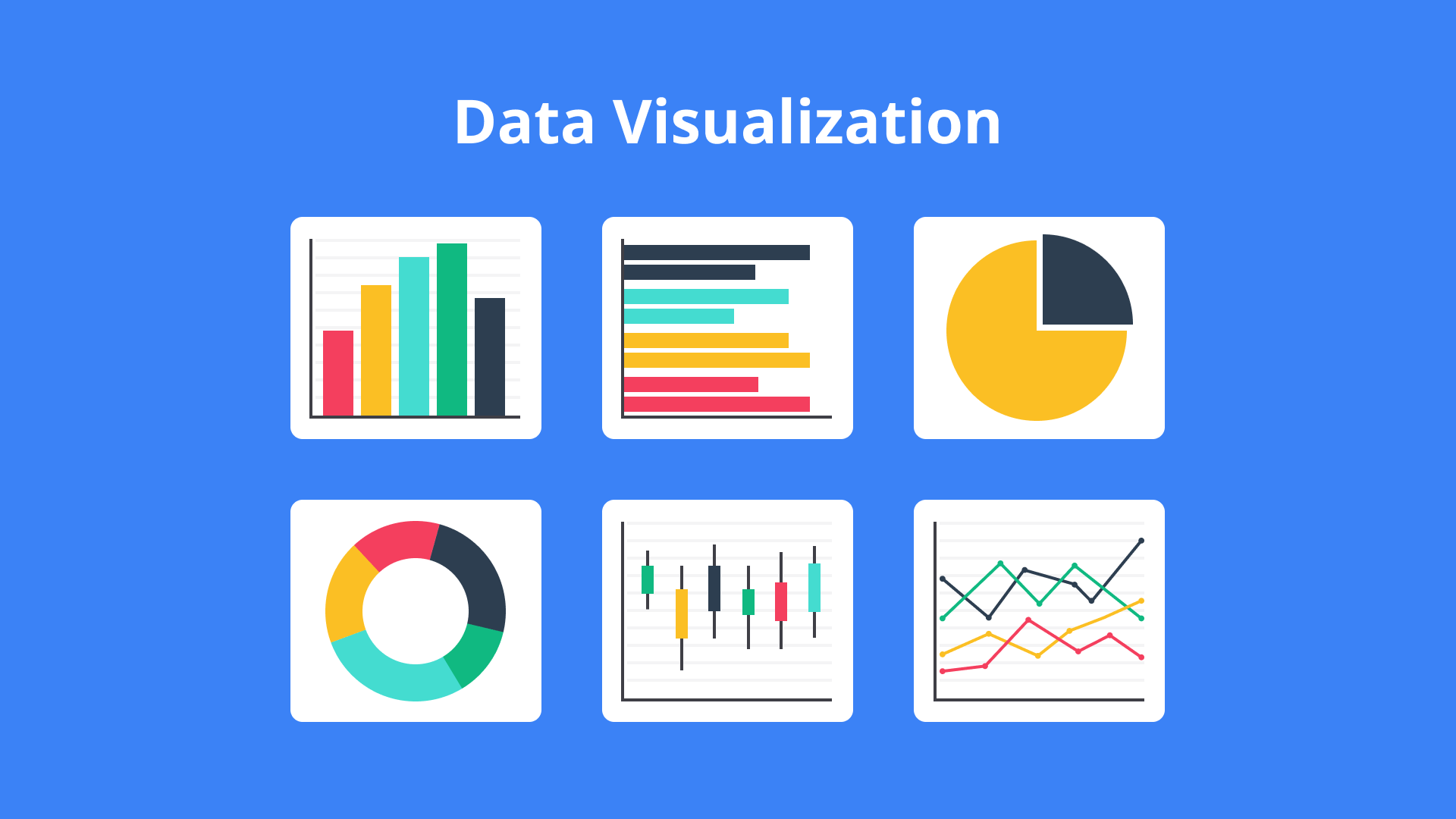
**Pengenalan Data Visualization**

*Yey*, akhirnya kita masuk ke pembahasan data visualization. Pembahasan ini merupakan salah satu pembahasan paling esensial dalam analisis data. Selain itu, kemampuan untuk memvisualisasikan data merupakan salah satu keterampilan penting yang harus dimiliki oleh seorang praktisi data.



Apakah Anda masih ingat dengan materi siklus analisis data. Pada siklus tersebut, data visualization merupakan tahapan yang harus kita lakukan sebelum membuat kesimpulan dan mengomunikasikan (*draw conclusion* & *communicate*) hasil dari proses analisis yang telah dilakukan. Hal ini dilakukan untuk mempermudah kita dalam memahami data dan membuat sebuah kesimpulan yang andal. Selain itu, visualisasi data juga dapat membantu kita dalam menyampaikan cerita atau temuan dari hasil analisis data kepada orang lain atau stakeholder.

Oke, dari tadi kita telah membahas tentang data visualization atau visualisasi data. Mungkin di benak Anda mulai muncul pertanyaan, sebenarnya apa sih visualisasi data itu?

Sederhananya, visualisasi data merupakan cara kita dalam menyajikan data dalam bentuk visual. Hal ini dilakukan untuk mempermudah kita dan orang lain dalam memahami data tersebut. Selain itu, visualisasi data yang baik juga akan sangat membantu kita dalam menyampaikan story dan pesan dari sebuah data.

Nah, pada beberapa materi ke depan, kita akan membahas berbagai teori dan best practice yang dapat diterapkan untuk menghasilkan visualisasi data yang baik dan benar. Pada bagian awal kita akan membahas tentang berbagai prinsip dalam visualisasi data. Pada bagian ini, kita akan berkenalan dengan berbagai ketentuan dan prinsip yang harus diterapkan guna menghasilkan visualisasi data yang efektif.

Setelah memahami seluruh prinsip desain tersebut, kita akan membahas cara penggunaan berbagai tool untuk membuat visualisasi data baik berupa univariate, bivariate, maupun multivariate. Selain itu, kita juga akan membahas apa saja yang harus dipertimbangkan ketika membuat visualisasi data untuk proses explanatory analysis.

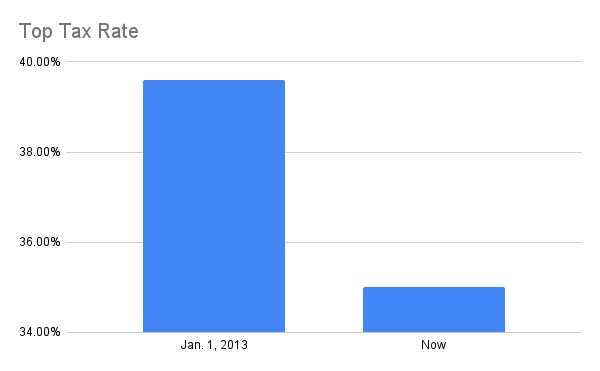
**Prinsip-Prinsip dalam Visualisasi Data**

Seperti yang telah disampaikan di awal, pada beberapa materi ke depan, kita akan fokus pada pembahasan visualisasi data. Ketika berbicara tentang visualisasi data, kita tidak hanya membahas cara menyajikan data ke dalam bentuk visual, melainkan juga harus memahami berbagai prinsip dasar dalam membuat visualisasi data yang baik dan benar.

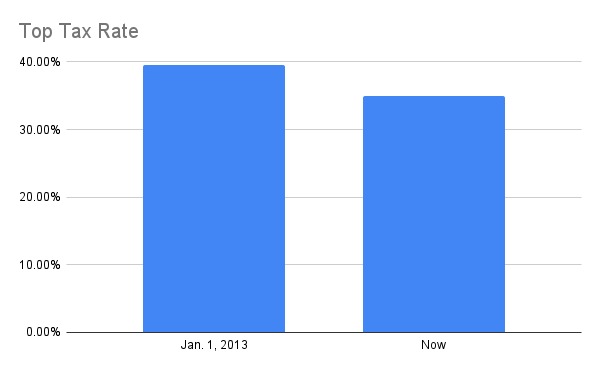
Nah, pada materi ini, kita akan fokus membahas tentang prinsip tersebut. Sebagai permulaan, mari kita lihat berbagai contoh visualisasi data yang buruk dan tidak mengikuti prinsip desain dalam visualisasi data.

Sederhananya, visualisasi data yang buruk merupakan bentuk visual yang tidak mampu menyampaikan pesan dan informasi terkait data secara baik serta efisien. Umumnya visualisasi data yang buruk memuat salah satu dari tiga masalah berikut.

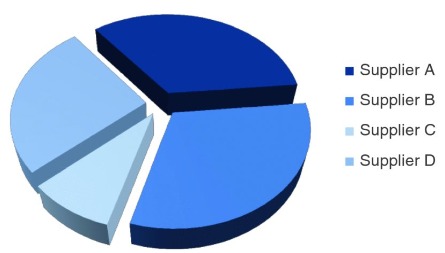
* **Misleading**  
  Masalah pertama dalam visualisasi data yang buruk ialah *misleading information* yang mampu mengakibatkan kesalahan dalam pengambilan kesimpulan dari sebuah data. Berikut contoh visualisasi data yang bisa mengakibatkan misleading information [10].



Apa yang ada di benak Anda ketika melihat visualisasi data tersebut? Tentunya Anda akan melihat bahwa terdapat gap yang cukup besar dari kedua nilai tersebut. Hal ini contoh kesalahan pengambilan kesimpulan karena buruknya sebuah visualisasi data.  
  
Apakah Anda bisa menebak kesalahan apa yang mengakibatkan visualisasi data di atas menghasilkan kesimpulan yang salah? *Yap*, benar sekali visualisasi datanya tidak mulai dari nol sehingga terkesan ada gap yang cukup besar dari kedua nilai tersebut. Berikut merupakan contoh perbaikan dari visualisasi tersebut.

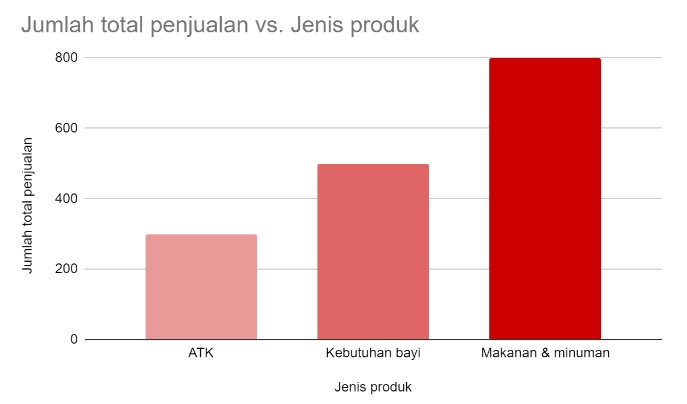


* **Hides**  
  Selain *misleading information*, masalah lain yang umum dijumpai dalam visualisasi data yang buruk ialah menyembunyikan informasi tertentu. Berikut contoh visualisasi data yang menyembunyikan informasi penting dari sebuah data [10].

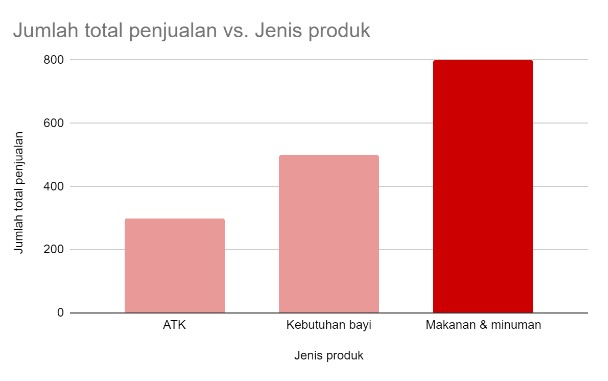


Berdasarkan visualisasi data di atas bisakah Anda menjawab berapa besar market share yang dimiliki supplier A? Lalu, supplier manakah yang memiliki market share terbesar? Anda mungkin akan kesulitan dalam menjawab kedua pertanyaan tersebut karena banyak sekali informasi yang disembunyikan dari visualisasi tersebut.

* **Distracts**  
  Masalah lain yang mungkin terjadi ialah distraksi. Visualisasi data yang buruk sering kali menyertakan komponen visual yang seharusnya tidak dibutuhkan dan malah mengganggu proses pengambilan kesimpulan dari sebuah visualisasi data. Berikut merupakan contoh visualisasi data yang memuat komponen visual yang tidak dibutuhkan.



Pada contoh visualisasi di atas, kita menggunakan warna untuk membedakan kategori pada data tersebut. Hal ini sebenarnya tidak dibutuhkan karena informasi terkait hal ini sudah tersedia pada sumbu x.  
  
Warna merupakan komponen visual yang penting dalam visualisasi data. Oleh karena itu, kita bisa menggunakannya untuk menyampaikan pesan yang menarik dan mempermudah orang lain untuk memahami data yang kita miliki. Gambar di bawah ini merupakan contoh visualisasi data yang baik dalam penggunaan warna untuk menyorot bagian tertentu.



Oke, kita telah melihat berbagai masalah beserta contohnya pada visualisasi data yang buruk. Kini saatnya kita mengenal berbagai prinsip dalam visualisasi data untuk menghindari kesalahan tersebut.

**Prinsip Desain dalam Visualisasi Data**

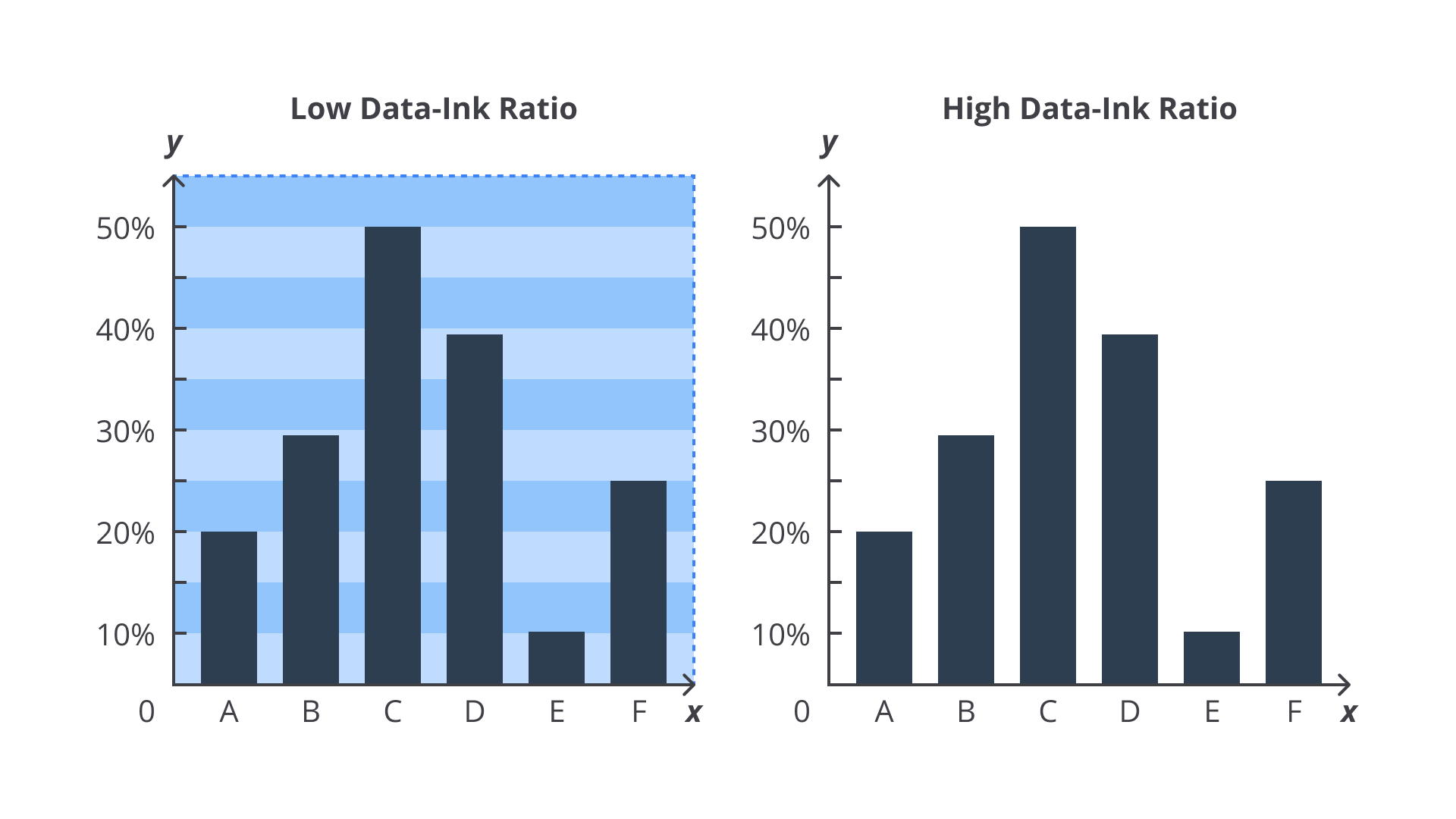
Jika ditelisik secara mendasar, visualisasi data merupakan proses dalam mengubah data ke dalam bentuk visual menggunakan berbagai elemen visual. Berikut merupakan beberapa elemen visual yang umum digunakan untuk membuat visualisasi data.

* **Position**: elemen ini akan membantu kita merepresentasikan titik data menggunakan sumbu tertentu (seperti sumbu X, Y, dan Z) sebagai acuan.
* **Size**: ukuran (panjang atau lebar) merupakan elemen visual yang umumnya kita gunakan untuk membedakan serta membandingkan nilai dari kategori atau titik data tertentu.
* **Shape**: bentuk merupakan salah satu elemen visual yang dapat digunakan untuk membedakan kategori atau titik data tertentu.
* **Color**: selain bentuk, warna juga merupakan pilihan elemen visual lain yang dapat digunakan untuk membedakan kategori atau titik data tertentu. Ketika menggunakan elemen ini, kita harus ingat bahwa tidak semua orang memiliki kemampuan untuk membedakan warna dengan baik.
* **Texture**: penambahan tekstur atau pola tertentu bisa menjadi alternatif lain dalam membedakan kategori atau titik data tertentu.
* **Angle**: pada beberapa pilihan bentuk visualisasi data, sudut merupakan salah satu elemen visual yang digunakan untuk merepresentasikan nilai dari suatu data.

Oke, itulah keenam elemen visual yang biasa digunakan dalam membuat visualisasi data. Tugas kita sebagai praktisi data ialah mencari cara untuk menerapkan seluruh elemen visual tersebut secara efektif untuk menyiratkan kisah dari sebuah data.

Nah, pertanyaan selanjutnya ialah bagaimana cara menerapkan keenam elemen tersebut secara efektif. Kunci utama dalam hal ini adalah menghindari penggunaan elemen visual yang tidak perlu. Untuk memastikan penggunaan elemen visual secara efektif, Edward Tufte telah mengenalkan sebuah konsep bernama **Data Ink Ratio**pada bukunya yang berjudul “*The Visual Display of Quantitative Data*”.

Data ink ratio merupakan sebuah konsep parameter yang digunakan sebagai penentu efektivitas penggunaan visual elemen dalam sebuah visualisasi data. Ia didefinisikan sebagai perbandingan antara tinta (bisa diartikan sebagai elemen visual) yang digunakan untuk mendeskripsikan data dan total tinta yang digunakan dalam satu visualisasi data. Rasio ini dapat digunakan untuk memastikan semua elemen visual yang digunakan relevan untuk kebutuhan penyampaian pesan dari sebuah data [12]. Jadi sederhananya, visualisasi data yang efektif haruslah memiliki nila ink ratio yang tinggi.



**Prinsip Integritas dalam Visualisasi Data**

Ketika membuat sebuah visualisasi data, kita tidak hanya memikirkan terkait faktor estetika (seni) saja. Melainkan, kita juga harus memastikan visualisasi data yang kita buat mampu menyampaikan berbagai informasi penting dari sebuah data tanpa menyembunyikan fakta, mengarahkan orang lain pada asumsi tertentu, serta menghindari *misinterpretasi*. Oleh karena itu, selain menerapkan prinsip desain kita juga menerapkan prinsip integritas dalam proses pembuatan visualisasi data.

Pada bukunya yang berjudul The Visual Display of Quantitative Data, Edward Tufte juga mengenalkan beberapa prinsip guna menjaga integritas dalam membuat visualisasi data. Berikut beberapa prinsip tersebut [12].

* Representasi angka pada sebuah grafik harus sebanding dengan kuantitas sebenarnya. Hal ini untuk menghindari terjadinya misinterpretasi dari sebuah visualisasi data.
* Pelabelan yang jelas, terperinci, dan menyeluruh merupakan kunci utama dalam menghindari ambiguitas dalam sebuah visualisasi data.
* Visualisasi data yang baik harus menunjukkan variasi pada data bukan variasi desain.
* Untuk menghindari asumsi, kita harus menggunakan unit yang umum digunakan (sesuai standar atau kesepakatan) dalam merepresentasikan suatu data tertentu.
* Jumlah variabel (*feature*/kolom) yang digambarkan tidak boleh melebihi jumlah dimensi dalam data. Sebagai contoh, untuk menggambarkan hubungan variabel X dan Y, kita disarankan untuk menggunakan plot dua dimensi ketimbang menggunakan plot tiga dimensi.
* Sebuah grafik tidak boleh menampilkan data yang di luar konteks.

Nah, itu dia keenam prinsip yang harus kita ikuti untuk menjaga integritas dan menghindari misinterpretasi dalam sebuah visualisasi data. Sebenarnya pada buku yang sama, Edward Tufte juga mengenalkan sebuah parameter bernama **Lie Factor** untuk mengukur kesalahan representasi dalam sebuah visualisasi data. Parameter ini didefinisikan sebagai rasio perbandingan ukuran yang ditampilkan pada grafik dan ukuran yang sebenarnya ada dalam data. Rasio ini tentunya akan memberikan kita gambaran sejauh mana visualisasi mendistorsi nilai dari sebuah data [12].

Yey, sekarang lengkaplah pengetahuan dasar kita terkait visualisasi data. Pengetahuan ini akan sangat membantu kita dalam membuat visualisasi data yang benar dan efektif. Pada materi selanjutnya, kita akan belajar berbagai teknik dan cara dalam membuat visualisasi data menggunakan library matplotlib dan seaborn.

**Univariate Visualization**

Sampai dengan tahap ini, Anda telah memahami berbagai konsep dasar dalam membuat visualisasi data yang efektif. Untuk melengkapi pengetahuan Anda, pada materi ini kita akan belajar cara membuat visualisasi data menggunakan library matplotlib dan seaborn.

Seperti yang telah kita bahas sebelumnya, visualisasi data merupakan teknik yang digunakan dalam merepresentasikan atau menyajikan data ke dalam bentuk visual yang lebih mudah dipahami. Nah, berdasarkan jumlah variabel (*feature*/kolom) yang ingin disajikan, visualisasi data dapat dibagi menjadi tiga kategori.

* **Univariate visualization**  
  Univariate visualization merupakan bentuk visualisasi data yang hanya merepresentasikan informasi yang terdapat pada satu variabel. Jenis visualisasi ini umumnya digunakan untuk memberikan kita gambaran terkait distribusi sebuah variabel dalam suatu dataset.
* **Bivariate visualization**  
  Bivariate visualization merupakan bentuk visualisasi data untuk menyajikan informasi yang terdapat dalam dua variabel. Visualisasi jenis ini dapat digunakan untuk menganalisis hubungan antar dua variabel dalam sebuah dataset.
* **Multivariate visualization**  
  *Last but not least*, multivariate visualization merupakan jenis visualisasi data untuk menggambarkan informasi yang terdapat dalam lebih dari dua variabel. Jenis visualisasi ini digunakan untuk merepresentasikan hubungan dan pola yang terdapat dalam multidimensional data.

Oke, itulah ketiga jenis visualisasi data berdasarkan jumlah variabel yang disajikan. Nah, pada materi ini, kita akan fokus membahas tentang univariate visualization.

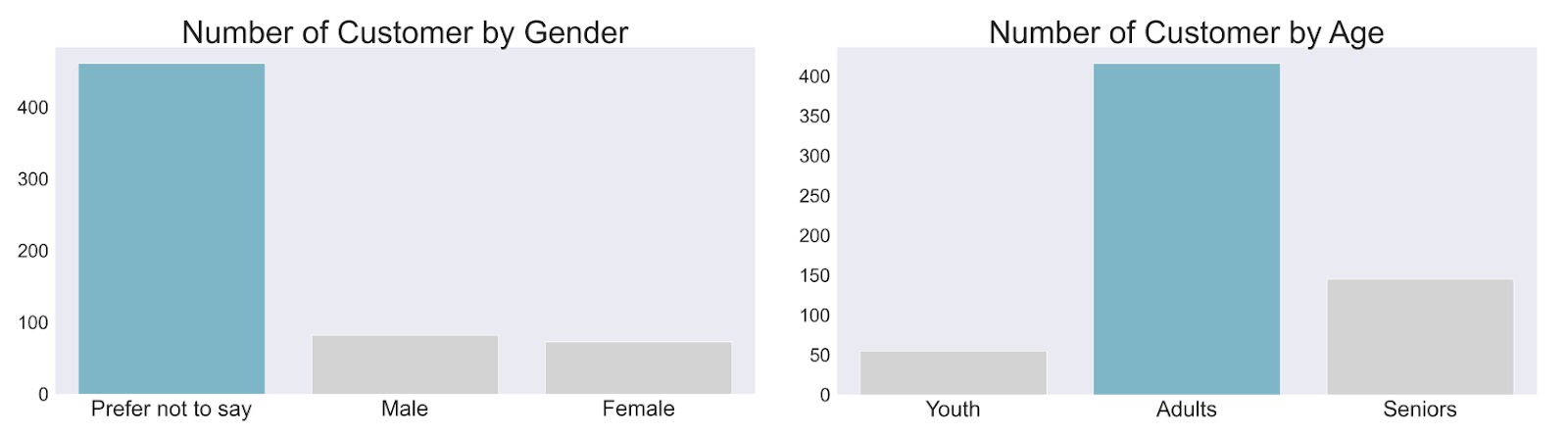
Secara umum, terdapat beberapa pilihan grafik yang dapat digunakan dalam univariate visualization, seperti *bar chart*, *pie chart*, *histogram*, *box plot*, dll. Bagaimana menurut Anda? Apakah Anda penasaran kegunaan dan cara membuat seluruh pilihan grafik tersebut? Jika iya, yuk, simak penjelasan berikut.

**Bar Chart**

Bentuk visualisasi data pertama yang akan kita bahas ialah bar chart atau dikenal juga dengan sebutan diagram batang. Ia merupakan pilihan visualisasi data yang digunakan untuk menggambarkan distribusi dari suatu data kategoris (*categorical data*). Pada bar chart, setiap kategori akan direpresentasikan sebagai sebuah batang. Tinggi dari sebuah batang menunjukkan frekuensi atau jumlah titik data dari kategori tersebut.

Sebagai contoh, katakanlah kita memiliki data survey dan ingin melihat distribusi responden berdasarkan gender. Nah, pada kasus ini, kita bisa menggunakan bar chart untuk merepresentasikan informasi tersebut seperti gambar di bawah ini.

Jika data kategori tersebut merupakan data nominal, kita bisa mengurutkan tampilan bar chart berdasarkan jumlahnya. Namun, ingat hal ini hanya berlaku untuk data nominal, sedangkan untuk data ordinal alangkah baiknya kita urutkan berdasarkan peringkat dari data tersebut. Hal ini dilakukan karena pada data ordinal peringkat merupakan informasi yang sangat penting untuk di informasikan.

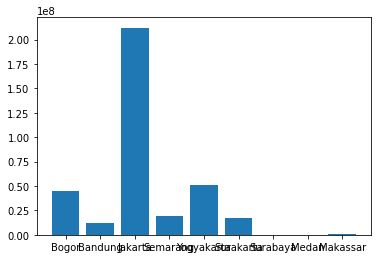


Salah satu variasi bar chart yang sering Anda jumpai ialah horizontal bar chart. Pada variasi ini,  sumbu X menggambarkan kategori dan sumbu Y memuat informasi frekuensi dari tiap kategori tersebut. Variasi ini umumnya digunakan ketika kita memiliki banyak kategori atau nama kategori yang cukup panjang.

Oke, sekarang kita telah memahami konsep dasar tentang bar chart. Pertanyaan selanjutnya adalah bagaimana cara membuatnya? Kabar baik untuk kita semua karena library matplotlib telah menyediakan sebuah function bernama pyplot.bar() untuk membuat sebuah bar chart. Untuk menggunakan function tersebut, kita perlu mendefinisikan parameter x dan height dari bar chart yang ingin dibuat. Berikut merupakan contoh untuk membuatnya.

|  |
| --- |
| import matplotlib.pyplot as plt    cities = ('Bogor', 'Bandung', 'Jakarta', 'Semarang', 'Yogyakarta',  'Surakarta','Surabaya', 'Medan', 'Makassar')    populations = (45076704, 11626410, 212162757, 19109629, 50819826, 17579085,  3481, 287750, 785409)    plt.bar(x=cities, height=populations)  plt.show() |

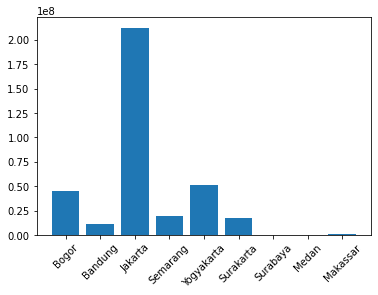
Kode di atas akan menghasilkan bentuk visualisasi seperti berikut.



*Hmm*.., apakah Anda bisa membaca nama kota yang ada pada sumbu X? Tentunya tidak, bukan? Oleh karena itu, kita harus melakukan sedikit modifikasi pada plot tersebut. Untuk melakukan modifikasi ini, kita bisa menggunakan function xticks() dan mengatur besar parameter rotation. Parameter tersebut akan mengatur besar rotasi nilai kategori pada sumbu X. Contoh penggunaanya bisa Anda lihat pada contoh kode di bawah ini.

|  |
| --- |
| plt.bar(x=cities, height=populations)  plt.xticks(rotation=45)  plt.show() |

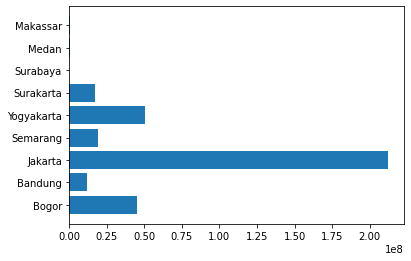
Kode di atas akan menghasilkan tampilan visualisasi yang sedikit lebih baik seperti berikut.



Hasil visualisasi di atas tentunya jauh lebih baik jika dibandingkan dengan hasil bar chart sebelumnya. Sebenarnya kita bisa merepresentasikan data ini secara lebih baik menggunakan function barh(). Untuk menggunakan function tersebut kita harus mendefinisikan parameter y dan width dari bar chart yang ingin dibuat. Berikut merupakan contoh kode untuk melakukannya.

|  |
| --- |
| plt.barh(y=cities, width=populations)  plt.show() |

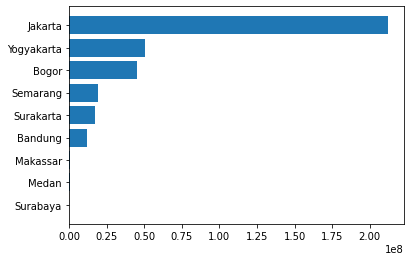
Kode di atas akan menghasilkan bentuk visualisasi data yang lebih mudah diamati seperti berikut.



Nah, jika diperhatikan data tersebut merupakan data nominal sehingga kita boleh mengurutkan nilainya berdasarkan jumlah populasi terbanyak. Untuk melakukan hal ini, kita membutuhkan bantuan library pandas dalam mengubah data menjadi DataFrame dan mengurutkannya menggunakan function sort\_values(). Berikut contoh kode yang dapat Anda gunakan.

|  |
| --- |
| import pandas as pd    df = pd.DataFrame({  'Cities': cities,  'Population': populations,  })    df.sort\_values(by='Population', inplace=True)    plt.barh(y=df["Cities"], width=df["Population"])  plt.show() |

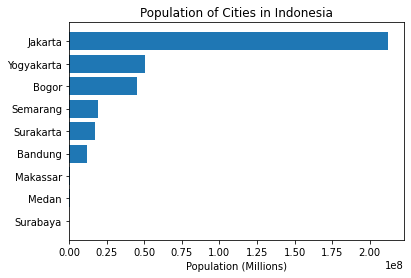
Berikut merupakan hasil visualisasi data dari kode di atas.



Bagaimana menurut Anda? Hasilnya sudah cukup baik, bukan? Untuk memperjelas visualisasi data yang dibuat, kita bisa menambahkan beberapa keterangan seperti judul dan keterangan sumbu. Untuk melakukan hal ini, kita bisa menggunakan function title(), xlabel(), dan ylabel(). Kode di bawah ini merupakan contoh kode untuk menambah keterangan pada hasil plot visualisasi data.

|  |
| --- |
| plt.barh(y=df["Cities"], width=df["Population"])  plt.xlabel("Population (Millions)")  plt.title("Population of Cities in Indonesia")  plt.show() |

Kode di atas akan menghasilkan visualisasi data seperti berikut.



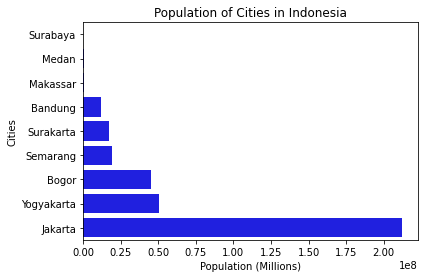
Selain menggunakan library matplotlib, kita juga bisa membuat bar chart menggunakan library seaborn. Untuk membuat bar chart menggunakan library seaborn, kita bisa menggunakan function barplot(). Function ini akan menyediakan beberapa parameter penting seperti berikut.

* data: menampung DataFrame yang akan digunakan.
* x, y: menampung nama kolom atau data yang divisualisasikan.
* orient: orientasi dari bar chart yang akan digunakan (“v” atau “h”).
* color: mendefinisikan warna yang akan digunakan.

Berikut contoh kode untuk membuat bar chart dengan library seaborn.

|  |
| --- |
| import seaborn as sns    sns.barplot(y=df["Cities"], x=df["Population"], orient="h", color='blue')  plt.xlabel("Population (Millions)")  plt.title("Population of Cities in Indonesia")  plt.show() |

Kode di atas akan menghasilkan visualisasi data seperti gambar berikut.

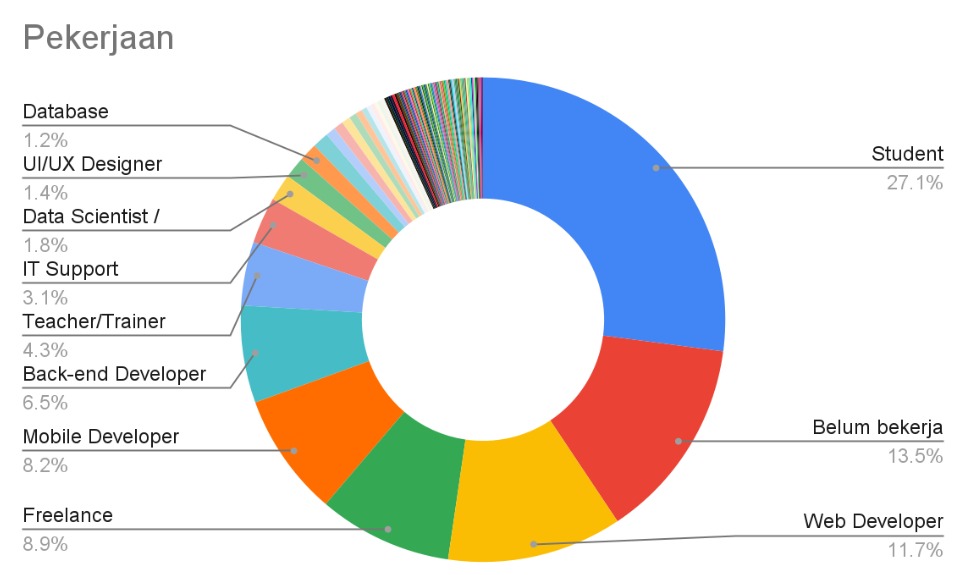


Bagaimana menurut Anda? Cukup mudah bukan untuk membuat bar chart menggunakan matplotlib dan seaborn? Nah, selanjutnya kita akan belajar cara membuat visualisasi data untuk bentuk grafik yang lain.

**Pie Chart**

Bentuk visualisasi data berikutnya yang akan kita bahas ialah pie chart. Ia merupakan bentuk visualisasi data yang dapat digunakan untuk menggambarkan perbandingan frekuensi tiap kategori pada suatu data kategoris. Pada pie chart, setiap kategori digambarkan oleh irisan dalam sebuah lingkaran dan frekuensi digambarkan sebagai luasan (*area*) dari irisan tersebut. Jadi pada pie chart semakin besar area dari sebuah irisan menandakan bahwa kategori tersebut memiliki jumlah titik data yang banyak. Sebaliknya, semakin sempit luasan pada sebuah irisan menandakan bahwa kategori tersebut memiliki frekuensi jumlah titik data yang sedikit.

Apabila Anda perhatikan informasi yang direpresentasikan dengan pie chart ini cukup mirip dengan bar chart. Oleh karena itu, pie chart sering digunakan sebagai alternatif dari bar chart. Namun, perlu Anda ingat bahwa sebaiknya pie chart digunakan untuk merepresentasikan data yang memiliki jumlah kategori yang relatif sedikit. Hal ini dilakukan untuk mempermudah orang lain dalam memahami informasi yang ingin kita sampaikan. Bisa Anda lihat pada gambar di bawah ini, betapa sulitnya membaca nilai dari kategori lain.



Terdapat variasi lain dari pie chart, yaitu donut plot. Sebenarnya, ia merupakan pie chart yang memiliki lubang pada bagian tengahnya sehingga membentuk donut. Selain itu, tidak ada perbedaan mendasar dalam penggunaan antara pie chart dan donut plot. Oleh karena itu, estetika merupakan alasan yang umum digunakan ketika memilih salah satu dari keduanya.

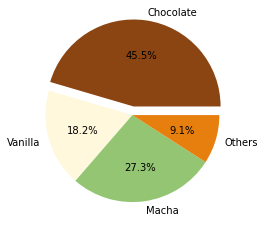
Seperti biasa setelah memahami konsepnya, kini saatnya kita melihat bagaimana cara membuatnya. Untuk membuat pie chart, kita bisa memanfaatkan library andalan, yaitu matplotlib. Ia telah menyediakan sebuah function bernama pie() untuk menghasilkan grafik pie chart. Function tersebut memiliki beberapa parameter penting yaitu seperti berikut.

* x: menampung data yang akan divisualisasikan.
* explode: menampung array atau list yang mengatur posisi tiap irisan lingkaran.
* labels: sekumpulan string yang digunakan untuk memberi label pada tiap irisan lingkaran.
* colors: sekumpulan warna yang akan digunakan pada tiap irisan lingkaran.
* autopct: string yang digunakan untuk memberi numerik label pada tiap irisan lingkaran.

Berikut contoh kode untuk membuat grafik pie chart.

|  |
| --- |
| import matplotlib.pyplot as plt    flavors = ('Chocolate', 'Vanilla', 'Macha', 'Others')  votes = (50, 20, 30, 10)  colors = ('#8B4513', '#FFF8DC', '#93C572', '#E67F0D')  explode = (0.1, 0, 0, 0)    plt.pie(  x=votes,  labels=flavors,  autopct='%1.1f%%',  colors=colors,  explode=explode  )  plt.show() |

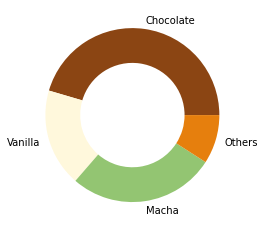
Kode di atas akan menghasilkan tampilan visualisasi data seperti berikut.



Selain itu, kita juga bisa menggunakan parameter wedgepros untuk menghasilkan donut plot. Parameter ini digunakan untuk mengatur ukuran dari lubang yang dihasilkan. Berikut contoh kodenya.

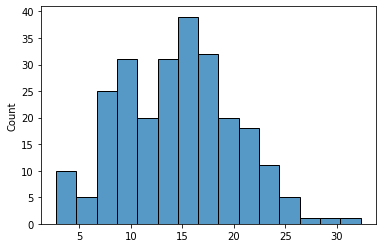
|  |
| --- |
| plt.pie(  x=votes,  labels=flavors,  colors=colors,  wedgeprops = {'width': 0.4}  )  plt.show() |

Kode di atas, akan menghasilkan tampilan visual seperti berikut.

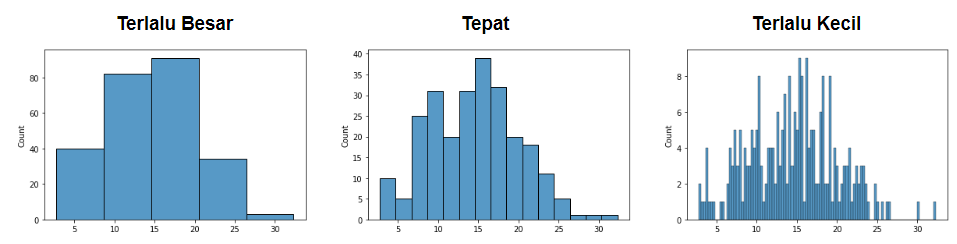


**Histogram**

Jika kedua grafik sebelumnya digunakan untuk data kategorik, sekarang kita kenalan dengan grafik untuk data kuantitatif yang bernama histogram. Grafik ini digunakan untuk menggambarkan distribusi dari data kuantitatif. Jika dilihat sekilas, bentuk grafik histogram mirip dengan bentuk bar chart perbedaannya hanya terdapat pada sumbu X. Pada grafik histogram, sumbu X digunakan untuk menampung range nilai dari data kuantitatif yang dikenal dengan istilah *bins*.



Pada grafik histogram, ukuran bins akan sangat berpengaruh terhadap cara kita dalam merepresentasikan data kuantitatif. Jika menggunakan ukuran bins yang terlalu besar, kita akan kehilangan cukup banyak informasi dari data kuantitatif yang ingin divisualisasikan. Sebaliknya, apabila ukuran bins yang digunakan terlalu kecil, hal ini akan menghasilkan banyak noise yang mendistraksi distribusi data yang ingin kita amati.



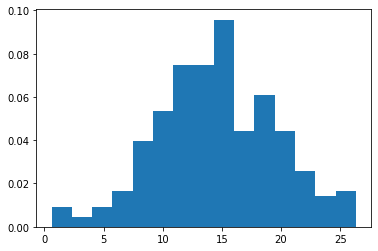
Untuk membuat grafik histogram, kita bisa menggunakan function hist() yang disediakan oleh library matplotlib. Function ini menerima beberapa parameter seperti berikut.

* x: menampung data yang akan divisualisasikan.
* bins: menampung jumlah bins (sebanding dengan ukurannya) yang akan digunakan untuk membuat grafik histogram.

Berikut merupakan contoh kode untuk membuat grafik histogram.

|  |
| --- |
| import matplotlib.pyplot as plt  import numpy as np    x = np.random.normal(15, 5, 250)    plt.hist(x=x, bins=15)  plt.show() |

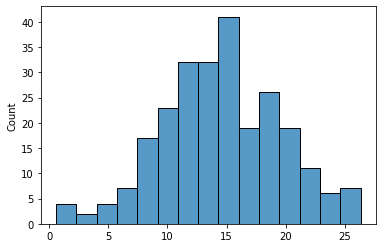
Pada contoh kode di atas, kita menggunakan function random.normal() yang disediakan oleh library numpy untuk menghasilkan sampel data acak yang memiliki distribusi normal. Sample data random tersebut memiliki jumlah **250 titik data** dengan nilai mean sebesar **15** dan standard deviation sebesar **5**. Kode di atas akan menghasilkan grafik seperti berikut.



Selain menggunakan library matplotlib, kita juga bisa membuat grafik histogram dengan library seaborn. Library ini menyediakan function histplot() untuk membuat grafik histogram. Berikut contoh kode untuk melakukannya.

|  |
| --- |
| import seaborn as sns    sns.histplot(x=x, bins=15)  plt.show() |

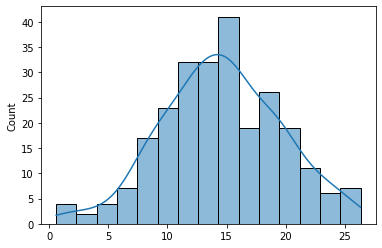
Kode di atas akan menghasilkan grafik histogram seperti berikut.



Untuk mempermudah dalam mengidentifikasi distribusi dari sebuah data kuantitatif, kita bisa menambahkan parameter kde=True seperti pada contoh kode di bawah ini.

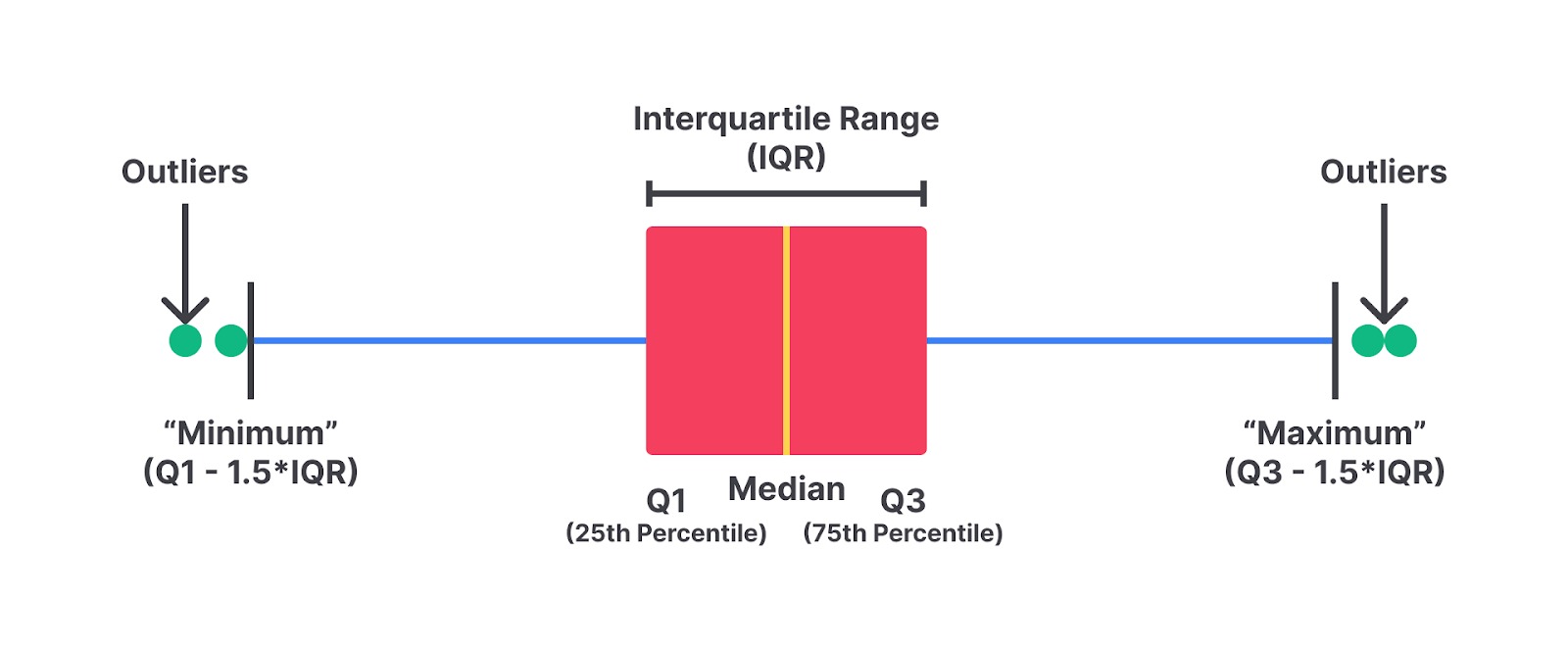
|  |
| --- |
| sns.histplot(x=x, bins=15, kde=True)  plt.show() |

Kode tersebut akan menghasilkan grafik histogram yang disertai sebuah garis. Garis tersebut menunjukkan densitas distribusi berdasarkan grafik histogram yang dihasilkan. Berikut merupakan visualisasi yang dihasilkan.



**Box Plot**

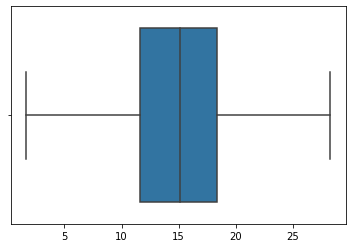
Bentuk visualisasi data lain yang bisa digunakan untuk mengidentifikasi distribusi pada data kuantitatif ialah box plot. Ia merupakan bentuk visual untuk merepresentasikan nilai IQR beserta ambang batas bawah dan atas dari sebuah data. Box plot ini juga bisa digunakan untuk mengidentifikasi outlier atau pencilan yang ada pada data kuantitatif.



Kita bisa menggunakan function boxplot() yang disediakan oleh library seaborn. Berikut contoh kode untuk menggunakannya.

|  |
| --- |
| import matplotlib.pyplot as plt  import seaborn as sns  import numpy as np    x = np.random.normal(15, 5, 250)  sns.boxplot(x)  plt.show() |

Kode di atas akan menghasilkan visualisasi data seperti berikut.



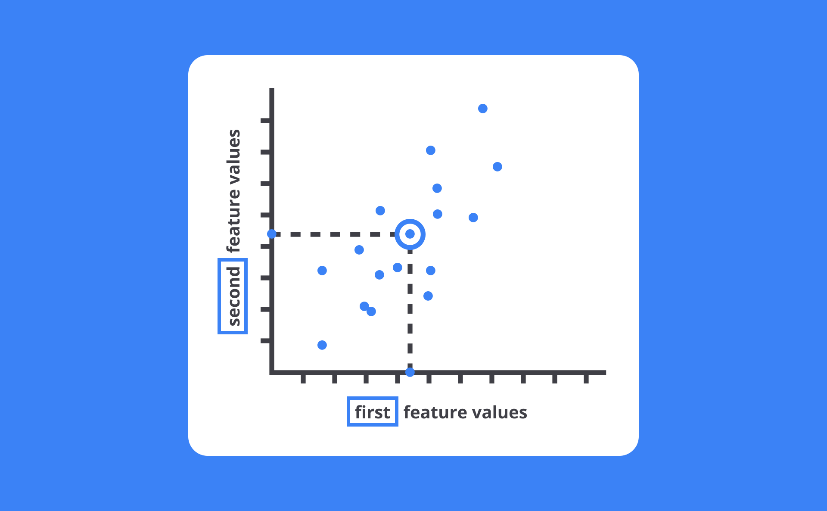
**Bivariate dan Multivariate Visualization**

Sederhananya, bivariate visualization merupakan bentuk visualisasi data untuk menyajikan informasi dari dua variabel dalam suatu dataset. Di lain sisi, multivariate visualization merupakan jenis visualisasi data untuk menggambarkan informasi yang terdapat dalam lebih dari dua variabel. Keduanya sering digunakan untuk mendeskripsikan hubungan antar variabel dalam suatu dataset.

Terdapat beberapa pilihan visualisasi data yang umum digunakan dalam bivariate dan multivariate visualization, seperti *scatter plot*, *line chart*, dan *clustered bar chart*. Selain itu, sebenarnya kita juga bisa menggunakan komponen visual tambahan, yaitu *color*, *shape,* dan *texture*untuk mendeskripsikan lebih banyak variabel. Nah, pada materi ini, kita akan belajar seluruh hal tersebut satu per satu.

**Scatter Plot**

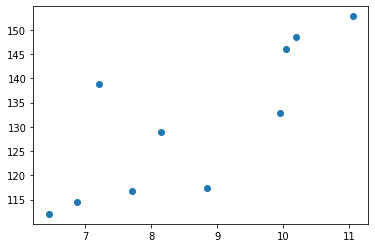
Bentuk visualisasi data pertama yang akan kita bahas ialah scatter plot. Ia merupakan bentuk visualisasi data yang digunakan untuk melihat hubungan antara dua atau lebih variabel data kuantitatif. Pada scatter plot sumbu X, Y, atau Z digunakan untuk menampung nilai dari setiap variabel yang akan divisualkan. Titik pada scatter plot merupakan titik temu data dari setiap sumbu yang digunakan. Gambar di bawah ini merupakan contoh tampilan scatter plot untuk dua variabel data.



Untuk membuat scatter plot, kita bisa menggunakan library matplotlib. Ia telah menyediakan sebuah function bernama scatter() untuk membantu kita membuat scatter plot secara lebih mudah. Berikut contoh kode untuk menggunakannya.

|  |
| --- |
| import matplotlib.pyplot as plt    lemon\_diameter = [6.44, 6.87, 7.7, 8.85, 8.15,  9.96, 7.21, 10.04, 10.2, 11.06]  lemon\_weight = [112.05, 114.58, 116.71, 117.4, 128.93,  132.93, 138.92, 145.98, 148.44, 152.81]    plt.scatter(x=lemon\_diameter, y=lemon\_weight)  plt.show() |

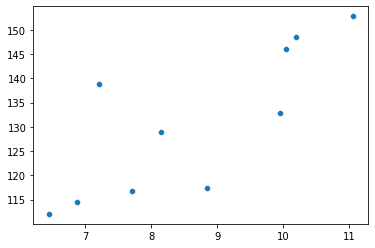
Kode di atas akan menghasilkan visualisasi data seperti berikut.



Bagaimana cukup mudah, bukan? Nah, selain menggunakan library matplotlib, kita juga bisa menggunakan function scatterplot() yang disediakan oleh library seaborn untuk membuat scatter plot. Berikut contoh kode untuk menggunakan function tersebut.

|  |
| --- |
| import matplotlib.pyplot as plt  import seaborn as sns    sns.scatterplot(x=lemon\_diameter, y=lemon\_weight)  plt.show() |

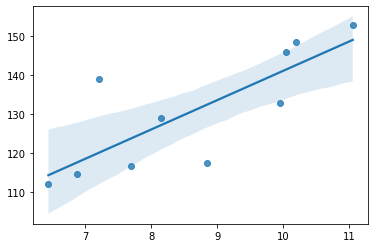
Berikut tampilan visualisasi data yang dihasilkan.



Nah, untuk mempermudah dalam melihat korelasi atau hubungan antar variabel, kita bisa menggunakan function regplot() yang disediakan oleh library seaborn. Function tersebut memadukan scatter plot dan*regression function* (metode statistik untuk memperkirakan korelasi antar *independent* dan *dependent variable*) untuk melihat tren serta korelasi antar variabel.

|  |
| --- |
| sns.regplot(x=lemon\_diameter, y=lemon\_weight)  plt.show() |

Kode di atas akan menghasilkan tampilan visualisasi data seperti berikut.



Pada visualisasi data di atas, garis biru menunjukkan garis korelasi dari hasil regression sedangkan pita di sekitar garis tersebut menunjukkan *confidence level* dari hasil regression tersebut. Semakin kecil bentuk pita yang dihasilkan menandakan tingkat confidence level yang tinggi.

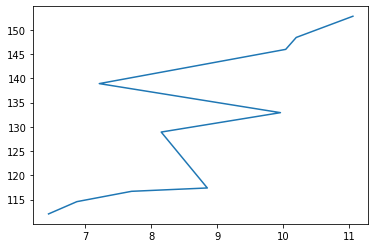
Selain itu, kita juga bisa membuat scatter plot untuk menggambarkan hubungan antar tiga variabel. Namun, untuk kebanyakan kasus kita harus menghindari penggunaan plot tiga dimensi karena sulit diamati dan dipahami.

**Line Chart**

Bentuk visualisasi data yang selanjutnya akan kita bahas ialah line chart. Ia merupakan bentuk grafik yang umum digunakan untuk menggambarkan tren nilai dari suatu variabel terhadap variabel lain. Kita bisa membuat line chart menggunakan function plot() yang disediakan oleh matplotlib. Berikut merupakan contoh kode untuk melakukannya.

|  |
| --- |
| import matplotlib.pyplot as plt    lemon\_diameter = [6.44, 6.87, 7.7, 8.85, 8.15,  9.96, 7.21, 10.04, 10.2, 11.06]  lemon\_weight = [112.05, 114.58, 116.71, 117.4, 128.93,  132.93, 138.92, 145.98, 148.44, 152.81]    plt.plot(lemon\_diameter, lemon\_weight)  plt.show() |

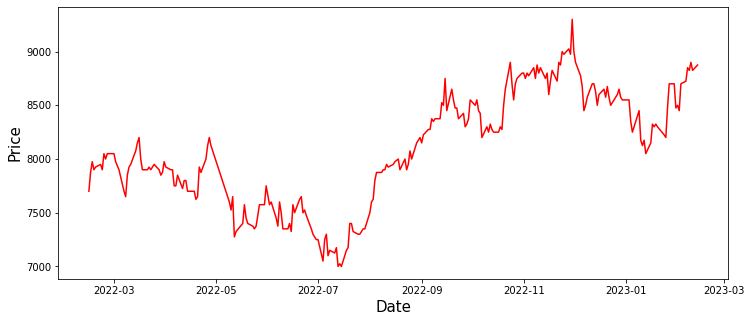
Kode di atas akan menghasilkan visualisasi data seperti berikut.



Bentuk visualisasi data ini umumnya digunakan untuk melihat trend dari data berbentuk *time series* (data yang direkam dalam interval waktu yang konsisten). Berikut contoh kode untuk menampilkan plot harga saham Bank Central Asia dengan kode saham BBCA.

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt    url = 'https://query1.finance.yahoo.com/v7/finance/download/BBCA.JK?period1=1644796800&period2=1676332800&interval=1d&events=history&includeAdjustedClose=true'  df = pd.read\_csv(url)  df['Date'] = pd.to\_datetime(df['Date'])    plt.figure(figsize=(12, 5))  plt.plot(df['Date'], df['Close'], color='red')  plt.xlabel('Date',size=15)  plt.ylabel('Price',size=15)  plt.show() |

Pada contoh kode di atas, kita memvisualkan data saham yang diperoleh dari [Yahoo Finance](https://finance.yahoo.com/" \t "_blank). Agar tampilan visualnya enak dilihat, pada contoh ini, kita mengatur ukuran plot-nya dengan argumen figsize pada function figure(). Berikut merupakan visualisasi data yang dihasilkan.



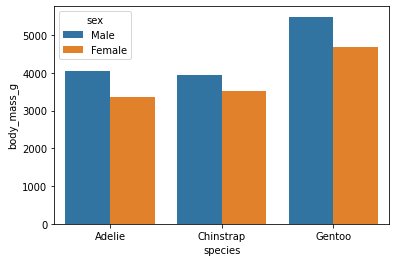
**Clustered Bar Chart**

Clustered bar chart merupakan bentuk modifikasi dari bar chart yang sebelumnya kita kenal dengan menambahkan variabel kategoris lain. Modifikasi ini memungkinkan untuk melihat distribusi serta hubungan antar dua atau lebih variabel kategoris.

Untuk membuat clustered bar chart, kita bisa menggunakan parameter hue pada function barplot() yang disediakan oleh library seaborn. Berikut contoh kode untuk membuatnya.

|  |
| --- |
| import seaborn as sns  import matplotlib.pyplot as plt    penguins = sns.load\_dataset("penguins")    sns.barplot(data=penguins, x="species", y="body\_mass\_g", hue="sex", errorbar=None)  plt.show() |

Pada contoh kode di atas, kita membuat bar chart untuk melihat distribusi berat badan dari setiap spesies penguin. Selanjutnya distribusi tersebut dikelompokkan berdasarkan jenis kelamin. Berikut tampilan visualisasi data yang diperoleh.



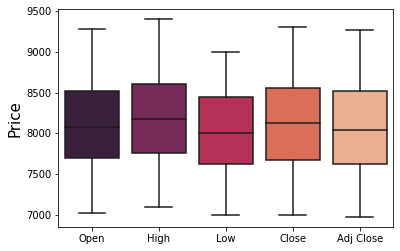
**Menggunakan Komponen Visual Tambahan**

Selain menggunakan scatter plot, line chart, dan clustered bar chart, kita juga bisa memodifikasi bentuk visualisasi data dengan menambahkan komponen visual tambahan seperti *color*, *shape,* serta *texture*untuk mendeskripsikan lebih banyak variabel.

Sebagai contoh, Anda bisa membuat lebih dari satu boxplot untuk menggambarkan distribusi nilai numerik dari beberapa variabel. Hal ini bisa Anda gunakan dengan memasukkan DataFrame yang akan divisualisasikan pada parameter data dalam function boxplot(). Selain itu, Anda juga bisa menggunakan parameter palette untuk mengatur palet warna yang akan digunakan. Berikut contoh kode untuk melakukannya.

|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt  import seaborn as sns    url = 'https://query1.finance.yahoo.com/v7/finance/download/BBCA.JK?period1=1644796800&period2=1676332800&interval=1d&events=history&includeAdjustedClose=true'  df = pd.read\_csv(url)  df['Date'] = pd.to\_datetime(df['Date'])    df\_boxplot = df[["Open", "High", "Low", "Close", "Adj Close"]]    sns.boxplot(data=df\_boxplot, palette="rocket")  plt.ylabel('Price',size=15)  plt.show() |

Sederhananya, pada contoh kode di atas, kita memvisualkan seluruh data yang ada di dalam df\_boxplot. Selain itu, pada kode tersebut, kita menggunakan palet warna bernama “rocket” (dokumentasi palet warna: [Choosing color palettes](https://seaborn.pydata.org/tutorial/color_palettes.html)). Berikut merupakan tampilan visualisasi data yang dihasilkan.

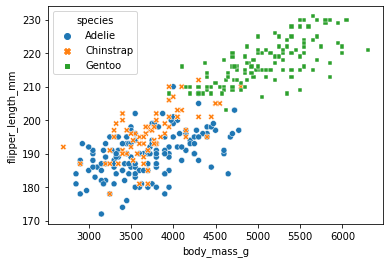


Nah, dengan visualisasi data tersebut, kita dapat membandingkan berbagai variabel yang digunakan untuk mendeskripsikan harga saham. Namun, perlu diingat bahwa hal ini hanya boleh dilakukan untuk membandingkan nilai variabel yang merepresentasikan unit atau satuan ukur yang sama (contoh di atas sama-sama menggambarkan harga saham).

Selain itu, kita juga bisa memodifikasi scatter plot menggunakan shape atau color yang berbeda guna menampilkan hubungan lebih dari dua variabel data. Seperti pada contoh berikut.

|  |
| --- |
| import seaborn as sns    penguins = sns.load\_dataset("penguins")    sns.scatterplot(data=penguins, x="body\_mass\_g", y="flipper\_length\_mm", hue="species", style="species")  plt.show() |

Pada contoh kode di atas, kita menampilkan hubungan antara berat badan dan flipper length dari setiap spesies penguin yang dibedakan menggunakan warna (hue) serta bentuk (style). Berikut hasil visualisasi data dari kode tersebut.



Nah, itulah berbagai pilihan bentuk visualisasi dan teknik yang dapat Anda gunakan untuk merepresentasikan dua atau lebih variabel dalam suatu dataset. Nah, pada materi berikutnya, kita akan melihat berbagai hal dan *best practice* dalam melakukan *explanatory analysis*.

**Explanatory Analysis**

Tentunya Anda masih ingat perbedaan antara explanatory analysis dan exploratory analysis, bukan? Sederhananya, exploratory analysis merupakan proses analisis data yang bertujuan untuk mengeksplorasi dan mengenal sebuah data. Di sisi lain, explanatory analysis merupakan proses analisis data yang bertujuan untuk membagikan beberapa *insight* yang menarik dari sebuah data [10]. Insight yang diperoleh dari proses explanatory analysis akan digunakan untuk menjawab permasalahan bisnis yang dihadapi. Insight tersebut juga akan dibagikan kepada orang lain (*stakeholder*) sehingga perlu disertakan dengan visualisasi data yang baik.

Nah, pada materi ini, kita akan membahas berbagai hal penting terkait explanatory analysis. Sebagai permulaan kita akan melihat bagaimana caranya memberikan informasi konteks dalam sebuah visualisasi data.

**Memberikan Informasi Konteks dalam Visualisasi Data**

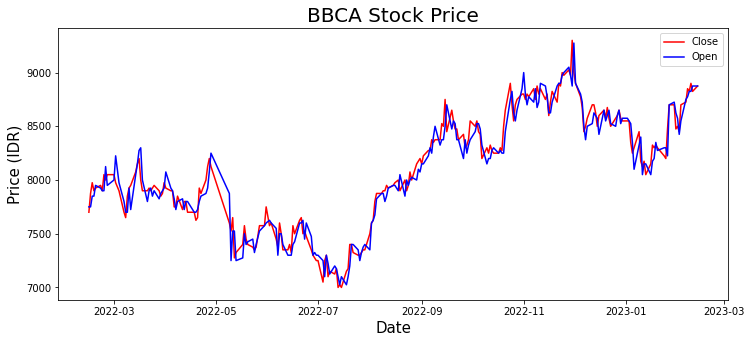
Pada proses explanatory analysis, visualisasi data digunakan sebagai media untuk menyampaikan pesan atau *story* dari sebuah data. Oleh karena itu, ia harus mampu menarik perhatian pembaca (atau audiens) dan menyampaikan pesan secara efektif. Untuk melakukan hal ini, kita harus mampu memilih bentuk grafik/diagram yang tepat dan menerapkan berbagai prinsip dalam visualisasi data. Selain itu, kita juga menyampaikan informasi terkait konteks dari sebuah visualisasi data. Hal ini bisa dicapai dengan menambahkan beberapa komponen berikut.

* **Title**  
  Title atau judul merupakan salah satu komponen penting guna menyampaikan konteks pada sebuah visualisasi data. Kita bisa menambahkan judul pada visualisasi data menggunakan function title() yang disediakan oleh matplotlib.
* **Label axes**  
  Selain judul visualisasi, kita juga perlu menambahkan label atau keterangan pada setiap sumbu yang terdapat dalam visualisasi data. Hal ini dilakukan untuk memberikan konteks terkait titik data yang direpresentasikan. Kita bisa membuat label axes menggunakan function xlabel() dan ylabel() yang disediakan oleh library matplotlib.
* **Legend**  
  Pastikan Anda menambahkan legend memberikan keterangan untuk variabel yang tidak digambarkan pada sumbu. Keterangan yang digunakan akan bergantung pada cara kita dalam merepresentasikan variabel tambahan tersebut. Sebagai contoh, kita bisa menggunakan legend() untuk memberikan keterangan pada variabel kategoris tambahan. Namun, apabila variabel tambahan tersebut berupa nilai numerik yang direpresentasikan menggunakan range warna, kita bisa memanfaatkan function colorbar() untuk memberikan keterangan pada variabel tersebut.

Berikut contoh kode untuk memberikan informasi terkait konteks dari sebuah visualisasi data.

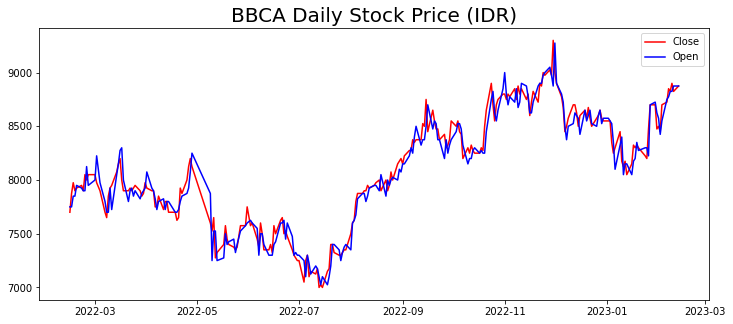
|  |
| --- |
| import pandas as pd  import matplotlib.pyplot as plt    url = 'https://query1.finance.yahoo.com/v7/finance/download/BBCA.JK?period1=1644796800&period2=1676332800&interval=1d&events=history&includeAdjustedClose=true'  df = pd.read\_csv(url)  df['Date'] = pd.to\_datetime(df['Date'])    plt.figure(figsize=(12, 5))  plt.plot(df['Date'], df['Close'], label='Close', color='red')  plt.plot(df['Date'], df['Open'], label='Open', color='blue')  plt.title('BBCA Stock Price', size=20)  plt.xlabel('Date',size=15)  plt.ylabel('Price (IDR)',size=15)  plt.legend()  plt.show() |

Kode di atas akan menghasilkan tampilan visual seperti berikut.



Seperti yang Anda lihat, visualisasi data di atas telah menyertakan informasi yang cukup detail terkait data yang divisualkan. Hal ini akan sangat membantu pembaca atau audiens dalam memahami konteks dari visualisasi data tersebut.

Untuk alasan estetika, biasanya kita akan mengurangi penggunaan keterangan yang sebenarnya tidak dibutuhkan. Sebagai contoh, kita bisa menghilangkan label pada sumbu X dan Y tanpa menghilangkan konteks dari visualisasi data di atas. Hal ini dilakukan dengan memberikan informasi yang eksplisit pada bagian judul dari visualisasi data tersebut seperti pada gambar di bawah ini.



Bagaimana menurut Anda? Visualisasi di atas terlihat lebih *clean* dan *minimalis,*bukan?

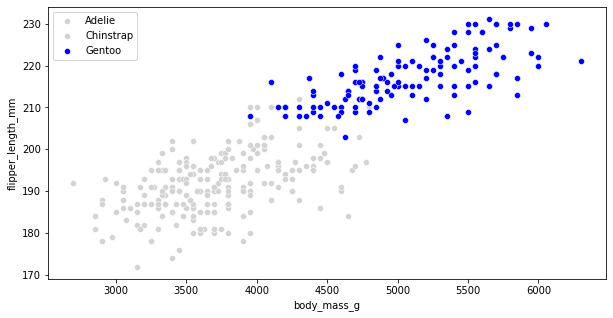
**Fokuskan Perhatian Audiens**

Selain informasi konteks, pastikan audiens fokus pada informasi yang ingin kita sampaikan. Oleh karena itu, sebagai seorang praktisi data yang andal, kita harus mampu mengarahkan perhatian audiens. Untuk melakukan hal ini, kita bisa menggunakan komponen visual tertentu seperti *color*, *shape*, dan *texture*.

Sebagai contoh, kita bisa menggunakan warna yang kontras untuk mengarahkan perhatian audiens pada bagian yang ingin di-*highlight*. Berikut merupakan contoh kode untuk meng-*highlight* bagian penting dalam sebuah visualisasi data.

|  |
| --- |
| import matplotlib.pyplot as plt  import seaborn as sns    penguins\_df = sns.load\_dataset("penguins")    adelie\_df = penguins\_df[penguins\_df.species == "Adelie"]  chinstrap\_df = penguins\_df[penguins\_df.species == "Chinstrap"]  gentoo\_df = penguins\_df[penguins\_df.species == "Gentoo"]    plt.figure(figsize=(10, 5))  sns.scatterplot(data=adelie\_df, x="body\_mass\_g", y="flipper\_length\_mm", facecolor="lightgrey", label="Adelie")  sns.scatterplot(data=chinstrap\_df, x="body\_mass\_g", y="flipper\_length\_mm", facecolor="lightgrey", label="Chinstrap")  sns.scatterplot(data=gentoo\_df, x="body\_mass\_g", y="flipper\_length\_mm", facecolor="blue", label="Gentoo")  plt.legend()  plt.show() |

Pada kode di atas kita ingin memfokuskan audiens pada data salah satu spesies penguin yaitu gentoo yang memiliki ukuran paling besar di antara ketiga spesies tersebut. Berikut merupakan tampilan visualisasi data yang dihasilkan.



Ketika melihat visualisasi data di atas, Anda pasti akan langsung fokus pada titik data dari spesies Gentoo karena ia memiliki warna yang kontras. Ini merupakan salah satu teknik yang umum digunakan untuk mengarahkan fokus audiens.

Namun, perlu diingat bahwa ketika menggunakan teknik ini kita harus menghindari penggunaan warna yang menyakitkan mata audiens atau warna yang memiliki arti berlawanan dengan pesan yang ingin disampaikan. Selain itu, kita juga memperhatikan konsistensi dalam pemilihan warna dalam seluruh visualisasi data yang kita buat.

**Meminta Feedback**

Pada proses pembuatan explanatory analysis, terkadang sulit untuk menilai seberapa efektif visualisasi data yang kita buat. Oleh karena itu, kita membutuhkan feedback serta sudut pandang orang lain guna memastikan bahwa visualisasi data yang kita buat sudah cukup efektif dalam menyampaikan story dan informasi penting dari sebuah data.

Pada proses ini kita bisa meminta teman atau rekan setim untuk melihat seluruh visualisasi data yang telah kita buat. Selanjutnya, mintalah mereka untuk membuat kesimpulan dari seluruh visualisasi data yang telah dibuat. Lalu, pastikan kesimpulan mereka sesuai dengan yang ingin Anda sampaikan. Pada prosesnya, Anda bisa mengajukan beberapa pertanyaan berikut.

* Informasi apa yang Anda peroleh dari setiap visualisasi data tersebut?
* Menurut Anda apa kesimpulan utama dari seluruh visualisasi data tersebut?
* Apakah Anda memiliki pertanyaan terkait data tersebut yang belum terjawab?
* Apakah ada bagian yang belum Anda pahami dari seluruh visualisasi data tersebut?
* Berapa lama waktu yang Anda butuhkan untuk memahami setiap visualisasi data tersebut?

Anda dapat menggunakan beberapa pertanyaan tersebut guna mendapatkan feedback atau masukan untuk meningkatkan kualitas visualisasi data yang dibuat. Selain itu, pertanyaan tersebut juga membantu Anda mengidentifikasi informasi penting lain yang mungkin Anda lewatkan.

Inilah akhir pembahasan tentang explanatory analysis. Semoga materi ini dapat membantu Anda dalam membuat visualisasi data yang mampu secara efektif menyampaikan seluruh informasi penting dari sebuah data.

Nah, pada materi selanjutnya, kita akan mempraktikkan seluruh pengetahuan yang telah dipelajari dalam sebuah studi kasus sederhana. *So*, yuk, kita lanjut ke materi berikutnya!