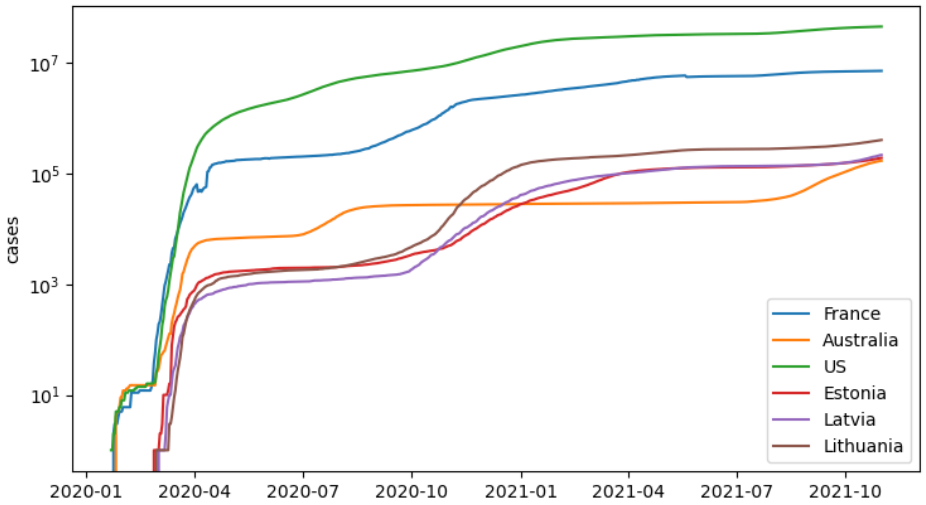
# Handout: Simulations and analysis of Covid-19 spread
# OpenData Forum
# 02.11.2021

## 1 Covid-19 Data

The real world analysis is based on data provided by John Hopkins University, found here, licensed under CC BY 4.0.
Below you will find a sample code that reads in global case data csv and plots the case data for selected countries (csv found under `csse_covid_19_data/csse_covid_19_time_series/` in the repo). The csv is read in using `pandas`, as it provides convenient data manipulation tools. Code is accessible here.

```
"""Example code for reading in worldwide case data and plotting it"""
import pandas as pd
from matplotlib import pyplot as plt

filename = "time_series_covid19_confirmed_global.csv"
df = pd.read_csv(filename)

for country in ["France", "Australia", "US", "Estonia", "Latvia", "Lithuania"]:
  df_country = df[df["Country/Region"] == country] # Filter out country
  df_country_cases = df_country.iloc[:, 4:] # case data is 5th column onward
  df_country_cases = df_country_cases.sum(axis=0) # some countries have multiple rows
  df_country_cases.index = pd.to_datetime(df_country_cases.index) # string to datetime
  plt.plot(df_country_cases.index, df_country_cases, label=country)

plt.legend()
plt.yscale("log")
plt.ylabel("cases")
plt.show()
```



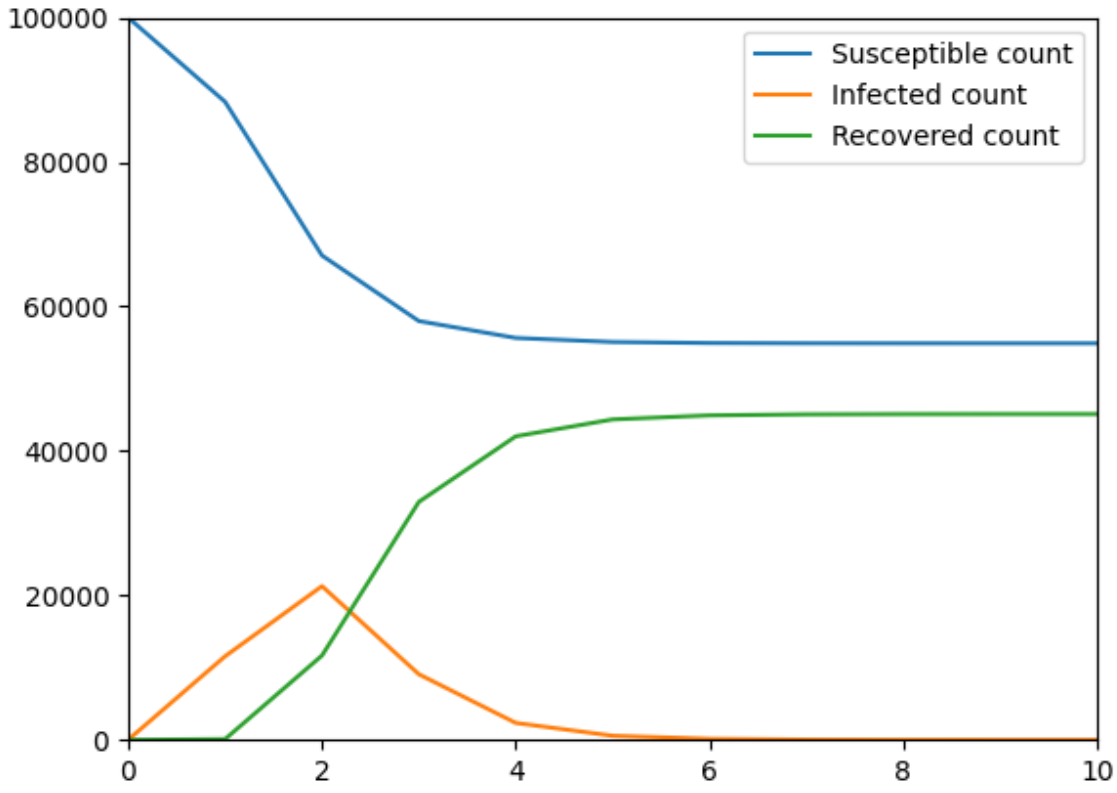

output plot

## 2 Sample code using the NetworkX package

`NetworkX` is a Python package that can be used to generate scale-free graphs. codeNetworkX documentation can be found here. What follows is simple code for running a stochastic SIR simulation on the generated graph. Infection is assumed to last for 1 day. The following code can be also accessed here.

```
"""Example code using the networkx package.
A simple stochastic SIR analysis is performed."""
import networkx as nx
import numpy as np
import random
from matplotlib import pyplot as plt

SUSCEPTIBLE, INFECTED, RECOVERED = 0, 1, 2

P = 0.30 # Probability of the infection spreading
N = 100000
graph = nx.scale_free_graph(N, alpha=0.15, beta=0.7,
                            gamma=0.15, delta_in=4.5, delta_out=4.5)

nodes = np.full(N, SUSCEPTIBLE) # List of the statuses of the nodes
# Infect 10 random nodes
for _ in range(10):
  nodes[random.randint(0, N - 1)] = INFECTED

S_count, I_count, R_count = [N - 10], [10], [0]
prev_infected_count, infected_count = 0, 0
for day in range(100):
  # Each day, let all infected people infect their susceptible neighbours
  currently_infected = (nodes == INFECTED)
  for node_id, status in enumerate(nodes):
    if nodes[node_id] == INFECTED:
      for _, neighbour in graph.edges(node_id):
        if nodes[neighbour] == SUSCEPTIBLE and random.random() < P:
          nodes[neighbour] = INFECTED
  # Recover the people who were infected beginning of the day
  nodes[currently_infected] = RECOVERED

  infected_count = np.sum(nodes==INFECTED)
  if infected_count == 0:
    print(f"Pandemic is over after {day} days,
                cumulative infection count {np.sum(nodes==SUSCEPTIBLE)}")
    break
  if day > 0:
    S_count.append(np.sum(nodes==SUSCEPTIBLE))
    I_count.append(np.sum(nodes==INFECTED))
    R_count.append(np.sum(nodes==RECOVERED))
```

```
plt.plot(S_count, label="Susceptible count")
plt.plot(I_count, label="Infected count")
plt.plot(R_count, label="Recovered count")
plt.xlim(0, len(S_count) - 1)
plt.ylim(0, N)
plt.legend()
plt.show()
```




output plot