Covid-19 und Feinstaubdatenanalyse

In der folgenden Datenanalyse wurden die Covid19-Daten des RKI und die Feinstaubdaten des Bundesministeriums zur Analyse verwendet. Es wurde untersucht ob die Daten miteinander korrelieren. Für die Datengewinnung und Datenbearbeitung wurde Pandas, numpy und scipy verwendet. Für das ploten der Daten, wurde die Matplotlib verwendet.

- Gruppenmitglieder:
  - Anna Kuhn
  - Benjamin Hamm
  - Michael Schulze
  - Jan Klotter

In [1]:
```
#Abhängigkeiten importieren

import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime
from datetime import timedelta
from datetime import date
import numpy as np
from scipy.stats import pearsonr
import collections
from scipy.ndimage.filters import uniform_filter1d
import matplotlib.dates as mdates
```

In [2]:
```
# Benutze Versionen anzeigen
print("Pandas-Version" + pd.__version__)
print("NumPy-Version" + np.__version__)
!python -V
```

```
Pandas-Version1.1.3
NumPy-Version1.19.2
Python 3.8.5
```

1. Datenaufbereitung

Datenquellen:

- RKI Corona Daten Stand 11.01.2021 https://npgeo-corona-npgeo-de.hub.arcgis.com/datasets/dd4580c810204019a7b8eb3e0b329dd6_0/data
- Umweltbundesamt Luftdaten der Station Mannheim Friedrichsring (DEBW098) https://www.umweltbundesamt.de/daten/luft/luftdaten/luftqualitaet/eJzrWJSSuMrIwMhQ18BQ19BoUUnmIkODRXmpCxYVlyw2srRcnOJWBJc3MF2cEpKPrDy3imNRbnLT4pzEktMOHtdiz0_Nmrk4Jy_9tIOS9gMGI

Als ersten Schritt müssen die Daten aufbereitet werden. Die Corona-Daten von Mannheim liegen in mehreren Einträgen vor und müssen deshalb auf einen Tag aufsummiert werden. Fehlende Einträge werden auf 0 gesetzt.

In [3]:
```
#Daten aus Datei auslesen mithilfe von Pandas

covid_data_germany = pd.read_csv('RKI_COVID19.csv')
feinstaub_data_mannheim = pd.read_csv('Luftqualitaet.csv',sep=';')
```

Die folgende Zelle filtriert die Corona-Daten aus Mannheim heraus und lediglich alle Daten aus dem Jahr 2020.

In [4]:
```
#Corona Daten nach Landkreis Mannheim filtern
covid_data_mannheim = covid_data_germany[covid_data_germany['Landkreis'] == 'SK Mannheim' ]

#Corona Daten nach Datum sortieren
#sortedByDate_covid_data_mannheim = covid_data_mannheim.sort_values(by='Refdatum')

#2021 Werte aussortiert
covid_data_mannheim = covid_data_mannheim[
        covid_data_mannheim['Refdatum'] <= '2020/12/31 00:00:00' ]
```

Die Feinstaubdaten hatten viele Leerdaten, welche aussortiert werden mussten.

In [5]:
```
#Tage an denen nicht aufgenommen wurde aussortieren
feinstaub_data_mannheim = feinstaub_data_mannheim[feinstaub_data_mannheim['Feinstaub (PM10) stündlich gleitendes Tag

#Umweltbundesamt Eintrag aussortieren
feinstaub_data_mannheim = feinstaub_data_mannheim[:-2]
```

Damit die Anzahl der Covid-19-Fälle mit den Feinstaubdaten verglichen werden kann, muss ein Tagesmittel für die Feinstaubdaten ausgerechnet werden. Die Covid-19-Fälle müssen auch gefiltert werden. Dadurch kann man die Anzahl der Covid-19-Fällen gegenüber den Tagesmittel der Feinstaubdaten korrelieren lassen.

In [6]:
```
#Fallzahlen und Todeszahlen pro Tag bestimmen

neuinfektionen_nach_datum = {}
todesfaelle_nach_datum = {}

for key, value in enumerate(covid_data_mannheim['Refdatum']):
    if value[:10] not in neuinfektionen_nach_datum:
        todesfaelle_nach_datum[value[:10]] = [np.array(covid_data_mannheim['AnzahlTodesfall'])[key]]
        neuinfektionen_nach_datum[value[:10]] = [np.array(covid_data_mannheim['AnzahlFall'])[key]]
    else:
        todesfaelle_nach_datum[value[:10]].append(np.array(covid_data_mannheim['AnzahlTodesfall'])[key])
        neuinfektionen_nach_datum[value[:10]].append(np.array(covid_data_mannheim['AnzahlFall'])[key])

        wert = neuinfektionen_nach_datum[value[:10]]
        neuinfektionen_nach_datum[value[:10]] = [np.sum(np.array(wert))]

        wert = todesfaelle_nach_datum[value[:10]]
        todesfaelle_nach_datum[value[:10]] = [np.sum(np.array(wert))]
```

In [7]:
```
#Tagesmittel Feinstaub (PM10) pro Tag bestimmen

feinstaub_data_mannheim_nach_datum = {}

for key, value in enumerate(feinstaub_data_mannheim['Datum']):
    if type(value) is float:
        break;

    new_date = str(datetime.strptime(value.strip("'")[:10],"%d.%m.%Y"))[:10]

    if new_date not in feinstaub_data_mannheim_nach_datum:

        feinstaub_data_mannheim_nach_datum[new_date] = [np.array(feinstaub_data_mannheim['Feinstaub (PM10) stündlich
    else:

        feinstaub_data_mannheim_nach_datum[new_date].append(np.array(feinstaub_data_mannheim['Feinstaub (PM10) stünd
```

Tage an denen keine Feinstaubdaten aufgenommen wurden, müssen 0 gesetzt werden. Sonst werden NaN-Werte korreliert.

In [8]:
```
#Tage ohne Neuinfektionen oder Toden nachtragen

for datum in feinstaub_data_mannheim_nach_datum.keys():

    if datum.replace("-", "/") not in neuinfektionen_nach_datum.keys():

        neuinfektionen_nach_datum[datum.replace("-", "/")] = [0]
        todesfaelle_nach_datum[datum.replace("-", "/")] = [0]

    feinstaub_data_mannheim_nach_datum[datum] = np.mean(np.array(feinstaub_data_mannheim_nach_datum[datum]))
```

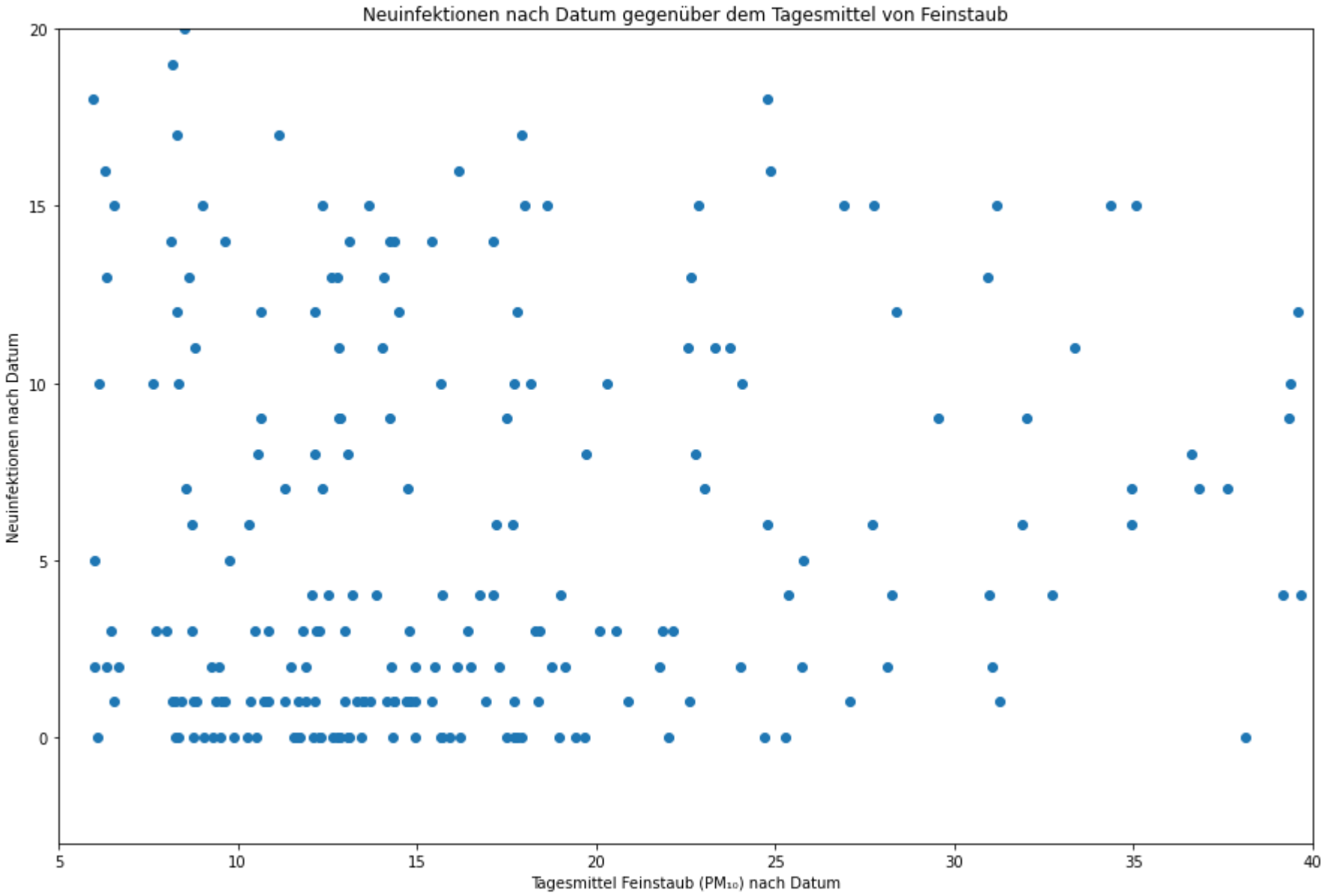
Die Daten müssen nach passenden Datum sortiert werden, da sonst falsche Tagesdaten miteinander korreliert werden.

In [9]:
```
#Datum in richtiger Reihenfolge
sorted_neuinfektionen_nach_datum = collections.OrderedDict(sorted(neuinfektionen_nach_datum.items()))
sorted_todesfaelle_nach_datum = collections.OrderedDict(sorted(todesfaelle_nach_datum.items()))
sorted_feinstaub_data_mannheim_nach_datum = collections.OrderedDict(sorted(feinstaub_data_mannheim_nach_datum.items(

anzahl_faelle_list = list(sorted_neuinfektionen_nach_datum.values())
anzahl_tode_list = list(sorted_todesfaelle_nach_datum.values())
feinstaub_list = list(sorted_feinstaub_data_mannheim_nach_datum.values())

for key, value in enumerate(anzahl_faelle_list):
    anzahl_faelle_list[key] = anzahl_faelle_list[key][0]
    anzahl_tode_list[key] = anzahl_tode_list[key][0]
```

2.Explorative Datenanalyse

Im folgenden Abschnitt werden die Daten geplotet.

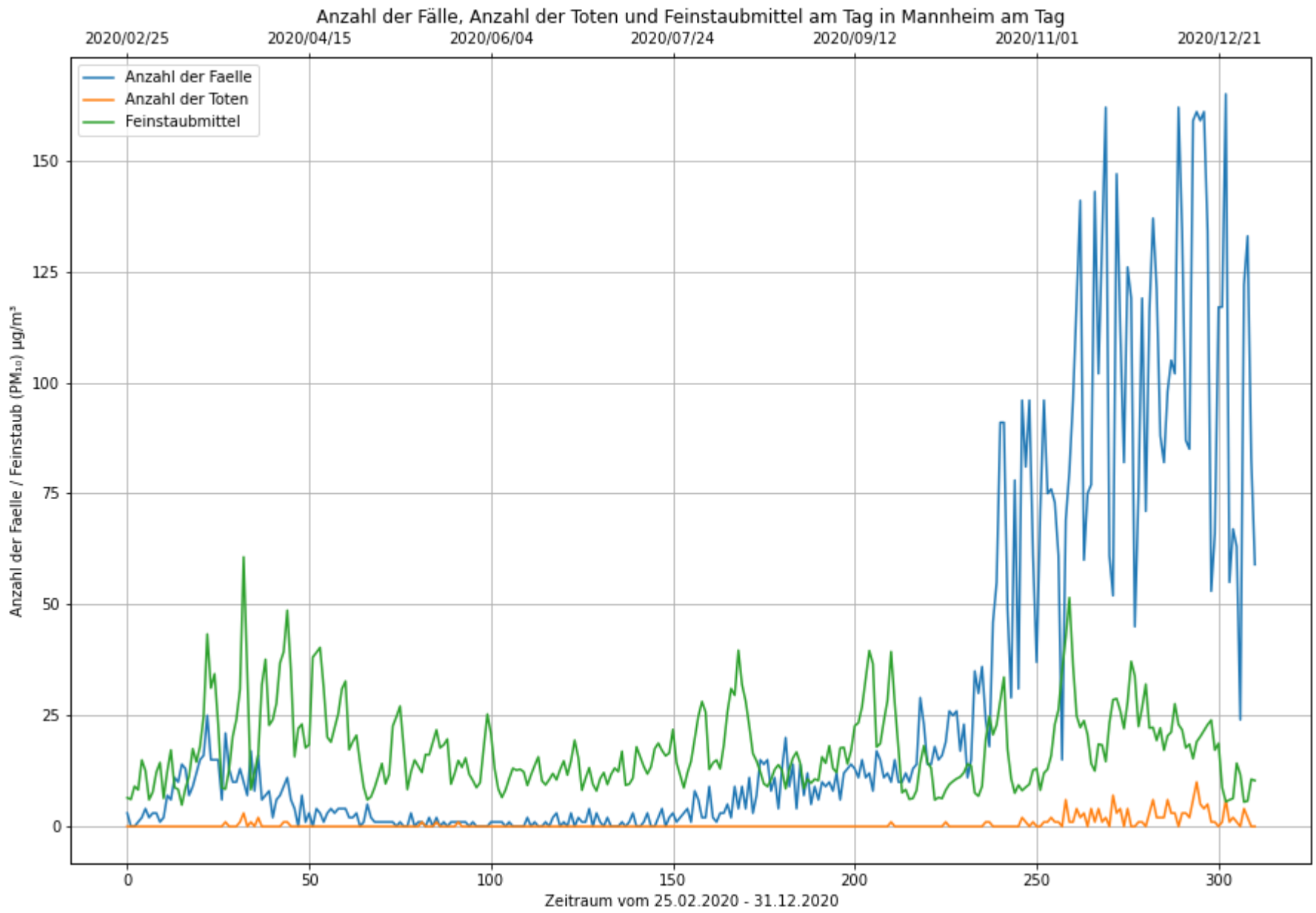
In [10]:
```
# Scatter Plot
plt.scatter(feinstaub_list, anzahl_faelle_list)
fig = plt.gcf()
fig.set_size_inches(15,10)
plt.xlim([5,40])
plt.ylim([-3,20])
plt.title("Neuinfektionen nach Datum gegenüber dem Tagesmittel von Feinstaub")
plt.ylabel("Neuinfektionen nach Datum")
plt.xlabel("Tagesmittel Feinstaub (PM10) nach Datum")
plt.show()

corr,_= pearsonr(feinstaub_list, anzahl_faelle_list)
print(corr)
```




```
0.12346439885572025
```

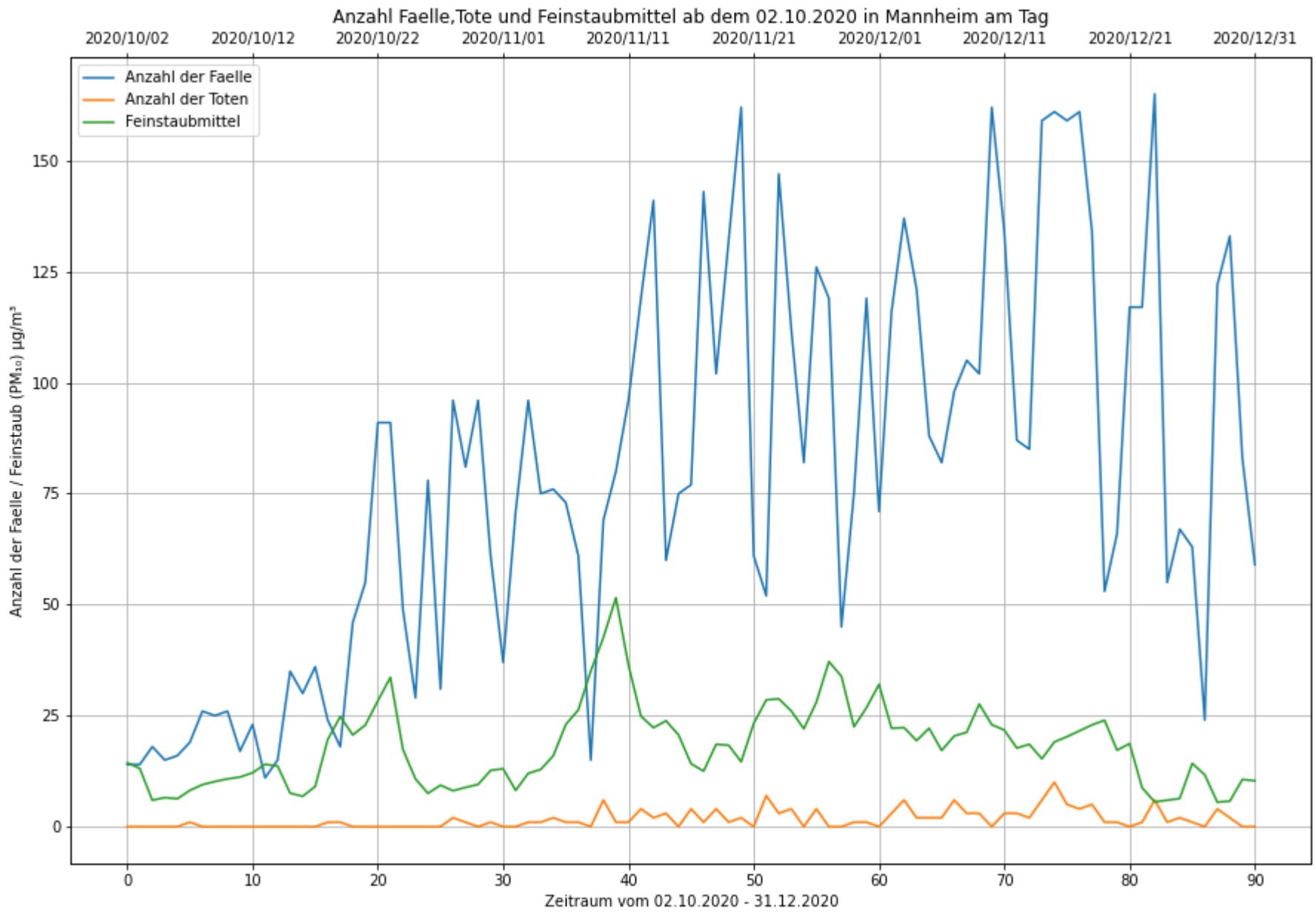
Aus dem Scatterplot kann man noch keinen linearen Zusammenhang zwischen den Daten erkennen. Es sieht sehr gestreut aus.

In [11]:
```
# Mannheimer Daten Corona und Feinstaub gesamt 2020

# Daten für x-Ticks erzeugen
datum_zuordnung = list(sorted_neuinfektionen_nach_datum)
xind = np.array(range(len(anzahl_faelle_list)))
tickabstand = 50

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(anzahl_faelle_list,label="Anzahl der Faelle")
ax1.plot(anzahl_tode_list,label="Anzahl der Toten")
ax1.plot(feinstaub_list,label="Feinstaubmittel")
ax1.set(title = "Anzahl der Fälle, Anzahl der Toten und Feinstaubmittel am Tag in Mannheim am Tag",
        xlabel = "Zeitraum vom 25.02.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle / Feinstaub (PM₁₀) µg/m³")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend()
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



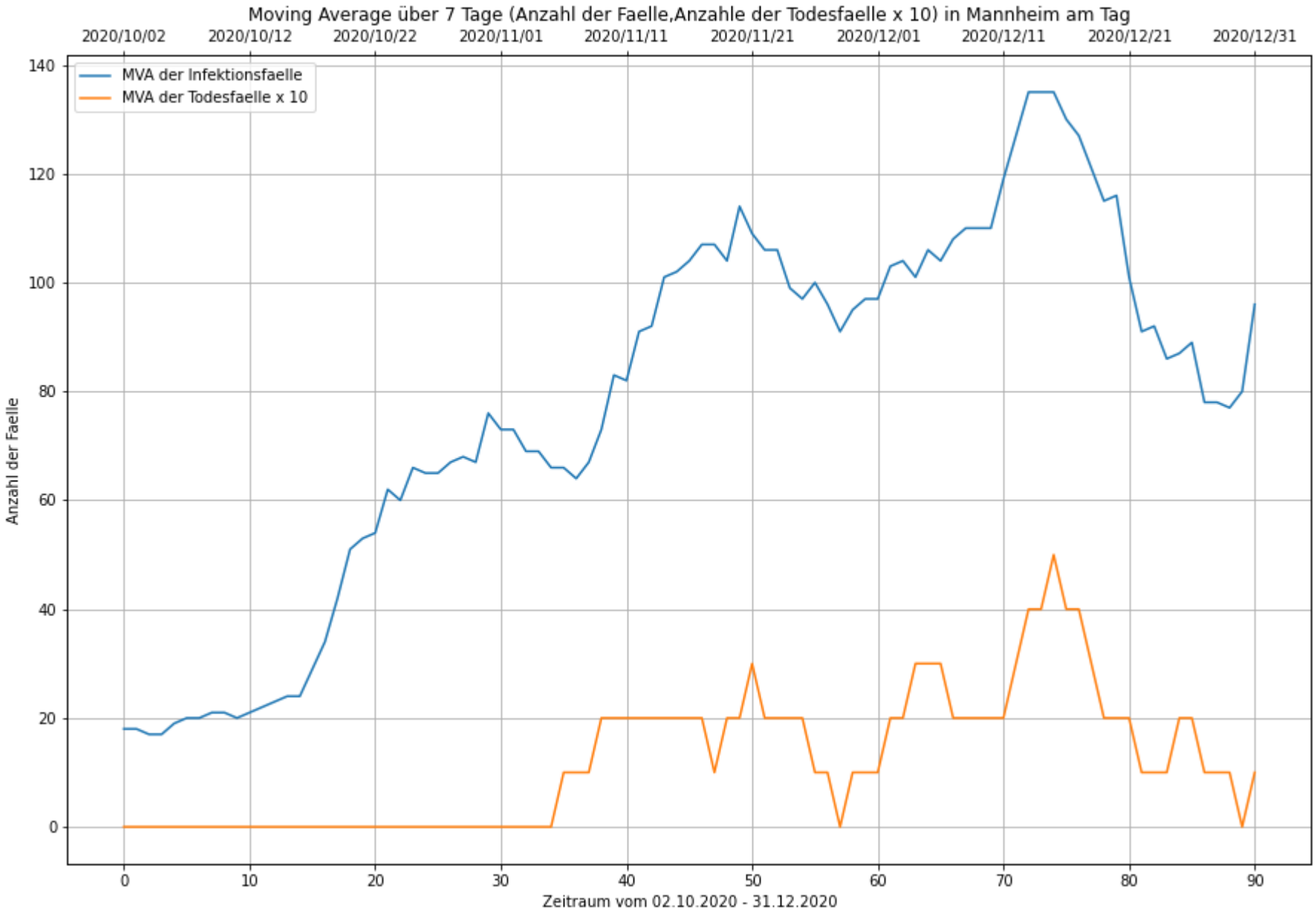
Anhand der Daten erkennt man schon teilweise eine gewisse Korrelation. Sobald die Feinstaub-Daten sich erhöhen, erhöhen sich auch die Anzahl der Faelle.

In [12]:
```
# Mannheimer Daten Corona und Feinstaub für zweite Welle 2020

# Daten für x-Ticks erzeugen
offset = 220
tickabstand = 10
datum_zuordnung = list(sorted_neuinfektionen_nach_datum)[offset:]
xind = np.array(range(len(anzahl_faelle_list)))[:len(anzahl_faelle_list)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(anzahl_faelle_list[offset:],label="Anzahl der Faelle")
ax1.plot(anzahl_tode_list[offset:],label="Anzahl der Toten")
ax1.plot(feinstaub_list[offset:],label="Feinstaubmittel")
ax1.set(title = "Anzahl Faelle,Tote und Feinstaubmittel ab dem 02.10.2020 in Mannheim am Tag",
        xlabel = "Zeitraum vom 02.10.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle / Feinstaub (PM10) µg/m³")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



Im folgenden Plot wurde der Moving-Average der Anzahl der Todesfälle und die Anzahl der Corona-Fälle in Mannheim von 7 Tagen geplottet. (Da das RKI am Wochenende nur wenig Daten übermittel bekommt, haben wir auf eine Woche gerundet, damit die Wochenendschwankungen herausgefiltert werden.)

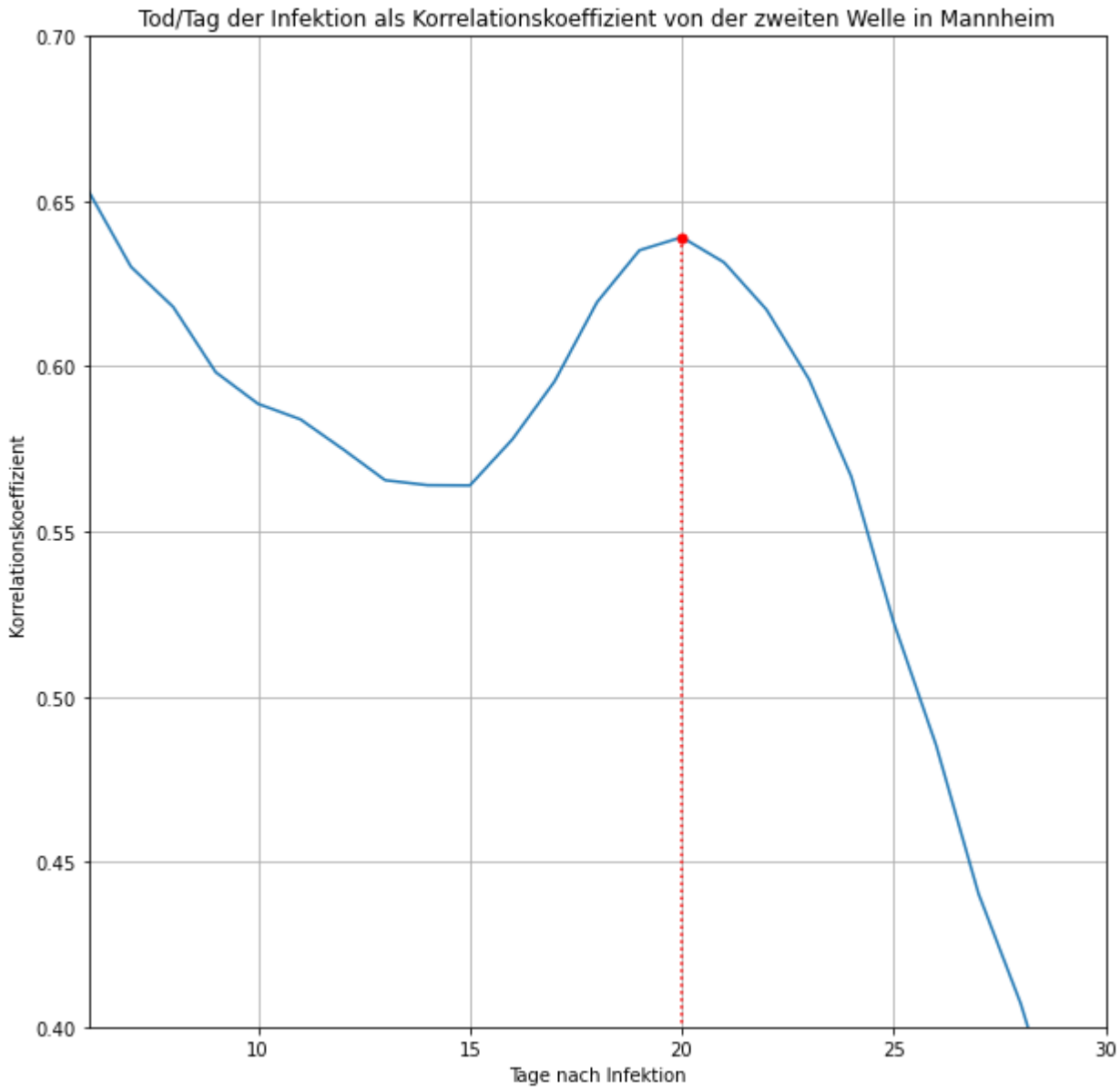
In [13]:
```
# Mannheimer Daten Corona für zweite Welle 2020

# Moving Average
N = 7
anzahl_faelle_list_rm = uniform_filter1d(anzahl_faelle_list, size=N)
anzahl_tode_list_rm = uniform_filter1d(anzahl_tode_list, size=N)


# Daten für x-Ticks erzeugen
offset = 220
tickabstand = 10
datum_zuordnung = list(sorted_neuinfektionen_nach_datum)[offset:]
xind = np.array(range(len(anzahl_faelle_list)))[:len(anzahl_faelle_list)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(anzahl_faelle_list_rm[offset:],label="MVA der Infektionsfaelle")
ax1.plot(anzahl_tode_list_rm[offset:] * 10,label="MVA der Todesfaelle x 10")
ax1.set(title = "Moving Average über 7 Tage (Anzahl der Faelle,Anzahle der Todesfaelle x 10) in Mannheim am Tag",
        xlabel = "Zeitraum vom 02.10.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



Anhand der Abbildung erkennt man sehr gut, dass nach einem Peak der Infektionen, ein Paar Tage später die Todeszahlen ansteigen. Da wie gesagt, ein Anstieg der Infektionszahlen erst zeitversetzt einen Anstieg der Todeszahlen verursacht, wurden die Korrelation für verschiedene Tagesdifferenzen (0 bis 50) zwischen Infektionszeitpunkt und Todeszeitpunkt ermittelt.

In [14]:
```
# Mannheimer Daten Corona für zweite Welle 2020 - Korrelation Infizierte mit Toten?

corrs = []
#anzahl_faelle_float = list(map(float, anzahl_faelle_list_rm ))
#anzahl_tode_float = list(map(float, anzahl_tode_list_rm ))

anzahl_faelle_float = [float(elem) for elem in anzahl_faelle_list_rm]
anzahl_tode_float = [float(elem) for elem in anzahl_tode_list_rm]

# Pearson Koeffizient für Verschiebung der Datenreihen um i-Tage
offset = 220

for i in range(1,50):
    corr,p= pearsonr(anzahl_faelle_list_rm[offset:-i], anzahl_tode_list_rm[offset+i:])
    corrs.append(corr)

# Maximum der Pearson Koeffizienten für Verschiebung
corr_max = np.max(corrs[10:])
corr_arg_max = np.argmax(corrs[10:])+10

print("Maximum von "+ str(round(corr_max,2)) + " nach " + str(corr_arg_max) + " Tagen")


# Daten für x-Ticks erzeugen
offset = 220
tickabstand = 5
datum_zuordnung = list(sorted_neuinfektionen_nach_datum)[offset:]
xind = np.array(range(len(anzahl_faelle_list)))[:len(anzahl_faelle_list)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(corrs,label="Korrelationskoeffizient der Infektionsfaellen zu den Todesfaellen")
ax1.set(title = "Tod/Tag der Infektion als Korrelationskoeffizient von der zweiten Welle in Mannheim",
        xlabel = "Tage nach Infektion",
        ylabel = "Korrelationskoeffizient")
ax1.set_xticks(xind[::tickabstand])
ax1.set_xlim([6,30])
ax1.set_ylim([0.4,0.7])
ax1.grid()
#ax1.legend(loc="upper left")
# Maximum markieren
ax1.plot(corr_arg_max, corr_max, marker='o', markersize=5, color="red")
ax1.vlines(corr_arg_max,0,corr_max,color='red', linestyle='dotted')

fig.set_size_inches(10,10)
```

```
Maximum von 0.64 nach 20 Tagen
```



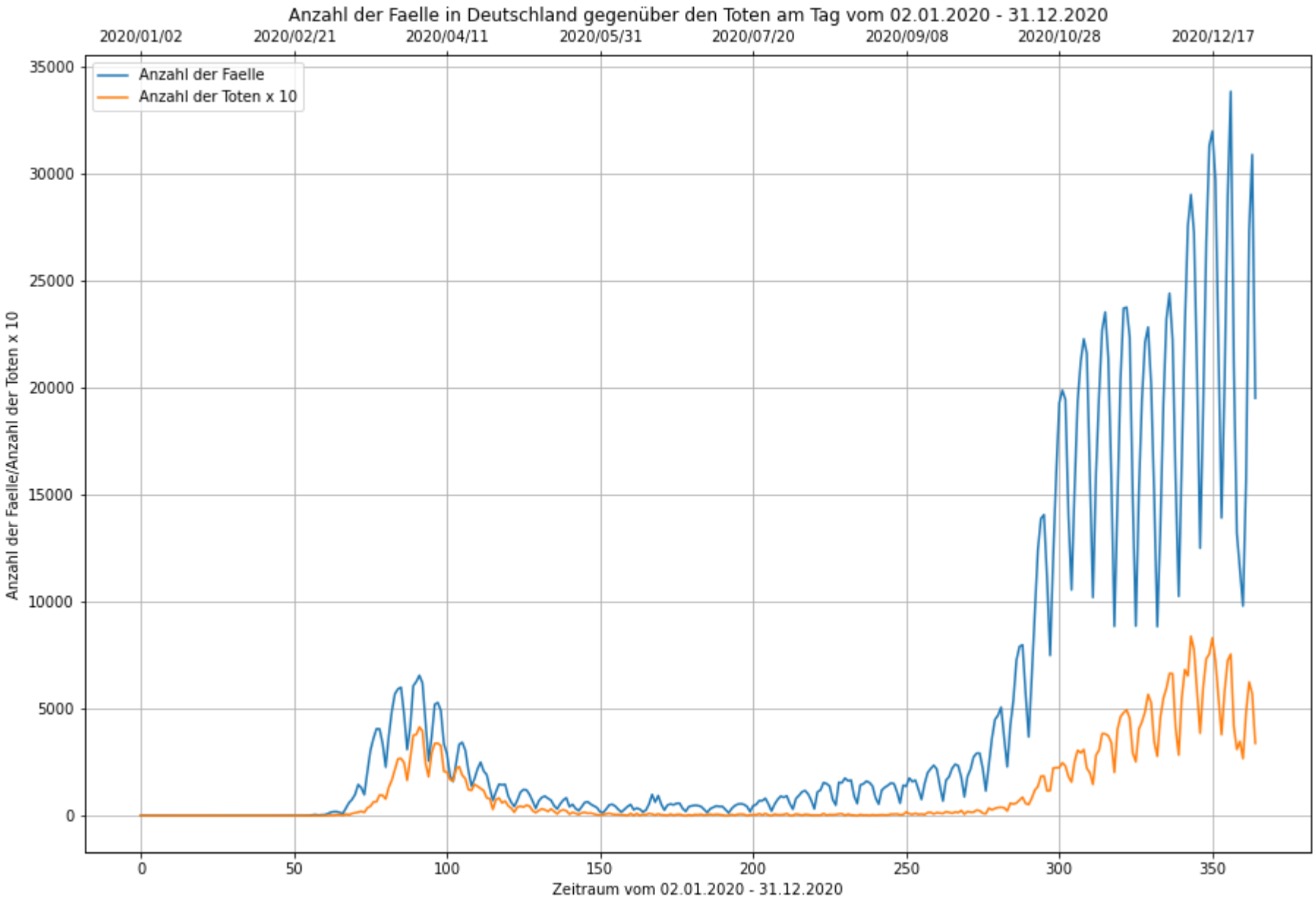
0.64 als Korrelationskoeffizient nach 20 Tagen, in klinischen Studien ist das eine Korrelation. (Korrelation bei 0-5 Tagen Differenz auch sehr hoch aus unbekanntem Grund, vielleicht durch niedrige Todeszahlen in Mannheim)

Vergleich mit Deutschland

Zum Vergleich werden die Daten von gesamt Deutschland dargestellt. Zur besseren Visualisierung der Korrelation wird die Anzahl der Toten um den Faktor 10 erhöht und mit den Infektionszahlen in einem Plot dargestellt.

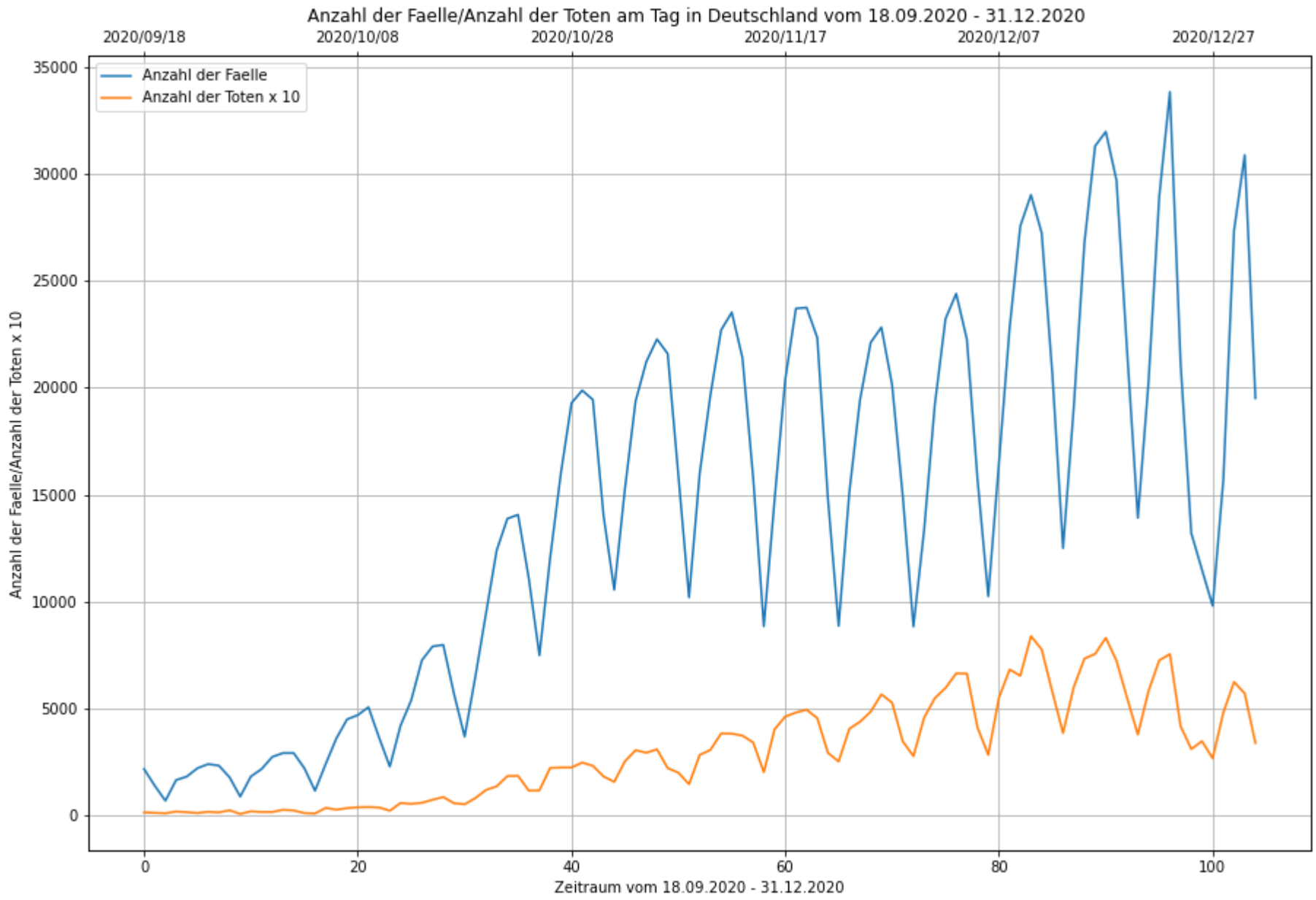
In [15]:
```
# Daten einlesen

covid_data_germany = pd.read_csv('covid-19__2.csv',sep=';')
covid_data_germany = covid_data_germany.dropna()

faelle_nach_datum = list(covid_data_germany['Faelle'])
todesfaelle_nach_datum = list(covid_data_germany['Todesfaelle']*10)

# Datum für Anzeige erstellen
sdate=date(2020,1,2)
edate=date(2020,12,31)
datumsliste = pd.date_range(sdate,edate,freq='d')
datumshort = list(datumsliste.strftime('%Y/%m/%d'))
```

In [16]:
```
# Deutschland Daten Corona für Jahr 2020

# Daten für x-Ticks erzeugen
tickabstand = 50
datum_zuordnung = datumshort
xind = np.array(range(len(faelle_nach_datum)))

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(faelle_nach_datum,label="Anzahl der Faelle")
ax1.plot(todesfaelle_nach_datum,label="Anzahl der Toten x 10")
ax1.set(title = "Anzahl der Faelle in Deutschland gegenüber den Toten am Tag vom 02.01.2020 - 31.12.2020",
        xlabel = "Zeitraum vom 02.01.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle/Anzahl der Toten x 10")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



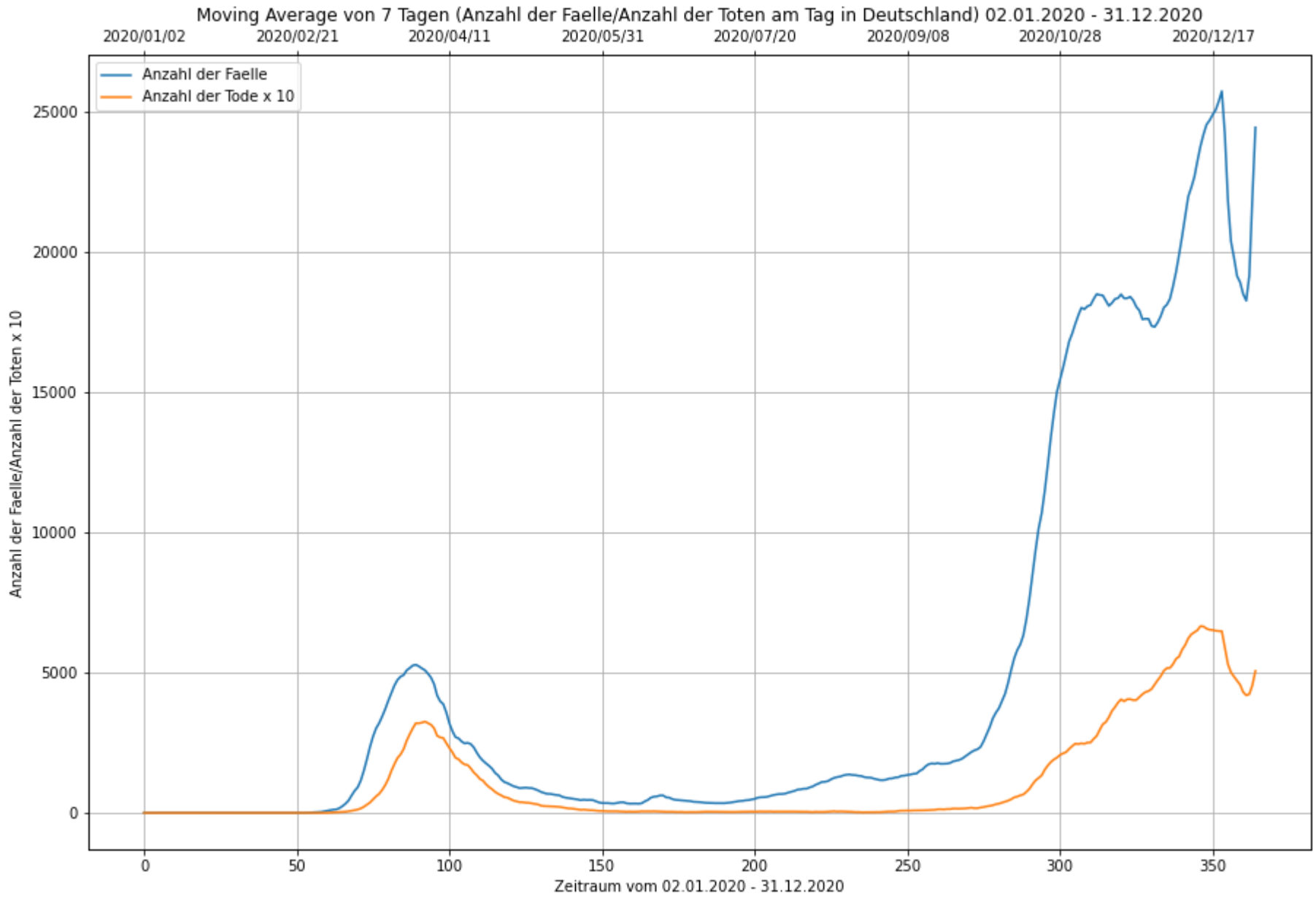
Des Weiteren wird eine Grafik geplottet, welche die Anzahl der Infektionen mit der Anzahl der Toten zwischen dem 18.09.2020 und 31.12.2020 in Deutschland dargestellt, da in diesem Zeitraum die Daten maximale Werte aufweisen und Zusammenhänge besser ersichtlich sind.

In [17]:
```
# Deutschland Daten Corona für zweite Welle 2020

# Daten für x-Ticks erzeugen
offset = 260
tickabstand = 20
datum_zuordnung = datumshort[offset:]
xind = np.array(range(len(faelle_nach_datum)))[:len(faelle_nach_datum)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(faelle_nach_datum[offset:],label="Anzahl der Faelle")
ax1.plot(todesfaelle_nach_datum[offset:],label="Anzahl der Toten x 10")
ax1.set(title = "Anzahl der Faelle/Anzahl der Toten am Tag in Deutschland vom 18.09.2020 - 31.12.2020 ",
        xlabel = "Zeitraum vom 18.09.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle/Anzahl der Toten x 10")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```

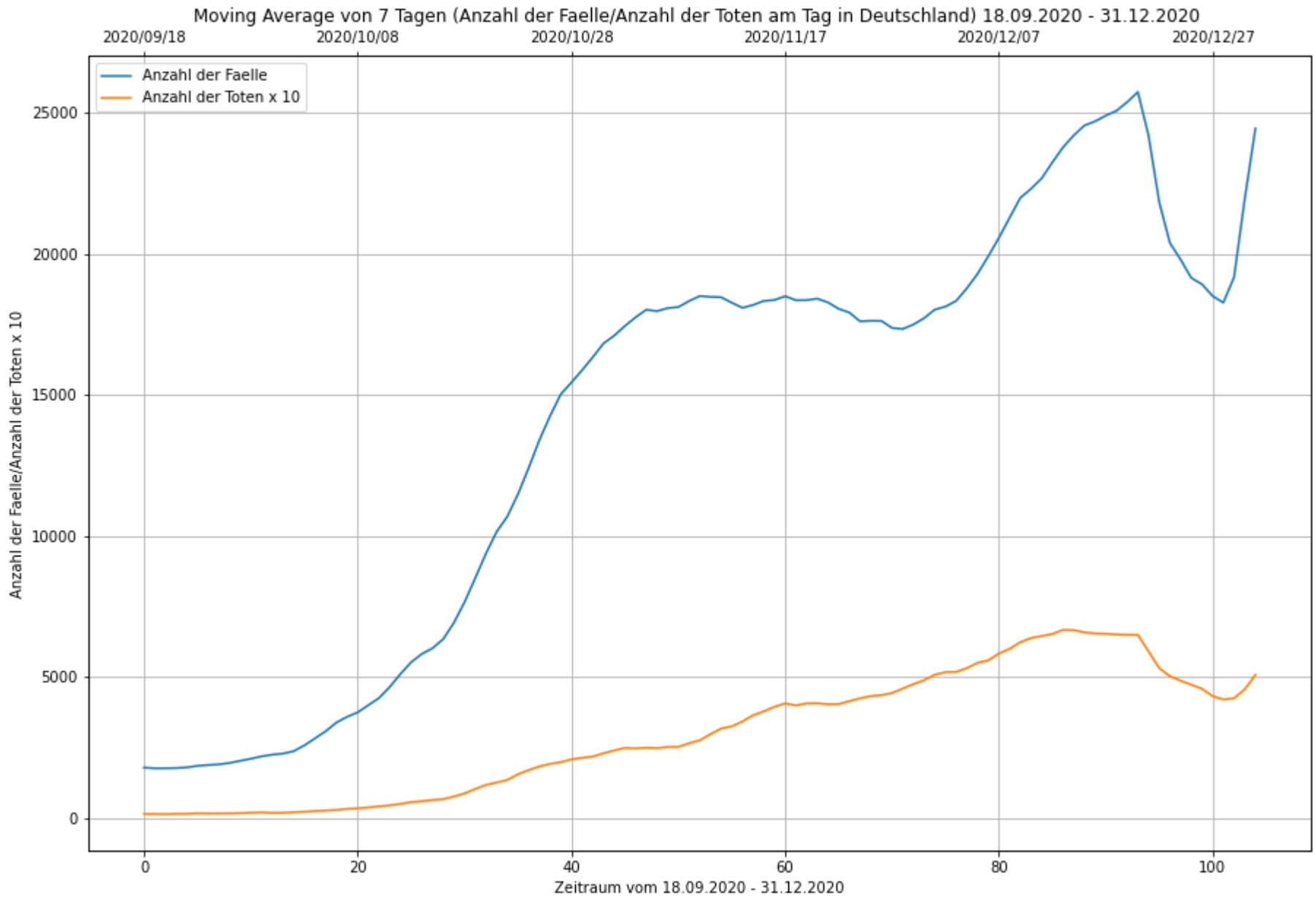


Aus bereits oben genannten Gründen wird wieder der 7 Tage Moving Average der Daten bestimmt.

In [18]:
```
# Deutschland Daten Corona für Jahr 2020

# Moving Average
N = 7
anzahl_faelle_list_rm = uniform_filter1d(faelle_nach_datum, size=N)
anzahl_tode_list_rm = uniform_filter1d(todesfaelle_nach_datum, size=N)


# Daten für x-Ticks erzeugen
tickabstand = 50
datum_zuordnung = datumshort
xind = np.array(range(len(faelle_nach_datum)))

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(anzahl_faelle_list_rm,label="Anzahl der Faelle")
ax1.plot(anzahl_tode_list_rm,label="Anzahl der Tode x 10")
ax1.set(title = "Moving Average von 7 Tagen (Anzahl der Faelle/Anzahl der Toten am Tag in Deutschland) 02.01.2020 -
        xlabel = "Zeitraum vom 02.01.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle/Anzahl der Toten x 10")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



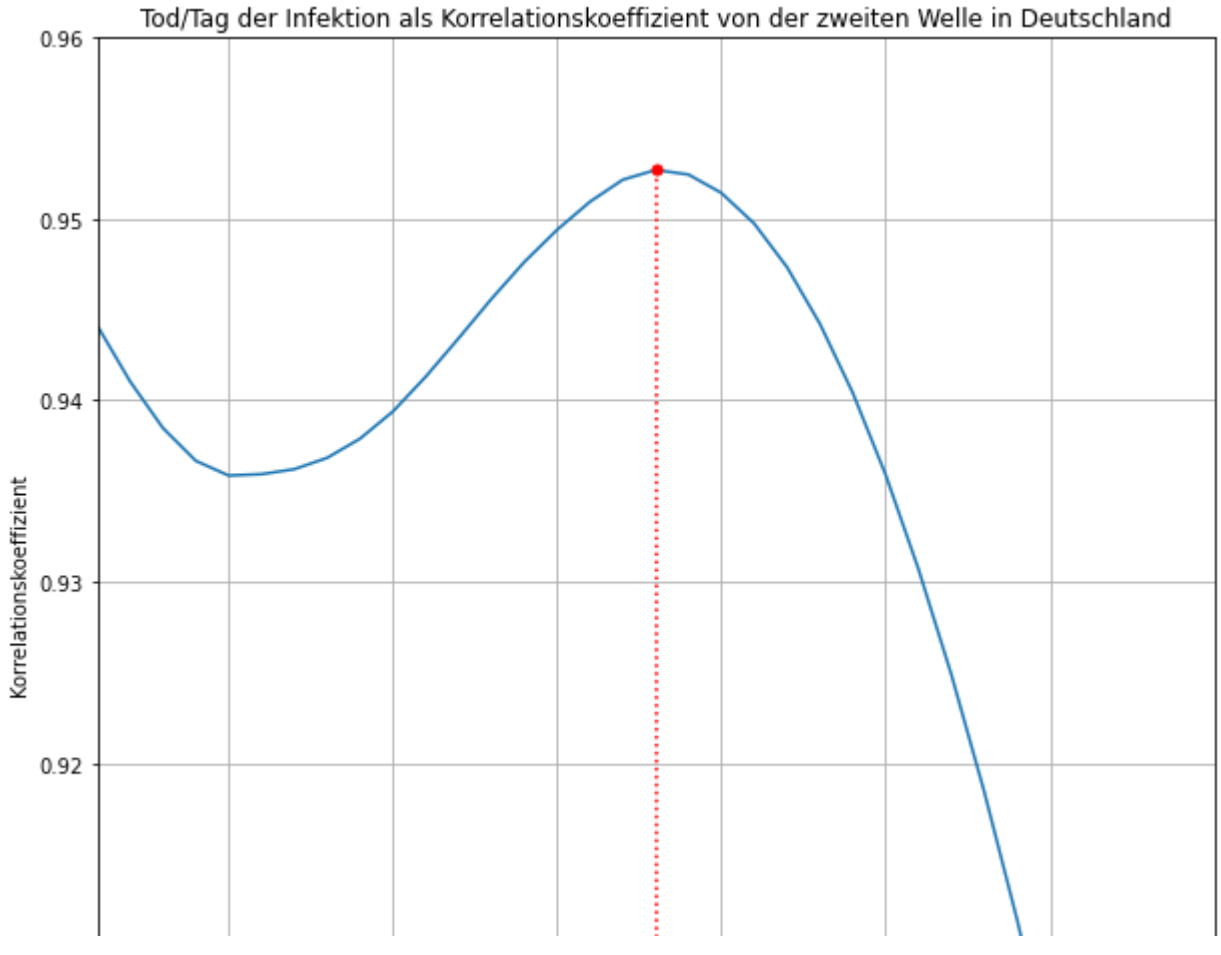
In [19]:
```
# Deutschland Daten Corona für zweite Welle 2020

# Daten für x-Ticks erzeugen
offset = 260
tickabstand = 20
datum_zuordnung = datumshort[offset:]
xind = np.array(range(len(faelle_nach_datum)))[:len(faelle_nach_datum)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(anzahl_faelle_list_rm[offset:],label="Anzahl der Faelle")
ax1.plot(anzahl_tode_list_rm[offset:],label="Anzahl der Toten x 10")
ax1.set(title = "Moving Average von 7 Tagen (Anzahl der Faelle/Anzahl der Toten am Tag in Deutschland) 18.09.2020 -
        xlabel = "Zeitraum vom 18.09.2020 - 31.12.2020",
        ylabel = "Anzahl der Faelle/Anzahl der Toten x 10")
ax1.set_xticks(xind[::tickabstand])
ax1.grid()
ax1.legend(loc="upper left")
# zweite x-Achse mit gleichen Ticks wie erste X-Achse, Beschriftung soll Datum sein
ax2 = ax1.twiny()
ax2.set_xticks(ax1.get_xticks())
ax2.set_xbound(ax1.get_xbound())
ax2.set_xticklabels(datum_zuordnung[::tickabstand])

fig.set_size_inches(15,10)
```



Außerdem wird die Korrelation bestimmt.

In [20]:
```
# Deutschland Daten Corona für zweite Welle 2020 - Korrelation Infizierte mit Toten?

corrs = []

# Pearson Koeffizient für Verschiebung der Datenreihen um i-Tage
offset = 220

for i in range(1,50):
    corr,p= pearsonr(anzahl_faelle_list_rm[offset:-i], anzahl_tode_list_rm[offset+i:])
    corrs.append(corr)

# Maximum der Pearson Koeffizienten für Verschiebung
corr_max = np.max(corrs[5:])
corr_arg_max = np.argmax(corrs[5:])+5

print("Maximum von "+ str(round(corr_max,2)) + " nach " + str(corr_arg_max) + " Tagen")


# Daten für x-Ticks erzeugen
offset = 220
tickabstand = 5
xind = np.array(range(len(anzahl_faelle_list)))[:len(anzahl_faelle_list)-offset]

# Plot erstellen
fig, ax1 = plt.subplots()

ax1.plot(corrs,label="Korrelationskoeffizient der Infektionsfaellen zu den Todesfaellen")
ax1.set(title = "Tod/Tag der Infektion als Korrelationskoeffizient von der zweiten Welle in Deutschland",
        xlabel = "Tage nach Infektion",
        ylabel = "Korrelationskoeffizient")
ax1.set_xticks(xind[::tickabstand])
ax1.set_xlim([6,40])
ax1.set_ylim([0.9,0.96])
ax1.grid()
#ax1.legend(loc="upper left")
# Maximum markieren
ax1.plot(corr_arg_max, corr_max, marker='o', markersize=5, color="red")
ax1.vlines(corr_arg_max,0,corr_max,color='red', linestyle='dotted')

fig.set_size_inches(10,10)
```

```
Maximum von 0.95 nach 23 Tagen
```



Die Korrelation hat bei 23 Tagen ihr Maximum mit 0.95.

Zusammenfassung:

- Es ist keine Korrelation zwischen Feinstaubdaten und Coronadaten Mannheim. Zu beachten ist allerdings, dass eine Korrelation auch nicht heißen würde, dass durch Erhöhung der Feinstaubwerte eine Erhöhung der Coronainfektionen hervorgerufen wird oder anders herum, sondern wenn, dann einen Zusammenhang mit den Maßnahmen gehabt hätte.
- Die Korrelation zwischen Infektionen und Todeszahlen in Mannheim zeigt eine Korrelation von 0.64 an Tag 20 (bezogen auf das Infektionsdatum) auf
- Die Korrelation zwischen Infektionen und Todeszahlen in Deutschland zeigt eine Korrelation von 0.95 bei ca. 23 Tagen (bezogen auf das Infektionsdatum)

Herausforderungen:

1. Daten einlesen (Datum konvertieren, nicht vorhandene Daten auffüllen, Daten aus mehreren Zellen aufsummieren)
2. Zeitverschiebung von Coronainfektionen und Todeszahlen
3. Korrelation am Anfang der Zeitdifferenz hoch

In [ ]: