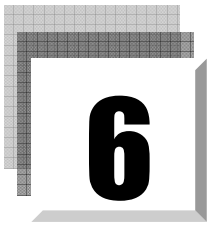


Bab 6 Tool-tool Untuk Membuat Grafik

Bab ini membahas tool-tool untuk visualisasi data.

6.1 Pengantar

Visualisasi data bertujuan untuk mengerti data melalui ektraksi informasi dan grafik untuk menunjukkan pola, tren, dan mengidentifikasi data pencilan. Ada dua dua jenis dasar visualisasi data yaitu *Eksploration* yang ditujukan untuk mengekstrask informasi dari data dan *Explanation* untuk menunjukkan informasi yang diekstrak.

Ada banyak jenis visualisasi data 2D, seperti temporal, multideimensional, hierarchical, dan network. Pada bagian ini akan dibahas beberapa teknik visualisasi data yang disediakan dalam bahasa pemrograman python.

Memuat Library

Beberapa library sudah dikemas dalam Python, namun ada juga yang lainnya yang harus di-download dan di-install sendiri. Sebagai contoh untuk menginstall library `maplotlib` dapat digunakan perintah pip sebagai berikut:

```
python -m pip install -U pip setuptools
python -m pip install matplotlib
```

Library untuk Python dapat di*install*, dicari, dan di*update* menggunakan Jupyter Notebook atau dengan *desktop Python IDE*

sperti *Spyder*. Table 2-2 menunjukkan bagaimana menggunakan perintah *pip* dan *conda*.

Table 2-2. Instalasi dan pembaruan Python Packages

Descripsi	pip	conda Anaconda
Berjalan pada	Python and Anaconda	Anaconda only
Pencarian *package*	`pip search matplolib`	`conda search matplolib`
Instalasi *package*	`pip install matplolib`	`conda install matplolib`
Upgrade package	`pip install matplolib-upgrade`	`conda install matplolib-upgrade`
Menampilkan *package* terinstall	`pip list`	`conda list`

Instalasi dan pembaruan package juga dapat dilakukan di Jupyter Notebook seperti ditampilkan dalam kode berikut.

```
In [5]: try:
            import matplotlib
            except:
                import pip pip.main(['install', 'matplotlib'])
                import matplotlib
```

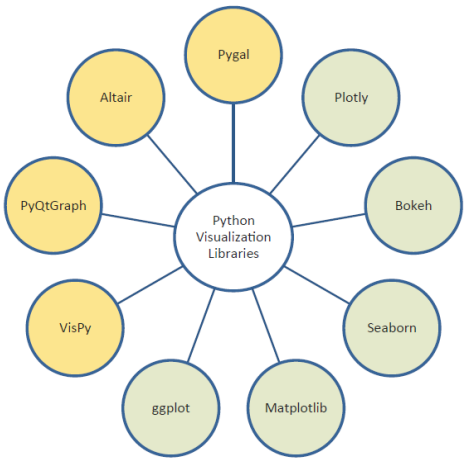
Modul yang diimport dapat menggunakan nama alias seperti daalam kode berikut:

```
In [ ]: import matplotlib.pyplot as plt import numpy as np
        import pandas as pd
        import seaborn as sns
        import pygal from mayavi
        import mlab
```

Sekali diimport dengan nama alias maka nama alias ini dapat digunakan pada kode setelah baris di atas.

6. 2 Library Populer *Data Visualization*

Python menyediakan banyak sekalai library untuk visualisasi data dan untuk *plotting*-nya. Diantaranya adalah Pygal, Altair, VisPy, PyQtGraph, Matplotlib, Bokeh, Seaborn, Plotly, dan ggplot, seperti ditunjukkan dalam Gambar 6-1.



Gambar 6.1 Library untuk visualisasi data dalam Python.

Masing-masing library memiliki fitur sendiri-sendiri. Beberapa library untuk diimplementasikan mungkin tergantung adanya library lain. Contohnya *Seaborn* merupakan library untuk visualisasi data statistik yang menggunakan *Matplotlib*. Library ini juga membutuhkan *Panda* dan mungkin *NumPy* untuk pemrosesan statistik sebelum memvisualisasi data.

Matplotlib

Matplotlib adalah library python untuk *plotting* 2D untuk visualisasi data data yang berbasis array *Numpy* dan didesain untuk bekerja dalam paket yang lebih besar yaitu *SciPy*. Library ini menghasilkan grafik berkualitas siap publikasi dalam berbagai format dan dalam lingkungan interaktif pada banyak platform. Terdapat dua opsi untuk menempelkan grafik secara langsung dalam sebuah notebook, yaitu :

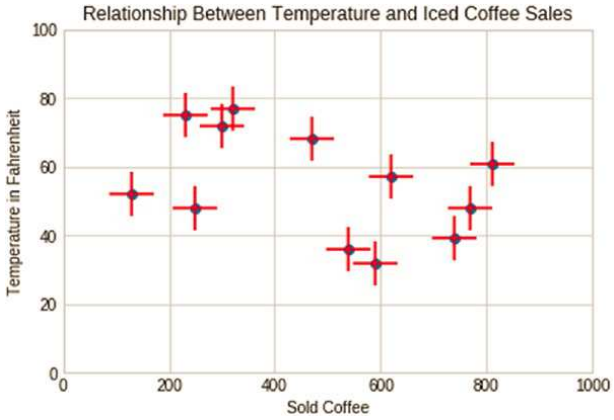
• perintah `%matplotlib notebook` akan menghasilkan gambar grafik *interaktif* yang menempel dalam notebook.
• perintah `%matplotlib inline` akan menghasilkan gambar grafik *statis* dalam notebook.

Listing 6-2 mem-plot data menggunakan Matplotlib dan menyesuaikan atribut plot.

Listing 6-2 Contoh mengimport library *Matplotlib*.

```
In [12]:import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.style.use('seaborn-whitegrid')
X = [590,540,740,130,810,300,320,230,470,620,770,250]
Y = [32,36,39,52,61,72,77,75,68,57,48,48]

plt.scatter(X,Y)
plt.xlim(0,1000)
plt.ylim(0,100)

#scatter plot color
plt.scatter(X, Y, s=60, c='red', marker='^')

#change axes ranges
plt.xlim(0,1000)
plt.ylim(0,100)

#add title
plt.title('Relationship Between Temperature and Iced Coffee Sales')

#add x and y labels
plt.xlabel('Sold Coffee')
plt.ylabel('Temperature in Fahrenheit')

#show plot
plt.show()
```

Gambar berikut menampilkan hasil visualisasi dari kode di atas.

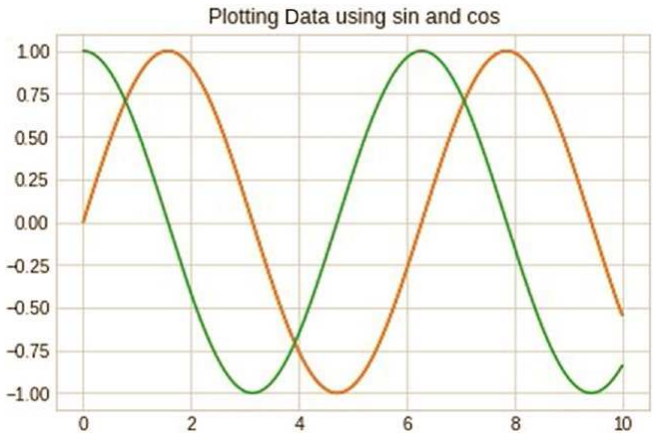


Gambar 6.2. Visualisasi data menggunakan *Matplotlib*.

Kode berikut mem-plot data menggunakan Matplotlib dan menyesuaikan atribut plot.

Listing 6-3 Plot atribut terakumulasi

```
In [20]:%matplotlib inline
        import matplotlib.pyplot as plt
        import numpy as np

plt.style.use('seaborn-whitegrid')

# Create empty figure
fig = plt.figure()
ax = plt.axes()
x = np.linspace(0, 10, 1000)
ax.plot(x, np.sin(x));
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))

# set the x and y axis range
plt.xlim(0, 11)
plt.ylim(-2, 2)
plt.axis('tight')

#add title
plt.title('Plotting data using sin and cos')
```



Gambar 6.3 Grafik fungsi sin dan cos dengan Matplotlib.

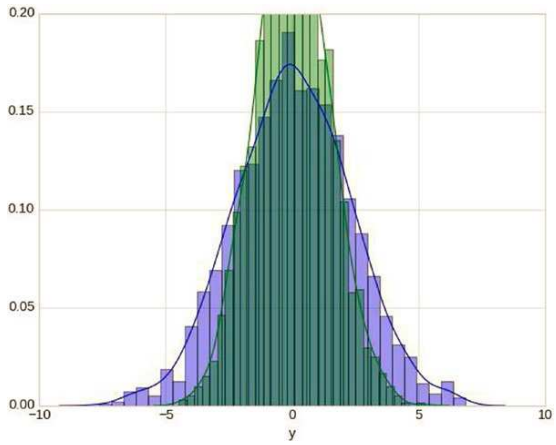
Terdapat format plot yang berbeda yang dapat dibuat dengan Matplotlib. Beberapa format akan dibahas dalam bab selanjutnya.

Seaborn

Seaborn adalah library visualisasi data Python berdasarkan Matplotlib yang menyediakan antarmuka tingkat tinggi untuk menggambar grafik statistik yang menarik dan informatif (lihat Listing 6-4).

Listing 6-4. Cara import dan penggunaan library Seaborn

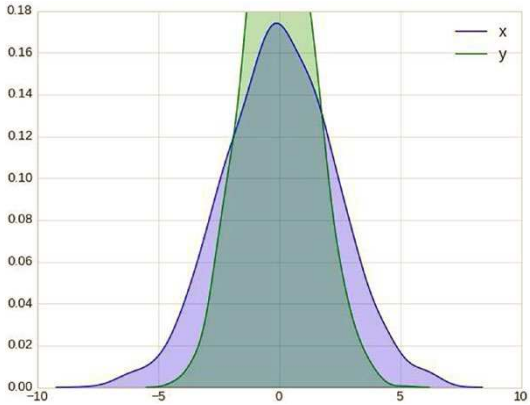
```
In [34]: import matplotlib.pyplot as plt
%matplotlib inline
import numpy as np
import pandas as pd
import seaborn as sns

plt.style.use('classic')
plt.style.use('seaborn-whitegrid')

# Create some data
data = np.random.multivariate_normal([0, 0], [[5, 2], [2, 2]],
size=2000)
data = pd.DataFrame(data, columns=['x', 'y'])

# Plot the data with seaborn
sns.distplot(data['x'])
sns.distplot(data['y']);
```



Gambar 6.4 Grafik menggunakan library seaborn

Data di atas dapat digunakan untuk memplot grafik yang berbeda, misalnya grafik *kernel density entimation*.

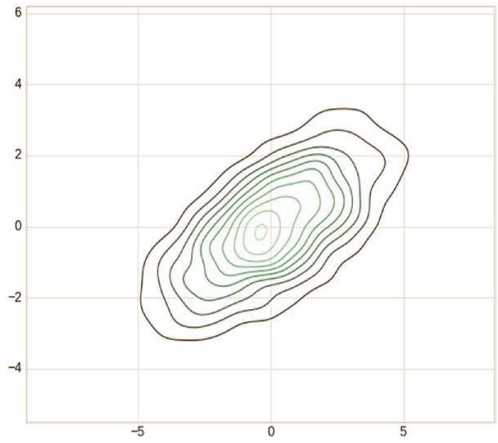
```
for col in 'xy':
sns.kdeplot(data[col], shade=True)
```



Gambar 6.5 grafik *kernel density entimation*

Menggunakan data yang sama, visualisasi dua dimensi menggunakan kdeplot:

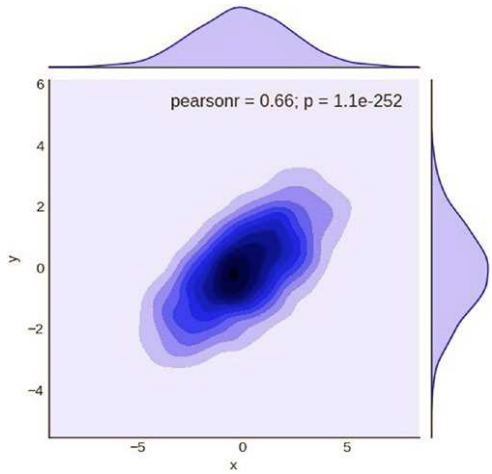
```
sns.kdeplot(data);
```



Gambar 2.6 Grafik *kernel-density* dua dimensi.

Berikut ini contoh penggabungan *joint distribution* dan *marginal distribution* menggunakan *sns.jointplot*:

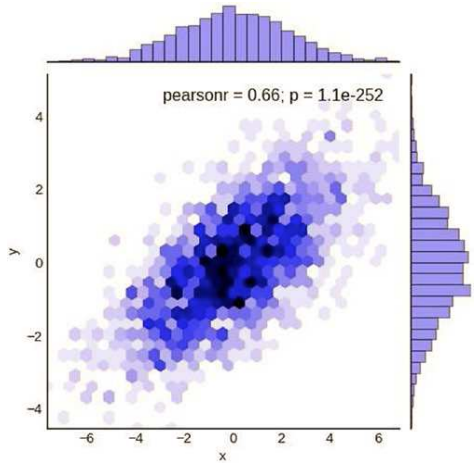
```
with sns.axes_style('white'):
     sns.jointplot("x", "y", data, kind='kde');
```



Gambar 2.7 Grafik *join distribution*

Plot gabungan dapat juga digabungkan dengan histogram seperti dalam Gambar 6.8.

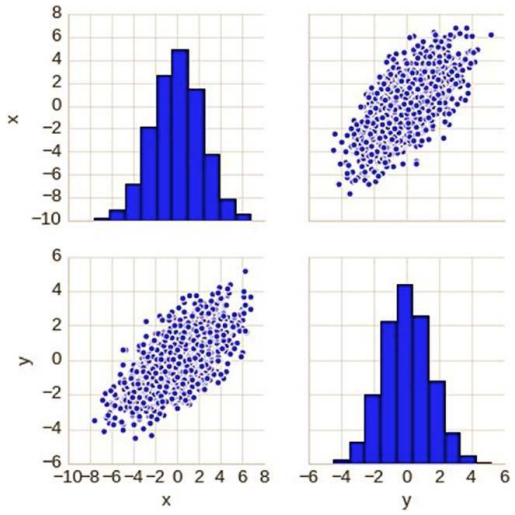
```
with sns.axes_style('white'):
     sns.jointplot("x", "y", data, kind='hex')
```



Gambar 6.7 Grafik *join distribution* dengan histogram.

Visualisasi hubungan antar variabel dapat dilakukan dengan menggunakan *sns.pairplot* (Gambar 6.9):

```
sns.pairplot(data)
```



Gambar 6.9 Grafik hubungan multi demensi.

Plotly

Library Plotly menyediakan fasilitas membuat grafik interaktif dan berkualitas baik untuk dicetak atau untuk online. Aneka format grafik tersedia baik untuk offline atau online. Listing 6-5 mencontohkan implementasi grafik dimanis *heatmap* (Gambar 6-10).

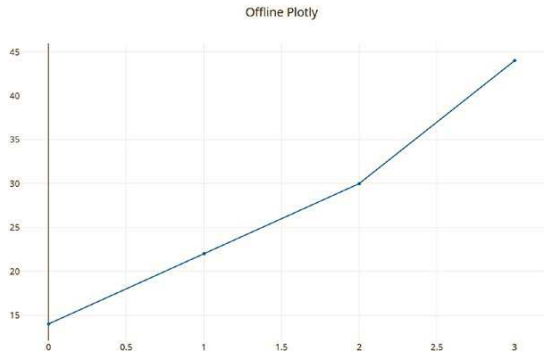
Listing 6-5. Importing and Using the Plotly Library

```
import plotly.express as px
data=[[1, 25, 30, 50, 1], [20, 1, 60, 80, 30], [30, 60, 1, 5, 20]]
fig = px.imshow(data, labels=dict(x="Hari", y="Waktu", color="Produktivitas"),
                x=['Senin', 'Selasa', 'Rabu', 'Kamis', 'Jumat'],
                y=['Pagi', 'Siang', 'Malam']                )
fig.update_xaxes(side="top")
fig.show()
```

Gambar 6.10 Grafik dinamis heatmap.

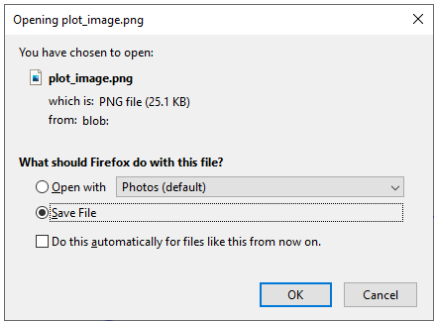
Dengan menggunakan plotly.offline dapat dibuat grafik offline seperti dalam Gambar 6.11

```
import plotly.offline as offline
import plotly.graph_objs as go
offline.plot({'data': [{'y': [14, 22, 30, 44]}],
              'layout': {'title': 'Offline Plotly', 'font': dict(size=16)}},
              image='png')
```



Gambar 6.11 Grafik *offline* plotly.

Hasil keluaran grafik akan ditawarkan untuk dibuka atau disimpan oleh browser.



Geoplotlib

Geoplotlib merupakan tool untuk membuat berbagai tipe dan plot peta data geografis. Geoplotlib membutuhkan Pyglet sebagai antarmuka pemrograman berorientasi objek.

Panda

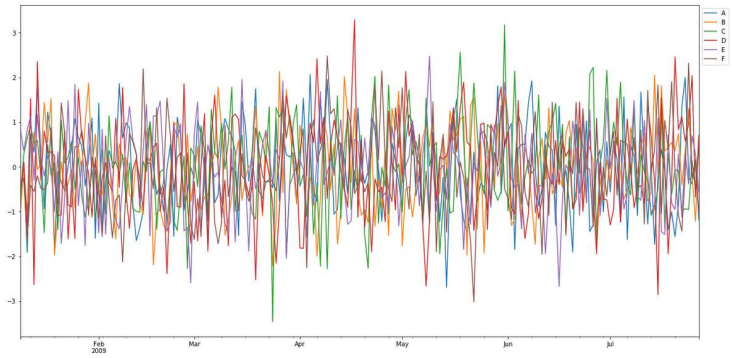
Pandas merupakan library Python yang ditulis untuk pengolahan dan analisis data. Panda diterapkan dalam berbagai bidang akademik dan komersial, termasuk keuangan, ekonomi, statistik, iklan, web analitik, dan lainnya.

6.3 Pengantar *Plotting* dalam Python

Berikut ini akan diberikan contoh plotting dasar secara langsung. Listing 6-7 menunjukkan implementasi dasar plotting suatu plot yang hasilnya ditampilkan dalam Gambar 6.13.

Listing 6-7

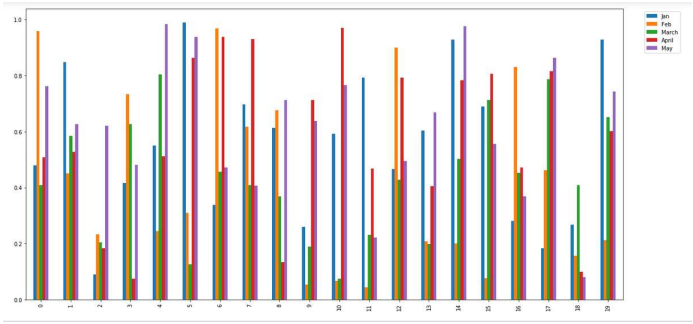
```
%matplotlib inline
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(200,6),
     index= pd.date_range('1/9/2009', periods=200),
     columns= list('ABCDEF'))
df.plot(figsize=(20, 10)).legend(bbox_to_anchor=(1, 1))
```



Gambar 6.13 Grafik plot secara langsung

Listing 6.8

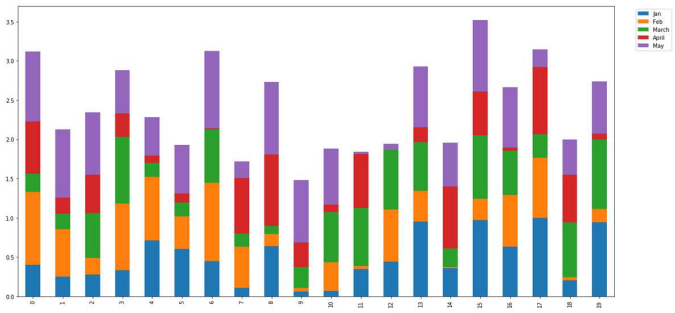
```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(20,5),
     columns=['Jan','Feb','March','April', 'May'])
df.plot.bar(figsize=(20, 10)).legend(bbox_to_anchor=(1.1, 1))
```



Gambar 6.14 Grafik batang dengan plot langsung.

Listing 6.9

```
import pandas as pd
df = pd.DataFrame(np.random.rand(20,5), columns=['Jan','Feb','March','April',
'May'])
df.plot.bar(stacked=True, figsize=(20, 10)).legend(bbox_to_anchor=(1.1, 1))
```

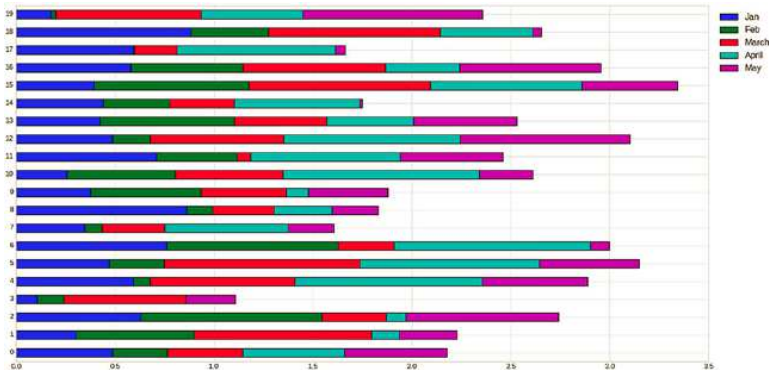


Gambar 6.15 Grafik batang bertumpuk dengan plot langsung.

Untuk membuat *horizontal bar plot*, dapat digunakan method *barh* seperti ditunjukkan dalam Listing 6-10. Gambar 6-16 menampilkan hasilnya.

Listing 2-10. Bar Plots

```
import pandas as pd
df = pd.DataFrame(np.random.rand(20,5), columns=['Jan','Feb',
'March','April', 'May']) df.plot.barh(stacked=True,
figsize=(20, 10)).legend(bbox_to_anchor=(1.1, 1))
```

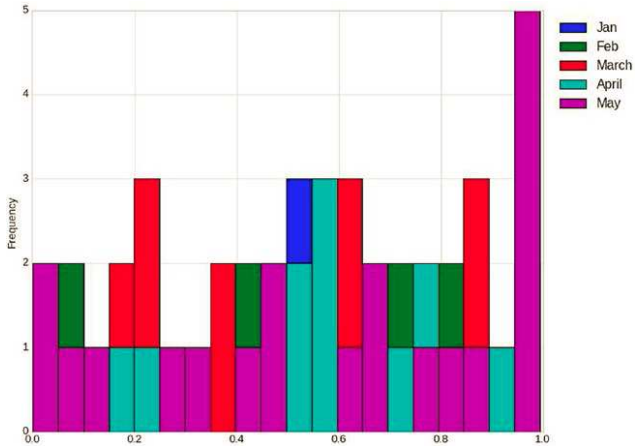


Gambar 6.16 Grafik *Bar Plot* horisontal

Histogram dapat di-*plot* menggunakan metoda *plot.hist()*; juga dapat diset jumlah batang yang ditampilkan seperti pada Listing 6-11. Grafil hasilnya ditampilkan dalam Gambar 6-17.

Listing 6-11. menggunakan atribut *bin*/batang

```
import pandas as pd
df = pd.DataFrame(np.random.rand(20,5), columns=['Jan','Feb',
'March','April', 'May'])
df.plot.hist(bins= 20, figsize=(10,8)).legend
bbox_to_anchor=(1.2, 1))
```

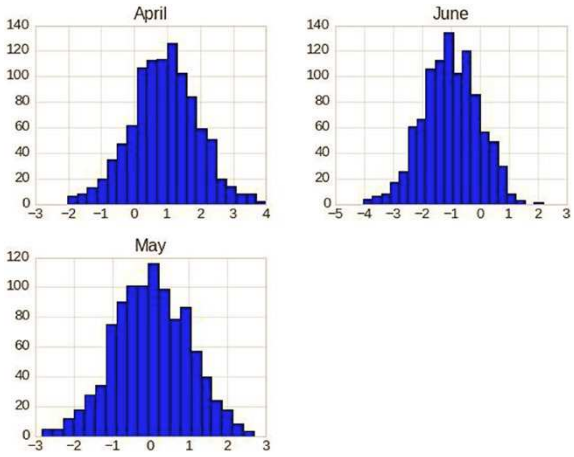


Gambar 6.16 Grafik Histogram

Listing 6-12 memplot histogram jamak per column dari data set (Gambar 6-18).

Listing 6-12. Histogram Jamak per Column

```
import pandas as pd
import numpy as np
df=pd.DataFrame({'April':np.random.randn(1000)+1,
                 'May':np.random.randn(1000),
                 'June': np.random.randn(1000) - 1},
                 columns=['April','May', 'June'])
df.hist(bins=20)
```

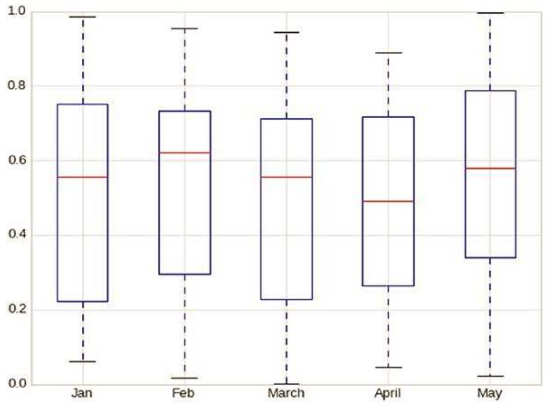


Gambar 6.18 Histogram Jamak per Kolom.

Listing 6-13 mengimplementasikan *box plot* (Lihat Gambar 6-19).

Listing 6-13. Membuat Box Plot

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(20,5),
                  columns=['Jan','Feb','March','April', 'May'])
df.plot.box()
```

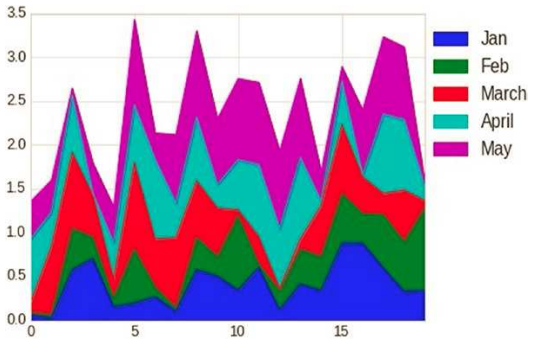


Gambar 6.19 Grafik *Box Plot*

Listing 6-14 mengimplementasikan area plot (Gambar 6-20).

Listing 6-14. Membuat *Area Plot*

```
import pandas as pd
import numpy as np
df = pd.DataFrame(np.random.rand(20,5),
                  columns= ['Jan','Feb','March','April', 'May'])
df.plot.area(figsize=(6, 4)).legend
(bbox_to_anchor=(1.3, 1))
```

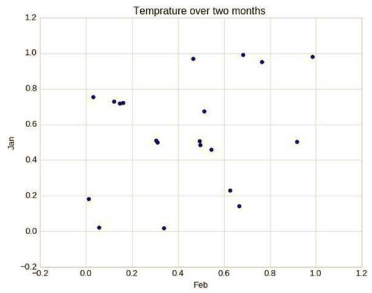


Gambar 6.20 Grafik *Area Plot*

Listing 6-15 menampilkan kode untuk membuat *scatter plot* (Gambar 6-21).

Listing 6-15. Kode membuat *Scatter Plot*

```
import pandas as pd
import numpy as np
df= pd.DataFrame(np.random.rand(20,5),columns=['Jan','Feb','March','April', 'May'])
df.plot.scatter(x='Feb', y='Jan', title='Temperature over two months ')
```



Gambar 6.21 Grafik *Scatter Plot*.