COVID_19

May 16, 2020

1 Packageimport

[0]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import matplotlib.lines as mlines
from matplotlib import pylab

# macht die Plots größer
plt.rcParams['figure.figsize'] = (16,8)
```

2 Daten einlesen und erkunden

[0]:
```
# als Index wird die Spalte "dateRep" verwendet, die mithilfe von parse_dates␣
 ↪in einen Date-Time-Format umgewandelt wurde
# der zusätzliche Parameter dayfirst sorgt dafür, dass der Index im Format␣
 ↪YYYY-MM-DD angezeigt wird
covid19 = pd.read_csv('Daten.csv', index_col='dateRep',␣
 ↪parse_dates=['dateRep'], dayfirst = True)

# mithilfe von sort_index() wird sichergestellt, dass der "älteste" Eintrag␣
 ↪"oben" und der jüngste Eintrag am Ende des Dataframes steht
covid19 = covid19.sort_index()
```

2.1 Übersicht über die ersten Einträge des Datensatzes

[45]:
```
covid19.head()
```

[45]:
```
            day  month  year  …  countryterritoryCode  popData2018
continentExp
dateRep                       …
2019-12-31   31     12  2019  …                   LTU    2789533.0
Europe
2019-12-31   31     12  2019  …                   DEU   82927922.0
Europe
```

```
2019-12-31   31     12  2019  …                    KOR   51635256.0
Asia
2019-12-31   31     12  2019  …                    SGP    5638676.0
Asia
2019-12-31   31     12  2019  …                    KHM   16249798.0
Asia

[5 rows x 10 columns]
```

- da die Indexspalte bereits alle notwendigen Datumsinformationen enthält, werden die Spalten “day”, “month” und ”year nicht mehr benötigt und gelöscht

[0]:
```
del covid19['day']
del covid19['month']
del covid19['year']
```

- der veränderte Datensatz hat nun folgende Gestalt

[47]:
```
covid19.head()
```

[47]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2019-12-31      0       0  …   2789533.0       Europe
2019-12-31      0       0  …  82927922.0       Europe
2019-12-31      0       0  …  51635256.0         Asia
2019-12-31      0       0  …   5638676.0         Asia
2019-12-31      0       0  …  16249798.0         Asia

[5 rows x 7 columns]
```

2.2 Informationen zum Datensatz

- Größe des Datensatzes
- Spaltenbezeichnungen
- Datentypen
- statistische Kennzahlen

[48]:
```
print('Der Datensatz hat folgende Größe:\n' + 'Anzahl Instanzen: ' +␣
 ↪str(covid19.shape[0]) + '\n' + 'Anzahl Spalten:   ' + str(covid19.shape[1]))
```

```
Der Datensatz hat folgende Größe:
Anzahl Instanzen: 17158
Anzahl Spalten:   7
```

[49]:
```
covid19.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 17158 entries, 2019-12-31 to 2020-05-15
Data columns (total 7 columns):
```

```
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   cases                   17158 non-null  int64
 1   deaths                  17158 non-null  int64
 2   countriesAndTerritories 17158 non-null  object
 3   geoId                   17096 non-null  object
 4   countryterritoryCode    16938 non-null  object
 5   popData2018             16947 non-null  float64
 6   continentExp            17158 non-null  object
dtypes: float64(1), int64(2), object(4)
memory usage: 1.0+ MB
```

Auffälligkeiten: - fehlende Werte in 3 Spalten

[50]:
```
covid19.describe()
```

[50]:

	cases	deaths	popData2018
count	17158.000000	17158.000000	1.694700e+04
mean	256.771185	17.607821	5.216303e+07
std	1665.681923	125.649992	1.772810e+08
min	-2461.000000	-6.000000	1.000000e+03
25%	0.000000	0.000000	2.448255e+06
50%	2.000000	0.000000	9.630959e+06
75%	37.000000	1.000000	3.705886e+07
max	48529.000000	4928.000000	1.392730e+09

Auffälligkeiten: - Minimum in den Spalten 'cases' und 'deaths' ist negativ

2.3 Überprüfen auf fehlende Werte

[51]:
```
missing_values_gesamt = covid19.isnull().sum().to_frame(name = 'Anzahl␣
↪fehlender Werte')
missing_values = missing_values_gesamt[missing_values_gesamt['Anzahl fehlender␣
↪Werte'] > 0]
# Anzeigen der Spalten, bei denen mindestens ein Wert fehlt
missing_values
```

[51]:

	Anzahl fehlender Werte
geoId	62
countryterritoryCode	220
popData2018	211

2.3.1 Betrachtung der Spalte 'geoId'

[52]:
```
covid19[covid19.geoId.isnull()].head()
```

[52]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2020-03-15      2       0  …   2448255.0       Africa
2020-03-16      0       0  …   2448255.0       Africa
2020-03-17      0       0  …   2448255.0       Africa
2020-03-18      0       0  …   2448255.0       Africa
2020-03-19      0       0  …   2448255.0       Africa

[5 rows x 7 columns]
```

- Vermutung: fehlende Werte betreffen ausschließlich Namibia

[53]:
```
# Probe
covid19[covid19.countryterritoryCode =='NAM'].shape
# --> es gibt 62 Einträge für Namibia
```

[53]:
```
(62, 7)
```

[54]:
```
# Probe
covid19[np.logical_and(covid19.geoId.isnull(), covid19.countryterritoryCode␣
 ↪=='NAM')].shape
# --> bei allen 62 fehlenden Werten für 'geoID' ist auch 'countryterritoryCode'␣
 ↪'NAM' gesetzt
```

[54]:
```
(62, 7)
```

Lösung - für Namibia wird manuell eine neue geoId 'NB' vergeben - dafür wurde mithilfe der Funktion .unique() überprüft, welche 'geoId' noch nicht vergeben wurde

[55]:
```
np.sort(list(covid19['geoId'].unique()))
```

[55]:
```
array(['AD', 'AE', 'AF', 'AG', 'AI', 'AL', 'AM', 'AO', 'AR', 'AT', 'AU',
       'AW', 'AZ', 'BA', 'BB', 'BD', 'BE', 'BF', 'BG', 'BH', 'BI', 'BJ',
       'BM', 'BN', 'BO', 'BQ', 'BR', 'BS', 'BT', 'BW', 'BY', 'BZ', 'CA',
       'CD', 'CF', 'CG', 'CH', 'CI', 'CL', 'CM', 'CN', 'CO', 'CR', 'CU',
       'CV', 'CW', 'CY', 'CZ', 'DE', 'DJ', 'DK', 'DM', 'DO', 'DZ', 'EC',
       'EE', 'EG', 'EH', 'EL', 'ER', 'ES', 'ET', 'FI', 'FJ', 'FK', 'FO',
       'FR', 'GA', 'GD', 'GE', 'GG', 'GH', 'GI', 'GL', 'GM', 'GN', 'GQ',
       'GT', 'GU', 'GW', 'GY', 'HN', 'HR', 'HT', 'HU', 'ID', 'IE', 'IL',
       'IM', 'IN', 'IQ', 'IR', 'IS', 'IT', 'JE', 'JM', 'JO', 'JP',
       'JPG11668', 'KE', 'KG', 'KH', 'KM', 'KN', 'KR', 'KW', 'KY', 'KZ',
       'LA', 'LB', 'LC', 'LI', 'LK', 'LR', 'LS', 'LT', 'LU', 'LV', 'LY',
       'MA', 'MC', 'MD', 'ME', 'MG', 'MK', 'ML', 'MM', 'MN', 'MP', 'MR',
       'MS', 'MT', 'MU', 'MV', 'MW', 'MX', 'MY', 'MZ', 'NC', 'NE', 'NG',
       'NI', 'NL', 'NO', 'NP', 'NZ', 'OM', 'PA', 'PE', 'PF', 'PG', 'PH',
       'PK', 'PL', 'PR', 'PS', 'PT', 'PY', 'QA', 'RO', 'RS', 'RU', 'RW',
       'SA', 'SC', 'SD', 'SE', 'SG', 'SI', 'SK', 'SL', 'SM', 'SN', 'SO',
       'SR', 'SS', 'ST', 'SV', 'SX', 'SY', 'SZ', 'TC', 'TD', 'TG', 'TH',
```

```
       'TJ', 'TL', 'TN', 'TR', 'TT', 'TW', 'TZ', 'UA', 'UG', 'UK', 'US',
       'UY', 'UZ', 'VA', 'VC', 'VE', 'VG', 'VI', 'VN', 'XK', 'YE', 'ZA',
       'ZM', 'ZW', 'nan'], dtype='<U8')
```

[0]:
```
# ersetze die fehlenden Werte in Spalte 'geoId' mit 'NB' und vollziehe die␣
 ↪Änderung unmittelbar im Dataframe covid19
covid19['geoId'].fillna('NB', inplace = True)
```

2.3.2 Betrachtung der Spalte 'countryterritoryCode'

[56]:
```
# Welche Instanzen weisen fehlende Werte auf?
mis_val_territory = covid19[covid19.countryterritoryCode.isnull()]
mis_val_territory
```

[56]:

dateRep	cases	deaths	…	popData2018	continentExp
2019-12-31	0	0	…	3000.0	Other
2020-01-01	0	0	…	3000.0	Other
2020-01-02	0	0	…	3000.0	Other
2020-01-03	0	0	…	3000.0	Other
2020-01-04	0	0	…	3000.0	Other
…	…	…	…	…	…
2020-05-14	0	0	…	NaN	Africa
2020-05-15	0	0	…	NaN	Africa
2020-05-15	0	0	…	NaN	America
2020-05-15	0	0	…	NaN	America
2020-05-15	0	0	…	NaN	America

[220 rows x 7 columns]

[57]:
```
# Wie viele Einträge je 'countriesAndTerritories' fehlen?
mis_val_territory['countriesAndTerritories'].value_counts()
```

[57]:
```
Cases_on_an_international_conveyance_Japan    64
Anguilla                                      50
Bonaire, Saint Eustatius and Saba             44
Falkland_Islands_(Malvinas)                   42
Western_Sahara                                20
Name: countriesAndTerritories, dtype: int64
```

- für diese 5 Fälle werden manuell neue 'countryterritoryCode' vergeben, die sich nicht mit den bisherigen überschneiden
- die Überprüfung hierfür erfolgt mithilfe der Funktion .unique()

[58]:
```
np.sort(list(covid19['countryterritoryCode'].unique()))
```

[58]:
```
array(['ABW', 'AFG', 'AGO', 'ALB', 'AND', 'ARE', 'ARG', 'ARM', 'ATG',
       'AUS', 'AUT', 'AZE', 'BDI', 'BEL', 'BEN', 'BFA', 'BGD', 'BGR',
       'BHR', 'BHS', 'BIH', 'BLR', 'BLZ', 'BMU', 'BOL', 'BRA', 'BRB',
       'BRN', 'BTN', 'BWA', 'CAF', 'CAN', 'CHE', 'CHL', 'CHN', 'CIV',
       'CMR', 'COD', 'COG', 'COL', 'COM', 'CPV', 'CRI', 'CUB', 'CUW',
       'CYM', 'CYP', 'CZE', 'DEU', 'DJI', 'DMA', 'DNK', 'DOM', 'DZA',
       'ECU', 'EGY', 'ERI', 'ESP', 'EST', 'ETH', 'FIN', 'FJI', 'FRA',
       'FRO', 'GAB', 'GBR', 'GEO', 'GGY', 'GHA', 'GIB', 'GIN', 'GMB',
       'GNB', 'GNQ', 'GRC', 'GRD', 'GRL', 'GTM', 'GUM', 'GUY', 'HND',
       'HRV', 'HTI', 'HUN', 'IDN', 'IMN', 'IND', 'IRL', 'IRN', 'IRQ',
       'ISL', 'ISR', 'ITA', 'JAM', 'JEY', 'JOR', 'JPN', 'KAZ', 'KEN',
       'KGZ', 'KHM', 'KNA', 'KOR', 'KWT', 'LAO', 'LBN', 'LBR', 'LBY',
       'LCA', 'LIE', 'LKA', 'LSO', 'LTU', 'LUX', 'LVA', 'MAR', 'MCO',
       'MDA', 'MDG', 'MDV', 'MEX', 'MKD', 'MLI', 'MLT', 'MMR', 'MNE',
       'MNG', 'MNP', 'MOZ', 'MRT', 'MSR', 'MUS', 'MWI', 'MYS', 'NAM',
       'NCL', 'NER', 'NGA', 'NIC', 'NLD', 'NOR', 'NPL', 'NZL', 'OMN',
       'PAK', 'PAN', 'PER', 'PHL', 'PNG', 'POL', 'PRI', 'PRT', 'PRY',
       'PSE', 'PYF', 'QAT', 'ROU', 'RUS', 'RWA', 'SAU', 'SDN', 'SEN',
       'SGP', 'SLE', 'SLV', 'SMR', 'SOM', 'SRB', 'SSD', 'STP', 'SUR',
       'SVK', 'SVN', 'SWE', 'SWZ', 'SXM', 'SYC', 'SYR', 'TCA', 'TCD',
       'TGO', 'THA', 'TJK', 'TLS', 'TTO', 'TUN', 'TUR', 'TWN', 'TZA',
       'UGA', 'UKR', 'URY', 'USA', 'UZB', 'VAT', 'VCT', 'VEN', 'VGB',
       'VIR', 'VNM', 'XKX', 'YEM', 'ZAF', 'ZMB', 'ZWE', 'nan'],
      dtype='<U3')
```

Lösung

Folgende Codes werden vergeben

- ICJ: Cases_on_an_international_conveyance_Japan
- ANG: Anguilla
- BON: Bonaire, Saint Eustatius and Saba
- FAL: Falkland_Islands_(Malvinas)
- WSH: Western_Sahara

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] ==␣
 ↪'Cases_on_an_international_conveyance_Japan', ['countryterritoryCode']] =␣
 ↪'ICJ'
covid19.loc[covid19['countriesAndTerritories'] == 'Anguilla',␣
 ↪['countryterritoryCode']] = 'ANG'
covid19.loc[covid19['countriesAndTerritories'] == 'Bonaire, Saint Eustatius and␣
 ↪Saba', ['countryterritoryCode']] = 'BON'
covid19.loc[covid19['countriesAndTerritories'] ==␣
 ↪'Falkland_Islands_(Malvinas)', ['countryterritoryCode']] = 'FAL'
```

```
covid19.loc[covid19['countriesAndTerritories'] == 'Western_Sahara',␣
↪['countryterritoryCode']] = 'WSH'
```

2.3.3 Betrachtung der Spalte 'popData2018'

[60]:
```
# Welche Instanzen weisen fehlende Werte auf?
mis_val_popData = covid19[covid19.popData2018.isnull()]
mis_val_popData
```

[60]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2020-03-22      1       0  …         NaN       Africa
2020-03-23      0       0  …         NaN       Africa
2020-03-24      0       0  …         NaN       Africa
2020-03-25      0       0  …         NaN       Africa
2020-03-26      3       0  …         NaN       Africa
…             …     …  …  …         …           …
2020-05-15      0       0  …         NaN       Africa
2020-05-15      0       0  …         NaN      America
2020-05-15      0       0  …         NaN      America
2020-05-15      0       0  …         NaN       Africa
2020-05-15      0       0  …         NaN      America

[211 rows x 7 columns]
```

[61]:
```
# Wie viele Einträge je 'countriesAndTerritories' fehlen?
mis_val_popData['countriesAndTerritories'].value_counts()
```

[61]:
```
Eritrea                             55
Anguilla                            50
Bonaire, Saint Eustatius and Saba   44
Falkland_Islands_(Malvinas)         42
Western_Sahara                      20
Name: countriesAndTerritories, dtype: int64
```

–> die 5 betroffenen Ländern werden nun im Einzelfall betrachtet

Eritrea

[62]:
```
covid19[covid19.countriesAndTerritories == 'Eritrea'].head()
```

[62]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2020-03-22      1       0  …         NaN       Africa
2020-03-23      0       0  …         NaN       Africa
2020-03-24      0       0  …         NaN       Africa
2020-03-25      0       0  …         NaN       Africa
2020-03-26      3       0  …         NaN       Africa
```

```
[5 rows x 7 columns]
```

Lösung - die Bevölkerungszahl wird manuell auf den Wert 6.500.000 gesetzt

Quelle:

"Im Jahr 2018 betrug die Einwohnerzahl von Eritrea geschätzt rund 6,05 Millionen Personen." (https://de.statista.com/statistik/daten/studie/417176/umfrage/gesamtbevoelkerung-von-eritrea/)

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] == 'Eritrea', ['popData2018']] =␣
↪6500000
```

Anguilla

[64]:
```
covid19[covid19.countriesAndTerritories == 'Anguilla'].head()
```

[64]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2020-03-27      2       0  …         NaN       America
2020-03-28      0       0  …         NaN       America
2020-03-29      0       0  …         NaN       America
2020-03-30      0       0  …         NaN       America
2020-03-31      0       0  …         NaN       America

[5 rows x 7 columns]
```

Lösung - die Bevölkerungszahl wird manuell auf den Wert 17.422 gesetzt

Quelle:

"Bevölkerung: 17.422 (Juli 2018 est.)" (https://www.indexmundi.com/de/anguilla/bevolkerung.html)

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] == 'Anguilla', ['popData2018']]␣
↪= 17422
```

Bonaire, Saint Eustatius and Saba

[66]:
```
covid19[covid19.countriesAndTerritories == 'Bonaire, Saint Eustatius and Saba'].
↪head()
```

[66]:
```
            cases  deaths  … popData2018 continentExp
dateRep                    …
2020-04-02      2       0  …         NaN       America
2020-04-03      0       0  …         NaN       America
2020-04-04      0       0  …         NaN       America
2020-04-05      0       0  …         NaN       America
2020-04-06      0       0  …         NaN       America
```

```
[5 rows x 7 columns]
```

Lösung - die Bevölkerungszahl wird manuell auf den Wert 25.157 gesetzt

Quelle:

"Einwohner Schätzung 2019-01-01: 25.157" (https://www.citypopulation.de/de/caribbeannetherlands/)

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] == 'Bonaire, Saint Eustatius and␣
↪Saba', ['popData2018']] = 25157
```

Falkland Islands (Malvinas)

[68]:
```
covid19[covid19.countriesAndTerritories == 'Falkland_Islands_(Malvinas)'].head()
```

[68]:

dateRep	cases	deaths	…	popData2018	continentExp
			…		
2020-04-04	1	0	…	NaN	America
2020-04-05	0	0	…	NaN	America
2020-04-06	1	0	…	NaN	America
2020-04-07	0	0	…	NaN	America
2020-04-08	3	0	…	NaN	America

```
[5 rows x 7 columns]
```

Lösung - die Bevölkerungszahl wird manuell auf den Wert 2.922 gesetzt

Quelle:

"Einwohnerzahl: 2.922" (https://de.wikipedia.org/wiki/Falklandinseln)

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] ==␣
↪'Falkland_Islands_(Malvinas)', ['popData2018']] = 2922
```

Western_Sahara

[70]:
```
covid19[covid19.countriesAndTerritories == 'Western_Sahara'].head()
```

[70]:

dateRep	cases	deaths	…	popData2018	continentExp
			…		
2020-04-26	6	0	…	NaN	Africa
2020-04-27	0	0	…	NaN	Africa
2020-04-28	0	0	…	NaN	Africa
2020-04-29	0	0	…	NaN	Africa
2020-04-30	0	0	…	NaN	Africa

```
[5 rows x 7 columns]
```

Lösung - die Bevölkerungszahl wird manuell auf den Wert 554.200 gesetzt

Quelle:

"Einwohner Projektion (P) 2018-07-01: 554.200" (https://www.citypopulation.de/WesternSahara_d.html)

[0]:
```
covid19.loc[covid19['countriesAndTerritories'] == 'Western_Sahara',␣
↪['popData2018']] = 554200
```

2.4 Überprüfen auf weitere Auffälligkeiten

2.4.1 negative Erfassungszahlen für (Neu-)Infizierte und Todesfälle

Neuinfektionen

[72]:
```
covid19[covid19.cases < 0]
```

[72]:

dateRep	cases	deaths	…	popData2018	continentExp
			…		
2020-03-10	-9	1	…	3000.0	Other
2020-04-19	-713	410	…	46723749.0	Europe
2020-04-29	-105	3	…	2789533.0	Europe
2020-05-03	-161	16	…	10281762.0	Europe
2020-05-07	-2461	49	…	17084357.0	America
2020-05-09	-1480	50	…	17084357.0	America
2020-05-11	-9	0	…	33785.0	Europe
2020-05-12	-50	18	…	17084357.0	America

[8 rows x 7 columns]

- mehrere Beobachtungen weisen negative Werte in der Spalte 'cases' auf
- diese erscheinen unplausibel und deuten auf eine Fehlerfassung hin

Lösung

- Umkehr der Vorzeichen

[0]:
```
# Funktion, die das Vorzeichen für negative Werte umkehrt
korr_negativ = lambda i: i*-1 if i < 0 else i

# Anwendung der Lambda-Funktion auf die Spalte 'cases'
covid19['cases'] = covid19['cases'].apply(korr_negativ)
```

Todesfälle

[74]:
```
covid19[covid19.deaths < 0]
```

[74]:

dateRep	cases	deaths	…	popData2018	continentExp
			…		
2020-05-13	78	-6	…	5797446.0	Europe

[1 rows x 7 columns]

- Handlungsbedarf in einer Zeile

[0]:
```
# Anwendung der Lambda-Funktion auf die Spalte 'deaths'
covid19['deaths'] = covid19['deaths'].apply(korr_negativ)
```

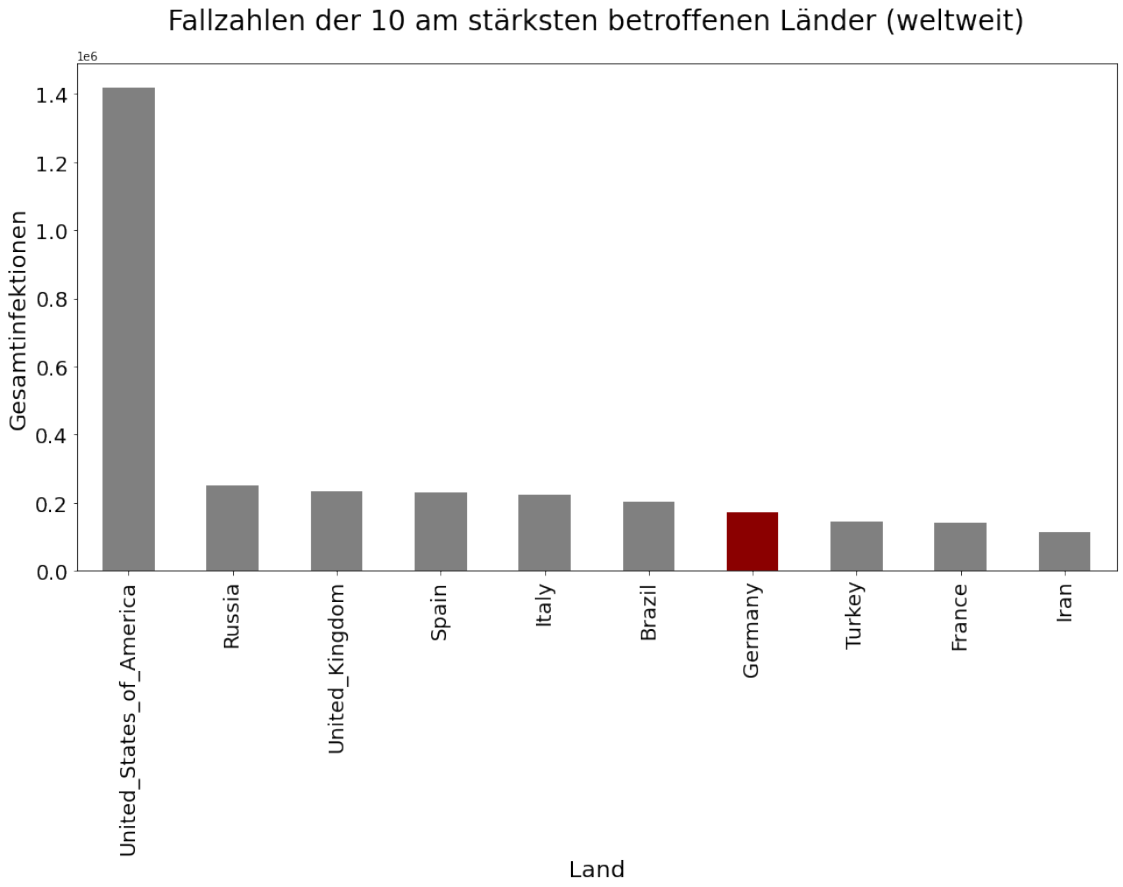
3 Visualisierung der Daten

3.1 Die 10 am stärksten betroffenen Länder weltweit

3.1.1 Gesamtinfektionen

[76]:
```
# im Dataframe covid19 werden die Einträge nach Ländern␣
 ↪('countriesAndTerritories') gruppiert
# das auf diese Weise neu entstandene Dataframe beinhaltet nun lediglich␣
 ↪aggregierte Werte (Summen) und keine Zeitdimension mehr
# in einem Säulendiagramm werden die 10 Länder visualisiert, die die größte␣
 ↪Anzahl an Gesamtinfektionen aufweisen
covid19.groupby('countriesAndTerritories')['cases'].sum().nlargest(10).
 ↪plot(kind = 'bar', color = ['gray', 'gray', 'gray', 'gray', 'gray', 'gray',␣
 ↪'darkred', 'gray', 'gray', 'gray'])

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.title('Fallzahlen der 10 am stärksten betroffenen Länder (weltweit)',␣
 ↪fontsize = 24, pad = 30)
plt.ylabel('Gesamtinfektionen', fontsize = 20)
plt.xlabel('Land', fontsize = 20)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

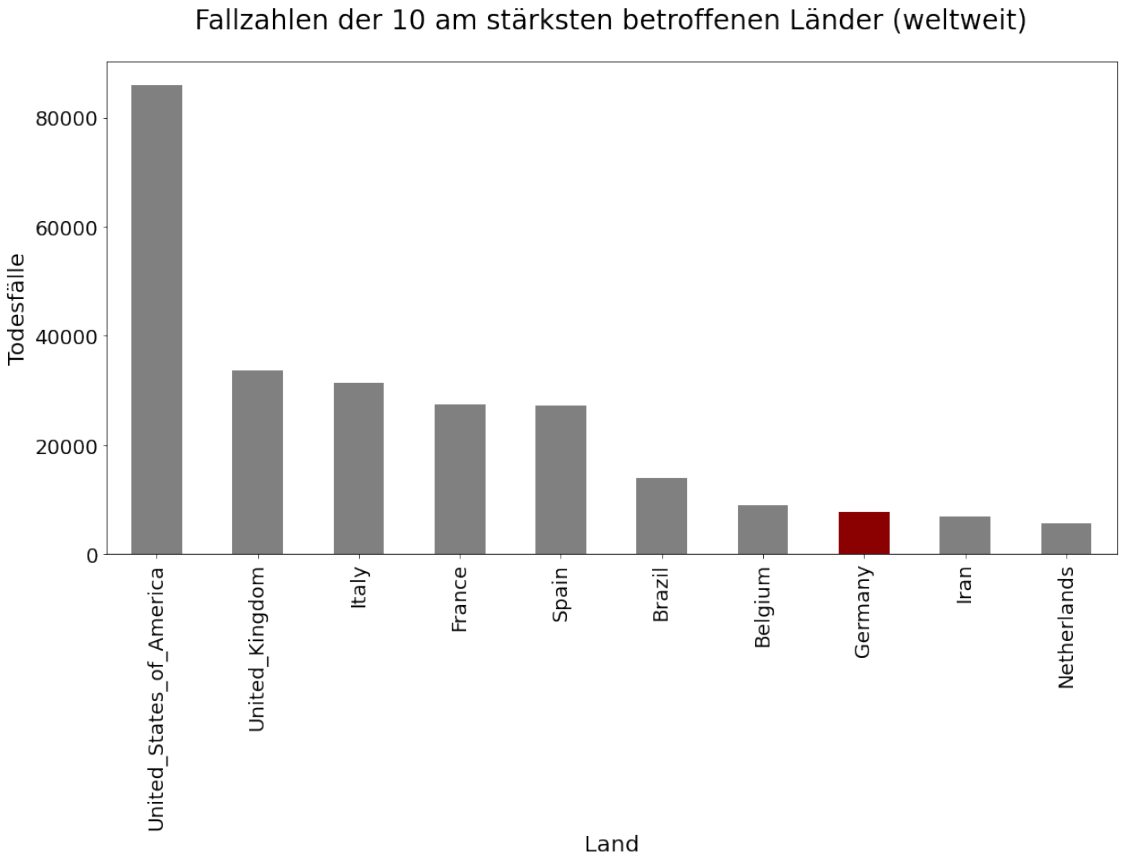
[76]:
```
(array([      0.,  200000.,  400000.,  600000.,  800000., 1000000.,
        1200000., 1400000., 1600000.]),
 <a list of 9 Text major ticklabel objects>)
```



3.1.2 Todesfälle

[77]:
```
# im Dataframe covid19 werden die Einträge nach Ländern␣
 ↪('countriesAndTerritories') gruppiert
# das auf diese Weise neu entstandene Dataframe beinhaltet nun lediglich␣
 ↪aggregierte Werte (Summen) und keine Zeitdimension mehr
# in einem Säulendiagramm werden die 10 Länder visualisiert, die die größte␣
 ↪Anzahl an Todesfällen aufweisen
covid19.groupby('countriesAndTerritories')['deaths'].sum().nlargest(10).
 ↪plot(kind = 'bar', color = ['gray', 'gray', 'gray', 'gray', 'gray', 'gray',␣
 ↪'gray', 'darkred', 'gray', 'gray'])

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.title('Fallzahlen der 10 am stärksten betroffenen Länder (weltweit)',␣
 ↪fontsize = 24, pad = 30)
plt.ylabel('Todesfälle', fontsize = 20)
plt.xlabel('Land', fontsize = 20)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

[77]:
```
(array([     0.,  20000.,  40000.,  60000.,  80000., 100000.]),
 <a list of 6 Text major ticklabel objects>)
```



3.2 Die 10 am stärksten betroffenen Länder Europas

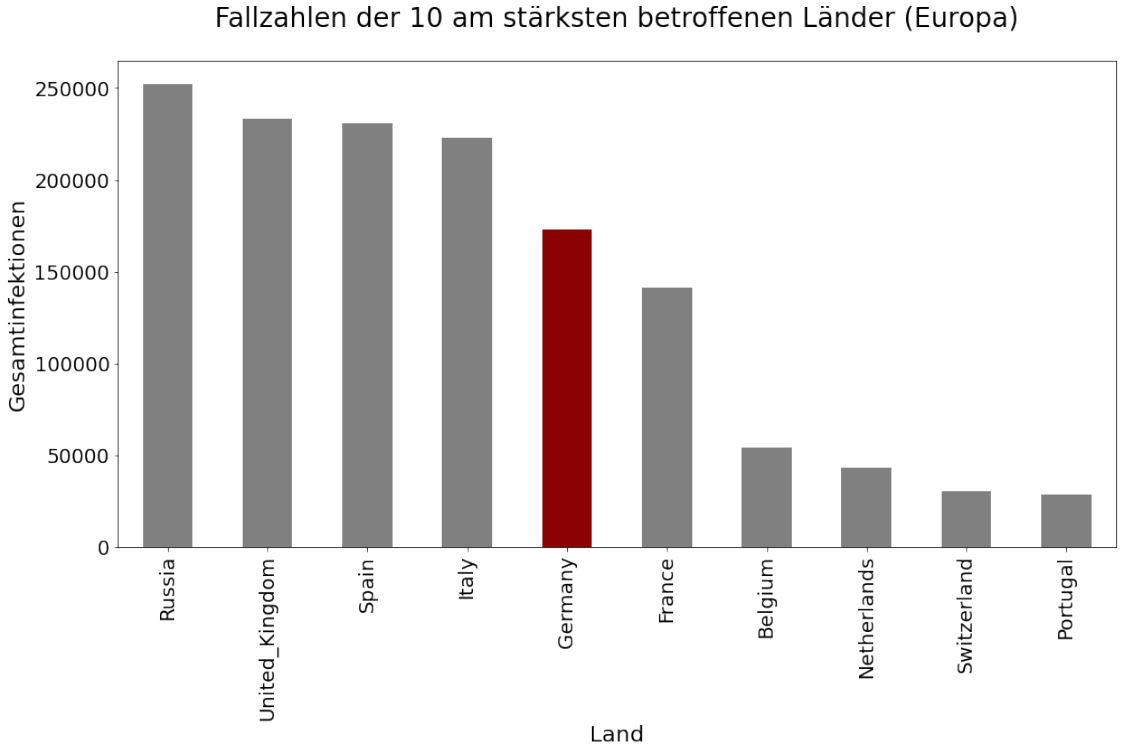
3.2.1 Gesamtinfektionen

[0]:
```
# Erstellung eines separaten Dataframes, das aus dem Dataframe covid19 alle␣
 ↪Einträge für Europa herausfiltert
europa = covid19[covid19.continentExp == 'Europe']
```

[79]:
```
# im Dataframe europa werden die Einträge nach Ländern␣
 ↪('countriesAndTerritories') gruppiert
# das auf diese Weise neu entstandene Dataframe beinhaltet nun lediglich␣
 ↪aggregierte Werte (Summen) und keine Zeitdimension mehr
# in einem Säulendiagramm werden die 10 Länder visualisiert, die die größte␣
 ↪Anzahl an Gesamtinfektionen aufweisen
europa.groupby('countriesAndTerritories')['cases'].sum().nlargest(10).plot(kind␣
 ↪= 'bar', color = ['gray', 'gray', 'gray', 'gray', 'darkred', 'gray', 'gray',␣
 ↪'gray', 'gray', 'gray'])
```

```
# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.title('Fallzahlen der 10 am stärksten betroffenen Länder (Europa)',␣
 ↪fontsize = 24, pad = 30)
plt.ylabel('Gesamtinfektionen', fontsize = 20)
plt.xlabel('Land', fontsize = 20)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

[79]:
```
(array([     0.,  50000., 100000., 150000., 200000., 250000., 300000.]),
 <a list of 7 Text major ticklabel objects>)
```

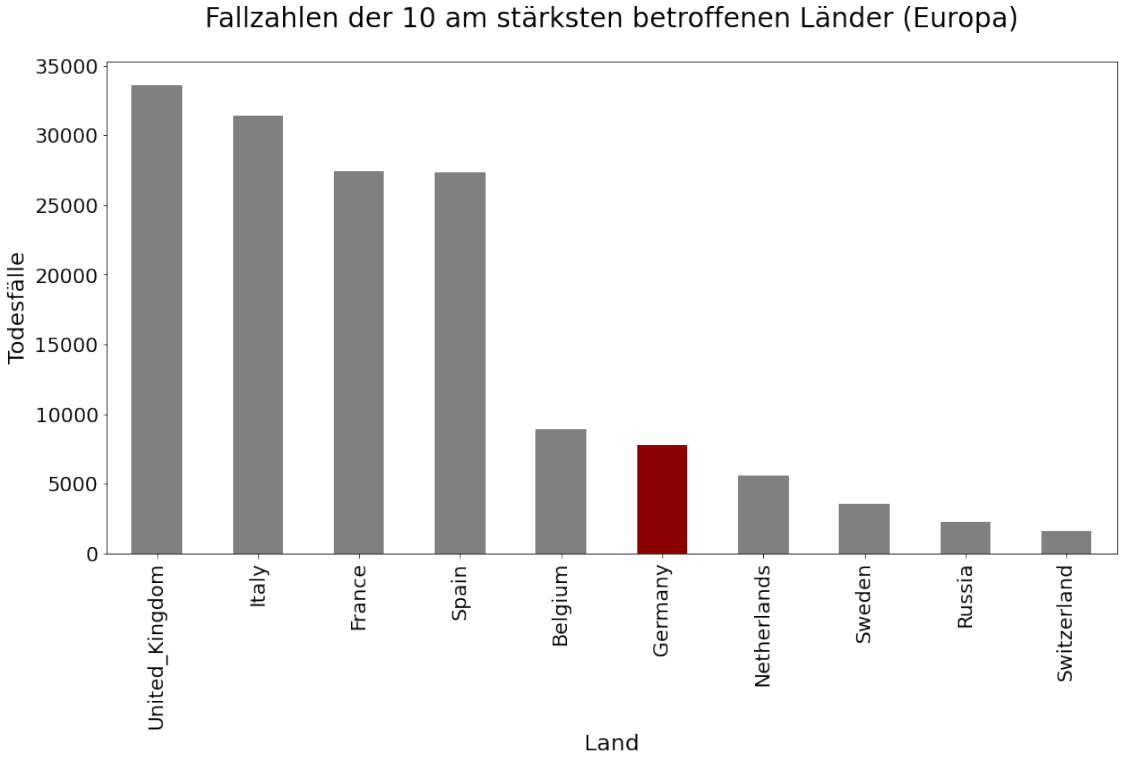


3.2.2 Todesfälle

[80]:
```
# im Dataframe europa werden die Einträge nach Ländern␣
 ↪('countriesAndTerritories') gruppiert
# das auf diese Weise neu entstandene Dataframe beinhaltet nun lediglich␣
 ↪aggregierte Werte (Summen) und keine Zeitdimension mehr
# in einem Säulendiagramm werden die 10 Länder visualisiert, die die größte␣
 ↪Anzahl an Todesfällen aufweisen
europa.groupby('countriesAndTerritories')['deaths'].sum().nlargest(10).
 ↪plot(kind = 'bar', color = ['gray', 'gray', 'gray', 'gray', 'gray',␣
 ↪'darkred', 'gray', 'gray', 'gray', 'gray'])
```

```
# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.title('Fallzahlen der 10 am stärksten betroffenen Länder (Europa)',␣
 ↪fontsize = 24, pad = 30)
plt.ylabel('Todesfälle', fontsize = 20)
plt.xlabel('Land', fontsize = 20)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

[80]:
```
(array([    0.,  5000., 10000., 15000., 20000., 25000., 30000., 35000.,
        40000.]), <a list of 9 Text major ticklabel objects>)
```

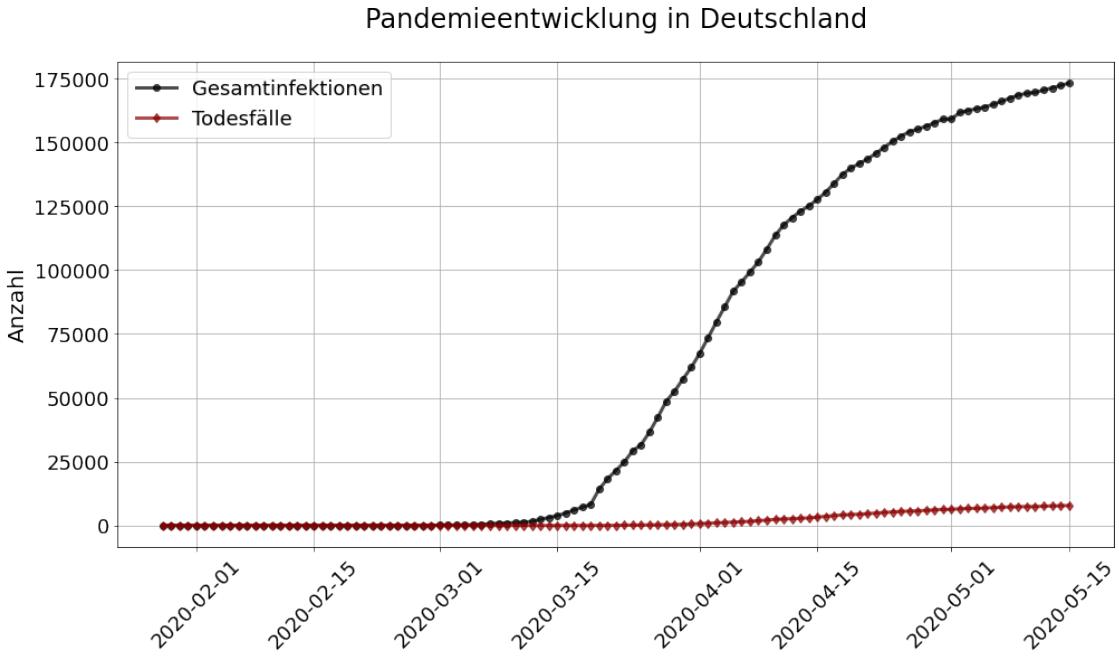


3.3 Entwicklungen in Deutschland

[81]:
```
# Erstellung eines separaten Dataframes, das aus dem Dataframe covid19 alle␣
 ↪Einträge für Deutschland herausfiltert
germany = covid19[covid19['countriesAndTerritories'] == 'Germany']

# Erstellung zweier zusätzlicher Spalten, die die kumulierten Gesamtinfektionen␣
 ↪und Todesfälle aufzeigen
germany['Fälle kumuliert'] = germany['cases'].cumsum()
germany['Todesfälle kumuliert'] = germany['deaths'].cumsum()
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:6:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

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

[82]:
```
# Kürzen des Dataframes, sodass die Einträge mit dem ersten registrierten Fall␣
 ↪beginnen (Datum gesondert ermittelt)
# alle Einträge mit 0 vor dem 28.01.2020 wurden 'abgeschnitten'
germany = germany['2020-01-28' : ]

# Zeichnen von je einer Linie für Gesamtinfektionen und Todesfälle
plt.plot(germany['Fälle kumuliert'], label = 'Gesamtinfektionen', alpha = 0.7,␣
 ↪color = 'black', lw = 3, marker = 'o')
plt.plot(germany['Todesfälle kumuliert'], label = 'Todesfälle', alpha = 0.7,␣
 ↪color = 'darkred', lw = 3, marker = 'd')

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.title('Pandemieentwicklung in Deutschland', fontsize = 24, pad = 30)
plt.ylabel('Anzahl', fontsize = 20)
plt.xticks(rotation = 45)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
plt.legend(fontsize = 18)
plt.grid()
```



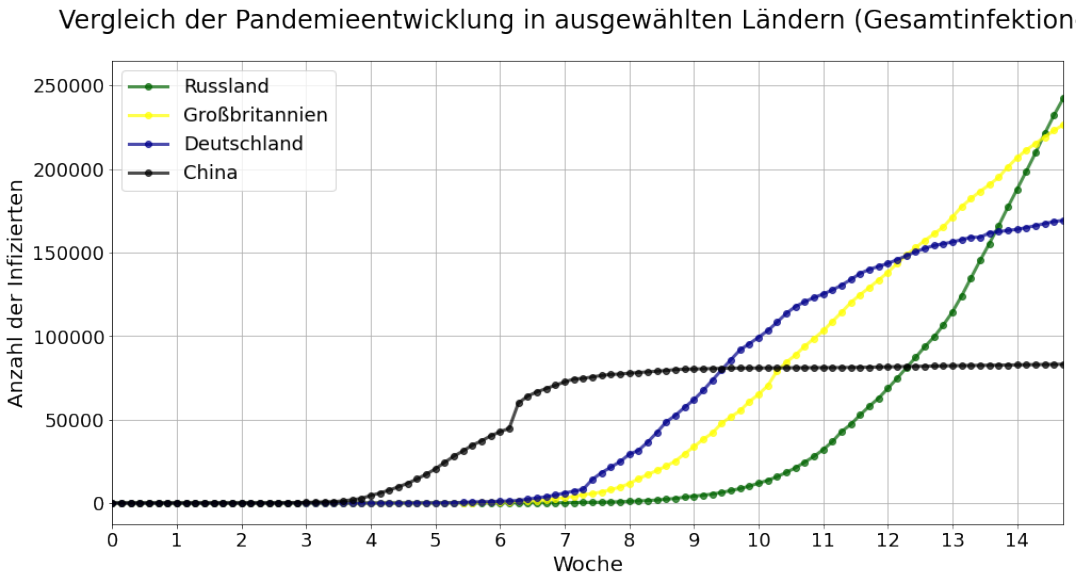
3.4 Vergleich mit Großbritannien (höchste Anzahl Todesfälle in Europa), Russland (höchste europäische Infektionszahl) und China (Ursprungsland)

[0]:
```
# Entfernen des Zeitindex zur Vergleichbarkeit der Entwicklungen mit anderen␣
 ↪Ländern
germany = germany.reset_index()
del germany['dateRep']



# Erstellung eines separaten Dataframes, das aus dem Dataframe covid19 alle␣
 ↪Einträge für China herausfiltert
china = covid19[covid19['countriesAndTerritories'] == 'China']
# Entfernen des Zeitindex zur Vergleichbarkeit der Entwicklungen mit anderen␣
 ↪Ländern
china = china.reset_index()
del china['dateRep']
# Erstellung zweier zusätzlicher Spalten, die die kumulierten Gesamtinfektionen␣
 ↪und Todesfälle aufzeigen
china['Fälle kumuliert'] = china['cases'].cumsum()
china['Todesfälle kumuliert'] = china['deaths'].cumsum()
```

```
# Erstellung eines separaten Dataframes, das aus dem Dataframe covid19 alle␣
 ↪Einträge für Großbritannien herausfiltert
uk = covid19[covid19['countriesAndTerritories'] == 'United_Kingdom']
# Kürzen des Dataframes, sodass die Einträge mit dem ersten registrierten Fall␣
 ↪beginnen (Datum gesondert ermittelt)
uk = uk['2020-01-31' :]
# Entfernen des Zeitindex zur Vergleichbarkeit der Entwicklungen mit anderen␣
 ↪Ländern
uk = uk.reset_index()
del uk['dateRep']
# Erstellung zweier zusätzlicher Spalten, die die kumulierten Gesamtinfektionen␣
 ↪und Todesfälle aufzeigen
uk['Fälle kumuliert'] = uk['cases'].cumsum()
uk['Todesfälle kumuliert'] = uk['deaths'].cumsum()



# Erstellung eines separaten Dataframes, das aus dem Dataframe covid19 alle␣
 ↪Einträge für Russland herausfiltert
russia = covid19[covid19['countriesAndTerritories'] == 'Russia']
# Kürzen des Dataframes, sodass die Einträge mit dem ersten registrierten Fall␣
 ↪beginnen (Datum gesondert ermittelt)
russia = russia['2020-02-01' :]
# Entfernen des Zeitindex zur Vergleichbarkeit der Entwicklungen mit anderen␣
 ↪Ländern
russia = russia.reset_index()
del russia['dateRep']
# Erstellung zweier zusätzlicher Spalten, die die kumulierten Gesamtinfektionen␣
 ↪und Todesfälle aufzeigen
russia['Fälle kumuliert'] = russia['cases'].cumsum()
russia['Todesfälle kumuliert'] = russia['deaths'].cumsum()
```

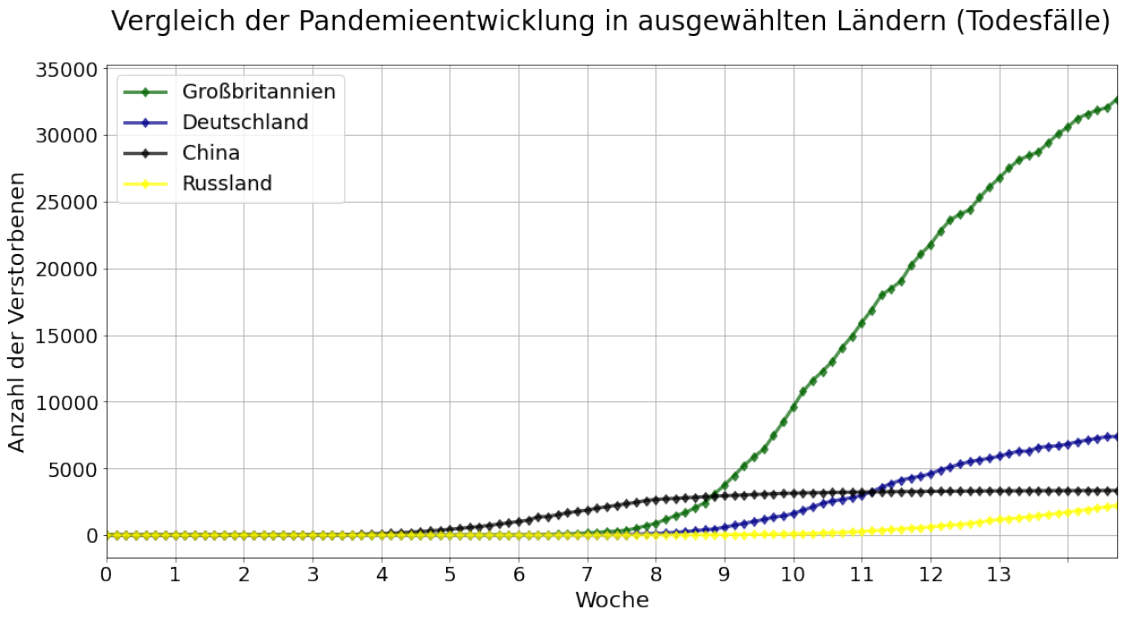
[87]:
```
# Zeichnen von je einer Linie für Gesamtinfektionen der 4 betrachteten Länder
plt.plot(russia['Fälle kumuliert'], label = 'Russland', alpha = 0.7, color =␣
 ↪'darkgreen', lw = 3, marker = 'o')
plt.plot(uk['Fälle kumuliert'], label = 'Großbritannien', alpha = 0.7, color =␣
 ↪'yellow', lw = 3, marker = 'o')
plt.plot(germany['Fälle kumuliert'], label = 'Deutschland', alpha = 0.7, color␣
 ↪= 'darkblue', lw = 3, marker = 'o')
plt.plot(china['Fälle kumuliert'], label = 'China', alpha = 0.7, color =␣
 ↪'black', lw = 3, marker = 'o')

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
plt.legend(fontsize = 18)
```

```
plt.xlim(0,103)
plt.xticks(np.arange(0, 103, 7), range(16))
plt.xlabel('Woche', fontsize = 20)
plt.ylabel('Anzahl der Infizierten', fontsize = 20)
plt.title('Vergleich der Pandemieentwicklung in ausgewählten Ländern␣
 ↪(Gesamtinfektionen)', fontsize = 24, pad = 30)
plt.grid()
```



[89]:
```
# Zeichnen von je einer Linie für Todesfälle der 4 betrachteten Länder
plt.plot(uk['Todesfälle kumuliert'], label = 'Großbritannien', alpha = 0.7,␣
 ↪color = 'darkgreen', lw = 3, marker = 'd')
plt.plot(germany['Todesfälle kumuliert'], label = 'Deutschland', alpha = 0.7,␣
 ↪color = 'darkblue', lw = 3, marker = 'd')
plt.plot(china['Todesfälle kumuliert'], label = 'China', alpha = 0.7, color =␣
 ↪'black', lw = 3, marker = 'd')
plt.plot(russia['Todesfälle kumuliert'], label = 'Russland', alpha = 0.7, color␣
 ↪= 'yellow', lw = 3, marker = 'd')

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
plt.legend(fontsize = 18)
plt.xlim(0,103)
plt.xticks(np.arange(0, 103, 7), range(14))
plt.xlabel('Woche', fontsize = 20)
plt.ylabel('Anzahl der Verstorbenen', fontsize = 20)
```

```
plt.title('Vergleich der Pandemieentwicklung in ausgewählten Ländern␣
 ↪(Todesfälle)', fontsize = 24, pad = 30)
plt.grid()
```



3.5 Abhängigkeit zwischen Gesamtinfektionen und Todesfällen

[91]:
```
# in einem neuen Dataframe world werden aus covid19 die Einträge nach Ländern␣
 ↪('countriesAndTerritories') gruppiert und gespeichert
# alle numerischen Werte werden als Summe aggregiert
world = covid19.groupby(['countriesAndTerritories']).sum()

# Erstellung einer zusätzlicher Spalten für den Kontinent mit Initialwert ''␣
 ↪und Setzen der Bevölkerungszahl auf 0
# durch die Aggregationsmethode .sum() werden auch die Bevölkerungszahlen␣
 ↪addiert
# um die tatsächliche Bevölkerungszahl zu erhalten, wird sie im nächsten␣
 ↪Schritt separat bestimmt
world['Kontinent'] = ''
world['popData2018'] = 0

# Anlegen einer Liste mit allen Ländern (jedes Land kommt nur einmal in dieser␣
 ↪Liste vor)
land = list(covid19.countriesAndTerritories.unique())

# Bestimme für jedes Land in der Liste den Kontinent und die Bevölkerungszahl␣
 ↪aus dem Dataframe covid19 und überschreibe die Einträge
```

```
# entsprechend im Dataframe world
for l in land:
  cont = covid19[covid19.countriesAndTerritories == l]['continentExp'].unique()
  cont = cont[0]
  world.loc[l, 'Kontinent'] = cont
  pop = covid19[covid19.countriesAndTerritories == l]['popData2018'].unique()
  pop = pop[0]
  world.loc[l, 'popData2018'] = pop

# Dictionary, das Kontinenten Farben zuordnet
col = {'Africa':'black',
       'America':'red',
       'Asia':'yellow',
       'Europe':'green',
       'Oceania':'blue',
       'Other' : 'purple'}

# Erstellung einer neuen Spalte, das jedem Land eine Farbe entsprechend des␣
 ↪Kontinents zuordnet
world['Farbe'] = world.Kontinent.map(col)
# Erstellung weiterer Spalten für die Auswertung
#### Wie viel Prozent der Bevölkerung haben sich infiziert?
world['% Infizierte'] = world.cases / world.popData2018 * 100
#### Wie hoch ist die Mortalitätsrate bei den Infizierten?
world['% Verstorbene'] = world.deaths / world.cases * 100
#### Wie viele Fälle sind derzeit aktiv erkrankt oder geheilt?
world['Aktive Fälle inkl. Geheilte'] = world.cases - world.deaths

# Anzeigen des neuen Dataframes
world
```

[91]:

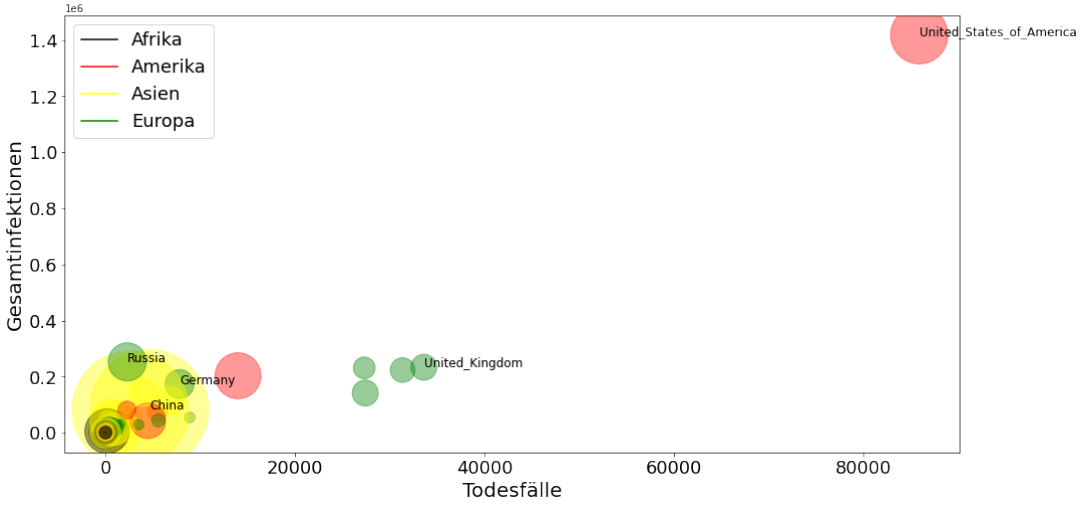
countriesAndTerritories	cases	…	Aktive Fälle inkl. Geheilte
Afghanistan	5339	…	5203
Albania	898	…	867
Algeria	6442	…	5913
Andorra	761	…	712
Angola	48	…	46
…	…	…	…
Vietnam	312	…	312
Western_Sahara	6	…	6
Yemen	87	…	74
Zambia	654	…	647
Zimbabwe	37	…	33

[210 rows x 8 columns]

[94]:
```
# Darstellung der Abhängigkeit zwischen Gesamtinfektionen und Todesfällen pro␣
 ↪Land
# Farbe des Punktes ergibt repräsentiert den jeweiligen Kontinent
# Größe des Punktes ergibt sich aus der Bevölkerungszahl (um die Punkte in␣
 ↪einer sinnvollen Größe darzustellen wurden sie mit dem Faktor
# 1/100000 herunterskaliert)
plt.scatter(world.deaths, world.cases, s = world.popData2018/100000, alpha = 0.
 ↪4, c = world.Farbe)

# Beschriftungen einzelner Datenpunkte
liste = ['Germany', 'Russia', 'United_Kingdom', 'China',␣
 ↪'United_States_of_America']
for l in liste:
  x = world.loc[l]['deaths']
  y = world.loc[l]['cases']
  pylab.text(x , y, l, fontsize = 12)

# Benutzerdefinierte Legende
black_line = mlines.Line2D([], [], color='black',markersize=15, label='Afrika')
red_line = mlines.Line2D([], [], color='red', markersize=15, label='Amerika')
yellow_line = mlines.Line2D([], [], color='yellow', markersize=15,␣
 ↪label='Asien')
green_line = mlines.Line2D([], [], color='green', markersize=15, label='Europa')
# blue_line = mlines.Line2D([], [], color='blue', markersize=15,␣
 ↪label='Ozeanien')
# purple_line = mlines.Line2D([], [], color='purple', markersize=15,␣
 ↪label='Other')
# handles = [black_line, red_line, yellow_line, green_line, blue_line,␣
 ↪purple_line]
handles = [black_line, red_line, yellow_line, green_line]
labels = [h.get_label() for h in handles]
plt.legend(handles=handles, labels=labels, fontsize = 18)

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.xlabel('Todesfälle', fontsize = 20)
plt.ylabel('Gesamtinfektionen', fontsize = 20)
plt.title('Abhängigkeit zwischen Gesamtinfektionen und Todesfällen der␣
 ↪einzelnen Länder (mit USA)', fontsize = 24, pad = 30)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

[94]:
```
(array([-200000.,       0.,  200000.,  400000.,  600000.,  800000.,
        1000000., 1200000., 1400000., 1600000.]),
 <a list of 10 Text major ticklabel objects>)
```

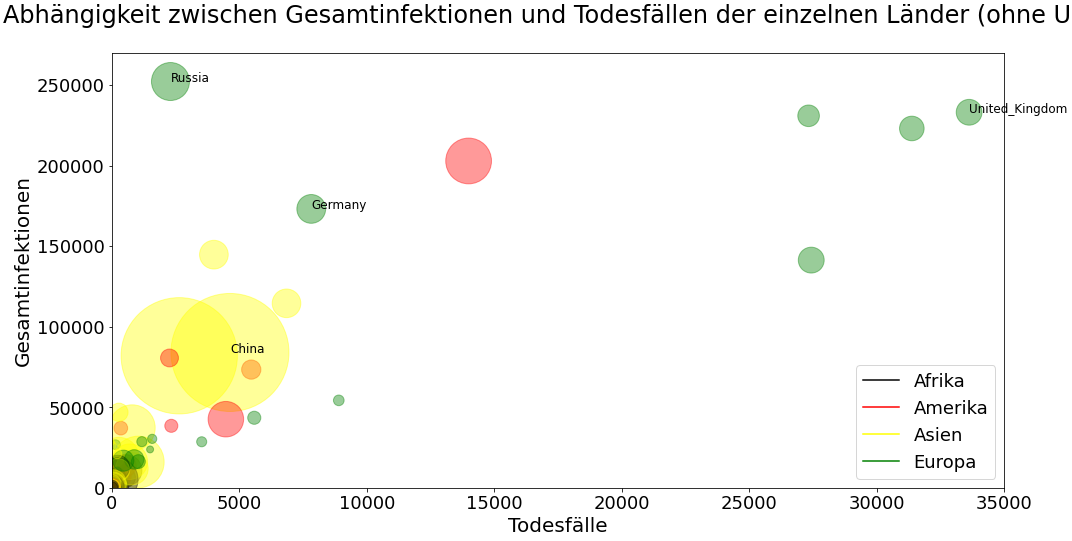
Abhängigkeit zwischen Gesamtinfektionen und Todesfällen der einzelnen Länder (mit USA)



[97]:
```
# erneute Darstellung der Abhängigkeit zwischen Gesamtinfektionen und␣
 ↪Todesfällen pro Land
# Farbe des Punktes ergibt repräsentiert den jeweiligen Kontinent
# Größe des Punktes ergibt sich aus der Bevölkerungszahl (um die Punkte in␣
 ↪einer sinnvollen Größe darzustellen wurden sie mit dem Faktor
# 1/100000 heruntersklaiert)
plt.scatter(world.deaths, world.cases, s = world.popData2018/100000, alpha = 0.
 ↪4, c = world.Farbe)

# Setzen von Limits für die Achsenwerte, um in einen bestimmten Bereich zu␣
 ↪zoomen
plt.xlim(0,35000)
plt.ylim(0,270000)

# Beschriftungen einzelner Datenpunkte
liste = ['Germany', 'Russia', 'United_Kingdom', 'China']
for l in liste:
  x = world.loc[l]['deaths']
  y = world.loc[l]['cases']
  pylab.text(x , y, l, fontsize = 12)

# Benutzerdefinierte Legende
black_line = mlines.Line2D([], [], color='black',markersize=15, label='Afrika')
red_line = mlines.Line2D([], [], color='red', markersize=15, label='Amerika')
yellow_line = mlines.Line2D([], [], color='yellow', markersize=15,␣
 ↪label='Asien')
green_line = mlines.Line2D([], [], color='green', markersize=15, label='Europa')
```

```
# blue_line = mlines.Line2D([], [], color='blue', markersize=15,␣
 ↪label='Ozeanien')
# purple_line = mlines.Line2D([], [], color='purple', markersize=15,␣
 ↪label='Other')
# handles = [black_line, red_line, yellow_line, green_line, blue_line,␣
 ↪purple_line]
handles = [black_line, red_line, yellow_line, green_line]
labels = [h.get_label() for h in handles]
plt.legend(handles=handles, labels=labels, fontsize = 18)

# Einstellungen für Titel, Beschriftungen, Schriftgröße ...
plt.xlabel('Todesfälle', fontsize = 20)
plt.ylabel('Gesamtinfektionen', fontsize = 20)
plt.title('Abhängigkeit zwischen Gesamtinfektionen und Todesfällen der␣
 ↪einzelnen Länder (ohne USA)', fontsize = 24, pad = 30)
plt.xticks(fontsize = 18)
plt.yticks(fontsize = 18)
```

[97]:
```
(array([     0.,  50000., 100000., 150000., 200000., 250000., 300000.]),
 <a list of 7 Text major ticklabel objects>)
```



ANMERKUNGEN - durch Veränderung der Werte für plt.xlim und plt.ylim kann der Plot beliebig angepasst und ein bestimmter Bereich hervorgehoben werden - ggf. müssen die auskommentierten Zeilen für die Legendeneinträge wieder eingeblendet werden - ggf. müssen die Datenbeschriftungen der Länder auskommentiert werden, wenn ein Bereich mit niedrigen Gesamtinfektionen und Todesfällen näher betrachtet werden soll - überlappen die Datenpunkte nicht mehr so stark, kann es ebenfalls wünschenswert sein, sämtliche Datenbeschriftungen anzuzeigen

3.6 Top 5 Mortalitätsrate

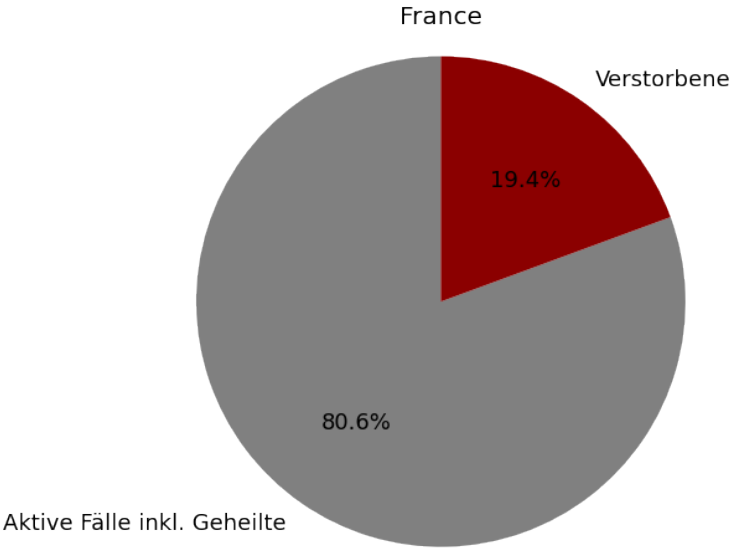
Fragestellung: In welchen Ländern sind vergleichsweise die meisten Menschen verstorben (verglichen mit der Infiziertenzahl)?

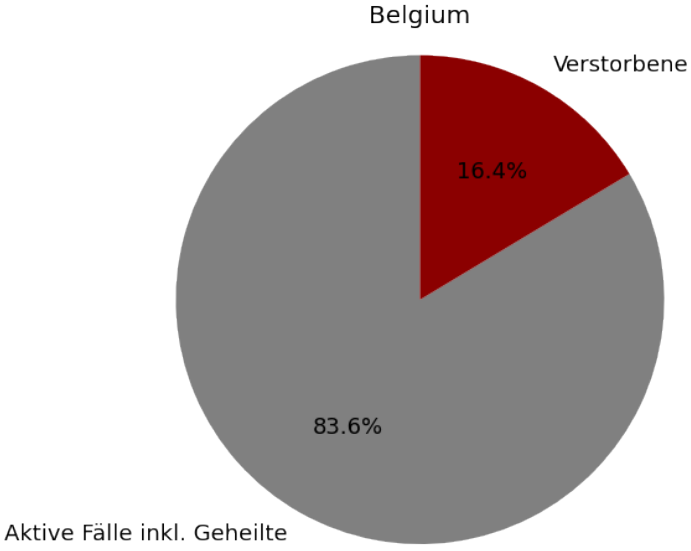
- Schwellenwert: mindestens 50 registrierte Todesfälle im Land
- um Länder auszuschließen, deren Fall- und Verstorbenenzahlen noch nicht repräsentativ sind (z.B. Nicaragua, cases: 25, deaths: 8, Mortalitätsrate: 32.00%)

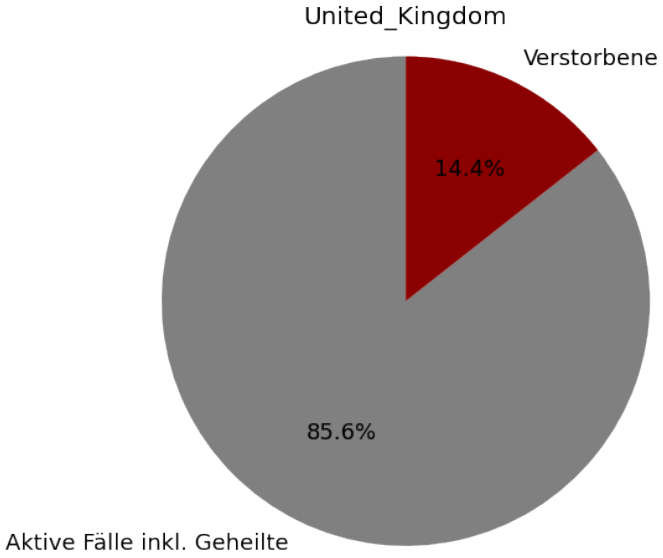
[98]:
```
# Erstellung eines separaten Dataframes, das aus dem Dataframe world die 5␣
 ↪größten Einträge für die Spalte "% Verstorbene" anzeigt
# unter vorheriger Anwendung eines Filters, der Länder mit weniger als 50␣
 ↪registrierten Todesfällen ausschließt
a = world[world.deaths >= 50]['% Verstorbene'].nlargest(5)

# Anzeigen von a
a
```

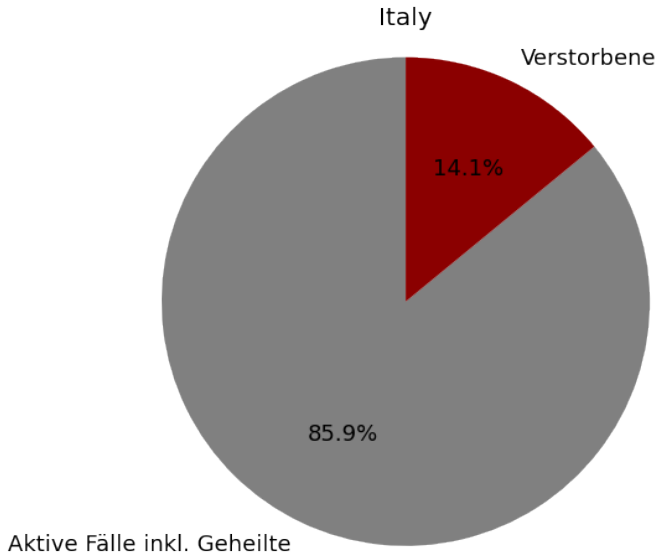
[98]:
```
countriesAndTerritories
France            19.401370
Belgium           16.399573
United_Kingdom    14.417266
Italy             14.060315
Hungary           12.935323
Name: % Verstorbene, dtype: float64
```

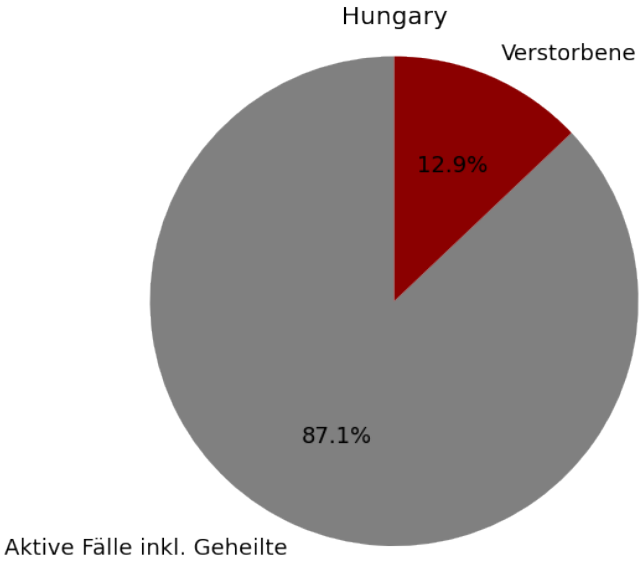
[99]:
```
import matplotlib as mpl
mpl.rcParams['font.size'] = 18

# Zeichne für jedes Land ein Kreisdiagramm mit dem Anteil der Verstorbenen␣
 ↪innerhalb der Infizierten
for l in list(a.index):
  akt = world.loc[l]['Aktive Fälle inkl. Geheilte']
  tot = world.loc[l]['deaths']
  labels = 'Aktive Fälle inkl. Geheilte', 'Verstorbene'
  sizes = [akt, tot]
  colors = ['grey', 'darkred']

  plt.pie(sizes,
          labels=labels,
          colors=colors,
          autopct='%1.1f%%',
          shadow=False,
          startangle=90)
  plt.axis('equal')
  plt.title(l, fontsize = 20)
  plt.show()
```


France
Verstorbene
19.4%
80.6%
Aktive Fälle inkl. Geheilte


Belgium
Verstorbene
16.4%
83.6%
Aktive Fälle inkl. Geheilte


United_Kingdom
Verstorbene
14.4%
85.6%
Aktive Fälle inkl. Geheilte


Italy
Verstorbene
14.1%
85.9%
Aktive Fälle inkl. Geheilte


Hungary
Verstorbene
12.9%
87.1%
Aktive Fälle inkl. Geheilte