
town_pop_corr[town_pop_corr.isna().any(axis=1)]
```

[256]:

| | PLZ | BFS-Nr | Gemeindename | Kantonskürzel | Sprache | pop_PLZ | pop_BFS | \ |
|---|---|---|---|---|---|---|---|---|
| 12 | 1015 | 5586 | Lausanne | VD | fr | NaN | 140202.0 | |
| 100 | 1143 | 5656 | Hautemorges | VD | fr | 1455.0 | NaN | |
| 101 | 1116 | 5656 | Hautemorges | VD | fr | 493.0 | NaN | |
| 102 | 1128 | 5656 | Hautemorges | VD | fr | 413.0 | NaN | |
| 103 | 1136 | 5656 | Hautemorges | VD | fr | 404.0 | NaN | |
| … | … | … | … | … | … | … | … | |
| 3380 | 9497 | 7004 | Triesenberg | LI | de | NaN | NaN | |
| 3381 | 9498 | 7006 | Planken | LI | de | NaN | NaN | |
| 3454 | 9999 | 9073 | Thunersee | BE | de | NaN | NaN | |
| 3455 | 9999 | 9089 | Brienzersee | BE | de | NaN | NaN | |
| 3456 | 9999 | 9149 | Bielersee (BE) | BE | de | NaN | NaN | |

| | PLZ_check | pop_BFS_real | PLZ_to_BFS_factor |
|---|---|---|---|
| 12 | True | NaN | NaN |
| 100 | True | 1455.0 | 1.0 |
| 101 | True | 493.0 | 1.0 |
| 102 | True | 413.0 | 1.0 |
| 103 | True | 404.0 | 1.0 |
| … | … | … | … |
| 3380 | True | NaN | NaN |

```
3381        True            NaN                 NaN
3454        True            NaN                 NaN
3455        True            NaN                 NaN
3456        True            NaN                 NaN

[65 rows x 10 columns]
```

Now many NaN-values are present, which is especially a problem for the factor column and the pop_BFS_real column, which are used later on:

[257]:
```
town_pop_corr[town_pop_corr["pop_BFS_real"].isna()]
```

[257]:
```
       PLZ  BFS-Nr                      Gemeindename Kantonskürzel Sprache  \
12    1015    5586                          Lausanne            VD      fr
525   1724    2238                     Bois-d'Amont            FR      fr
664   1933    6037                    Val de Bagnes            VS      fr
1458  3801    6058                      Fieschertal            VS      de
1479  4031    2701                            Basel            BS      de
1713  4716    2430         Welschenrohr-Gänsbrunnen            SO      de
2169  6441    1215                       Seelisberg            UR      de
2214  6549    3834                    Roveredo (GR)            GR      it
2354  6809    5391  Comunanza Cadenazzo/Monteceneri            TI      it
2376  6867    5160                  Brusino Arsizio            TI      it
2638  7433    3715               Muntogna da Schons            GR      rm
3369  9487    7009                          Gamprin            LI      de
3370  9488    7011                     Schellenberg            LI      de
3371  9490    7001                            Vaduz            LI      de
3372  9491    7010                          Ruggell            LI      de
3373  9492    7007                           Eschen            LI      de
3374  9485    7007                           Eschen            LI      de
3375  9493    7008                           Mauren            LI      de
3376  9486    7008                           Mauren            LI      de
3377  9494    7005                           Schaan            LI      de
3378  9495    7002                          Triesen            LI      de
3379  9496    7003                          Balzers            LI      de
3380  9497    7004                      Triesenberg            LI      de
3381  9498    7006                          Planken            LI      de
3454  9999    9073                        Thunersee            BE      de
3455  9999    9089                      Brienzersee            BE      de
3456  9999    9149                    Bielersee (BE)            BE      de

      pop_PLZ   pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
12        NaN  140202.0       True           NaN                NaN
525    3593.0       NaN       True           NaN                NaN
664    1128.0       NaN       True           NaN                NaN
1458      NaN     326.0       True           NaN                NaN
1479      NaN  173863.0       True           NaN                NaN
```

| | | | | | |
|---|---|---|---|---|---|
| 1713 | 1176.0 | NaN | True | NaN | NaN |
| 2169 | NaN | 688.0 | True | NaN | NaN |
| 2214 | NaN | 2597.0 | True | NaN | NaN |
| 2354 | 359.0 | NaN | True | NaN | NaN |
| 2376 | NaN | 451.0 | True | NaN | NaN |
| 2638 | 363.0 | NaN | True | NaN | NaN |
| 3369 | NaN | NaN | True | NaN | NaN |
| 3370 | NaN | NaN | True | NaN | NaN |
| 3371 | NaN | NaN | True | NaN | NaN |
| 3372 | NaN | NaN | True | NaN | NaN |
| 3373 | NaN | NaN | True | NaN | NaN |
| 3374 | NaN | NaN | True | NaN | NaN |
| 3375 | NaN | NaN | True | NaN | NaN |
| 3376 | NaN | NaN | True | NaN | NaN |
| 3377 | NaN | NaN | True | NaN | NaN |
| 3378 | NaN | NaN | True | NaN | NaN |
| 3379 | NaN | NaN | True | NaN | NaN |
| 3380 | NaN | NaN | True | NaN | NaN |
| 3381 | NaN | NaN | True | NaN | NaN |
| 3454 | NaN | NaN | True | NaN | NaN |
| 3455 | NaN | NaN | True | NaN | NaN |
| 3456 | NaN | NaN | True | NaN | NaN |

Most NaN values are not surprising and come due to differences in the structure between the two population tables. The communities from Liechteinstein (BFS-Nr. of 70xx) are only existent in the town directory, but I don't use them and therefore, these entries can be deleted. The same is valid for the three entries of Thuner-, Brienzer- and Bielersee. All these entries do neither show a population number for the PLZ nor for the BFS-Nr. These places must be deleted from the calculation:

[258]:
```
town_pop_corr.dropna(subset=['pop_PLZ', 'pop_BFS'], how='all', inplace=True)
town_pop_corr.reset_index(inplace=True, drop=True)
town_pop_corr[town_pop_corr["pop_BFS_real"].isna()]
```

[258]:

| | PLZ | BFS-Nr | Gemeindename | Kantonskürzel | Sprache | \ |
|---|---|---|---|---|---|---|
| 12 | 1015 | 5586 | Lausanne | VD | fr | |
| 525 | 1724 | 2238 | Bois-d'Amont | FR | fr | |
| 664 | 1933 | 6037 | Val de Bagnes | VS | fr | |
| 1458 | 3801 | 6058 | Fieschertal | VS | de | |
| 1479 | 4031 | 2701 | Basel | BS | de | |
| 1713 | 4716 | 2430 | Welschenrohr-Gänsbrunnen | SO | de | |
| 2169 | 6441 | 1215 | Seelisberg | UR | de | |
| 2214 | 6549 | 3834 | Roveredo (GR) | GR | it | |
| 2354 | 6809 | 5391 | Comunanza Cadenazzo/Monteceneri | TI | it | |
| 2376 | 6867 | 5160 | Brusino Arsizio | TI | it | |
| 2638 | 7433 | 3715 | Muntogna da Schons | GR | rm | |

```
      pop_PLZ   pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
12        NaN  140202.0       True           NaN                NaN
525    3593.0       NaN       True           NaN                NaN
664    1128.0       NaN       True           NaN                NaN
1458      NaN     326.0       True           NaN                NaN
1479      NaN  173863.0       True           NaN                NaN
1713   1176.0       NaN       True           NaN                NaN
2169      NaN     688.0       True           NaN                NaN
2214      NaN    2597.0       True           NaN                NaN
2354    359.0       NaN       True           NaN                NaN
2376      NaN     451.0       True           NaN                NaN
2638    363.0       NaN       True           NaN                NaN
```

Also some other entries with non-existent PLZ population numbers are not that problematic, because we can just assume, that the factor must be 1, as no PLZ is occurrent twice!

As for "Bois d'Amont", "Val de Bagnes", "Comunanza Cadenazzo/Monteceneri", "Muntogna da Schons" and "Welchenrohr-Gänsbrunnen", the BFS Nr is not found in the population table. These municipalities were created through fusions in 2021, what makes the reason for this circumstance.

We have to check the PLZ in these cases, as multiple occurrences can be there:

[259]:
```
town_pop_corr[town_pop_corr["PLZ"]==1724]
```

[259]:
```
      PLZ  BFS-Nr  Gemeindename Kantonskürzel Sprache  pop_PLZ  pop_BFS  \
525  1724    2238  Bois-d'Amont            FR      fr   3593.0      NaN
527  1724    2194     Ferpicloz            FR      fr   3593.0    267.0
528  1724    2220     Le Mouret            FR      fr   3593.0   3148.0

    PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
525      True           NaN                NaN
527      True         267.0           0.074311
528      True        3148.0           0.876148
```

This PLZ us used by 3 different municipalities. Therefore, the factor must be 1 - the already used factors:

[260]:
```
town_pop_corr.at[525, "PLZ_to_BFS_factor"] = 1 - (town_pop_corr.
↪iloc[527]["PLZ_to_BFS_factor"] + town_pop_corr.
↪iloc[528]["PLZ_to_BFS_factor"])
```

[261]:
```
town_pop_corr[town_pop_corr["PLZ"]==1933]
```

[261]:
```
      PLZ  BFS-Nr   Gemeindename Kantonskürzel Sprache  pop_PLZ  pop_BFS  \
664  1933    6037  Val de Bagnes            VS      fr   1128.0      NaN
670  1933    6035    Sembrancher            VS      fr   1128.0   1050.0

    PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
664      True           NaN                NaN
```

```
670       True        1050.0            0.930851
```

[262]:
```
town_pop_corr.at[664, "PLZ_to_BFS_factor"] = 1 - town_pop_corr.
 ↪iloc[670]["PLZ_to_BFS_factor"]
```

[263]:
```
town_pop_corr[town_pop_corr["PLZ"]==4716]
```

[263]:
```
       PLZ  BFS-Nr              Gemeindename Kantonskürzel Sprache  pop_PLZ  \
1713  4716    2430  Welschenrohr-Gänsbrunnen            SO      de   1176.0

      pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
1713      NaN       True           NaN                NaN
```

[264]:
```
town_pop_corr.at[1713, "PLZ_to_BFS_factor"] = 1
```

[265]:
```
town_pop_corr[town_pop_corr["PLZ"]==6809]
```

[265]:
```
       PLZ  BFS-Nr                     Gemeindename Kantonskürzel Sprache  \
2349  6809    5238                      Monteceneri            TI      it
2354  6809    5391  Comunanza Cadenazzo/Monteceneri            TI      it

      pop_PLZ  pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
2349    359.0   4535.0       True         359.0                1.0
2354    359.0      NaN       True           NaN                NaN
```

[266]:
```
town_pop_corr.at[2354, "PLZ_to_BFS_factor"] = 1 - town_pop_corr.
 ↪iloc[2349]["PLZ_to_BFS_factor"]
```

[267]:
```
town_pop_corr[town_pop_corr["PLZ"]==7433]
```

[267]:
```
       PLZ  BFS-Nr        Gemeindename Kantonskürzel Sprache  pop_PLZ  \
2638  7433    3715  Muntogna da Schons            GR      rm    363.0

      pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
2638      NaN       True           NaN                NaN
```

[268]:
```
town_pop_corr.at[2638, "PLZ_to_BFS_factor"] = 1
```

[269]:
```
town_pop_corr.fillna(value = {"PLZ_to_BFS_factor":1}, inplace=True)
```

[270]:
```
town_pop_corr[town_pop_corr["pop_BFS_real"].isna()]
```

[270]:
```
       PLZ  BFS-Nr                     Gemeindename Kantonskürzel Sprache  \
12    1015    5586                        Lausanne            VD      fr
525   1724    2238                    Bois-d'Amont            FR      fr
664   1933    6037                   Val de Bagnes            VS      fr
1458  3801    6058                      Fieschertal            VS      de
```

```
1479  4031    2701                            Basel            BS       de
1713  4716    2430          Welschenrohr-Gänsbrunnen            SO       de
2169  6441    1215                        Seelisberg            UR       de
2214  6549    3834                     Roveredo (GR)            GR       it
2354  6809    5391  Comunanza Cadenazzo/Monteceneri            TI       it
2376  6867    5160                   Brusino Arsizio            TI       it
2638  7433    3715                Muntogna da Schons            GR       rm

      pop_PLZ   pop_BFS  PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
12        NaN  140202.0       True           NaN           1.000000
525    3593.0       NaN       True           NaN           0.049541
664    1128.0       NaN       True           NaN           0.069149
1458      NaN     326.0       True           NaN           1.000000
1479      NaN  173863.0       True           NaN           1.000000
1713   1176.0       NaN       True           NaN           1.000000
2169      NaN     688.0       True           NaN           1.000000
2214      NaN    2597.0       True           NaN           1.000000
2354    359.0       NaN       True           NaN           0.000000
2376      NaN     451.0       True           NaN           1.000000
2638    363.0       NaN       True           NaN           1.000000
```

The missing pop_BFS_real values should then taken by the "pop_PLZ" value. The "BFS_Nr" value, if it is present, often is already distributed to the different PLZ's, so this should be avoided here.

[271]:
```python
town_pop_corr["pop_BFS_real"].fillna(town_pop_corr["pop_PLZ"], inplace=True) #␣
↪if PLZ value is present
# town_pop_corr["pop_BFS_real"].fillna(town_pop_corr["pop_BFS"], inplace=True)␣
↪# if BFS value is present
```

[272]:
```python
town_pop_corr[town_pop_corr["pop_BFS_real"].isna()]
```

[272]:
```
       PLZ  BFS-Nr      Gemeindename Kantonskürzel Sprache  pop_PLZ   pop_BFS  \
12    1015    5586          Lausanne            VD      fr      NaN  140202.0
1458  3801    6058        Fieschertal            VS      de      NaN     326.0
1479  4031    2701             Basel            BS      de      NaN  173863.0
2169  6441    1215        Seelisberg            UR      de      NaN     688.0
2214  6549    3834     Roveredo (GR)            GR      it      NaN    2597.0
2376  6867    5160   Brusino Arsizio            TI      it      NaN     451.0

      PLZ_check  pop_BFS_real  PLZ_to_BFS_factor
12         True           NaN                1.0
1458       True           NaN                1.0
1479       True           NaN                1.0
2169       True           NaN                1.0
2214       True           NaN                1.0
2376       True           NaN                1.0
```

The last 6 entries are left with NaN. Possibly, there are no values to join in these PLZ's, and if there are still values, each case must be looked at independently.

### 6.4.4 Renaming and saving

[273]:
```
town_pop_corr.rename(columns={"BFS-Nr":"BFS_Nr", "Gemeindename":"municipality",
                              "Kantonskürzel":"canton", "Sprache":"language",
                              }, inplace=True)
```

[274]:
```
town_pop_corr[town_pop_corr["PLZ"]==2882]
```

[274]:
```
      PLZ  BFS_Nr  municipality canton language  pop_PLZ  pop_BFS  PLZ_check  \
913  2882    6808  Clos du Doubs     JU       fr    685.0   1263.0       True
919  2882    6758    Saint-Brais     JU       fr    685.0    227.0       True

     pop_BFS_real  PLZ_to_BFS_factor
913         670.0           0.978102
919           8.0           0.011679
```

[275]:
```
town_pop_corr[town_pop_corr["BFS_Nr"]==6808]
```

[275]:
```
      PLZ  BFS_Nr  municipality canton language  pop_PLZ  pop_BFS  PLZ_check  \
912  2889    6808  Clos du Doubs     JU       fr    125.0   1263.0       True
913  2882    6808  Clos du Doubs     JU       fr    685.0   1263.0       True
914  2883    6808  Clos du Doubs     JU       fr     98.0   1263.0       True
915  2884    6808  Clos du Doubs     JU       fr     87.0   1263.0       True
916  2885    6808  Clos du Doubs     JU       fr    158.0   1263.0       True
917  2886    6808  Clos du Doubs     JU       fr     77.0   1263.0       True
918  2888    6808  Clos du Doubs     JU       fr     48.0   1263.0       True

     pop_BFS_real  PLZ_to_BFS_factor
912         125.0           1.000000
913         670.0           0.978102
914          98.0           1.000000
915          87.0           1.000000
916         158.0           1.000000
917          77.0           1.000000
918          48.0           1.000000
```

[276]:
```
town_pop_final = town_pop_corr.drop(columns=["pop_PLZ", "pop_BFS", "PLZ_check"])
```

[277]:
```
town_pop_final.sort_values(axis=0, by="PLZ", inplace=True)
```

### 6.4.5 Writing csv

[278]:
```
town_pop_final.to_csv("../Data/2_Joined_entities/PLZ_to_BFS_factor.csv",␣
↪index=False)
```

# 7 Joining on PLZ level + aggregating on BFS level

In case of the travelcards dataset and the population_marital, the data is available on the level of the PLZ. This has to be brought to the level of the municipality. To to this, we need the prepared town_directory dataset with the defined factors to deal with the aggregation problem, as well as the two datasets mentioned above

## 7.1 Loading datasets

[279]:
```
join_base = pd.read_csv("../Data/2_Joined_entities/PLZ_to_BFS_factor.csv")
join_base
```

[279]:
```
       PLZ  BFS_Nr            municipality canton language  pop_BFS_real  \
0     1000    5586                Lausanne     VD       fr        3991.0
1     1003    5586                Lausanne     VD       fr        6528.0
2     1004    5586                Lausanne     VD       fr       31084.0
3     1005    5586                Lausanne     VD       fr       12465.0
4     1006    5586                Lausanne     VD       fr       15520.0
…      …       …                       …      …        …            …
3436  9652    3360                 Nesslau     SG       de         699.0
3437  9655    3360                 Nesslau     SG       de         342.0
3438  9656    3359  Wildhaus-Alt St. Johann     SG       de         638.0
3439  9657    3359  Wildhaus-Alt St. Johann     SG       de         714.0
3440  9658    3359  Wildhaus-Alt St. Johann     SG       de        1272.0

      PLZ_to_BFS_factor
0                   1.0
1                   1.0
2                   1.0
3                   1.0
4                   1.0
…                     …
3436                1.0
3437                1.0
3438                1.0
3439                1.0
3440                1.0

[3441 rows x 7 columns]
```

[280]:
```
pop = pd.read_csv("../Data/1_Cleaned/population_marital.csv")
pop
```

[280]:
```
       PLZ  pop_count  single_count  married_count  widowed_count  \
0     1000     3991.0        2378.0         1314.0           81.0
1     1003     6528.0        4102.0         1687.0          178.0
2     1004    31084.0       17357.0         9411.0         1261.0
3     1005    12465.0        7397.0         3549.0          397.0
4     1006    15520.0        8725.0         4700.0          616.0
…      …         …           …             …              …
3177  9652      699.0         293.0          320.0           36.0
3178  9655      342.0         144.0          149.0           21.0
3179  9656      638.0         286.0          270.0           33.0
3180  9657      714.0         293.0          313.0           40.0
3181  9658     1272.0         522.0          550.0           88.0

      divorced_count
0              218.0
1              561.0
2             3053.0
3             1121.0
4             1479.0
…               …
3177            50.0
3178            28.0
3179            49.0
3180            68.0
3181           112.0

[3182 rows x 6 columns]
```

[281]:
```
tr_cards = pd.read_csv("../Data/1_Cleaned/travelcards.csv")
tr_cards
```

[281]:
```
       PLZ      GA      HTA  fn_tck
0     1000    75.0   1258.0   716.0
1     1003   677.0   3449.0   772.0
2     1004  1653.0  10657.0  4383.0
3     1005   825.0   5237.0  1796.0
4     1006  1217.0   6811.0  2355.0
…      …      …       …       …
3286  9495     0.0      0.0    20.0
3287  9496     0.0      0.0    16.0
3288  9497     0.0      0.0     5.0
3289  9572     0.0      0.0     5.0
3290  9721     0.0      0.0     5.0
```

```
[3291 rows x 4 columns]
```

## 7.2 Joining population data

[282]:
```python
pop_join = join_base.merge(pop, how = "left", on = "PLZ")
pop_join[6:9]
```

[282]:
```
    PLZ  BFS_Nr     municipality canton language  pop_BFS_real  \
6  1008    5585  Jouxtens-Mézery     VD       fr        1412.0
7  1008    5589           Prilly     VD       fr       12360.0
8  1009    5590            Pully     VD       fr       18568.0

   PLZ_to_BFS_factor  pop_count  single_count  married_count  widowed_count  \
6           0.102654    13755.0        6537.0         5297.0          633.0
7           0.898582    13755.0        6537.0         5297.0          633.0
8           1.000000    18568.0        8364.0         7398.0          999.0

   divorced_count
6          1282.0
7          1282.0
8          1807.0
```

All the count numbers should now be multiplied with the defined "PLZ_to_BFS_factor" to get the real numbers per BFS (or part of the BFS-Nr which belongs to the specific PLZ)

[283]:
```python
pop_join["pop_count_BFS"] = pop_join["pop_count"] *␣
 ↪pop_join["PLZ_to_BFS_factor"]
pop_join["single_count_BFS"] = pop_join["single_count"] *␣
 ↪pop_join["PLZ_to_BFS_factor"]
pop_join["married_count_BFS"] = pop_join["married_count"] *␣
 ↪pop_join["PLZ_to_BFS_factor"]
pop_join["widowed_count_BFS"] = pop_join["widowed_count"] *␣
 ↪pop_join["PLZ_to_BFS_factor"]
pop_join["divorced_count_BFS"] = pop_join["divorced_count"] *␣
 ↪pop_join["PLZ_to_BFS_factor"]
pop_join[6:9]
```

[283]:
```
    PLZ  BFS_Nr     municipality canton language  pop_BFS_real  \
6  1008    5585  Jouxtens-Mézery     VD       fr        1412.0
7  1008    5589           Prilly     VD       fr       12360.0
8  1009    5590            Pully     VD       fr       18568.0

   PLZ_to_BFS_factor  pop_count  single_count  married_count  widowed_count  \
6           0.102654    13755.0        6537.0         5297.0          633.0
7           0.898582    13755.0        6537.0         5297.0          633.0
8           1.000000    18568.0        8364.0         7398.0          999.0
```

```
   divorced_count  pop_count_BFS  single_count_BFS  married_count_BFS  \
6          1282.0         1412.0        671.046456         543.756016
7          1282.0        12360.0       5874.032715        4759.790622
8          1807.0        18568.0       8364.000000        7398.000000

   widowed_count_BFS  divorced_count_BFS
6          64.979716          131.601890
7         568.802617         1151.982552
8         999.000000         1807.000000
```

2 things can be observed: 1. The calculated pop_count_BFS gets the same population number as the pop_BFS_real column. This is as expected, but works here like a control function if everything works as expected. One of the columns can be deleted. 2. All the old count columns can now be deleted

[284]:
```
pop_join.drop(["pop_count", "single_count", "married_count", "widowed_count",␣
 ↪"divorced_count", "pop_BFS_real"], axis=1, inplace=True)
pop_join[:3]
```

[284]:
```
    PLZ  BFS_Nr municipality canton language  PLZ_to_BFS_factor  \
0  1000    5586     Lausanne     VD       fr                1.0
1  1003    5586     Lausanne     VD       fr                1.0
2  1004    5586     Lausanne     VD       fr                1.0

   pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
0         3991.0            2378.0             1314.0               81.0
1         6528.0            4102.0             1687.0              178.0
2        31084.0           17357.0             9411.0             1261.0

   divorced_count_BFS
0               218.0
1               561.0
2              3053.0
```

### 7.3 Joining travelcards data

[285]:
```
tr_cards_pop_join = pop_join.merge(tr_cards, how = "left", on = "PLZ")
tr_cards_pop_join[6:9]
```

[285]:
```
    PLZ  BFS_Nr     municipality canton language  PLZ_to_BFS_factor  \
6  1008    5585  Jouxtens-Mézery     VD       fr           0.102654
7  1008    5589          Prilly     VD       fr           0.898582
8  1009    5590           Pully     VD       fr           1.000000

   pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
```

```
6       1412.0     671.046456       543.756016        64.979716
7      12360.0    5874.032715      4759.790622       568.802617
8      18568.0    8364.000000      7398.000000       999.000000

   divorced_count_BFS     GA     HTA  fn_tck
6          131.601890  343.0  3129.0  1177.0
7         1151.982552  343.0  3129.0  1177.0
8         1807.000000  781.0  7323.0  2265.0
```

Also here, the three columns "GA", "HTA" and "fn_tck" must be multiplied with the defined factor.

[286]:
```python
tr_cards_pop_join["GA_BFS"] = tr_cards_pop_join["GA"] *␣
 ↪tr_cards_pop_join["PLZ_to_BFS_factor"]
tr_cards_pop_join["HTA_BFS"] = tr_cards_pop_join["HTA"] *␣
 ↪tr_cards_pop_join["PLZ_to_BFS_factor"]
tr_cards_pop_join["fn_tck_BFS"] = tr_cards_pop_join["fn_tck"] *␣
 ↪tr_cards_pop_join["PLZ_to_BFS_factor"]
tr_cards_pop_join[6:9]
```

[286]:
```
    PLZ  BFS_Nr     municipality canton language  PLZ_to_BFS_factor  \
6  1008    5585  Jouxtens-Mézery     VD       fr           0.102654
7  1008    5589           Prilly     VD       fr           0.898582
8  1009    5590            Pully     VD       fr           1.000000

   pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
6         1412.0        671.046456         543.756016          64.979716
7        12360.0       5874.032715        4759.790622         568.802617
8        18568.0       8364.000000        7398.000000         999.000000

   divorced_count_BFS     GA     HTA  fn_tck      GA_BFS      HTA_BFS  \
6          131.601890  343.0  3129.0  1177.0   35.210178   321.203053
7         1151.982552  343.0  3129.0  1177.0  308.213740  2811.664122
8         1807.000000  781.0  7323.0  2265.0  781.000000  7323.000000

    fn_tck_BFS
6   120.823264
7  1057.631407
8  2265.000000
```

All the old columns can be deleted now

[287]:
```python
tr_cards_pop_join.drop(["GA", "HTA", "fn_tck"], axis=1, inplace=True)
tr_cards_pop_join[6:9]
```

[287]:
```
    PLZ  BFS_Nr     municipality canton language  PLZ_to_BFS_factor  \
6  1008    5585  Jouxtens-Mézery     VD       fr           0.102654
7  1008    5589           Prilly     VD       fr           0.898582
```

```
8  1009   5590          Pully    VD      fr          1.000000

   pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
6         1412.0        671.046456         543.756016          64.979716
7        12360.0       5874.032715        4759.790622         568.802617
8        18568.0       8364.000000        7398.000000         999.000000

   divorced_count_BFS      GA_BFS      HTA_BFS   fn_tck_BFS
6          131.601890   35.210178   321.203053   120.823264
7         1151.982552  308.213740  2811.664122  1057.631407
8         1807.000000  781.000000  7323.000000  2265.000000
```

Now the data has been loaded and can be aggregated to the BFS level finally!

### 7.4 Aggregating to BFS level

[288]:
```python
bfs_base = tr_cards_pop_join.groupby(["BFS_Nr", "municipality", "canton",␣
 ↪"language"]).sum().reset_index()
bfs_base
```

[288]:
```
      BFS_Nr        municipality canton language    PLZ  PLZ_to_BFS_factor  \
0          1      Aeugst am Albis     ZH       de   8914           1.000000
1          2    Affoltern am Albis    ZH       de  17819           2.000000
2          3          Bonstetten      ZH       de   8906           1.000000
3          4      Hausen am Albis     ZH       de  17840           2.000000
4          5            Hedingen      ZH       de   8908           1.000000
…        …                  …   …       …   …                 …
2145    6806        Vendlincourt      JU       fr   2943           1.000000
2146    6807        Basse-Allaine     JU       fr   8772           3.000000
2147    6808        Clos du Doubs     JU       fr  20197           6.978102
2148    6809         Haute-Ajoie      JU       fr  11639           4.000000
2149    6810          La Baroche      JU       fr  11800           4.000000

      pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
0            2014.0        835.000000         923.000000          74.000000
1           12289.0       5312.000000        5311.000000         586.000000
2            5610.0       2435.000000        2577.000000         175.000000
3            3781.0       1603.000000        1683.000000         136.000000
4            3795.0       1618.000000        1729.000000         115.000000
…               …               …                …                …
2145          560.0        220.000000         255.000000          40.000000
2146         1235.0        497.000000         518.000000         103.000000
2147         1263.0        542.518248         513.459854          78.992701
2148         1096.0        443.000000         483.000000          71.000000
2149         1129.0        480.000000         458.000000          83.000000
```

```
      divorced_count_BFS      GA_BFS      HTA_BFS  fn_tck_BFS
0             182.000000  104.000000   846.000000  189.000000
1            1080.000000  656.000000  4359.000000  772.000000
2             423.000000  303.000000  2555.000000  647.000000
3             359.000000  248.000000  1470.000000  133.000000
4             333.000000  310.000000  1688.000000  448.000000
…                    …          …            …         …
2145           45.000000    8.740000    98.000000   26.000000
2146          117.000000   39.480000   171.000000   90.000000
2147          128.029197   68.152555   215.156204   65.474453
2148           99.000000   26.220000   189.480000   79.000000
2149          108.000000   34.960000   174.000000  109.000000

[2150 rows x 14 columns]
```

[289]:
```python
bfs_base[bfs_base["BFS_Nr"].duplicated(keep=False)]
```

[289]:
```
      BFS_Nr       municipality canton language    PLZ  PLZ_to_BFS_factor  \
1130    3661              Cazis     GR       de   7421                1.0
1131    3661              Cazis     GR       rm  29677                4.0
1212    3988  Obersaxen Mundaun     GR       de   7134                1.0
1213    3988  Obersaxen Mundaun     GR       rm  14275                2.0

      pop_count_BFS  single_count_BFS  married_count_BFS  widowed_count_BFS  \
1130          454.0             194.0              209.0               18.0
1131         1839.0             772.0              799.0               82.0
1212          801.0             316.0              375.0               59.0
1213          363.0             137.0              186.0               13.0

      divorced_count_BFS  GA_BFS  HTA_BFS  fn_tck_BFS
1130                33.0    6.77     88.0         0.0
1131               186.0   51.31    534.0         3.0
1212                51.0   20.00    189.0         4.0
1213                27.0   13.54     94.0         0.0
```

There is a small problem, as 2 BFS_Nr occur twice due to different languages in the specific PLZ's. This can be solved manually: 1. As for Cazis, most of the population speak rumantsch, therefore I will classify "rm" as language here and sum up the numbers. 2. In Obersaxen Mundaun, the bigger part speaks German, therefore the language to be classified to is German ("de").

[290]:
```python
bfs_base.iloc[1130,4:] = bfs_base.iloc[1130,4:] + bfs_base.iloc[1131,4:] #␣
↪adding count values of both entries
bfs_base.iloc[1130, 3] = "rm"   # overwrite language with defined language
```

[291]:
```python
bfs_base.iloc[1212,4:] = bfs_base.iloc[1212,4:] + bfs_base.iloc[1213,4:] #␣
↪adding count values of both entries
bfs_base.iloc[1212, 3] = "de"   # overwrite language with defined language
```

```
[292]: bfs_base.drop([1131,1213], axis=0, inplace=True) # dropping the both other␣
        ↪columns
       # bfs_base.drop(1211, axis=0, inplace=True)
```

```
[293]: bfs_base[bfs_base["BFS_Nr"].duplicated(keep=False)]
```

```
[293]: Empty DataFrame
       Columns: [BFS_Nr, municipality, canton, language, PLZ, PLZ_to_BFS_factor,
       pop_count_BFS, single_count_BFS, married_count_BFS, widowed_count_BFS,
       divorced_count_BFS, GA_BFS, HTA_BFS, fn_tck_BFS]
       Index: []
```

No more duplicates on the level of BFS are available

Now we don't need the PLZ anymore, as it is a senseless summing up of the different PLZ numbers per municipality now. Neither the PLZ_to_BFS_factor does have any function left.

```
[294]: bfs_base.drop(columns=["PLZ", "PLZ_to_BFS_factor"], inplace=True)
       bfs_base
```

```
[294]:       BFS_Nr        municipality canton language  pop_count_BFS  \
       0          1      Aeugst am Albis     ZH       de         2014.0
       1          2   Affoltern am Albis     ZH       de        12289.0
       2          3           Bonstetten     ZH       de         5610.0
       3          4      Hausen am Albis     ZH       de         3781.0
       4          5             Hedingen     ZH       de         3795.0
       …        …                  …    …        …             …
       2145    6806         Vendlincourt     JU       fr          560.0
       2146    6807        Basse-Allaine     JU       fr         1235.0
       2147    6808        Clos du Doubs     JU       fr         1263.0
       2148    6809         Haute-Ajoie      JU       fr         1096.0
       2149    6810           La Baroche     JU       fr         1129.0

             single_count_BFS  married_count_BFS  widowed_count_BFS  \
       0           835.000000         923.000000          74.000000
       1          5312.000000        5311.000000         586.000000
       2          2435.000000        2577.000000         175.000000
       3          1603.000000        1683.000000         136.000000
       4          1618.000000        1729.000000         115.000000
       …                 …                  …                  …
       2145        220.000000         255.000000          40.000000
       2146        497.000000         518.000000         103.000000
       2147        542.518248         513.459854          78.992701
       2148        443.000000         483.000000          71.000000
       2149        480.000000         458.000000          83.000000

             divorced_count_BFS      GA_BFS     HTA_BFS  fn_tck_BFS
       0             182.000000  104.000000  846.000000  189.000000
```

```
1              1080.000000  656.000000  4359.000000  772.000000
2               423.000000  303.000000  2555.000000  647.000000
3               359.000000  248.000000  1470.000000  133.000000
4               333.000000  310.000000  1688.000000  448.000000
…                     …           …           …           …
2145             45.000000    8.740000    98.000000   26.000000
2146            117.000000   39.480000   171.000000   90.000000
2147            128.029197   68.152555   215.156204   65.474453
2148             99.000000   26.220000   189.480000   79.000000
2149            108.000000   34.960000   174.000000  109.000000

[2148 rows x 12 columns]
```

## 7.5 Calculating share values

All the values present in the dataframe now, must now be brought to shares as described in the ER model.

[295]:
```
bfs_base["single_share"] = bfs_base["single_count_BFS"] /␣
 ↪bfs_base["pop_count_BFS"]
bfs_base["married_share"] = bfs_base["married_count_BFS"] /␣
 ↪bfs_base["pop_count_BFS"]
bfs_base["widowed_share"] = bfs_base["widowed_count_BFS"] /␣
 ↪bfs_base["pop_count_BFS"]
bfs_base["divorced_share"] = bfs_base["divorced_count_BFS"] /␣
 ↪bfs_base["pop_count_BFS"]
bfs_base["GA_share"] = bfs_base["GA_BFS"] / bfs_base["pop_count_BFS"]
bfs_base["HTA_share"] = bfs_base["HTA_BFS"] / bfs_base["pop_count_BFS"]
bfs_base["FNT_share"] = bfs_base["fn_tck_BFS"] / bfs_base["pop_count_BFS"]
```

[296]:
```
bfs_base[:3]
```

[296]:
```
   BFS_Nr        municipality canton language  pop_count_BFS  \
0       1      Aeugst am Albis     ZH       de         2014.0
1       2   Affoltern am Albis     ZH       de        12289.0
2       3           Bonstetten     ZH       de         5610.0

   single_count_BFS  married_count_BFS  widowed_count_BFS  divorced_count_BFS  \
0             835.0              923.0               74.0               182.0
1            5312.0             5311.0              586.0              1080.0
2            2435.0             2577.0              175.0               423.0

   GA_BFS  HTA_BFS  fn_tck_BFS  single_share  married_share  widowed_share  \
0   104.0    846.0       189.0      0.414598       0.458292       0.036743
1   656.0   4359.0       772.0      0.432256       0.432175       0.047685
2   303.0   2555.0       647.0      0.434046       0.459358       0.031194
```

```
   divorced_share  GA_share  HTA_share  FNT_share
0        0.090367  0.051639   0.420060   0.093843
1        0.087883  0.053381   0.354707   0.062820
2        0.075401  0.054011   0.455437   0.115330
```

### 7.6 Writing csv

[297]:
```
bfs_base.to_csv("../Data/2_Joined_entities/bfs_base.csv", index=False)
```

## 8 Joining on BFS level

All other datasets with possible explanation variables are available on the level of the BFS. In the next step, I will add these to the before created dataset.

The datasets to be joined in this step are ”city_distances”, ”population_shares”, ”stop_list_cleaned”, ”cars” and ”commuter share”.

In case of the travelcards dataset and the population_marital, the data is available on the level of the PLZ. This has to be brought to the level of the municipality. To to this, we need the prepared town_directory dataset with the defined factors to deal with the aggregation problem, as well as the two datasets mentioned above

### 8.1 Loading datasets

[298]:
```
pop_shares = pd.read_csv("../Data/1_Cleaned/population_shares.csv")
pop_shares
```

[298]:
```
      BFS_Nr  pop_count   age0_20  age20_40  age40_60    age60+  age0_20cnt  \
0          1       2014  0.189672  0.187190  0.350050  0.273088         382
1          2      12289  0.201969  0.278298  0.275856  0.243877        2482
2          3       5610  0.240642  0.225312  0.308734  0.225312        1350
3          4       3801  0.220994  0.189687  0.337543  0.251776         840
4          5       3795  0.216074  0.220553  0.327009  0.236364         820
…        …          …         …         …         …         …
2193    6806        560  0.173214  0.228571  0.262500  0.335714          97
2194    6807       1241  0.216761  0.189363  0.275584  0.318292         269
2195    6808       1263  0.182106  0.229612  0.250990  0.337292         230
2196    6809       1096  0.170620  0.208029  0.250912  0.370438         187
2197    6810       1135  0.207048  0.200881  0.279295  0.312775         235

      age20_40cnt  age40_60cnt  age60+cnt  …  resid_6_10y_cnt  \
0             377          705        550  …              324
1            3420         3390       2997  …             1598
```

```
2             1264         1732        1264  …              759
3              721         1283         957  …              470
4              837         1241         897  …              533
…            …          …        …   …           …
2193           128          147         188  …               68
2194           235          342         395  …              102
2195           290          317         426  …              121
2196           228          275         406  …              100
2197           228          317         355  …              103

      resid_10+y_cnt      hh_1      hh_2    hh_3_5     hh_6+  hh_1_cnt  \
0               1076  0.306727  0.369441  0.313569  0.010262       269
1               6827  0.361575  0.341255  0.284470  0.012700      1993
2               3295  0.289775  0.338142  0.365295  0.006788       683
3               2218  0.291772  0.345570  0.345570  0.017089       461
4               2236  0.301768  0.335859  0.343434  0.018939       478
…                …       …       …       …       …       …
2193             365  0.334677  0.387097  0.250000  0.028226        83
2194             836  0.366972  0.322936  0.280734  0.029358       200
2195             879  0.403685  0.345059  0.239531  0.011725       241
2196             745  0.354902  0.372549  0.258824  0.013725       181
2197             772  0.371542  0.308300  0.296443  0.023715       188

      hh_2_cnt  hh_3_5_cnt  hh_6+_cnt
0          324         275          9
1         1881        1568         70
2          797         861         16
3          546         546         27
4          532         544         30
…        …          …         …
2193        96          62          7
2194       176         153         16
2195       206         143          7
2196       190         132          7
2197       156         150         12

[2198 rows x 38 columns]
```

[299]:
```
dist = pd.read_csv("../Data/2_Joined_entities/city_distances.csv")
dist
```

[299]:
```
      BFS_Nr  PT_dist_medium  PT_time_medium  PT_dist_big  PT_time_big  \
0          1          21.327          51.392       25.793       61.008
1          2          15.384          33.779       25.355       45.628
2          3          22.463          43.891       18.120       37.031
3          4          15.902          44.969       30.128       63.564
4          5          17.715          36.447       22.436       39.591
```

```
…     …            …            …         …          …
2170   6806        74.164        97.112    77.084     110.411
2171   6807        72.741        93.558    75.660     110.916
2172   6808        55.915        74.479    58.536      89.818
2173   6809        77.381       117.050    79.608     126.877
2174   6810        72.658        96.194    75.406     113.674

      str_dist_medium  str_time_medium  str_dist_big  str_time_big  \
0              22.158           32.677        22.288        35.522
1              17.267           22.651        21.131        27.870
2              27.129           28.739        14.706        23.281
3              11.590           23.337        23.171        37.718
4              20.315           29.129        17.598        26.014
…                 …               …             …            …
2170           64.815           62.915        46.482        72.767
2171           74.429           66.274        65.179        83.016
2172           51.676           53.887        64.969        72.605
2173           68.819           62.720        75.191        81.352
2174           58.764           59.812        44.557        66.530

      PT_fact_big  PT_fact_medium
0        1.717471        1.572727
1        1.637173        1.491281
2        1.590610        1.527228
3        1.685243        1.926940
4        1.521911        1.251227
…           …               …
2170     1.517322        1.543543
2171     1.336080        1.411685
2172     1.237077        1.382133
2173     1.559605        1.866231
2174     1.708613        1.608273

[2175 rows x 11 columns]
```

[300]:

```
stops = pd.read_csv("../Data/2_Joined_entities/stop_list_final.csv")
stops
```

[300]:

```
      BFS_Nr  bus_count  other_count  train_count  bus_stat  other_stat  \
0        1.0   210319.0          0.0          0.0       6.0         0.0
1        2.0   488680.0          0.0      51616.0      13.0         0.0
2        3.0   249494.0          0.0      51616.0       7.0         0.0
3        4.0   234267.0          0.0          0.0      10.0         0.0
4        5.0    43000.0          0.0      51616.0       2.0         0.0
…     …          …          …           …          …          …
2077  6806.0        0.0          0.0      15420.0       0.0         0.0
2078  6807.0    64218.0          0.0      34654.0       6.0         0.0
```

```
2079  6808.0   162731.0          0.0      29848.0      22.0          0.0
2080  6809.0    82398.0          0.0          0.0       8.0          0.0
2081  6810.0   225457.0          0.0          0.0      12.0          0.0

      train_stat  pop_count  bus_stops_per_pop  train_stops_per_pop  \
0            0.0       2014         104.428500             0.000000
1            1.0      12289          39.765644             4.200179
2            1.0       5610          44.473084             9.200713
3            0.0       3801          61.632991             0.000000
4            1.0       3795          11.330698            13.601054
…            …          …                  …                   …
2077         1.0        560           0.000000            27.535714
2078         3.0       1241          51.746978            27.924255
2079         1.0       1263         128.844814            23.632621
2080         0.0       1096          75.180657             0.000000
2081         0.0       1135         198.640529             0.000000

      other_stops_per_pop  bus_stat_per_1000  train_stat_per_1000  \
0                     0.0           2.979146             0.000000
1                     0.0           1.057857             0.081374
2                     0.0           1.247772             0.178253
3                     0.0           2.630887             0.000000
4                     0.0           0.527009             0.263505
…                     …                  …                   …
2077                  0.0           0.000000             1.785714
2078                  0.0           4.834811             2.417405
2079                  0.0          17.418844             0.791766
2080                  0.0           7.299270             0.000000
2081                  0.0          10.572687             0.000000

      other_stat_per_1000
0                     0.0
1                     0.0
2                     0.0
3                     0.0
4                     0.0
…                     …
2077                  0.0
2078                  0.0
2079                  0.0
2080                  0.0
2081                  0.0

[2082 rows x 14 columns]
```

[301]:
```
cars = pd.read_csv("../Data/2_Joined_entities/cars_final.csv")
cars
```

[301]:
```
      BFS_Nr  Combustion  Electric  comb_car_1000  el_car_1000
0          1        1400        78     695.134062    38.728898
1         10        3525       124     609.967122    21.456999
2        100        1654        57     708.047945    24.400685
3       1001         557        10     682.598039    12.254902
4       1002        2129        32     659.133127     9.907121
…        …           …        …              …            …
2151     990         205         8     903.083700    35.242291
2152     991         463        10     766.556291    16.556291
2153     992        1683        23     708.035339     9.676062
2154     993         298         5     732.186732    12.285012
2155     995        1616        54     678.421495    22.670025

[2156 rows x 5 columns]
```

[302]:
```python
comm = pd.read_csv("../Data/1_Cleaned/commuters.csv")
comm
```

[302]:
```
      BFS_Nr  inbound_share  outbound_share
0          1       0.476998        0.757576
1          2       0.597780        0.623587
2          3       0.482213        0.828609
3          4       0.420207        0.704678
4          5       0.697987        0.753500
…        …             …               …
2891    6802       0.000000        0.133333
2892    6803       0.064516        0.618421
2893    6804       0.681004        0.491429
2894    6805       0.200000        0.333333
2895    6806       0.449782        0.550000

[2896 rows x 3 columns]
```

And finally the table after the first joining step which serves as join base for the second step:

[303]:
```python
join_base = pd.read_csv("../Data/2_Joined_entities/bfs_base.csv")
join_base
```

[303]:
```
      BFS_Nr        municipality canton language  pop_count_BFS  \
0          1      Aeugst am Albis     ZH       de         2014.0
1          2   Affoltern am Albis     ZH       de        12289.0
2          3           Bonstetten     ZH       de         5610.0
3          4      Hausen am Albis     ZH       de         3781.0
4          5             Hedingen     ZH       de         3795.0
…        …                    …   …      …             …
2143    6806         Vendlincourt     JU       fr          560.0
2144    6807        Basse-Allaine     JU       fr         1235.0
```

```
2145    6808       Clos du Doubs     JU       fr         1263.0
2146    6809         Haute-Ajoie     JU       fr         1096.0
2147    6810          La Baroche     JU       fr         1129.0

      single_count_BFS  married_count_BFS  widowed_count_BFS  \
0           835.000000         923.000000          74.000000
1          5312.000000        5311.000000         586.000000
2          2435.000000        2577.000000         175.000000
3          1603.000000        1683.000000         136.000000
4          1618.000000        1729.000000         115.000000
…                  …                …                 …
2143        220.000000         255.000000          40.000000
2144        497.000000         518.000000         103.000000
2145        542.518248         513.459854          78.992701
2146        443.000000         483.000000          71.000000
2147        480.000000         458.000000          83.000000

      divorced_count_BFS      GA_BFS      HTA_BFS  fn_tck_BFS  single_share  \
0             182.000000  104.000000   846.000000  189.000000      0.414598
1            1080.000000  656.000000  4359.000000  772.000000      0.432256
2             423.000000  303.000000  2555.000000  647.000000      0.434046
3             359.000000  248.000000  1470.000000  133.000000      0.423962
4             333.000000  310.000000  1688.000000  448.000000      0.426350
…                    …           …            …           …            …
2143           45.000000    8.740000    98.000000   26.000000      0.392857
2144          117.000000   39.480000   171.000000   90.000000      0.402429
2145          128.029197   68.152555   215.156204   65.474453      0.429547
2146           99.000000   26.220000   189.480000   79.000000      0.404197
2147          108.000000   34.960000   174.000000  109.000000      0.425155

      married_share  widowed_share  divorced_share  GA_share  HTA_share  \
0          0.458292       0.036743        0.090367  0.051639   0.420060
1          0.432175       0.047685        0.087883  0.053381   0.354707
2          0.459358       0.031194        0.075401  0.054011   0.455437
3          0.445120       0.035969        0.094948  0.065591   0.388786
4          0.455599       0.030303        0.087747  0.081686   0.444796
…               …              …              …        …          …
2143       0.455357       0.071429        0.080357  0.015607   0.175000
2144       0.419433       0.083401        0.094737  0.031968   0.138462
2145       0.406540       0.062544        0.101369  0.053961   0.170353
2146       0.440693       0.064781        0.090328  0.023923   0.172883
2147       0.405669       0.073516        0.095660  0.030965   0.154119

      FNT_share
0      0.093843
1      0.062820
2      0.115330
```

```
3       0.035176
4       0.118050
…            …
2143    0.046429
2144    0.072874
2145    0.051840
2146    0.072080
2147    0.096546

[2148 rows x 19 columns]
```

## 8.2 Joining all together

[304]:
```python
data_frames = [join_base, pop_shares, dist, stops, cars, comm]
```

With the "reduce"-function, it is possible to merge all dataframes in the list together. To get only useful information, I will use a "left" join, as no more municipalities are known than in the town directory file. Other BFS_Nr in some dataframes occur due to older municipalities in earlier years. E.g. for the commuter share table which dates from the year 2000! Therefore, special attention must also be paid to the usefulness of this data.

[305]:
```python
inf_factors = reduce(lambda left, right: pd.merge(left, right, on="BFS_Nr",
 ↪how="left"), data_frames)
inf_factors
```

[305]:
```
      BFS_Nr        municipality canton language  pop_count_BFS  \
0          1      Aeugst am Albis     ZH       de         2014.0
1          2   Affoltern am Albis     ZH       de        12289.0
2          3           Bonstetten     ZH       de         5610.0
3          4      Hausen am Albis     ZH       de         3781.0
4          5             Hedingen     ZH       de         3795.0
…        …                  …   …      …              …
2143    6806         Vendlincourt     JU       fr          560.0
2144    6807         Basse-Allaine    JU       fr         1235.0
2145    6808         Clos du Doubs    JU       fr         1263.0
2146    6809          Haute-Ajoie     JU       fr         1096.0
2147    6810           La Baroche     JU       fr         1129.0

      single_count_BFS  married_count_BFS  widowed_count_BFS  \
0           835.000000         923.000000          74.000000
1          5312.000000        5311.000000         586.000000
2          2435.000000        2577.000000         175.000000
3          1603.000000        1683.000000         136.000000
4          1618.000000        1729.000000         115.000000
…                    …                  …                  …
2143        220.000000         255.000000          40.000000
```

```
2144        497.000000          518.000000          103.000000
2145        542.518248          513.459854           78.992701
2146        443.000000          483.000000           71.000000
2147        480.000000          458.000000           83.000000

      divorced_count_BFS      GA_BFS  …  other_stops_per_pop  \
0             182.000000  104.000000  …                  0.0
1            1080.000000  656.000000  …                  0.0
2             423.000000  303.000000  …                  0.0
3             359.000000  248.000000  …                  0.0
4             333.000000  310.000000  …                  0.0
…                    …          …  …                  …
2143           45.000000    8.740000  …                  0.0
2144          117.000000   39.480000  …                  0.0
2145          128.029197   68.152555  …                  0.0
2146           99.000000   26.220000  …                  0.0
2147          108.000000   34.960000  …                  0.0

      bus_stat_per_1000  train_stat_per_1000  other_stat_per_1000  Combustion  \
0              2.979146             0.000000                  0.0      1400.0
1              1.057857             0.081374                  0.0      6866.0
2              1.247772             0.178253                  0.0      3142.0
3              2.630887             0.000000                  0.0      2412.0
4              0.527009             0.263505                  0.0      2179.0
…                   …                  …                  …         …
2143           0.000000             1.785714                  0.0       386.0
2144           4.834811             2.417405                  0.0       859.0
2145          17.418844             0.791766                  0.0       946.0
2146           7.299270             0.000000                  0.0       804.0
2147          10.572687             0.000000                  0.0       807.0

      Electric  comb_car_1000  el_car_1000  inbound_share  outbound_share
0         78.0     695.134062    38.728898       0.476998        0.757576
1        271.0     558.711042    22.052242       0.597780        0.623587
2        145.0     560.071301    25.846702       0.482213        0.828609
3        100.0     634.569850    26.308866       0.420207        0.704678
4        106.0     574.176548    27.931489       0.697987        0.753500
…          …            …          …            …             …
2143      13.0     689.285714    23.214286       0.449782        0.550000
2144      20.0     692.183723    16.116035            NaN             NaN
2145      14.0     749.010293    11.084719            NaN             NaN
2146      16.0     733.576642    14.598540            NaN             NaN
2147      23.0     711.013216    20.264317            NaN             NaN

[2148 rows x 85 columns]
```

[306]:

```
inf_factors.columns
```

```
[306]: Index(['BFS_Nr', 'municipality', 'canton', 'language', 'pop_count_BFS',
              'single_count_BFS', 'married_count_BFS', 'widowed_count_BFS',
              'divorced_count_BFS', 'GA_BFS', 'HTA_BFS', 'fn_tck_BFS', 'single_share',
              'married_share', 'widowed_share', 'divorced_share', 'GA_share',
              'HTA_share', 'FNT_share', 'pop_count_x', 'age0_20', 'age20_40',
              'age40_60', 'age60+', 'age0_20cnt', 'age20_40cnt', 'age40_60cnt',
              'age60+cnt', 'birth_munic', 'birth_cant', 'birth_CH', 'birth_notCH',
              'birth_munic_cnt', 'birth_cant_cnt', 'birth_CH_cnt', 'birth_notCH_cnt',
              'male', 'female', 'male_cnt', 'female_cnt', 'resid_0_1y', 'resid_1_5y',
              'resid_6_10y', 'resid_10+y', 'resid_0_1y_cnt', 'resid_1_5y_cnt',
              'resid_6_10y_cnt', 'resid_10+y_cnt', 'hh_1', 'hh_2', 'hh_3_5', 'hh_6+',
              'hh_1_cnt', 'hh_2_cnt', 'hh_3_5_cnt', 'hh_6+_cnt', 'PT_dist_medium',
              'PT_time_medium', 'PT_dist_big', 'PT_time_big', 'str_dist_medium',
              'str_time_medium', 'str_dist_big', 'str_time_big', 'PT_fact_big',
              'PT_fact_medium', 'bus_count', 'other_count', 'train_count', 'bus_stat',
              'other_stat', 'train_stat', 'pop_count_y', 'bus_stops_per_pop',
              'train_stops_per_pop', 'other_stops_per_pop', 'bus_stat_per_1000',
              'train_stat_per_1000', 'other_stat_per_1000', 'Combustion', 'Electric',
              'comb_car_1000', 'el_car_1000', 'inbound_share', 'outbound_share'],
             dtype='object')
```

There are 3 population columns, which is only needed once!

```
[307]: inf_factors[["pop_count_BFS", "pop_count_x", "pop_count_y"]][:3]
```

```
[307]:    pop_count_BFS  pop_count_x  pop_count_y
       0         2014.0       2014.0       2014.0
       1        12289.0      12289.0      12289.0
       2         5610.0       5610.0       5610.0
```

```
[308]: inf_factors.drop(columns=["pop_count_x", "pop_count_y"], inplace=True)
```

## 8.3 Creating influence factors shares table

Now an additional table is created only using the population number and all share values (without absolute numbers, which are dependent on population):

```
[309]: inf_fac_share = inf_factors[['BFS_Nr', 'municipality', 'canton', 'language',␣
        ↪'pop_count_BFS',
               'single_share','married_share', 'widowed_share', 'divorced_share',
               'GA_share', 'HTA_share', 'FNT_share', 'age0_20',
               'age20_40', 'age40_60', 'age60+',
               'birth_munic', 'birth_cant', 'birth_CH', 'birth_notCH',
               'male', 'female', 'resid_0_1y', 'resid_1_5y',
               'resid_6_10y', 'resid_10+y', 'hh_1', 'hh_2',
               'hh_3_5', 'hh_6+', 'PT_dist_medium', 'PT_time_medium',
```

```
       'PT_dist_big', 'PT_time_big', 'str_dist_medium', 'str_time_medium',
       'str_dist_big', 'str_time_big',  'PT_fact_big', 'PT_fact_medium',
       'bus_stops_per_pop', 'train_stops_per_pop', 'other_stops_per_pop',␣
↪'bus_stat_per_1000',
       'train_stat_per_1000', 'other_stat_per_1000', 'comb_car_1000',
       'el_car_1000', 'inbound_share', 'outbound_share']]
inf_fac_share
```

[309]:

```
      BFS_Nr        municipality canton language  pop_count_BFS  single_share  \
0          1     Aeugst am Albis     ZH       de         2014.0      0.414598
1          2  Affoltern am Albis     ZH       de        12289.0      0.432256
2          3          Bonstetten     ZH       de         5610.0      0.434046
3          4     Hausen am Albis     ZH       de         3781.0      0.423962
4          5            Hedingen     ZH       de         3795.0      0.426350
…        …                  …     …      …       …              …
2143    6806         Vendlincourt     JU       fr          560.0      0.392857
2144    6807        Basse-Allaine     JU       fr         1235.0      0.402429
2145    6808        Clos du Doubs     JU       fr         1263.0      0.429547
2146    6809          Haute-Ajoie     JU       fr         1096.0      0.404197
2147    6810           La Baroche     JU       fr         1129.0      0.425155

      married_share  widowed_share  divorced_share  GA_share  …  \
0          0.458292       0.036743        0.090367  0.051639  …
1          0.432175       0.047685        0.087883  0.053381  …
2          0.459358       0.031194        0.075401  0.054011  …
3          0.445120       0.035969        0.094948  0.065591  …
4          0.455599       0.030303        0.087747  0.081686  …
…               …            …            …       …  …
2143       0.455357       0.071429        0.080357  0.015607  …
2144       0.419433       0.083401        0.094737  0.031968  …
2145       0.406540       0.062544        0.101369  0.053961  …
2146       0.440693       0.064781        0.090328  0.023923  …
2147       0.405669       0.073516        0.095660  0.030965  …

      bus_stops_per_pop  train_stops_per_pop  other_stops_per_pop  \
0            104.428500             0.000000                  0.0
1             39.765644             4.200179                  0.0
2             44.473084             9.200713                  0.0
3             61.632991             0.000000                  0.0
4             11.330698            13.601054                  0.0
…                   …                  …                   …
2143           0.000000            27.535714                  0.0
2144          51.746978            27.924255                  0.0
2145         128.844814            23.632621                  0.0
2146          75.180657             0.000000                  0.0
2147         198.640529             0.000000                  0.0
```

```
      bus_stat_per_1000  train_stat_per_1000  other_stat_per_1000  \
0              2.979146             0.000000                  0.0
1              1.057857             0.081374                  0.0
2              1.247772             0.178253                  0.0
3              2.630887             0.000000                  0.0
4              0.527009             0.263505                  0.0
…                   …                  …                    …
2143           0.000000             1.785714                  0.0
2144           4.834811             2.417405                  0.0
2145          17.418844             0.791766                  0.0
2146           7.299270             0.000000                  0.0
2147          10.572687             0.000000                  0.0

      comb_car_1000  el_car_1000  inbound_share  outbound_share
0        695.134062    38.728898       0.476998        0.757576
1        558.711042    22.052242       0.597780        0.623587
2        560.071301    25.846702       0.482213        0.828609
3        634.569850    26.308866       0.420207        0.704678
4        574.176548    27.931489       0.697987        0.753500
…                 …          …              …             …
2143     689.285714    23.214286       0.449782        0.550000
2144     692.183723    16.116035            NaN             NaN
2145     749.010293    11.084719            NaN             NaN
2146     733.576642    14.598540            NaN             NaN
2147     711.013216    20.264317            NaN             NaN

[2148 rows x 50 columns]
```

### 8.4 Creating influence factors count table

Now an additional table is created only using count values (without share values):

[310]:
```
inf_fac_count = inf_factors[['BFS_Nr', 'municipality', 'canton', 'language',␣
 ↪'pop_count_BFS',
       'single_count_BFS', 'married_count_BFS', 'widowed_count_BFS',
       'divorced_count_BFS',
       'GA_BFS', 'HTA_BFS', 'fn_tck_BFS', 'age0_20cnt',
       'age20_40cnt', 'age40_60cnt', 'age60+cnt',
       'birth_munic_cnt', 'birth_cant_cnt', 'birth_CH_cnt', 'birth_notCH_cnt',
       'male_cnt', 'female_cnt', 'resid_0_1y_cnt', 'resid_1_5y_cnt',
       'resid_6_10y_cnt', 'resid_10+y_cnt', 'hh_1_cnt', 'hh_2_cnt',
       'hh_3_5_cnt', 'hh_6+_cnt', 'PT_dist_medium', 'PT_time_medium',
       'PT_dist_big', 'PT_time_big', 'str_dist_medium', 'str_time_medium',
       'str_dist_big', 'str_time_big',  'PT_fact_big', 'PT_fact_medium',
       'bus_count', 'other_count', 'train_count', 'bus_stat', 'other_stat',
       'train_stat', 'Combustion', 'Electric']]
```

```
inf_fac_count
```

[310]:

```
      BFS_Nr        municipality canton language  pop_count_BFS  \
0          1      Aeugst am Albis     ZH       de         2014.0
1          2   Affoltern am Albis     ZH       de        12289.0
2          3           Bonstetten     ZH       de         5610.0
3          4      Hausen am Albis     ZH       de         3781.0
4          5             Hedingen     ZH       de         3795.0
…        …                    …   …      …                …
2143    6806        Vendlincourt     JU       fr          560.0
2144    6807       Basse-Allaine     JU       fr         1235.0
2145    6808       Clos du Doubs     JU       fr         1263.0
2146    6809        Haute-Ajoie     JU       fr         1096.0
2147    6810          La Baroche     JU       fr         1129.0

      single_count_BFS  married_count_BFS  widowed_count_BFS  \
0           835.000000         923.000000          74.000000
1          5312.000000        5311.000000         586.000000
2          2435.000000        2577.000000         175.000000
3          1603.000000        1683.000000         136.000000
4          1618.000000        1729.000000         115.000000
…                  …                  …                  …
2143        220.000000         255.000000          40.000000
2144        497.000000         518.000000         103.000000
2145        542.518248         513.459854          78.992701
2146        443.000000         483.000000          71.000000
2147        480.000000         458.000000          83.000000

      divorced_count_BFS      GA_BFS  …  PT_fact_big  PT_fact_medium  \
0             182.000000  104.000000  …     1.717471        1.572727
1            1080.000000  656.000000  …     1.637173        1.491281
2             423.000000  303.000000  …     1.590610        1.527228
3             359.000000  248.000000  …     1.685243        1.926940
4             333.000000  310.000000  …     1.521911        1.251227
…                    …          …  …          …               …
2143           45.000000    8.740000  …     1.517322        1.543543
2144          117.000000   39.480000  …     1.336080        1.411685
2145          128.029197   68.152555  …     1.237077        1.382133
2146           99.000000   26.220000  …     1.559605        1.866231
2147          108.000000   34.960000  …     1.708613        1.608273

      bus_count  other_count  train_count  bus_stat  other_stat  train_stat  \
0      210319.0          0.0          0.0       6.0         0.0         0.0
1      488680.0          0.0      51616.0      13.0         0.0         1.0
2      249494.0          0.0      51616.0       7.0         0.0         1.0
3      234267.0          0.0          0.0      10.0         0.0         0.0
```

```
4       43000.0          0.0      51616.0       2.0          0.0          1.0
…          …         …          …        …         …
2143        0.0          0.0      15420.0       0.0          0.0          1.0
2144    64218.0          0.0      34654.0       6.0          0.0          3.0
2145   162731.0          0.0      29848.0      22.0          0.0          1.0
2146    82398.0          0.0          0.0       8.0          0.0          0.0
2147   225457.0          0.0          0.0      12.0          0.0          0.0

      Combustion  Electric
0         1400.0      78.0
1         6866.0     271.0
2         3142.0     145.0
3         2412.0     100.0
4         2179.0     106.0
…            …         …
2143       386.0      13.0
2144       859.0      20.0
2145       946.0      14.0
2146       804.0      16.0
2147       807.0      23.0

[2148 rows x 48 columns]
```

### 8.5 Writing csv's

[311]:
```
inf_fac_share.to_csv("../Data/3_Output/inf_fac_share.csv", index=False)
inf_fac_count.to_csv("../Data/3_Output/inf_fac_count.csv", index=False)
inf_factors.to_csv("../Data/3_Output/influence_factors.csv", index=False)
```

## 9 Aggregating on cantonal level

In order to be able to perform the cluster analysis, an aggregation on cantonal level will help to get some insights, as the municipality-level-data are too wide-spreaded to allow a meaningful cluster analysis. This can be done using the count table which can be used afterwards to calculate the shares again.

### 9.1 Loading Count table

[312]:
```
inf_fac_count = pd.read_csv("../Data/3_Output/inf_fac_count.csv")
inf_fac_count[:2]
```

[312]:
```
   BFS_Nr        municipality canton language  pop_count_BFS  \
0       1     Aeugst am Albis     ZH       de         2014.0
```

```
1       2  Affoltern am Albis     ZH       de        12289.0

   single_count_BFS  married_count_BFS  widowed_count_BFS  divorced_count_BFS  \
0             835.0              923.0               74.0               182.0
1            5312.0             5311.0              586.0              1080.0

   GA_BFS  …  PT_fact_big  PT_fact_medium  bus_count  other_count  \
0   104.0  …     1.717471        1.572727   210319.0          0.0
1   656.0  …     1.637173        1.491281   488680.0          0.0

   train_count  bus_stat  other_stat  train_stat  Combustion  Electric
0          0.0       6.0         0.0         0.0      1400.0      78.0
1      51616.0      13.0         0.0         1.0      6866.0     271.0

[2 rows x 48 columns]
```

### 9.2 Aggregating count data on cantonal level

[313]:
```
inf_fac_cant_count = inf_fac_count.groupby(by="canton").sum().reset_index()
inf_fac_cant_count[:2]
```

[313]:
```
  canton  BFS_Nr  pop_count_BFS  single_count_BFS  married_count_BFS  \
0     AG  831702       692755.0     297471.100902      307349.854842
1     AI   18626        16293.0       7532.487499        6989.385568

   widowed_count_BFS  divorced_count_BFS        GA_BFS        HTA_BFS  \
0       30602.460531        57310.589420  44570.009972  200954.690404
1         810.286557          960.840376    366.168493    4729.636522

     fn_tck_BFS  …  PT_fact_big  PT_fact_medium   bus_count  other_count  \
0  20423.510508  …   303.027337      329.199098  39965497.0      11943.0
1    270.637418  …     9.591087       11.822487    382319.0       4212.0

   train_count  bus_stat  other_stat  train_stat  Combustion  Electric
0    4685977.0    1258.0         6.0       104.0    441124.0   15155.0
1     240360.0      51.0         6.0        10.0     11454.0     341.0

[2 rows x 46 columns]
```

The column "BFS_Nr" doesn't make any sense now, it can be deleted. Additionally, all data coming from the str_PT_dist_time-table cannot be aggregated via sum, the mean has to be used instead. Therefore, these columns can be deleted as well here

[314]:
```
inf_fac_cant_count.drop(columns=["BFS_Nr", "PT_dist_medium", "PT_time_medium",␣
↪"PT_dist_big",
```

```
                                  "PT_time_big", "str_dist_medium",␣
↪"str_time_medium", "str_dist_big",
                                  "str_time_big", "PT_fact_big",␣
↪"PT_fact_medium"], inplace=True)
inf_fac_cant_count
```

[314]:

```
   canton  pop_count_BFS  single_count_BFS  married_count_BFS  \
0      AG       692755.0     297471.100902      307349.854842
1      AI        16293.0       7532.487499        6989.385568
2      AR        55473.0      23643.010044       24158.829953
3      BE      1042905.0     459326.394434      436561.139551
4      BL       291047.0     118605.263806      131474.466137
5      BS       196667.0      96055.000000       71439.000000
6      FR       323635.0     150430.837065      132807.005462
7      GE       506962.0     245525.548012      191833.576373
8      GL        40383.0      16978.000000       17722.000000
9      GR       200224.0      85223.444093       87447.718939
10     JU        73670.0      32401.759908       30115.022381
11     LU       415620.0     190661.193106      176391.348258
12     NE       176004.0      79969.277875       67701.200215
13     NW        43256.0      18616.335877       18983.536638
14     OW        38187.0      16751.610592       17029.315680
15     SG       515763.0     224882.499133      223692.600375
16     SH        83109.0      34223.277476       36841.006232
17     SO       277420.0     117155.511648      121002.577625
18     SZ       162285.0      70602.059016       71220.158145
19     TG       282783.0     120310.468140      125198.208119
20     TI       352033.0     150270.146489      148901.754581
21     UR        36940.0      15889.826086       17070.383701
22     VD       813243.0     389380.035338      317549.747054
23     VS       346901.0     149846.558183      148044.423722
24     ZG       128830.0      56532.582950       57177.231125
25     ZH      1555643.0     732784.479834      625360.848883

    widowed_count_BFS  divorced_count_BFS        GA_BFS        HTA_BFS  \
0        30602.460531        57310.589420  44570.009972  200954.690404
1          810.286557          960.840376    366.168493    4729.636522
2         2771.984687         4899.175317   2166.346971   18833.928230
3        54655.165267        92342.401032  90529.151380  390061.558181
4        15811.440620        25153.855079   8742.634516   81323.437347
5         9964.000000        19205.000000   9456.000000   67743.000000
6        13332.480338        27051.680694  13903.014944   68175.795636
7        20206.418116        49382.457556   5729.943780   86214.243947
8         2263.000000         3420.000000   2083.650000   10936.000000
9        10721.717195        16828.219773   7944.660116   63225.652452
10        4452.298846         6700.918865   2778.902639   14739.249196
11       18684.296239        29871.061814  21911.922149  148160.130552
```

| | | | | |
|---|---|---|---|---|
| 12 | 9459.270019 | 18874.251891 | 6524.721336 | 38923.332783 |
| 13 | 2041.286789 | 3609.884159 | 1282.159528 | 17492.601064 |
| 14 | 1811.227414 | 2593.846314 | 949.716303 | 14513.380187 |
| 15 | 24257.188231 | 42899.664357 | 24277.826649 | 156530.073890 |
| 16 | 4535.574969 | 7508.141324 | 4811.337837 | 26145.047292 |
| 17 | 14085.835174 | 25153.654706 | 18725.130522 | 81696.053587 |
| 18 | 7172.147408 | 13287.886200 | 8073.619530 | 55923.978813 |
| 19 | 12603.503506 | 24670.820234 | 12576.388947 | 81813.262834 |
| 20 | 21300.060726 | 31559.057872 | 3701.621190 | 44519.306410 |
| 21 | 1973.757203 | 2006.033010 | 1367.067164 | 12470.026546 |
| 22 | 33549.795843 | 72738.898909 | 28709.934723 | 215268.418759 |
| 23 | 18374.333242 | 30619.684853 | 17107.908915 | 98866.473623 |
| 24 | 4996.111934 | 10121.073992 | 8004.442703 | 54419.568660 |
| 25 | 62256.820278 | 135193.849968 | 86434.012661 | 614953.638719 |

| | fn_tck_BFS | age0_20cnt | … | hh_3_5_cnt | hh_6+_cnt | bus_count | \ |
|---|---|---|---|---|---|---|---|
| 0 | 20423.510508 | 138566.0 | … | 90887.0 | 4685.0 | 39965497.0 | |
| 1 | 270.637418 | 3414.0 | … | 2134.0 | 170.0 | 382319.0 | |
| 2 | 2332.170164 | 11163.0 | … | 6982.0 | 486.0 | 3756606.0 | |
| 3 | 49341.333719 | 197839.0 | … | 125499.0 | 6169.0 | 81837057.0 | |
| 4 | 393.090527 | 55753.0 | … | 38030.0 | 1455.0 | 21933757.0 | |
| 5 | 158.000000 | 34218.0 | … | 22161.0 | 1012.0 | 31795495.0 | |
| 6 | 1180.150804 | 69932.0 | … | 45722.0 | 2288.0 | 21361315.0 | |
| 7 | 93181.962742 | 106488.0 | … | 69422.0 | 6056.0 | 66078513.0 | |
| 8 | 1289.000000 | 7821.0 | … | 5262.0 | 249.0 | 1812977.0 | |
| 9 | 996.460392 | 34869.0 | … | 24832.0 | 924.0 | 23565523.0 | |
| 10 | 3804.762369 | 15243.0 | … | 9815.0 | 480.0 | 5860403.0 | |
| 11 | 24181.211809 | 83872.0 | … | 54032.0 | 2893.0 | 40169780.0 | |
| 12 | 15804.142196 | 33882.0 | … | 21501.0 | 824.0 | 16792440.0 | |
| 13 | 1643.791970 | 7863.0 | … | 5619.0 | 215.0 | 1854407.0 | |
| 14 | 1125.014538 | 7589.0 | … | 5042.0 | 292.0 | 1196415.0 | |
| 15 | 25012.358804 | 105183.0 | … | 66254.0 | 4233.0 | 41456933.0 | |
| 16 | 5999.969379 | 15515.0 | … | 10172.0 | 539.0 | 11326955.0 | |
| 17 | 5982.661926 | 52616.0 | … | 34888.0 | 1820.0 | 19294954.0 | |
| 18 | 5989.329416 | 31099.0 | … | 21529.0 | 1095.0 | 11118077.0 | |
| 19 | 7550.563983 | 57168.0 | … | 36901.0 | 2118.0 | 13990290.0 | |
| 20 | 32567.631980 | 62029.0 | … | 46960.0 | 1412.0 | 41677433.0 | |
| 21 | 102.425358 | 7298.0 | … | 4710.0 | 275.0 | 3499620.0 | |
| 22 | 67292.010980 | 174170.0 | … | 110973.0 | 4797.0 | 73918324.0 | |
| 23 | 654.182167 | 64386.0 | … | 44380.0 | 1990.0 | 21728126.0 | |
| 24 | 13401.742826 | 25990.0 | … | 17948.0 | 627.0 | 12520802.0 | |
| 25 | 187611.124129 | 307069.0 | … | 202104.0 | 9401.0 | 24293097.0 | |

| | other_count | train_count | bus_stat | other_stat | train_stat | Combustion | \ |
|---|---|---|---|---|---|---|---|
| 0 | 11943.0 | 4685977.0 | 1258.0 | 6.0 | 104.0 | 441124.0 | |
| 1 | 4212.0 | 240360.0 | 51.0 | 6.0 | 10.0 | 11454.0 | |
| 2 | 1704.0 | 858480.0 | 206.0 | 2.0 | 30.0 | 36578.0 | |

| | | | | | | |
|---|---|---|---|---|---|---|
| 3 | 326799.0 | 10218184.0 | 2994.0 | 95.0 | 335.0 | 611504.0 |
| 4 | 1456.0 | 734217.0 | 525.0 | 2.0 | 21.0 | 167741.0 |
| 5 | 30509.0 | 360780.0 | 218.0 | 3.0 | 4.0 | 71629.0 |
| 6 | 12330.0 | 1581417.0 | 874.0 | 14.0 | 61.0 | 206655.0 |
| 7 | 15461.0 | 1096644.0 | 736.0 | 15.0 | 20.0 | 267263.0 |
| 8 | 17756.0 | 362422.0 | 146.0 | 11.0 | 18.0 | 26712.0 |
| 9 | 108020.0 | 2181111.0 | 1479.0 | 104.0 | 120.0 | 128135.0 |
| 10 | 0.0 | 644688.0 | 371.0 | 0.0 | 32.0 | 47813.0 |
| 11 | 66179.0 | 2198763.0 | 878.0 | 27.0 | 61.0 | 249365.0 |
| 12 | 105248.0 | 2206993.0 | 571.0 | 17.0 | 69.0 | 106636.0 |
| 13 | 75010.0 | 303309.0 | 98.0 | 28.0 | 9.0 | 30907.0 |
| 14 | 23734.0 | 307824.0 | 123.0 | 20.0 | 14.0 | 26280.0 |
| 15 | 75587.0 | 3089728.0 | 1297.0 | 51.0 | 79.0 | 320319.0 |
| 16 | 5795.0 | 420634.0 | 246.0 | 5.0 | 5.0 | 53123.0 |
| 17 | 4174.0 | 1549963.0 | 738.0 | 5.0 | 40.0 | 178698.0 |
| 18 | 84104.0 | 1089857.0 | 496.0 | 42.0 | 38.0 | 116063.0 |
| 19 | 26026.0 | 2548884.0 | 665.0 | 15.0 | 76.0 | 199665.0 |
| 20 | 87988.0 | 2223996.0 | 1586.0 | 39.0 | 81.0 | 256407.0 |
| 21 | 99483.0 | 266194.0 | 175.0 | 33.0 | 15.0 | 23322.0 |
| 22 | 58938.0 | 7898632.0 | 1991.0 | 42.0 | 259.0 | 451089.0 |
| 23 | 865851.0 | 2352616.0 | 1628.0 | 159.0 | 116.0 | 239561.0 |
| 24 | 6398.0 | 971370.0 | 279.0 | 13.0 | 21.0 | 92972.0 |
| 25 | 135536.0 | 10011063.0 | 1238.0 | 25.0 | 182.0 | 825159.0 |

| | Electric |
|---|---|
| 0 | 15155.0 |
| 1 | 341.0 |
| 2 | 1090.0 |
| 3 | 23755.0 |
| 4 | 6037.0 |
| 5 | 2913.0 |
| 6 | 7329.0 |
| 7 | 10711.0 |
| 8 | 717.0 |
| 9 | 3407.0 |
| 10 | 1381.0 |
| 11 | 8020.0 |
| 12 | 2998.0 |
| 13 | 1157.0 |
| 14 | 853.0 |
| 15 | 9581.0 |
| 16 | 1580.0 |
| 17 | 4844.0 |
| 18 | 4190.0 |
| 19 | 6266.0 |
| 20 | 10267.0 |
| 21 | 478.0 |

```
22    18919.0
23     5999.0
24     5489.0
25    37470.0

[26 rows x 35 columns]
```

For the time and distance tables, a second groupby via "mean" can be used:

```
[315]: inf_fac_cant_count_2 = inf_fac_count[["canton", "PT_dist_medium",␣
        ↪"PT_time_medium", "PT_dist_big",
                                  "PT_time_big", "str_dist_medium",␣
        ↪"str_time_medium", "str_dist_big",
                                  "str_time_big", "PT_fact_big",␣
        ↪"PT_fact_medium"]].groupby(by="canton").mean().reset_index()
       inf_fac_cant_count_2[:2]
```

```
[315]:   canton  PT_dist_medium  PT_time_medium  PT_dist_big  PT_time_big  \
       0     AG       51.897556       82.191404    40.289374    64.859354
       1     AI       24.931167       65.814833    78.148667   113.781833

          str_dist_medium  str_time_medium  str_dist_big  str_time_big  PT_fact_big  \
       0        46.198854        50.497818     36.920348      43.25202     1.530441
       1        20.396000        34.469167     71.320833      71.46150     1.598514

          PT_fact_medium
       0        1.662622
       1        1.970415
```

These informations must be brought together to the count table:

```
[316]: inf_fac_cant_count[["canton", "PT_dist_medium", "PT_time_medium", "PT_dist_big",
                          "PT_time_big", "str_dist_medium", "str_time_medium",␣
        ↪"str_dist_big",
                         "str_time_big", "PT_fact_big", "PT_fact_medium"]] =␣
        ↪inf_fac_cant_count_2

       inf_fac_cant_count[:2]
```

```
[316]:   canton  pop_count_BFS  single_count_BFS  married_count_BFS  \
       0     AG       692755.0     297471.100902      307349.854842
       1     AI        16293.0       7532.487499        6989.385568

          widowed_count_BFS  divorced_count_BFS        GA_BFS        HTA_BFS  \
       0       30602.460531        57310.589420  44570.009972  200954.690404
       1         810.286557          960.840376    366.168493    4729.636522
```

```
     fn_tck_BFS  age0_20cnt  …  PT_dist_medium  PT_time_medium  PT_dist_big  \
0  20423.510508    138566.0  …       51.897556       82.191404    40.289374
1    270.637418      3414.0  …       24.931167       65.814833    78.148667

   PT_time_big  str_dist_medium  str_time_medium  str_dist_big  str_time_big  \
0    64.859354        46.198854        50.497818     36.920348      43.25202
1   113.781833        20.396000        34.469167     71.320833      71.46150

   PT_fact_big  PT_fact_medium
0     1.530441        1.662622
1     1.598514        1.970415

[2 rows x 45 columns]
```

## 9.3 Calculating shares on cantonal level

[317]:
```python
inf_fac_cant_share = copy.deepcopy(inf_fac_cant_count)
inf_fac_cant_share.columns
```

[317]:
```
Index(['canton', 'pop_count_BFS', 'single_count_BFS', 'married_count_BFS',
       'widowed_count_BFS', 'divorced_count_BFS', 'GA_BFS', 'HTA_BFS',
       'fn_tck_BFS', 'age0_20cnt', 'age20_40cnt', 'age40_60cnt', 'age60+cnt',
       'birth_munic_cnt', 'birth_cant_cnt', 'birth_CH_cnt', 'birth_notCH_cnt',
       'male_cnt', 'female_cnt', 'resid_0_1y_cnt', 'resid_1_5y_cnt',
       'resid_6_10y_cnt', 'resid_10+y_cnt', 'hh_1_cnt', 'hh_2_cnt',
       'hh_3_5_cnt', 'hh_6+_cnt', 'bus_count', 'other_count', 'train_count',
       'bus_stat', 'other_stat', 'train_stat', 'Combustion', 'Electric',
       'PT_dist_medium', 'PT_time_medium', 'PT_dist_big', 'PT_time_big',
       'str_dist_medium', 'str_time_medium', 'str_dist_big', 'str_time_big',
       'PT_fact_big', 'PT_fact_medium'],
      dtype='object')
```

[318]:
```python
inf_fac_cant_share["single_share"] = inf_fac_cant_share["single_count_BFS"] /␣
 ↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["married_share"] = inf_fac_cant_share["married_count_BFS"] /␣
 ↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["widowed_share"] = inf_fac_cant_share["widowed_count_BFS"] /␣
 ↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["divorced_share"] = inf_fac_cant_share["divorced_count_BFS"]␣
 ↪/ inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["GA_share"] = inf_fac_cant_share["GA_BFS"] /␣
 ↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["HTA_share"] = inf_fac_cant_share["HTA_BFS"] /␣
 ↪inf_fac_cant_share["pop_count_BFS"]
```

```python
inf_fac_cant_share["FNT_share"] = inf_fac_cant_share["fn_tck_BFS"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["age0_20_share"] = inf_fac_cant_share["age0_20cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["age20_40_share"] = inf_fac_cant_share["age20_40cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["age40_60_share"] = inf_fac_cant_share["age40_60cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["age60+_share"] = inf_fac_cant_share["age60+cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["birth_munic_share"] = inf_fac_cant_share["birth_munic_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["birth_cant_share"] = inf_fac_cant_share["birth_cant_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["birth_CH_share"] = inf_fac_cant_share["birth_CH_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["birth_notCH_share"] = inf_fac_cant_share["birth_notCH_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["male_share"] = inf_fac_cant_share["male_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["female_share"] = inf_fac_cant_share["female_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["resid_0_1y_share"] = inf_fac_cant_share["resid_0_1y_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["resid_1_5y_share"] = inf_fac_cant_share["resid_1_5y_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["resid_6_10y_share"] = inf_fac_cant_share["resid_6_10y_cnt"] / inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["resid_10+y_share"] = inf_fac_cant_share["resid_10+y_cnt"] / inf_fac_cant_share["pop_count_BFS"]

# the hh-tables are related to households, not people, therefore the share is calculated differently:
inf_fac_cant_share["hh_1_share"] = inf_fac_cant_share["hh_1_cnt"] / (inf_fac_cant_share["hh_1_cnt"] + inf_fac_cant_share["hh_2_cnt"] + inf_fac_cant_share["hh_3_5_cnt"] + inf_fac_cant_share["hh_6+_cnt"])
inf_fac_cant_share["hh_2_share"] = inf_fac_cant_share["hh_2_cnt"] / (inf_fac_cant_share["hh_1_cnt"] + inf_fac_cant_share["hh_2_cnt"] + inf_fac_cant_share["hh_3_5_cnt"] + inf_fac_cant_share["hh_6+_cnt"])
inf_fac_cant_share["hh_3_5_share"] = inf_fac_cant_share["hh_3_5_cnt"] / (inf_fac_cant_share["hh_1_cnt"] + inf_fac_cant_share["hh_2_cnt"] + inf_fac_cant_share["hh_3_5_cnt"] + inf_fac_cant_share["hh_6+_cnt"])
```

```
inf_fac_cant_share["hh_6+_share"] = inf_fac_cant_share["hh_6+_cnt"] /␣
↪(inf_fac_cant_share["hh_1_cnt"] + inf_fac_cant_share["hh_2_cnt"] +
                                                                    ␣
↪inf_fac_cant_share["hh_3_5_cnt"] + inf_fac_cant_share["hh_6+_cnt"])

inf_fac_cant_share["bus_stops_per_pop"] = inf_fac_cant_share["bus_count"] /␣
↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["other_stops_per_pop"] = inf_fac_cant_share["other_count"] /␣
↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["train_stops_per_pop"] = inf_fac_cant_share["train_count"] /␣
↪inf_fac_cant_share["pop_count_BFS"]
inf_fac_cant_share["bus_stat_per_1000"] = inf_fac_cant_share["bus_stat"] /␣
↪inf_fac_cant_share["pop_count_BFS"] * 1000
inf_fac_cant_share["other_stat_per_1000"] = inf_fac_cant_share["other_stat"] /␣
↪inf_fac_cant_share["pop_count_BFS"] * 1000
inf_fac_cant_share["train_stat_per_1000"] = inf_fac_cant_share["train_stat"] /␣
↪inf_fac_cant_share["pop_count_BFS"] * 1000
inf_fac_cant_share["comb_car_per_1000"] = inf_fac_cant_share["Combustion"] /␣
↪inf_fac_cant_share["pop_count_BFS"] * 1000
inf_fac_cant_share["el_car_per_1000"] = inf_fac_cant_share["Electric"] /␣
↪inf_fac_cant_share["pop_count_BFS"] * 1000

inf_fac_cant_share.drop(columns=['single_count_BFS', 'married_count_BFS',
       'widowed_count_BFS', 'divorced_count_BFS', 'GA_BFS', 'HTA_BFS',
       'fn_tck_BFS', 'age0_20cnt', 'age20_40cnt', 'age40_60cnt', 'age60+cnt',
       'birth_munic_cnt', 'birth_cant_cnt', 'birth_CH_cnt', 'birth_notCH_cnt',
       'male_cnt', 'female_cnt', 'resid_0_1y_cnt', 'resid_1_5y_cnt',
       'resid_6_10y_cnt', 'resid_10+y_cnt', 'hh_1_cnt', 'hh_2_cnt',
       'hh_3_5_cnt', 'hh_6+_cnt', 'bus_count', 'other_count', 'train_count',
       'bus_stat', 'other_stat', 'train_stat', 'Combustion', 'Electric'],
       inplace=True)

inf_fac_cant_share[:2]
```

[318]:

```
  canton  pop_count_BFS  PT_dist_medium  PT_time_medium  PT_dist_big  \
0     AG       692755.0       51.897556       82.191404    40.289374
1     AI        16293.0       24.931167       65.814833    78.148667

   PT_time_big  str_dist_medium  str_time_medium  str_dist_big  str_time_big  \
0    64.859354        46.198854        50.497818     36.920348      43.25202
1   113.781833        20.396000        34.469167     71.320833      71.46150

   …  hh_3_5_share  hh_6+_share  bus_stops_per_pop  other_stops_per_pop  \
0  …      0.304797     0.015712          57.690666             0.017240
1  …      0.320132     0.025503          23.465230             0.258516
```

```
   train_stops_per_pop  bus_stat_per_1000  other_stat_per_1000  \
0             6.764263           1.815938             0.008661
1            14.752348           3.130179             0.368256

   train_stat_per_1000  comb_car_per_1000  el_car_per_1000
0             0.150125         636.767688        21.876421
1             0.613761         703.001289        20.929233

[2 rows x 45 columns]
```

[319]:
```
inf_fac_cant_share["hh_1_share"] + inf_fac_cant_share["hh_2_share"] +␣
↪inf_fac_cant_share["hh_3_5_share"] + inf_fac_cant_share["hh_6+_share"]
```

[319]:
```
0     1.0
1     1.0
2     1.0
3     1.0
4     1.0
5     1.0
6     1.0
7     1.0
8     1.0
9     1.0
10    1.0
11    1.0
12    1.0
13    1.0
14    1.0
15    1.0
16    1.0
17    1.0
18    1.0
19    1.0
20    1.0
21    1.0
22    1.0
23    1.0
24    1.0
25    1.0
dtype: float64
```

[320]:
```
inf_fac_cant_share["resid_0_1y_share"] + inf_fac_cant_share["resid_1_5y_share"]␣
↪+ inf_fac_cant_share["resid_6_10y_share"] +␣
↪inf_fac_cant_share["resid_10+y_share"]
```

[320]:
```
0     0.986581
1     0.999018
```

```
2     0.996990
3     0.997165
4     0.999605
5     1.000107
6     0.981056
7     0.988757
8     1.011465
9     0.991989
10    0.999620
11    0.997731
12    0.930564
13    1.005733
14    0.997879
15    0.997353
16    0.999820
17    0.995451
18    0.999051
19    1.000180
20    0.987887
21    0.991906
22    0.980310
23    0.955688
24    0.999674
25    0.998396
dtype: float64
```

There are some differences in the population number in the different tables, therefore the sum over all cantons is not always 1 here. But only 2 cantons do reach 93% and 95%, while the rest is between 98% and 101%. This is quite ok.

### 9.4 Writing csv's

This count table can be stored as csv now:

[321]:
```
inf_fac_cant_count.to_csv("../Data/3_Output/inf_fac_cant_count.csv",␣
↪index=False)
```

As well as the share table:

[322]:
```
inf_fac_cant_share.to_csv("../Data/3_Output/inf_fac_cant_share.csv",␣
↪index=False)
```

# 10 Tests (not runnable)

## 10.1 Connecting to SQLite

This part was not used but could be used for further applications when establishing database with sqlite for example. Therefore it is not deleted yet.

### 10.1.1 Set up connection

[ ]:
```
# my_conn=create_engine("sqlite:////content/drive/MyDrive/MasterThesis/Data/
↪Database/PT_influences.db")
# # con = sqlite3.connect("sqlite:///../Data/Database/PT_influences.db")
# # my_conn.cursor()
```

### 10.1.2 Create tables in database

[ ]:
```
# my_conn.execute('''CREATE TABLE IF NOT EXISTS town_directory (
#   PLZ INT,
#   BFS_Nr INT,
#   municipality VARCHAR(100),
#   canton VARCHAR(2),
#   Ecoord FLOAT,
#   Ncoord FLOAT,
#   language VARCHAR(2),
#   PRIMARY KEY (PLZ, BFS_Nr)
# );''')
```

[ ]:
```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x7f9ba6fedf90>
```

[ ]:
```
# inf_fac_share.columnsbu
```

[ ]:
```
Index(['BFS_Nr', 'municipality', 'canton', 'language', 'pop_count_BFS',
       'single_share', 'married_share', 'widowed_share', 'divorced_share',
       'GA_share', 'HTA_share', 'FNT_share', '<20', '20-40', '40-60', '>60',
       'birth_munic', 'birth_cant', 'birth_CH', 'birth_notCH', 'male',
       'female', 'resid_<1y', 'resid_1-5y', 'resid_6-10y', 'resid_>10y',
       'hh_1', 'hh_2', 'hh_3-5', 'hh_>6', 'PT_dist_medium', 'PT_time_medium',
       'PT_dist_big', 'PT_time_big', 'str_dist_medium', 'str_time_medium',
       'str_dist_big', 'str_time_big', 'bus_stops_per_pop',
       'train_stops_per_pop', 'other_stops_per_pop', 'comb_car_1000',
       'el_car_1000', 'inbound share %', 'outbound share %'],
      dtype='object')
```

[ ]:
```
# # my_conn.execute('''CREATE TABLE IF NOT EXISTS influence_factors (
#   BFS_Nr INT,
```

```
#   municipality VARCHAR(100),
#   canton VARCHAR(2),
#   language VARCHAR(2),
#   single_share FLOAT,
#   married_share FLOAT,
#   widowed_share FLOAT,
#   divorced_share FLOAT,
#   GA_share FLOAT,
#   HTA_share FLOAT,
#   FNT_share FLOAT,
#   pop_0_20_share FLOAT,
#   pop_20_40_share FLOAT,
#   pop_40_60_share FLOAT,
#   pop_60plus_share FLOAT,
#   birth_munic_share FLOAT,
#   birth_cant_share FLOAT,
#   birth_CH_share FLOAT,
#   birth_notCH_share FLOAT,
#   male_share FLOAT,
#   female_share FLOAT,
#   resid_0_1y_share FLOAT,
#   resid_1_5y_share FLOAT,
#   resid_6_10y_share FLOAT,
#   resid_10yplus_share FLOAT,
#   hh_1_share FLOAT,
#   hh_2_share FLOAT,
#   hh_3_5_share FLOAT,
#   hh_6plus_share FLOAT,
#   PT_dist_medium FLOAT,
#   PT_time_medium FLOAT,
#   PT_dist_big FLOAT,
#   PT_time_big FLOAT,
#   str_dist_medium FLOAT,
#   str_time_medium FLOAT,
#   str_dist_big FLOAT,
#   str_time_big FLOAT,
#   bus_stops_per_pop FLOAT,
#   train_stops_per_pop FLOAT,
#   other_stops_per_pop FLOAT,
#   comb_car_1000 FLOAT,
#   el_car_1000 FLOAT,
#   inbound_share FLOAT,
#   outbound_share FLOAT,
#   PRIMARY KEY (BFS_Nr)
# );''')
```

[ ]: <sqlalchemy.engine.cursor.LegacyCursorResult at 0x7f6af8ff0590>

### 10.1.3 Write tables to database

[ ]:
```
# town_dir_PLZ.to_sql("town_directory", my_conn, if_exists = 'replace',
#                     index=False)
```

[ ]:
```
# my_conn.execute("SELECT * FROM town_directory").fetchone()
```

[ ]:
```
# inf_fac_share.to_sql("influence_factors", my_conn, if_exists = 'replace',
#                     index=False)
```

[ ]:
```
# my_conn.execute("SELECT * FROM influence_factors").fetchone()
```

### 10.1.4 Convert notebook to pdf

[333]:
```
# !apt-get install texlive texlive-xetex texlive-latex-extra pandoc
# !pip install pypandoc
!jupyter nbconvert --to PDF "ETL_Influence_factors.ipynb"
```

```
[NbConvertApp] Converting notebook ETL_Influence_factors.ipynb to PDF
[NbConvertApp] Writing 549149 bytes to ./notebook.tex
[NbConvertApp] Building PDF
[NbConvertApp] Running xelatex 3 times: ['xelatex', './notebook.tex', '-quiet']
[NbConvertApp] Running bibtex 1 time: ['bibtex', './notebook']
[NbConvertApp] WARNING | bibtex had problems, most likely because there were no
citations
[NbConvertApp] PDF successfully created
[NbConvertApp] Writing 417875 bytes to ETL_Influence_factors.pdf
```