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 3](#_Toc72097788)

[2 Datenquellen 3](#_Toc72097789)

[3 Fragestellungen 4](#_Toc72097790)

[4 Tools 4](#_Toc72097791)

[5 ETL-Prozess 4](#_Toc72097792)

[6 Arbeitsteilung 4](#_Toc72097793)

[7 Lösungsschritte 5](#_Toc72097794)

[7.1 Extract 5](#_Toc72097795)

[7.1.1 A1 – a1\_rgdpna\_stage 5](#_Toc72097796)

[7.1.2 B1 – b1\_web\_scrapper.py 5](#_Toc72097797)

[7.1.3 B2 6](#_Toc72097798)

[7.1.4 B3 6](#_Toc72097799)

[7.1.5 C1 – c1\_web\_scrapper.py 6](#_Toc72097800)

[7.1.6 C2 6](#_Toc72097801)

[7.2 Transform 6](#_Toc72097802)

[7.2.1 A1 6](#_Toc72097803)

[7.2.2 B1 – b1\_cleaning.py 7](#_Toc72097804)

[7.2.3 B2 7](#_Toc72097805)

[7.2.4 B3 7](#_Toc72097806)

[7.2.5 C1 7](#_Toc72097807)

[7.2.6 C2 7](#_Toc72097808)

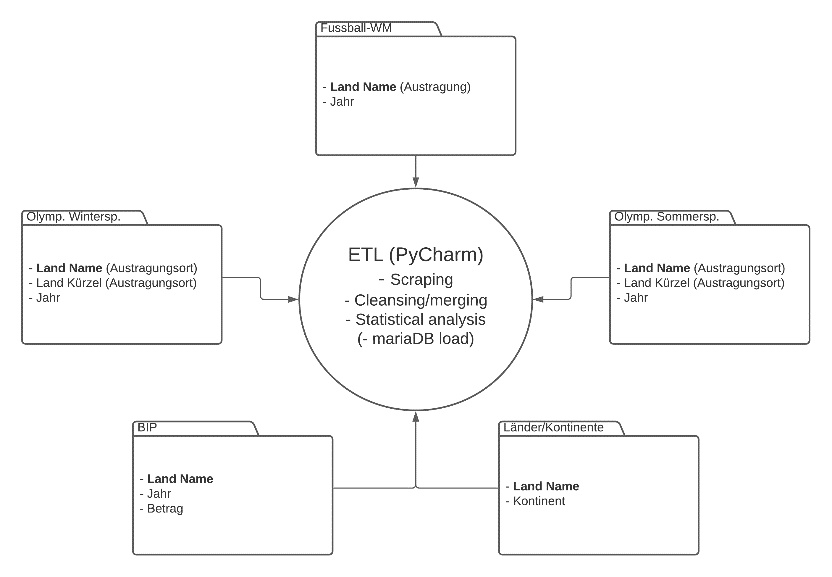
[7.3 Load 7](#_Toc72097809)

[8 Fragestellungen 8](#_Toc72097810)

[9 Reflexion 8](#_Toc72097811)

# 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 (Olympiaden/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 Daten allerdings erst ab der Nachkriegszeit (1950) bis 2019 ausgewertet.

# Datenquellen

|  |  |  |  |
| --- | --- | --- | --- |
| **ID** | **Daten** | **Attribute** | **Link** |
| A1[[1]](#footnote-1) | BIP nach Land [[2]](#footnote-2) 1950-2019 | * Land Name * Jahr * Betrag | <https://febpwt.webhosting.rug.nl/Dmn/AggregateXs/PivotShow> |
| B1 | Fussball-Weltmeisterschaften 1930-2018 | * Land Name Austragungsort * Jahr | <https://www.fussball-wm-total.de/History/histehre.html> |
| B2 | Olympische Winterspiele 1924-2018 | * Land Name  Austragungsort * Land Kürzel * Jahr | <https://www.taschenhirn.de/sport/olympische-winterspiele/> |
| B3 | Olympische Sommerspiele 1896-2016 | * Land Name  Austragungsort * Land Kürzel * Jahr | <https://www.taschenhirn.de/sport/olympische-sommerspiele/> |
| C1 | Länder und Kontinente | * Land Name * Kontinent | [https://www.bernhard-gaul.de/wissen/staatenerde.php#uebneu](https://www.bernhard-gaul.de/wissen/staatenerde.php%23uebneu) |
| C2 | Ländercodes | * Land Name * Land Kürzel | https://www.laenderdaten.de/kuerzel/iso\_3166-1.aspx |

Die Tabellen der Sportgrossanlässe werden in einem Dataset zusammengefasst und nach Jahr sortiert. Die Korrelationsberechnung zum BIP werden auf Basis Jahr und Land durchgeführt.

# Fragestellungen

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?

# Tools

|  |  |
| --- | --- |
| **Zweck** | **Tools** |
| Repository Code | GitHub |
| Repository Daten | GitHub/Switch Drive |
| Kommunikation | Slack, Zoom |
| Entwicklungsumgebung | Ubuntu (VM oder nativ), PyCharm, mariaDB, Microsoft Excel/LibreOffice Calc (manuelle Verunreinigung Daten) |
| Diagramme | Lucidchart |

# ETL-Prozess

Nach dem Scrapen der Daten müssen diese verunreinigt werden, da sie bereits eine sehr hohe Qualität aufweisen. Dafür werden in Microsoft Excel oder LibreOffice Calc BIP-Beträge in verschiedene Währungen umgerechnet, Strings mit Zahlen angereichert und unterschiedliche Sprachen verwendet.

Im nächsten Schritt werden diese 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.

# Arbeitsteilung

|  |  |  |
| --- | --- | --- |
| **Wer** | **Datenquelle** | **Was** |
| Régis Andréoli | A1, B1, C1 | Datenquelle scrapen (B1, C1), CSV exportieren (A1) 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 |

# Lösungsschritte

Im folgenden Abschnitt wird die Projektarbeit nach dem ETL-Schema, Extract, Transform und Load dokumentiert. Die Lösungswege werden erläutert und

## Extract

In der Extract Phase werden Daten aus Quellen im Internet gesammelt. Die Daten werden allesamt, bis auf A1, mittels eines Webscappers von den jeweiligen Webpage HTML’s extrahiert.

### A1 – a1\_rgdpna\_stage.csv

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

Die Daten werden von einer Webpage manuel via eines CSV heruntergeladen. Der ursprüngliche Versuch dieses CSV via Python herunterzuladen wurde eingestellt, als sich herausstellte, das das CSV dynamisch gebildet wird via einer Javascript Funktion im HTML Code. Allen Anschein nach triggert der Browser Button-Klick eine Javascript-Funktion, die aus dem eigenen HTML Informationen herauszieht, um die Funktion selbst zu vervollständigen. Danach wird der komplette Javascript Befehl ausgeführt. Die Daten sowie die Javascript Funktion sind beide in HTML Codes versteckt oder verteilt und macht die Extraction kompliziert. Daher wird von der Automation des Downloads via Python abgesehen, zumal diese Datenquelle nicht der Web-Scrapping Auflage unterliegt.

### B1 – b1\_web\_scrapper.py

Der B1 Web Scrapper ladet und erstellt aus einer Webseite eine List mit den Fussbalweltmeisterschaften und das jeweilige, dazughörige Jahr.

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 Scrappen der WM-Fussballspiele war nicht komplet konventionell. Die webpage www.fussball-wm-total.de hat eine multiple, vernestete Struktur aus Table Tags. Es wurde generell wenig bis gar nicht mit Attributen gearbeitet und so ist das Erkennen einer geeigneten HTML Struktur zum Scrappen mehr ‘’Try and Error’’ als methodisch.

Nach dem Webscrapping werden die Daten bereinigt und abgespeichert. Dies erfolgt zeitgleich in der Loop.

### B2

### B3

### C1 – c1\_web\_scrapper.py

Der C1 Web Scrapper ladet ein HTML runter und erstellt daraus eine Liste mit Ländername und das dazugehörige Kontinent.

Ähnlich wie bei B1, ist die Wepage aus HTML Sicht sehr einfach formatiert. Die Webpage beinhaltet nahezu keine brauchbare Struktur, und wäre an sich nicht scrappbar. Da sich aber der Webinhalt auf fast genau das begrenzt, was benötigt wird, wird schlichthin grob gescrappt. Die Datenverunreinigung hält sich in grenzen und ist überschaiber.

Wie bei B1 werden die Daten in der Loop verarbeitet, in der die Daten auch im CSV gespeichert werden.

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()

### C2 – c2\_laendercode\_src.csv

Die C2 Daten enthalten die Länder-Spezifikation im ISO-3166 alpha-3 festgehalten sowie die volständige Länderbezeichnung

Die C2 Daten werden ebenfalls manuel gewonnen. Ein Copy-Paste aus einer Web-Tabelle macht dies zu einer schnellen Aufgabe.

## Transform

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

### A1

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

### B1 – b1\_cleaning.py

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

1. Umlauten umschreiben
2. Zeilenfehler wegen Semicolon
3. Gross- & Kleinschreibung
4. Fehlerhafte Datums

1.  
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.print(df[**'Land'**][14])  
df[**'Land'**][14] = **"USA"**df[**'Jahr'**][14] = **"1994"**df.count()  
  
*#second - search for duplicates*df.duplicated() *#no duplicates!*3.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.count = 0  
for p in df[**"Jahr"**]:  
 if len(df[**"Jahr"**][count]) > 4:  
 print(p)  
 df[**"Jahr"**][count] = df[**"Jahr"**][count][-4:]  
 count += 1

### B2

### B3

### C1

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

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

Natürliche Verunreinigungen:

1. Titel enthält ungültige Zeichen
2. Im Datensatz folgen ungewollte Zeilen
3. Mit .isnull() werden jeweils Nan Zeilen Ermittelt. Dabei ist der DF Index Sichtbar. Der Index kann genutz werden um, die Zeilen direkt zu verändern

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

df.loc[23]  
df.loc[23] = [**"Afrika"**, **"Benin"**]  
  
df.loc[32]  
df.loc[32] = [**"Afrika"**, **"Burkina Faso"**]  
  
df.loc[67]  
df.loc[67] = [**"Australien"**, **"Guam"**]  
  
df.loc[80]  
df.loc[80] = [**"Europa"**, **"Irland"**]  
  
df[df[**'Land'**].isnull()]  
df.loc[107][**"Land"**] = **"Korea (Nord)"**df = df.drop([108])

1. Mit der Übersicht aus groupby() wird ersichtlich, dass Europa ein paar mal falsch geschrieben ist. Die Abfrage \*== ‘Europe’ liefert alle fehlerhaften Zeilen auf einmal, inkl. DF Index

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

1. Schlaufe: überprüft ob Anfangsbuchstabe klein geschrieben ist. Wenn ja, wird dieser entnommen und als Grossbuchstabe ersetzt.

count = 1  
for p in df[**"Land"**]:  
 if p[0].islower():  
 df[**"Land"**][count] = (p[0].upper() + df[**"Land"**][count][1:])  
 print(p) *# to see what is wrong* print(df[**"Land"**][count]) *# to see te correction* count += 1

1. Da der Titel Fehlerhaft ist und beim einlesen Schwierigkeiten bereitet, wird dieser ignoriert (header=none) und die Zeile ausgelassen (skiprows=1). Der Titel wird gleich manuel angegeben (names=[…)

df = pd.read\_csv(**'c1\_country\_src\_dirty.csv'**, header=None, skiprows=1, names=[**'Kontinent'**, **'Land'**], encoding=**'utf-8'**)

1. Mit df[-15] werden dien letzten Einträge angeschaut und es wird visual ermittelt, wo im Index der überflüssige Abschnitt anfängt.

df[-15:]  
  
df.drop(df.loc[242:252].index, inplace=True)

### C2

## Load

# Fragestellungen

# Reflexion

1. CSV-Export [↑](#footnote-ref-1)
2. Tabelle unter Quickstart Query «RGDPNA (1950-2019)» [↑](#footnote-ref-2)