eda

June 18, 2025

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

sns.set(style="whitegrid")
pd.set_option('display.max_columns', 100)
```

0.0.1 1. Data Samples

[2]:
```
data_path = "D:/Programming/Summer-2025-DL-Project-PINN/data/covid_county_cases.csv"
df = pd.read_csv(data_path)

# Restore dates from "t"
start_date = datetime(2020, 1, 21)
df["date"] = pd.to_datetime(df["t"], unit="D", origin=start_date)

print("Loaded shape:", df.shape)
display(df.head())
print(df.info())
```

```
Loaded shape: (2409352, 5)
```

	lon	lat	t	u	date
0	-164.040108	64.942121	85	10.167768	2020-04-15
1	-164.040108	64.942121	86	10.167768	2020-04-16
2	-164.040108	64.942121	87	10.167768	2020-04-17
3	-164.040108	64.942121	88	10.167768	2020-04-18
4	-164.040108	64.942121	89	10.167768	2020-04-19

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2409352 entries, 0 to 2409351
Data columns (total 5 columns):
 #   Column  Dtype
---  ------  -----
 0   lon     float64
 1   lat     float64
```

```
 2   t       int64
 3   u       float64
 4   date    datetime64[ns]
dtypes: datetime64[ns](1), float64(3), int64(1)
memory usage: 91.9 MB
None
```

0.0.2 2. Summary Statistics & Nulls

[3]:
```
print("Summary Statistics:")
display(df.describe())

print("Null Value Check:")
print(df.isnull().sum())
```

```
Summary Statistics:
```

	lon	lat	t	u	\
count	2.409352e+06	2.409352e+06	2.409352e+06	2.409352e+06	
mean	-9.214093e+01	3.838049e+01	4.573247e+02	1.011351e+04	
min	-1.640401e+02	1.960199e+01	0.000000e+00	1.028686e-02	
25%	-9.808979e+01	3.464985e+01	2.650000e+02	2.138229e+03	
50%	-9.024324e+01	3.832408e+01	4.580000e+02	9.257088e+03	
75%	-8.343685e+01	4.174292e+01	6.510000e+02	1.522107e+04	
max	-6.762834e+01	6.935328e+01	8.430000e+02	3.843137e+05	
std	1.261735e+01	5.230411e+00	2.235216e+02	8.760030e+03	

	date
count	2409352
mean	2021-04-22 07:47:36.891645440
min	2020-01-21 00:00:00
25%	2020-10-12 00:00:00
50%	2021-04-23 00:00:00
75%	2021-11-02 00:00:00
max	2022-05-13 00:00:00
std	NaN

```
Null Value Check:
lon     0
lat     0
t       0
u       0
date    0
dtype: int64
```

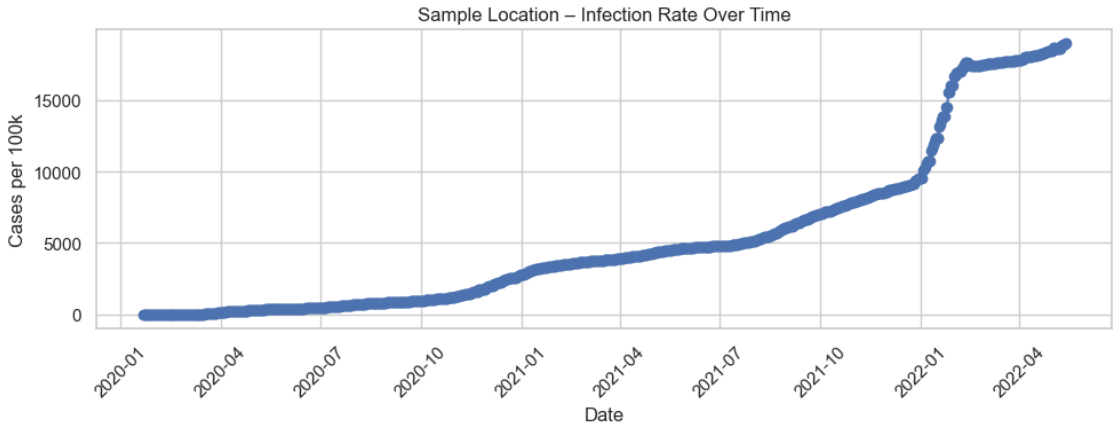
0.0.3 3.Unique Locations & Data Coverage

[4]:
```
print(f"Unique locations (lon, lat): {df[['lon','lat']].drop_duplicates().
  ↪shape[0]}")
print(f"Time Range: {df['date'].min().date()} to {df['date'].max().date()}")
```

```
Unique locations (lon, lat): 3124
Time Range: 2020-01-21 to 2022-05-13
```

0.0.4 4.Sample Time Series for a Random or Top County

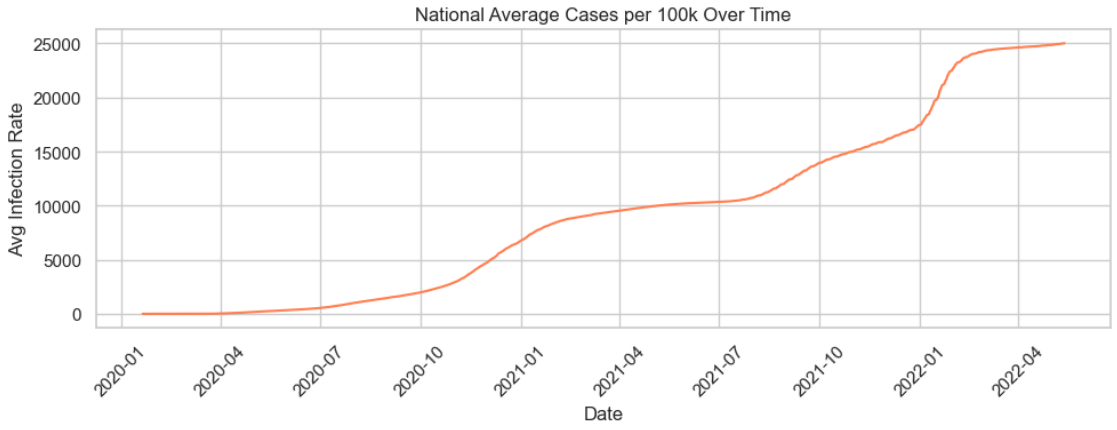
[5]:
```
top_location = df.groupby(["lon", "lat"]).size().sort_values(ascending=False).
  ↪index[0]
df_sample = df[(df["lon"] == top_location[0]) & (df["lat"] == top_location[1])]

plt.figure(figsize=(10, 4))
plt.plot(df_sample["date"], df_sample["u"], marker='o')
plt.title("Sample Location - Infection Rate Over Time")
plt.xlabel("Date")
plt.ylabel("Cases per 100k")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



0.0.5 5. National Average Trend

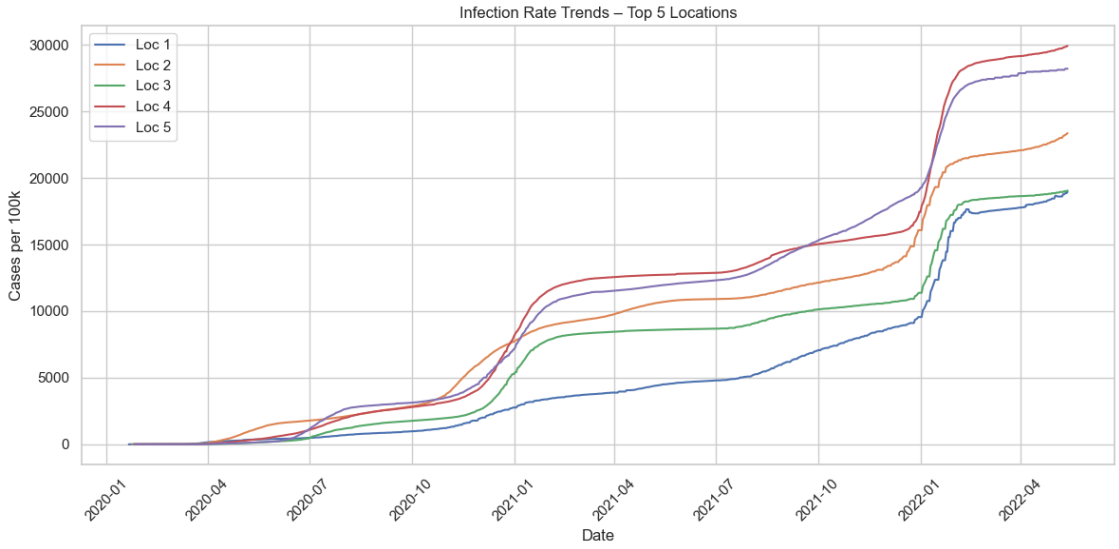
[6]:
```
df_national = df.groupby("date")["u"].mean().reset_index()

plt.figure(figsize=(10, 4))
sns.lineplot(data=df_national, x="date", y="u", color="coral")
plt.title("National Average Cases per 100k Over Time")
```

```
plt.xlabel("Date")
plt.ylabel("Avg Infection Rate")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



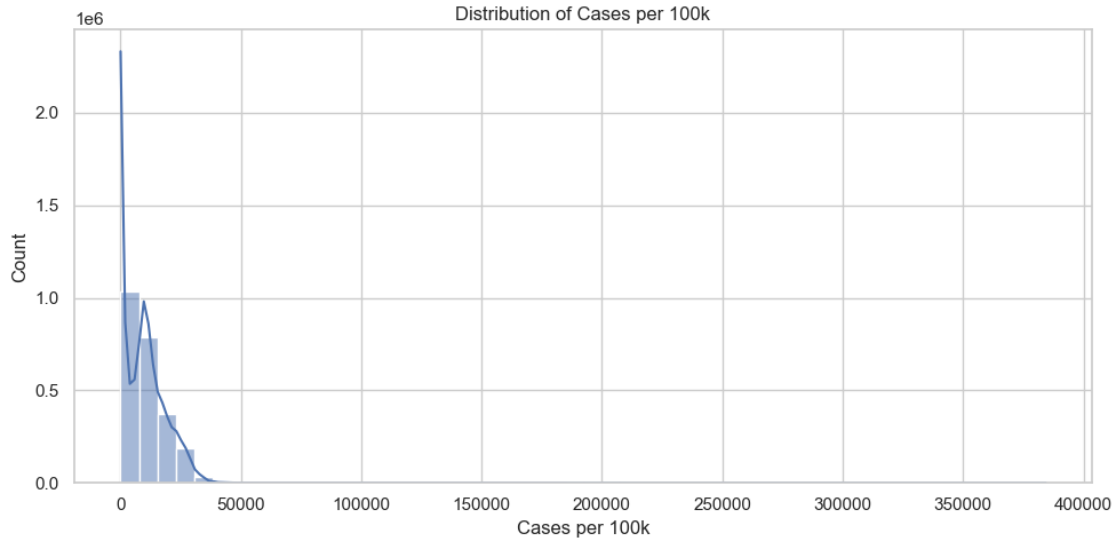
0.0.6 6. Heatmap: Top Locations

[7]:
```
# Get top 5 locations with most records
top5_locs = df.groupby(["lon", "lat"]).size().sort_values(ascending=False).
 ↪head(5).index

# Plot time series for each
plt.figure(figsize=(12, 6))
for i, loc in enumerate(top5_locs):
    df_sub = df[(df["lon"] == loc[0]) & (df["lat"] == loc[1])]
    plt.plot(df_sub["date"], df_sub["u"], label=f"Loc {i+1}")
plt.legend()
plt.title("Infection Rate Trends - Top 5 Locations")
plt.xlabel("Date")
plt.ylabel("Cases per 100k")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()
```



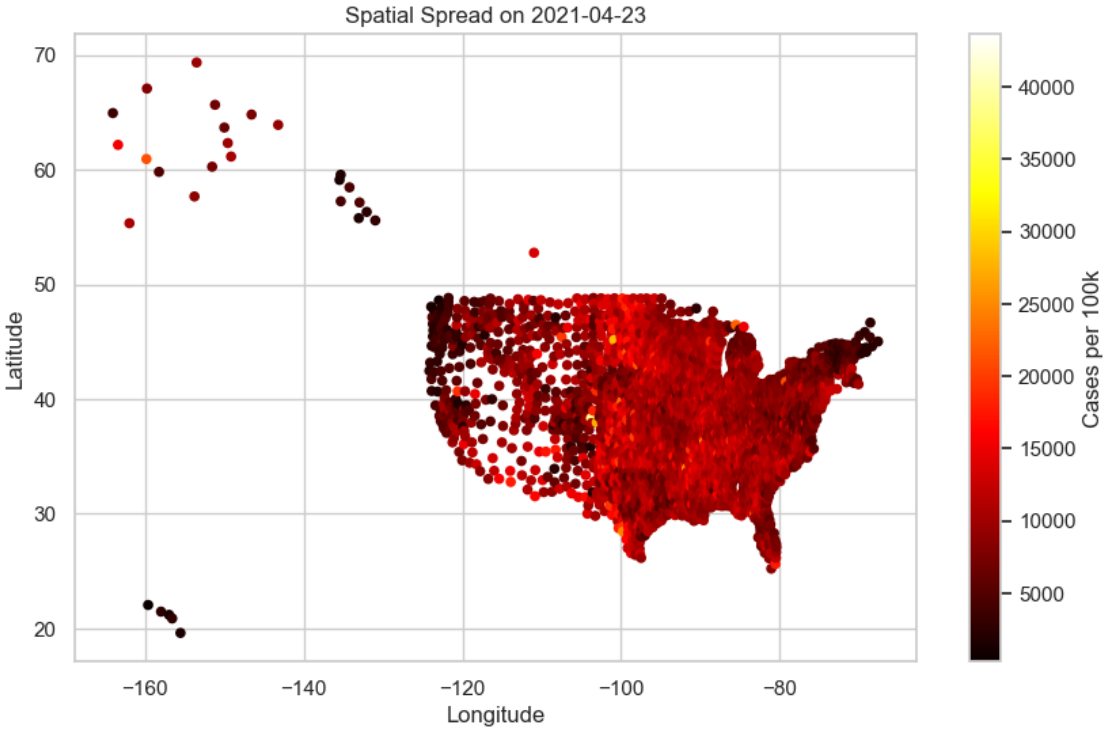
0.0.7 7.Daily Histogram of Cases per 100k

[8]:
```
plt.figure(figsize=(10, 5))
sns.histplot(df["u"], bins=50, kde=True)
plt.title("Distribution of Cases per 100k")
plt.xlabel("Cases per 100k")
plt.tight_layout()
plt.show()
```



0.0.8 8.Geographic Snapshots for a Specific Date

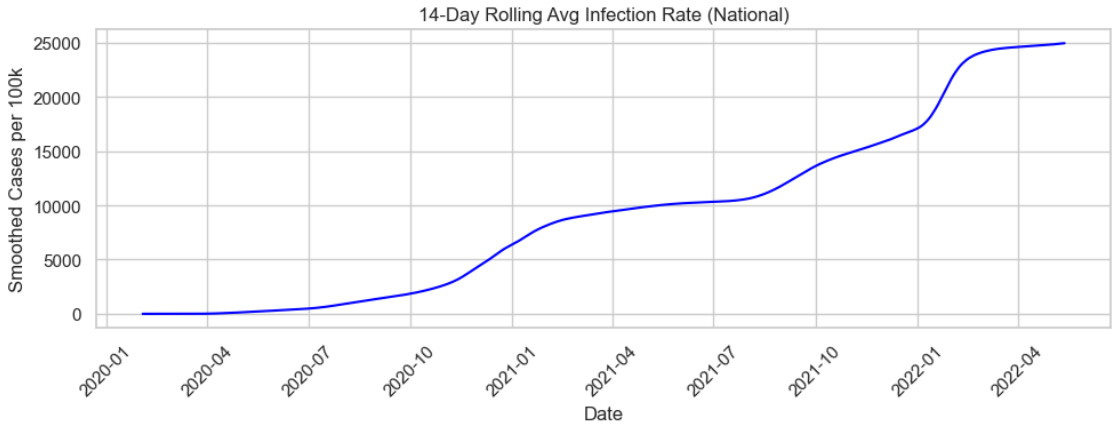
[9]:
```
# Pick a snapshot day (e.g., peak period)
snapshot_day = df["date"].median()
df_day = df[df["date"] == snapshot_day]

plt.figure(figsize=(10, 6))
plt.scatter(df_day["lon"], df_day["lat"], c=df_day["u"], cmap="hot", s=20)
plt.colorbar(label="Cases per 100k")
plt.title(f"Spatial Spread on {snapshot_day.date()}")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(True)
plt.show()
```



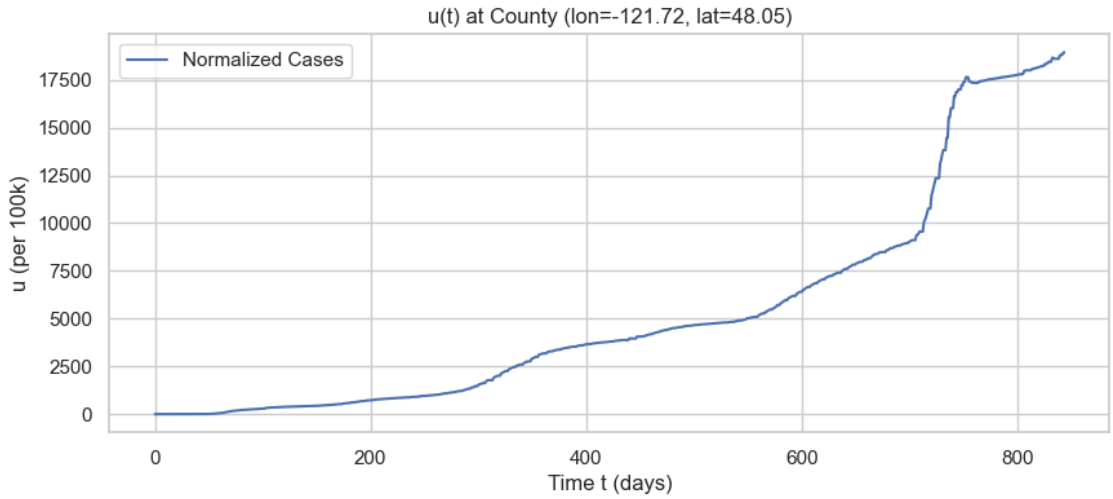
0.0.9 9.Rolling Average (Optional Smoothing)

[10]:
```
df_rolling = df.groupby("date")["u"].mean().rolling(window=14).mean().
  ↪reset_index()

plt.figure(figsize=(10, 4))
sns.lineplot(data=df_rolling, x="date", y="u", color="blue")
plt.title("14-Day Rolling Avg Infection Rate (National)")
```

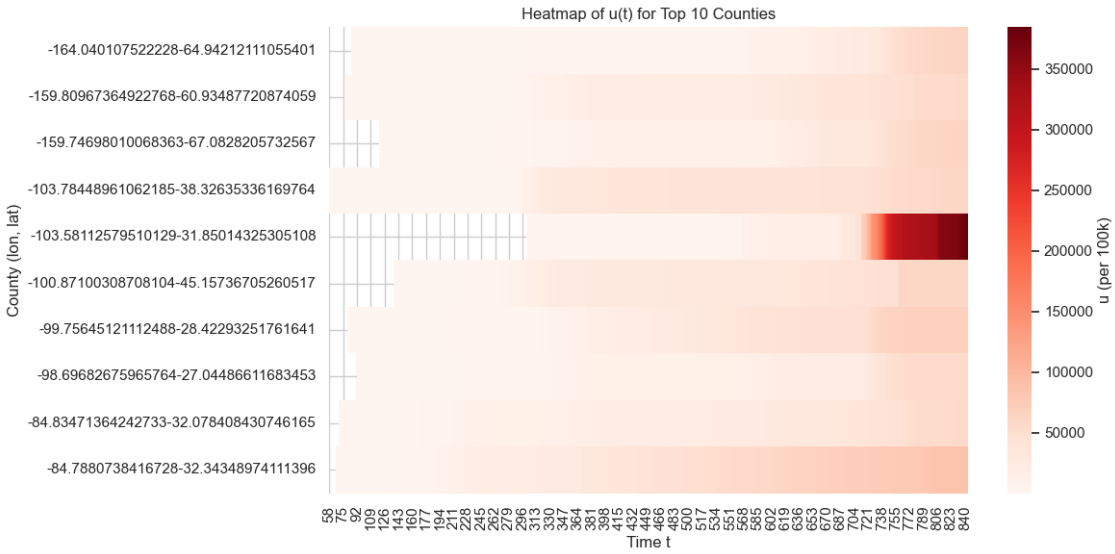
```
plt.xlabel("Date")
plt.ylabel("Smoothed Cases per 100k")
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```



[11]:
```
# Pick a random location
sample = df.groupby(["lon", "lat"]).size().sort_values(ascending=False).index[0]
df_sample = df[(df["lon"] == sample[0]) & (df["lat"] == sample[1])]

plt.figure(figsize=(10, 4))
plt.plot(df_sample["t"], df_sample["u"], label="Normalized Cases")
plt.title(f"u(t) at County (lon={sample[0]:.2f}, lat={sample[1]:.2f})")
plt.xlabel("Time t (days)"); plt.ylabel("u (per 100k)")
plt.grid(True)
plt.legend()
plt.show()
```



[12]:
```
top_coords = df.groupby(["lon", "lat"])["u"].max().nlargest(10).index

df_top = df[df.set_index(["lon", "lat"]).index.isin(top_coords)]

pivot = df_top.pivot_table(index="t", columns=["lon", "lat"], values="u")

plt.figure(figsize=(12, 6))
sns.heatmap(pivot.T, cmap="Reds", cbar_kws={'label': 'u (per 100k)'})
plt.title("Heatmap of u(t) for Top 10 Counties")
plt.xlabel("Time t"); plt.ylabel("County (lon, lat)")
plt.tight_layout()
plt.show()
```



[13]:
```
print("  Preprocessing Insights:\n")

preprocessing = {
    "Temporal Range": f"{df['t'].min()} to {df['t'].max()} (days)",
    "Spatial Resolution": f"{len(df['lon'].unique())} unique counties",
    "Missing Values": "None (handled)",
    "Normalization": "u scaled as cases per 100,000 population",
    "Smoothing": "Optional: rolling average or Savitzky-Golay smoothing",
    "Train/Valid/Test Split": {
        "Train": "First 80% of t range",
        "Validation/Test": "Last 20% of t range (future prediction)"
    },
    "Collocation Sampling": "Sampled (x, t) from grid or randomly"
}

import json
print(json.dumps(preprocessing, indent=4))
```

```
 Preprocessing Insights:

{
    "Temporal Range": "0 to 843 (days)",
    "Spatial Resolution": "3124 unique counties",
    "Missing Values": "None (handled)",
    "Normalization": "u scaled as cases per 100,000 population",
    "Smoothing": "Optional: rolling average or Savitzky\u2013Golay smoothing",
    "Train/Valid/Test Split": {
        "Train": "First 80% of t range",
        "Validation/Test": "Last 20% of t range (future prediction)"
    },
    "Collocation Sampling": "Sampled (x, t) from grid or randomly"
}
```