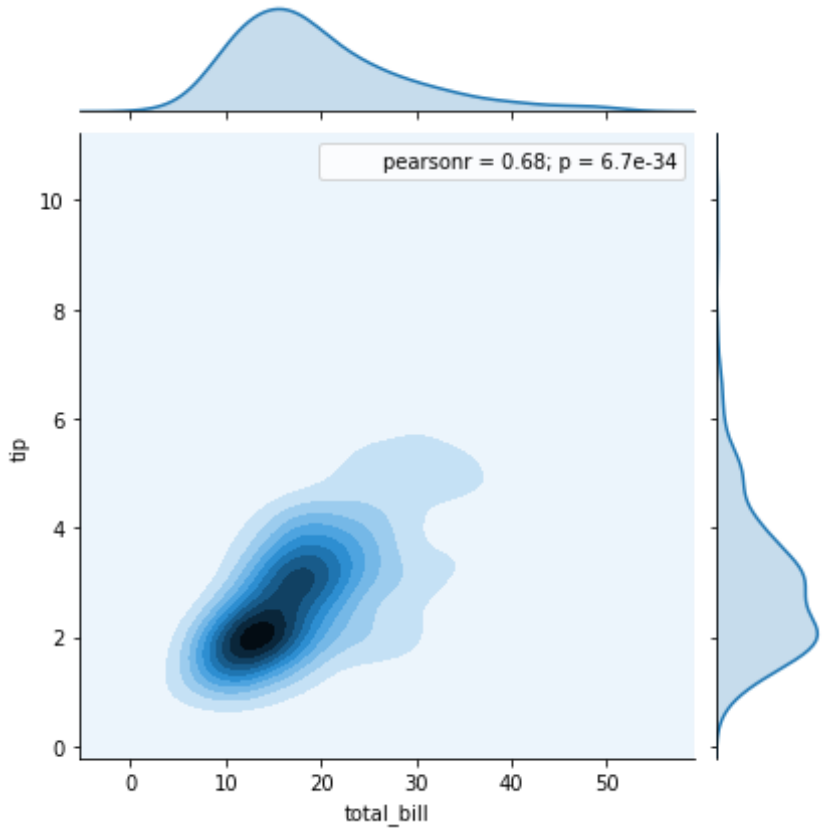
SERI KOMPUTASI



TUTORIAL VISUALISASI DATA MENGGUNAKAN SEABORN

Sandy Hardian Susanto Herho

Tutorial Visualisasi Data

Menggunakan Seaborn

Sandy Hardian Susanto Herho[1]

14 Juni 2019

[1]https://sandyherho.github.io/

Untuk Guru - Guru saya: **M.R. Syahputra**,
D.E. Irawan, dan **F.R. Fajary**

Daftar Isi

Kata Pengantar

Seaborn merupakan pustaka visualisasi data pada lingkungan Python bersifat sumber terbuka yang berlisensi BSD dan dibangun di atas pustaka matplotlib. Seaborn akan mempermudah kita sebagai analis data untuk memproduksi visualisasi yang indah tanpa kostumisasi rumit seperti yang kita hadapi pada matplotlib. Seaborn awalnya diperkenalkan Michael askom[1], yang kala itu merupakan seorang mahasiswa doktoral neurosains di Universitas Stanford, untuk memvisualisasikan data untuk analisis jaringan saraf pada awal tahun 2014 silam. Saat ini seaborn telah memasuki versi stabil 0.9.0, dan akan terus dikembangkan oleh komunitasnya yang cukup besar.

Tutorial ini sendiri dimaksudkan untuk membantu pembaca untuk memahami dasar - dasar penggunaan seaborn untuk memvisualisasikan analisis statistik sederhana yang biasa dihadapi oleh mahasiswa tingkat dua di seluruh jurusan perguruan tinggi di Indonesia pada praktikum matakuliah statistika. Diharapkan tutorial ini dapat dijadikan alternatif penyelesaian tugas praktikum yang umumnya menggunakan piranti lunak berbayar yang tidak mendidik, seperti SPSS, MS Excel, Matlab, dll.Untuk memahami tutorial ini, pembaca diharapkan telah terbiasa dengan sintaks - sintaks dalam bahasa pemrograman Python dan mengenal dasar - dasar penggunaan matplotlib.

Saya memohon maaf jika tutorial ini punya banyak sekali kekurangan, karena pada hakikatnya saya bukan bekerja sebagai analis data profesional yang bekerja di persuahaan besar, yang menggunakan Python sebagai rutinitas hariannya. Pada dasarnya tutorial ini hanya merupakan catatan - catatan pembelajaran yang saya himpun dari berbagai sumber, utamanya dari dokumentasi resmi[2], sehingga tentunya jika ada yang tidak jelas dalam tutorial ini atau ingin memperdalam bagian tertentu, pembaca dapat mengunjungi laman tersebut.

Tutorial ini juga bersifat sumber terbuka, karena dituliskan dengan menggunakan LaTeX dan berlisensi milik publik (*copy-left*), sehingga para pembaca dapat mengopi dan mengubahnya, bahkan untuk kepentingan ko-

[1] `https://www.cns.nyu.edu/~mwaskom/`

[2] `http://seaborn.pydata.org/`

mersil sekalipun, secara cuma - cuma di `https://github.com/sandyherho/buku_seaborn`.

Akhir kata, semoga tutorial singkat dapat membantu dan saya menanti dengan tangan terbuka kolaborasi pembelajaran berbasis kode terbuka di laman GitHub saya.

Sandy H.S. Herho

College Park, MD.

1

Pendahuluan

Pada bidang keilmuan analisis data, cara terbaik untuk memperoleh pemahaman yang mendalam terhadap suatu data adalah dengan cara memvisualisasikannya. Representasi visual data mempermudah pemahaman otak kita sebagai manusia (berbeda dengan komputer yang lebih memahami angka dibandingkan visual), sehingga memungkinkan kita untuk mengeksplorasi data tersebut secara lebih jauh.

Sebagai pengguna setia Python, tentunya kita telah terbiasa menggunakan pustaka matplotlib guna memvisualisasikan data dalam keseharian kita. Seaborn sesungguhnya tidaklah jauh berbeda dibandingkan matplotlib, sebab seaborn dibangun di atas matplotlib sebagai pustaka fondasi utama-nya.

Plot - plot rumit yang sukar dikerjakan oleh para analis data pemula dalam matplotlib dapat dengan mudah dikerjakan dalam seaborn. Bahkan terkadang dapat ddikerjakan dengan menggunakan satu baris perintah saja (seperti yang akan kita sama - sama lihat dalam bab - bab berikutnya).

Seaborn memberikan solusi terhadap dua permasalahan berikut yang umumnya dihadapi oleh para analis data pemula ketika menggunakan matplotlib:

- Kerumitan penggunakan parameter - parameter standar dalam matplotlib.
- Kerumitan matplotlib ketika bekerja dengan *data frame*.

Sebagaimana yang telah saya bahas di atas, seaborn dibangun dengan matplotlib sebagai pustaka fondasi utama-nya. Seaborn dalam hal ini merupakan pelengkap matplotlib, bukan pengganti. Namun, seaborn dilengkapi oleh fitur - fitur penting yang mempermudah kita dalam memvisualisasikan data. Berikut ini merupakan fitur - fitur tersebut:

- Tema *built-in* yang mempercantik tampilan grafis matplotlib.
- Visualisasi data univariat dan bivariat.

- Penyesuaian dan visualisasi model regresi linier.
- Plot data deret waktu (*time-series*) statistik.
- Dapat bekerja dengan baik bersama dengan berbagai jenis struktur data pada numpy dan Pandas.

Dalam banyak kasus, kita masih akan menggunakan matplotlib untuk menghasilkan plot - plot sederhana. Oleh karena itu, pengetahuan tentang matplotlib sangat dibutuhkan untuk mengubah plot - plot standar dalam seaborn.

2

Pengaturan Lingkungan Komputasi

Pada bagian ini kita akan membahas proses penyesuaian lingkungan komputasi yang dibutuhkan untuk melakukan visualisasi data dengan menggunakan seaborn. Kita memulainya dengan melakukan instalasi seaborn.

Untuk pengguna Python standar kita dapat menggunakan *pip installer* dengan menjalankan perintah berikut di Terminal (GNU/Linux dan MacOS) atau PowerShell (MS Windows):

```
pip install seaborn
```

Namun saya sangat menyarankan agar kita menggunakan distribusi Anaconda yang akan berguna bagi para analis data kedepannya. Distribusi ini dapat diunduh secara gratis di `https://www.anaconda.com/distribution/`. Distribusi ini memuat hampir seluruh pustaka yang dibutuhkan untuk melakukan komputasi ilmiah.

Selain itu, kita juga dapat melakukan instalasi seaborn secara langsung dari repositori github-nya Michael Waskom, dengan menjalankan perintah berikut ini:

```
pip install git+https://github.com/mwaskom/seaborn.git
```

Sebagai penutup bagian ini, saya akan mengingatkan bahwa seaborn merupakan pustaka Python yang bergantung pada hal - hal berikut ini:

- Python 2.7+ atau Python 3.4+
- numpy
- scipy
- pandas
- matplotlib

Jadi pastikan kalian telah melakukan instalasi hal - hal tersebut sebelum melanjutkan tutorial ini.

3

Mengimpor Dataset dan Pustaka

Pada bagian ini kita akan membahas cara mengimpor dataset dan pustaka yang akan digunakan untuk visualisasi data menggunakan seaborn.

Kita mengawalinya dengan mengimpor Pandas, yang mana merupakan pustaka Python paling populer untuk mengelola dataset bersifat relasional (berformat tabel). Seaborn akan sangat bermanfaat ketika digunakan untuk memvisualisasikan *data frame* yang banyak digunakan untuk analisis data dewasa ini.

Jalankan perintah berikut ini untuk mengimpor Pandas pada program Python kalian:

```
import pandas as pd
```

Seperti yang telah saya bahas pada bagian sebelumnya, kita juga perlu menggunakan matplotlib untuk pengaturan lebih lanjut pada plot yang dihasilkan oleh seaborn:

```
import matplotlib.pyplot as plt
```

Kita dapat mengimpor seaborn dengan menjalankan perintah ini:

```
import seaborn as sns
```

Sesudah mengimpor pustaka - pustaka yang dibutuhkan, maka kini tiba waktunya bagi kita untuk mengimpor dataset yang hendak diolah. Seaborn mempunyai beberapa dataset bawaan yang telah diinstal secara otomatis ketika kita melakukan instalasi pustaka ini. Pada tutorial ini kita akan menggunakan beberapa dataset bawaan ini sebagai medium berlatih. Kita menggunakan fungsi berikut ini guna memuat dataset:

```
sns.load_dataset()
```

Dataset yang kita gunakan untuk visualisasi dalam seaborn akan berbentuk *data frame* Pandas secara *default*. Berikut ini salah satu contoh program yang dapat kalian jalankan untuk memuat salah satu dataset bawaan seaborn, yakni `tips`:

```
import seaborn as sns
df = sns.load_dataset('tips')
print(df.head())
```

Berikut ini hasil yang akan kalian dapatkan pasca menjalankan program sederhana tersebut:

```
total_bill   tip     sex smoker  day    time  size
0       16.99  1.01  Female     No  Sun  Dinner     2
1       10.34  1.66    Male     No  Sun  Dinner     3
2       21.01  3.50    Male     No  Sun  Dinner     3
3       23.68  3.31    Male     No  Sun  Dinner     2
4       24.59  3.61  Female     No  Sun  Dinner     4
```

Untuk melihat berbagai dataset bawaan yang terdapat dalam seaborn, kita dapat menggunakan fungsi `get_dataset_names()` berikut ini:

```
import seaborn as sns
print(sns.get_dataset_names())
```

Berikut ini luaran yang akan kalian lihat pada layar Terminal kalian:

```
['anscombe', 'attention', 'brain_networks', 'car_crashes', '
    diamonds', 'dots', 'exercise', 'flights', 'fmri', 'gammas', '
    iris', 'mpg', 'planets', 'tips', 'titanic']
```

Dengan menggunakan *data frame*, kia dapat melakukan pemeriksaan data dengan mudah karena data disimpan dalam bentuk tabular persegi panjang yang 'ramah' untuk dilihat. Setiap baris dari tabel ini memuat nilai untuk setiap conto, dan masing - masing kolom dari tabel ini merepresentasikan vektor untuk variabel tertentu. *Data frame* dapat memuat berbagai jenis tipe data yang tersedia dalam Python, seperti numerik; karakter; logika; dll. *Data frame* merupakan produk bawaan dari pustaka Pandas yang berfungsi untuk memuat data tabular dua dimensi. Untuk pembahasan lebih lanjut soal *data frame*, kalian dapat mengunjungi laman dokumentasi Pandas berikut ini: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`.

4

Dimensi Estetis Dalam Visualisasi Data

Visualisasi data tidak melulu berkaitan dengan keindahan gambar hasil visualisasi data tersebut. Karena pengaturan estetika gambar agar 'enak' dipandang mata merupakan proses yang bebeda. Sebagai analis data tentunya visualisasi merupakan tahapan penting untuk penyampaian ide, sehingga ide kita diperhatikan oleh audiens.

Estetika (saya tidak mendiskusikan estetika secara filosofis, kalau kalian mau mendiskusikan hal ini silakan baca **buku** saya yang lain, atau kontak saya secara pribadi hehe) secara umum merupakan bidang keilmuan yang terkait dengan pembahasan tentang apresiasi terhadap keindahan, khususnya pada bidang seni. Dalam hal visualisasi data ilmiah yang kerap dilakukan oleh para analis data, tentunya konsep estetika ini dipadukan dengan tujuan penyampaian data agar efektif dan mudah diterima oleh audiens (sebetulnya kata yang benar bukan audiens, kan audiens dari kata dasar latin *audentia* yang artinya mendengar, tapi ya sudah lah...).

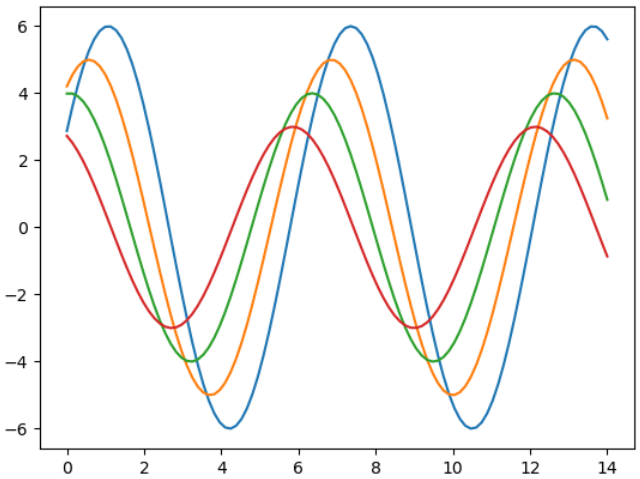
Matplotlib hadir dengan berbagai fitur yang sangat mendukung upaya kostumisasi pengguna untuk mendapatkan visual yang menyesuaikan selera pengguna, namun sayangnya karena terlalu banyak membebaskan pengguna, matplotlib kadang membingungkan buat pengguna pemula dan/atau yang tidak berjiwa seni sama sekali (karena tidak mengerti soal komposisi warna yang 'pas' untuk visualisasi data ilmiah). Berbeda dengan matplotlib, seaborn hadir dengan tema - tema standar yang 'indah' dan kita juga dapat melakukan pengelolaan visual secara lebih leluasa dibandingkan dengan menggunakan matplotlib.

Berikut ini saya berikan salah satu contoh program visualisasi data dengan menggunakan matplotlib:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3
4 def plotsin(flip=1):
```

```
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x, np.sin(x + i*0.5)*(7 - i) * flip)

plotsin()

plt.show()
```

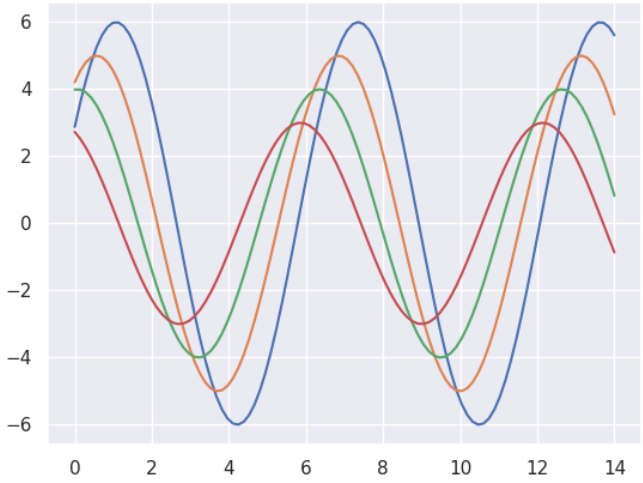
Visualnya ditunjukkan pada gambar berikut ini:



Untuk mengubah visual tersebut menjadi plot *default* seaborn, kita dapat memanfaatkan fungsi `set()`:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

sns.set()

plotsin()

plt.show()
```

Berikut ini luaran visual-nya:



Dari kedua gambar tersebut, kita dapat memahami bahwa pada dasarnya representasi data yang dihasilkan oleh matplotlib dan seaborn bersifat identik, namun perbedaaan utamanya terletak pada estetika visualnya. Secara umum terdapat dua buah perbedaan antara seaborn dan matplotlib, yakni pada *style* dan skala visualnya.

Untuk mengatur *style* visual pada seaborn kita dapat memanfaatkan fungsi `set_style()`. Dengan menggunakan fungsi ini kita dapat mengatur tema visual pada plot kita dengan lima tema berikut ini:

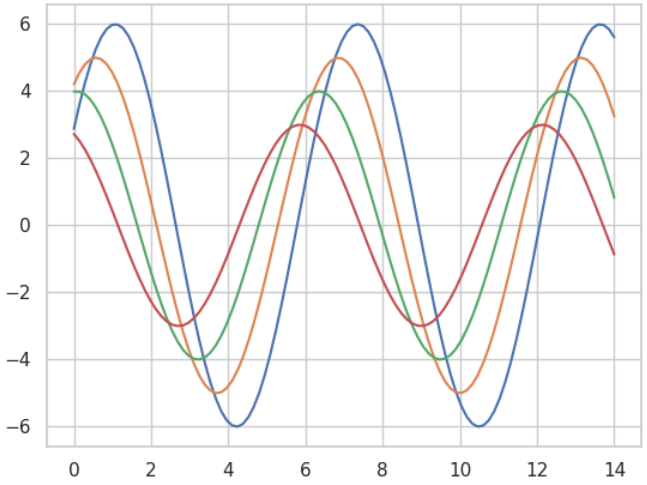
- Darkgrid
- Whitegrid
- Dark
- White
- Ticks

Ok, sekarang waktunya kita mencoba mengubah tema pada plot kita menjadi whitegrid (karena *default*-nya darkgrid):

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

```

```
sns.set()

sns.set_style('whitegrid')

plotsin()

plt.show()
```

Berikut ini tampilannya:

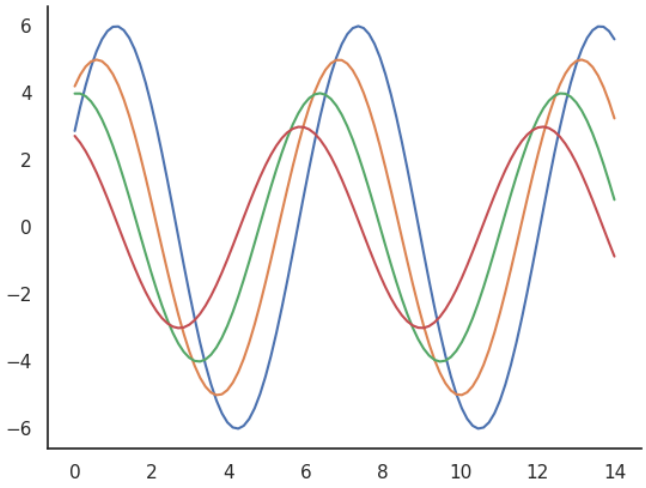


Yang menjadi perbedaan dibandingkan plot sebelumnya adalah tema latar belakangnya.

Pada tema white dan ticks, kita dapat membuang *axis spine* pada sisi atas dan kanan plot dengan menggunakan fungsi `despine()`:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

sns.set()

sns.set_style('white')

plotsin()

sns.despine()

plt.show()
```

Berikut ini visualnya:



Jika kalian ingin mengkostumisasi *style* pada tampilan seaborn kalian, kita dapat memasukkan *dictionary* ke dalam fungsi `set_style()`. Parameter - parameter yang dapat kita jadikan *input*-an dapat dilihat menggunakan fungsi `axes_style()`:

```
import seaborn as sns
print(sns.axes_style())
```

Berikut ini luarannya:

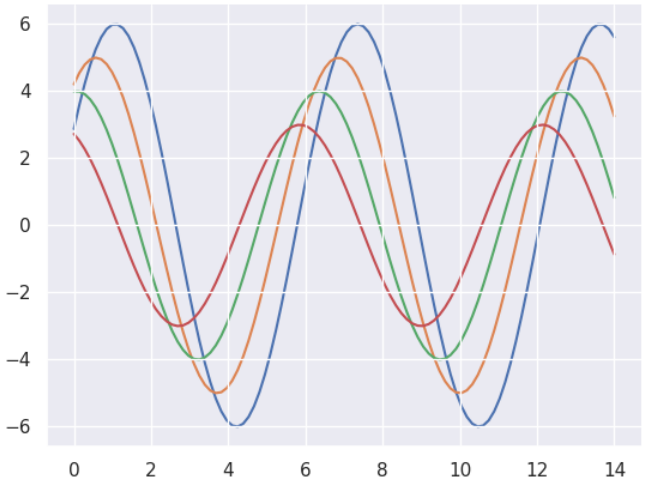
```
{'axes.facecolor': 'w', 'axes.edgecolor': 'k', 'axes.grid':
    False, 'axes.axisbelow': 'line', 'axes.labelcolor': 'k', '
    figure.facecolor': 'w', 'grid.color': '#b0b0b0', 'grid.
    linestyle': '-', 'text.color': 'k', 'xtick.color': 'k', '
    ytick.color': 'k', 'xtick.direction': 'out', 'ytick.direction
    ': 'out', 'lines.solid_capstyle': 'projecting', 'patch.
    edgecolor': 'k', 'image.cmap': 'viridis', 'font.family': ['
    sans-serif'], 'font.sans-serif': ['DejaVu Sans', 'Bitstream
    Vera Sans', 'Computer Modern Sans Serif', 'Lucida Grande', '
    Verdana', 'Geneva', 'Lucid', 'Arial', 'Helvetica', 'Avant
    Garde', 'sans-serif'], 'patch.force_edgecolor': False, 'xtick
    .bottom': True, 'xtick.top': False, 'ytick.left': True, '
    ytick.right': False, 'axes.spines.left': True, 'axes.spines.
    bottom': True, 'axes.spines.right': True, 'axes.spines.top':
    True}
```

Berikut ini salah satu contoh penerapannya:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

```

```
def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

sns.set()

sns.set_style('darkgrid', {'axes.axisbelow':False})

plotsin()

sns.despine()

plt.show()
```

Yang akan menghasilkan visual seperti berikut ini:



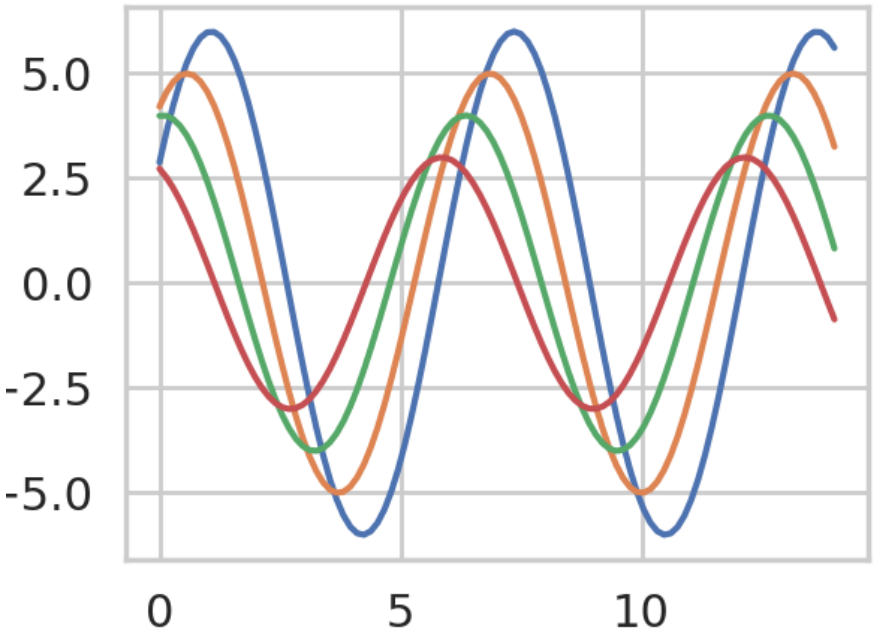
Di samping itu, seperti yang telah saya terangkan sebelumnya, kita juga dapat mengatur skala visual yang hendak kita hasilkan dengan menggunakan fungsi `set.context()`. Pengaturan skala ini disesuaikan secara relatif terhadap konteks penggunaannya. Secara *default* seaborn akan menyesuaikan dalam konteks notebook. Berikut ini konteks - konteks skala yang tersedia dalam pustaka seaborn:

- Paper
- Notebook
- Talk
- Poster

Karena saya di sini menulisnya tidak pakai Jupyter notebook jadi tidak begitu terlihat perbedaan skalanya, namun coba *deh* kalian jalankan program **ini** nanti akan terlihat perbedaannya dibandingkan plot - plot sebelumnya:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

sns.set()

sns.set_style('whitegrid')

sns.set_context('poster')
plotsin()

plt.show()
```

Punten kalau perbedaan visual-nya tidak begitu terlihat, maklum *scripting* LaTeX dan Python -nya masih pakai vim hehe:



5

Palet Warna

Warna memainkan peranan paling penting dalam visualisasi data jika dibandingkan dengan aspek - aspek lainnya. Ketika kita menggunakan komposisi warna secara 'pas' maka plot kita akan tampak lebih bermakna dimata audiens, meskipun data yang kita hadirkan pada dasarnya sampah. Palet sendiri pada dasarnya bermakna sebagai suatu wadah tempat pelukis mencampurkan kombinasi warna. Hal yang mirip akan kita pelajari di bab ini, namun bentuknya tentu saja berbeda dengan palet para pelukis.

Untuk membuat palet warna, seaborn menyediakan fungsi `color_palette()` yang dapat digunakan sebagai berikut ini:

```
import seaborn as sns
sns.color_palette(palette=None, n_colors=None, desat=None)
```

Pada tabel berikut ini, saya menampilkan parameter - parameter seaborn yang dibutuhkan untuk membentuk suatu palet warna:

Jenis Parameter	**Deskripsi**
`palette`	Jenis palet yang hendak digunakan.
`n_colors`	Jumlah warna dalam palet. Secara *default* akan diisi dengan enam buah warna.
`desat`	Proporsi desaturasi setiap warna.

Berikut ini jenis - jenis palet yang telah disediakan oleh seaborn:

- Deep
- Muted
- Bright
- Pastel

- Dark
- Colorblind

Di samping itu, kita juga dapat membuat palet warna sesuai keinginan kita sendiri.

Akan sangat sulit bagi kita untuk menentukan palet mana yang hendak kita pilih tanpa mengetahui karakteristik data yang hendak kita plot. Maka dari itu berhati-hatilah sebelum menentukan palet warna mana yang kalian pilih. Secara umum dapat diklasifikasikan tiga tipe data yang (mungkin) butuh palet warna yang berbeda, yakni:

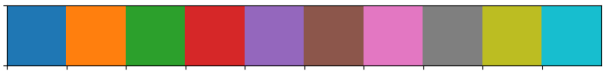
- kualitatif
- sekuensial
- kontras (*diverging*)

Untuk menampilkan palet - palet warna tersebut, seaborn mempunyai fungsi `palplot()` yang ditujukkan untuk memvisualisasikan palet warna ke dalam bentuk *array* horizontal.

Palet warna kualitatif seperti namanya, sangat cocok untuk memplot data yang bersifat kategoris. Berikut ini contoh visualnya:

```
1 import matplotlib.pyplot as plt
2 import seaborn as sns
3
4 current_palette = sns.color_palette()
5
6 sns.palplot(current_palette)
7
8 plt.show()
```

Hasilnya:

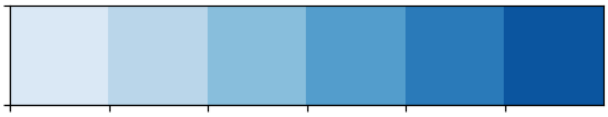


Kita belum memasukkan parameter apapun ke dalam fungsi `color_palette()`, maka kita hanya dapat melihat enam buah warna yang terplot secara *default*. Kita dapat mengubahnya dengan memasukkan jumlah warna yang kita kehendaki ke dalam parameter `n_colors()`. Secara *default* pula, fungsi `palplot()` akan memplot *array* palet warna secara horizontal.

Palet sekuensial sendiri bermanfaat untuk memvisualisasikan data yang membentang dari nilai paling rendah ke nilai paling tinggi. Kita harus menambahkan karakter `s` sebagai masukkan warna dalam fungsi `color_palette()` guna memplot palet sekuensial ini:

```
import matplotlib.pyplot as plt
import seaborn as sns

current_palette = sns.color_palette()

sns.palplot(sns.color_palette('Blues')) #jgn lupa tambahin 's'

plt.show()
```

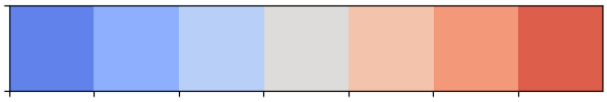
Hasilnya:



Jenis palet terakhir adalah palet kontras (*diverging*) yang utamanya tersusun dari dua kutub warna yang berlainan dengan asumsi masing - masing kutub bernilai -1 dan +1 dengan titik pusat secara *default* bernilai nol:

```
import matplotlib.pyplot as plt
import seaborn as sns

current_palette = sns.color_palette()

sns.palplot(sns.color_palette("coolwarm", 7))

plt.show()
```

Berikut ini visualnya:

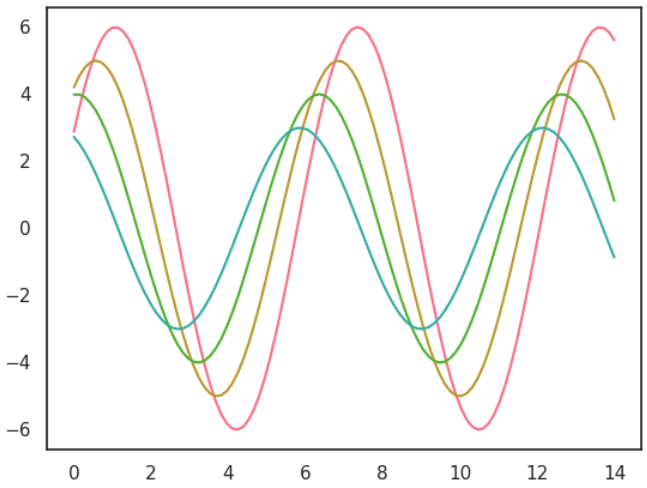


Terdapat juga fungsi lain yang mempunyai banyak kemiripan dengan fungsi `color_palette()`, yakni fungsi `set_palette()`. Berikut ini contoh penggunaannya:

```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

def plotsin(flip=1):
    x = np.linspace(0,14,100)
    for i in range(1,5):
        plt.plot(x,np.sin(x + i * 0.5) * (7 - i) * flip)

sns.set()

sns.set_style('white')

```

```
14 sns.set_palette('husl')
15
16 plotsin()
17
18 plt.show()
```

Hasilnya:



6

Visualisasi Distribusi Univariat

Distribusi dari suatu data merupakan hal pokok yang wajib diketahui oleh para analis ketika pertama kali dihadapkan pada data. Pada bab ini kita akan melihat bagaimana seaborn akan mempermudah kita dalam menganalisis distribusi data univariat.

Kita akan memanfaatkan fungsi `distplot()` guna memvisualisasikan distribusi data univariat. Fungsi ini akan memvisualisasikan histogram yang sesuai dengan estimasi kepadatan kernel/*Kernel Density Estimation* (KDE) dari data. Berikut ini bentuk umum dari fungsi ini:

```
sns.distplot()
```

Dengan parameter - parameter sebagai berikut:

Parameter	**Jenis Data**
`data`	Jenis data yang dapat dijadikan masukkan antara lain adalah *Series*, *array* 1D, dan *list*
`bins`	Jumlah batang dalam histogram.
`hist`	Berbentuk *boolean*
`kde`	Berbentuk *boolean.*

6.1 Histogram

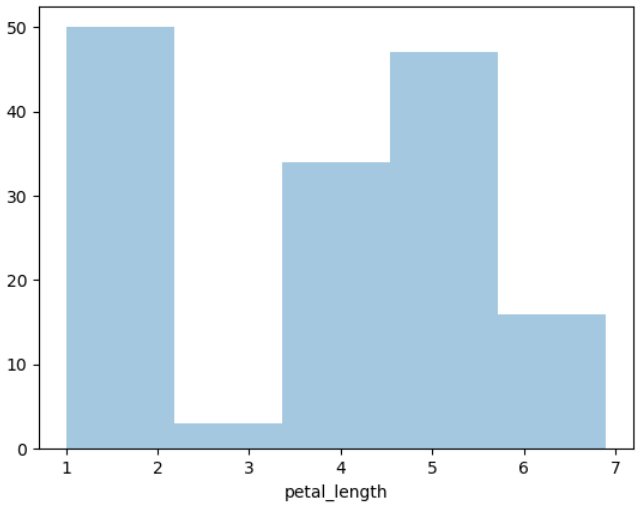
Histogram merupakan visualisasi grafis distrbusi data dalam bentuk batang yang membentang di sepanjang rentang data.

Seperti yang telah saya terangkan pada bab - bab awal, seaborn dilengkapi dengan beberapa dataset bawaan yang dapat kita manfaatkan sebagai

medan berlatih. Pada bab ini kita akan menggunakan dataset `iris`. Berikut ini program untuk menunjukkan histogram panjang kelopak bunga iris (`petal_length`):

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.distplot(df['petal_length'], kde=False)

plt.show()
```

Berikut ini tampilan histogram-nya:



Pada contoh tersebut, saya mengatur parameter `kde=False`, sehingga grafik tersebut tidak menampilkan KDE, hanya histogram saja yang ditampilkan.

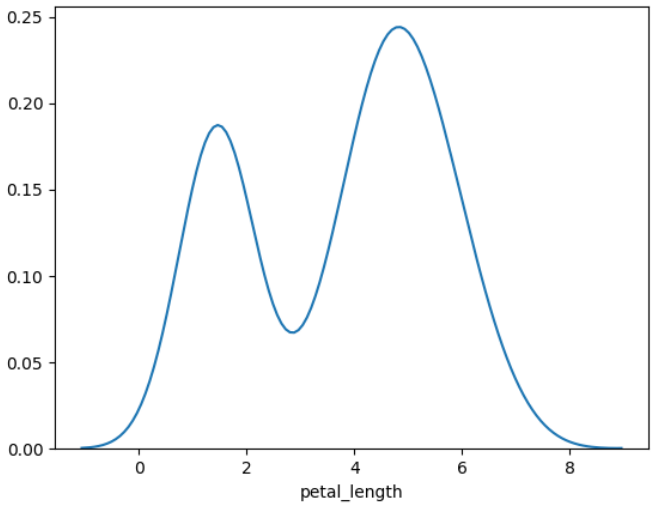
6.2 Estimasi Kepadatan Kernel (KDE)

KDE merupakan salah satu cara untuk menghitung fungsi kepadatan peluang dari suatu peubah acak yang bersifat kontinyu. KDE digunakan untuk analisis non-parametrik. Untuk memvisualisasikan KDE dengan fungsi `distplot()`, kita harus mengatur parameter `hist=False` seperti pada contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
4
5 df = sns.load_dataset('iris')
6
7 sns.distplot(df['petal_length'], hist=False)
8
9 plt.show()
```

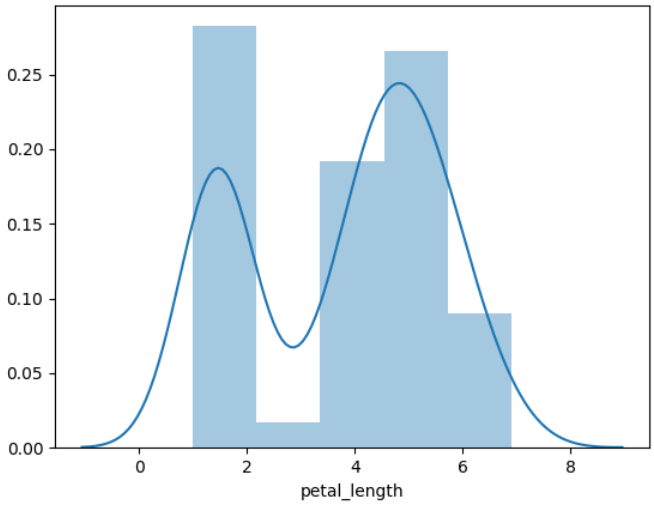
Berikut ini visual-nya:



6.3 Grafik KDE-Histogram

Kita juga dapat menggabungkan kedua grafik distribusi univariat ini:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('iris')
6
7 sns.distplot(df['petal_length'])
8
9 plt.show()
```


0.25
0.20
0.15
0.10
0.05
0.00
0
2
4
6
8
petal_length

7

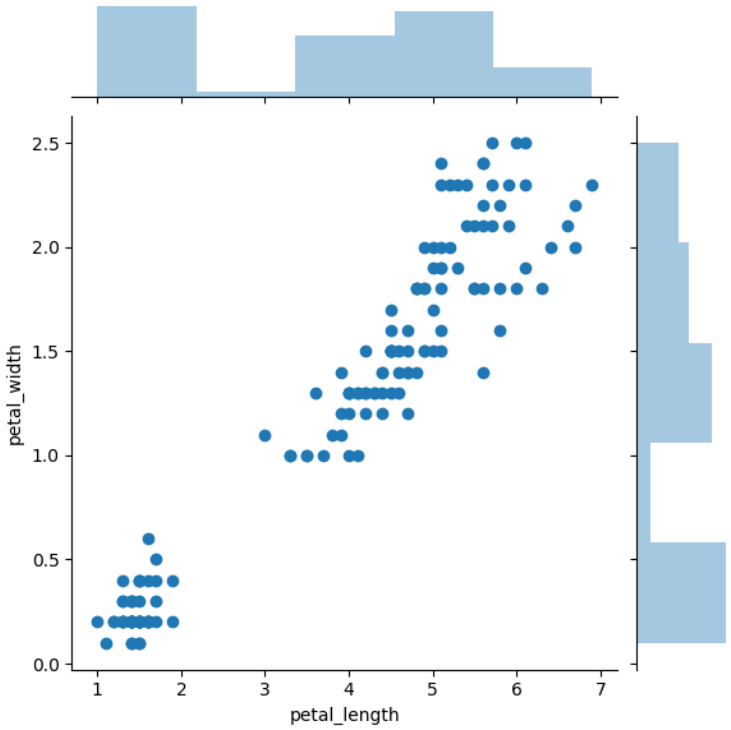
Visualisasi Distribusi Bivariat

Distribusi bivariat digunakan untuk menentukan hubungan antar dua peubah. Cara terbaik untuk mengetahui hubungan antara dua peubah ini adalah dengan memvisualisasikan distribusinya dengan menggunakan fungsi `jointplot()`. Fungsi ini dapat menggambarkan hubungan antar kedua peubah, sekaligus juga mengetahui distribusi univariat masing - masing peubah pada sumbu terpisah.

7.1 *Scatter Plot*

Scatter plot merupakan cara paling sederhana untuk memvisualisasikan distribusi data dua peubah yang masing - masing diwakili oleh sumbu-x dan sumbu-y:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('iris')
6
7 sns.jointplot(x='petal_length', y='petal_width', data=df)
8
9 plt.show()
```

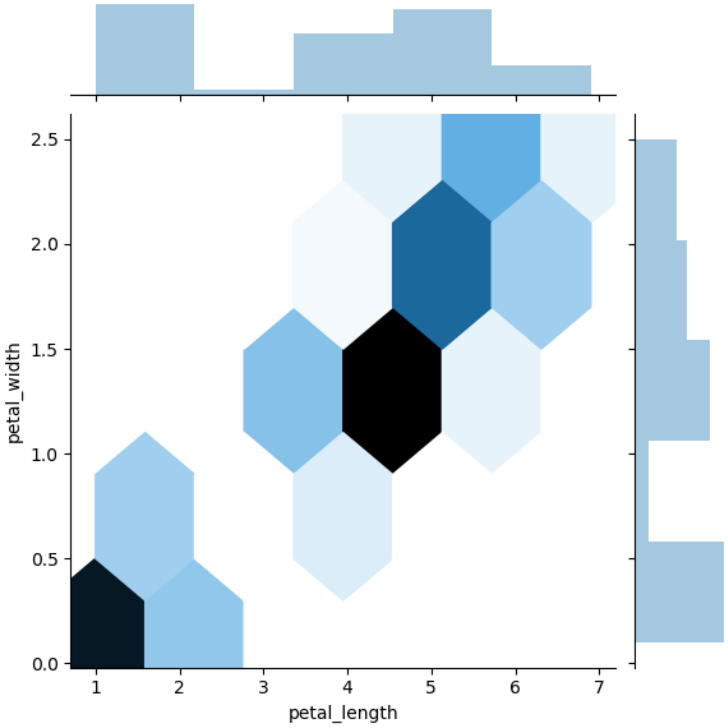


Gambar di atas menunjukkan hubungan antara peubah panjang dan lebar kelopak bunga iris. Nampak pada grafik hubungan berbanding lurus (korelasi positif) antar kedua peubah tersebut.

7.2 Grafik *Hexbin*

Grafik heksagonal umumnya digunakan untuk menganalisis data bivariat yang sangat tersebar dan sukar dianalisis ketika divisualisasikan dengan *scatter plot*. Untuk memvisualisasikan grafik ini, kita tinggal mengganti parameter `kind='hex'` pada fungsi `jointplot()`:

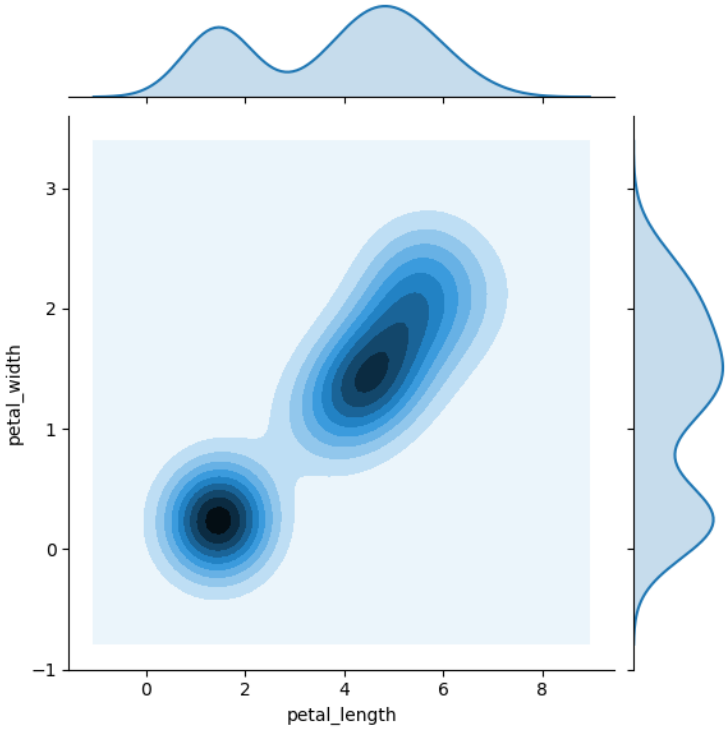
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('iris')
6
7 sns.jointplot(x='petal_length', y='petal_width', data=df, kind='
    hex')
8
9 plt.show()
```



7.3 Grafik KDE Bivariat

Seperti yang saya jelaskan pada bab sebelumnya, KDE merupakan salah satu cara non-parametrik untuk mengestimasikan distribusi suatu peubah. Untuk distribusi bivariat, kita juga dapat memvisualisasikan grafik KDE dengan memasukkan parameter `kind='kde'`. Perhatikanlah contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.jointplot(x='petal_length', y='petal_width', data=df, kind='
    kde')

plt.show()
```


petal_width
petal_length
3
2
1
0
−1
0
2
4
6
8

8

Visualisasi Hubungan Antar Banyak Peubah

Dalam penelitian yang sebenarnya, kita kerap kali dihadapkan dengan data yang mempunyai banyak peubah. Dalam kasus ini, tentunya kita harus menganalisis hubungan antar peubah secara keseluruhan. Namun, tentu kita berpikir akan sangat sulit dan memakan banyak waktu untuk memvisualisasikan distribusi bivariat untuk setiap kombinasi (n,2) pada masing - masing peubah. Untungnya dalam pustaka seaborn, kita dimudahkan dengan terdapatnya fungsi `pairplot()`. Fungsi ini dapat menggambarkan grafik kombinasi (n,2) dari setiap peubah data dalam bentuk matriks, dengan diagonal utamanya merupakan representasi grafis dari data univariat. Berikut ini bentuk umum penggunaan fungsi `pairplot()`:

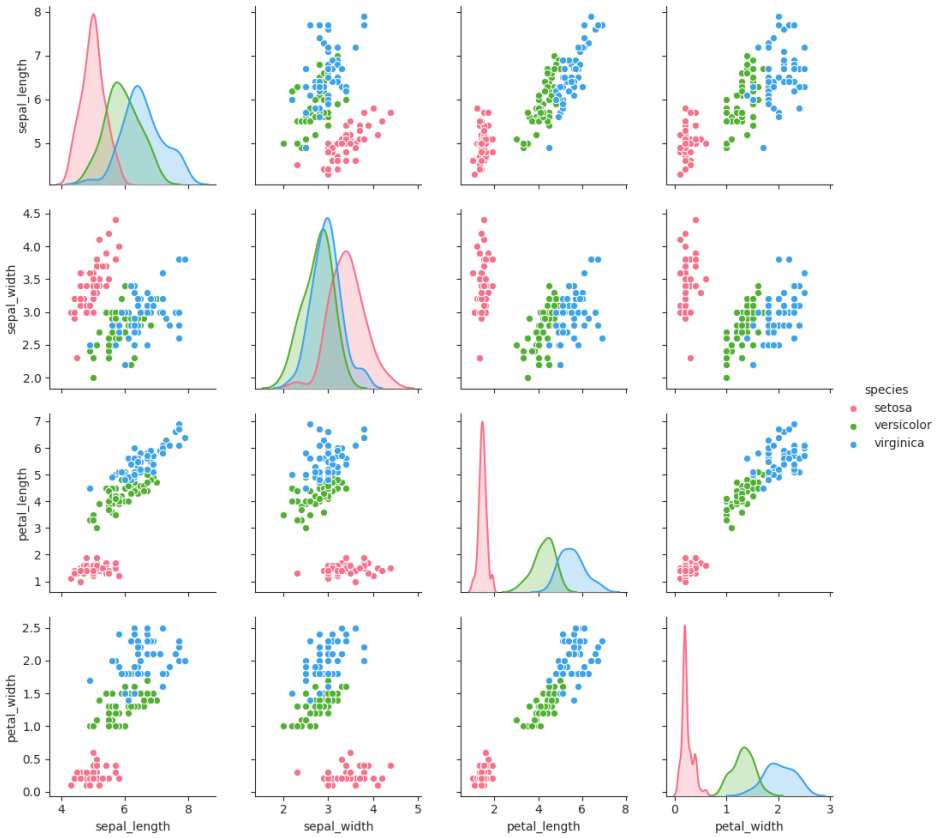
```
sns.pairplot(data,...)
```

Parameter - parameter yang diperhitungkan adalah sebagai berikut:

Jenis Parameter	**Deskripsi**
`data`	Berbentuk *Data Frame*.
`hue`	Pembedaan warna untuk aspek - aspek tertentu dalam data
`palette`	Himpunan warna yang digunakan untuk pemetaan parameter `hue`.
`kind`	Jenis visualisasi data untuk peubah yang tidak identik, misalnya `{'scatter', 'reg'}`.
`diag_kind`	Jenis visualisasi data untuk visualisasi univariat pada diagonal utama dalam bentuk `{'hist', 'kde'}`.

Selain parameter data, parameter - parameter lainnya bersifat opsional.

Selain parameter - parameter tersebut, sesungguhnya fungsi `pairplot()` masih dapat menerima masukkan parameter - parameter lainnya yang tidak akan dibahas dalam tutorial ini karena tidak lazim digunakan dalam analisis konvensional. Berikut ini contoh penerapan `pairplot()`:



Melalui diagram tersebut kita dapat menganalisis perubahan hubungan bivariat pada masing - masing peubah.

9

Visualisasi Data Kategoris

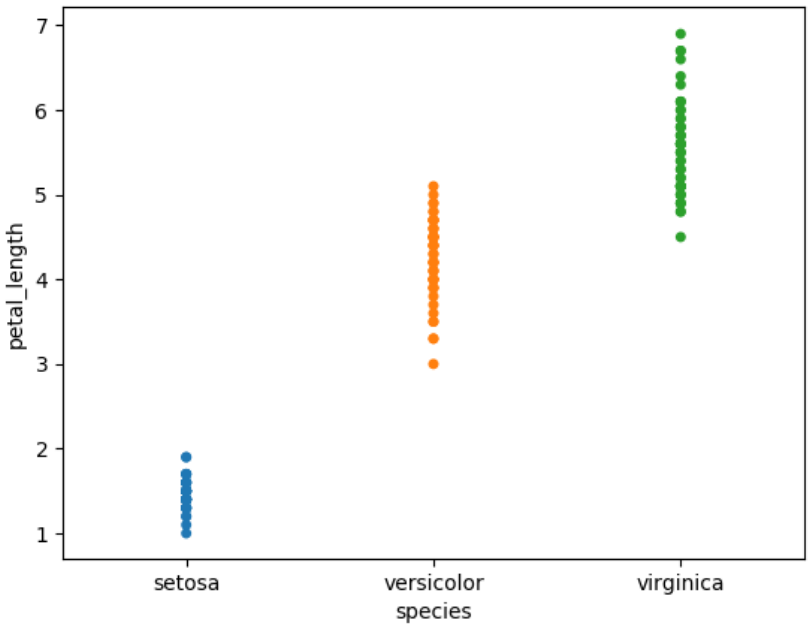
Pada bagian sebelumnya kita telah mempelajari tentang *scatter plot*, grafik *hexbin*, dan grafik KDE. Grafik - grafik tersebut ditujukkan untuk memvisualisasikan data yang bersifat kontinyu, namun tidak dapat digunakan untuk visualisasi data yang bersifat kategoris. Pada bagian ini kita akan membahas fitur - fitur visualisasi data kategoris dengan menggunakan seaborn.

9.1 *Categorical Scatter Plots*

Terdapat dua jenis *scatter plot* yang dapat kita manfaatkan untuk memvisualisasikan data kategoris, yakni `stripplot()` dan `swarmplot()`.

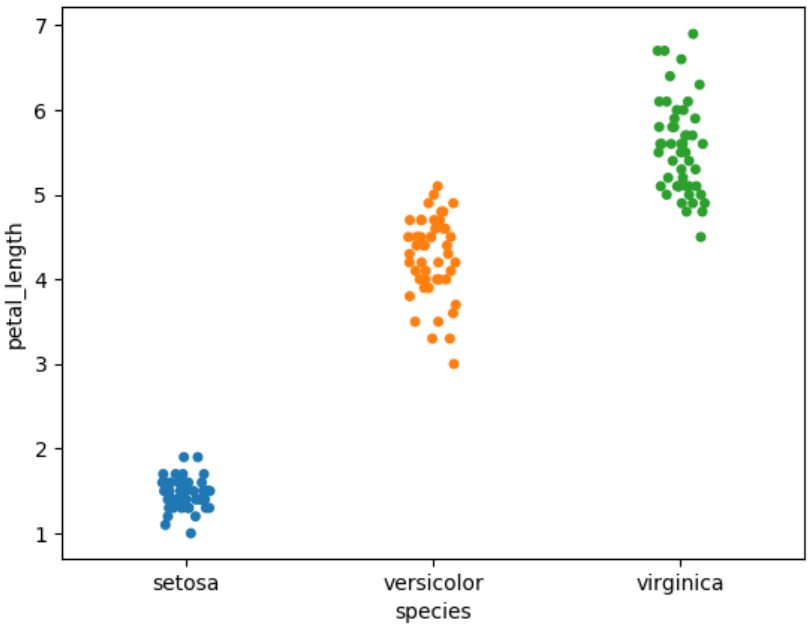
Grafik `stripplot()` digunakan untuk menjelaskan distribusi suatu data, jika salah satu dari peubahnya bersifat kategoris. Grafik ini merepresentasikan data yang diurutkan pada salah satu sumbu. Berikut ini contohnya:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('iris')
6
7 sns.stripplot(x='species', y='petal_length', data=df, jitter=
      False)
8
9 plt.show()
```



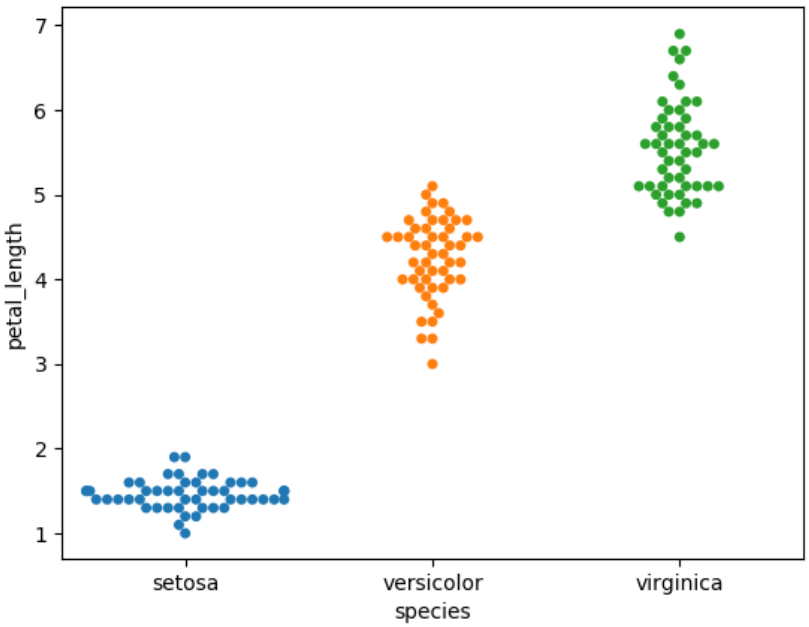
Pada grafik di atas kita dapat dengan jelas melihat perbedaan panjang kelopak bunga pada masing - masing spesies. Namun, salah satu permasalahan utama pada *scatter plot* adalah bertumpuknya titik - titik data, sehingga distribusi data tidak terlihat secara jelas. Guna memperjelas distribusi data, kita umumnya menggunakan parameter `jitter=True` untuk menambahkan *random noise* pada data:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.stripplot(x='species', y='petal_length', data=df, jitter=
    True)

plt.show()
```



Salah satu opsi lain untuk menghindari titik - titik data yang bertumpuk adalah dengan menggunakan grafik `swarmplot()`. Fungsi ini menempatkan setiap titik data dalam *scatter plot* pada masing - masing sumbu kategoris, sehingga menghindari bertumpang tindihnya titik - titik data:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.swarmplot(x='species', y='petal_length', data=df)

plt.show()
```



9.2 Grafik - Grafik Untuk Mengetahui Perbandingan Data Kategoris

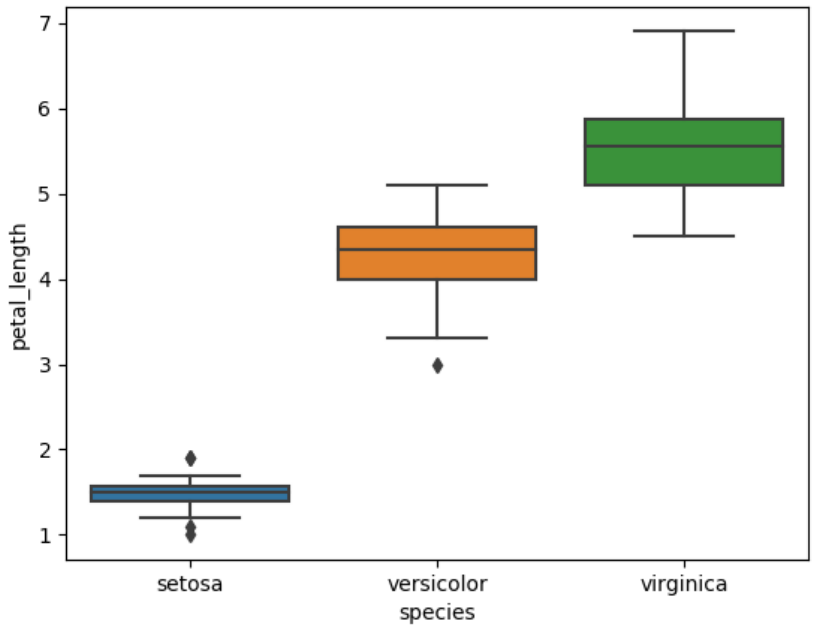
Pada sub-bab sebelumnya kita telah melihat bagaimana perbandingan distribusional antar kategori - kategori data. Namun hal tersebut masih tampak kabur karena yang terlihat hanya persebaran data secara kualitatif. Pada bagian ini kita mencoba menggambarkan distribusi data secara lebih jelas dengan menggunakan *box plot* dan *violin plot*.

9.2.1 *Box Plot*

Box plot merupakan cara paling mudah untuk menggambarkan distribusi data kategoris sesuai dengan jangkauan kuartilnya. *Box plot* mempunyai garis vertikal yang memanjang keluar dari kotak yang dikenal dengan istilah *whisker*. *Whisker* inilah yang menggambarkan variabilitas data di luar batas kuartil atas dan kuartil bawah (atau yang dikenal sebagai pencilan). Maka dari itu, terkadang *box plot* juga dikenal sebagai diagram *box-whisker*. Seluruh pencilan dalam *box plot* divisualisasikan sebagai titik - titik individual di luar kotak. Berikut ini contoh penerapan *box plot* dengan menggunakan seaborn:

```
1 import pandas as pd
```

```
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.boxplot(x='species', y='petal_length', data=df)
plt.show()
```



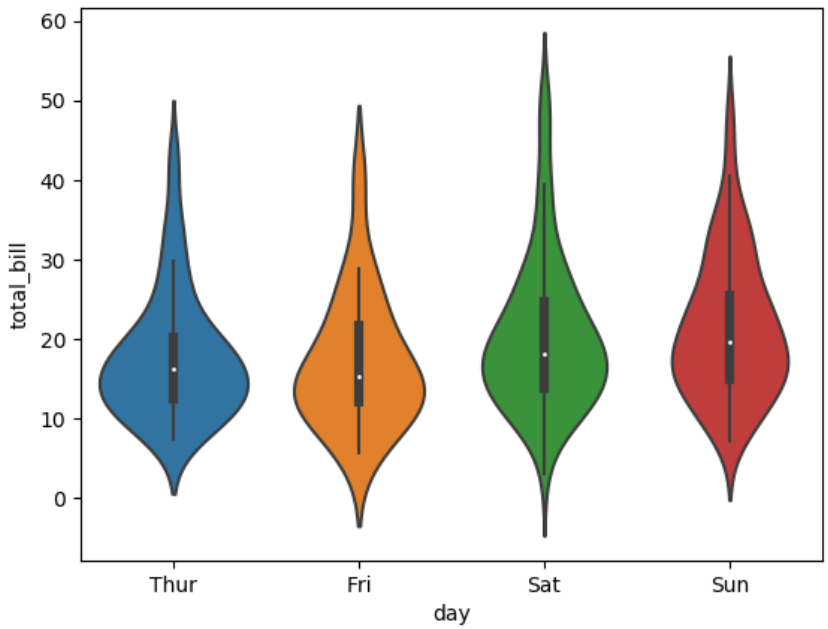
Titik - titik di bagian luar kotak mengindikasikan data pencilan.

9.2.2 *Violin Plot*

Violin plot merupakan kombinasi dari *box plot* dan KDE, sehingga memudahkan para analis data untuk memahami distribusi data kontinyu pada masing - masing kategori. Untuk lebih memahami *violin plot*, kita akan mempleajarinya dengan menggunakan data bawaan `tips` pada seaborn:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('tips')

sns.violinplot(x='day', y='total_bill', data=df)

```

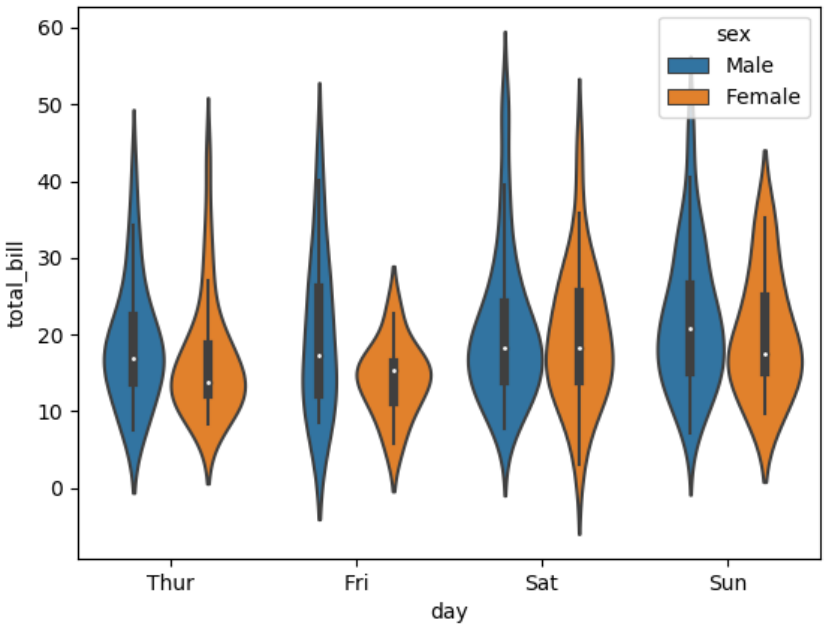
```
plt.show()
```



Pada *violin plot*, nilai -nilai kuartil dan *whisker* dimasukkan di dalam grafik. Oleh karena dalam perhitungannya *violin plot* memasukkan fungsi KDE, maka bagian grafik yang tampak cembung mengindikasikan kepadatan peluang yang besar, pun berlaku sebaliknya pada bagian yang pipih mengindikasikan rendahnya nilai kepadatan peluang. Jangkauan antar kuartil pada *box plot* berada pada wilayah yang sama dengan KDE berkepadatan tinggi pada *violin plot*.

Grafik di atas menunjukkan distribusi total pembayaran pada empat hari dalam seminggu. Kita dapat melakukan eksplorasi data secara lebih mendalam pada grafik tersebut dengan memisahkan total pembayaran tip berdasarkan jenis kelamin, seperti pada contoh berikut ini:

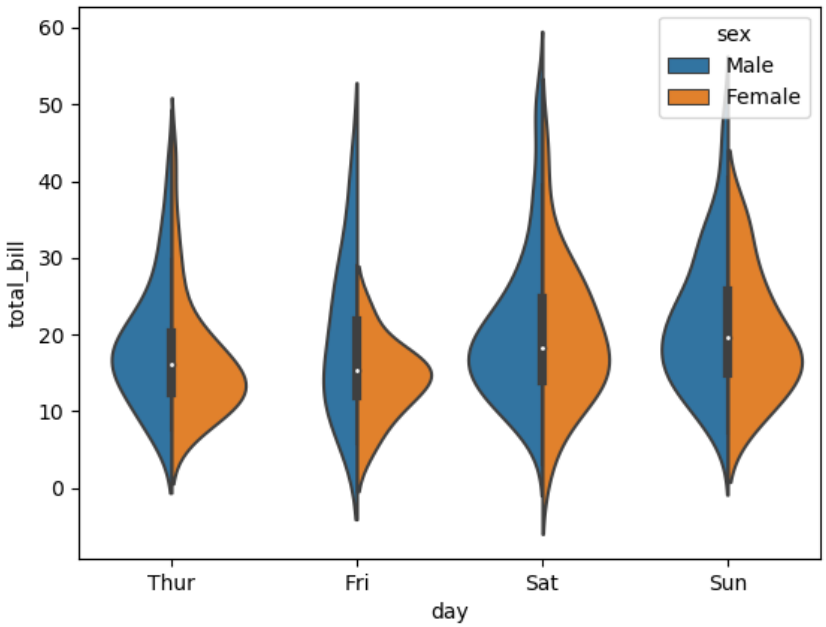
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('tips')

sns.violinplot(x='day', y='total_bill', hue='sex', data=df)

plt.show()
```



Sekarang kita sudah dapat meihat secara jelas distribusi pengeluaran per hari untuk tip berdasarkan jenis kelamin pelanggan. Bukan bermaksud anti-feminis, namun dari data 'bohongan' di atas terlihat bahwa pria menghabiskan lebih banyak uang untuk tip di restoran tersebut dibandingkan dengan wanita.

Karena `hue` data ini hanya terdiri dari dua kategori, yakni pria dan wanita, maka untuk alasan estetika kita dapat membagi *violin plot* menjadi dua bagian agar grafik-nya lebih 'enak' dipandang mata:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('tips')
6
7 sns.violinplot(x='day', y='total_bill', hue='sex', split=True,
      data=df)
8 plt.show()
```


sex
Male
Female
60
50
40
30
20
10
0
total_bill
Thur
Fri
Sat
Sun
day

10

Visualisasi Tendensi Sentral

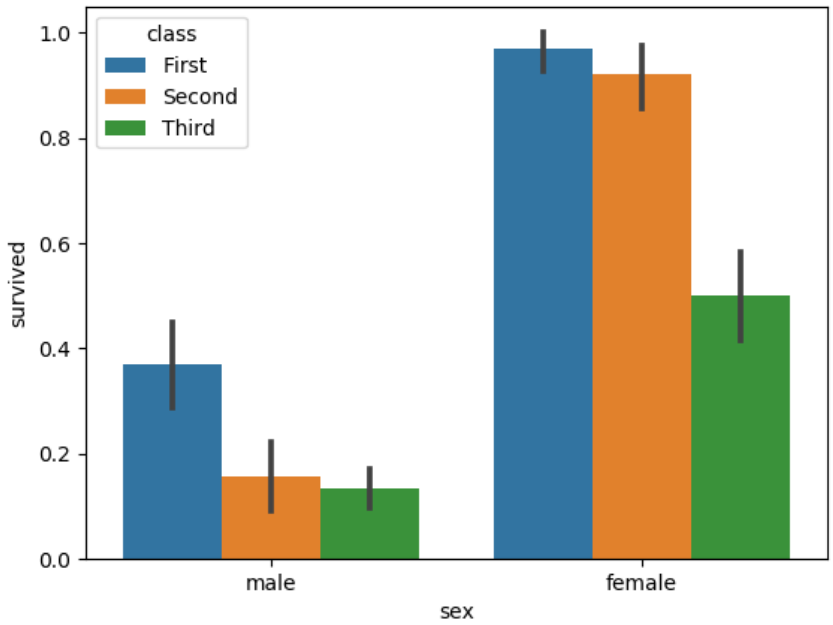
Dalam kebanyakan kasus penelitian, utamanya pada bidang klimatologi, kita harus berhadapan dengan estimasi seluruh distrbusi data. Namun terkadang dalam penelitian - penelitian tertentu, terkadang kita hanya perlu untuk merangkum estimasi sentral dari suatu data. Rata - rata dan median merupakan dua ukuran tendensi sentral yang selalu dikerjakan dalam segala jenis penelitian.

Pada bagian - bagian sebelumnya kita telah melakukan visualisasi untuk seluruh distribusi data, pada bagian ini kita akan membahas visualisasi yang berhubungan dengan estimasi tendensi sentral dari distribusi data secara statistik.

10.1 Diagram Batang

Diagram batang menunjukkan relasi antara peubah kategoris dan peubah kontinyu. Data disajikan dalam format persegi panjang, di mana panjang setiap batang merepresentasikan proporsi data pada masing - masing kategori. Melalui diagram batang kita dapat memperkirakan kisaran nilai tendensi sentral dari suatu data. Berikut ini contoh visualisasi diagram batang dengan menggunakan dataset `titanic` pada seaborn:

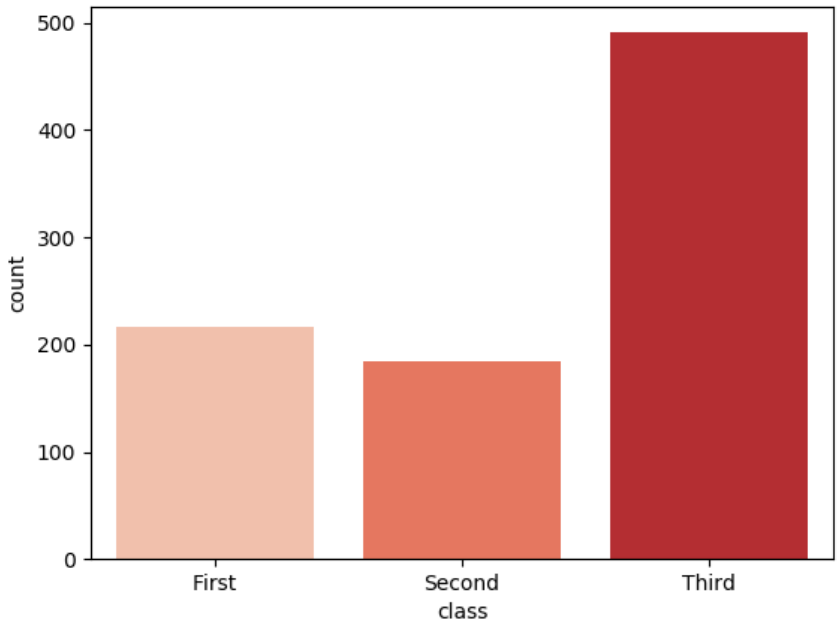
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('titanic')

sns.barplot(x='sex', y='survived', hue='class', data=df)

plt.show()
```



Melalui diagram di atas, kita dapat mengetahui rata - rata pria dan wanita yang selamat dalam kasus tenggelamnya kapal titanic pada masing - masing kelas. Nampak bahwa jumlah wanita yang selamat lebih banyak dari pria, dan jumlah penumpang yang selamat lebih tinggi pada kelas 1.

Untuk mengetahui jumlah eksak masing - masing kategori data, kita dapat menggunakan fungsi `countplot()`:

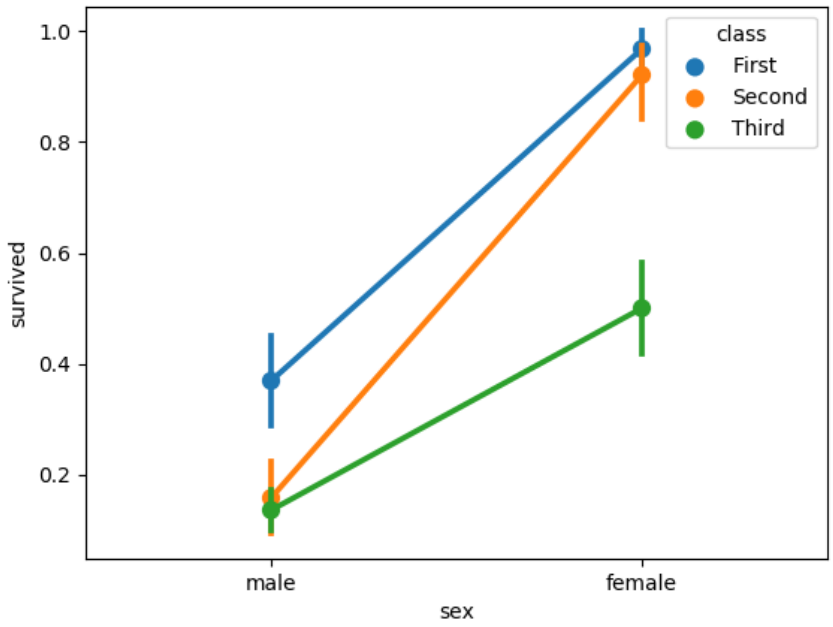
```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
5 df = sns.load_dataset('titanic')
6
7 sns.countplot(x='class', data=df, palette='Reds')
8
9 plt.show()
```



10.2 Diagram Titik

Diagram titik pada dasarnya mempunyai banyak kemiripan dengan diagram batang, hanya gaya visualisasinya saja yang berbeda, yakni dengan merepresentasikan sebaran data pada titik dengan ketinggian tertentu. Perhatikanlah contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('titanic')

sns.pointplot(x='sex', y='survived', hue='class', data=df)

plt.show()
```

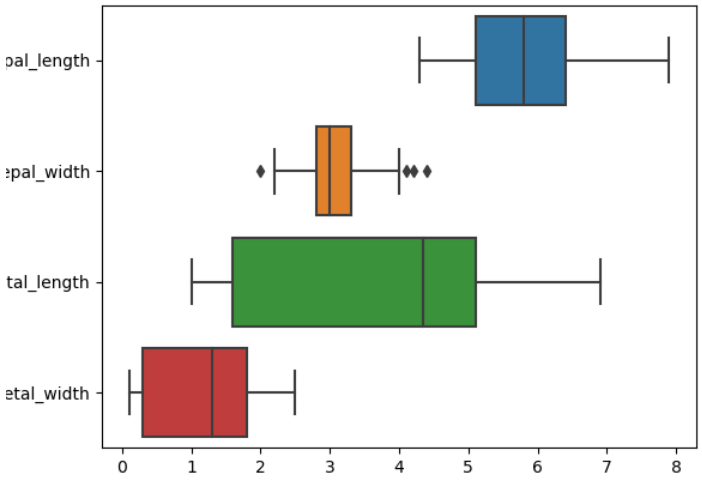

class
First
Second
Third
1.0
0.8
0.6
0.4
0.2
survived
male
female
sex

11

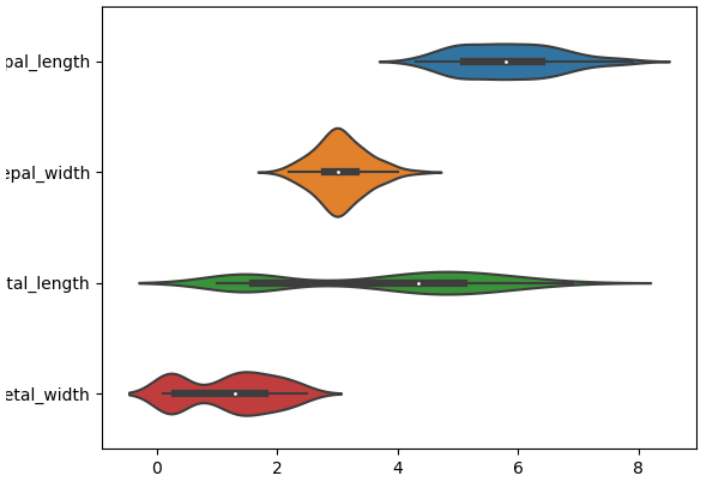
Visualisasi Data Berformat Melebar

Umumnya sebagai analis data kita mengharapkan data yang hendak kita olah berformat memanjang dengan sedikit nilai kosong, namun terkadang kita dihadapkan pada data yang berformat melebar, sehingga mau tidak mau kita harus melakukan pengolahan tanpa melakukan pengubahan dimensi (`reshape()`) karena dikhawatirkan dapat mengubah makna fisis dari data tersebut. Untuk menangani visualisasi data tersebut, kita hanya tinggal mengubah parameter orientasi pada grafik seaborn yang hendak kita plot seperti pada dua contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

sns.boxplot(data=df, orient='h')

plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')
sns.violinplot(data=df, orient='h')
plt.show()
```



12

Visualisasi Multi-Panel

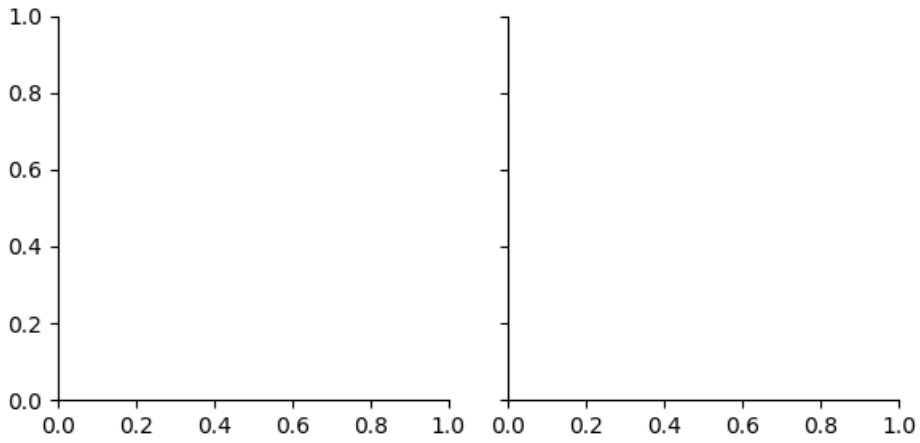
Salah satu pendekatan yang berguna untuk mendapatkan intisari dari suatu data berdimensi sedang adalah dengan cara memvisualisasikan masing - masing peubah pada kanvas yang sama, sehingga memungkinkan kita sebagai analis untuk menarik kesimpulan lebih mendalam. Bagian ini akan menampilkan dua bentuk visualisasi multi panel, yakni *Facet Grid* dan *Pair Grid*

12.1 *Facet Grid*

Facet grid digunakan untuk memvisualisasikan panel grafik berbentuk matriks, di mana setiap kolom dan baris merepresentasikan peubah data. Grafik ini sangat membantu ketika kita hendak menganalisis dua peubah diskrit.

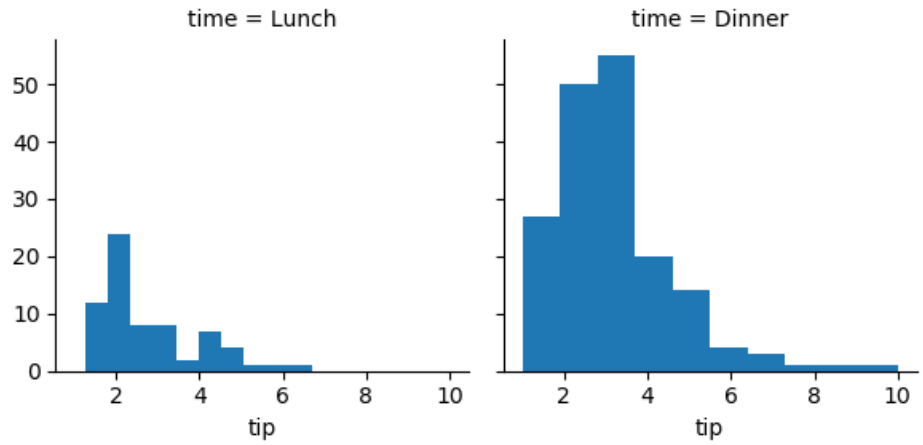
Secara umum *facet grid* tersusun dari tiga dimensi, yakni baris, kolom, dan *hue*. Baris dan kolom berkorespondensi langsung terhadap sumbu data dalam grafik, sementara *hue* dapat dikatakan sebagai sumbu 'z' dari data yang merepresentasikan peubah tertentu. *Facet grid* hanya menerima data bertipe *data frame* sebagai masukan, dan menjadikan nama dari peubah terpilih sebagai judul baris, kolom, dan *hue*. Perlu kalian ingat, bahwa setiap peubah yang dijadikan masukan dalam *facet grid* harus bersifat kategoris. Perhatikanlah contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('tips')

g = sns.FacetGrid(df, col = "time")
plt.show()
```



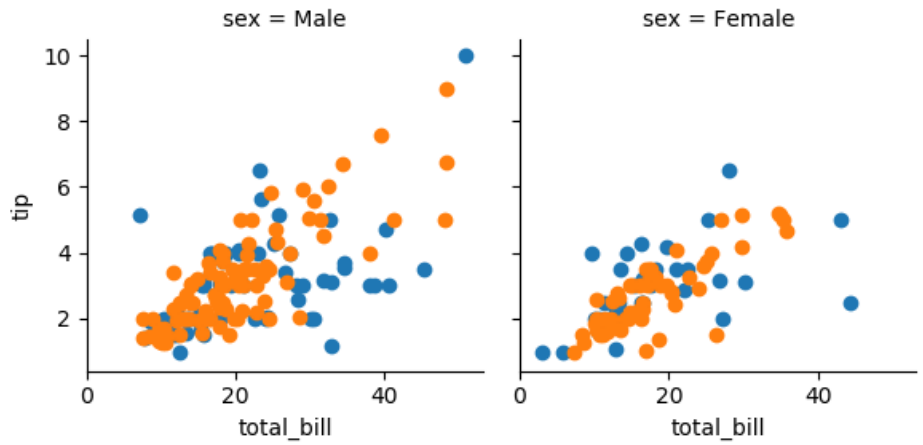
Pada contoh di atas, kita telah menginisialisasi objek *facet grid* kosong pada kanvas. Untuk memvisualisasikan data pada kanvas tersebut umumnya para analis data menggunakan metode `FacetGrid.map()` guna memetakan peubah yang hendak divisualisasikan dalam kanvas kosong tersebut. Dalam kasus ini, kita akan mencoba melihat distribusi tip yang diberikan berdasarkan waktu makan:

```
1  import pandas as pd
2  import  matplotlib.pyplot as plt
3  import seaborn as sns
4
5  df = sns.load_dataset('tips')
6
7  g = sns.FacetGrid(df, col = "time")
8
9  g.map(plt.hist, "tip")
10
11 plt.show()
```



Jumlah plot-nya lebih dari satu karena ita masukkan parameter `col` dengan peubah waktu makan. Berikut ini contoh visual penggunaan *facet grid* dengan masukan banyak peubah:

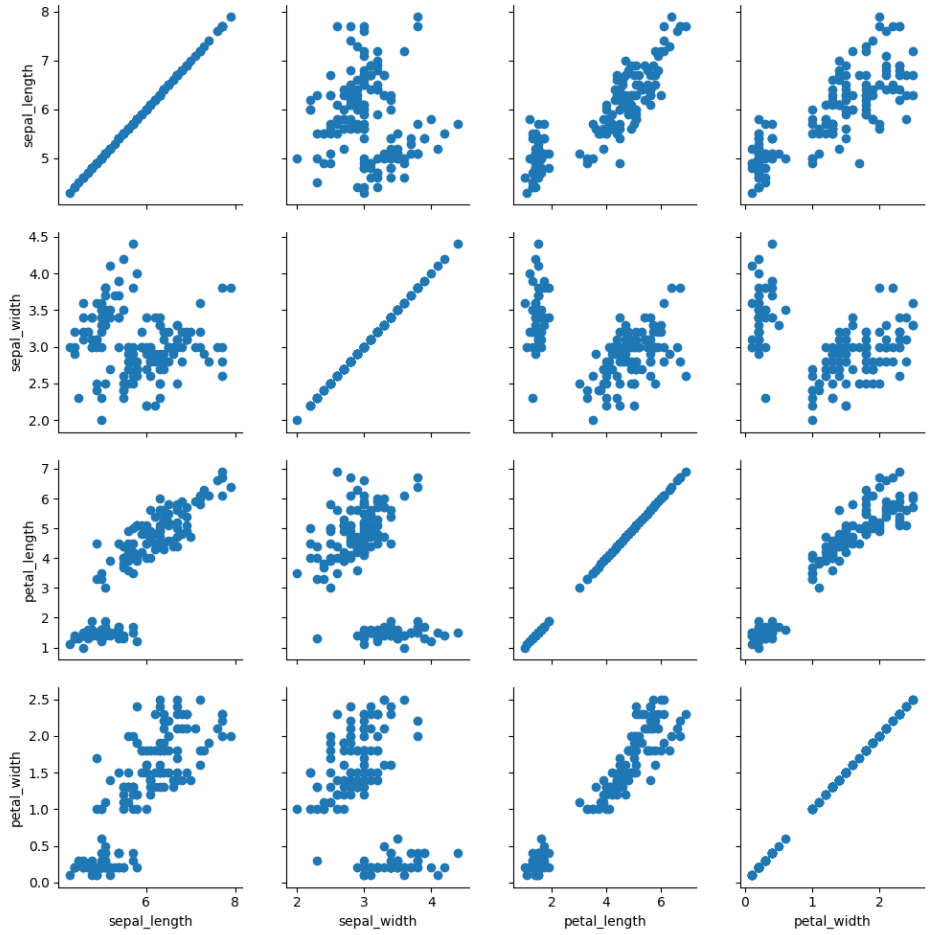
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('tips')

g = sns.FacetGrid(df, col = "sex", hue = "smoker")

g.map(plt.scatter, "total_bill", "tip")

plt.show()
```



12.2 *Pair Grid*

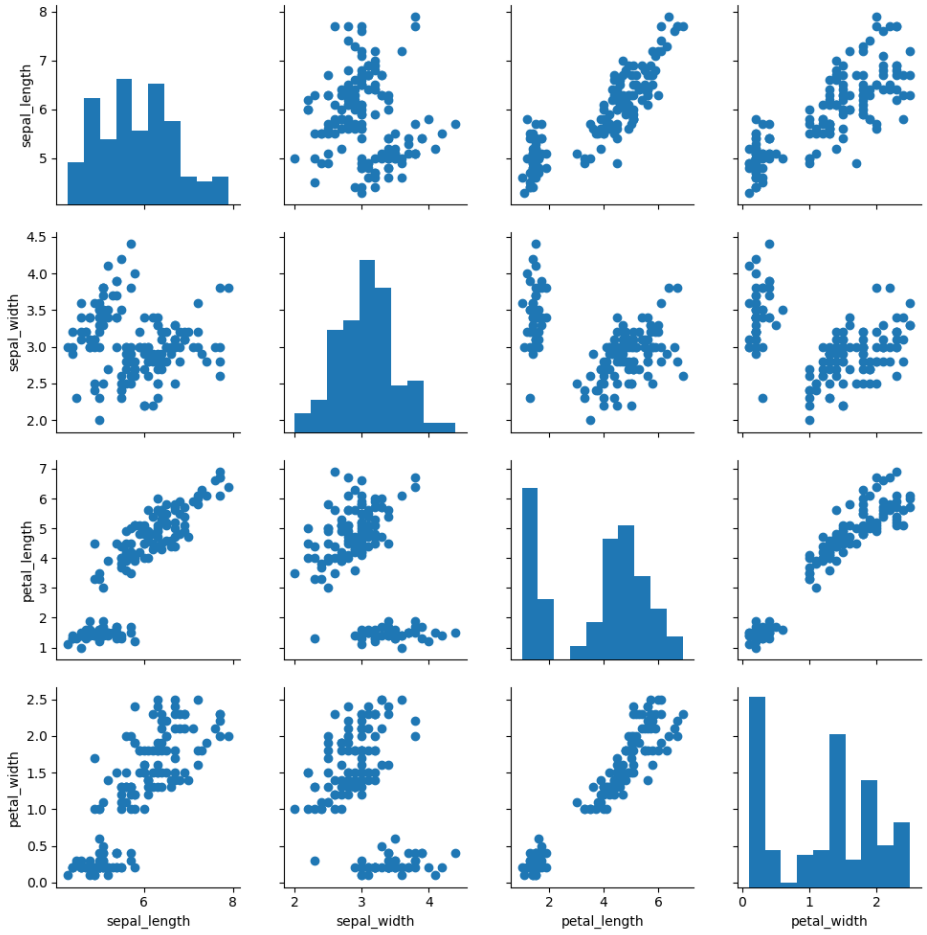
Pair grid memungkinkan kita untuk menggunakan jenis visualisasi yang sama untuk setiap grid dalam sub-plot. Namun tidak sperti *facet grid*, *pair grid* menggunakan peubah yang berbeda untuk setiap peubah yang digunakan dalam sub-plot-nya. *Pair grid* secara visual akan membentuk apa yang dinamakan sebagai matriks *scatter plot*. Cara pengunaan *pair grid* mirip dengan *facet grid*, yakni diawali dengan inisialisasi objek dalam kanvas, kemudian menggunakan metode `map()` untuk memetakan peubah yang hendak kita plot ke dalam objek kosong dalam kanvas. Berikut ini contoh penggunaannya:

```
1 import pandas as pd
2 import seaborn as sns
3 import matplotlib.pyplot as plt
4
5 df = sns.load_dataset('iris')
6
7 g = sns.PairGrid(df)
8
9 g.map(plt.scatter)
10
11 plt.show()
```



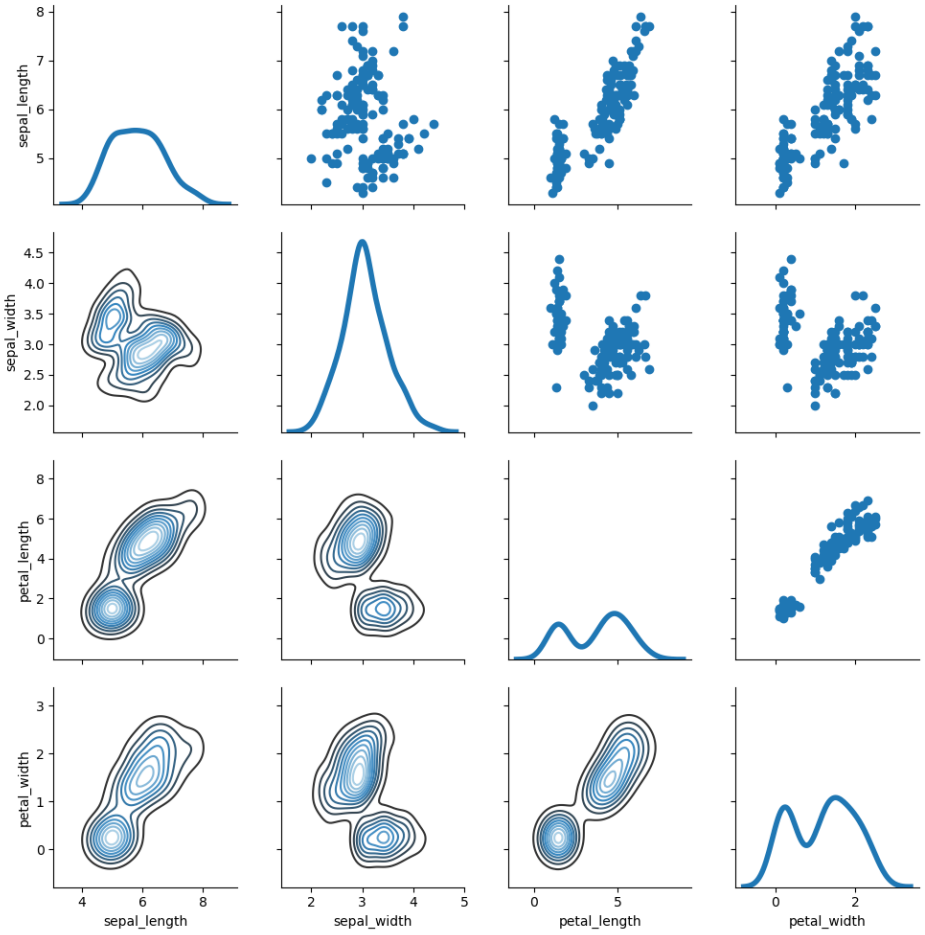
Selain itu, kita juga dapat mengubah tipe visualisasi pada diagonal matriks-nya, seperti dalam contoh ini saya menunjukkan distribusi univariat pada diagonal-nya:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

g = sns.PairGrid(df)

g.map_diag(plt.hist)

g.map_offdiag(plt.scatter)

plt.show()
```



Kita juga dapat mengubah segitiga atas dan segitiga bawah pada matriks dengan tipe visualisasi yang berbeda:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('iris')

g = sns.PairGrid(df)

g.map_upper(plt.scatter)

g.map_lower(sns.kdeplot, cmap='Blues_d')

g.map_diag(sns.kdeplot, lw=4, legend=False)

plt.show()
```


sepal_length
sepal_width
petal_length
petal_width
sepal_length
sepal_width
petal_length
petal_width

13

Visualisasi Regresi Linier

Kerap kali sebagai analis data kita diminta untuk menganalisis hubungan kuantitatif antar dua peubah. Untuk memenuhi hal ini, kita dapat melakukannya dengan visualisasi melalui metode regresi. Dengan menerapkan model regresi ini, kita dapat memeriksa multikolinieritas antar peubah yang hendak kita cari hubungannya. Bab ini akan membahas tentang visualisasi melalui model regresi dengan menggunakan seaborn.

13.1 Fungsi - Fungsi Untuk Memvisualisasikan Regresi Linier

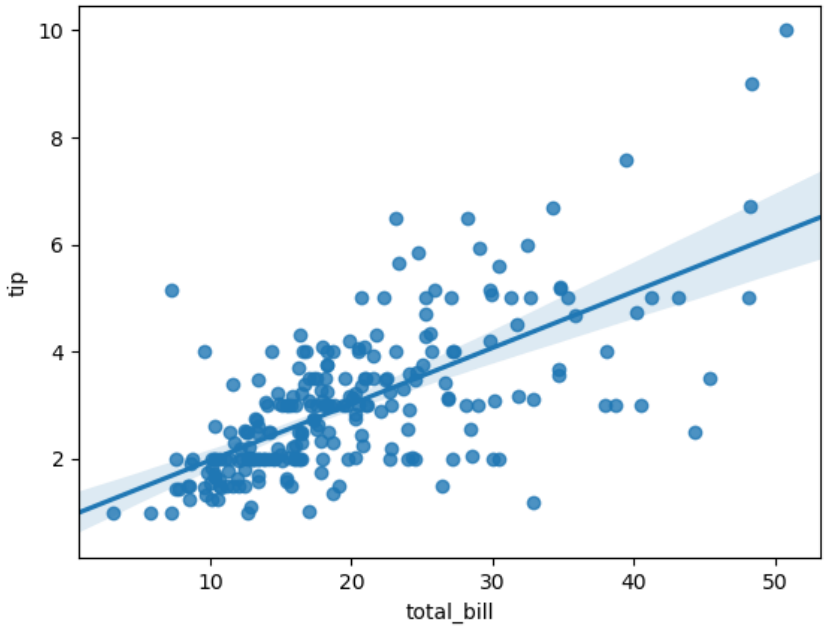
Terdapat dua buah fungsi dalam seaborn yang dapat kita manfaatkan untuk memvisualisasikan regresi linier, yakni `regplot()` dan `lmplot()`, berikut ini perbedaan antar keduanya:

`regplot()`	Menerima berbagai jenis data masukkan, seperti *array* numpy sederhana, *series*, hingga *data frame*. Hanya dapat memvisualisasikan regresi linier.
`lmplot()`	Umumnya bekerja dengan lebih baik pada data berformat panjang (*long-form format*). Dapat memproses regresi polinomial.

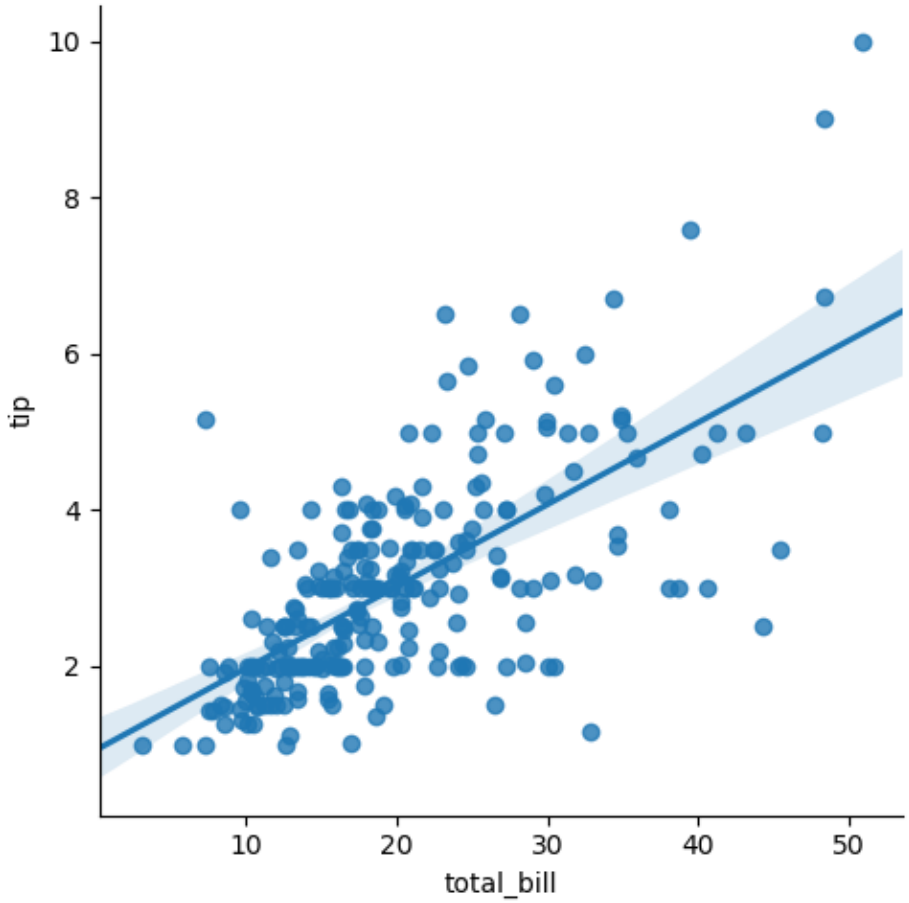
Untuk melihat perbedaannya secara visual, perhatikanlah contoh berikut ini:

```
1 import pandas as pd
2 import matplotlib.pyplot as plt
3 import seaborn as sns
4
```

```
5  df = sns.load_dataset('tips')
6
7  sns.regplot(x='total_bill', y='tip', data=df)
8
9  sns.lmplot(x='total_bill', y='tip', data=df)
10
11 plt.show()
```



Regresi Linier dengan fungsi regplot()

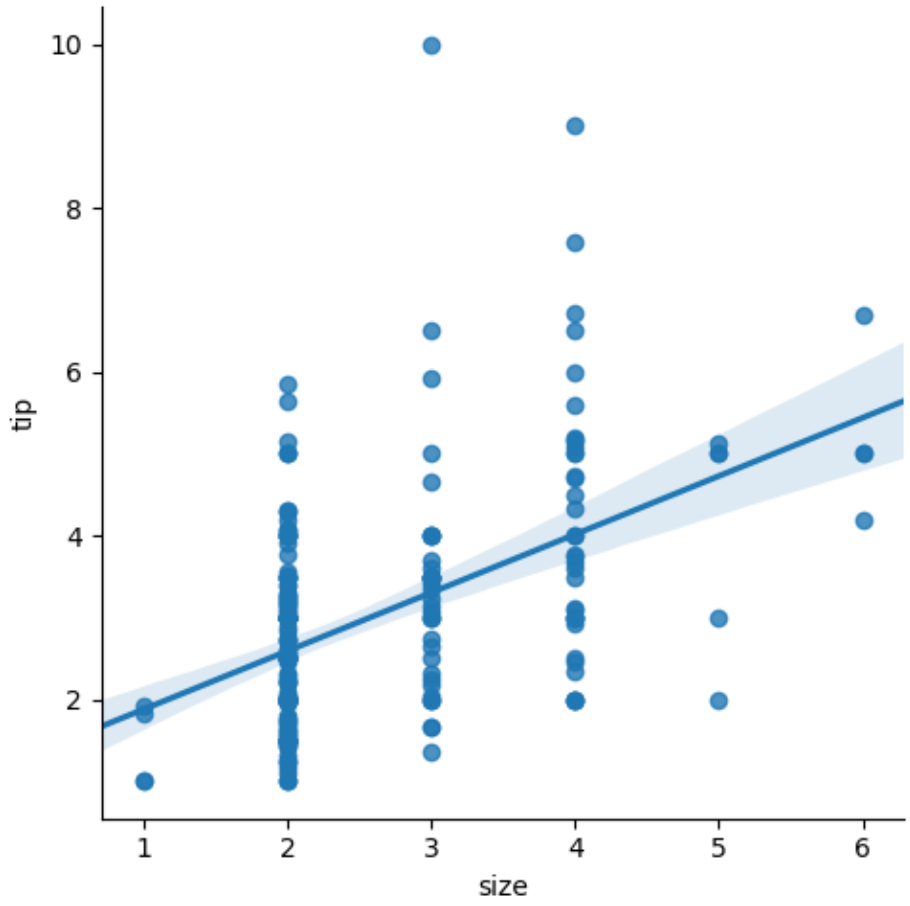


Regresi Linier dengan fungsi lmplot()

Kedua grafik tersebut tampak identik, hanya saja terdapat perbedaan dari segi ukuran saja. Jadi kesimpulannya untuk visualisasi regresi linier, kita dapat memilih untuk menggunakan `regplot()` atau `lmplot()` sesuai kehendak kita.

Kita juga dapat melakukan pemodelan regresi linier pada dua peubah, di mana salah satu peubahnya bersifat diskrit:

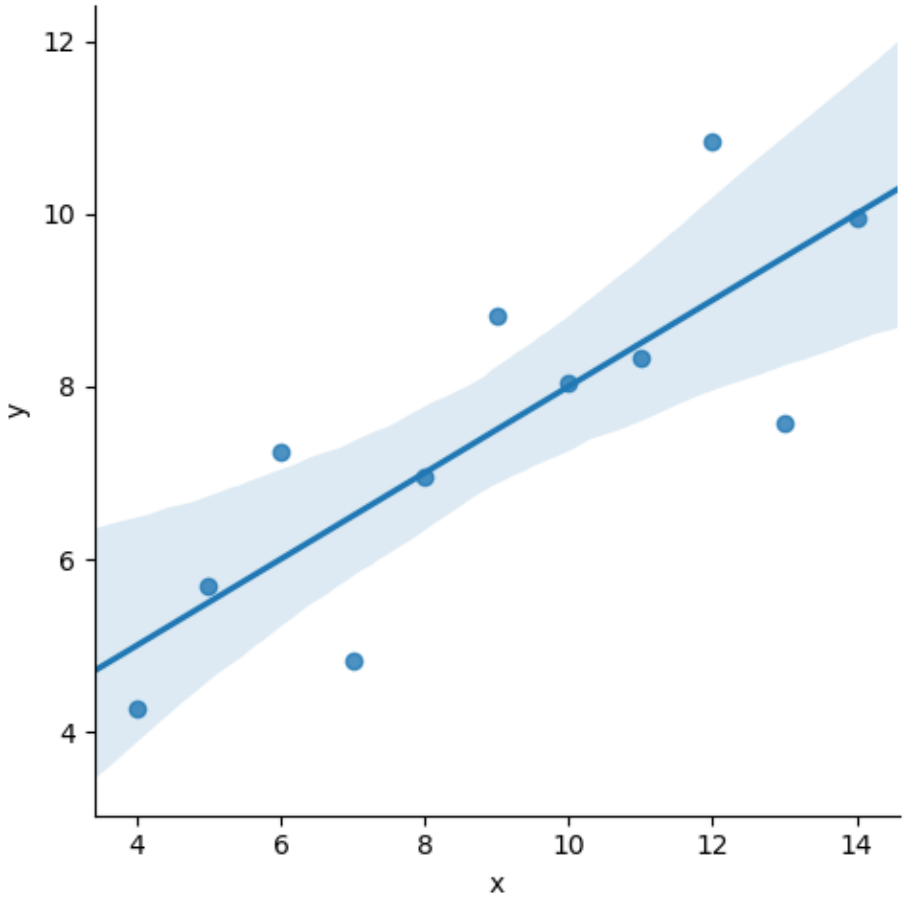
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('tips')

sns.lmplot(x='size', y='tip', data=df)

plt.show()
```



13.2 *Fitting* Model Non-linier

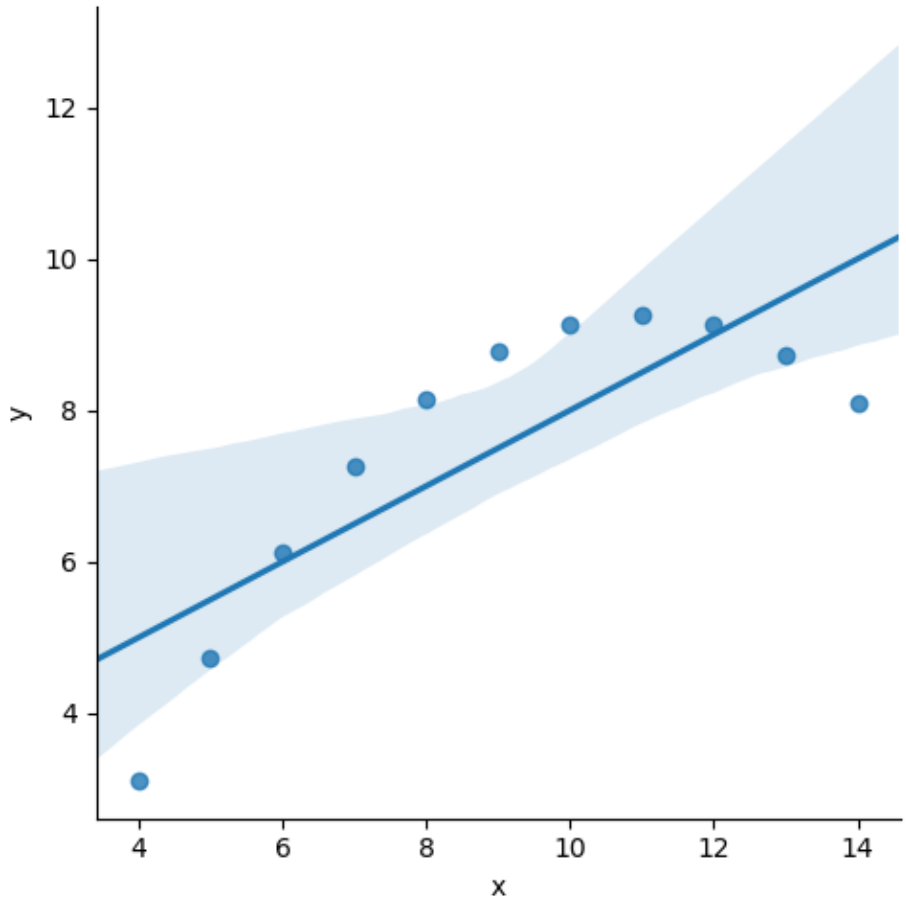
Model regresi linier tidak selalu cocok untuk diterapkan pada semua kasus. Terkadang kita sebagai analis data harus memikirkan pendekatan non-linier ketika dihadapkan dengan kasus semacam ini. Perhatikanlah contoh - contoh pengolahan data `anscombe` berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('anscombe')

sns.lmplot(x='x', y='y', data=df.query("dataset == 'I'"))
plt.show()
```



Pada kasus ini, kita beruntung karena model regresi linier cocok untuk diterapkan pada data ini dengan nilai variansi yang relatif kecil. Coba kita lihat contoh berikutnya dengan data bernilai deviasi lebih besar:

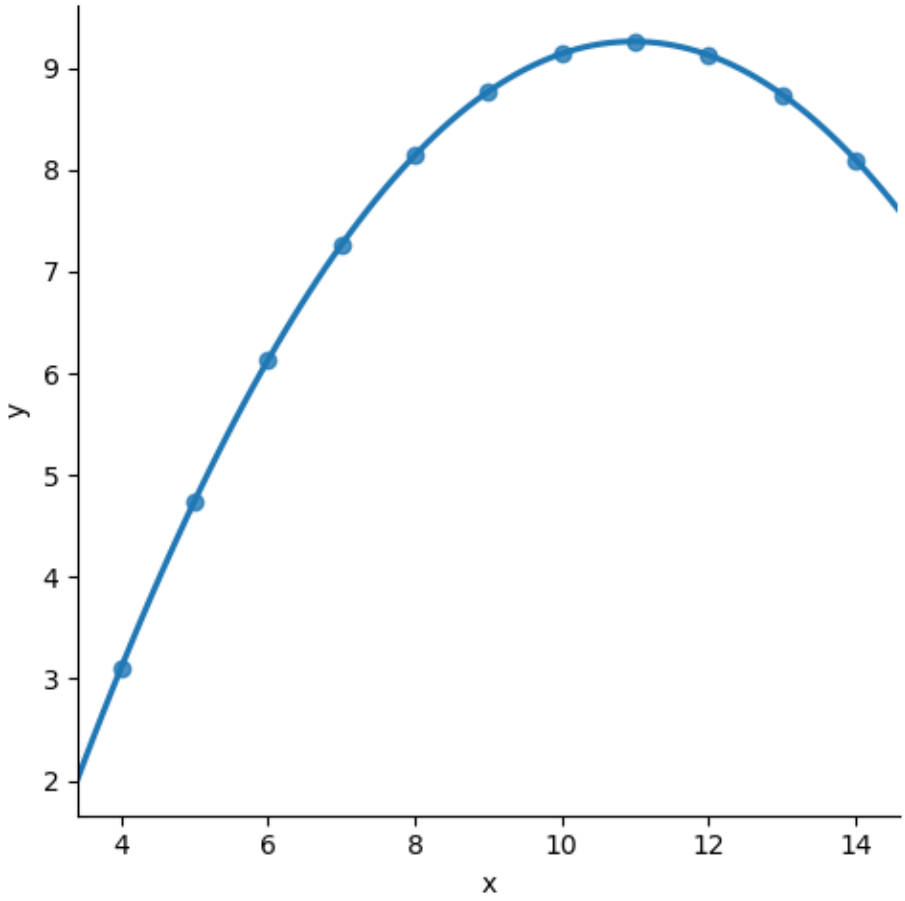
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('anscombe')

sns.lmplot(x='x', y='y', data=df.query("dataset == 'II'"))

plt.show()
```



Nampaknya data tidak 'pas' untuk dimodelkan dengan regresi linier, maka kita sebagai analis data harus mencoba metode pemodelan non-linier, yakni regresi polinomial orde 2:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('anscombe')

sns.lmplot(x='x', y='y', data=df.query("dataset == 'II'"), order
    =2)

plt.show()
```

Dan hasilnya dapat dilihat, bahwa model yang kita terapkan sangat sesuai dengan data:


9
8
7
6
5
4
3
2
y
4
6
8
10
12
14
x

14

Visualisasi Matriks

Terdapat dua jenis visualisasi matriks yang umum digunakan dalam seaborn, yakni *heatmap* dan *clustermap*. Pada bagian ini kita akan membahas keduanya dengan menggunakan dua data bawaan dari seaborn, yakni `tips` dan `flights`. Karena kita belum melihat dataset `flights` yang sebetulnya hanya data bulanan jumlah penumpang pesawat dari tahun 1949 hingga 1960, maka ada baiknya kita lihat dulu datanya:

```
import seaborn as sns

df = sns.load_dataset('flights')

print(df.head())
```

Hasilnya adalah:

```
year     month  passengers
0  1949   January         112
1  1949  February         118
2  1949     March         132
3  1949     April         129
4  1949       May         121
```

14.1 *Heatmap*

Guna memvisualisasikan data dengan menggunakan *heatmap*, kita memerlukan data yang berformat matriks. Artinya adalah jumlah peubah pada indeks sesuai dengan jumlah peubah pada kolom data. Untuk membentuk bentuk data yang sesuai, kita umumnya menggunakan metode korelasi data atau tabel pivot. Berikut kita akan mencoba dulu dengan menggunakan metode korelasi data pada dataset `tips`:

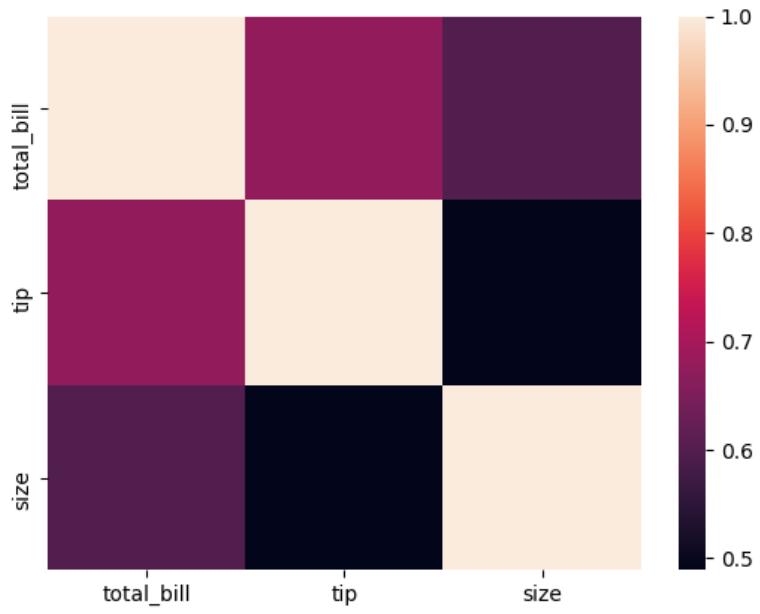
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```
df = sns.load_dataset('tips')

df = df.corr()

print(df)

sns.heatmap(df)

plt.show()
```

Inilah tabel korelasi yang dihasilkan:

```
total_bill       tip      size
total_bill    1.000000  0.675734  0.598315
tip           0.675734  1.000000  0.489299
size          0.598315  0.489299  1.000000
```

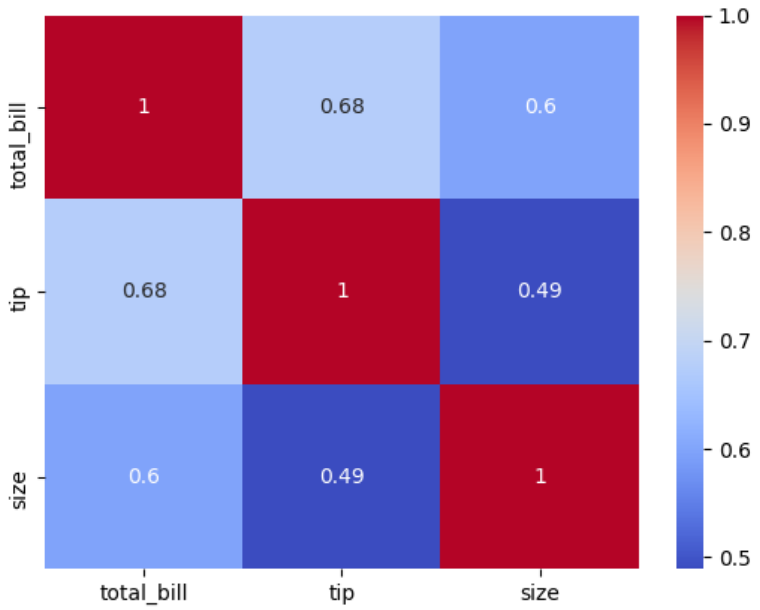
Berikut ini visual *heatmap*-nya:



Kita juga dapat menampilkan nilai setiap elemen dalam matriks dengan menambahkan argumen `annot=True`, serta mengganti warna pada plot sebagai berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

```

```
df = sns.load_dataset('tips').corr()

sns.heatmap(df, annot=True, cmap='coolwarm')

plt.show()
```



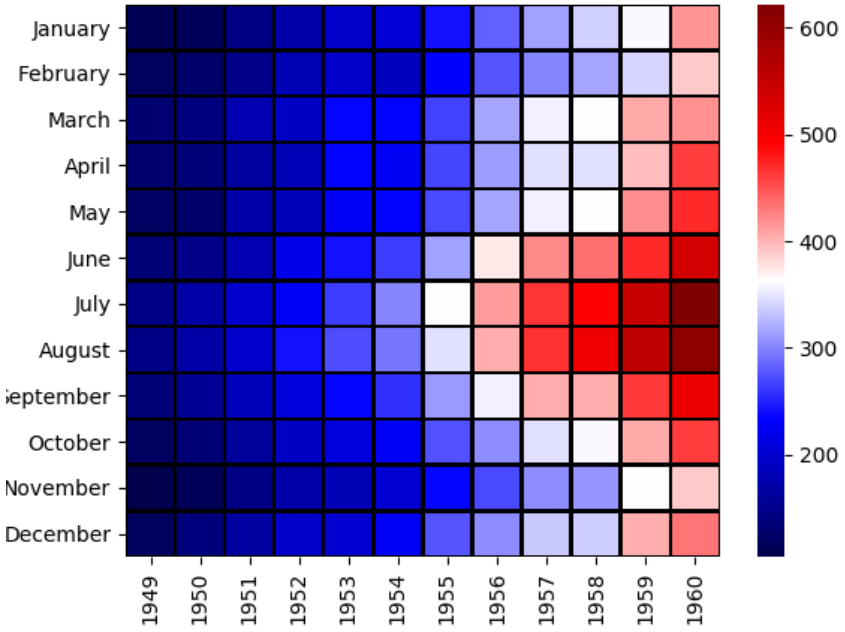
Sekarang kita akan mencoba menggunakan tabel pivot (*kan* orang Indonesia pada *jago* pakai **Excel** jadi ga usah saya *jelasin* lagi hehe). Dalam konteks ini, kita akan menggunakan dataset `flights` dengan menjadikan bulan sebagai baris dan tahun sebagai kolom:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('flights')

df = df.pivot_table(index='month', columns='year', values='
    passengers')

print(df)

sns.heatmap(df, cmap='seismic', linecolor='black', lw=1)

plt.show()
```

Berikut ini tabel pivot yang kita hasilkan:

```
year         1949  1950  1951  1952  1953  ...  1956  1957  1958
     1959  1960
month                                       ...
January       112   115   145   171   196  ...   284   315   340
       360   417
February      118   126   150   180   196  ...   277   301   318
       342   391
March         132   141   178   193   236  ...   317   356   362
       406   419
April         129   135   163   181   235  ...   313   348   348
       396   461
May           121   125   172   183   229  ...   318   355   363
       420   472
June          135   149   178   218   243  ...   374   422   435
       472   535
July          148   170   199   230   264  ...   413   465   491
       548   622
August        148   170   199   242   272  ...   405   467   505
       559   606
September     136   158   184   209   237  ...   355   404   404
       463   508
October       119   133   162   191   211  ...   306   347   359
       407   461
November      104   114   146   172   180  ...   271   305   310
       362   390
December      118   140   166   194   201  ...   306   336   337
       405   432

[12 rows x 12 columns]
```

Hasil *heatmap*-nya adalah sebagai berikut:

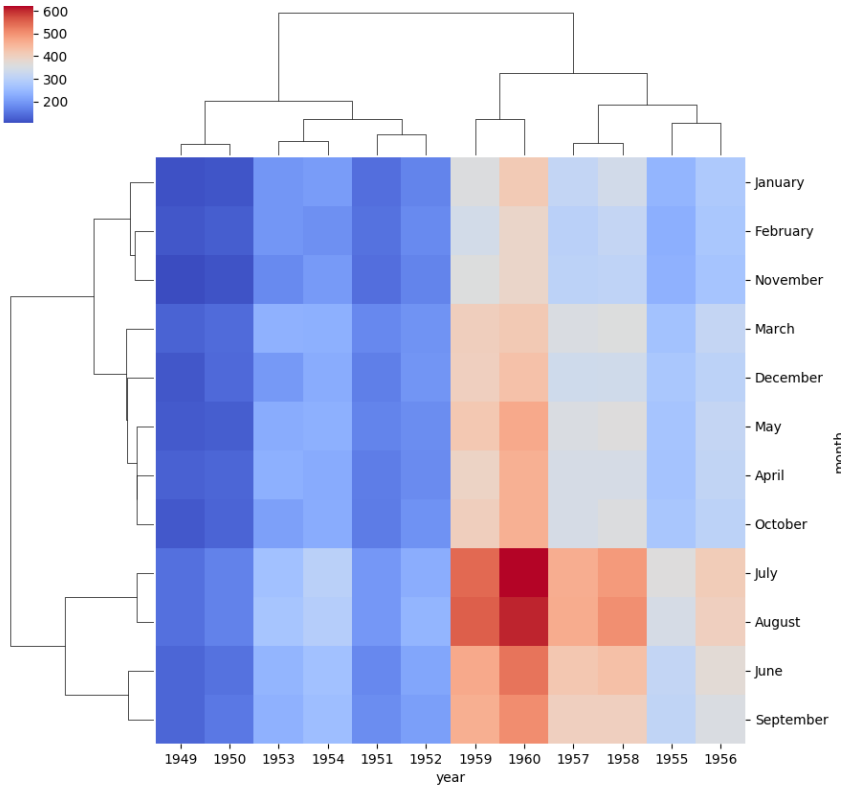


Melalui grafik di atas, kita dapat melihat bahwa jumlah penumpang pesawat paling banyak pada bulan - bulan musim panas di belahan bumi utara (karena konteks data ini mengambil contoh kasus di Amerika Serikat) dan semakin bertambah banyak seiring dengan proses keberjalanan waktu.

14.2 *Clustermap*

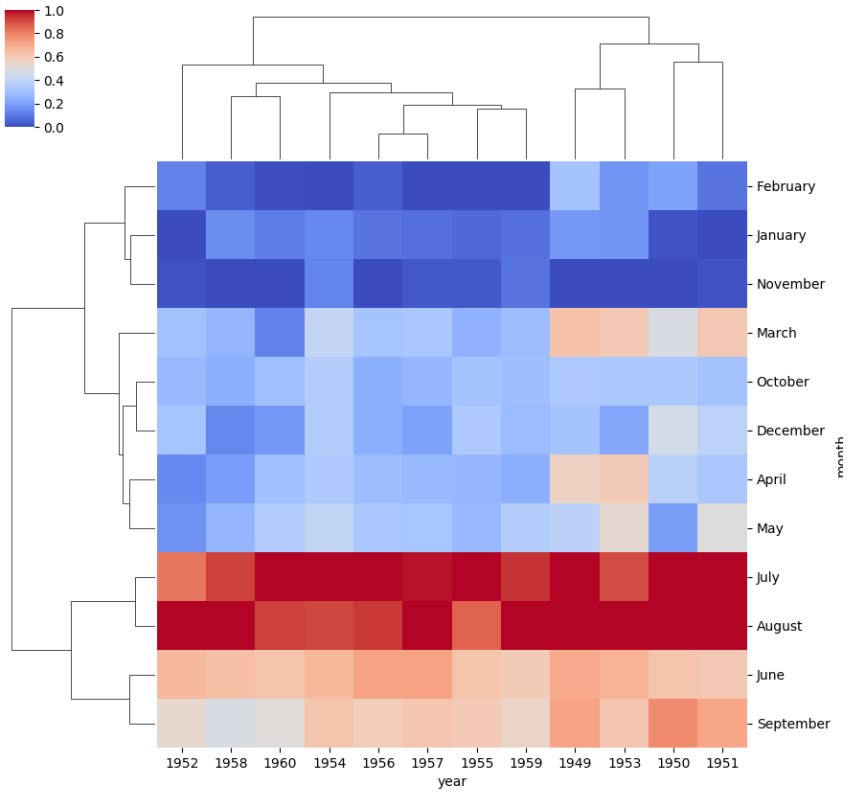
Untuk melanjukan proses analisis pengelompokan jumlah penumpang pesawat berdasarkan bulan dan tahun pada contoh di atas, kita dapat menerapkan **algoritma *clustering* hierarkis** dan memvisualisasikannya dengan menggunakan *clustermap* seperti contoh berikut ini:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('flights').pivot_table(index='month',
    columns='year', values='passengers')

sns.clustermap(df, cmap='coolwarm')

plt.show()
```



Karena skalanya masih dalam skala jumlah penumpang pesawat, kita dapat memperjelasnya dengan melakukan standarisasi skala dari 0 hingga 1:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df = sns.load_dataset('flights').pivot_table(index='month',
    columns='year', values='passengers')

sns.clustermap(df, cmap='coolwarm', standard_scale=1)

plt.show()
```



Perhatikanlah bahwa kini tahun dan bulan pada grafik tidak lagi berurutan, melainkan dikelompokan berdasarkan kesamaan jumlah penumpangnya. Nampak kemiripan antara bulan Juli dan Agustus sebagai bulan dengan jumlah penumpang terbanyak, dan hal ini masuk akal karena sedang berlangsung liburan musim panas di Amerika pada bulan - bulan itu.

SERI KOMPUTASI

Seaborn merupakan salah satu pustaka visual Python yang berlandaskan pada matplotlib. Seaborn menyediakan antar-muka tingkat tinggi untuk menangani permasalahan terkait visualisasi data secara statistik agar tampak lebih menarik.

Tutorial ini mencoba menjabarkan dasar – dasar penggunaan seaborn dengan cara mengupas satu demi satu jenis visualisasi data yang umumnya dikerjakan dalam pustaka seaborn. Diharapkan dengan hadirnya tutorial ini dapat menjadi pelengkap bahan praktikum matakuliah analisis data dan/atau statistika di jurusan – jurusan humaniora, rekayasa, dan ilmu pengetahuan alam di Indonesia dalam era *Big Data* dewasa ini.

Sandy Hardian Susanto Herho lahir di Cirebon pada tanggal 13 Maret 1993. Pendidikan dasar dan menengah ia selesaikan di Indramayu dan Jakarta. Ia menamatkan pendidikan tinggi dengan gelar sarjana sains bidang Meteorologi dengan spesialisasi pada bidang sejarah iklim dari Institut Teknologi Bandung pada tahun 2017. Saat ini ia lebih menekuni dunia penelitian lepas dengan minat utama pada kajian iklim regional Benua Maritim.

DOI 10.5281/zenodo.3245514

WEATHER AND CLIMATE PREDICTION LABORATORY

Labtek XI Lantai 2, Institut Teknologi Bandung.

Jl. Ganesha 10 Bandung. Jawa Barat 40132.
Telp : +62 22 2500494.

http://www.weather.meteo.itb.ac.id/