Setup and Context

Introduction

Dr Ignaz Semmelweis was a Hungarian physician born in 1818 who worked in the Vienna General Hospital. In the past people thought of illness as caused by "bad air" or evil spirits. But in the 1800s Doctors started looking more at anatomy, doing autopsies and started making arguments based on data. Dr Semmelweis suspected that something was going wrong with the procedures at Vienna General Hospital. Semmelweis wanted to figure out why so many women in maternity wards were dying from childbed fever (i.e., puerperal fever).

Analyse the same data collected from 1841 to 1849.

The Data Source

Dr Semmelweis published his research in 1861. The scanned pages of the full text with the original tables in German, but an excellent English translation can be found here.

Upgrade plotly (only Google Colab Notebook) if necessary

```
# %pip install --upgrade plotly
```

Import Statements

```
import pandas as pd
import numpy as np
import plotly.express as px
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import scipy.stats as stats
```

Notebook Presentation

```
pd.options.display.float_format = '{:,.2f}'.format

# Create locators for ticks on the time axis
years = mdates.YearLocator()
months = mdates.MonthLocator()
years_fmt = mdates.DateFormatter("%Y")

from pandas.plotting import register_matplotlib_converters
register_matplotlib_converters()
```

Read the Data

```
df_yearly = pd.read_csv('data/annual_deaths_by_clinic.csv')
# parse_dates avoids DateTime conversion later
df_monthly = pd.read_csv('data/monthly_deaths.csv',
                         parse_dates=['date'])
```

Preliminary Data Exploration

- What is the shape of df_yearly and df_monthly? How many rows and columns?
- What are the column names?
- Which years are included in the dataset?
- Are there any NaN values or duplicates?
- What were the average number of births that took place per month?
- What were the average number of deaths that took place per month?

```
print(f"shape of df_year: {df_yearly.shape}")
print(f"shape of df_monthly: {df_monthly.shape}")
```

```
shape of df_year: (12, 4)
shape of df_monthly: (98, 3)
```

```
print(f"Column names of df_year: {df_yearly.columns}")
print(f"Column names of df_monthly: {df_monthly.columns}")
```

```
Column names of df_year: Index(['year', 'births', 'deaths', 'clinic'], dtype='object')
Column names of df_monthly: Index(['date', 'births', 'deaths'], dtype='object')
```

```
df_yearly
```

	year	births	deaths	clinic
0	1841	3036	237	clinic 1
1	1842	3287	518	clinic 1
2	1843	3060	274	clinic 1
3	1844	3157	260	clinic 1
4	1845	3492	241	clinic 1
5	1846	4010	459	clinic 1
6	1841	2442	86	clinic 2
7	1842	2659	202	clinic 2
8	1843	2739	164	clinic 2
9	1844	2956	68	clinic 2
10	1845	3241	66	clinic 2
11	1846	3754	105	clinic 2

```
df_monthly.head()
```

	date	births	deaths
0	1841-01-01	254	37
1	1841-02-01	239	18
2	1841-03-01	277	12
3	1841-04-01	255	4
4	1841-05-01	255	2

```
df_monthly.tail()
```

	date	births	deaths
93	1848-11-01	310	9
94	1848-12-01	373	5
95	1849-01-01	403	9
96	1849-02-01	389	12
97	1849-03-01	406	20

Checking for Nan Values and Duplicates

```
# df_monthly.isna().values.any()
df_monthly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 98 entries, 0 to 97
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   date    98 non-null     datetime64[ns]
 1   births  98 non-null     int64
 2   deaths  98 non-null     int64
dtypes: datetime64[ns](1), int64(2)
memory usage: 2.4 KB
```

```
# df_yearly.isna().values.any()
df_yearly.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12 entries, 0 to 11
Data columns (total 4 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   year    12 non-null     int64
 1   births  12 non-null     int64
 2   deaths  12 non-null     int64
 3   clinic  12 non-null     object
dtypes: int64(3), object(1)
memory usage: 512.0+ bytes
```

```
print(f'Any yearly duplicates? {df_yearly.duplicated().values.any()}')
print(f'Any monthly duplicates? {df_monthly.duplicated().values.any()}')
```

```
Any yearly duplicates? False
Any monthly duplicates? False
```

Descriptive Statistics

```
df_monthly.describe()
```

	births	deaths
count	98.00	98.00
mean	267.00	22.47
std	41.77	18.14
min	190.00	0.00
25%	242.50	8.00
50%	264.00	16.50
75%	292.75	36.75
max	406.00	75.00

```
df_yearly.describe()
```

Percentage of Women Dying in Childbirth

How dangerous was childbirth in the 1840s in Vienna?

- Using the annual data, calculating the percentage of women giving birth who died throughout the 1840s at the hospital.

min 1,841.00 2,442.00 66.00

```
prob = df_yearly.deaths.sum() / df_yearly.births.sum() *100
print(f"Chance of dying in the 1940s in Vienna: {prob:.3}%")
```

```
Chance of dying in the 1940s in Vienna: 7.08%
```

Visualising the Total Number of Births and Deaths over Time

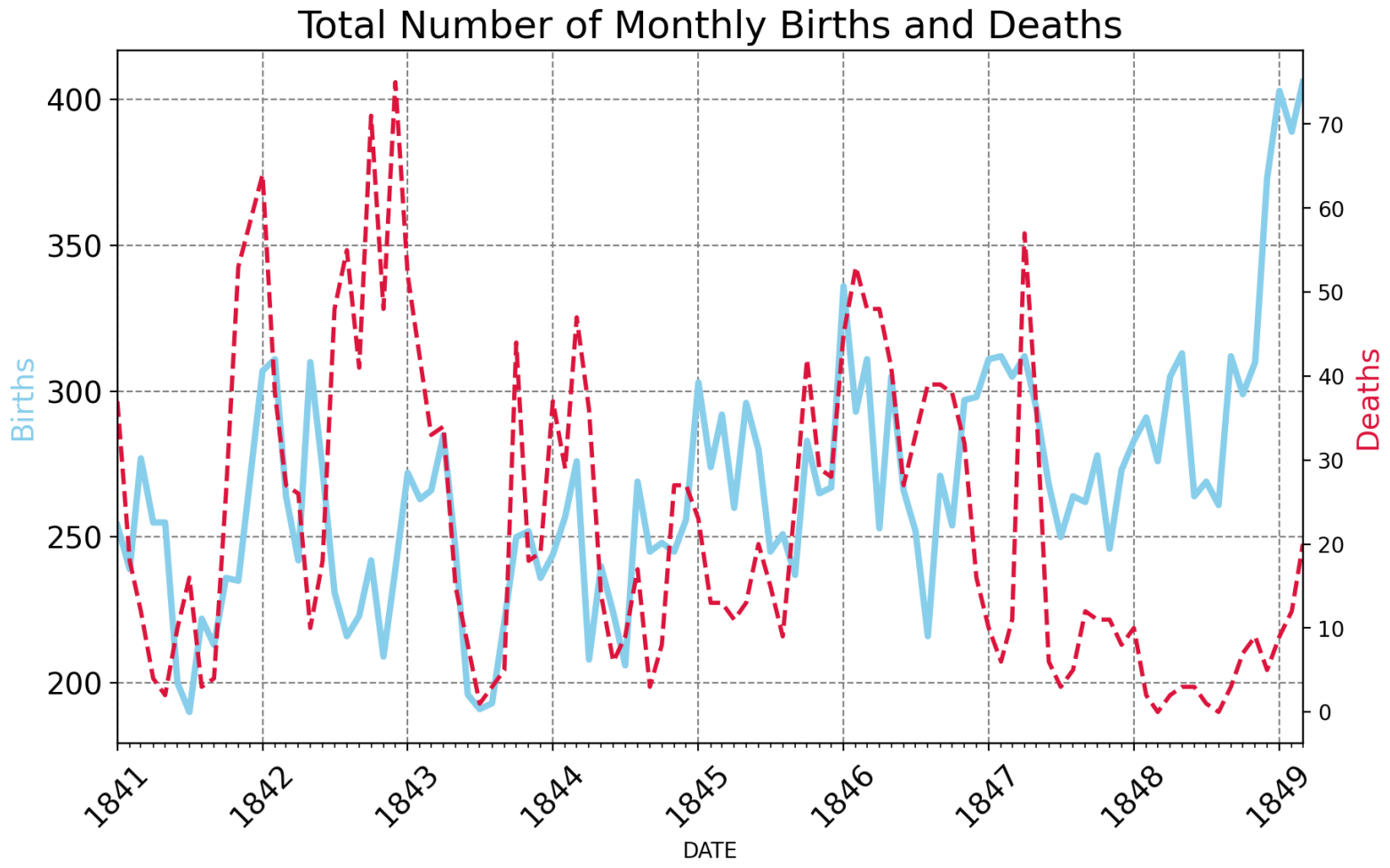
Plot the Monthly Data on Twin Axes

Challenge: Creating a Matplotlib chart with twin y-axes.

- Formating the x-axis using locators for the years and months
- Setting the range on the x-axis so that the chart lines touch the y-axes
- Adding gridlines
- Using `skyblue` and `crimson` for the line colours
- Using a dashed line style for the number of deaths
- Changing the line thickness to 3 and 2 for the births and deaths respectively.
- Do you notice anything in the late 1840s?

```
plt.figure(figsize=(10,6), dpi=200)
# the title of the graph
plt.title("Total Number of Monthly Births and Deaths", fontsize=18)

plt.yticks(fontsize=14)
# Increase the size adn rotate the labels on the x-axis
plt.xticks(fontsize=14, rotation=45)

# getting the axis
ax1 = plt.gca()
# create another axis that share the same x-axis
ax2 = ax1.twinx()

# labels
ax1.set_xlabel("DATE")
ax1.set_ylabel("Births", color='skyblue', fontsize=14)
ax2.set_ylabel("Deaths", color='crimson', fontsize=14)

# setting the min and max values on the axes
ax1.set_xlim([df_monthly.date.min(), df_monthly.date.max()])

# formating where the major and minor ticks should be
ax1.xaxis.set_major_locator(years)
ax1.xaxis.set_major_formatter(years_fmt)
ax1.xaxis.set_minor_locator(months)


ax1.grid(color='grey', linestyle='--')

# ploting different axis
ax1.plot(
    df_monthly.date,
    df_monthly.births,
    color='skyblue',
    linewidth=3,
    )
ax2.plot(
    df_monthly.date,
    df_monthly.deaths,
    color='crimson',
    linewidth=2,
    linestyle='--',
    )
```

```
# Display all open figures.
plt.show()
```



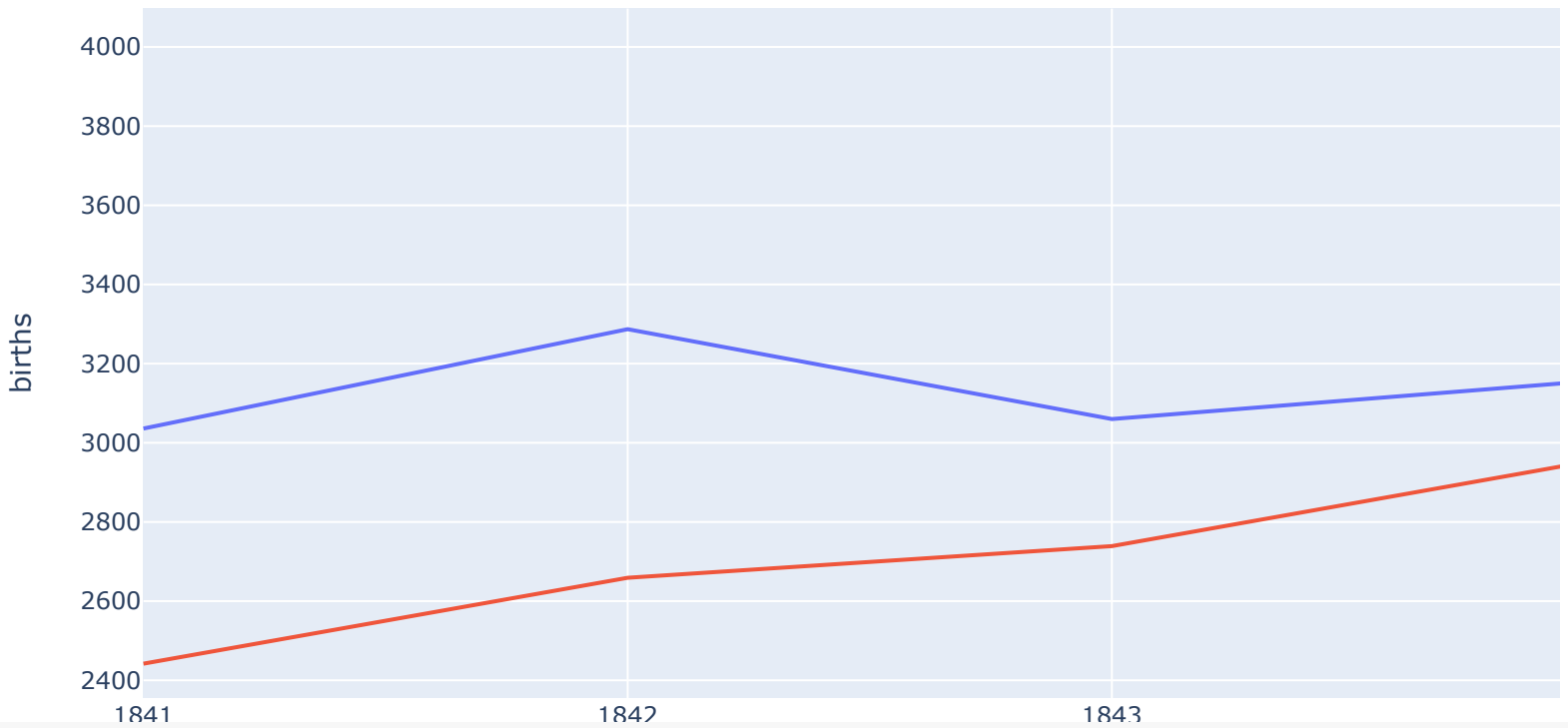
The Yearly Data Split by Clinic

Looking at the annual data.

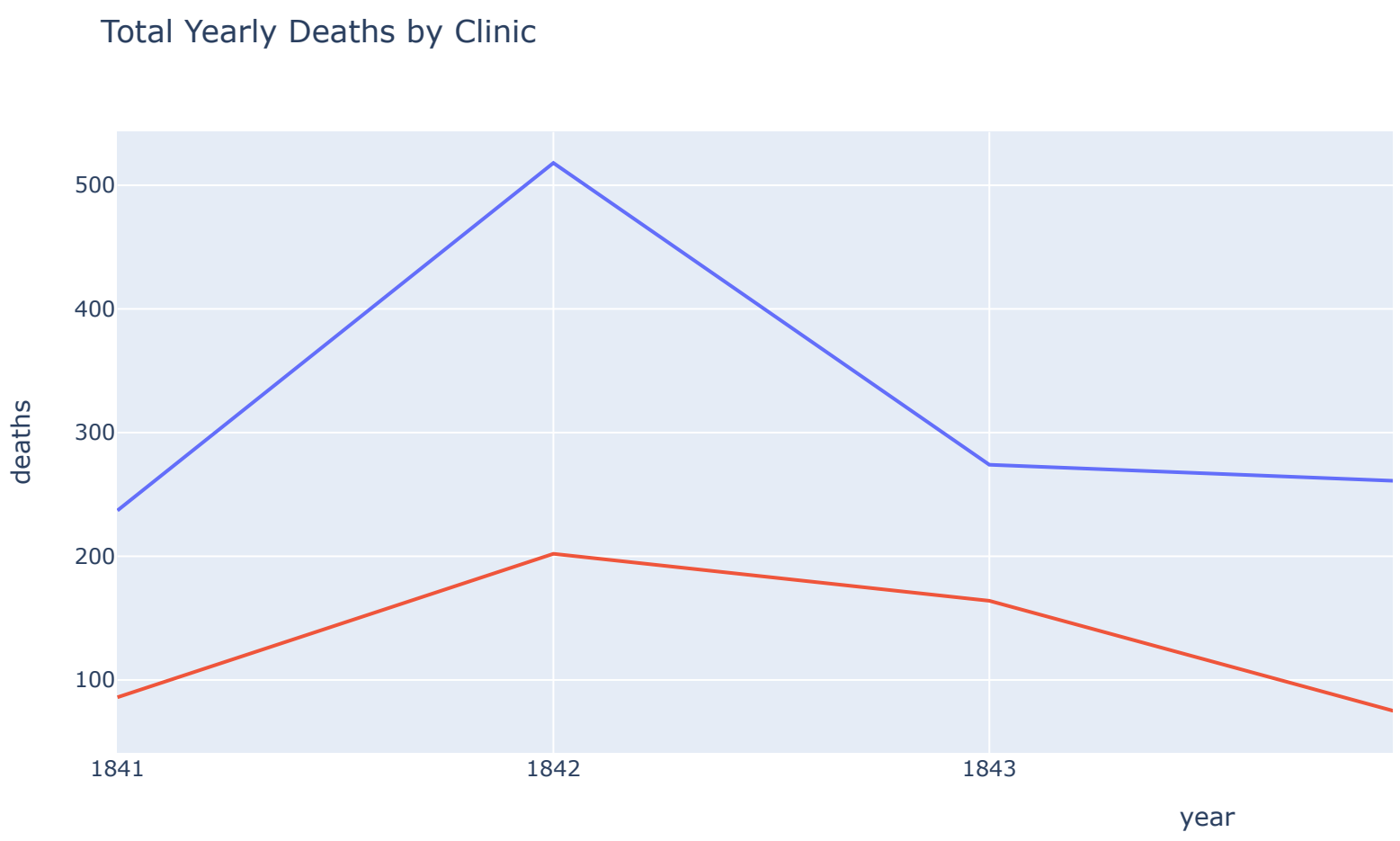
Using plotly to create line charts of the births and deaths of the two different clinics at the Vienna General Hospital.

- Which clinic is bigger or more busy judging by the number of births?
- Has the hospital had more patients over time?
- What was the highest number of deaths recorded in clinic 1 and clinic 2?

```
line = px.line(
    df_yearly,
    x='year',
    y='births',
    color='clinic',
    title='Total Yearly Births by Clinic',
)

line.show()
```




```
line = px.line(
    df_yearly,
    x='year',
    y='deaths',
    color='clinic',
    title='Total Yearly Deaths by Clinic',
)

line.show()
```



Calculating the Proportion of Deaths at Each Clinic

Calculating the proportion of maternal deaths per clinic.

- Working out the percentage of deaths for each row in the `df_yearly` DataFrame by adding a column called "pct_deaths".
- Calculating the average maternal death rate for clinic 1 and clinic 2 (i.e., the total number of deaths per the total number of births).
- Creating another plotly line chart to see how the percentage varies year over year with the two different clinics.
- Which clinic has a higher proportion of deaths?
- What is the highest monthly death rate in clinic 1 compared to clinic 2?

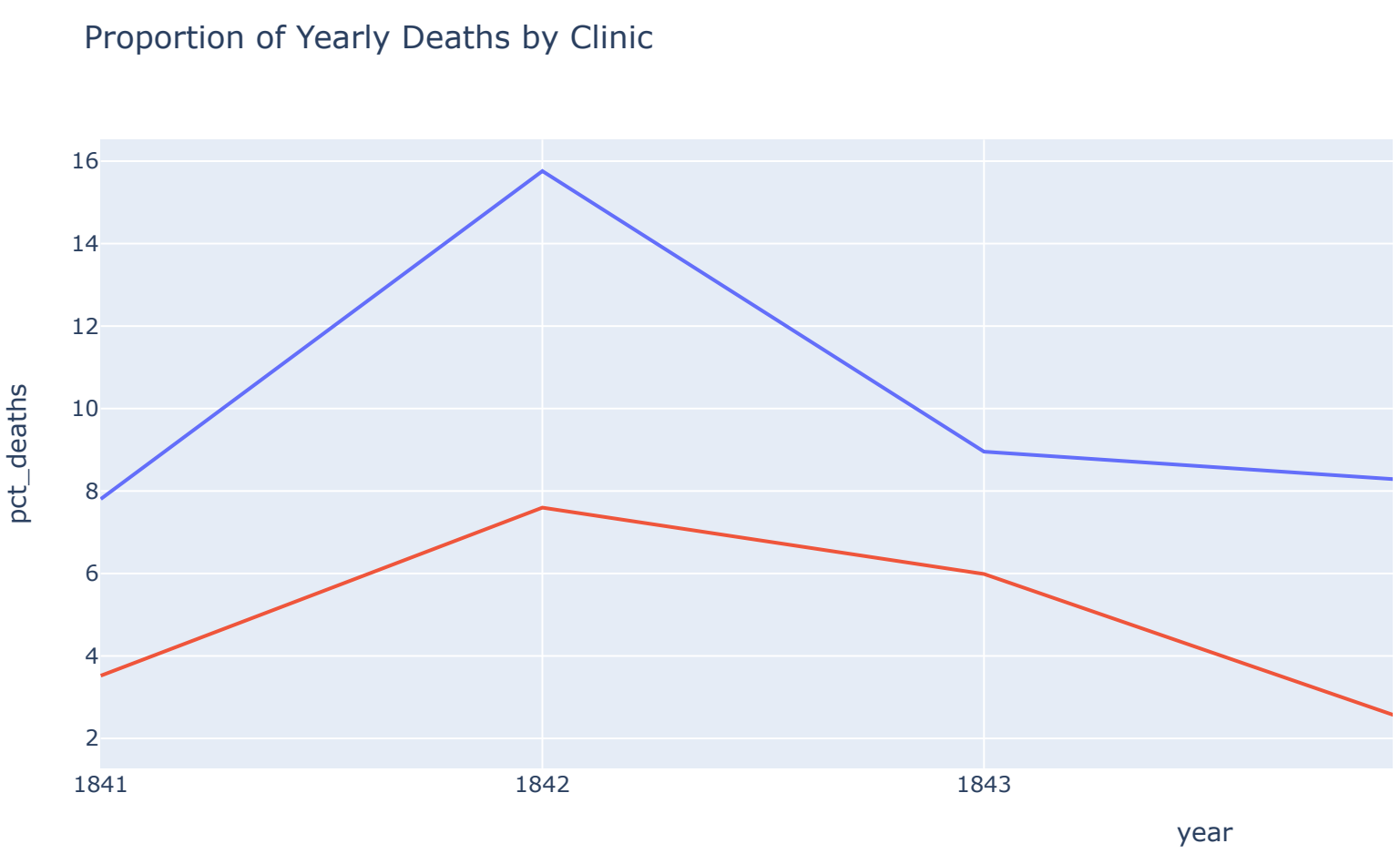
```
df_yearly["pct_deaths"] = df_yearly.deaths / df_yearly.births * 100
```

```
clinic_1 = df_yearly[df_yearly.clinic == 'clinic 1']
avg_c1 = clinic_1.deaths.sum() / clinic_1.births.sum() *100
print(f"Average death rate in clinic 1 is {avg_c1:.3}%.")

clinic_2 = df_yearly[df_yearly.clinic == 'clinic 2']
avg_c2 = clinic_2.deaths.sum() / clinic_2.births.sum() *100
print(f"Average death rate in clinic 2 is {avg_c2:.3}%.")
```

```
Average death rate in clinic 1 is 9.92%.
Average death rate in clinic 2 is 3.88%.
```

Plotting the Proportion of Yearly Deaths by Clinic

```
line = px.line(
    df_yearly,
    x='year',
    y='pct_deaths',
    color='clinic',
    title='Proportion of Yearly Deaths by Clinic',
)

line.show()
```



The Effect of Handwashing

Dr Semmelweis made handwashing obligatory in the summer of 1947. In fact, he ordered people to wash their hands with clorine (instead of water).

```
# Date when handwashing was made mandatory
handwashing_start = pd.to_datetime('1847-06-01')
```

- Adding a column called "pct_deaths" to `df_monthly` that has the percentage of deaths per birth for each row.
- Creating two subsets from the `df_monthly` data: before and after Dr Semmelweis ordered washing hand.
- Calculating the average death rate prior to June 1947.
- Calculating the average death rate after June 1947.

```
df_monthly['pct_deaths'] = df_monthly.deaths / df_monthly.births * 100
```

```
before_washing = df_monthly[df_monthly.date < handwashing_start]
after_washing = df_monthly[df_monthly.date >= handwashing_start]
```

```
bw_rate = before_washing.deaths.sum() / before_washing.births.sum() * 100
aw_rate = after_washing.deaths.sum() / after_washing.births.sum() * 100
print(f"Average death rate before 1847 was {bw_rate:.4}%.")
print(f"Average death rate after 1847 was {aw_rate:.3}%.")
```

```
Average death rate before 1847 was 10.53%.
Average death rate after 1847 was 2.15%.
```

Calculating a Rolling Average of the Death Rate

Creating a DataFrame that has the 6 month rolling average death rate prior to mandatory handwashing.

We'll need to set the dates as the index in order to avoid the date column being dropped during the calculation.

```
roll_df = before_washing.set_index('date')
roll_df = roll_df.rolling(window=6).mean()
```

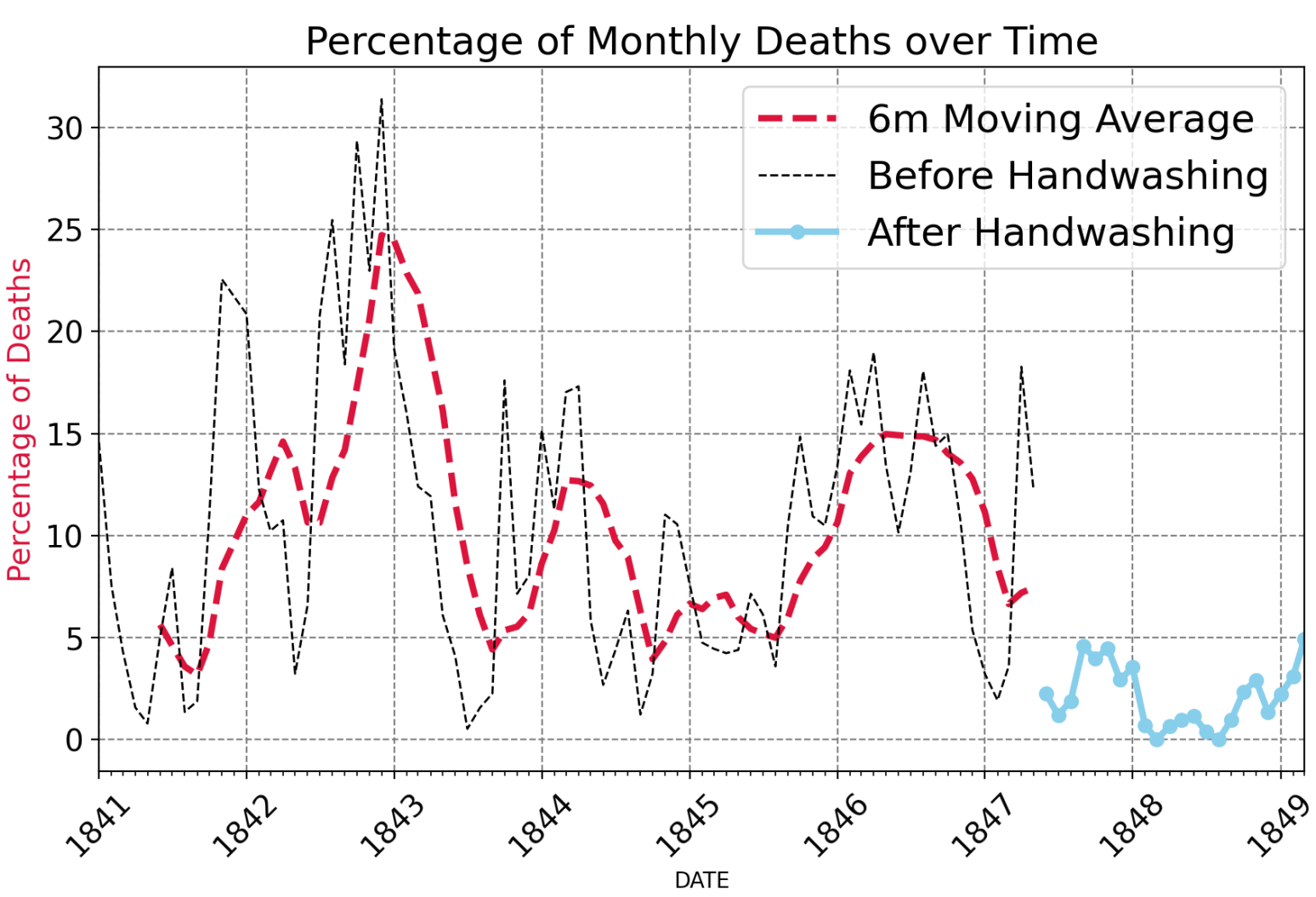
Highlighting Subsections of a Line Chart

Copy-paste and then modifing the Matplotlib chart from before to plot the monthly death rates (instead of the total number of births and deaths). The chart should look something like this:

- Adding 3 seperate lines to the plot: the death rate before handwashing, after handwashing, and the 6-month moving average before handwashing.
- Showing the monthly death rate before handwashing as a thin dashed black line.
- Showing the moving average as a thicker, crimson line.
- Showing the rate after handwashing as a skyblue line with round markers.
- Looking at the code snippet in the documentation to see how you can add a legend to the chart.

```
plt.figure(figsize=(10,6), dpi=200)
# the title of the graph
plt.title("Percentage of Monthly Deaths over Time", fontsize=18)

plt.yticks(fontsize=14)
# Increase the size adn rotate the labels on the x-axis
plt.xticks(fontsize=14, rotation=45)

plt.ylabel('Percentage of Deaths', color='crimson', fontsize=14)


# getting the axis
ax = plt.gca()

# labels
ax.set_xlabel("DATE")


# setting the min and max values on the axes
ax.set_xlim([df_monthly.date.min(), df_monthly.date.max()])
# formating where the major and minor ticks should be
ax.xaxis.set_major_locator(years)
ax.xaxis.set_major_formatter(years_fmt)
ax.xaxis.set_minor_locator(months)


plt.grid(color='grey', linestyle='--')


ma_line, = plt.plot(
    roll_df.index,
    roll_df.pct_deaths,
    color='crimson',
    linewidth=3,
    linestyle='--',
    label='6m Moving Average',
)

bw_line, = plt.plot(
    before_washing.date,
    before_washing.pct_deaths,
    color='black',
```

```
    linewidth=1,
    linestyle='--',
    label='Before Handwashing',
)

aw_line, = plt.plot(
    after_washing.date,
    after_washing.pct_deaths,
    color='skyblue',
    linewidth=3,
    marker='o',
    label='After Handwashing',
)

plt.legend(
    handles=[ma_line, bw_line, aw_line],
    fontsize=18,
)

# Display all open figures.
plt.show()
```



Statistics - Calculating the Difference in the Average Monthly Death Rate

- What was the average percentage of monthly deaths before handwashing?
- What was the average percentage of monthly deaths after handwashing was made obligatory?

- By how much did handwashing reduce the average chance of dying in childbirth in percentage terms?
- How do these numbers compare to the average for all the 1840s that we calculated earlier?
- How many times lower are the chances of dying after handwashing compared to before?

```
avg_prob_before = before_washing.pct_deaths.mean()
print(f"Chance of death during childbirth before handwashing: {avg_prob_before:.3}%.")

avg_prob_after = after_washing.pct_deaths.mean()
print(f"Chance of death during childbirth after handwashing: {avg_prob_after:.3}%.")

mean_diff = avg_prob_before - avg_prob_after
print(f"Handwashing reduced the monthly proportion of deaths by {mean_diff:.3}%.")

times_diff = avg_prob_before / avg_prob_after
print(f"This is a {times_diff:.2}x improvement.")
```

```
Chance of death during childbirth before handwashing: 10.5%.
Chance of death during childbirth after handwashing: 2.11%.
Handwashing reduced the monthly proportion of deaths by 8.4%.
This is a 5.0x improvement.
```

Using Box Plots to Show How the Death Rate Changed Before and After Handwashing

- Using NumPy's `.where()` function to add a column to `df_monthly` that shows if a particular date was before or after the start of handwashing.
- Using plotly to create box plot of the data before and after handwashing.
- How did key statistics like the mean, max, min, 1st and 3rd quartile changed as a result of the new policy?

```
df_monthly['washing_hands'] = np.where(
                        df_monthly.date < handwashing_start,
                        'No',
                        'Yes',)
```

```
box = px.box(
    df_monthly,
    x='washing_hands',
    y='pct_deaths',
    color='washing_hands',
    title='How have the stats changed with Handwashing?'
)

box.update_layout(
    xaxis_title='Washing Hands?',
    yaxis_title='Percentage of Monthly Deaths',
)

box.show()
```

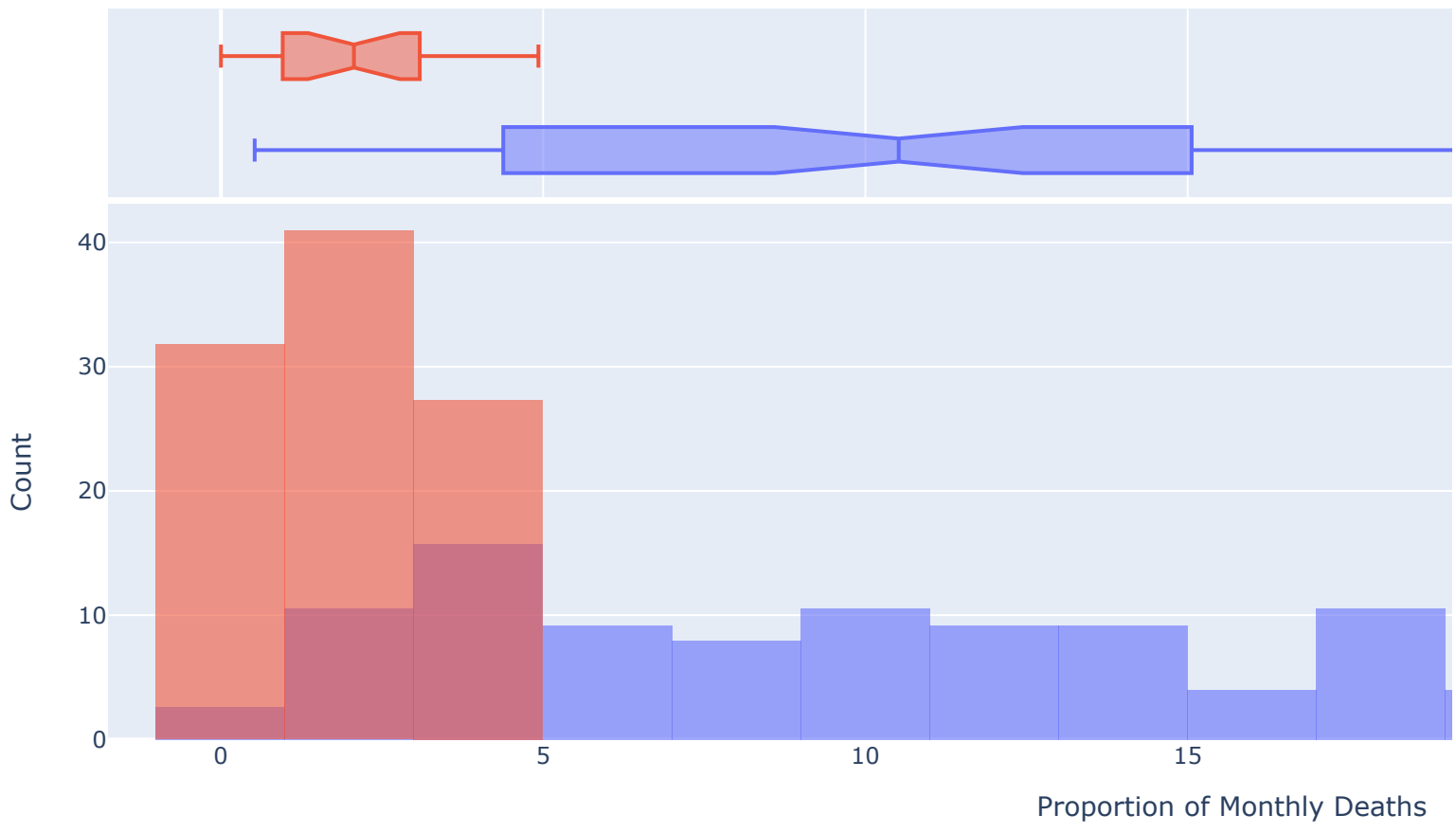
Double-click (or enter) to edit

Using Histograms to Visualise the Monthly Distribution of Outcomes

Creating a plotly histogram to show the monthly percentage of deaths.

- Using the `color` parameter to display two overlapping histograms.
- The time period of handwashing is shorter than not handwashing. Changing `histnorm` to `percent` to make the time periods comparable.
- Making the histograms slighlty transparent
- Experimenting with the number of bins on the histogram. Which number work well in communicating the range of outcomes?
- Displaying your box plot on the top of the histogram using the `marginal` parameter.

```
hist = px.histogram(
    df_monthly,
    x='pct_deaths',
    color='washing_hands',
    nbins=30,
    opacity=0.6,
    barmode='overlay',
    histnorm='percent',
    marginal='box',
)

hist.update_layout(
    xaxis_title="Proportion of Monthly Deaths",
    yaxis_title='Count',
)

hist.show()
```

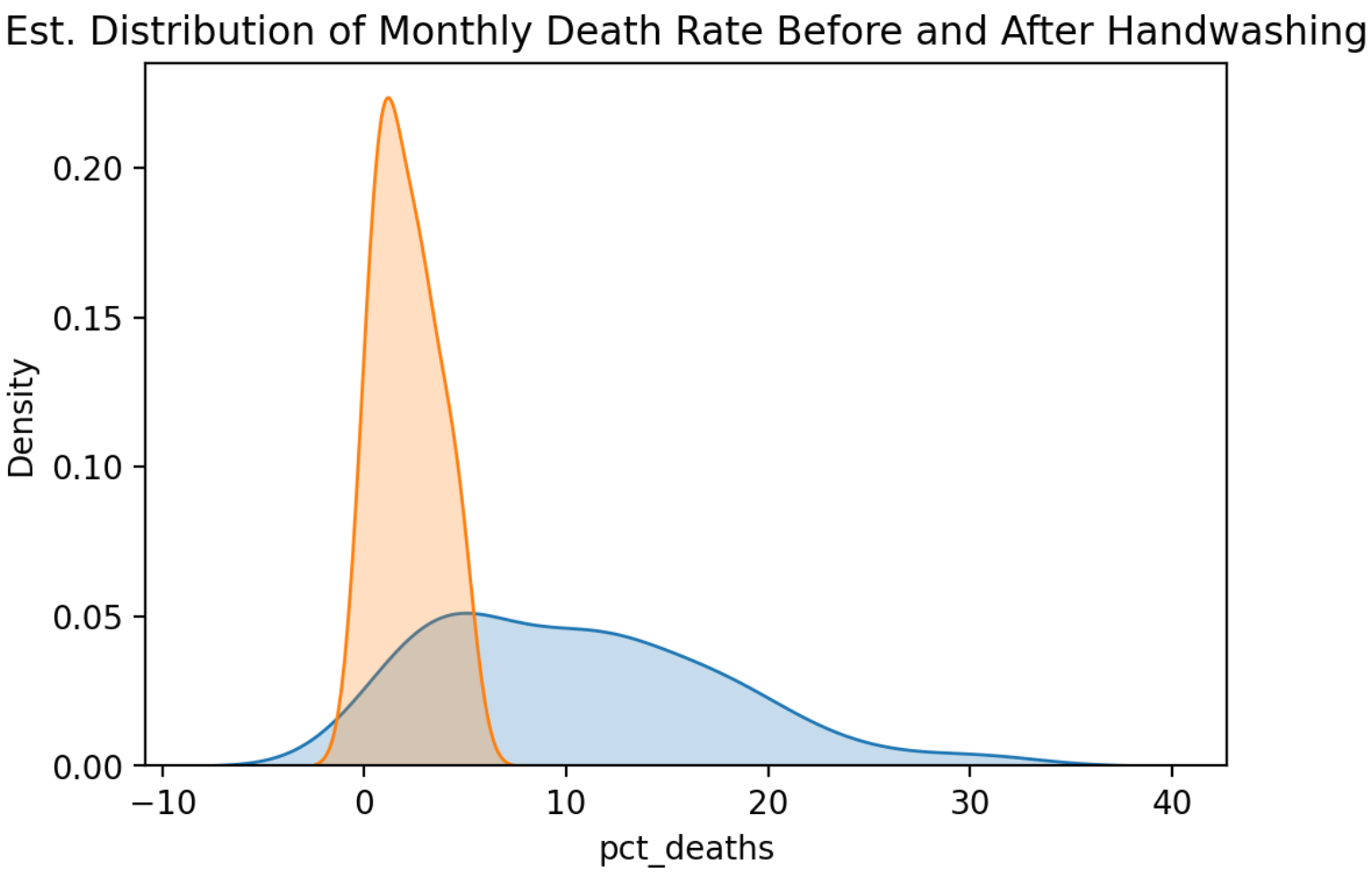


Using a Kernel Density Estimate (KDE) to visualise a smooth distribution

Useing Seaborn's `.kdeplot()` to create two kernel density estimates of the `pct_deaths`, one for before handwashing and one for after.

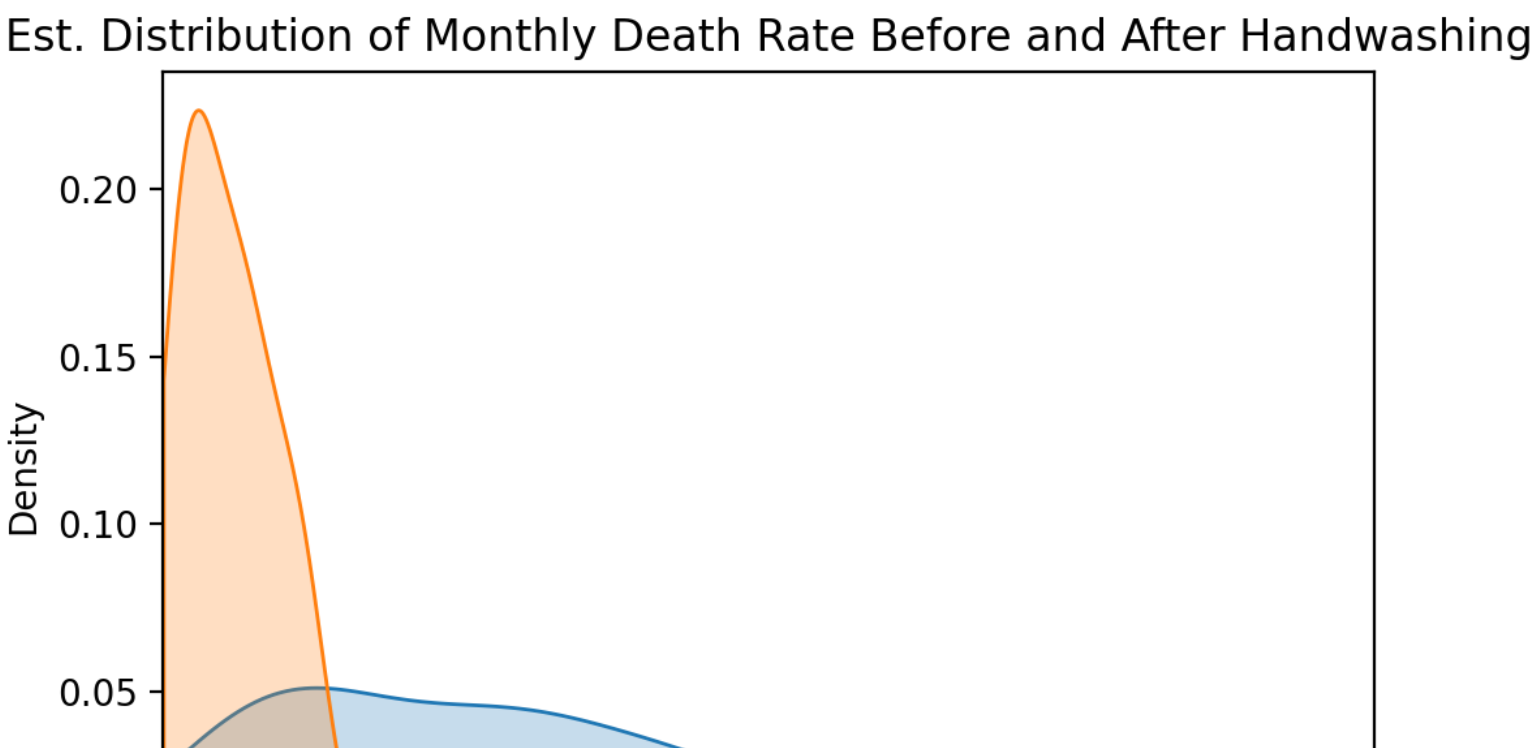
- Using the `fill` parameter to give your two distributions different colours.

```
plt.figure(dpi=200)

# By default the distribution estimate includes a negative death rate!
sns.kdeplot(before_washing.pct_deaths, fill=True)
sns.kdeplot(after_washing.pct_deaths, fill=True)
plt.title('Est. Distribution of Monthly Death Rate Before and After Handwashing')
plt.show()
```



- The problem is that we end up with a negative monthly death rate !!
    - Using the `clip` parameter to address the problem.

```
plt.figure(dpi=200)

# By default the distribution estimate includes a negative death rate!
#  lower bound of 0 for the death rate - the clip parameter
sns.kdeplot(before_washing.pct_deaths, fill=True, clip=(0,100))
sns.kdeplot(after_washing.pct_deaths, fill=True, clip=(0,100))
plt.title('Est. Distribution of Monthly Death Rate Before and After Handwashing')
plt.xlim(0, 40)
plt.show()
```



Using a T-Test to Show Statistical Significance

Using a t-test to determine if the differences in the means are statistically significant or purely due to chance.

If the p-value is less than 1% then we can be 99% certain that handwashing has made a difference to the average monthly death rate.

- Importing `stats` from scipy
- Using the `.ttest_ind()` function to calculate the t-statistic and the p-value
- Is the difference in the average proportion of monthly deaths statistically significant at the 99% level?

```
t_stat, p_value = stats.ttest_ind(
    a=before_washing.pct_deaths,
    b=after_washing.pct_deaths,
)
print(f'p-palue is {p_value:.10f}')
print(f't-statstic is {t_stat:.4}')
```

```
p-palue is 0.0000002985
t-statstic is 5.512
```

So better wash your hands !!!