Visualização de Dados – Parte 03

Prof. Wellington Franco

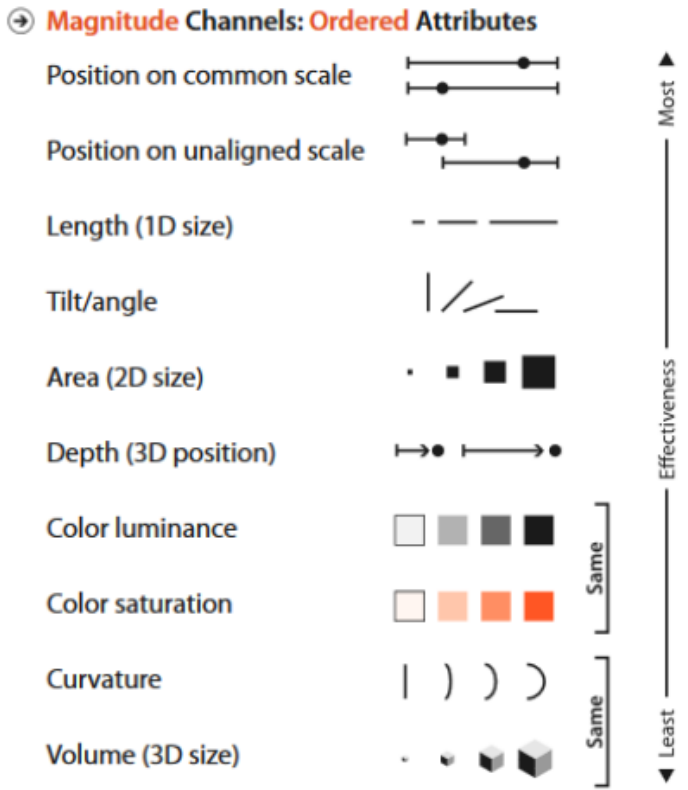
Rules of Thumb (Regras de Ouro)

Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - **Idéia errada:** “Se em duas dimensões já é bom, em três dimensões vai ser melhor ainda! Afinal, nós vivemos num mundo tridimensional”
  - Existe grande dificuldade em codificar visualmente informação com a terceira dimensão espacial: profundidade.
  - 3D só é fácil de se justificar quando as tarefas do usuário envolvem compreensão de forma de estruturas inerentemente tridimensionais, como por exemplo, dados médicos (raio-x, por exemplo).

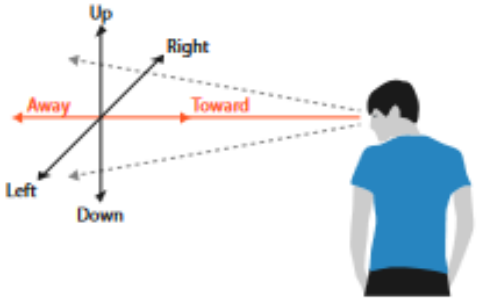
Rules of Thumb (Regras de Ouro)

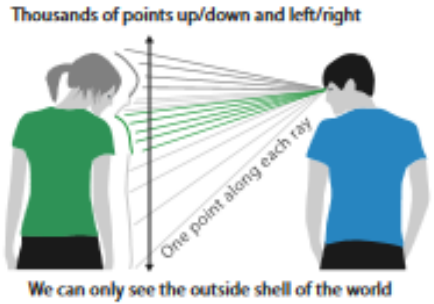
- **Não usar 3D sem justificativa!**
  - Canais de posição espacial de ranking mais alto: posição espacial PLANAR - e não profundidade;



Rules of Thumb (Regras de Ouro)

- **Os perigos da profundidade:**
  - Nós não vivemos realmente em 3D: nós vemos em 2.05D
    - Obtemos mais informação no plano de imagem 2D rapidamente pelo movimento dos olhos;
    - Obtemos informação de profundidade de forma mais devagar, pelo movimento da cabeça e do corpo.

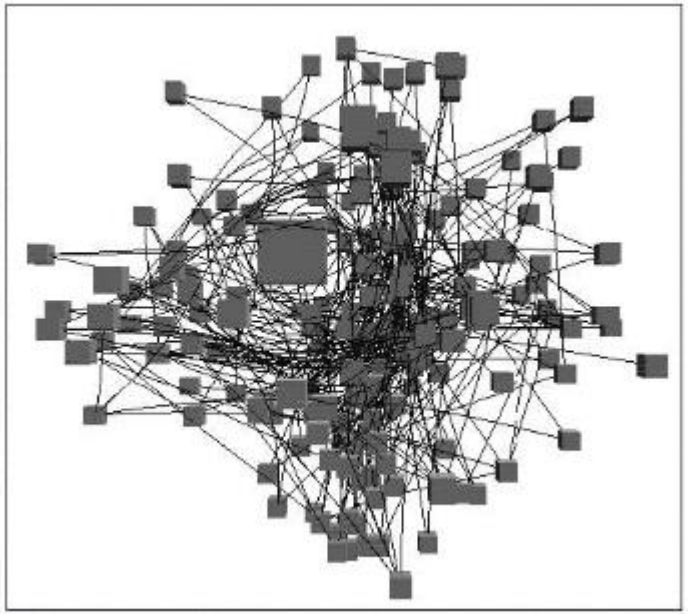




À esquerda, basta o movimento dos olhos para observação dos dados contido em X, Y; à direita, é necessário mais esforço para percepção de informações devido à profundidade.

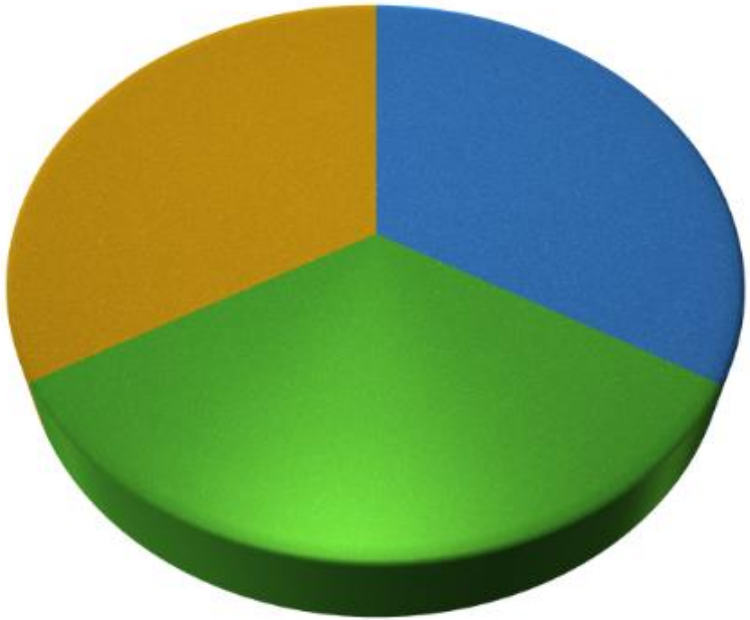
Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - **Oclusão ou sobreposição:** uma das características mais importante na percepção de profundidade;
  - Alguns objetos não podem ser vistos porque eles estão escondidos atrás de outros;
  - Complexidade de interação.



Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - Qual das três partes é a maior?

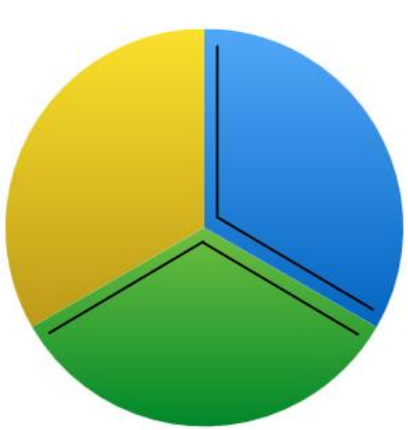


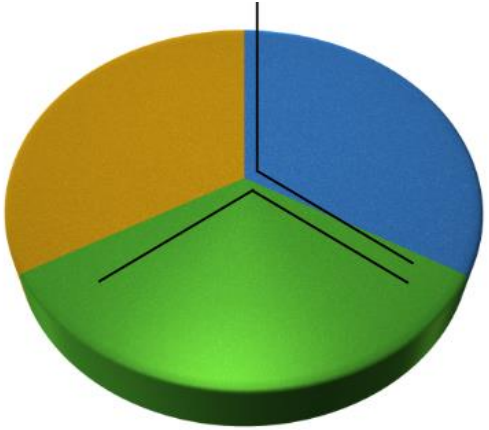
Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - Qual das três partes é a maior?
    - São todas do mesmo tamanho!

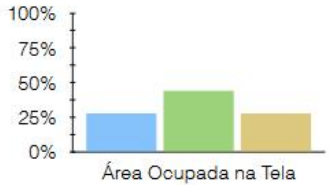
Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - Qual das três partes é a maior? Veja a área ocupada em tela:





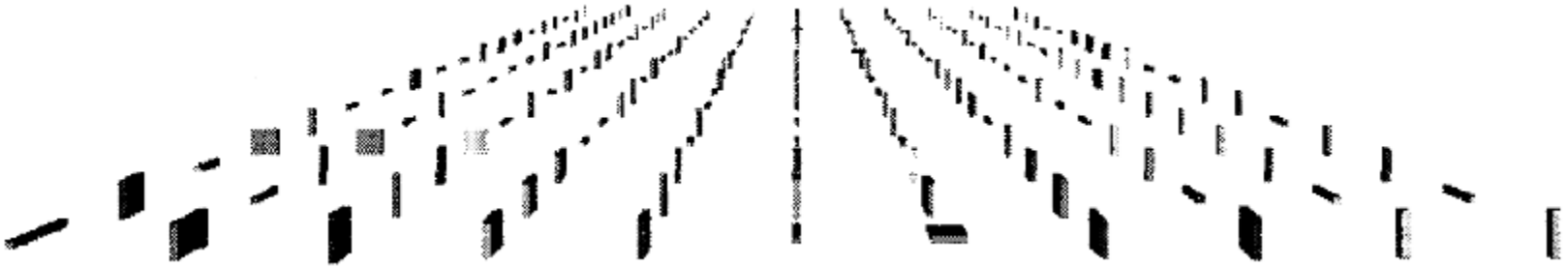
O *Pie Chart* em 2D ocupa o mesmo espaço para cada fatia (33.3%) em tela.



O *Pie Chart* em 3D tende a dar mais espaço em tela para a fatia mais próxima do usuário.

Rules of Thumb (Regras de Ouro)

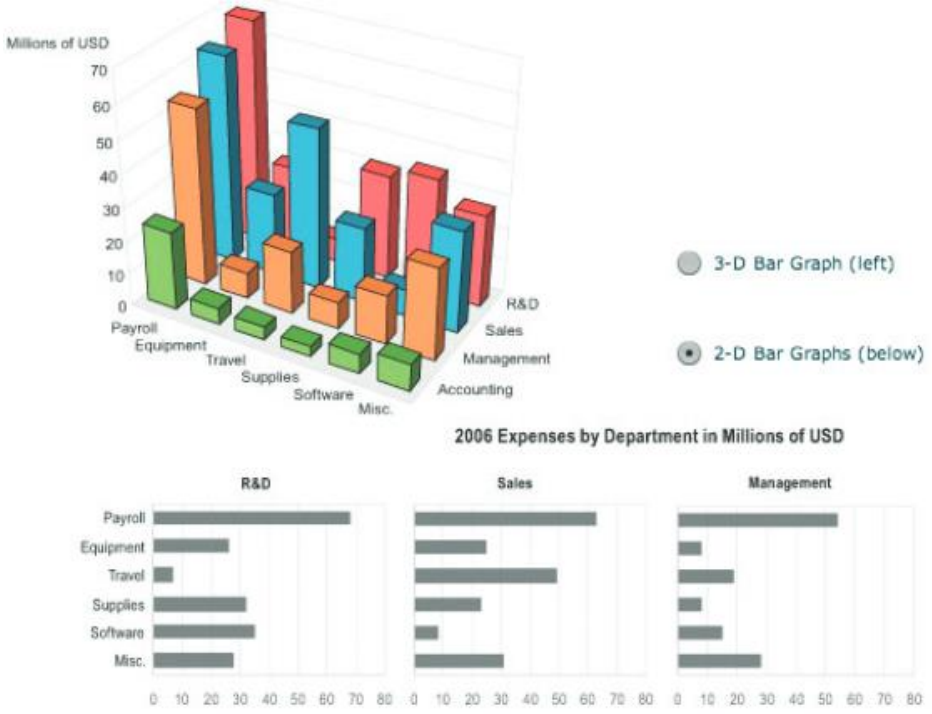
- **Não usar 3D sem justificativa!**
  - Distorção de perspectiva perde informação
    - Interfere com todas as codificações de canal de tamanho.



Rules of Thumb (Regras de Ouro)

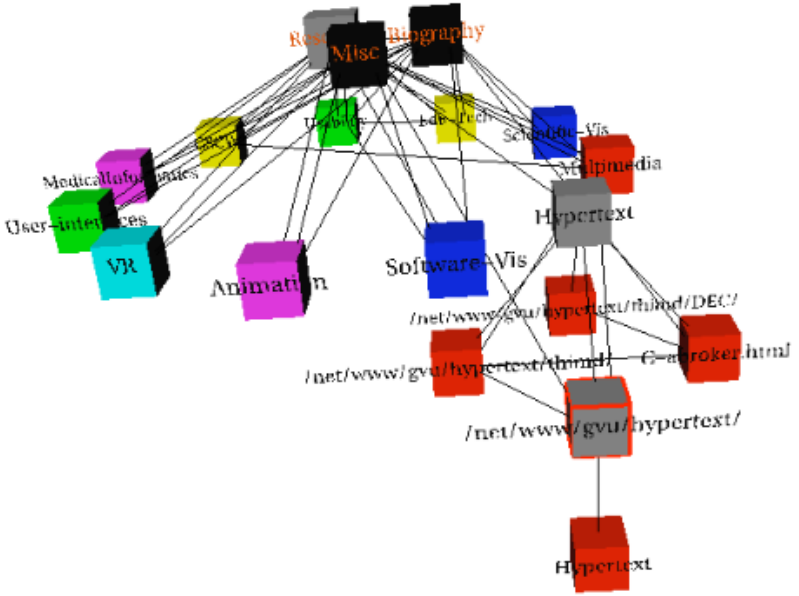
- **Não usar 3D sem justificativa!**

- No gráfico de barras ao lado, temos a sensação que Payroll de R&D ultrapassa o valor máximo do eixo Y (que é 70); entretanto, na legenda vemos que não é verdade.
- Barras pequenas ficam atrás das barras grandes.



Rules of Thumb (Regras de Ouro)

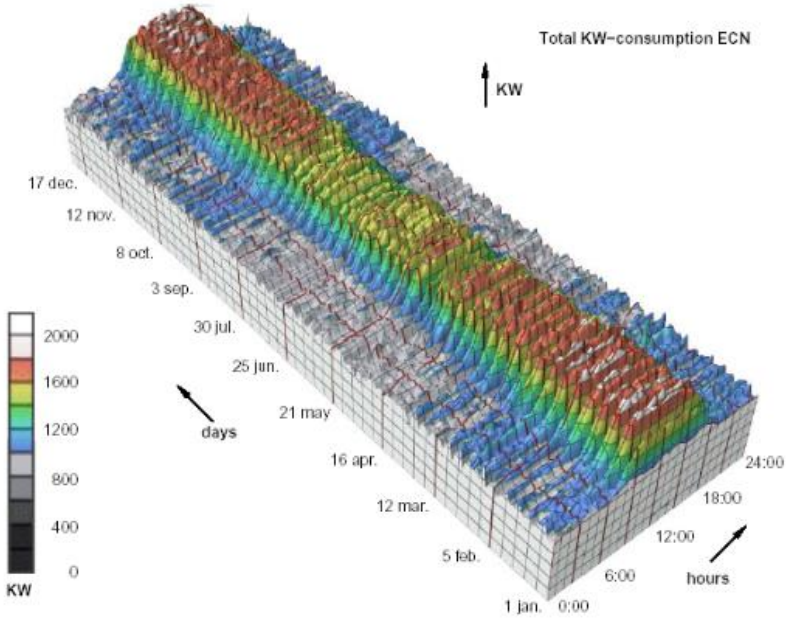
- **Não usar 3D sem justificativa!**
  - Texto inclinado não é legível:
    - A legibilidade do texto piora quando inclinado e afastado do plano de imagem;
    - Fontes de texto são cuidadosamente projetadas para visibilidade máxima em ambiente 2D.



[Visualizing the World-Wide Web with the Navigational View Builder, Mukherjea and Foley. Computer Networks and ISDN Systems, 1995.]

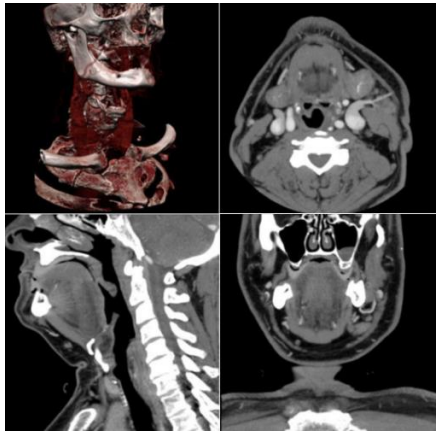
Rules of Thumb (Regras de Ouro)

- **Não usar 3D sem justificativa!**
  - Extrusão de curvas:
    - Impossível de fazer comparações detalhadas.
    - Apenas os padrões em grande-escala são visíveis

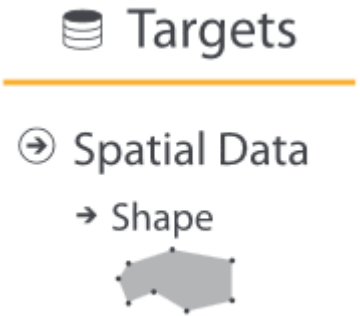


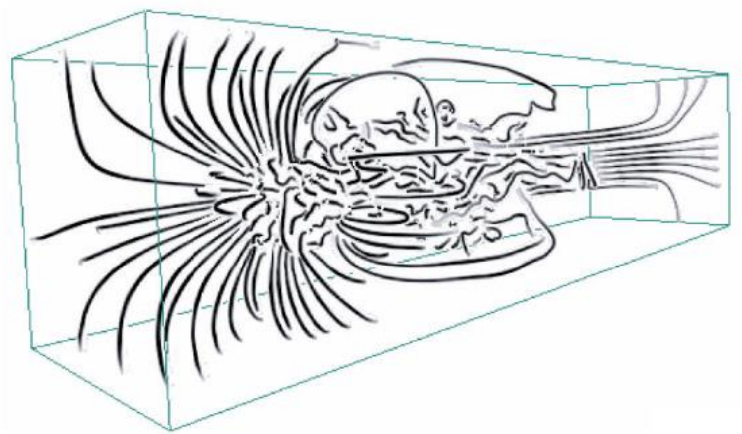
Sobre o gráfico: maior consumo de energia (KW) durante as horas de trabalho e a variação entre verão e inverno.

Rules of Thumb (Regras de Ouro)



- **Não usar 3D sem justificativa!**
  - Benefício do 3D:
    - Quando o objetivo da tarefa for mostrar dados inerentemente espaciais.





[Image-Based Streamline Generation and Rendering. Li and Shen. IEEE Trans. Visualization and Computer Graphics (TVCG) 13:3 (2007), 630–640.]

Rules of Thumb (Regras de Ouro)

- **Não usar 2D sem justificativa!**
  - A disposição dos dados no espaço 2D também deve ser explicitamente justificada, em comparação com a alternativa de simplesmente mostrar os dados com uma lista 1D (em forma de *array*).

Rules of Thumb (Regras de Ouro)

- **Não usar 2D sem justificativa!**
  - As vezes é mais aconselhável mostrar listas 1D do que gerar grafos/ árvores:
    - Os layouts 2D, como representações *redes*, requerem consideravelmente mais espaço para mostrar o mesmo número de rótulos;
    - 2D têm uma densidade de informações notavelmente mais alta e as vezes complexa;
    - Listas 1D são excelentes para tarefas onde os dados são ordenados;
      - Listas ordenadas de forma crescentes/ alfabéticas são melhores de visualizar valores específicos do que grafo, que dependendo do tamanho a busca se faz nó a nó.

PLOTNINE

Introdução

PlotNine é uma biblioteca gráfica para visualização de dados;

Ela é baseada na função *ggplot()* da linguagem R;

Documentação rica de exemplos: https://plotnine.readthedocs.io/en/stable/

Estrutura

Basicamente a *ggplot()* do Plotnine possui a seguinte estrutura:

```
(ggplot(data,aesthetics)
 + layer1()
 + layer2()
 + layerN()
)
```

data: matriz de dados
aes(x,y): matriz aesthetic que receberá duas colunas de data que serão x e y no plot
layers: camadas que informações de canais

Toy Dataset

weather = pd.read_csv('https://raw.githubusercontent.com/alanjones2/dataviz/master/london2018.csv')

```
In [4]: weather
```

Out[4]:

	Year	Month	Tmax	Tmin	Rain	Sun
0	2018	1	9.7	3.8	58.0	46.5
1	2018	2	6.7	0.6	29.0	92.0
2	2018	3	9.8	3.0	81.2	70.3
3	2018	4	15.5	7.9	65.2	113.4
4	2018	5	20.8	9.8	58.4	248.3
5	2018	6	24.2	13.1	0.4	234.5
6	2018	7	28.3	16.4	14.8	272.5
7	2018	8	24.5	14.5	48.2	182.1
8	2018	9	20.9	11.0	29.4	195.0
9	2018	10	16.5	8.5	61.0	137.0
10	2018	11	12.2	5.8	73.8	72.9
11	2018	12	10.7	5.2	60.6	40.3

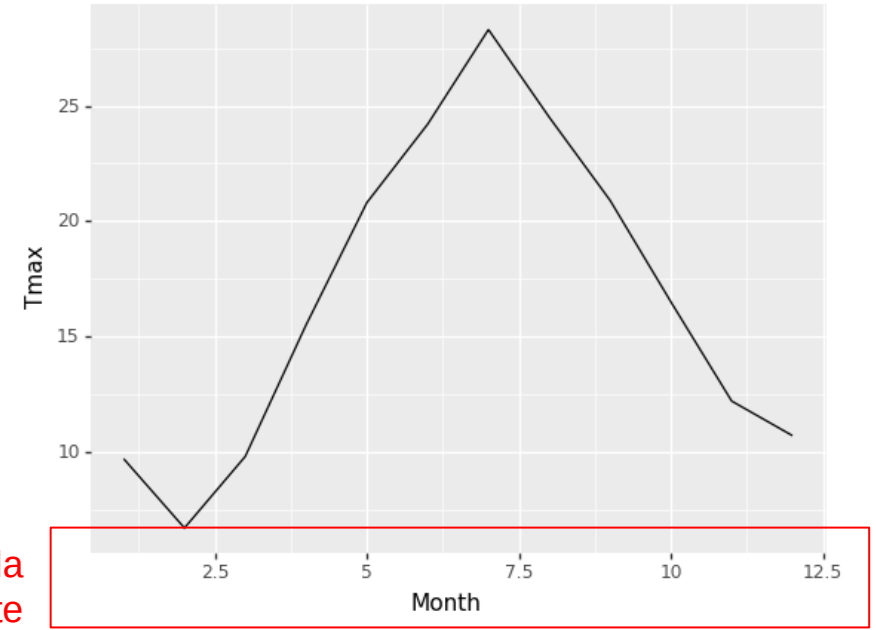
Dataset

Nosso Toy Dataset será derivado do UK Meteorological Office que contém dados do clima de Londres em 2018, registrando a temperatura mínima e máxima (em graus ºC) de cada mês e o número de horas de intensidade solar e de chuvas.

geom_line()

Estrutura básica:
Matriz, Eixo X e Eixo Y

```
In [5]: (ggplot(weather,aes('Month', 'Tmax'))
          + geom_line()
        )
```



Escala gerada automaticamente

geom_line()

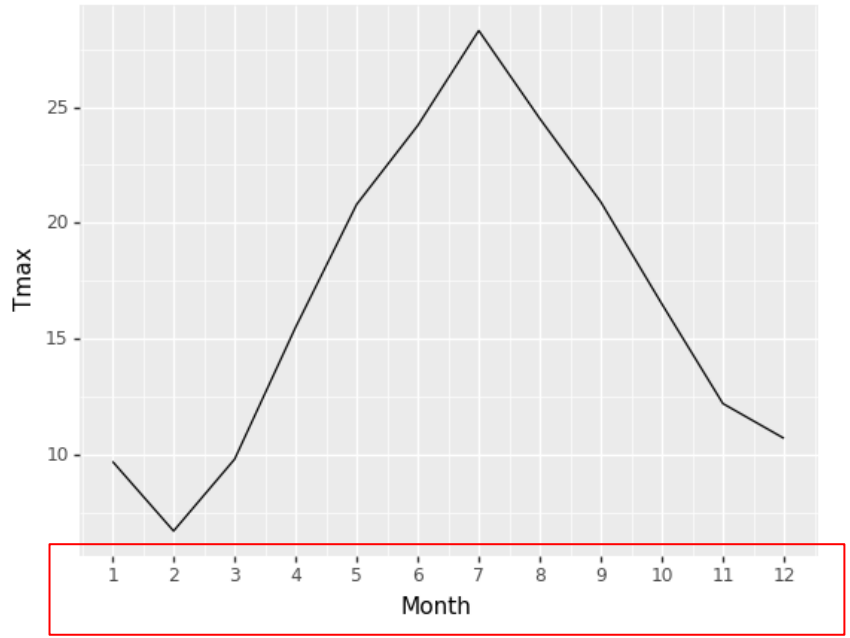
```
In [6]: months = weather['Month']

In [7]: months

Out[7]: 0      1
        1      2
        2      3
        3      4
        4      5
        5      6
        6      7
        7      8
        8      9
        9     10
        10    11
        11    12
        Name: Month, dtype: int64
```

Ajustando eixo X

```
In [8]: (ggplot(weather,aes('Month', 'Tmax'))
          + geom_line()
          + scale_x_continuous(breaks=months)
        )
```

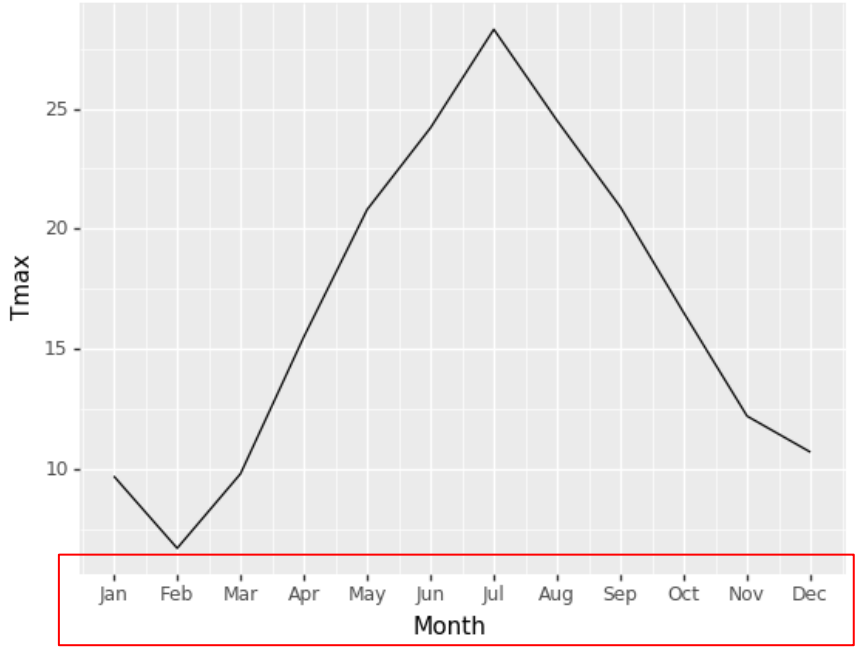


geom_line()

```
In [9]: month_labels=("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec")
```

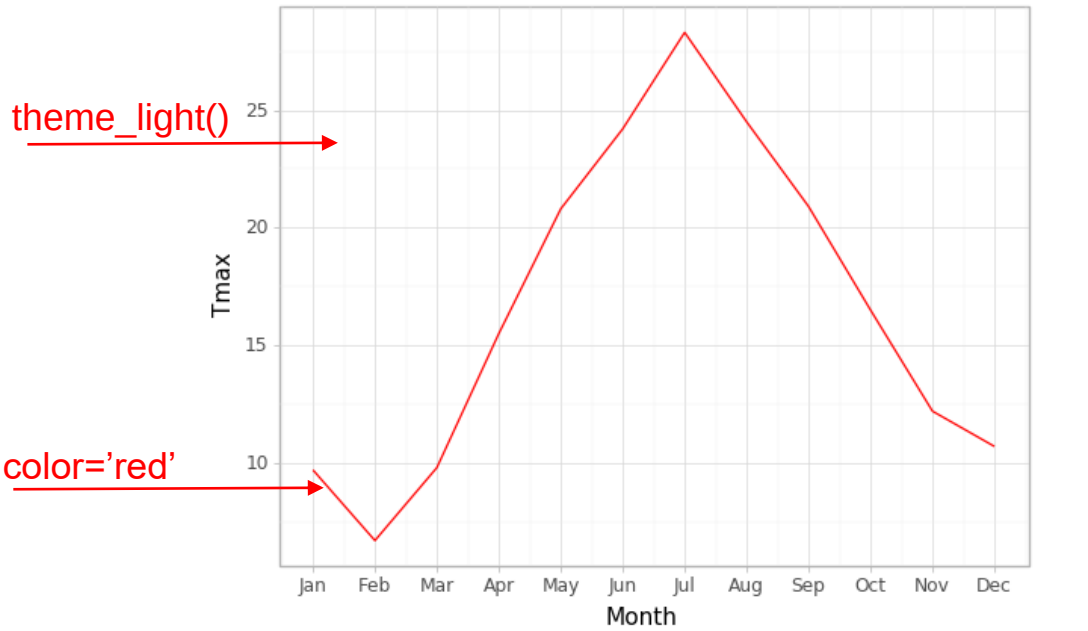
Ajustando labels

```
In [10]: (ggplot(weather,aes('Month', 'Tmax'))
           + geom_line()
           + scale_x_continuous(breaks=months, labels=month_labels)
         )
```



geom_line()

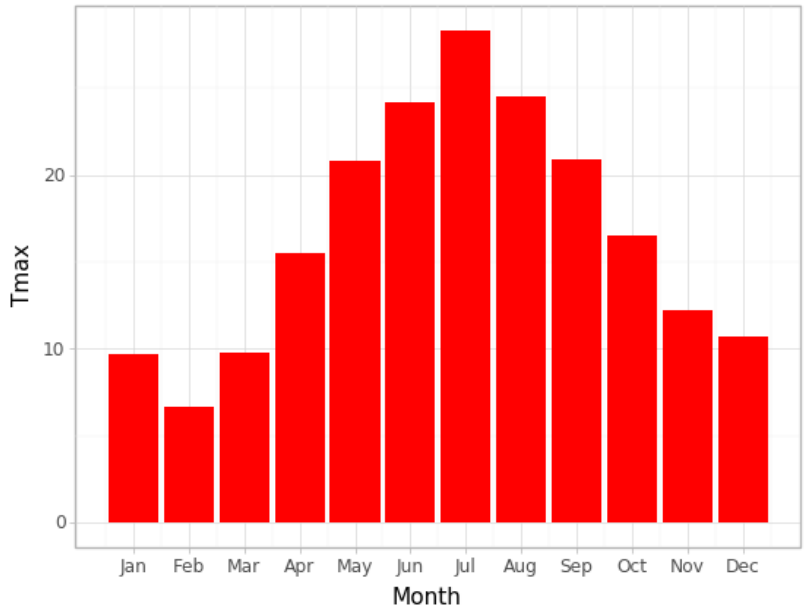
```
In [11]: (ggplot(weather,aes('Month', 'Tmax'))
           + geom_line(color='red')
           + scale_x_continuous(breaks=months, labels=month_labels)
           + theme_light()
         )
```



geom_col()

Para *plotarmos* o gráfico anterior na versão gráfico de colunas, basta substituir *geom_line* por *geom_col*.
Atente-se que o parâmetro muda de *color* pra *fill*.

```
In [12]: (ggplot(weather,aes('Month', 'Tmax'))
           + geom_col(fill='red')
           + scale_x_continuous(breaks=months, labels=month_labels)
           + theme_light()
         )
```



Gráficos Sobrepostos

Vamos gerar um novo *DataFrame* chamado *temps*:
Usaremos a função *melt* do pandas para fazer junção de colunas;
Queremos mostrar gráfico de barras contendo a temperatura máxima e mínima (em ºC) de cada mês.

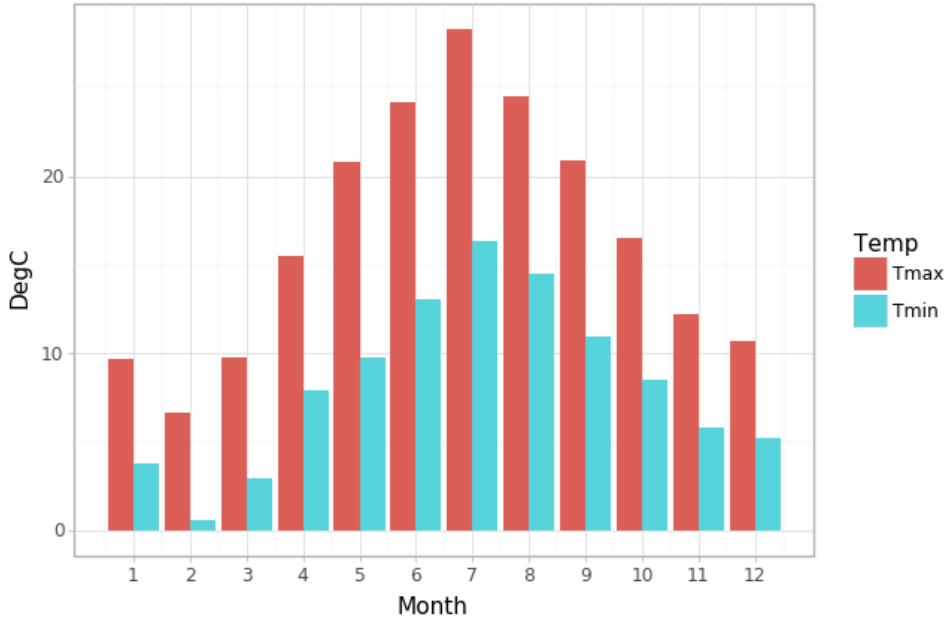
```
In [14]: temps = pd.melt(weather, id_vars=['Month'], value_vars=['Tmax', 'Tmin'],
         var_name='Temp', value_name='DegC' )
         temps
```

Out[14]:

	Month	Temp	DegC
0	1	Tmax	9.7
1	2	Tmax	6.7
2	3	Tmax	9.8
3	4	Tmax	15.5
4	5	Tmax	20.8
5	6	Tmax	24.2
6	7	Tmax	28.3
7	8	Tmax	24.5
8	9	Tmax	20.9
9	10	Tmax	16.5
10	11	Tmax	12.2
11	12	Tmax	10.7
12	1	Tmin	3.8
13	2	Tmin	0.6
14	3	Tmin	3.0
15	4	Tmin	7.9
16	5	Tmin	9.8
17	6	Tmin	13.1
18	7	Tmin	16.4
19	8	Tmin	14.5
20	9	Tmin	11.0
21	10	Tmin	8.5
22	11	Tmin	5.8
23	12	Tmin	5.2

Gráficos Sobrepostos

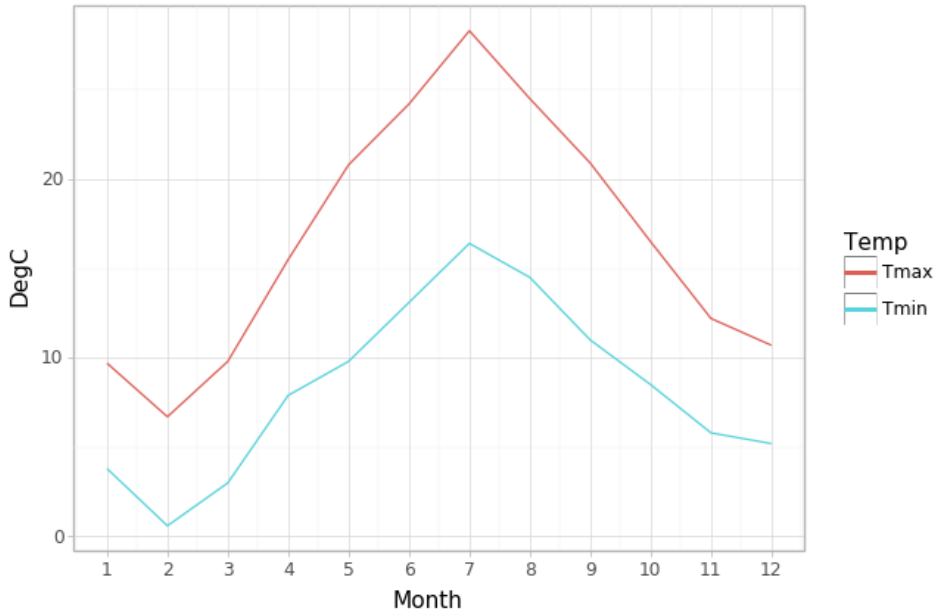
```
In [15]: (ggplot(temps,aes('Month','DegC',fill='Temp'))
           + geom_col(position='dodge')
           + scale_x_continuous(breaks=months)
           + theme_light()
         )
```



Gráficos Sobrepostos

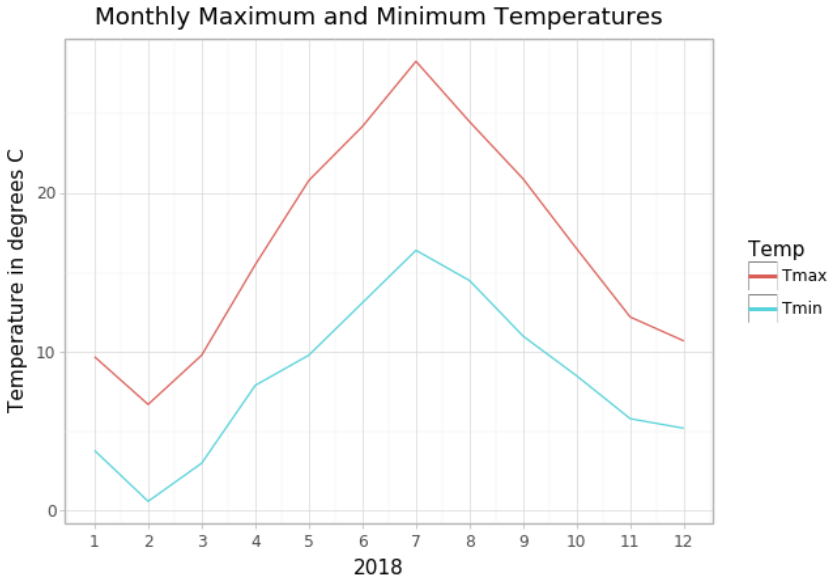
Substituindo geom_col por geom_line.

```
In [16]: (ggplot(temps,aes('Month','DegC',color='Temp'))
           + geom_line()
           + scale_x_continuous(breaks=months)
           + theme_light()
         )
```



Adicionando LABELS

```
In [17]: (ggplot(temps,aes('Month','DegC',color='Temp'))
           + geom_line()
           + scale_x_continuous(breaks=months)
           + theme_light()
           + xlab('2018')
           + ylab('Temperature in degrees C')
           + ggtitle('Monthly Maximum and Minimum Temperatures')
         )
```



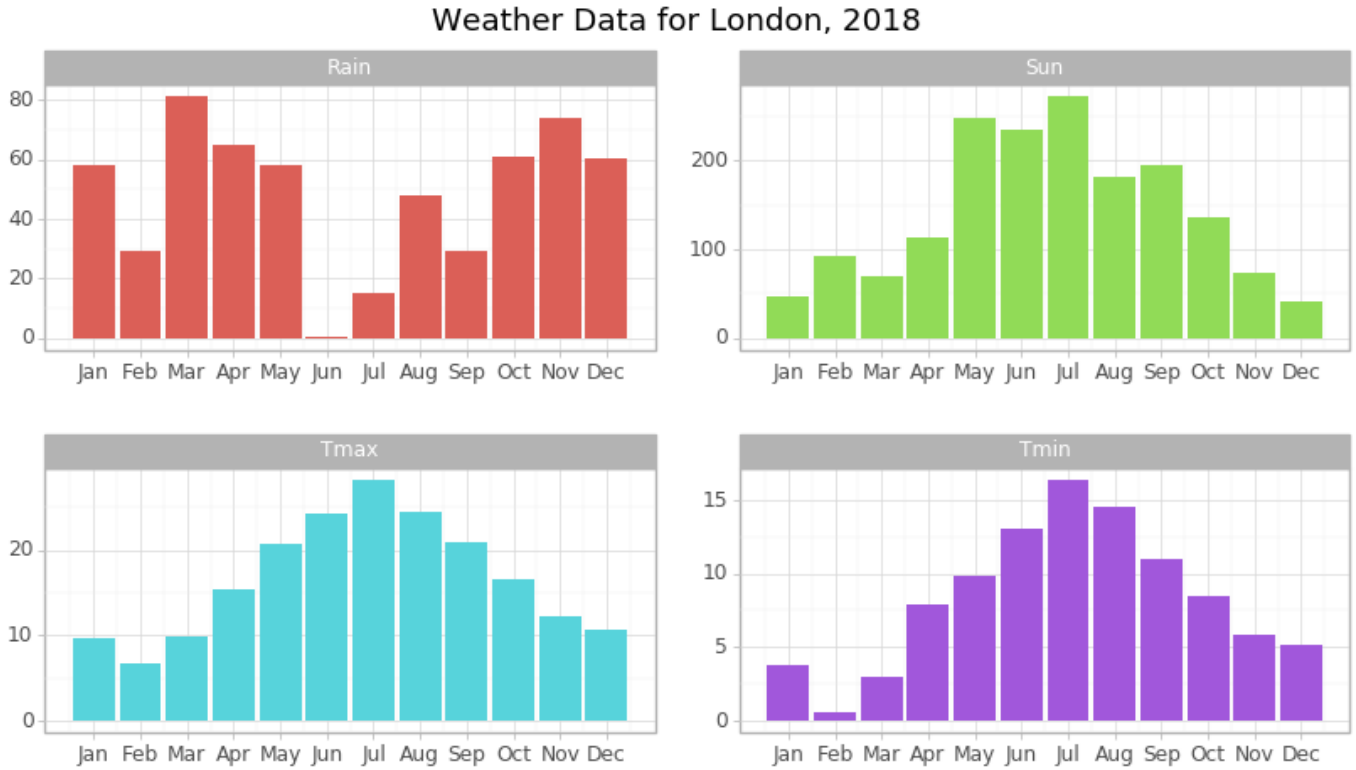
Vários gráficos na tela

In [18]:
```
data = pd.melt(weather, id_vars=['Month'], value_vars=['Tmax','Tmin','Rain','Sun'], var_name='Measure', value_name='Value')
```

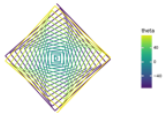
In [19]:
```
(ggplot(data, aes('Month','Value', fill='Measure'))
  + geom_col(show_legend=False)
  + scale_x_continuous(breaks=months,labels=month_labels)
  + facet_wrap('Measure', scales='free')
  + xlab('')
  + ylab('')
  + ggtitle('Weather Data for London, 2018')
  + theme_light()
  + theme(panel_spacing=0.5, figure_size=(10,5))
)
```

```
C:\ProgramData\Anaconda3\lib\site-packages\plotnine\facets\facet.py:552: PlotnineWarning: If you need more space for the x-axis tick text use ... + theme(subplots_adjust={'wspace': 0.25}). Choose an appropriate value for 'wspace'.
C:\ProgramData\Anaconda3\lib\site-packages\plotnine\facets\facet.py:558: PlotnineWarning: If you need more space for the y-axis tick text use ... + theme(subplots_adjust={'hspace': 0.25}). Choose an appropriate value for 'hspace'
```

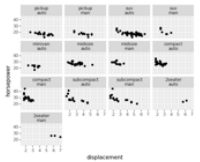
Vários gráficos na tela

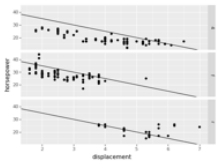


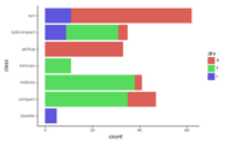
Mais gráficos do PlotNine:

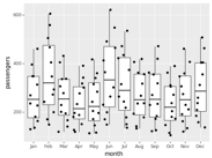

Spiral Animation

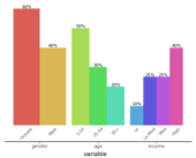

Facet grid

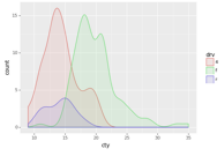

Facet wrap

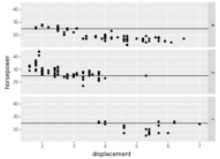

AB line

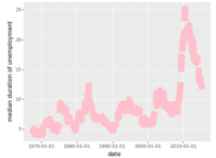

Bar chart


A box and whiskers plot

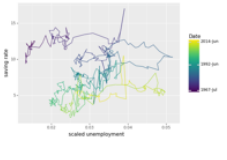

Two Variable Bar Plot

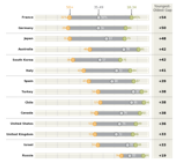

Density Plot


Horizontal line

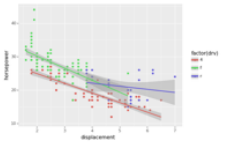

Line plots


The Political Territories of Westeros


Path plots

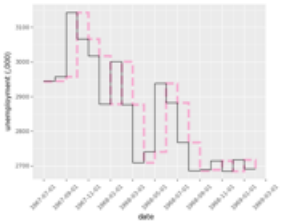

Ranges of Similar Variables


Change in Rank

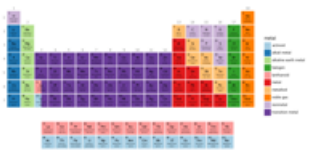

Smoothed conditional means

https://plotnine.readthedocs.io/en/stable/gallery.html

Mais gráficos do PlotNine:

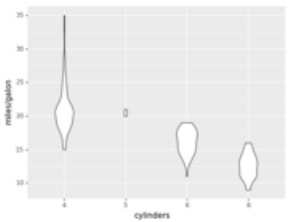


Step plots

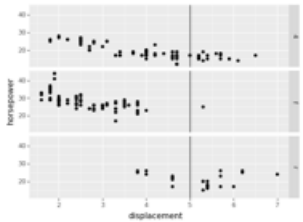


Periodic Table of Elements

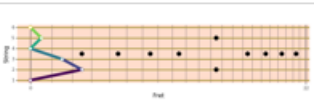
Annotated Heatmap



Violin plot



Vertical line



Guitar Neck

https://plotnine.readthedocs.io/en/stable/gallery.html

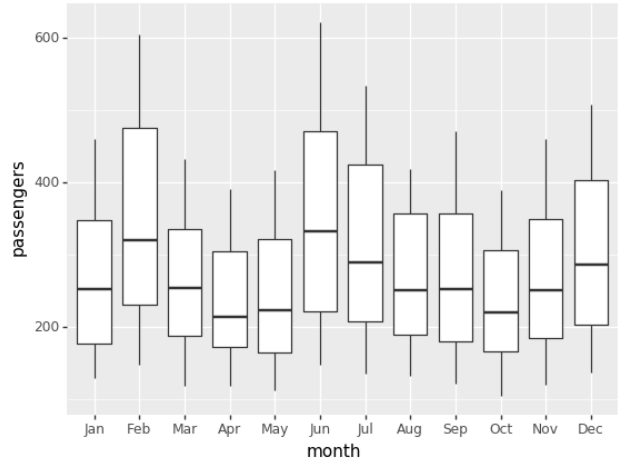
Exemplos de gráficos (e sua simplicidade)

Basic boxplot

[3]:
```
months = [month[:3] for month in flights.month[:12]]
print(months)
```

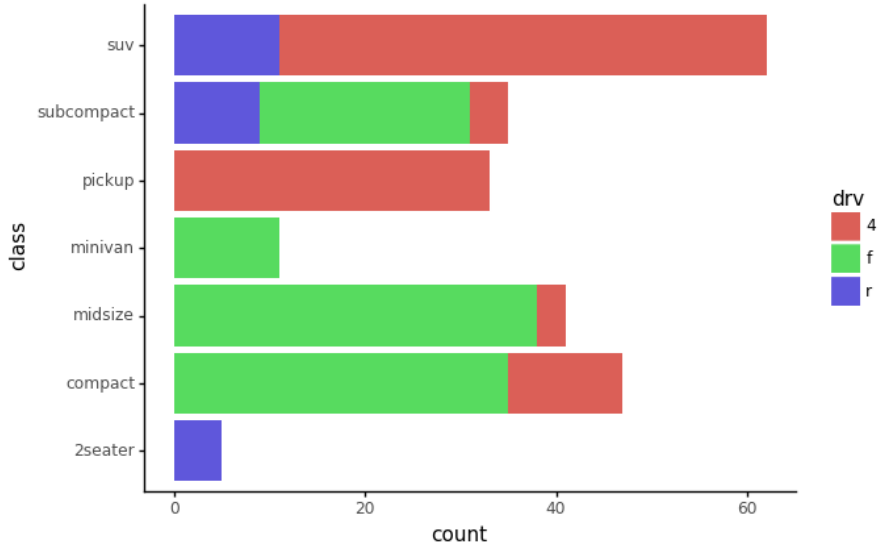
```
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

[4]:
```
(
    ggplot(flights)
    + geom_boxplot(aes(x='factor(month)', y='passengers'))
    + scale_x_discrete(labels=months, name='month')  # change ticks labels on OX
)
```



https://plotnine.readthedocs.io/en/stable/generated/plotnine.geoms.geom_boxplot.html#a-box-and-whiskers-plot

[5]:
```
(
    ggplot(mpg)
    + geom_bar(aes(x='class', fill='drv'))
    + coord_flip()
    + theme_classic()
)
```



https://plotnine.readthedocs.io/en/stable/generated/plotnine.geoms.geom_bar.html#bar-chart