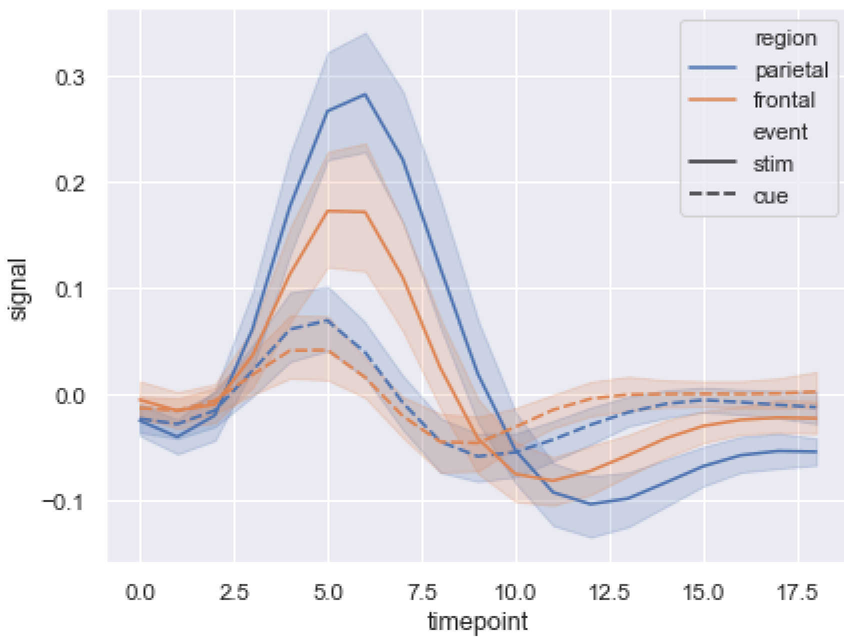
Lecture Notes (Lecture 11 - 2 Hours)

Duration: 2 hours

Timeseries plot with error bands



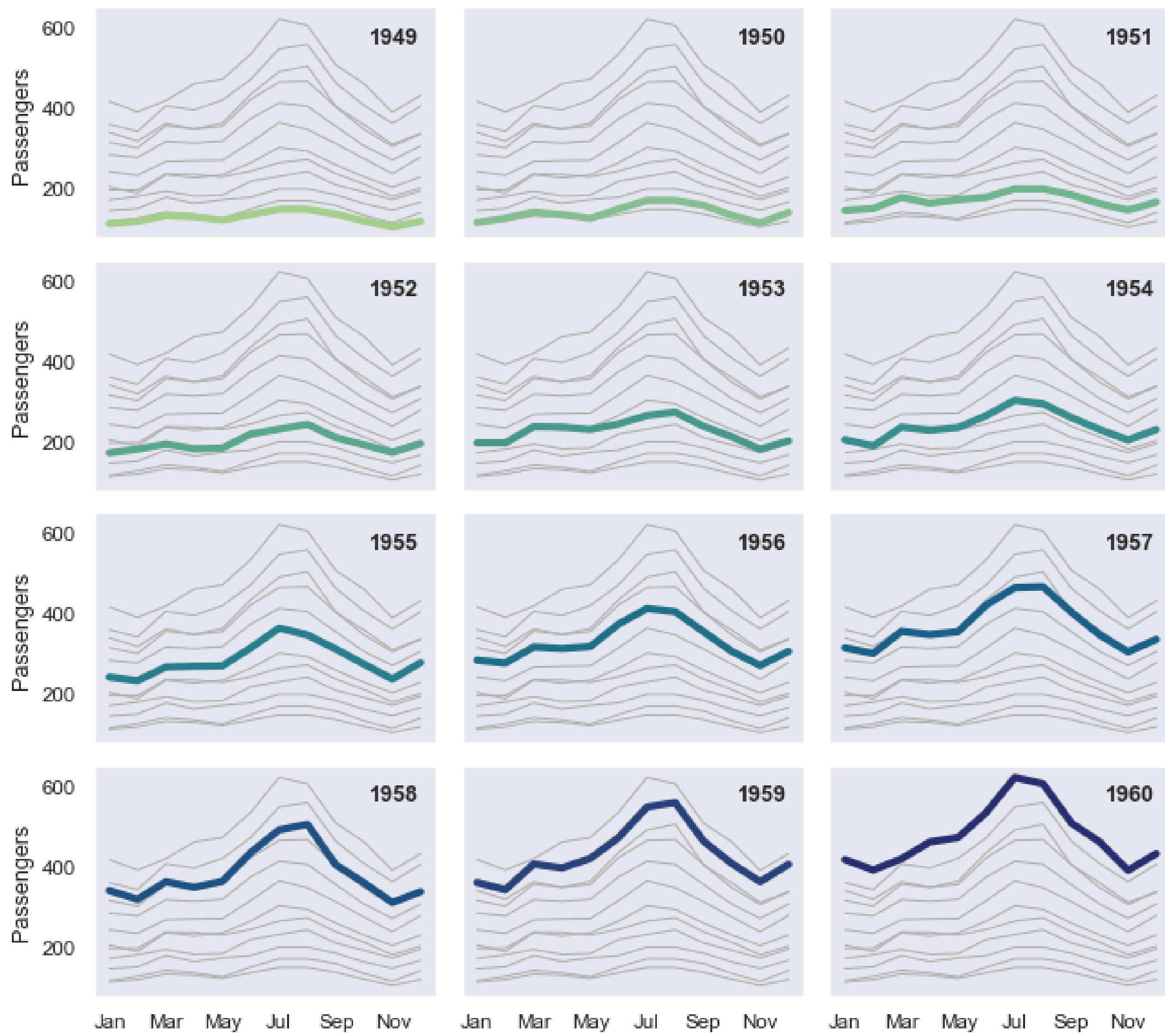
seaborn components used: `set_theme()`, `load_dataset()`, `lineplot()`

```python
import seaborn as sns
sns.set_theme(style="darkgrid")

# Load an example dataset with long-form data
fmri = sns.load_dataset("fmri")

# Plot the responses for different events and regions
sns.lineplot(x="timepoint", y="signal",
             hue="region", style="event",
             data=fmri)
```

Small multiple time series



seaborn components used: `set_theme()`, `load_dataset()`, `relplot()`, `lineplot()`

```
import seaborn as sns

sns.set_theme(style="dark")
flights = sns.load_dataset("flights")

# Plot each year's time series in its own facet
g = sns.relplot(
    data=flights,
    x="month", y="passengers", col="year", hue="year",
    kind="line", palette="crest", linewidth=4, zorder=5,
    col_wrap=3, height=2, aspect=1.5, legend=False,
)
```

```python
# Iterate over each subplot to customize further
for year, ax in g.axes_dict.items():

    # Add the title as an annotation within the plot
    ax.text(.8, .85, year, transform=ax.transAxes, fontweight="bold")

    # Plot every year's time series in the background
    sns.lineplot(
        data=flights, x="month", y="passengers", units="year",
        estimator=None, color=".7", linewidth=1, ax=ax,
    )

# Reduce the frequency of the x axis ticks
ax.set_xticks(ax.get_xticks()[::2])

# Tweak the supporting aspects of the plot
g.set_titles("")
g.set_axis_labels("", "Passengers")
g.tight_layout()
```

Time Series Visualizations in Python

Time series data consists of observations collected sequentially over time. Visualizing time series helps in identifying patterns such as trends, seasonality, and anomalies.

---

1. Basic Line Plot for Time Series

A line plot is the simplest way to visualize time series data. It connects data points using straight lines and is helpful in observing trends over time.

Example: Line Plot using `matplotlib`

python
Copy code

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# Create a date range
date_range = pd.date_range(start='2022-01-01', periods=100, freq='D')
```

```
# Generate random time series data
data = np.random.randn(100).cumsum()

# Create a DataFrame
df = pd.DataFrame({'Date': date_range, 'Value': data})

# Plot the time series data
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Value'], label='Time Series Data')
plt.title('Basic Time Series Line Plot')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.legend()
plt.show()
```

---

2. Time Series with Rolling Mean (Smoothing)

A rolling mean smooths out the time series to highlight trends by averaging values over a sliding window.

Example: Plot with Rolling Mean

python

Copy code

```
# Calculate the rolling mean
df['Rolling Mean'] = df['Value'].rolling(window=10).mean()

# Plot time series with rolling mean
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Value'], label='Original Data')
plt.plot(df['Date'], df['Rolling Mean'], label='Rolling Mean
(Window=10)', color='orange')
plt.title('Time Series with Rolling Mean')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.legend()
plt.show()
```

3. Time Series with Trend and Seasonality Decomposition

Seasonal decomposition allows you to break the time series into trend, seasonal, and residual components.

Example: Decomposition using `statsmodels`

python
Copy code

```
import statsmodels.api as sm

# Generate synthetic seasonal data
np.random.seed(0)
data_seasonal = np.sin(np.linspace(0, 2 * np.pi, 100)) + 0.1 *
np.random.randn(100)

df_seasonal = pd.DataFrame({'Date': date_range, 'Value':
data_seasonal})

# Decompose the time series
decomposition = sm.tsa.seasonal_decompose(df_seasonal['Value'],
period=30, model='additive')

# Plot the decomposition
decomposition.plot()
plt.show()
```

---

4. Autocorrelation Plot

Autocorrelation shows how the values of a time series are related to their previous values. An autocorrelation plot helps in identifying patterns like seasonality and trends.

Example: Autocorrelation Plot using `pandas`

python
Copy code

```
from pandas.plotting import autocorrelation_plot
```

```
# Autocorrelation plot
plt.figure(figsize=(10, 6))
autocorrelation_plot(df['Value'])
plt.title('Autocorrelation Plot')
plt.show()
```

---

5. Time Series Heatmap

A heatmap can represent time series data aggregated into different time intervals (e.g., by day, month) and visualize patterns.

Example: Time Series Heatmap using `seaborn`

python
Copy code
```
import seaborn as sns

# Create month-day data
df['Month'] = df['Date'].dt.month
df['Day'] = df['Date'].dt.day

# Pivot the data for heatmap
pivot_data = df.pivot('Day', 'Month', 'Value')

# Plot heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(pivot_data, cmap='coolwarm', annot=True, fmt='.1f')
plt.title('Time Series Heatmap')
plt.show()
```

---

6. Lag Plot

Lag plots are used to determine if a time series is random or if there is an underlying pattern. A lag plot compares the time series data with its lagged values.

Example: Lag Plot using `pandas`

python
Copy code

```python
from pandas.plotting import lag_plot

# Lag plot for time series
plt.figure(figsize=(6, 6))
lag_plot(df['Value'], lag=1)
plt.title('Lag Plot (Lag=1)')
plt.show()
```

---

7. Time Series with Confidence Intervals

A line plot with confidence intervals can show the uncertainty or variability in the data.

Example: Plot with Confidence Intervals

python
Copy code

```python
# Create upper and lower bounds (confidence intervals)
df['Upper Bound'] = df['Rolling Mean'] + 1.96 * df['Value'].std()
df['Lower Bound'] = df['Rolling Mean'] - 1.96 * df['Value'].std()

# Plot time series with confidence intervals
plt.figure(figsize=(10, 6))
plt.plot(df['Date'], df['Rolling Mean'], label='Rolling Mean',
color='orange')
plt.fill_between(df['Date'], df['Upper Bound'], df['Lower Bound'],
color='gray', alpha=0.3, label='Confidence Interval (95%)')
plt.title('Time Series with Confidence Intervals')
plt.xlabel('Date')
plt.ylabel('Value')
plt.grid(True)
plt.legend()
plt.show()
```

These examples show how to visualize time series data using various techniques in Python, helping you uncover patterns, trends, and underlying structures.