




 [learn-co-curriculum](#) / [dsc-basic-time-series-models-lab](#) Public [View license](#) 1 star  161 forks Star Watch ▾

<> Code

 Issues Pull requests Actions Projects Security Insights solution ▾

...

This branch is [3 commits ahead](#), [4 commits behind](#) master.

Cheffrey2000 fixed date error ...

on Nov 18, 2021

 8[View code](#) README.md

Basic Time Series Models - Lab

Introduction

Now that you have some basic understanding of the white noise and random walk models, its time for you to implement them!

Objectives

In this lab you will:

- Generate and analyze a white noise model
- Generate and analyze a random walk model
- Implement differencing in a random walk model

A White Noise model

To get a good sense of how a model works, it is always a good idea to generate a process. Let's consider the following example:

- Every day in August, September, and October of 2018, Nina takes the subway to work. Let's ignore weekends for now and assume that Nina works every day
- We know that on average, it takes her 25 minutes, and the standard deviation is 4 minutes
- Create and visualize a time series that reflects this information

Let's import `pandas`, `numpy`, and `matplotlib.pyplot` using their standard alias.

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

# Do not change this seed
np.random.seed(12)
```

Create the dates. You can do this using the `date_range()` function Pandas. More info [here](#).

```
dates = pd.date_range('2018-08-01', '2018-10-31', freq='B')
len(dates)
```

65

Generate the values for the white noise process representing Nina's commute in August and September.

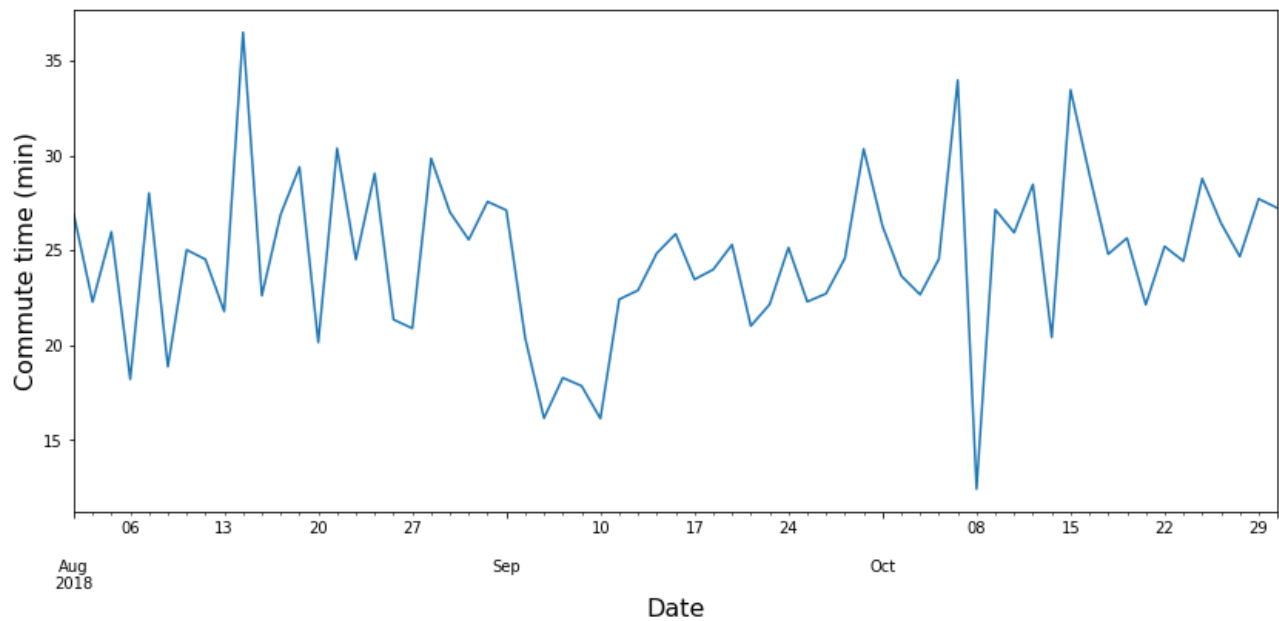
```
commute = np.random.normal(25, 4, len(dates))
```

Create a time series with the dates and the commute times.

```
commute_series = pd.Series(commute, index=dates)
```

Visualize the time series and set appropriate axis labels.

```
ax = commute_series.plot(figsize=(14,6))  
ax.set_ylabel('Commute time (min)', fontsize=16)  
ax.set_xlabel('Date', fontsize=16)  
plt.show()
```



Print Nina's shortest and longest commute.

```
min(commute)
```

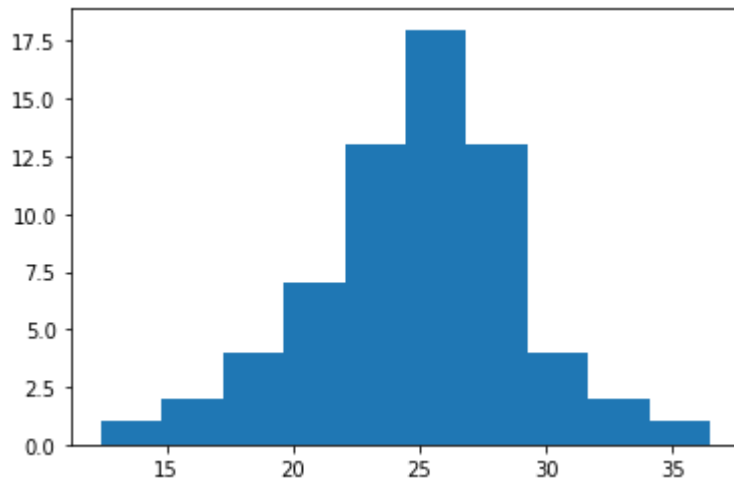
```
12.41033391382408
```

```
max(commute)
```

```
36.487277579955666
```

Look at the distribution of commute times.

```
commute_series.hist(grid=False);
```



Compute the mean and standard deviation of `commute_series`. The fact that the mean and standard error are constant over time is crucial!

```
np.mean(commute_series)
```

```
24.62933183905801
```

```
np.std(commute_series)
```

```
4.2480368492867395
```

Now, let's look at the mean and standard deviation for August and October.

```
aug_series = commute_series['08-2018']  
oct_series = commute_series['10-2018']  
print('August: ', np.mean(aug_series), np.std(aug_series), '\n')  
print('October: ', np.mean(oct_series), np.std(oct_series))
```

```
August:  25.35433780425335  4.206438796880027
```

```
October:  25.677914014894995  4.273672352538592
```

Because you've generated this data, you know that the mean and standard deviation will be the same over time. However, comparing mean and standard deviation over time is useful practice for real data examples to check if a process is white noise!

A Random Walk model

Recall that a random walk model has:

- No specified mean or variance
- A strong dependence over time

Mathematically, this can be written as:

$$Y_t = Y_{t-1} + \epsilon_t$$

Because today's value depends on yesterday's, you need a starting value when you start off your time series. In practice, this is what the first few time series values look like: $Y_0 = \text{some specified starting value}$

$$Y_1 = Y_0 + \epsilon_1$$

$$Y_2 = Y_1 + \epsilon_2 = Y_0 + \epsilon_1 + \epsilon_2$$

$$Y_3 = Y_2 + \epsilon_3 = Y_0 + \epsilon_1 + \epsilon_2 + \epsilon_3$$

...

Keeping this in mind, let's create a random walk model:

Starting from a value of 1000 USD of a share value upon a company's first IPO (initial public offering) on January 1 2010 until end of November of the same year, generate a random walk model with a white noise error term, which has a standard deviation of 10.

```
# Keep the random seed
np.random.seed(11)

# Create a series with the specified dates
dates = pd.date_range('2010-01-01', '2010-11-30', freq='B')

# White noise error term
error = np.random.normal(0, 10, len(dates))

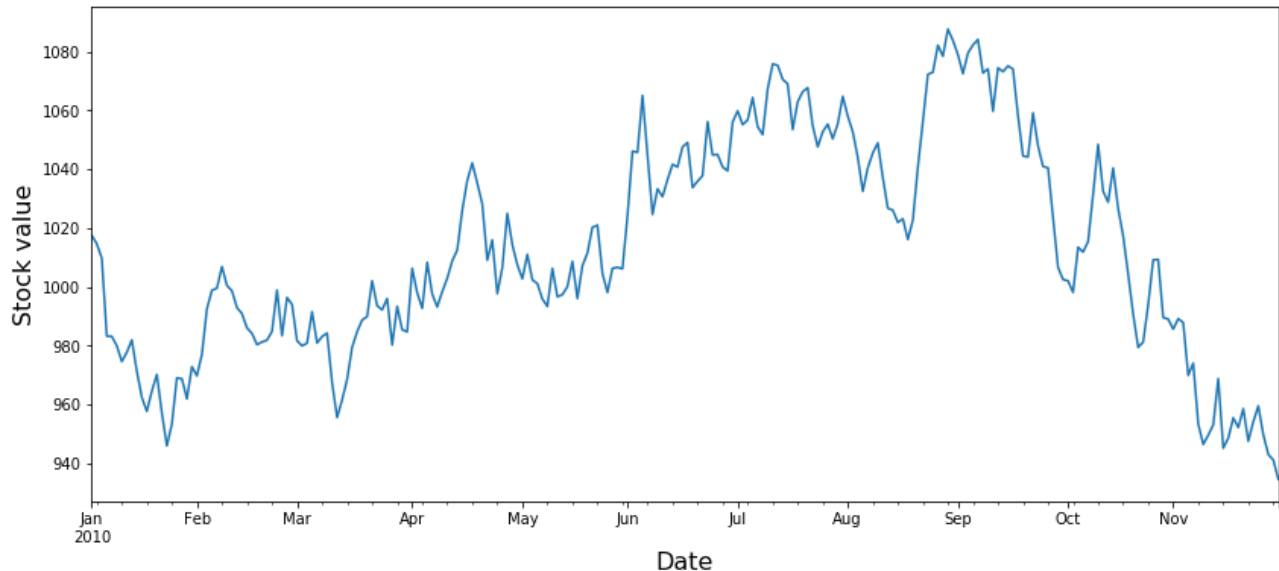
# Define random walk
def random_walk(start, error):
    Y_0 = start
    cum_error = np.cumsum(error)
    Y = cum_error + Y_0
    return Y

shares_value = random_walk(1000, error)

shares_series = pd.Series(shares_value, index=dates)
```

Visualize the time series with correct axis labels.

```
ax = shares_series.plot(figsize=(14,6))
ax.set_ylabel('Stock value', fontsize=16)
ax.set_xlabel('Date', fontsize=16)
plt.show()
```



You can see how this very much looks like the exchange rate series you looked at in the lesson!

Random Walk with a Drift

Repeat the above, but include a drift parameter c of 8 now!

```
# Keep the random seed
np.random.seed(11)

# Create a series with the specified dates
dates = pd.date_range('2010-01-01', '2010-11-30', freq='B')

# White noise error term
error = np.random.normal(0, 10, len(dates))

# Define random walk
def random_walk(start, error):
    Y_0 = start
    cum_error = np.cumsum(error + 8)
    Y = cum_error + Y_0
    return Y
```

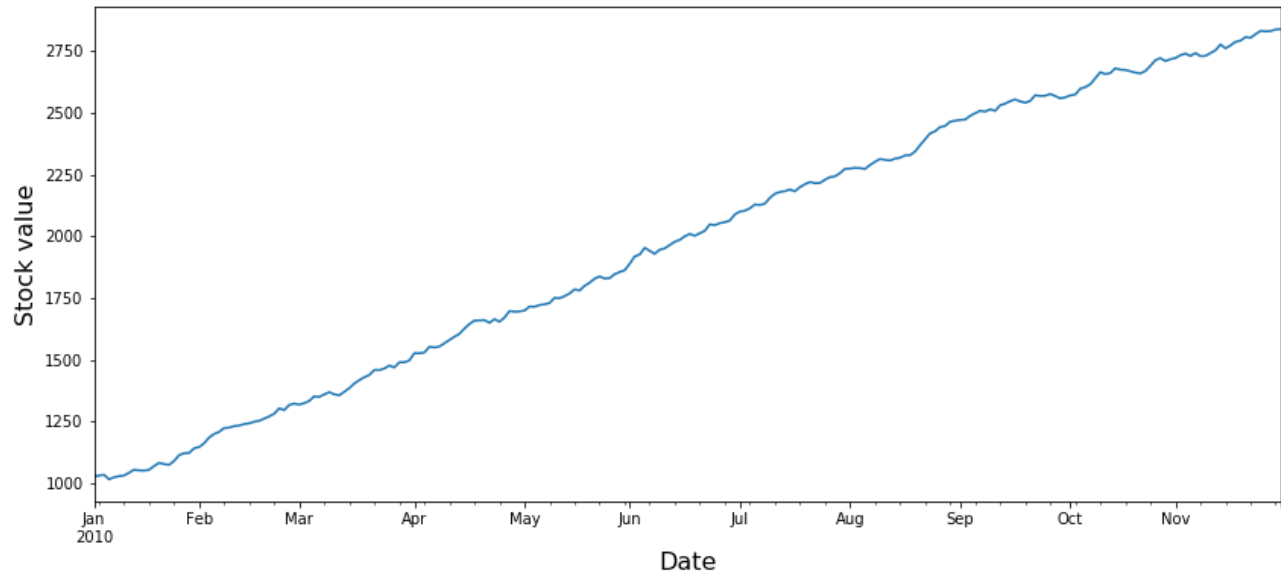
```

shares_value_drift = random_walk(1000, error)

shares_series_drift = pd.Series(shares_value_drift, index=dates)

ax = shares_series_drift.plot(figsize=(14,6))
ax.set_ylabel('Stock value', fontsize=16)
ax.set_xlabel('Date', fontsize=16)
plt.show()

```



Note that there is a very strong drift here!

Differencing in a Random Walk model

One important property of the random walk model is that a differenced random walk returns a white noise. This is a result of the mathematical formula:

$$Y_t = Y_{t-1} + \epsilon_t$$

and we know that ϵ_t is a mean-zero white noise process!

Plot the differenced time series (time period of 1) for the shares time series (no drift).

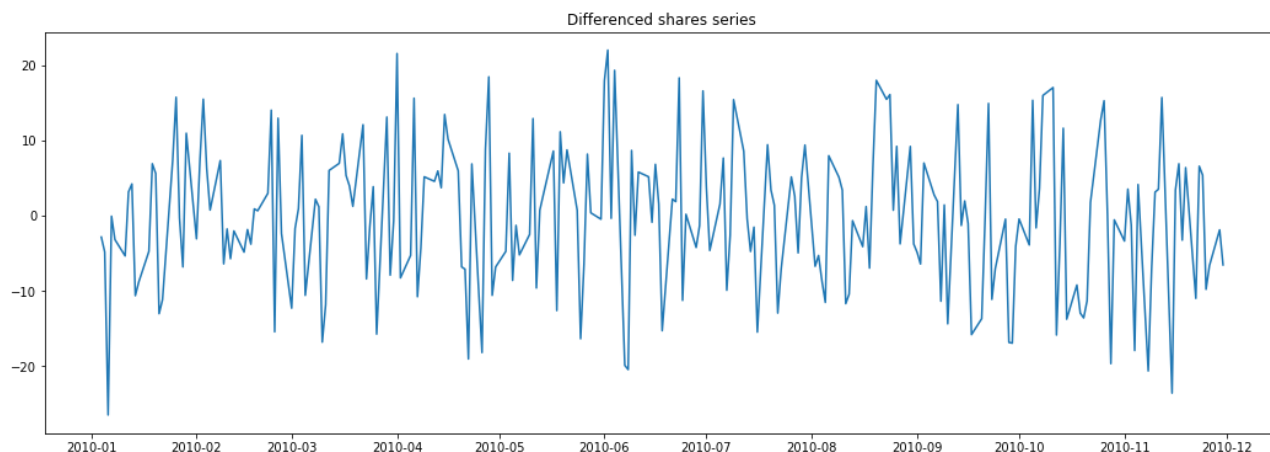
```

shares_diff = shares_series.diff(periods=1)

fig = plt.figure(figsize=(18,6))
plt.plot(shares_diff)

```

```
plt.title('Differenced shares series')
plt.show(block=False)
```

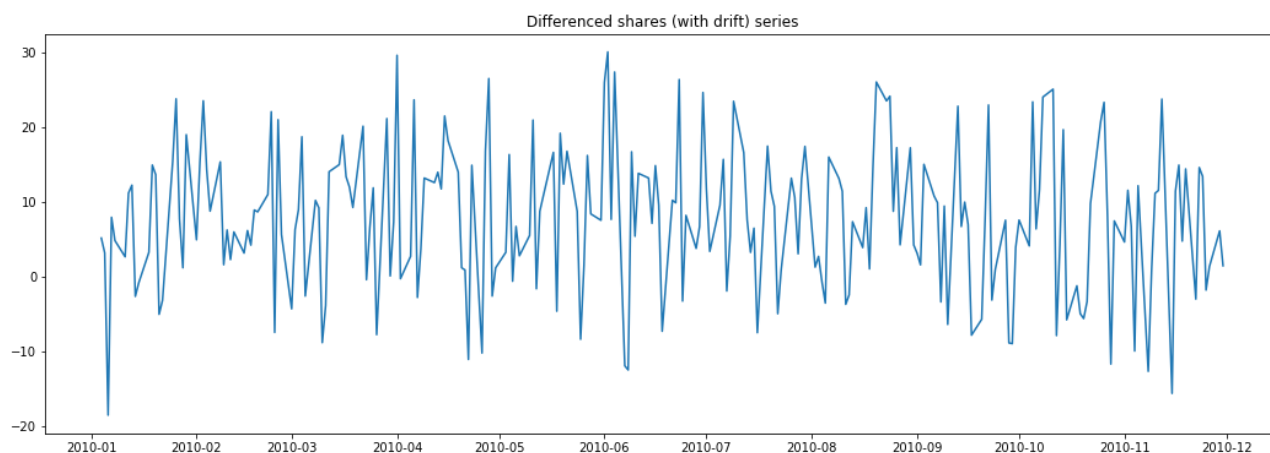


This does look a lot like a white noise series!

Plot the differenced time series for the shares time series (with a drift).

```
shares_drift_diff = shares_series_drift.diff(periods=1)

fig = plt.figure(figsize=(18,6))
plt.plot(shares_drift_diff)
plt.title('Differenced shares (with drift) series')
plt.show(block=False)
```



This is also a white noise series, but what can you tell about the mean?

The mean is equal to the drift c , so 8 for this example!

Summary

Great, you now know how white noise and random walk models work!

Releases

No releases published

Packages

No packages published

Contributors 3



LoreDirick Lore Dirick



sumedh10 Sumedh Panchadhar



Cheffrey2000 Jeffrey Hinkle

Languages

● **Jupyter Notebook** 100.0%