Semesterarbeit des Moduls

Data Collection, Integration and Pre-processing (CIP02)

ETL-Projektdokumentation

zum Thema

Entwicklung von Sportgrossanlässen und Bruttoinlandprodukt sowie deren Korrelation von 1950-2019

Hochschule Luzern HSLU

Studiengang: MSc Applied Information and Data Science

Studenten: Régis Andréoli, Micha Käser

Einreichdatum: 19.05.2021

Gruppennummer: 50

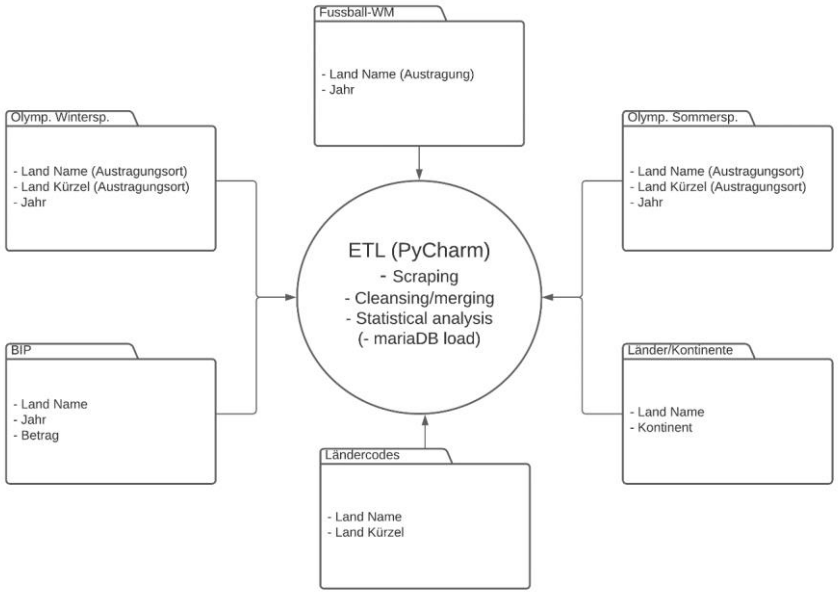
Inhaltsverzeichnis

1 Ausgangslage

Während des Masterstudiengangs Applied Information and Data Science erarbeiten die Studierenden innerhalb des Moduls CIP02 eine Semesterarbeit, welche die erlernten Extract-, Transform- und Load-Methoden der Vorlesung in die Praxis umsetzt. Ursprünglich zu dritt gestartet, brach ein Mitstudent das Studium ab – der knappen Zeit geschuldet setzen Régis Andréoli und Micha Käser das Projekt nun zu zweit um.

Als grosse Sportenthusiasten lag es nahe, ein entsprechendes Thema zu wählen. Damit auch eine gewisse Zeitachse (Time Series) gewährleistet ist, fiel die Wahl schlussendlich auf Sportgrossanlässe (Sommer- und Winterolympiaden/Fussballweltmeisterschaften) und deren Auswirkungen auf das Bruttoinlandprodukt (BIP) der jeweiligen austragenden Nation. Solche Anlässe sind gut dokumentiert und lassen sich viele Jahre zurückverfolgen, auch das BIP als wichtige wirtschaftliche Kennzahl lässt sich bis zur Industrialisierung nachvollziehen. Damit die Resultate verlässlich und aussagekräftig sind, werden die BIP-relevanten Daten allerdings erst ab der Nachkriegszeit (1950) bis 2019 ausgewertet.

Das nachfolgende Kontextdiagramm zeigt die sechs verschiedenen Datenquellen, die zur Beantwortung der Fragestellungen benötigt werden. Bis auf die Datenquellen «BIP» und «Ländercodes» wurden alle mit Python gescraped.



2 Datenquellen

ID	Daten	Attribute	Link
A1[1]	BIP nach Land [2] 1950-2019	• Land Name • Jahr • Betrag	https://febpwt.web-hosting.rug.nl/Dmn/AggregateXs/PivotShow
B1	Fussball-Weltmeisterschaften 1930-2018	• Land Name Austragungsort • Jahr	https://www.fussball-wm-total.de/History/histehre.html

[1] CSV-Export
[2] Tabelle unter Quickstart Query «RGDPNA (1950-2019)»

B2	Olympische Winterspiele 1924-2018	• Land Name Austragungsort • Land Kürzel • Jahr	https://www.taschen-hirn.de/sport/olympische-winterspiele/
B3	Olympische Sommerspiele 1896-2016	• Land Name Austragungsort • Land Kürzel • Jahr	https://www.taschen-hirn.de/sport/olympische-sommerspiele/
C1	Länder und Kontinente	• Land Name • Kontinent	https://www.bernhard-gaul.de/wissen/staaten-erde.php#uebneu
C2	Ländercodes[3]	• Land Name • Land Kürzel	https://www.laenderdaten.de/kuerzel/iso_3166-1.aspx

3 Tools

Zweck	**Tools**
Repository Code	GitHub (https://github.com/michakaeser/CIP_Project)
Repository Daten	GitHub/Switch Drive
Kommunikation	Slack, Teams
Entwicklungsumgebung	Ubuntu (VM oder nativ), PyCharm, mariaDB, Microsoft Excel/LibreOffice Calc (manuelle Verunreinigung Daten)
Diagramme	Lucidchart

4 Fragestellungen

Nachfolgend sind die vier Fragestellungen ausgeführt, die mit den gewonnenen Daten beantwortet werden sollen.

1. Welches Land war zwischen 1950-2019 am häufigsten Gastgeber eines Sportgrossanlasses (Sommer- und Winterolympiade, Fussballweltmeisterschaft)?
2. Welcher Kontinent verzeichnet den grössten BIP-Zuwachs in der Periode von 2000-2019?
3. In welcher Dekade wurden auf welchem Kontinent die meisten Sportgrossanlässe (Sommer- und Winterolympiade, Fussballweltmeisterschaft) durchgeführt?
4. Gibt es eine statistisch feststellbare, signifikante Korrelation zwischen der Durchführung von Sportgrossanlässen und der Veränderung des BIPs der Gastgebernation?

5 ETL-Prozess

Zu Beginn werden die Robots.txt-Files gecheckt und keine relevanten Scraping-Einschränkungen festgestellt. Nach dem Scrapen der Daten müssen einige CSV-Files (B1, C1) verunreinigt werden, da sie bereits eine hohe Qualität aufweisen. Die Quellen B2 und B3 enthielten Unreinheiten und wurden so weiterverarbeitet. A1 und C2 werden nach der manuellen Gewinnung via Webseite (CSV-Export & Copy/paste) im bereits reinen Zustand belassen.

[3] Manuelle Kopie von Webseite in Textverarbeitungstool

Im nächsten Schritt werden die Verunreinigungen mit PyCharm korrigiert bzw. bereinigt. Die sauberen Tabellen werden gemerged und die Fragestellungen in PyCharm ausgewertet und ausgerechnet. Zusätzlich sollen die sauberen Datasets in eine MariaDB-Datenbank eingelesen werden.

6 Arbeitsteilung

Wer	Datenquelle	Was
Régis Andréoli	A1, B1, C1, C2	Datenquelle scrapen (B1, C1), CSV exportieren (A1, C2) und Daten mit Python/Pandas bereinigen/anreichern
Micha Käser	B2, B3	Datenquelle scrapen (B2, B3) und Daten mit Python/Pandas bereinigen/anreichern
Team	Alle	Daten mergen, joinen und gruppieren. Fragestellungen beantworten. Danach mariaDB aufsetzen und bereinigte Datasets einlesen

7 Lösungsschritte

Im folgenden Abschnitt wird die Projektarbeit nach dem ETL-Schema, Extract, Transform und Load dokumentiert. Da der ausdokumentierte Quellcode mit dieser Arbeit abgegeben wird, wird in dieser Projektdokumentation darauf verzichtet, den kompletten Code erneut wiederzugeben. Stattdessen werden ausgewählte, spannenden Sachverhalte und Codeabschnitte aufgezeigt und erläutert.

7.1 Extract

In der Extract-Phase werden Daten aus Quellen im Internet gesammelt. Die Daten werden allesamt, bis auf A1 und C2, mittels eines Web Scrapers von einer Webpage extrahiert.

7.1.1 A1 – a1_rgdpna_stage.csv

Die A1-Daten enthalten das BIP pro Jahr und pro Land. Die Länder-Spezifikation ist jeweils im ISO-3166 alpha-3 festgehalten.

Die Daten werden von einer Webpage manuell via CSV-File heruntergeladen. Der ursprüngliche Versuch, dieses CSV via Python herunterzuladen, wurde eingestellt. Dies, da sich herausstellte, dass sich das CSV dynamisch bildet mittels einer Javascript-Funktion im HTML-Code. Allem Anschein nach triggert der Browserbutton eine Javascript-Funktion, die aus dem eigenen HTML Informationen herauszieht, um die Funktion selbst zu vervollständigen. Danach wird der komplette Javascript-Befehl ausgeführt und das CSV erstellt. Die Daten sowie die Javascript-Funktion sind beide im HTML-Code versteckt oder verteilt, was die Extraktion kompliziert gestaltet. Daher wird von der Python-Lösung abgesehen, zumal vier weitere Datenquellen effektiv mit Python gescraped werden.

7.1.2 B1 – b1_web_scrapper.py

Der B1 Web Scraper ladet und erstellt aus einer Webseite eine List mit den Fussballweltmeisterschaften und das dazugehörige Eventjahr.

```
import requests
from bs4 import BeautifulSoup as bs

#download html file and convert into soup object
raw_html = requests.get("https://www.fussball-wm-total.de/History/histehre.html")
soup_html = bs(raw_html.text, "html.parser")

#get most distinctive html structure to target input
extract = soup_html.find("table", width="430", border="1", cellpadding="10")
```

```
#loop through html lines, eliminate unwanted input values and store as csv
csv_file = open("b1_wm_src.csv", "w", encoding='utf-8')
for p in extract.select("tr"):
    y = p.select("td")
    try:
        a = y[0].text
        b = y[1].text
        csv_file.write(a + "," + b + "\n")
        print(a + "," + b + "\n")
    except:
        1
csv_file.close()
```

Das Scrapen der WM-Fussballspiele ist nicht komplett konventionell. Die Webpage www.fussball-wm-total.de hat eine multiple, vernestete Struktur aus HTML-Tabellentags. Es wurde generell wenig bis gar nicht mit Attributen gearbeitet und so ist die richtige HTML-Struktur zum Scrapen schwierig zu erkennen.

Nach dem Download werden die gesuchten Daten aus dem HTML extrahiert und abgespeichert. Dies erfolgt zeitgleich in der For-Loop.

7.1.3 B2 – b2_web-scrapper.py

Die Quellentabelle der olympischen Winterspiele ist bereits in gut strukturierter und einfach lesbarer Form vorhanden und erlaubt es, über Tags die benötigten Informationen auszulesen. Der rohe HTML-Code wird mit LXML geparsed und danach die gewünschte Tabelle mit den jeweiligen Zeilen ausgewertet.

```
# Als -olympics_table wird der Inhalt des Tags <table> mit der Klasse "dataList" geladen
# Danach wird -olympics_table nach allen Tags <tr> durchsucht
winter_olympics_table = soup.find("table", attrs={"class": "dataList"})
winter_olympics_table_data = winter_olympics_table.tbody.find_all("tr")
```

Auch die Überschriften wollen übernommen werden, weshalb nun nach den Tags «th» gesucht wird und die Werte im Array «Headers» gespeichert werden.

```
# Damit die Daten bzw. Kolonnen auch benennt werden, scrapen wir hier die Überschriften der
Tabelle
headers = []
for i in winter_olympics_table.find_all('th'):
    title = i.text.strip()
    headers.append(title)
```

Im erstellten Dataframe «olympics_table_data» werden alle «td»-Tags gesucht und als Rows gespeichert.

```
# In _olympics_table_data werden nun alle Tags <td> gesucht und in rows[] gespeichert.
# Diese Tags beinhalten pro Datensatz einen Wert (Jahr, Ort etc.)
rows = []
for row in winter_olympics_table_data:
    value = row.find_all('td')
    beautified_value = [ele.text.strip() for ele in value]
# Data Arrays werden entfernt, falls sie leer sind
    if len(beautified_value) == 0:
        continue
    rows.append(beautified_value)
```

Header sowie Rows werden schlussendlich in einem CSV-File ausgegeben.

```
# Schlussendlich wird unter dem eingangs definitierten Namen eine CSV-Datei exportiert und
kann im nächsten Schritt
# (Cleaning) weiterverarbeitet werden
with open(csv_output_name, 'w', newline="") as output:
    writer = csv.writer(output)
```

```
    writer.writerow(headers)
    writer.writerows(rows)
```

7.1.4 B3 – b3_web_scrapper

Wie die Tabelle der olympischen Winterspiele ist auch diejenige der Sommerspiele in gut strukturierter Form vorhanden und erlaubt es, mit Tags die benötigten Informationen auszulesen. Der geschriebene Code weicht nur in Details von demjenigen im vorgängigen Unterkapitel ab und wird hier deshalb nicht erneut wiedergegeben.

7.1.5 C1 – c1_web_scrapper.py

Der C1 Web Scraper downloaded einen HTML-Code und erstellt daraus eine Liste mit Ländernamen und den zugehörigen Kontinenten.

Ähnlich wie bei B1 ist die Webpage aus HTML-Sicht sehr einfach konzipiert. Die Webpage beinhaltet nahezu keine brauchbare Struktur und wäre an sich nicht scrapbar. Da sich aber der Webinhalt mehrheitlich auf das begrenzt, was benötigt wird, wird einfachheitshalber grob gescrapt. Die dabei entstehende Datenverunreinigung wird später im Cleaning behoben.

Wie bei B1 werden die Daten in der For-Loop verarbeitet und anschliessend im CSV abgespeichert.

```
import requests
from bs4 import BeautifulSoup as bs

#download html file and convert into soup object
raw_html = requests.get("https://www.bernhard-gaul.de/wissen/staatenerde.php#uebneu")
soup_html = bs(raw_html.text, "html.parser")

#load extracted text into csv file
csv_file = open("c1_country_src.csv", "w", encoding='utf-8')
for p in soup_html.select("tr"):
    y = p.select("td")
    a = y[0].text
    b = y[1].text
    csv_file.write(a + "," + b + "\n")
    print(a + ",")
csv_file.close()
```

7.1.6 C2 – c2_laendercode_src.csv

Die C2-Daten enthalten die Länder-Spezifikation im ISO-3166 alpha-3 Format sowie die vollständige Länderbezeichnung. Die C2-Daten werden manuell gewonnen. Ein Copy-Paste aus einer Web-Tabelle macht dies zu einer einfachen Aufgabe.

7.2 Transform

In der Transform Phase werden die Daten bereinigt und für die weitere Verarbeitung passend formatiert.

7.2.1 A1

Die A1-Daten werden bereits als CSV in höchster Datengüte abgespeichert und werden auch nicht manuell verunreinigt. Der Datensatz bedingt keine weiteren Massnahmen.

7.2.2 B1 – b1_cleaning.py

Die B1-Daten werden manuell verunreinigt und müssen auf folgendes korrigiert werden:

1. Umlaute umschreiben
2. Zeilenfehler wegen Semikolon
3. Gross- & Kleinschreibung
4. Fehlerhaftes Datum

1. replace() findet alle Umlaute und wandelt diese um, regex=True ist dabei notwendig für die Suche nach dem Umlauten.

```
df = df.replace('ä', 'ae', regex=True)
df = df.replace('ö', 'oe', regex=True)
df = df.replace('ü', 'ue', regex=True)
df = df.replace('Ä', 'Ae', regex=True)
df = df.replace('Ö', 'Oe', regex=True)
df = df.replace('Ü', 'Ue', regex=True)
```

2. Folgend ein Beispiel wie mittels Dataframe Index eine spezifische Zeile angezeigt und anschliessend korrigiert wird.

```
print(df['Land'][14])
df['Land'][14] = "USA"
df['Jahr'][14] = "1994"
df.count()
```

3. Diese For-Loop vermag eine DF-Spalte durchzuschlaufen und erkennt dabei Kleinschreibung. Beginnt ein Name mit einem Kleinbuchstabe, wird dieser mit dem selben in Grossschreibung ersetzt.

```
count = 0
for p in df["Land"]:
    if df["Land"][count][0].islower():
        print(p)
        df["Land"][count] = (df["Land"][count][0].upper() + df["Land"][count][1:])
    count += 1
```

4. Diese For-Loop überprüft ob die Daten in einer Spalten einen bestimmten Format einhalten. Hier dürfen die Felder nicht mehr als vier Charaktere haben. Falls dem nicht so ist, wird von links beginnend alles weggeschnitten, was vier Charaktere übersteigt.

```
count = 0
for p in df["Jahr"]:
    if len(df["Jahr"][count]) > 4:
        print(p)
        df["Jahr"][count] = df["Jahr"][count][-4:]
    count += 1
```

Anschliessend werden die Veränderungen im ‘b1_wm_stage.csv’ abgespeichert.

7.2.3 B2 – b2_cleaning.py

Die Daten des Source-Files werden nicht manuell verunreinigt, da sie bereits einige Fehler und Unschönheiten aufweisen. Nach dem initialen Load von b2_wolympics_src.csv werden gleich einige nicht benötigte Kolonnen und Zeilen gedroppt:

```
# Alle Zeilen, die komplett leer sind (NaN, also keine Values haben), werden jetzt gedroppt
df = df.dropna(how='all')

# Alle Kolonnen, die nicht gebraucht werden, werden ebenso gedroppt
to_drop = ['Teilnehmer',
           'Wettbewerbe',
           'Beste Nationen',
           'BesteÂ Sportler']

df.drop(to_drop, inplace=True, axis=1)
```

Nun werden die Jahrzahlen und danach die Länderkürzel extrahiert:

```
# Jahrzahlen in neue Kolonne 'Jahr_stage' extrahieren, alle Whitespaces löschen und Datatype
INT setzen
df['Jahr_stage'] = df['Jahr'].str[:4]
df['Jahr_stage'] = df['Jahr_stage'].str.replace(' ', '')
df['Jahr_stage'] = df['Jahr_stage'].astype('int64')

# Länderkürzel in neue Kolonne 'Land_code' extrahieren
df['Land_code'] = df['Land'].str.replace(r'[^(]*\(|\)[^)]*', '')
```

Da nicht sämtliche Ländercodes dem ISO-3-Standard entsprechen, müssen einige anhand eines Dictionarys ersetzt werden:

```
# Länderkürzel in neuer Kolonne 'Land_code_stage' durch ISO3-Norm-Values ersetzen
df['Land_code_stage'] = df['Land_code'].replace(['F', 'CH', 'D', 'JAP', 'IT', 'JP', 'JUG',
'SÃ¼dkorea'],
                                                ['FRA', 'CHE', 'DEU', 'JPN', 'ITA', 'JPN',
'BIH', 'KOR'])
```

Nicht benötigte Kolonnen werden jetzt gedroppt, die übriggebliebenen umbenannt und das gesäuberte Dataframe als CSV ausgegeben.

```
# Alle Kolonnen, die nach dem Cleaning nicht mehr gebraucht werden, werden vor Output gedroppt
to_drop = ['Jahr',
           'Land',
           'Land_code', ]

df.drop(to_drop, inplace=True, axis=1)

# Umbenennung der sauberen, korrekten Spalten zur Ausgabe
df['Land_code'] = df['Land_code_stage']
df['Jahr'] = df['Jahr_stage']
df['Anlass'] = df['Anlass_stage']

# Schlussendlich werden die Daten in ein neues CSV-File geschrieben, der Index dabei entfernt,
der Header belassen
df[['Jahr', 'Land_code', 'Anlass']].to_csv(csv_output_name, index=False, header=True)
```

7.2.4 B3 – b3_cleaning.py

Auch diese Daten werden nicht manuell verunreinigt – sie sind denjenigen aus dem vorgängigen Kapitel sehr ähnlich, und damit auch der Code. Als einzige Abweichung muss eine Zeile dupliziert werden, da der Sportanlass auf zwei Kontinenten stattgefunden hatte:

```
# Zeile 14 wird dupliziert, da die Olympiade in diesem Jahr in zwei Ländern stattgefunden
hatte
# Danach Index +1 und neu sortieren
df.loc[15] = ['1940 # XII', 'FIN']
df.index = df.index + 1
df = df.sort_index()
```

Der restliche Code ist ähnlich und wird hier deshalb nicht erneut ausgeführt.

7.2.5 C1 – c1_cleaning.py

Die C1-Daten werden manuell verunreinigt und müssen auf folgendes korrigiert werden:

1. Spalten- und Zeilenfehler
2. Wörterkorrektur
3. Gross- & Kleinschreibung

Natürliche Verunreinigungen kommen ebenfalls vor:

4. Titel enthält ungültige Zeichen
5. Im Datensatz folgen ungewollte Zeilen

1. Da der Titel fehlerhaft ist und das Einlesen des CSVs nicht funktioniert, wird dieser ignoriert (header=none) und die Zeile ausgelassen (skiprows=1). Der Titel wird gleich manuell angegeben (names=[…)

```
df = pd.read_csv('c1_country_src_dirty.csv', header=None, skiprows=1, names=['Kontinent',
'Land'], encoding='utf-8')
```

2. Mit .isnull() werden jeweils Null-Zeilen ermittelt. Dabei ist der DF Index sichtbar. Der Index kann anschliessend genutzt werden, um die Zeilen direkt aufzurufen und zu verändern.

```
df[df['Land'].isnull()]

df.loc[23]
df.loc[23] = ["Afrika", "Benin"]
```

3. Mit der Übersichtsausgabe aus groupby() wird ersichtlich, dass Europa ein paar Mal falsch geschrieben ist. Die Abfrage *== 'Europe' wiedergibt alle fehlerhaften Zeilen, und es kann wieder mittels Index korrigiert werden.

```
print(df.groupby('Kontinent').count())
df[df['Kontinent'] == 'Europe']
df['Kontinent'][4] = 'Europa'
df['Kontinent'][81] = 'Europa'
```

4. Mit df[-15] werden die letzten Einträge angeschaut und es wird visuell ermittelt, wo im Index der überflüssige Abschnitt anfängt.

```
df[-15:]

df.drop(df.loc[242:252].index, inplace=True)
```

7.2.6 C2 – c2_cleaning.py

Die C2-Daten werden nicht manuell verunreinigt, enthalten allerdings Elemente, die verändert werden müssen. Die Umlaute werden wie zuvor umgeformt und zwei Spalten werden gedroppt.

7.3 Load

Die bereinigten Datasets werden nun in eine MariaDB-Datenbank eingelesen. Diese wurde unter Ubuntu 20.04.2 LTS erstellt.

7.3.1 Erstellen Datenbank und Tabellen

Nach erfolgter Installation und Konfiguration via Terminal wird die Datenbank aufgesetzt:

```
mysql -u root -p
mysql> CREATE DATABASE CIP_Project_Group50;
mysql> USE CIP_Project_Group50;
```

Innerhalb dieser Datenbank werden nun den Daten entsprechende Tabellen erstellt:

```
CREATE TABLE s_olympics (
        Jahr INT NOT NULL,
        Land_code VARCHAR(255) NOT NULL,
        Anlass VARCHAR(255) NOT NULL);

CREATE TABLE w_olympics (
        Jahr INT NOT NULL,
        Land_code VARCHAR(255) NOT NULL,
        Anlass VARCHAR(255) NOT NULL);

CREATE TABLE rgdpna (
        VariableCode VARCHAR(255) NOT NULL,
        RegionCode VARCHAR(255) NOT NULL,
        YearCode INT NOT NULL,
        AggValue FLOAT NOT NULL);

CREATE TABLE wm (
        Land VARCHAR(255) NOT NULL,
        Jahr INT NOT NULL);
```

```
CREATE TABLE country (
        Kontinent VARCHAR(255) NOT NULL,
        Land VARCHAR(255) NOT NULL);

CREATE TABLE laendercode (
        Land VARCHAR(255) NOT NULL,
        ISO3 VARCHAR(255) NOT NULL);
```

7.3.2 Einlesen CSVs

Die Tabellen werden mit Informationen aus den CSV-Files befüllt:

```
LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/b3_solympics_stage.csv'
INTO TABLE s_olympics
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Jahr, Land_code, Anlass)

LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/b2_wolympics_stage.csv'
INTO TABLE s_olympics
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Jahr, Land_code, Anlass)

LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/b1_wm_stage.csv'
INTO TABLE wm
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Land, Jahr)

LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/a1_rgdpna_stage.csv'
INTO TABLE rgdpna
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(VariableCode, RegionCode, YearCode, AggValue)

LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/c1_country_stage.csv'
INTO TABLE country
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Kontinent, Land)

LOAD DATA LOCAL INFILE '/home/micha/PycharmProjects/CIP_Project/c2_laender-
code_stage_v3.csv'
INTO TABLE laendercode
FIELDS TERMINATED BY ','
LINES TERMINATED BY '\n'
IGNORE 1 ROWS
(Land, ISO3)
```

7.3.3 Inspektion Datenbank und Tabellen

Der folgende Befehl zeigt eine Übersicht aller auf dem System gehosteter Datenbanken:

```
MariaDB [(none)]> SHOW DATABASES;
```

```
+--------------------+
| Database           |
+--------------------+
| CIP_Project_Group50 |
| information_schema  |
| mysql               |
| performance_schema  |
+--------------------+
4 rows in set (0.008 sec)
```

Die Datenbank “CIP_Project_Group50” wird angewählt…

```
MariaDB [(none)]> USE CIP_Project_Group50;
```

```
Database changed
MariaDB [CIP_Project_Group50]>
```

…und die darin enthaltenen Tabellen angezeigt:

```
MariaDB [CIP_Project_Group50]> SHOW TABLES;
```

```
+------------------------------+
| Tables_in_CIP_Project_Group50 |
+------------------------------+
| country                      |
| laendercode                  |
| rgdpna                       |
| s_olympics                   |
| w_olympics                   |
| wm                           |
+------------------------------+
6 rows in set (0.000 sec)
```

Mittels eines einfachen, auf 10 Records beschränkten *Selects* ist sichergestellt, dass die Daten erfolgreich eingelesen wurden:

```
MariaDB [CIP_Project_Group50]> SELECT * FROM rgdpna LIMIT 10;
```

```
+--------------+------------+-----------+-----------+
| VariableCode | RegionCode | YearCode  | AggValue  |
+--------------+------------+-----------+-----------+
| rgdpna       | ABW        |      1970 |   262.309 |
| rgdpna       | ABW        |      1971 |   286.148 |
| rgdpna       | ABW        |      1972 |   312.155 |
| rgdpna       | ABW        |      1973 |   340.525 |
| rgdpna       | ABW        |      1974 |   371.474 |
| rgdpna       | ABW        |      1975 |   405.235 |
| rgdpna       | ABW        |      1976 |   442.065 |
| rgdpna       | ABW        |      1977 |   482.242 |
| rgdpna       | ABW        |      1978 |    526.07 |
| rgdpna       | ABW        |      1979 |   573.882 |
+--------------+------------+-----------+-----------+
10 rows in set (0.000 sec)
```

8 Fragestellungen

8.1 Frage 1

Frage Welches Land war zwischen 1950-2019 am häufigsten Gastgeber eines Sportgrossanlasses (Sommer- und Winterolympiade, Fussballweltmeisterschaft)?

Lösung Die vereinigten Staaten von Amerika mit total 9 Sportgrossanlässen, gefolgt von Frankreich und Italien mit 8 respektive 7.

Für diese Fragestellung werden die folgenden vier CSVs benötigt:

```
# Datenquellen werden eingelesen
df_wm = pd.read_csv('b1_wm_stage.csv', header=0, encoding='utf-8')
df_wo = pd.read_csv('b2_wolympics_stage.csv', header=0, encoding='utf-8')
df_so = pd.read_csv('b3_solympics_stage.csv', header=0, encoding='utf-8')
df_lc = pd.read_csv('c2_laendercode_stage.csv', header=0, encoding='utf-8')
```

Damit nicht nur die Ländercodes, sondern auch die Ländernamen ausgewertet werden können, erfolgt für die CSVs der olympischen Spiele ein *inner join* auf die Tabelle Ländercodes. Die Codes werden nach Import der Ländernamen direkt wieder verworfen.

```
# Aufbereitung des Datasets Olympische Sommerspiele (Join mit CSV der Ländercodes, danach neu anordnen der Spalten)
df_so_new = pd.merge(df_so, df_lc_new, on='Land_code', how='inner')
df_so_new.pop('Land_code')
df_so_new = df_so_new[['Jahr', 'Land', 'Anlass']]
```

Danach werden alle Sportanlässe-Datasets zusammengeführt, gefiltert und sortiert. Anschliessend kann die Lösung in ein Excel-File exportiert werden.

```
# Zusammenführen der drei Datasets Sportanlässe und sortieren nach Jahr
df_concatenated = pd.concat([df_wo_new, df_so_new, df_wm_new])
df_concatenated = df_concatenated.sort_values(by=['Jahr'],ignore_index=True)

# Bevor das XSLX generiert wird, werden Filter angewendet und die Werte gezählt & nach der Anzahl sortiert
df_final = df_concatenated.loc[(df_concatenated['Jahr'] > 1950) & (df_concatenated['Jahr'] < 2019)]
df_final = df_concatenated['Land'].value_counts(sort=True)
```

8.2 Frage 2

Frage Welcher Kontinent verzeichnet den grössten BIP-Zuwachs in der Periode von 2000-2019?

Lösung Der Kontinent Asien schlägt mit dem fast vierfachen BIP-Zuwachs den zweitplatzierten Kontinent Nordamerika.

Um die Fragestellung 2 zu beantworten, werden drei Datensätze miteinander verbunden.

Die zwei folgenden Datensätze enthalten die relevanten Informationen, um die Fragestellung zu lösen…

a1_rgdpna_stage.csv

```
VariableCode,RegionCode,YearCode,AggValue
rgdpna,ABW,1970,262.30859375000000000
rgdpna,ABW,1971,286.14843750000000000
rgdpna,ABW,1972,312.15496826171900000
rgdpna,ABW,1973,340.52511596679700000
rgdpna,ABW,1974,371.47366333007800000
rgdpna,ABW,1975,405.23498535156300000
rgdpna,ABW,1976,442.06466674804700000
```

c1_country_stage.csv

```
Kontinent,Land
Afrika,Aegypten
Afrika,Aequatorial-Guinea
Afrika,Aethiopien
Asien,Afghanistan
Europa,Albanien
Afrika,Algerien
```

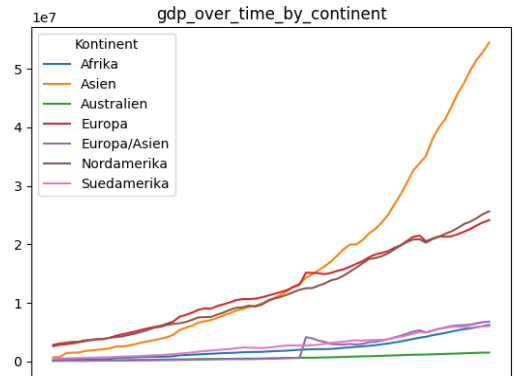
…während der dritte Datensatz dazu benötigt wird, um die ersten zwei Datensätze (A1 & C1) zusammenführen zu können.

c2_laendercode_stage.csv

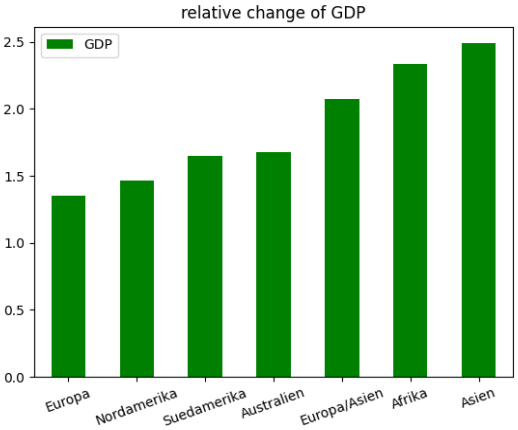
```
Land,ISO-3
Afghanistan,AFG
Aegypten,EGY
Albanien,ALB
Algerien,DZA
Amerikanisch-Samoa,ASM
Andorra,AND
```

Nachdem die Daten gemerged und diversen Filter- und Auswahlverfahren durchlaufen werden, wie z.B. groupby(), werden diese in Plots visualisiert.

Folgend sind die drei erstellten Plots. Der Plot ‘fragestellung_2_absolute_change_of_gdp’ beantwortet die eigentliche Fragestellung. Die beiden anderen Plots liefern Zusatzinhalt, für das Interesse und die Neugier.

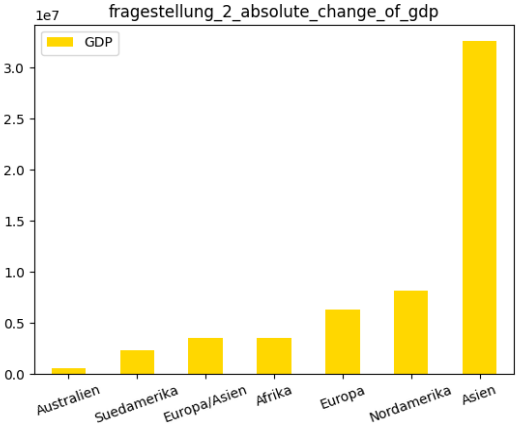


Die Grafik ‘gdp_over_time_by_continent’ zeigt das BIP pro Jahr pro Kontinent. Zu erkennen ist der sprunghafte Aufschung von Asien Ende 90er oder die Einführung der Kategorie Europa/Asien Ende der 80er.



Die Grafik ‘relative change of GDP’ zeigt die relative Veränderung des BIP zwischen dem Jahr 2000 und 2019. Das BIP im Jahre 2000 dient als Basis und würde in dieser Grafik den Wert 1 erhalten. Abgebildet wird die Veränderung im Jahre 2019, als Rate zum Jahre 2000.

Zu vermerken ist, dass aus terminologischer Sicht bei dieser Rechnung eher die ‘Rate’ [x2/x1] ausgerechnet wurde als die ‘Relative Veränderung’ [(x2-x1)/x1]



Diese Grafik ‘fragestellunf_2_absolute_change_of_gdp’ zeigt die absolute Differenz zwischen dem Jahr 2019 und 2000. Diese Grafik löst die gestellte Frage – die Antwort ist, dass *der Kontinent Asien den grössten BIP-Zuwachs in der Periode von 2000-2019 verzeichnet.*

Alle drei Grafiken werden als png abgespeichert und zusätzlich in das Excel File ‘Result_Question_02’ zusammengeführt.

8.3 Frage 3

Frage In welcher Dekade wurden auf welchem Kontinent die meisten Sportgrossanlässe (Sommer- und Winterolympiade, Fussballweltmeisterschaft) durchgeführt?

Lösung Es gibt gleich drei «Gewinner»: In den Dekaden 1920, 1950 und 1990 wurden in Europa jeweils fünf Sportgrossanlässe durchgeführt. Europa belegt insgesamt sogar die ersten 10 Plätze dieses Rankings.

Für diese Fragestellung sind fünf der sechs Datasets notwendig und werden eingelesen.

```
# Datenquellen werden gelesen
df_wm = pd.read_csv('b1_wm_stage.csv', header=0, encoding='utf-8')
df_wo = pd.read_csv('b2_wolympics_stage.csv', header=0, encoding='utf-8')
df_so = pd.read_csv('b3_solympics_stage.csv', header=0, encoding='utf-8')
df_lc = pd.read_csv('c2_laendercode_stage.csv', header=0, encoding='utf-8')
df_ln = pd.read_csv('c1_country_stage.csv', header=0, encoding='utf-8')
```

Die Sportanlässe werden, ähnlich wie für die Fragestellung 1, für die weitere Verarbeitung aufeinander abgestimmt. Danach erfolgen *inner joins* auf die Ländercodes, damit die olympischen Anlässe ebenso wie die Fussball-WM mit einem Ländernamen aufwarten können.

```
# Aufbereitung des Datasets Olympische Sommerspiele (Join mit Ländercode und Neuanordnung der
Spalten)
df_so_new = pd.merge(df_so, df_lc_new, on='Land_code', how='inner')
df_so_new.pop('Land_code')
df_so_new = df_so_new[['Jahr', 'Land', 'Anlass']]

# Aufbereitung des Datasets Olympische Winterspiele (Join mit Ländercode und Neuanordnung der
Spalten)
df_wo_new = pd.merge(df_wo, df_lc_new, on='Land_code', how='inner')
df_wo_new.pop('Land_code')
df_wo_new = df_wo_new[['Jahr', 'Land', 'Anlass']]
```

Nach einem *merge* der drei Sportdatasets wir eine weitere Kolonne «Dekade» geschaffen. Diese erhält ihre Werte durch eine Manipulation der Spalte «Jahr»:

```
# Neue Kolonne 'Dekade' wird erstellt, dafür die bestehenden Jahrzahlen kopiert und angepasst
df_events['Dekade'] = df_events['Jahr'].astype('str').str[:3] + '0'
df_events['Dekade'] = df_events['Dekade'].astype('int64')
```

Nach der Bildung des finalen Dataframes «df_final» erfolgt ein *groupby* nach Dekade und Kontinent, diese Zeilen werden mit durch *count* generierte Werte angereichert. Diese Tabelle wird dann als Excel-File und Lösung ausgegeben.

```
# Danach werden die Values pro Dekade und Kontinent gezählt, die Kolonne umbenennt und nach
Häufigkeit (Count) sortiert
df_final = df_final.groupby(["Dekade", "Kontinent"], as_index=False).count()
df_final = df_final.rename(columns={'Anlass': 'Anzahl Anlässe'})
df_final = df_final.sort_values(by=['Anzahl Anlässe'], ignore_index=True, ascending=False)

# Schlussendlich wird die Auswertung in ein Excelfile geschrieben
df_final.to_excel(xlsx_output_name, index=False)
```

8.4 Frage 4

Frage Gibt es eine statistisch feststellbare, signifikante Korrelation zwischen der Durchführung von Sportgrossanlässen und der Veränderung des BIPs der Gastgebernation?

Lösung Die ausgewählten Methoden liefern kein eindeutiges Resultat, resp. ist eine Veränderung nicht nach visueller Inspektion feststellbar (s. Grafiken).

Um die Frage 4 zu beantworten, werden die Datensets 'a1_rgdpna_stage.csv' und 'c2_laender-code_stage.csv' benötigt und zusammen gemerget. Der dritte Datenset 'fragestellung_4_top4_laender.csv' ist eine Liste mit Sportevents und Jahr, der vier Nationen, die am häufigsten Gastgeber solcher Sportanlässe waren.

Datensets

a1_rgdpna_stage.csv

```
VariableCode,RegionCode,YearCode,AggValue
rgdpna,ABW,1970,262.30859375000000000
rgdpna,ABW,1971,286.14843750000000000
rgdpna,ABW,1972,312.15496826171900000
```

c1_country_stage.csv

```
Kontinent,Land
Afrika,Aegypten
Afrika,Aequatorial-Guinea
Afrika,Aethiopien
```

fragestellung_4_top4_laender.csv

```
Jahr,Land_code,Anlass
1900,FRA,Olympische Sommerspiele
1904,USA,Olympische Sommerspiele
1916,DEU,Olympische Sommerspiele
1924,FRA,Olympische Winterspiele
1924,FRA,Olympische Sommerspiele
```

Die Datensets werden eingelesen, gemerged und in ein brauchbares Format gebracht. Die nachfolgende Sequenz, hier am Beispiel der USA, wird für die Länder Frankreich, Italien und Deutschland wiederholt.

1. Erstellen zweier, jeweils länderspezifischen DF: df_usa mit der Time Series des BIP und df_date_usa, mit einer Liste der Daten von Sportanlässen.

```
df_usa = df_j[df_j['Land'] == 'USA']
df_date_usa = df_4c['Jahr'][(df_4c.Land_code == "USA") & ((df_4c.Jahr > 1949) & (df_4c.Jahr < 2020))]
```

2. Danach wird mit der Time Series des BIP ein erstes Plot erstellt und als PNG abgespeichert.

```
df_usa.plot(x='YearCode', y='AggValue', marker='', color='goldenrod', linewidth=1.0, figsize=(8,5))
plt.xlabel('year')
plt.ylabel('USD')
plt.title('Time Series of US GDP')
plt.savefig('fragestellung_4_usa_bip_time_series')
```

3. Die Time Series wird dann enttrendet

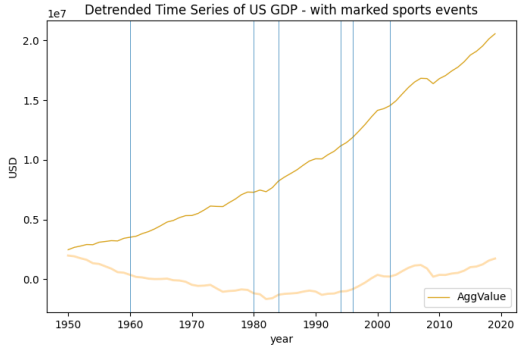
```
usa_detrended = signal.detrend(df_usa['AggValue'])
```

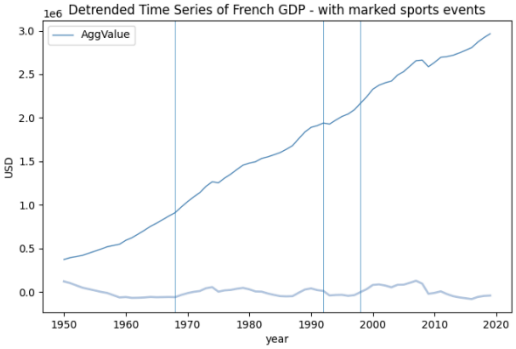
4. Die enttrendete Time Series wird zum Schluss ebenfalls in einen Plot visualisiert. Dabei werden noch die Jahre, an denen einen Sportevent stattfand, als Linie eingefügt. Da die Funktion axvline() nur eine einzelne Linie im Plot einzufügen vermag, muss eine For-Loop dieses Prozedere immer wiederholen.

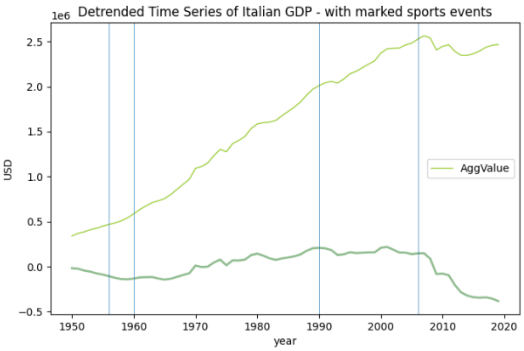
```
plt.figure(1, figsize=(8, 5))
plt.plot(df_usa['YearCode'], usa_detrended, label="GDP_detrended", color='navajowhite', linewidth=2)
x = df_date_usa.tolist()
for a in x: plt.axvline(a, linewidth=0.5)
plt.title('Detrended Time Series of US GDP - with marked sports events')
plt.savefig('fragestellung_4_usa_time_series_analyse')
plt.show()
```

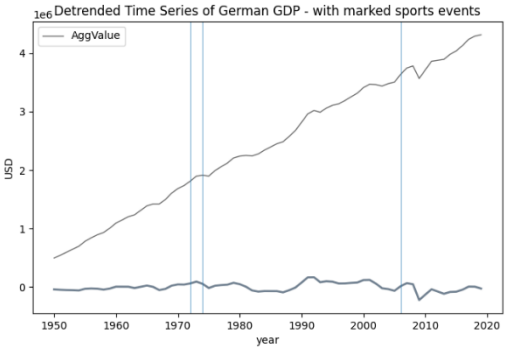
Es liegen nun zwei Grafiken pro Land vor, eine Grafik mit dem BIP über die Zeitachse und eine mit der enttrendeten Time Series. Folgend jeweils die zweite Grafik, mit der enttrendeten Time Series und die

Eventkennzeichnung als vertikale Linie. Die Grafik enthaltet ebenfalls die normale BIP Time Series. Diese unterscheidet sich durch ihren trendigen anstieg über die Zeit, während die enttrendete flacher liegt.









Nach einer visuellen Inspektion lassen sich keine Rückschlüsse auf den Einfluss von Sportgrossanlässen auf das BIP ziehen. Es könnte sein, dass das BIP eine Summe ist die die Dimension, wie eine Olympiade oder Fussball-WM, schlichthin bei weitem übertrifft.

9 Reflexion

Bei der Aufbereitung von gescrapten Daten ist es eminent wichtig, dass die Originaldaten beibehalten werden. Das bedeutet also, dass vor einer Manipulation am Dataframe eine exakte Kopie der betroffenen Spalte/Zeile erstellt wird. Deshalb muss ein Naming-Scheme entwickelt und innerhalb des Teams angewendet und einhalten werden. Dabei haben sich die Tipps der Dozenten als sehr wertvoll erwiesen.

BeautifulSoup und Pandas sind mächtige Werkzeuge und erlauben es mit wenig Aufwand elegant Webseiten zu scrapen. Wichtig dabei ist, dass man sich zuerst eine gute Übersicht über die Webseite verschafft - schliesslich muss man wissen, mit was man da arbeitet. Dafür ist ein gewisses HTML- und CSS-Verständnis notwendig, welches innerhalb dieser Projektarbeit weiter vertieft wurde. Herangehensweisen, die zum selben Ziel führen.
Es ist sehr wichtig, dass gleich zu Beginn des Scraping definiert ist, wie die exportierten und aufbereiteten Daten aussehen sollen. So lässt sich vermeiden, dass in einem späteren Prozessschritt «cleane» Datasets erneut überarbeitet und angereicht werden müssen. GitHub mit der zentralisierten Repository und der Version Control hat sehr dabei geholfen, jederzeit "Herr" der Daten zu sein und den Überblick zu wahren.

Aus Sicht der Datenanalyse hat das Projekt gezeigt, dass als bald auf mehrere Quellen zugegriffen wird, die Daten unbedingt auf ihre Konsistenz zu überprüfen sind. Aber auch dann ist immer mit einer „bösen“ Überraschung zu rechnen, die das Projekt in die Länge ziehen kann.

10 Einwilligung

Diese Arbeit beinhaltet keine vertraulichen Daten und darf frei geteilt werden.