Time Series Data: Introduction

Glenn Bruns
CSUMB

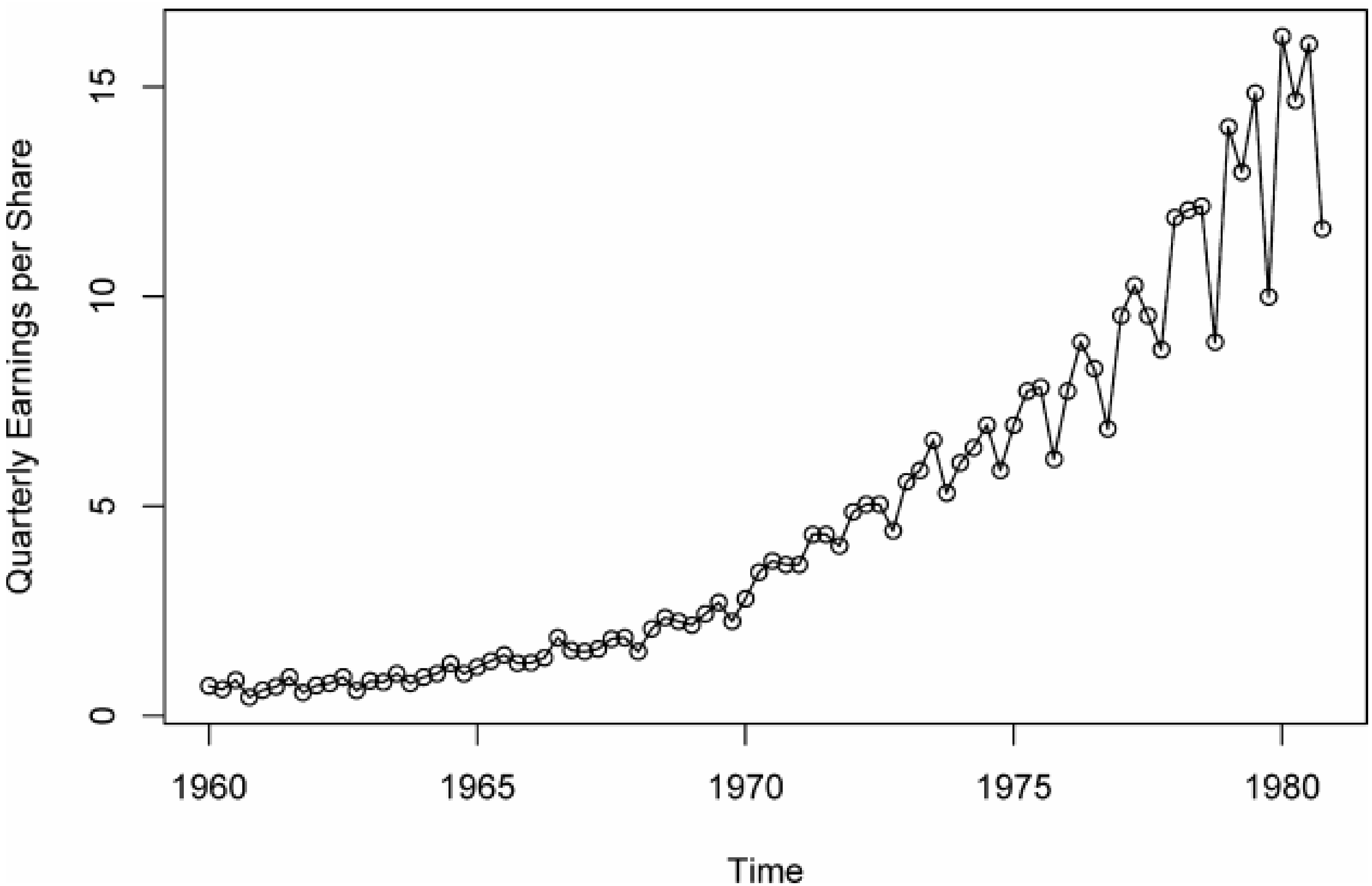
Learning outcomes

After this lecture you should be able to:

1. Define what 'time series data' means
2. Explain how to derive time series data from other kinds of data
3. List tasks associated with time series data
4. Explain time series decomposition

An example time series

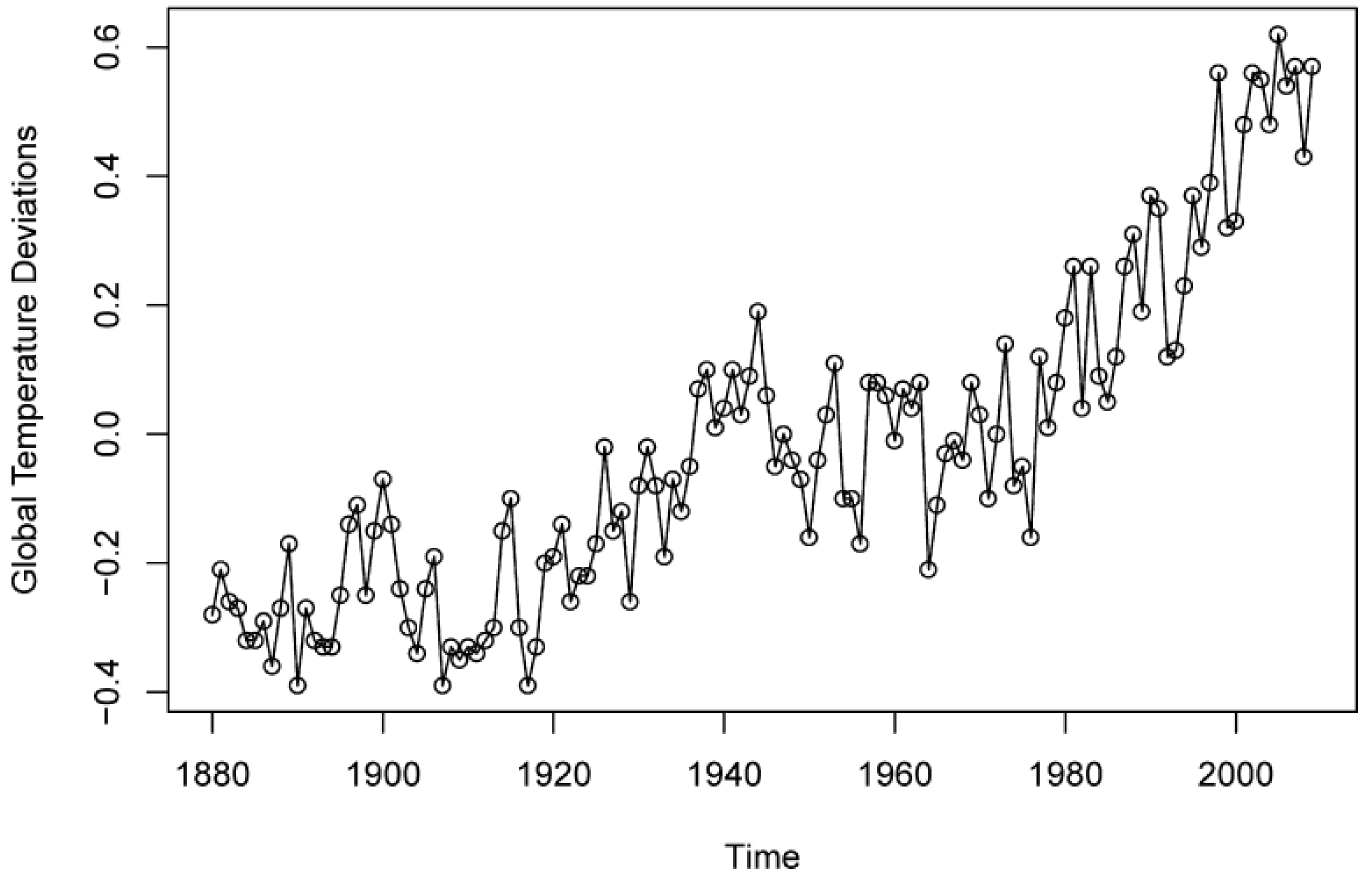
Quarterly earnings per share for Johnson & Johnson



source: Shumway & Stoffer: Timer Series Analysis and Applications

Global warming

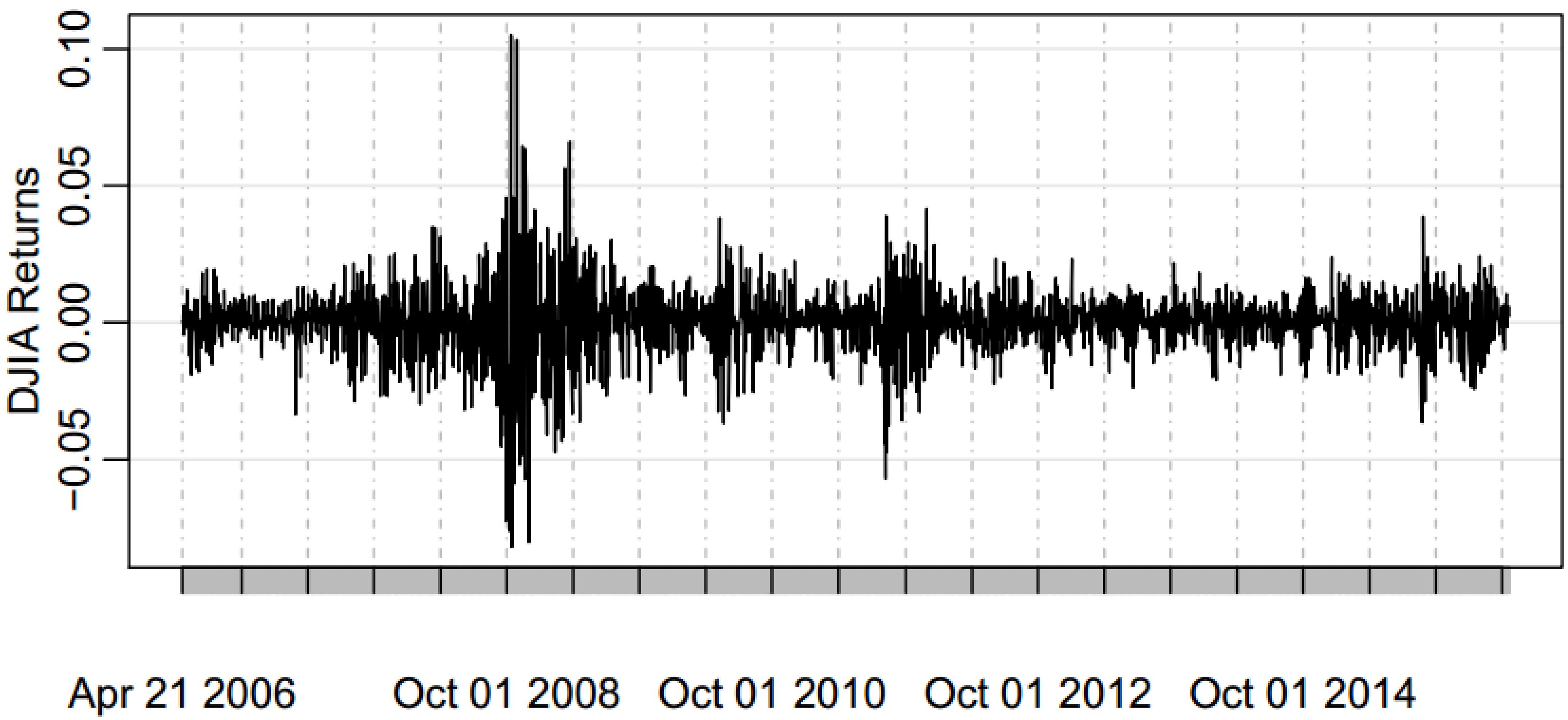
Deviations (°C) in the global mean land-ocean temperature index relative to the 1951-1980 average



source: Shumway & Stoffer: Timer Series Analysis and Applications

Stock market

Daily returns in the Dow Jones Industrial Average



source: Shumway & Stoffer: Timer Series Analysis and Applications

What is time series data?

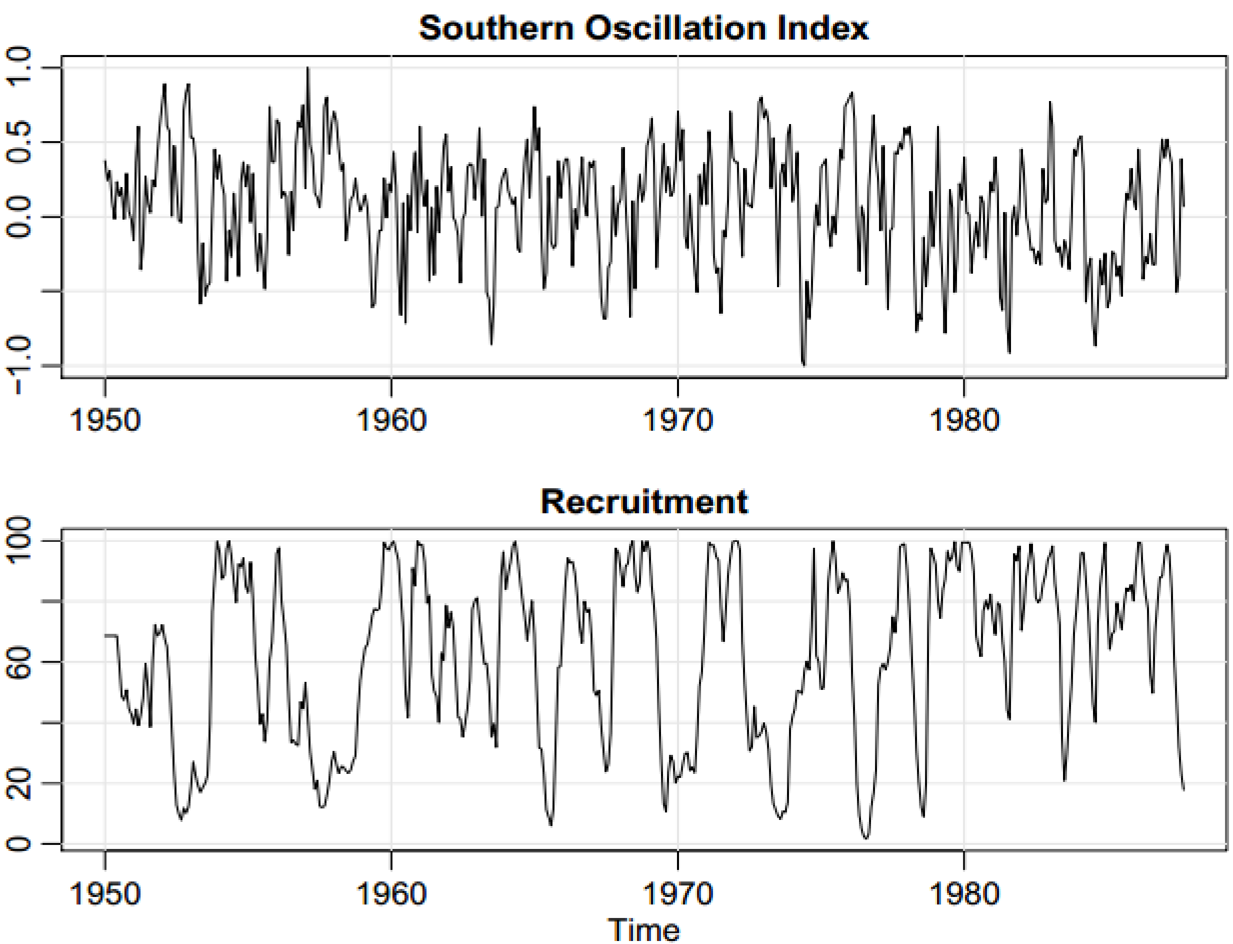
Time series: measurements of a variable at regular intervals over some period of time

The above definition is for univariate (single variable) time series.

We can also have multivariate time series

A multivariate time series

El Nino and number of new fish (monthly)



source: Shumway & Stoffer: Timer Series Analysis and Applications

Loading time series data

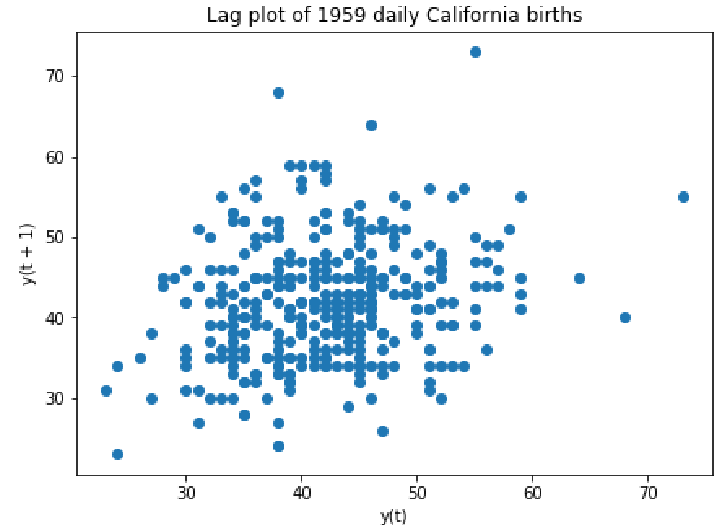
```
# load data
series = Series.from_csv("daily-total-female-births.csv",
header=0, parse_dates=[0], index_col=0, squeeze=True)

# explore
print(type(series))
print(series.head())
print(series.describe())

# query
print(series['1959-01'])
```

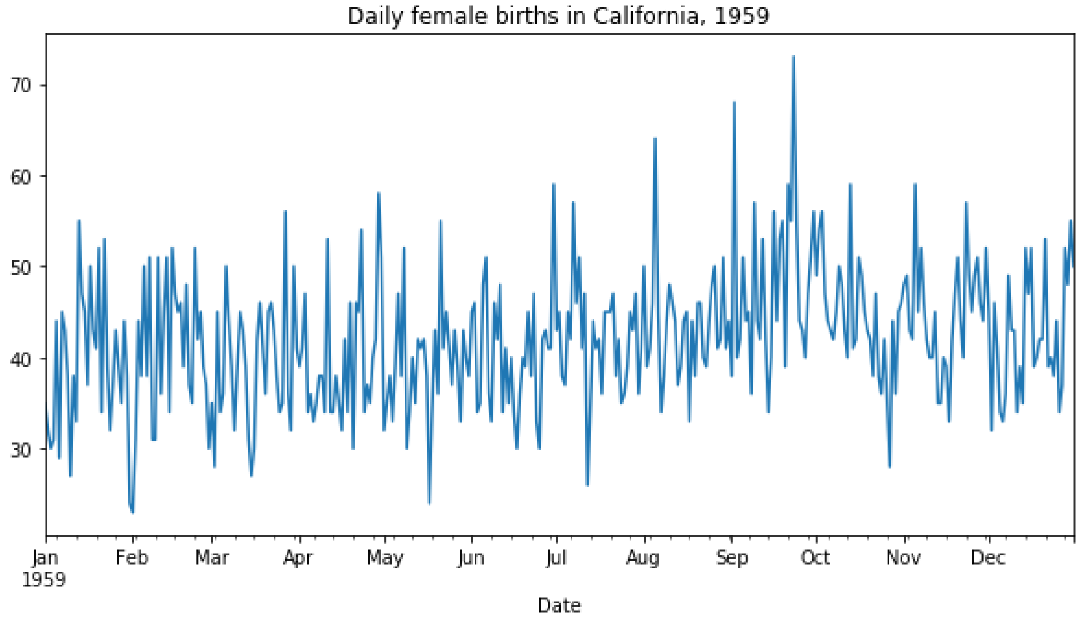
options in read_csv:

header=0	header is row 0
parse_dates=[0]	first column contains dates
index_col=0	first column is index information
squeeze=True	only one data column; use Pandas series

Plotting time series data

```python
import pandas as pd
from pandas.plotting import lag_plot

# line plot
series.plot()

# lag plot
lag_plot(series)
```





Aggregating to get a time series

- every so often a car passes by a point on the road

  → time series: hourly count of cares that pass by the point

car times
1:12 pm
1:22 pm
1:23 pm
1:45 pm
2:02 pm
2:14 pm
2:17 pm
2:50 pm
3:12 pm

time	count
1 pm	4
2 pm	5
3 pm	…

A lot of time series data is created through aggregation

Tasks with time series data

Forecasting: predict future values of a time series

- very popular! People like to know what will happen in the future ☺
- cool idea: train your stock market predictor with data from 2-5 years ago, test it on last year's data

Anomaly detection:

- find abnormal regions of a time series

Clustering

Pattern matching/similarity

Classification,…

Time series decomposition

It's often useful to break a time series down into:

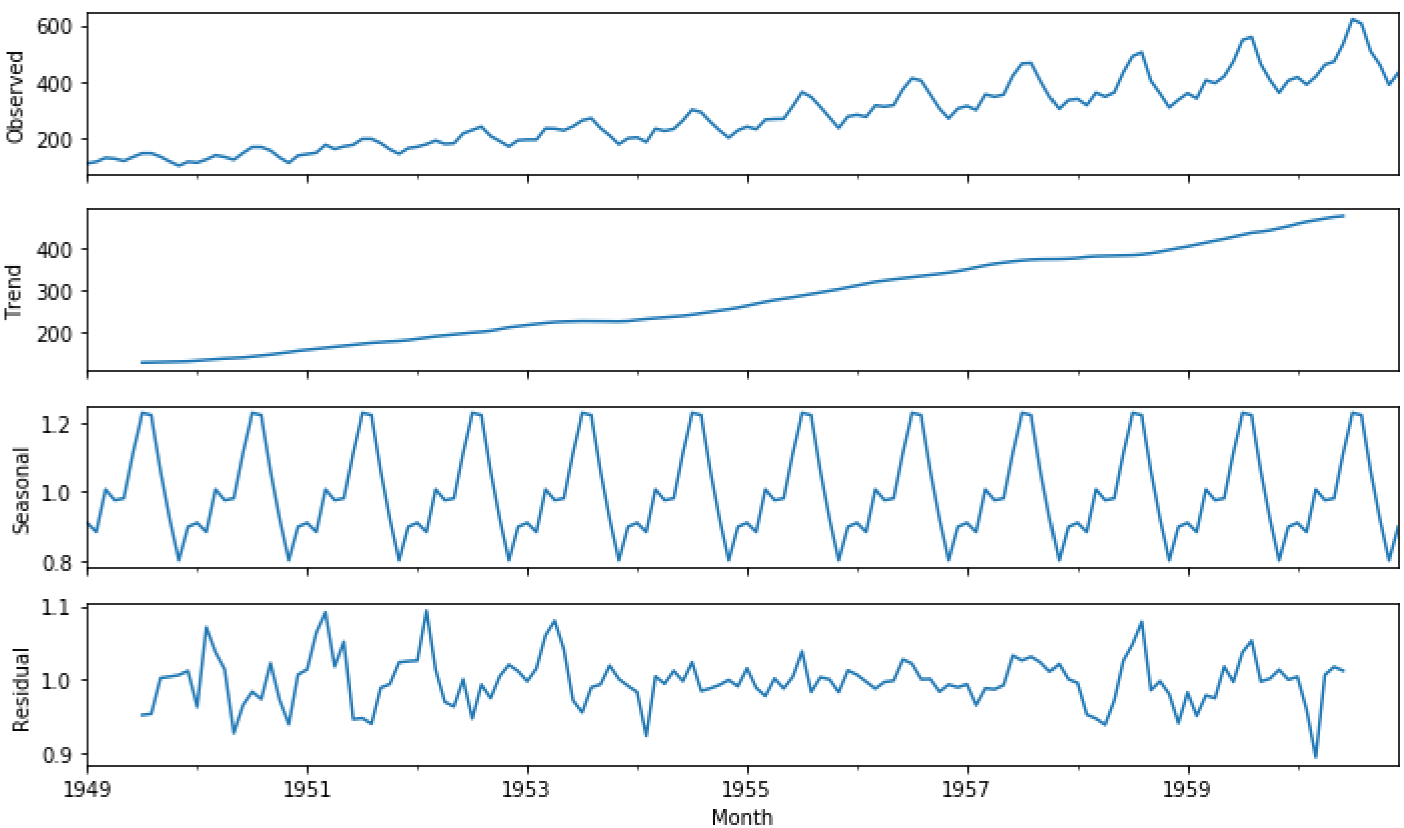
- a seasonal component (weekly, yearly, etc.)
- a trend
- noise

You get the original time series by either:

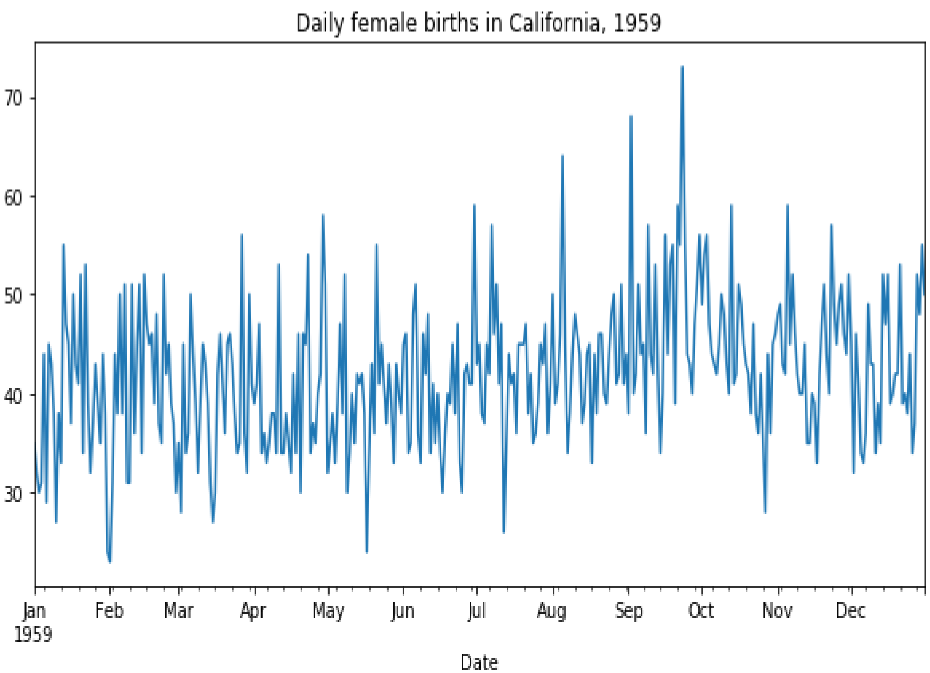
- adding the parts: $x_t = T_t + S_t + N_t$
- multiplying the parts: $x_t = T_t \cdot S_t + N_t$

Example: multiplicative decomposition

```
from statsmodels.tsa.seasonal import seasonal_decompose
series = Series.from_csv('intl-airline-passengers.csv', header=0)
result = seasonal_decompose(series, model='multiplicative')
        result.plot()
```



Identifying and removing trends



Identifying trends is useful.

Removing trends is common in traditional time series analysis.

To identify a trend:

- study a plot of the time series
- moving average
- model fitting

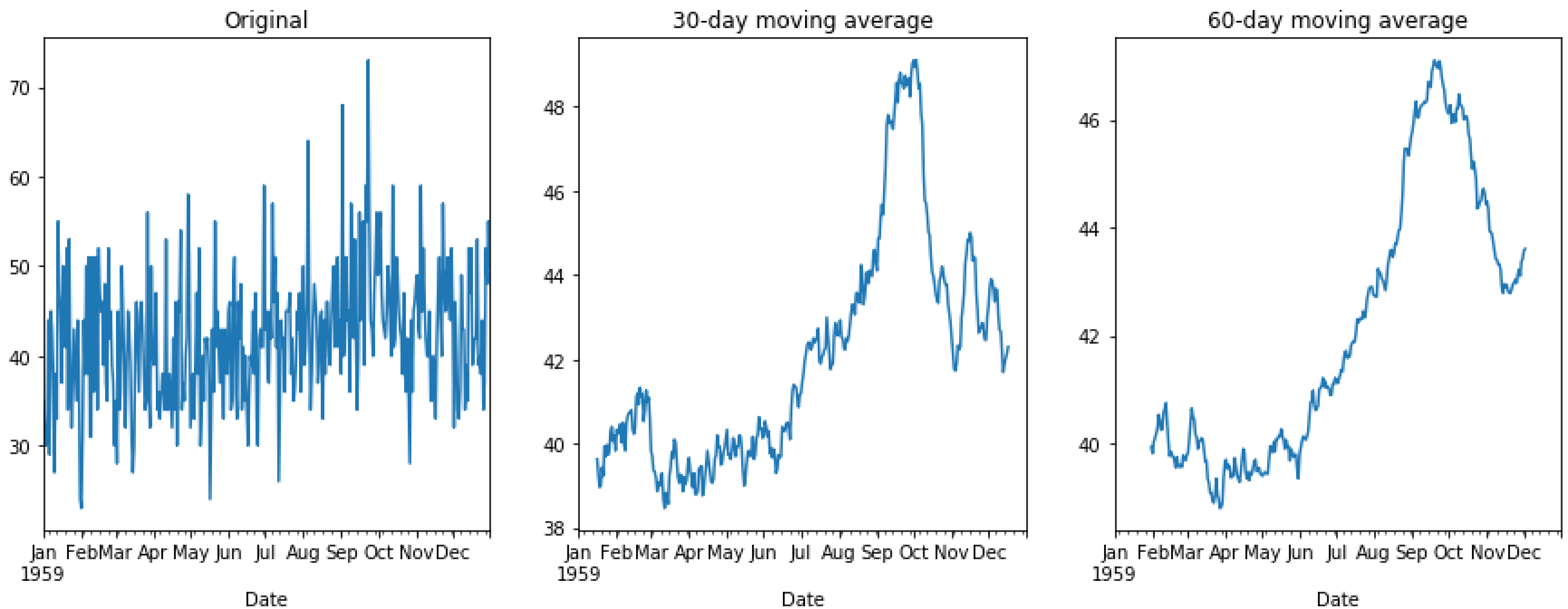
To remove a trend:

- differencing
- model fitting

Identifying trend with moving avg.

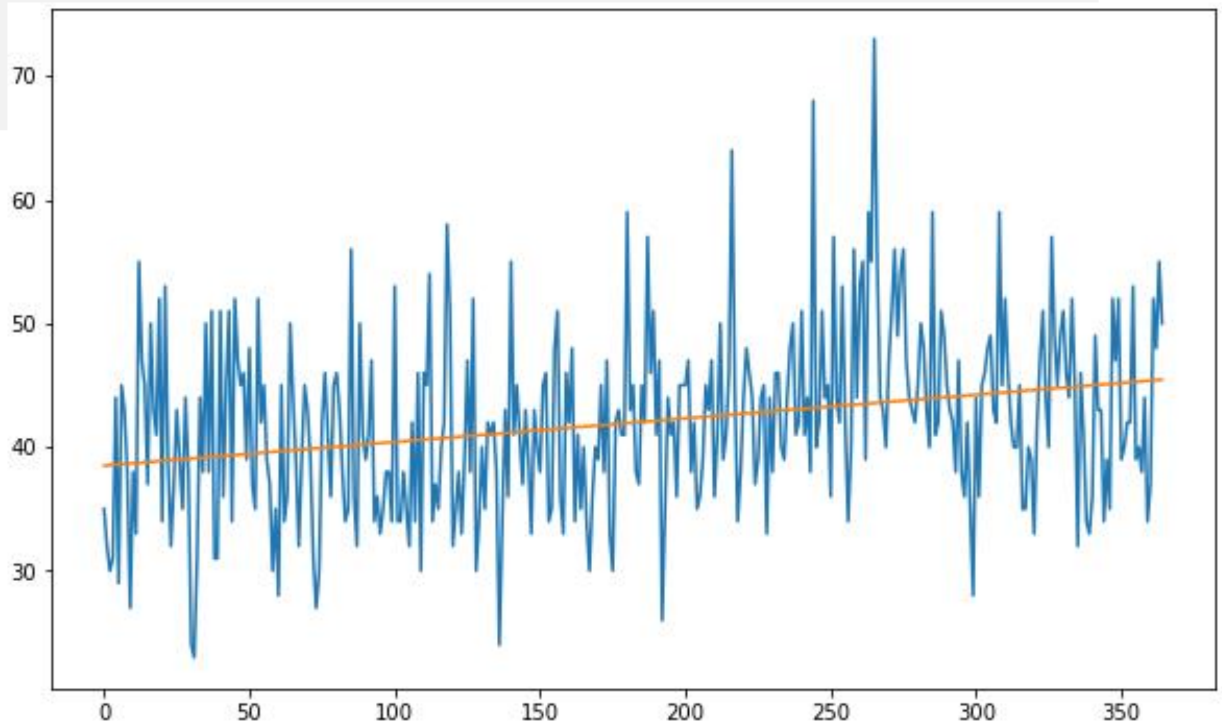
1 3 6 2 4 6 8 4 6 2 4 $\xrightarrow{\text{window=3}}$ - 10/3 11/3 4 4 6 6 4 4 -

```
series_ma_30 = series.rolling(window=30, center=True).mean()
series_ma_60 = series.rolling(window=60, center=True).mean()
```



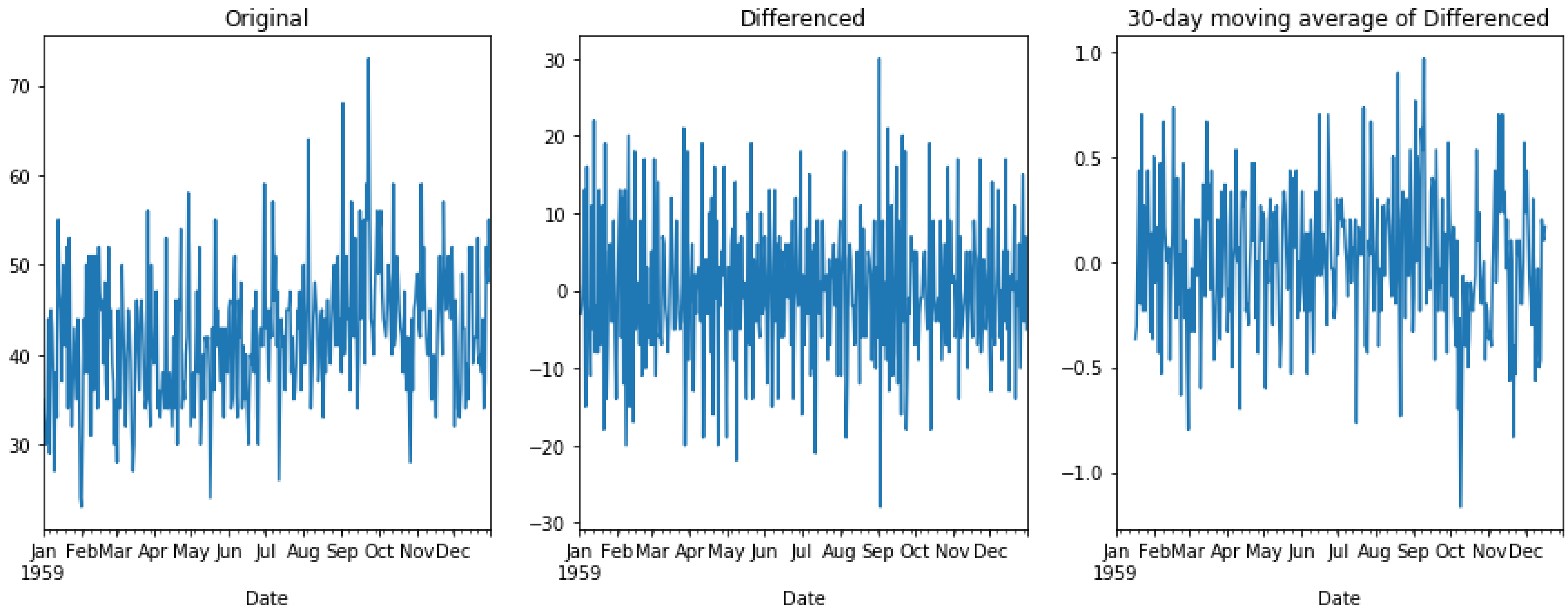
Identifying trend with model fitting

```
x = np.arange(len(series), dtype='int64').reshape(-1,1)
y = series.values.reshape(-1,1)
regr = LinearRegression()
regr.fit(x,y)
trend = regr.predict(x)
print(regr.score(x,y))        // R^2 score is 0.08

plt.plot(y)
plt.plot(trend)
```



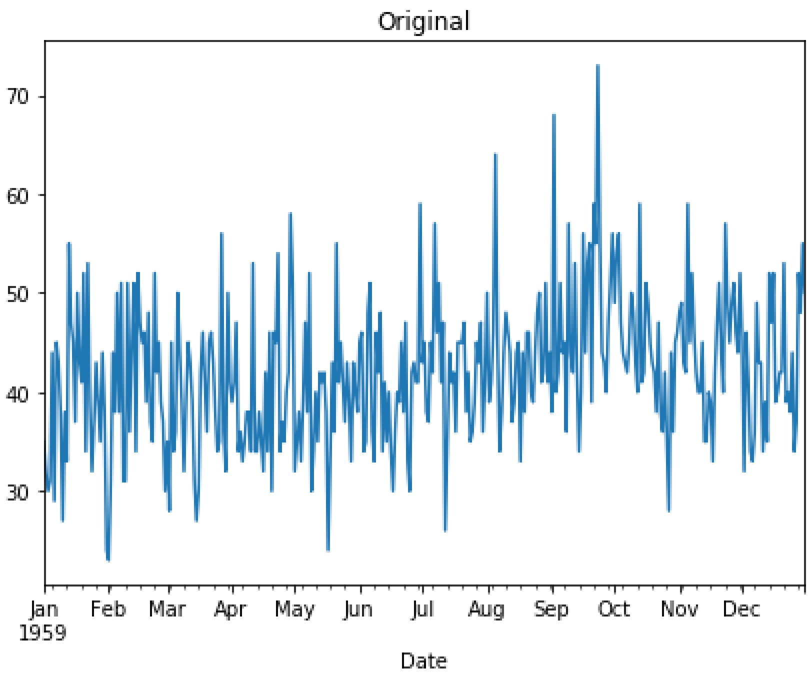
Removing trend with differencing

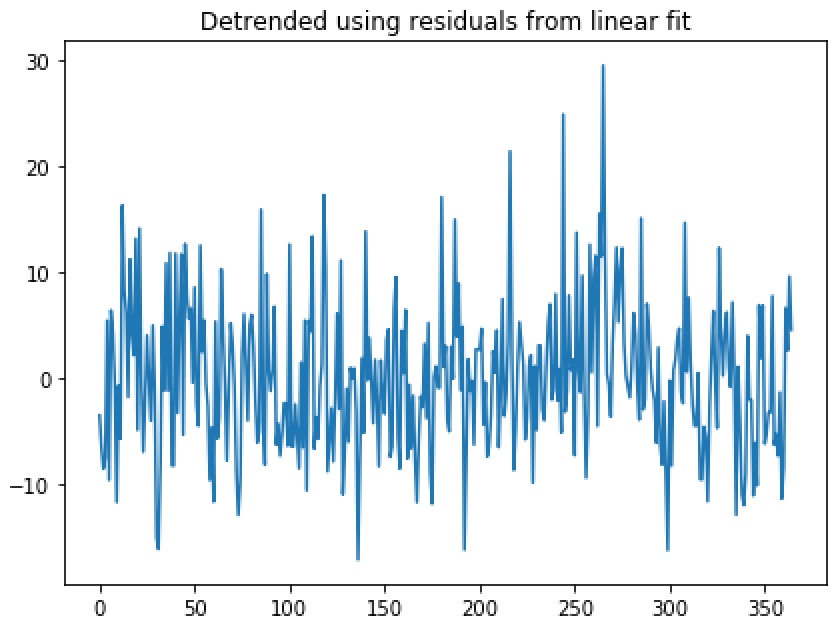
```
diff = series.diff()
diff_ma = diff.rolling(window=30, center=True).mean()
```



Removing trend with model

```
regr = LinearRegression()
regr.fit(x,y)
trend = regr.predict(x)
detrended = y - trend
```





Identifying and removing seasonality

Identifying seasonality is useful – can make relationships clearer.

Removing seasonality is common in traditional time series analysis.

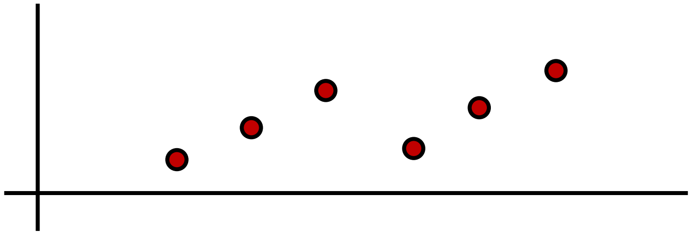
To identify seasonality:

- study a plot of the time series
- use periodicity detection algorithms; spectral analysis

To remove seasonality:

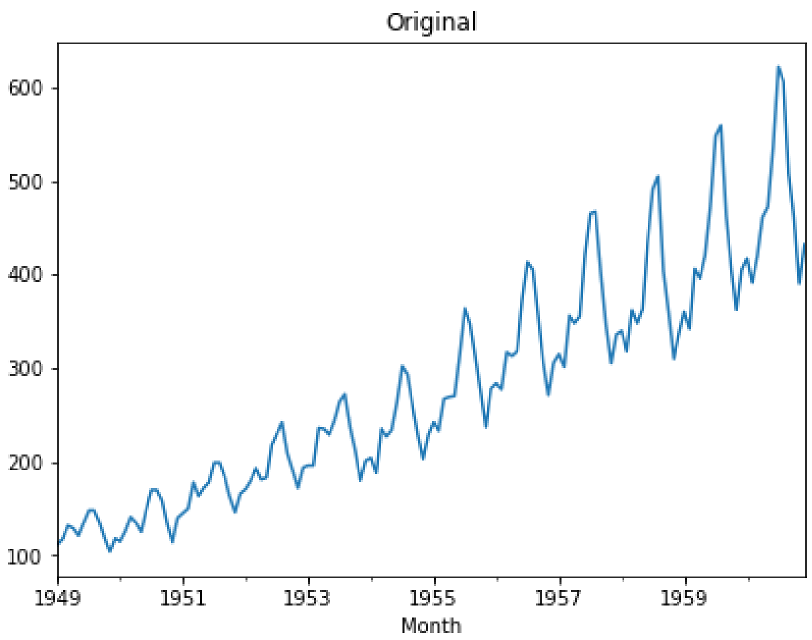
- differencing on the period
- aggregate data to the appropriate level
- use model fitting

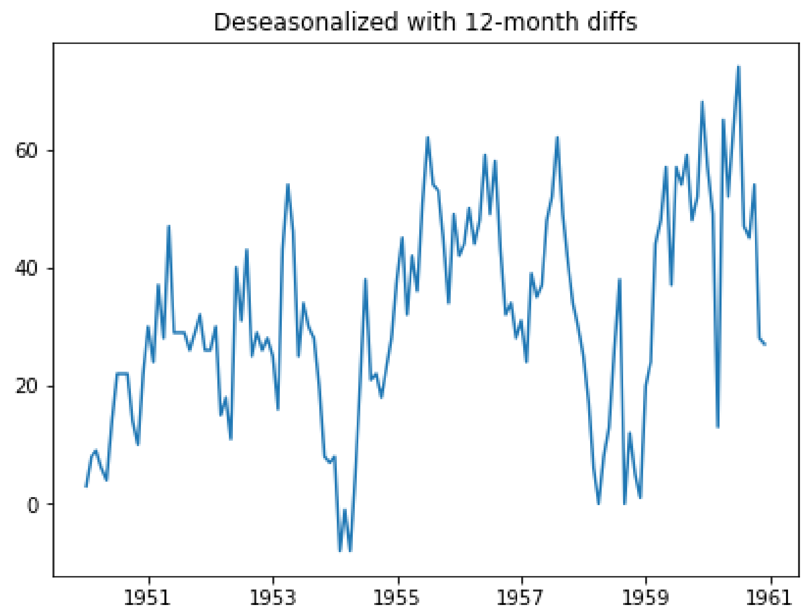
“Deseasonalizing” with differencing



period is length 3, so subtract by value 3 time units in past

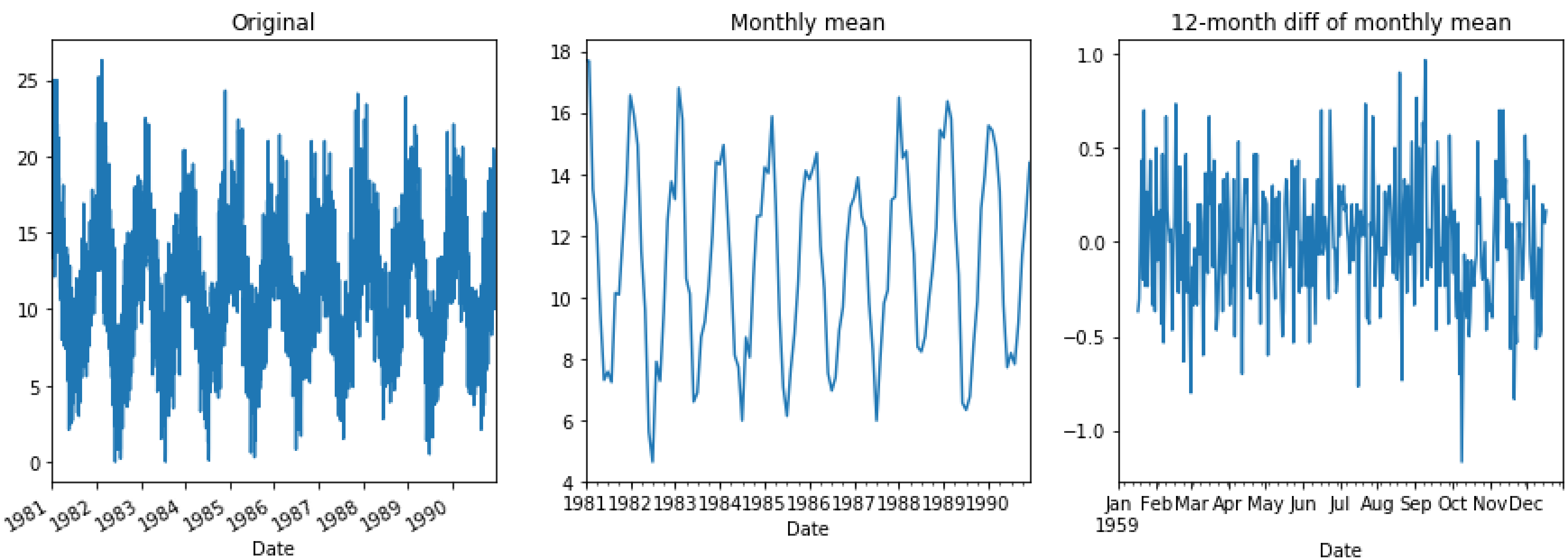
```
# monthly international airline passengers data
months_per_year = 12
diff = series.diff(months_per_year)      # pandas
```





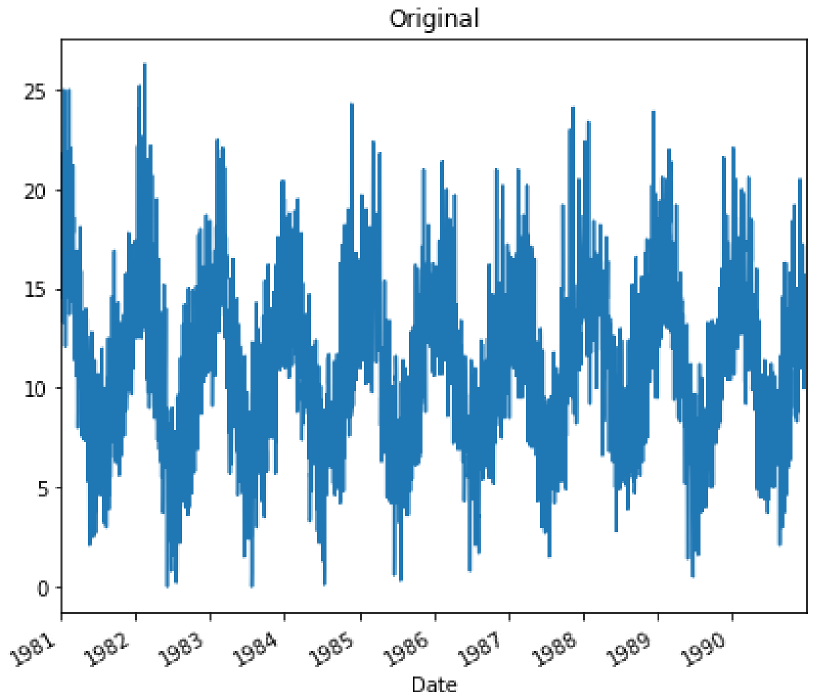
Resampling plus differencing

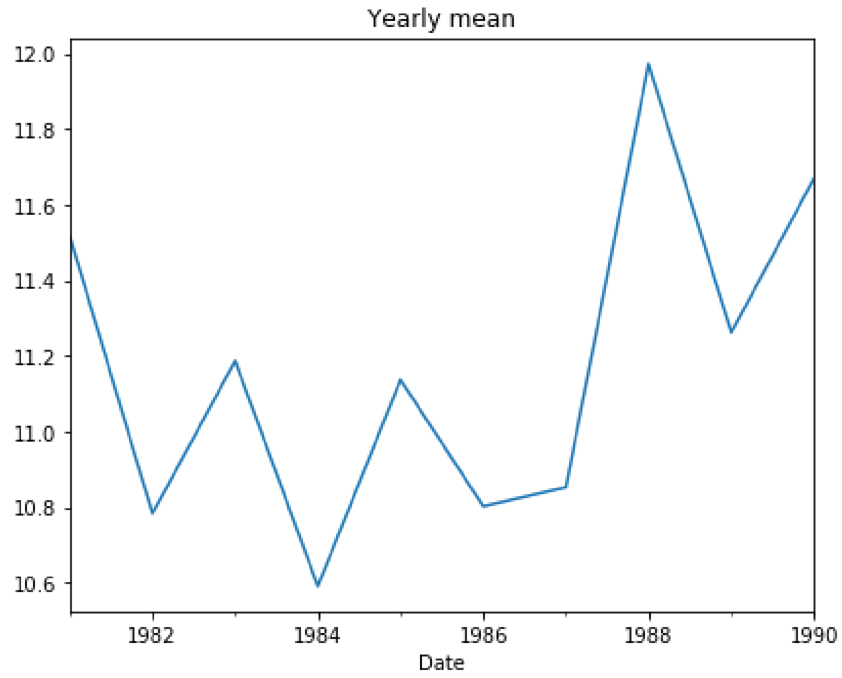
```
# minimum daily temperatures data, Melbourne, Australia
resample = series.resample('M' )
monthly_mean = resample.mean()

# take the diffs
diff = monthly_mean.diff(months_per_year)
```



Deasonalizing with aggregation

```python
# minimum daily temperatures data, Melbourne, Australia
resample = series.resample('A')   # "annual", year end
yearly_mean = resample.mean()
```





Summary

- examples of time series data
- definition of time series data
- how to derive time series data
- tasks for time series data
- time series decomposition
- removing trend and seasonality