Data Report

Laura Conti Sabrina Popp Alex Leccadito John Truninger

2023-06-11

Table of contents

1 Raw Data

The raw data sets can all be found in the directory `data/01_raw`. Each data set is then located in its own folder, which includes a spreadsheet (`.xlsx`, `.xls`, `.csv` or `.ods`), as well as the URL to the source for downloading the data.

Number	Dataset Name	Source	Folder in data/01_raw
R01	Swiss Real Estate Offer Index	https://www.iazicifi.ch/	01_Swiss Real Estate Offer Index
R02	Distribution of population by tenure status	https://ec.europa.eu/	02_Distribution of population by tenure status
R03	Average sales price of new and existing residential property	https://www.statista.com/	03_Average price per square meter in european countries
R04	General overview “Buildings” by cantons 2021	https://www.bfs.admin.ch/	04_Allgemeine Übersicht Gebäude nach Kantonen 2021
R05	Structure of the permanent resident population by canton, 1999-2021	https://www.bfs.admin.ch/	05_Struktur der ständigen Wohnbevölkerung nach Kanton, 1999-2021
R06	Swiss Construction Price Index - Average unit prices in Switzerland and the major regions	https://www.bfs.admin.ch/	06_Schweizerischer Baupreisindex - Durchschnittliche Einheitspreise in der Schweiz und in den Grossregionen
R07	Taxes & Public Finances	https://www.statista.com/	07_Steuern_&_Staatsfinanzen

```python
from pathlib import Path
import numpy as np
import pandas as pd
import openpyxl

import warnings
warnings.filterwarnings('ignore', category=UserWarning)
warnings.filterwarnings('ignore', category=FutureWarning)
```

```python
data_path = Path('../data')
raw_data = Path(data_path, '01_raw')
prc_data = Path(data_path, '02_processed')
exp_data = Path(data_path, '03_exports')
```

1.1 Dataset R01: Swiss Real Estate Offer Index

1.1.1 Details Dataset

The source of this dataset consists of an excel file with multiple spreadsheets. Collectivly, it shows the monthly average price of real estate in selected regions of switzerland.

- Sheet: `Miet_Indizes`: This sheet contains the monthly averaged offer prices for rental appartments in selected regions of switherland.
  - Description of rows:
    - respective month index for the body of the dataset
  - Description of columns:
    - `t`: date of the index in format `dd.mm.yyyy`
    - `Eigenthumswohnungen`: prices for condominiums in switzerland
    - `Einfamilienhäuser`: prices for houses in switzerland
- Sheet: `Preis_Indizes`: This sheet contains the monthly averaged price offers for condominiumns and houses in switzerland.
  - Description of rows:
    - respective month index for the body of the dataset
  - Description of columns:
    - `t`: date of the index in format `dd.mm.yyyy`
    - `Eigenthumswohnungen`: prices for condominiums in switzerland
    - `Einfamilienhäuser`: prices for houses in switzerland

  This dataset was created by the company IAZI AG - CIFI SA and ImmoScout24. The Swiss Real Estate Offer Index is a real estate index family. They are the world's first hedonic indices that are calculated and updated in real time. Thanks to the continuous updating and their methodology, they allow the timely monitoring of the development of offer prices and offer rents on the largest real estate platform in Switzerland.

In the following steps, the data will be extracted from 2 sheets and in the end combined to create the raw dataset R01.

1.1.2 Data Quality

The data is already preprocessed which means, handling `NaN` values won't be necessairy. However since two sheet will be merged, a cut-off from a specific year is nescessairy since the two subdatasets do not have the same time range.

```
rDataSet_10 = Path(
    raw_data,
    '01_Swiss Real Estate Offer Index',
    'Swiss Real Estate Offer Index.xlsx'
)
```

1.1.3 Data Extraction & Cleansing

As before mentioned a new dataset will be created through extracting the specific sheets from the main dataset and combining them into one dataset.

Data extraction and reshaping `Miet_Indizes`

```
dataframe_10 = pd.read_excel(
    rDataSet_10,
    sheet_name='Miet_Indizes',
    skiprows=6,
    skipfooter=3,
)

# drop unused columns
dataframe_10 = dataframe_10.drop(
    dataframe_10.columns[2:],
    axis=1
)

# rename columns
dataframe_10 = dataframe_10.rename(
    columns={
        't': 'date',
        'CH': 'rental_index'
    }
)

# format column date to datetime
dataframe_10['date'] = pd.to_datetime(
                            dataframe_10['date'],
                            dayfirst=True
                       ).dt.date


# reset index & sort by date
dataframe_10.reset_index(drop=True, inplace=True)
dataframe_10.sort_values(by='date', inplace=True)

dataframe_10.head()
```

	date	rental_index
0	2014-12-31	260.82
1	2015-01-31	259.36
2	2015-02-28	258.83
3	2015-03-31	258.10
4	2015-04-30	261.57

Data extraction and reshaping `Preis_Indizes`

```
dataframe_11 = pd.read_excel(
    rDataSet_10,
    sheet_name='Preis_Indizes',
    skiprows=6,
    skipfooter=3,
)

# rename columns
dataframe_11 = dataframe_11.rename(
    columns={
        't': 'date',
        'Eigentumswohnungen': 'condos',
        'Einfamilienhäuser': 'houses'
    }
)

# format column date to datetime
dataframe_11['date'] = pd.to_datetime(
                            dataframe_11['date'],
                            dayfirst=True
                       ).dt.date

# reset index & sort by date
dataframe_11.reset_index(drop=True, inplace=True)
dataframe_11.sort_values(by='date', inplace=True)

dataframe_11.head()
```

	date	condos	houses
0	2010-12-31	5822.53	5458.33
1	2011-01-31	5852.88	5502.88
2	2011-02-28	5846.56	5509.60
3	2011-03-31	5964.04	5597.43
4	2011-04-30	5943.01	5594.06

1.1.4 Data merging

```
dataframe_10 = pd.merge(
    dataframe_10,
    dataframe_11,
    on='date'
)

dataframe_10.head()
```

	date	rental_index	condos	houses
0	2014-12-31	260.82	6898.10	5990.61
1	2015-01-31	259.36	6916.64	5964.14
2	2015-02-28	258.83	6894.75	5974.35
3	2015-03-31	258.10	6879.93	5988.68
4	2015-04-30	261.57	6901.96	5987.68

1.2 Dataset R02: Distribution of population by tenure status

1.2.1 Details Dataset

Collectivly, this presents the distribution of population by tenure status (tenure, owner) in selected european countries.

- Sheet: `Summary`: This sheet gives additional informationa bout the source and the contents of the dataset
- Sheet: `Structure`: The summary sheet gives additionaly information about the structure of the data.
- Sheet: `Sheet 1`: This sheet shows the distribution of population by tenure status, type of household and income group with the tenure type `owner`
  - Description of rows:
    - respective country index for the body of the dataset
  - Description of columns:
    - `2003 - 2022`: year of the data
- Sheet: `Sheet 2`: This sheet shows the distribution of population by tenure status, type of household and income group with the tenure type `tenure`

  Note: The relevant sheet for this task is `Sheet 1`

1.2.2 Data Quality

Although there are some NaN entries in the dataset, they wont be replaced or looked at differently. Instead, the visualisation will only cover reference a specific column where there are the least NaN values. Aditionally there are some countries whose names are abnormal. These will be renamed to match the rest of the data.

```
rDataSet_20 = Path(
    raw_data,
    '02_Distribution of population by tenure status',
    'ilc_lvho02__custom_6240818_spreadsheet.xlsx'
)
```

1.2.3 Data Extraction & Cleansing

```
dataframe_20 = pd.read_excel(
    rDataSet_20,
    sheet_name='Sheet 1',
    skiprows=11,
    skipfooter=3,
)

# keep only relevant columns
dataframe_20 = dataframe_20.iloc[:, [0, 16]]

# rename columns
dataframe_20 = dataframe_20.rename(
    columns={
        'GEO (Labels)': 'country',
        'Unnamed: 16': '2018'
    }
)

# rename row with Germany
dataframe_20 = dataframe_20.replace(
    'Germany (until 1990 former territory of the FRG)',
    'Germany'
)

# reset index
dataframe_20.reset_index(drop=True, inplace=True)


dataframe_20.head()
```

	country	2018
0	Belgium	72.3
1	Bulgaria	83.6
2	Czechia	78.7
3	Denmark	60.5
4	Germany	51.5

1.3 Dataset R03: Average price per square meter in european countries

1.3.1 Details Dataset

This dataset contains the monthly averaged price per square meter for owned real estate in selected european countries.

- Sheet: `Overview`: This sheet containes the additionaly information about the data and its source
- Sheet: `Data`: This sheet contains the monthly averaged price offers for condominiumns and houses in switzerland
  - Description of Rows:
    - `Country`: respective country
  - Description of Columns:
    - `New Housing`: average sales price for new built houses
    - `Existing Housing`: average sales price for existing houses

1.3.2 Data Quality

The data for this dataset was originally published by the company Deloitte in August 2022 in a pdf File. Afterwards the relevant data has been copied into a workable Excel file and uploaded to Statista. In the following steps an extraction of the sheet 'data' will be made. Since only one price is needed in the end, the average of the price for existing houses and new houses gets calculated. This also removes the problematic of the NaN Values because in that case only the existing value is kept.

```
rDataSet_30 = Path(
    raw_data,
    '03_Average price per square meter in european countries',
    'statistic_id722905_average-residential-real-estate-square-meter-prices-in-europe-2021-b
)
```

1.3.3 Data Extraction & Cleansing

```
dataframe_30 = pd.read_excel(
    rDataSet_30,
    sheet_name='Data',
    skiprows=4,
)

# drop unused columns
dataframe_30.drop(
    ['Unnamed: 0'],
    axis=1,
    inplace=True
)

# rename columns
dataframe_30 = dataframe_30.rename(
    columns={
        'Unnamed: 1': 'country',
        'New housing': 'new_housing',
        'Existing housing': 'existing_housing'
    }
)

# calculate average price
dataframe_30['avg_price'] = np.nanmean(
                                dataframe_30[['new_housing', 'existing_housing']],
                                axis=1
                            )

# drop unused columns
dataframe_30.drop(
    ['new_housing', 'existing_housing'],
    axis=1,
    inplace=True
)

dataframe_30.head()
```

	country	avg_price
0	UK	4235.0
1	Austria	4205.0
2	France	4012.0
3	Netherlands	3599.5
4	Norway	4286.0

1.4 Dataset R04: General overview “Buildings” by cantons 2021

1.4.1 Details Dataset

This dataset contains detailed information on the types of habitable buildings in switzerland, categorized by swiss cantons. Each sheet in this dataset represents a canton of switzerland, as well as a sheet for the entire country.

- Description of rows:
  - `Total`: contains different types of housings and buildings
  - `Bauperiode`: contains ranges of years when the houses were built
  - `Grossanzahl`: number of storeys of the buildings
  - `Energiequelle der Heizung`: energy source for heating
  - `Energiequelle für die Warmwasseraufbereitung`: energy source for warm water treatment
  - `Eigentümertyp`: type of ownership
- Description of columns:
  - `Total`: cumulative number of habitable buildings
  - `Bauperiode`: building periods
  - `Mit ... Wohnungen`: number of appartments within a building
- Body of the dataset:
  - The body of the dataset is filled with the number of buildings that satisfy both, the row and column conditions.

1.4.2 Data Quality

The files provided by the BFS (Bundesamt für Statistik, Federal Statistics Office) are already cleaned datasets. Therefore, handling of NaN values will mostly not be necessary. The current dataset exhibits some cells without values, but this is due to yearly numbers being present in the rows and the columns. This leads to values along the diagonal and NaN values in the triangles above and below the diagonal. The main objective of processing this dataset is to extract the necessary columns and rows for later analysis and visualizations.

```
rDataSet_40 = Path(
    raw_data,
    '04_Allgemeine Übersicht Gebäude nach Kantonen 2021',
    'je-d-09.02.00-2021.xlsx'
)
```

1.4.3 Data Extraction & Cleansing

```
wb_40 = openpyxl.load_workbook(rDataSet_40, read_only=True)
sheets_40 = wb_40.sheetnames


def buildings_extraction(path: str, sheet: str, df_out: pd.DataFrame) -> None:
    """
    Extracts the total number of buildings and single family homes for
    a given sheet (canton)
    :param path: path to the excel file
    :param sheet: sheet name
    :param df_out: dataframe to store the extracted data
    :return: None
    """

    df_temp = pd.read_excel(io=path, sheet_name=sheet, skiprows=4)
    canton = sheet[-2:]
    total = int(df_temp.iloc[0, 1])
    single_house = int(df_temp.iloc[2, 1])

    df_out.loc[canton] = [total, single_house]

dataframe_40 = pd.DataFrame(
    columns=['buildings_total', 'single_family_home']
)

for sheet in sheets_40:
    buildings_extraction(
        path=rDataSet_40,
        sheet=sheet,
        df_out=dataframe_40
    )

dataframe_40 = dataframe_40.iloc[1:].copy()
dataframe_40.index.name = 'canton'

dataframe_40 = dataframe_40.reset_index()
dataframe_40 = dataframe_40.sort_values(by='canton')

dataframe_40.head()
```

	canton	buildings_total	single_family_home
18	AG	153894	102206
15	AI	5299	2917
14	AR	16323	9251
1	BE	238111	114053
12	BL	67390	46632

1.5 Dataset R05: Structure of the permanent resident population by canton, 1999-2021

1.5.1 Details Dataset

This dataset contains information about the population size in each canton of switzerland.

- Description of rows:
  - The rows represent the different cantons of Switzerland by regions
- Description of columns:

  - The columns categorize the population size by age, sex, nationality (Swiss or non-Swiss), civil status and the typology of the residential area
  - `Total`: aggregates the data from all other columns

- Body of the dataset:
  - The body of the dataset contains the population count that satisfies both, the row and column conditions.

1.5.2 Data Quality

As previously mentioned, the datasets from the BFS are known for their high data quality, requiring primarily the extraction of relevant information.

```
rDataSet_50 = Path(
    raw_data,
    '05_Struktur der ständigen Wohnbevölkerung nach Kanton, 1999-2021',
    'je-d-01.02.03.04.xlsx'
)
```

1.5.3 Data Extraction & Cleansing

```
dataframe_50 = pd.read_excel(
    io=rDataSet_50,
    sheet_name='2021',
    skiprows=4,
)

# selecting and renaming relevant columns
dataframe_50 = dataframe_50[['Unnamed: 0', 'Unnamed: 1']].copy()
dataframe_50.rename(
    inplace=True,
    columns={
        'Unnamed: 0': 'canton',
        'Unnamed: 1': 'inhabitants'
    },
)

# select canton rows (present in dataframe_40)
dataframe_50 = dataframe_50[dataframe_50['canton'].isin(dataframe_40['canton'])]

dataframe_50.head()
```

	canton	inhabitants
2	VD	822968.0
3	VS	353209.0
4	GE	509448.0
6	BE	1047473.0
7	FR	329809.0

1.6 Dataset R06: Swiss Construction Price Index - Average unit prices in Switzerland and the major regions

1.6.1 Details Dataset

This dataset contains detailed information about pricing in the building industry across different regions of Switzerland. The columns represent the different regions, while the rows contain various segments of the building process, such as attic, framework, flooring, and more.

- Description of rows:
  - The rows contain different categories and specifications of construction activities.
- Description of columns:
  - The columns represent the unit and quantity of construction activities for the different regions of Switzerland.
- Body of the Dataset:
  - The body of the dataset is filled with the number of inhabitants that satisfy both the row and column conditions.

1.6.2 Data Quality

As mentioned previously, the datasets from BFS are very clean. Therefore, this part mainly involves extracting the relevant data. Since this dataset contains macros, a copy was created with the suffix '_raw' for convenience. Some rows consist of NaN (Not a Number) values, which is a result of the dataset structure. Additionally, some values are missing for the region of Ticino. This information is provided at the end of the Excel sheet as a comment: "Drei Punkte (...) bedeuten, dass der Wert nicht vorhanden, nicht genügend repräsentativ oder unter Datenschutz ist." (EN: Three dots (...) mean that the value is not present, not sufficiently representative, or under data protection.)

```
rDataSet_60 = Path(
    raw_data,
    '06_Schweizerischer Baupreisindex - Durchschnittliche Einheitspreise in der Schweiz und
    'cc-t-05.05.02.xlsx'
)
```

1.6.3 Data Extraction & Cleansing

```
dataframe_60 = pd.read_excel(
    io=rDataSet_60,
    sheet_name='BAP_PCO',
    skiprows=9,
)

# drop unused columns
dataframe_60 = dataframe_60.drop(
    columns=[
        'Unnamed: 0',
        'Unnamed: 1',
        'Unnamed: 2',
        'Einheit',
        'Menge',
        'Unnamed: 4',
        'Unnamed: 15',
        'Unnamed: 16'
    ]
)

# rename columns
dataframe_60 = dataframe_60.rename(
    columns={
        'NPK Position': 'Category',
        'Schweiz': 'Switzerland',
        'Genferseeregion (VD,VS,GE)': 'Lake Geneva Region',
        'Espace Mittelland (BE, FR, SO, NE, JU)': 'Espace Midland',
        'Nordwestschweiz (BS, BL, AG)': 'Northwestern Switzerland',
        'Zürich (ZH)': 'Zurich',
        'Ostschweiz (GL, SH, AR, AI, SG, GR, TG)': 'Eastern Switzerland',
        'Zentralschweiz (LU, UR, SZ, OW, NW, ZG)': 'Central Switzerland',
        'Tessin (TI)': 'Ticino'
    }
)

def assignCategory(data: pd.DataFrame) -> pd.DataFrame:
    categories = []
    for index, row in data.iterrows():
        if type(row['Category']) != float:
            category = ' '.join(row['Category'].split(' ')[1:])
        categories.append(category)

    current_category = 'none'
    for i in range(len(categories)):
        if categories[i] != '':
            current_category = categories[i]
        categories[i] = current_category

    data['Category'] = categories
    return data

dataframe_61 = assignCategory(dataframe_60)
dataframe_61.head()
```

	Category	Switzerland	Lake Geneva Region	Espace Midland	Northwestern Switzerland	Zurich
0	Erdarbeiten	NaN	NaN	NaN	NaN	NaN
1	Erdarbeiten	6.5806	9.649058	5.1200	5.0588	5.8022
2	Erdarbeiten	6.3088	8.066795	5.2969	5.5250	6.2000
3	Erdarbeiten	52.3058	55.199414	47.0450	46.8138	52.0889
4	Erdarbeiten	12.4308	17.165547	11.8777	10.9088	11.7300

1.6.4 Handling NaN values

```
dataframe_61 = dataframe_61.replace('...', np.NaN)
dataframe_61 = dataframe_61.dropna()

print(dataframe_61.isna().sum())
print(dataframe_61.dtypes)
```

```
Category                    0
Switzerland                 0
Lake Geneva Region          0
Espace Midland              0
Northwestern Switzerland    0
Zurich                      0
Eastern Switzerland         0
Central Switzerland         0
Ticino                      0
dtype: int64
Category                     object
Switzerland                 float64
Lake Geneva Region          float64
Espace Midland              float64
Northwestern Switzerland    float64
Zurich                      float64
Eastern Switzerland         float64
Central Switzerland         float64
Ticino                      float64
dtype: object
```

1.6.5 Normalization

```
dataframe_62 = dataframe_61.groupby('Category').sum()

def normalize_rows(data: pd.DataFrame) -> pd.DataFrame:
    regions = list(data.columns)
    categories = data.index
    data2 = pd.DataFrame(columns = regions)

    for index, row in data.iterrows():
        row_list_floats = [float(x) for x in row]
        ch = float(row['Switzerland'])
        normalized_row = [x * 100 / ch - 100 for x in row_list_floats]
        data2.loc[len(data2.index)] = normalized_row
    data2['Category'] = categories
    data2 = data2.set_index('Category')
    return data2

dataframe_63 = normalize_rows(dataframe_62)
dataframe_63 = dataframe_63.drop(columns=['Switzerland'])

dataframe_63.head()
```

	Lake Geneva Region	Espace Midland	Northwestern Switzerland	Zurich
Category				
Allgemeine Schreinerarbeiten	9.782592	-5.054522	-5.332594	-1.278220
Aufzüge	-1.834763	5.337382	-1.834763	-1.834763
Baureinigung	12.377640	10.988312	-0.829673	-8.518666
Bodenbeläge	6.990752	-4.967931	-3.518762	2.796378
Dichtungsbeläge	6.729232	-5.094807	1.559295	-6.350533

Sorting columns by the most positive values (above swiss average)

```
dataframe_63_T = dataframe_63.T
dataframe_63_T['POS_COUNT'] = dataframe_63_T.select_dtypes(include='float64').gt(0).sum(axis

dataframe_64 = dataframe_63_T.sort_values(by='POS_COUNT', ascending=False)
dataframe_64 = dataframe_64.drop(columns=['POS_COUNT'])

dataframe_64.head()
```

Category	Allgemeine Schreinerarbeiten	Aufzüge	Baureinigung	Bodenbeläge	Dichtung
Lake Geneva Region	9.782592	-1.834763	12.377640	6.990752	6
Ticino	10.482749	-0.301372	6.612687	12.403766	10
Espace Midland	-5.054522	5.337382	10.988312	-4.967931	-5
Central Switzerland	-2.317067	1.768710	-12.249670	1.671983	-5
Northwestern Switzerland	-5.332594	-1.834763	-0.829673	-3.518762	1

1.6.6 English translation mapping

```
translation_dict = {
    'Allgemeine Schreinerarbeiten': 'General carpentry work',
    'Aufzüge': 'Elevators',
    'Baureinigung': 'Construction cleaning',
    'Bodenbeläge': 'Floor coverings',
    'Dichtungsbeläge': 'Sealing coatings',
    'Erdarbeiten': 'Earthworks',
    'Estriche ': 'Attic',
    'Fenster': 'Windows',
    'Gerüste': 'Scaffolding',
    'Gipserarbeiten': 'Plastering work',
    'Gärtnerarbeiten': 'Gardening work',
    'Kücheneinrichtungen': 'Kitchen fittings',
    'Malerarbeiten': 'Painting work',
    'Maurer- und Stahlbetonarbeiten': 'Masonry and reinforced concrete work',
    'Metallbauarbeiten': 'Metal construction work',
    'Plattenarbeiten': 'Tiling work',
    'Spenglerarbeiten': 'Sheet metal work',
    'Tiefbauarbeiten': 'Civil engineering works',
    'Zimmerarbeiten': 'Carpentry work'
}

dataframe_64 = dataframe_64.rename(columns=translation_dict)
```

```
dataframe_64['region'] = dataframe_64.index
dataframe_64.reset_index(drop=True, inplace=True)
dataframe_64.head()
```

Category	General carpentry work	Elevators	Construction cleaning	Floor coverings	Sealing coatings	E
0	9.782592	-1.834763	12.377640	6.990752	6.729232	
1	10.482749	-0.301372	6.612687	12.403766	10.569095	
2	-5.054522	5.337382	10.988312	-4.967931	-5.094807	-
3	-2.317067	1.768710	-12.249670	1.671983	-5.195145	
4	-5.332594	-1.834763	-0.829673	-3.518762	1.559295	-

1.7 Dataset R7: Taxes & Public Finances

1.7.1 Details Dataset

The source of these comprehensive data is Statista, renowned for its quality and reliability. Researchers, analysts, and interested individuals can rely on this data to conduct informed studies and analyses.

The files provided by Statista are preprocessed datasets, typically requiring minimal treatment of missing values or NaNs. The current datasets are cleaned.

The primary objective of processing this dataset is to extract the necessary columns and rows for subsequent analysis and visualization.

- File `Schweiz - Einkommenssteuersätze nach Kantonen 2022_Statista.csv`:
  - Description of columns:
    - `KANTON`: represents the canton names of Switzerland
    - `PROZENT`: represents the income tax rates in percentage
- File `Schweiz - Gewinnsteuersätze nach Kantonen 2022 _ Statista.csv`
  - Description of columns:
    - `KANTON`: represents the canton names of Switzerland
    - `PROZENT`: represents the profit tax rates in percentage
- File `Schweiz - Vermögenssteuersätze nach Kantonen 2018 _ Statista.csv`
  - Description of columns:
    - `KANTON`: represents the canton names of Switzerland
    - `PROZENT`: represents the wealth tax rates in promille

```
rDataSet_70 = Path(
    raw_data,
    '07_Steuern_&_Staatsfinanzen',
    'Schweiz - Einkommenssteuersätze nach Kantonen 2022 _ Statista.csv'
)
rDataSet_71 = Path(
    raw_data,
    '07_Steuern_&_Staatsfinanzen',
    'Schweiz - Gewinnsteuersätze nach Kantonen 2022 _ Statista.csv'
)
rDataSet_72 = Path(
    raw_data,
    '07_Steuern_&_Staatsfinanzen',
    'Schweiz - Vermögenssteuersätze nach Kantonen 2018 _ Statista.csv'
)
```

1.7.2 Data Extraction & Cleansing

```
dataframe_70 = pd.read_csv(
    rDataSet_70,
    sep=';',
    encoding='latin1'
)

dataframe_71 = pd.read_csv(
    rDataSet_71,
    sep=';',
    encoding='latin1'
)

dataframe_72 = pd.read_csv(
    rDataSet_72,
    sep=';',
    encoding='latin1'
)

# join dataframes
dataframe_73 = pd.merge(
    dataframe_70,
    dataframe_71,
    on='KANTON'
)

dataframe_73 = pd.merge(
    dataframe_73,
    dataframe_72,
    on='KANTON'
)

# rename columns
dataframe_73 = dataframe_73.rename(
    columns={
        'KANTON': 'canton',
        'PROZENT_x': 'income_tax',
        'PROZENT_y': 'profit_tax',
        'Promille': 'wealth_tax'
    }
)

dataframe_73.head()
```

	canton	income_tax	profit_tax	wealth_tax
0	Genf	44.75	14.00	10.1
1	Basel-Landschaft	42.17	17.97	7.6
2	Waadt	41.50	14.00	7.9
3	Bern	41.04	21.04	5.8
4	Basel-Stadt	40.34	13.04	8.0

2 Processed Data

The processed data sets can all be found in the directory `data/02_processed`.

```
# save R01 to csv
dataframe_10.to_csv(
    Path(prc_data, 'P01_price_indices.csv'),
    index=False
)

# save R02 to csv
dataframe_20.to_csv(
    Path(prc_data, 'P02_ratio_homeowners_eu.csv'),
    index=False
)

# save R03 to csv
dataframe_30.to_csv(
    Path(prc_data, 'P03_avg_price_smeter.csv'),
    index=False
)

# save R04 to csv
dataframe_40.to_csv(
    Path(prc_data, 'P04_buildings_by_canton.csv'),
    index=False
)

# save R05 to csv
dataframe_50.to_csv(
    Path(prc_data, 'P05_population_by_canton.csv'),
    index=False
)

# save R06 to csv
dataframe_64.to_csv(
    Path(prc_data, 'P06_construction_prices.csv'),
    index=False
)

# save R07 to csv
dataframe_73.to_csv(
    Path(prc_data, 'P07_taxes.csv'),
    index=False
)
```

Number	Dataset Name	Source
P01	P01_price_indices.csv	R01
P02	P02_ratio_homeowners_eu.csv	R02
P03	P03_avg_price_smeter.csv	R03
P04	P04_buildings_by_canton.csv	R04
P05	P05_population_by_canton.csv	R05
P06	P06_construction_prices.csv	R06
P07	P07_taxes.csv	R07

3 Exploratory Data Analysis

3.1 EDA P01: Price Indices

```
dataframe_10 = pd.read_csv(
    Path(prc_data, 'P01_price_indices.csv')
)

display(dataframe_10.describe())
dataframe_10.info()
```

	rental_index	condos	houses
count	101.000000	101.000000	101.000000
mean	261.415050	7392.592772	6478.379406
std	3.221622	518.387040	495.529958
min	256.620000	6856.950000	5940.580000
25%	259.100000	7024.970000	6057.130000
50%	260.820000	7157.740000	6315.930000
75%	263.460000	7617.250000	6786.410000
max	271.970000	8589.250000	7479.460000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 101 entries, 0 to 100
Data columns (total 4 columns):
 #   Column        Non-Null Count  Dtype  
---  ------        --------------  -----  
 0   date          101 non-null    object 
 1   rental_index  101 non-null    float64
 2   condos        101 non-null    float64
 3   houses        101 non-null    float64
dtypes: float64(3), object(1)
memory usage: 3.3+ KB
```

3.2 EDA P02: Ratio Homeowners EU

```
dataframe_20 = pd.read_csv(
    Path(prc_data, 'P02_ratio_homeowners_eu.csv')
)

display(dataframe_20.describe())
dataframe_20.info()
```

	2018
count	36.000000
mean	75.527778
std	12.510170
min	42.500000
25%	69.800000
50%	74.800000
75%	84.100000
max	96.400000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 36 entries, 0 to 35
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   country  36 non-null     object 
 1   2018     36 non-null     float64
dtypes: float64(1), object(1)
memory usage: 704.0+ bytes
```

3.3 EDA P03: Average Price per Square Meter

```
dataframe_30 = pd.read_csv(
    Path(prc_data, 'P03_avg_price_smeter.csv')
)

display(dataframe_30.describe())
dataframe_30.info()
```

	avg_price
count	18.000000
mean	2841.213889
std	1518.824498
min	1246.500000
25%	1687.625000
50%	2377.000000
75%	3908.875000
max	7126.350000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18 entries, 0 to 17
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype  
---  ------     --------------  -----  
 0   country    18 non-null     object 
 1   avg_price  18 non-null     float64
dtypes: float64(1), object(1)
memory usage: 416.0+ bytes
```

3.4 EDA P04: Buildings by Canton

```
dataframe_40 = pd.read_csv(
    Path(prc_data, 'P04_buildings_by_canton.csv')
)

display(dataframe_40.describe())
dataframe_40.info()
```

	buildings_total	single_family_home
count	26.000000	26.000000
mean	68236.961538	38733.923077
std	64860.525980	35848.970768
min	5299.000000	2686.000000
25%	17153.000000	9472.250000
50%	55635.500000	29065.500000
75%	103385.750000	62761.500000
max	238111.000000	118612.000000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 3 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   canton              26 non-null     object
 1   buildings_total     26 non-null     int64 
 2   single_family_home  26 non-null     int64 
dtypes: int64(2), object(1)
memory usage: 752.0+ bytes
```

3.5 EDA P05: Population by Canton

```
dataframe_50 = pd.read_csv(
    Path(prc_data, 'P05_population_by_canton.csv')
)

display(dataframe_50.describe())
dataframe_50.info()
```

	inhabitants
count	2.600000e+01
mean	3.361073e+05
std	3.613990e+05
min	1.636000e+04
25%	7.634725e+04
50%	2.408105e+05
75%	4.035468e+05
max	1.564662e+06

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 2 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   canton       26 non-null     object 
 1   inhabitants  26 non-null     float64
dtypes: float64(1), object(1)
memory usage: 544.0+ bytes
```

3.6 EDA P06: Construction Prices

```
dataframe_60 = pd.read_csv(
    Path(prc_data, 'P06_construction_prices.csv')
)

display(dataframe_60.describe())
dataframe_60.info()
```

	General carpentry work	Elevators	Construction cleaning	Floor coverings	Sealing coatings	Eartl
count	7.000000	7.000000	7.000000	7.000000	7.000000	7.(
mean	0.336175	-0.076333	-1.243108	0.801317	0.722615	-0.4
std	6.846241	2.742609	11.704542	7.542107	6.546152	10.4
min	-5.332594	-1.834763	-17.082390	-9.766969	-6.350533	-12.(
25%	-4.492118	-1.834763	-10.384168	-4.243347	-5.144976	-6.8
50%	-2.317067	-1.834763	-0.829673	1.671983	1.559295	-2.1
75%	4.252186	0.733669	8.800500	4.893565	4.785198	3.(
max	10.482749	5.337382	12.377640	12.403766	10.569095	18.8

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7 entries, 0 to 6
Data columns (total 20 columns):
 #   Column                                Non-Null Count  Dtype  
---  ------                                --------------  -----  
 0   General carpentry work                7 non-null      float64
 1   Elevators                             7 non-null      float64
 2   Construction cleaning                 7 non-null      float64
 3   Floor coverings                       7 non-null      float64
 4   Sealing coatings                      7 non-null      float64
 5   Earthworks                            7 non-null      float64
 6   Attic                                 7 non-null      float64
 7   Windows                               7 non-null      float64
 8   Scaffolding                           7 non-null      float64
 9   Plastering work                       7 non-null      float64
 10  Gardening work                        7 non-null      float64
 11  Kitchen fittings                      7 non-null      float64
 12  Painting work                         7 non-null      float64
 13  Masonry and reinforced concrete work  7 non-null      float64
 14  Metal construction work               7 non-null      float64
 15  Tiling work                           7 non-null      float64
 16  Sheet metal work                      7 non-null      float64
 17  Civil engineering works                  7 non-null      float64
 18  Carpentry work                           7 non-null      float64
 19  region                                   7 non-null      object
dtypes: float64(19), object(1)
memory usage: 1.2+ KB
```

3.7 EDA P07: Taxes

```
dataframe_70 = pd.read_csv(
    Path(prc_data, 'P07_taxes.csv')
)

display(dataframe_70.describe())
dataframe_70.info()
```

	income_tax	profit_tax	wealth_tax
count	26.000000	26.000000	26.000000
mean	33.516154	14.683462	4.665385
std	6.507109	2.548612	2.353711
min	22.220000	11.850000	1.300000
25%	30.015000	12.815000	2.650000
50%	33.250000	13.935000	4.400000
75%	39.447500	15.822500	6.250000
max	44.750000	21.040000	10.100000

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26 entries, 0 to 25
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   canton      26 non-null     object
 1   income_tax  26 non-null     float64
 2   profit_tax  26 non-null     float64
 3   wealth_tax  26 non-null     float64
dtypes: float64(3), object(1)
memory usage: 960.0+ bytes
```

4 Preparation for Visualization

To use the data correctly in our visualizations, we exported the specific needed data to `.js` constants variables. These files can be found in the directory `data/03_exports`.

Chart Name	Used Dataframes	Exported File
S1	P02	s1Data.js
S2	P03	s2Data.js
S3	P01	s3Data.js
S4	P04, P05	s4Data.js
S5	P06	s5Data.js
S6	P07	s6Data.js

Export Function

```
def generate_data_js(df: pd.DataFrame, js_file_name: str, variable_name: str, orient: str) -
    """
    Generates a JS file with a variable name and the data from the dataframe
    -------------------------------------------------------------------------
    :param df: dataframe to export
    :param js_file_name: name of the JS file
    :param variable_name: name of the variable
    :param orient: orientation of the dataframe
    :return: None
    """
    json_data = df.to_json(orient=orient, force_ascii=False)
    with open(Path(exp_data, js_file_name), 'w', encoding='utf-8') as f:
        f.write(f"export const {variable_name} = {json_data};")
```

4.1 S1 Chart

```
s1_country_filter = [
    'Switzerland',
    'Norway',
    'United Kingdom',
    'Austria',
    'France',
    'Netherlands',
    'Denmark',
    'Czechia',
    'Slovenia',
    'Belgium',
    'Spain',
    'Italy',
    'Croatia',
    'Poland',
    'Hungary',
    'Portugal',
    'Serbia',
    'Latvia',
]

s1_data = pd.read_csv(
    Path(prc_data, 'P02_ratio_homeowners_eu.csv')
)

# filter out countries
s1_data = s1_data[s1_data['country'].isin(s1_country_filter)]

# sort data descending
s1_data = s1_data.sort_values(by='2018', ascending=False)

# export data
generate_data_js(
    df=s1_data,
    js_file_name='s1Data.js',
    variable_name='s1Data',
    orient='records'
)

s1_data.head()
```

	country	2018
10	Croatia	90.1
16	Hungary	86.0
34	Serbia	84.4
20	Poland	84.0
13	Latvia	81.6

4.2 S2 Chart

```
s2_data = pd.read_csv(
    Path(prc_data, 'P03_avg_price_smeter.csv')
)

# sort data descending
s2_data = s2_data.sort_values(by='avg_price', ascending=False)

# export data
generate_data_js(
    df=s2_data,
    js_file_name='s2Data.js',
    variable_name='s2Data',
    orient='records'
)

s2_data.head()
```

	country	avg_price
17	Switzerland	7126.35
4	Norway	4286.00
0	UK	4235.00
1	Austria	4205.00
2	France	4012.00

4.3 S3 Chart

```
s3_data = pd.read_csv(
    Path(prc_data, 'P01_price_indices.csv')
)

# sort data descending
s3_data = s3_data.sort_values(by='date', ascending=True)

# filter out data before 2020-01-01
s3_data = s3_data[s3_data['date'] >= '2020-01-01']

# calculate relative differences
s3_data['prc_rental'] = (100/258.26)*s3_data['rental_index']
s3_data['prc_condos'] = (100/7157.74)*s3_data['condos']
s3_data['prc_houses'] = (100/6308.97)*s3_data['houses']

# export data
generate_data_js(
    df=s3_data,
    js_file_name='s3Data.js',
    variable_name='s3Data',
    orient='records'
)

s3_data.head()
```

	date	rental_index	condos	houses	prc_rental	prc_condos	prc_houses
61	2020-01-31	258.26	7157.74	6308.97	100.000000	100.000000	100.000000
62	2020-02-29	258.80	7245.38	6316.64	100.209092	101.224409	100.121573
63	2020-03-31	258.43	7366.09	6398.48	100.065825	102.910835	101.418774
64	2020-04-30	256.67	7383.67	6384.32	99.384341	103.156443	101.194331
65	2020-05-31	258.01	7342.29	6434.78	99.903198	102.578328	101.994145

4.4 S4 Chart

```
region_mapping_s4 = {
    'VD': 'Lake Geneva Region',
    'VS': 'Lake Geneva Region',
    'GE': 'Lake Geneva Region',
    'BE': 'Espace Midland',
    'FR': 'Espace Midland',
    'SO': 'Espace Midland',
    'NE': 'Espace Midland',
    'JU': 'Espace Midland',
    'BS': 'Northwestern Switzerland',
    'BL': 'Northwestern Switzerland',
    'AG': 'Northwestern Switzerland',
    'ZH': 'Zurich',
    'GL': 'Eastern Switzerland',
    'SH': 'Eastern Switzerland',
    'AR': 'Eastern Switzerland',
    'AI': 'Eastern Switzerland',
    'SG': 'Eastern Switzerland',
    'GR': 'Eastern Switzerland',
    'TG': 'Eastern Switzerland',
    'LU': 'Central Switzerland',
    'UR': 'Central Switzerland',
    'SZ': 'Central Switzerland',
    'OW': 'Central Switzerland',
    'NW': 'Central Switzerland',
    'ZG': 'Central Switzerland',
    'TI': 'Ticino'
}

s4_data_1 = pd.read_csv(
    Path(prc_data, 'P04_buildings_by_canton.csv')
)

s4_data_2 = pd.read_csv(
    Path(prc_data, 'P05_population_by_canton.csv')
)

# merge dataframes
s4_data = pd.merge(
    s4_data_1,
    s4_data_2,
    on='canton'
)

s4_data['prc_single_family_home'] = s4_data['single_family_home'] / s4_data['buildings_total
s4_data['region'] = s4_data['canton'].map(region_mapping_s4)

# export data
generate_data_js(
    df=s4_data,
    js_file_name='s4Data.js',
    variable_name='s4Data',
    orient='records'
)

s4_data.head()
```

	canton	buildings_total	single_family_home	inhabitants	prc_single_family_home	region
0	AG	153894	102206	703086.0	66.413245	Northwestern Sw
1	AI	5299	2917	16360.0	55.048122	Eastern Switzerl
2	AR	16323	9251	55585.0	56.674631	Eastern Switzerl
3	BE	238111	114053	1047473.0	47.899089	Espace Midland
4	BL	67390	46632	292817.0	69.197210	Northwestern Sw

4.5 S5 Chart

```
s5_data = pd.read_csv(
    Path(prc_data, 'P06_construction_prices.csv')
)

# move last column to first position
cols = list(s5_data.columns)
cols = [cols[-1]] + cols[:-1]
s5_data = s5_data[cols]

# transform dataframe
regions = s5_data['region'].to_list()
s5_data = s5_data.T
s5_data.columns = regions

# reset & rename index
s5_data.reset_index(inplace=True)
s5_data.rename(columns={'index': 'work_category'}, inplace=True)

# drop first row
s5_data = s5_data.iloc[1:].copy()

# export data
generate_data_js(
    df=s5_data,
    js_file_name='s5Data.js',
    variable_name='s5Data',
    orient='records'
)

s5_data.head()
```

	work_category	Lake Geneva Region	Ticino	Espace Midland	Central Switzerland	North
1	General carpentry work	9.782592	10.482749	-5.054522	-2.317067	-5.332
2	Elevators	-1.834763	-0.301372	5.337382	1.76871	-1.834
3	Construction cleaning	12.37764	6.612687	10.988312	-12.24967	-0.829
4	Floor coverings	6.990752	12.403766	-4.967931	1.671983	-3.518
5	Sealing coatings	6.729232	10.569095	-5.094807	-5.195145	1.5592

4.6 S6 Chart

```
region_mapping_s6 = {
    'Genf': 'Lake Geneva Region',
    'Waadt': 'Lake Geneva Region',
    'Wallis': 'Lake Geneva Region',
    'Bern': 'Espace Midland',
    'Freiburg': 'Espace Midland',
    'Solothurn': 'Espace Midland',
    'Neuenburg': 'Espace Midland',
    'Jura': 'Espace Midland',
    'Basel-Stadt': 'Northwestern Switzerland',
    'Basel-Landschaft': 'Northwestern Switzerland',
    'Aargau': 'Northwestern Switzerland',
    'Zürich': 'Zurich',
    'Glarus': 'Eastern Switzerland',
    'Schaffhausen': 'Eastern Switzerland',
    'Appenzell Ausserrhoden': 'Eastern Switzerland',
    'Appenzell Innerrhoden': 'Eastern Switzerland',
    'St. Gallen': 'Eastern Switzerland',
    'Graubünden': 'Eastern Switzerland',
    'Thurgau': 'Eastern Switzerland',
    'Luzern': 'Central Switzerland',
    'Uri': 'Central Switzerland',
    'Schwyz': 'Central Switzerland',
    'Obwalden': 'Central Switzerland',
    'Nidwalden': 'Central Switzerland',
    'Zug': 'Central Switzerland',
    'Tessin': 'Ticino'
}

s6_data = pd.read_csv(
    Path(prc_data, 'P07_taxes.csv',
    encoding='latin1')
)

# sort data descending
s6_data = s6_data.sort_values(by='canton', ascending=False)
s6_data['region'] = s6_data['canton'].map(region_mapping_s6)

# group data by region
s6_data = s6_data.groupby('region').mean().reset_index()

# export data
```

```
generate_data_js(
    df=s6_data,
    js_file_name='s6Data.js',
    variable_name='s6Data',
    orient='records'
)

s6_data.head()
```

	region	income_tax	profit_tax	wealth_tax
0	Central Switzerland	25.516667	12.575000	2.066667
1	Eastern Switzerland	30.308571	13.455714	3.614286
2	Espace Midland	37.384000	15.954000	5.280000
3	Lake Geneva Region	40.916667	15.040000	8.100000
4	Northwestern Switzerland	38.963333	16.143333	6.666667