Data visualization and reshaping

Lecture 3

IFT6758, Fall 2020

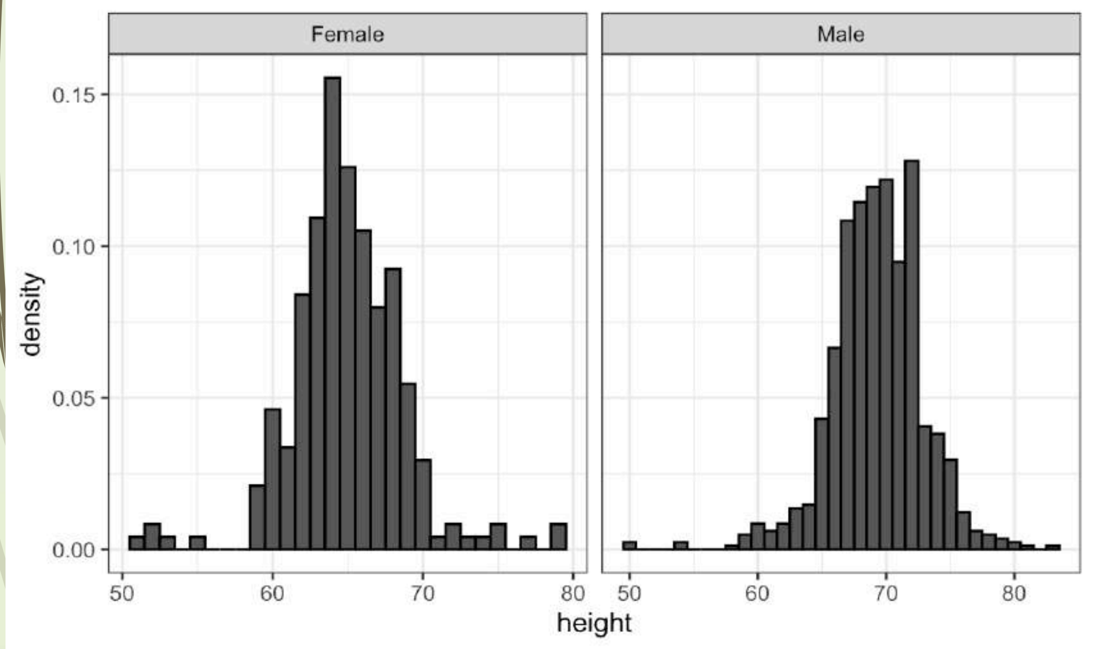
Principles of Data Visualization: **4. Ease comparisons**

Principles of Data Visualization: **4. Ease comparisons**

- **(i) Use common axes**

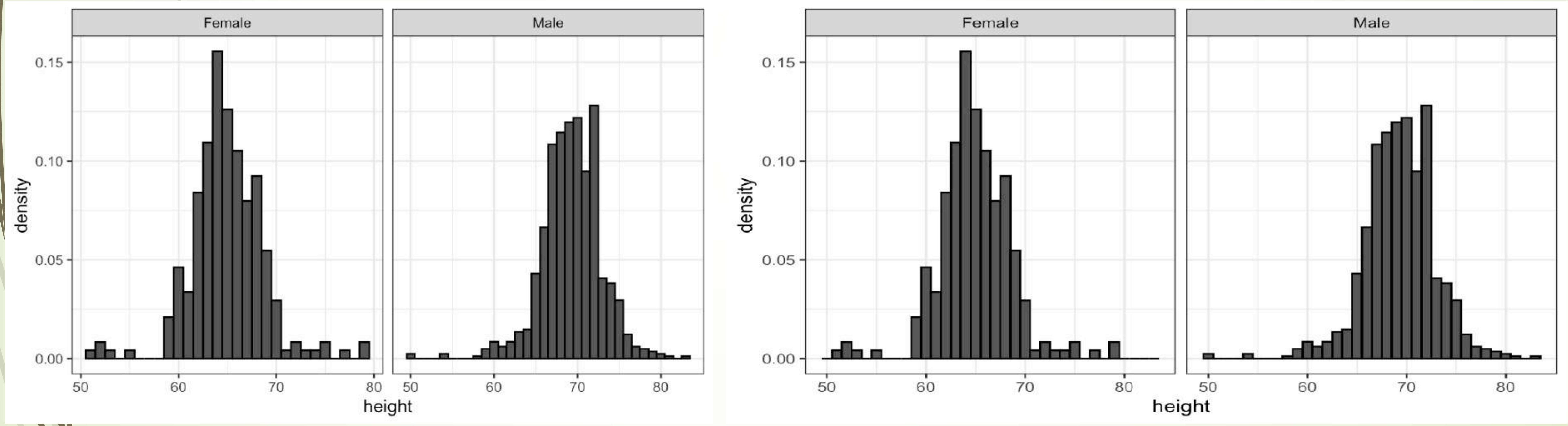
Principles of Data Visualization: **4. Ease comparisons**

- **(i) Use common axes**



Principles of Data Visualization: **4. Ease comparisons**

- **(i) Use common axes**



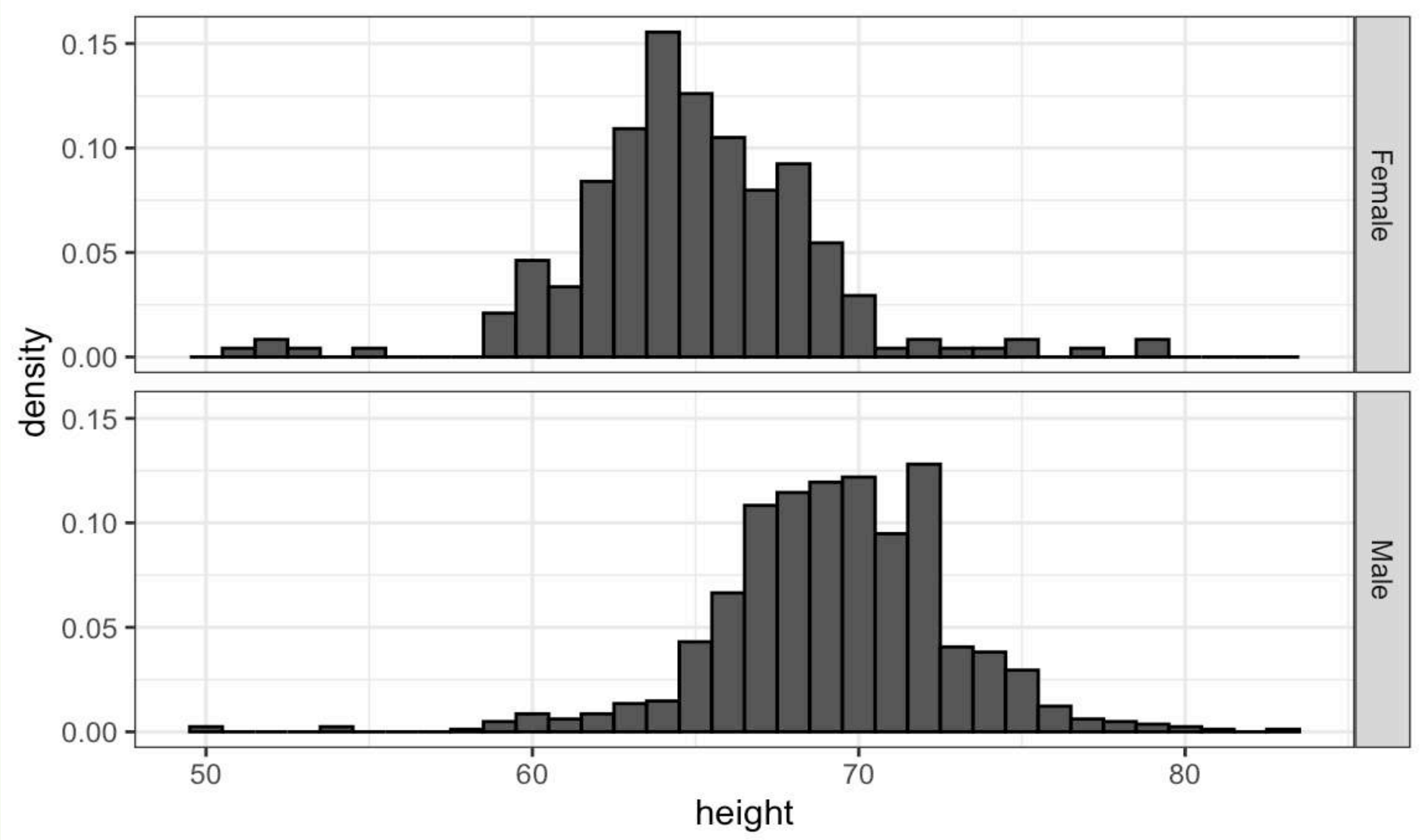
Principles of Data Visualization: **4. Ease comparisons**

Principles of Data Visualization: **4. Ease comparisons**

- **(ii) Align plots vertically to see horizontal changes and horizontally to see vertical changes**

Principles of Data Visualization: **4. Ease comparisons**

- **(ii) Align plots vertically to see horizontal changes and horizontally to see vertical changes**

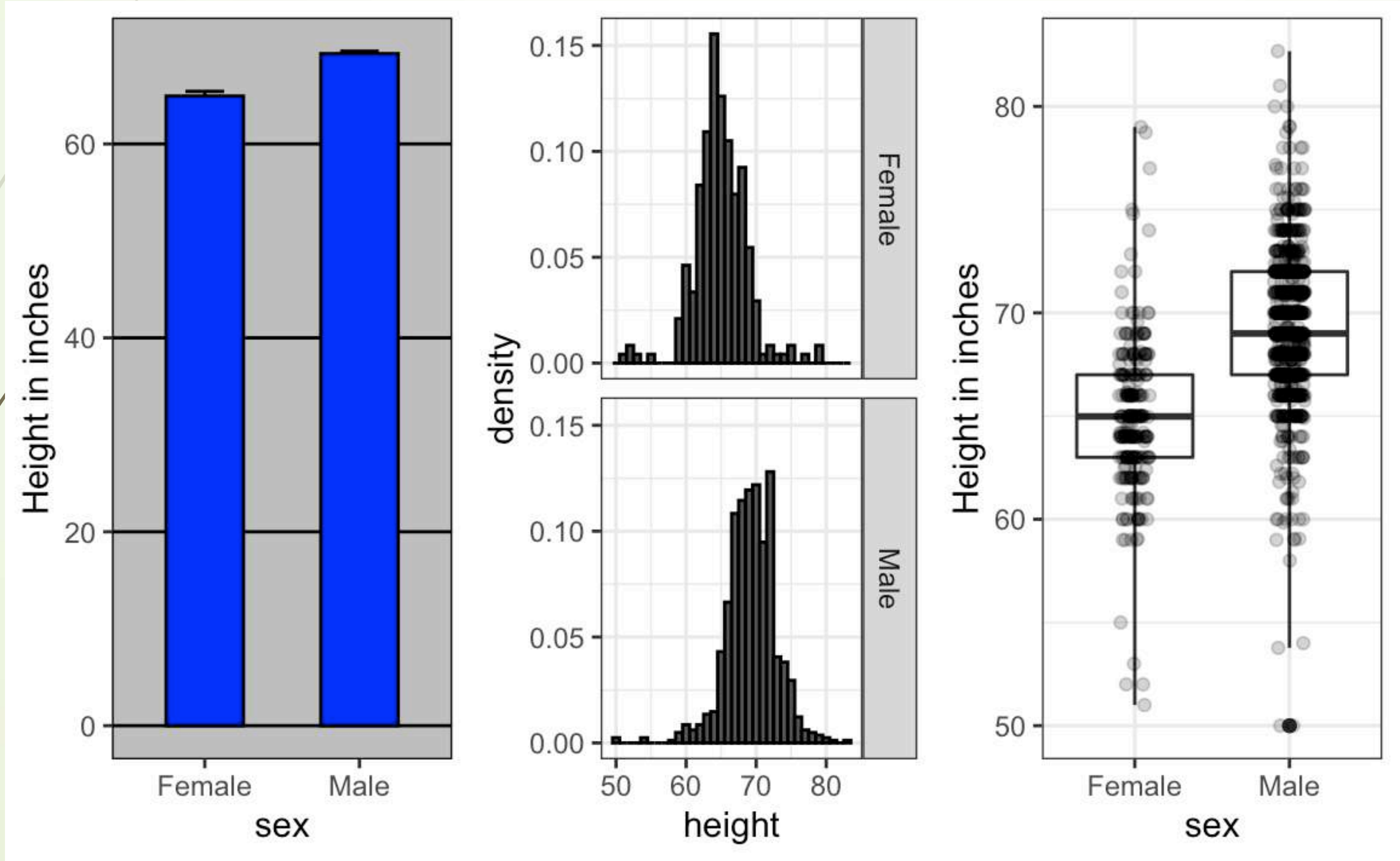


Principles of Data Visualization: **4. Ease comparisons**

- **(iii) Use appropriate representation to facilitate distribution and summary**

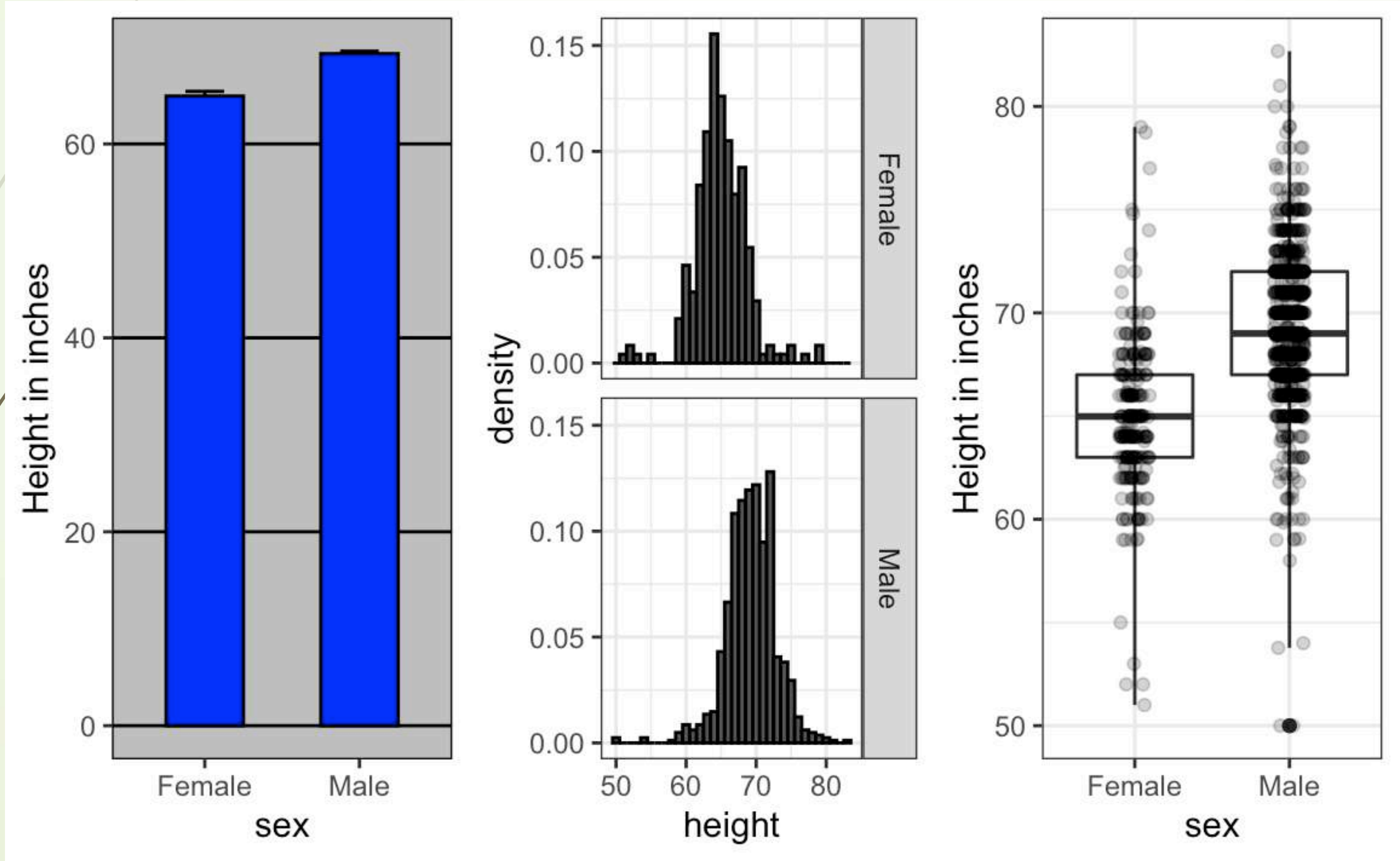
Principles of Data Visualization: **4. Ease comparisons**

- **(iii) Use appropriate representation to facilitate distribution and summary**



Principles of Data Visualization: **4. Ease comparisons**

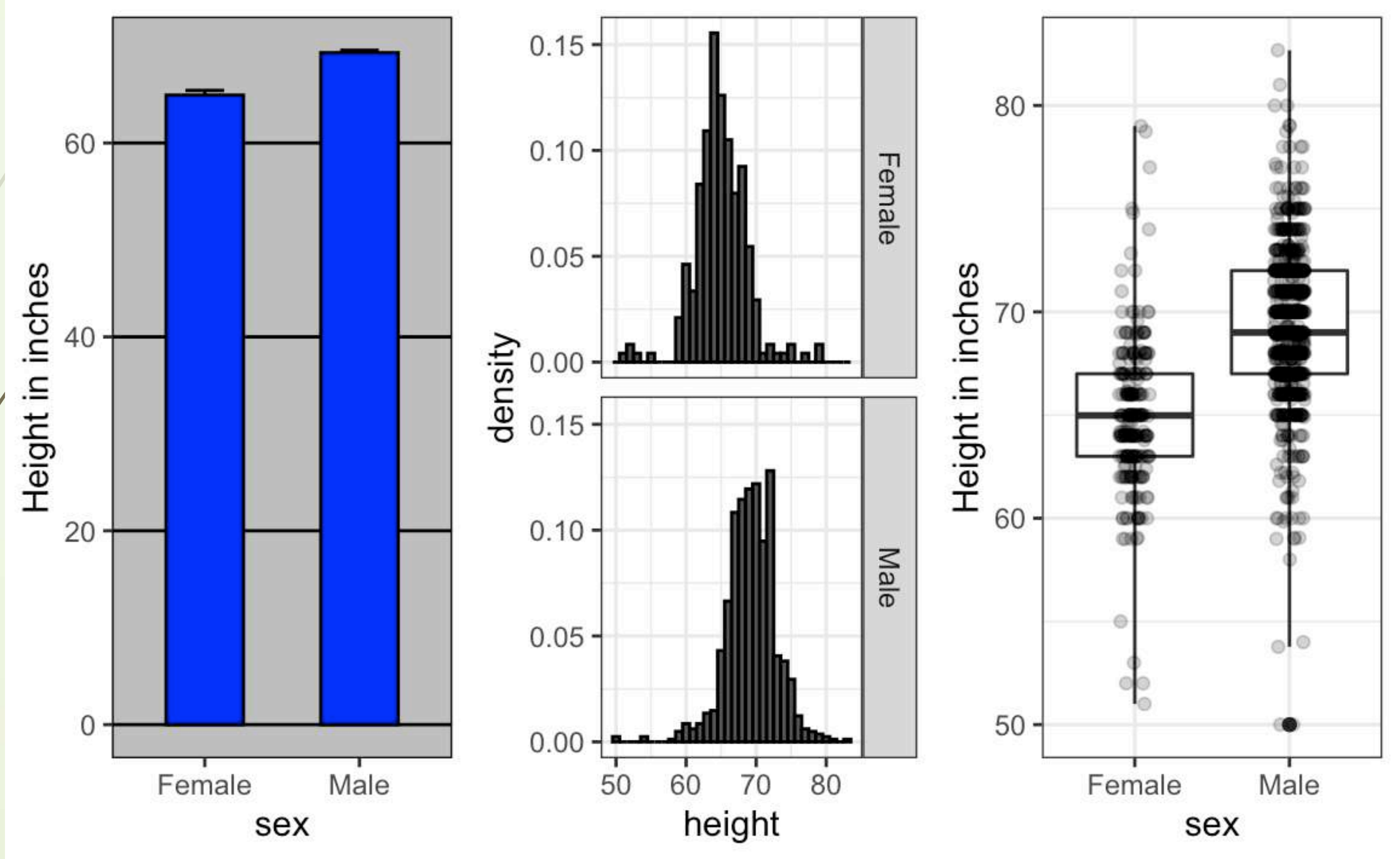
- **(iii) Use appropriate representation to facilitate distribution and summary**



matplotlib: pybeeswarm

Principles of Data Visualization: **4. Ease comparisons**

- **(iii) Use appropriate representation to facilitate distribution and summary**

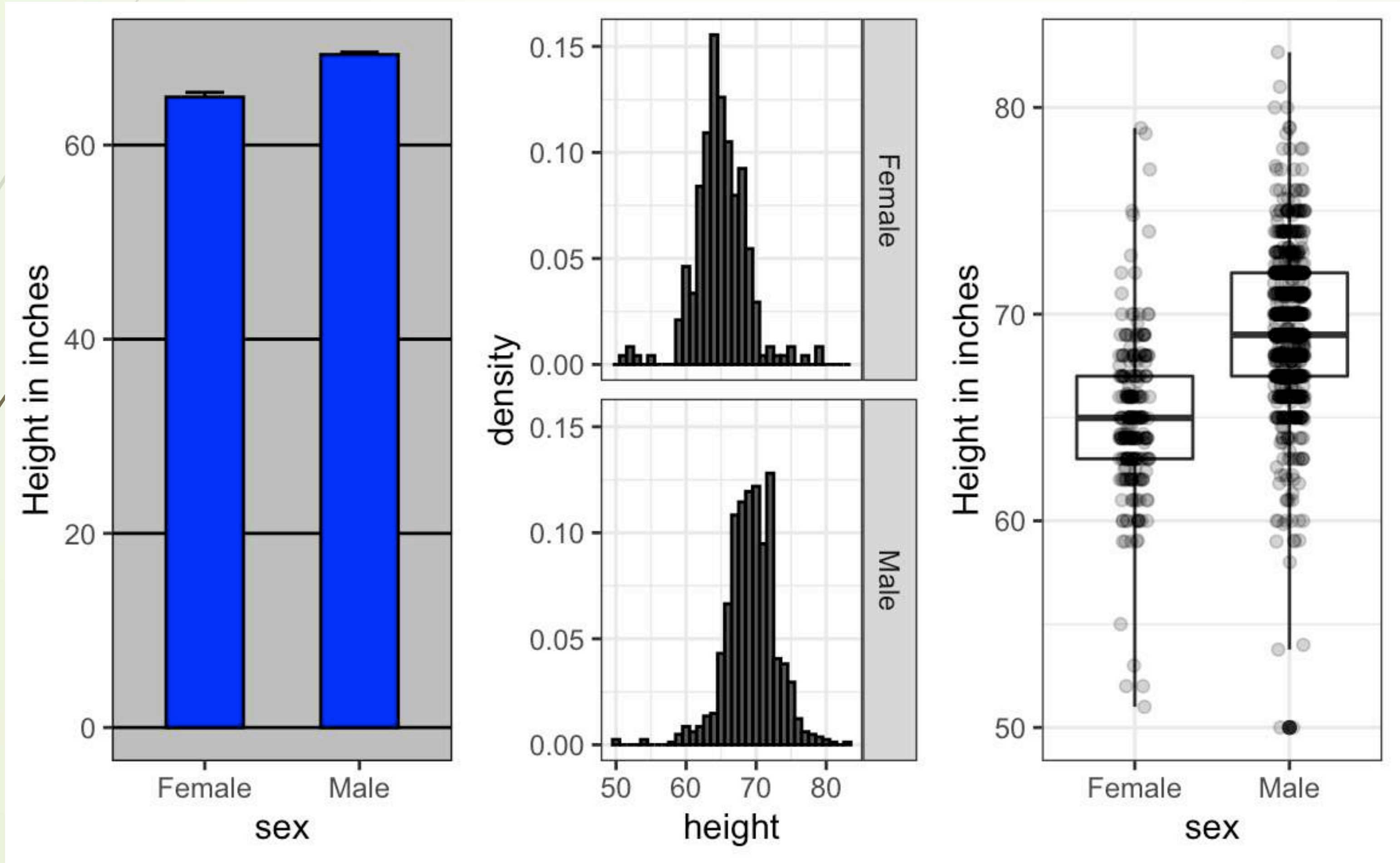


matplotlib: pybeeswarm

```
from beeswarm import *
import matplotlib.pyplot as plt
import numpy as np
d1 = np.random.uniform(low=-3, high=3, size=100)
d2 = np.random.normal(size=100)
```

Principles of Data Visualization: **4. Ease comparisons**

- **(iii) Use appropriate representation to facilitate distribution and summary**



matplotlib: pybeeswarm

```
from beeswarm import *
import matplotlib.pyplot as plt
import numpy as np
d1 = np.random.uniform(low=-3, high=3, size=100)
d2 = np.random.normal(size=100)

bs, ax = beeswarm([d1,d2], method="swarm", labels=["sample 1", "sample 2"], col=["blue","red"])
```

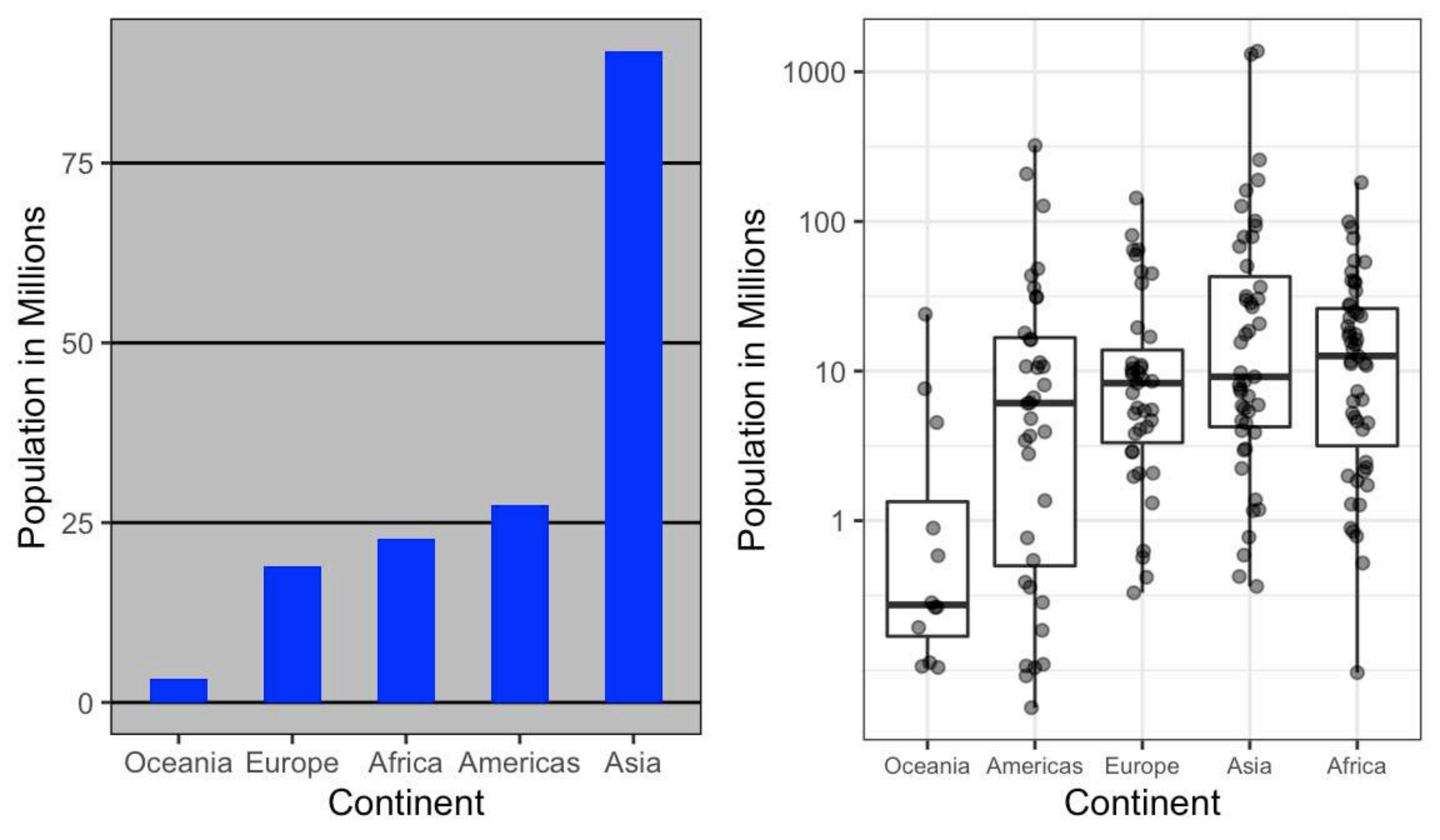
Principles of Data Visualization: **4. Ease comparisons**

Principles of Data Visualization: **4. Ease comparisons**

- **(iv) Consider transformations**

Principles of Data Visualization: **4. Ease comparisons**

- **(iv) Consider transformations**



Principles of Data Visualization: **5. Think of the color blind**

Principles of Data Visualization: **5. Think of the color blind**

- **Seaborn color pallettes**
- Color Brewer website provides some guidance on **which palettes are color blind safe.**
- Seaborn choose_colorbrewer_palette: interactive widget to browse options and tweak parameters; must be used in a Jupyter notebook
- Example code to understand how the the seaborn color palettes compare for different type of colorblindness (e.g., deuteranopia, protan)
- There is a variety of kinds of color blindness, but the **most common** variant leads to **difficulty distinguishing reds and greens**.
- Seaborn colorblind palette:

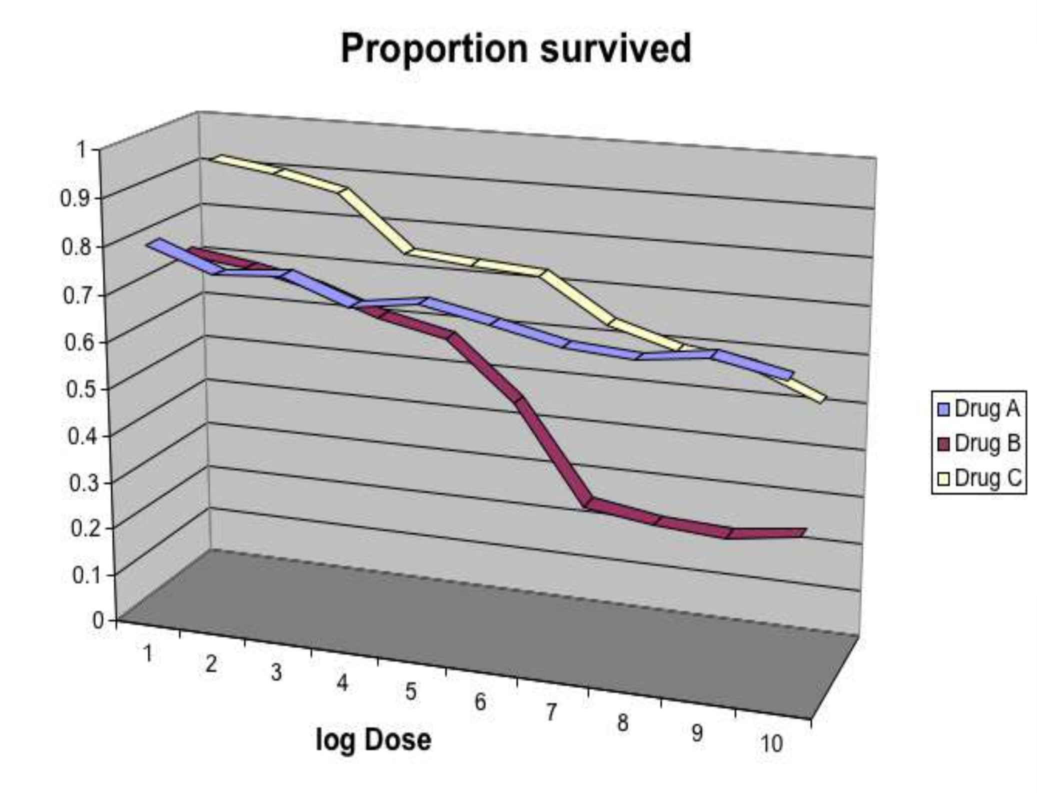
Principles of Data Visualization: **5. Think of the color blind**

- **Seaborn color pallettes**
- Color Brewer website provides some guidance on **which palettes are color blind safe.**
- Seaborn choose_colorbrewer_palette: interactive widget to browse options and tweak parameters; must be used in a Jupyter notebook
- Example code to understand how the the seaborn color palettes compare for different type of colorblindness (e.g., deuteranopia, protan)
- There is a variety of kinds of color blindness, but the **most common** variant leads to **difficulty distinguishing reds and greens.**
- Seaborn colorblind palette:

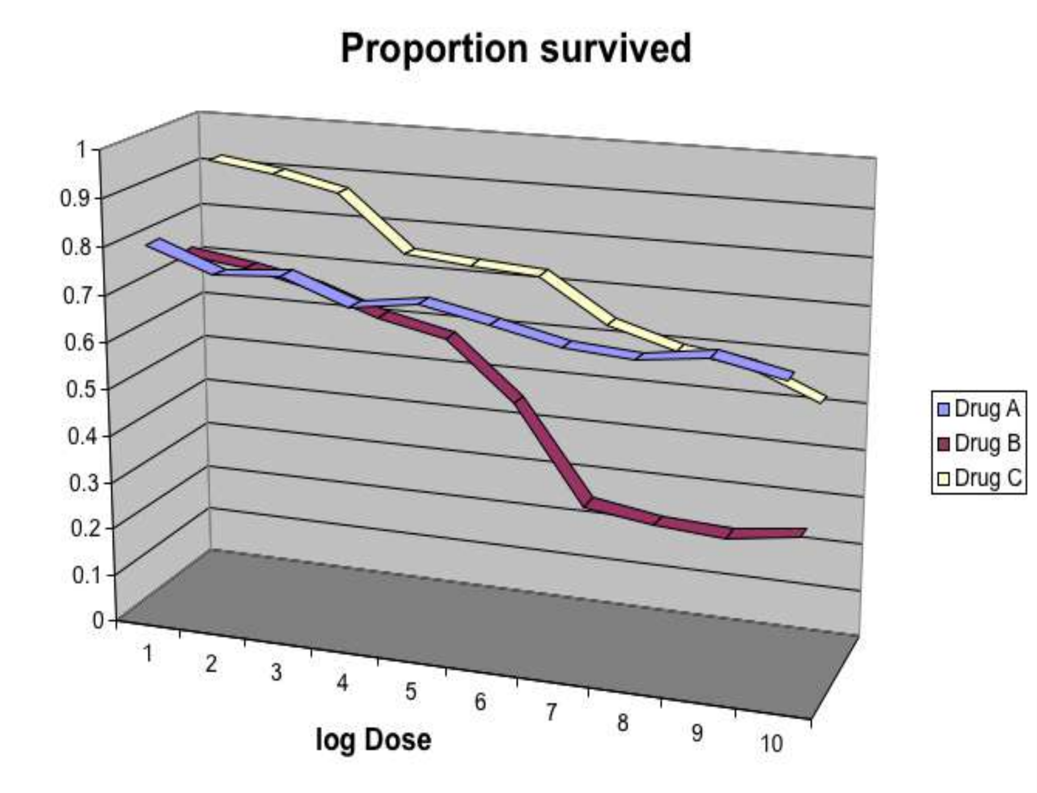
*Example code: color_pal = sns.color_palette("**colorblind**", 6).as_hex()*

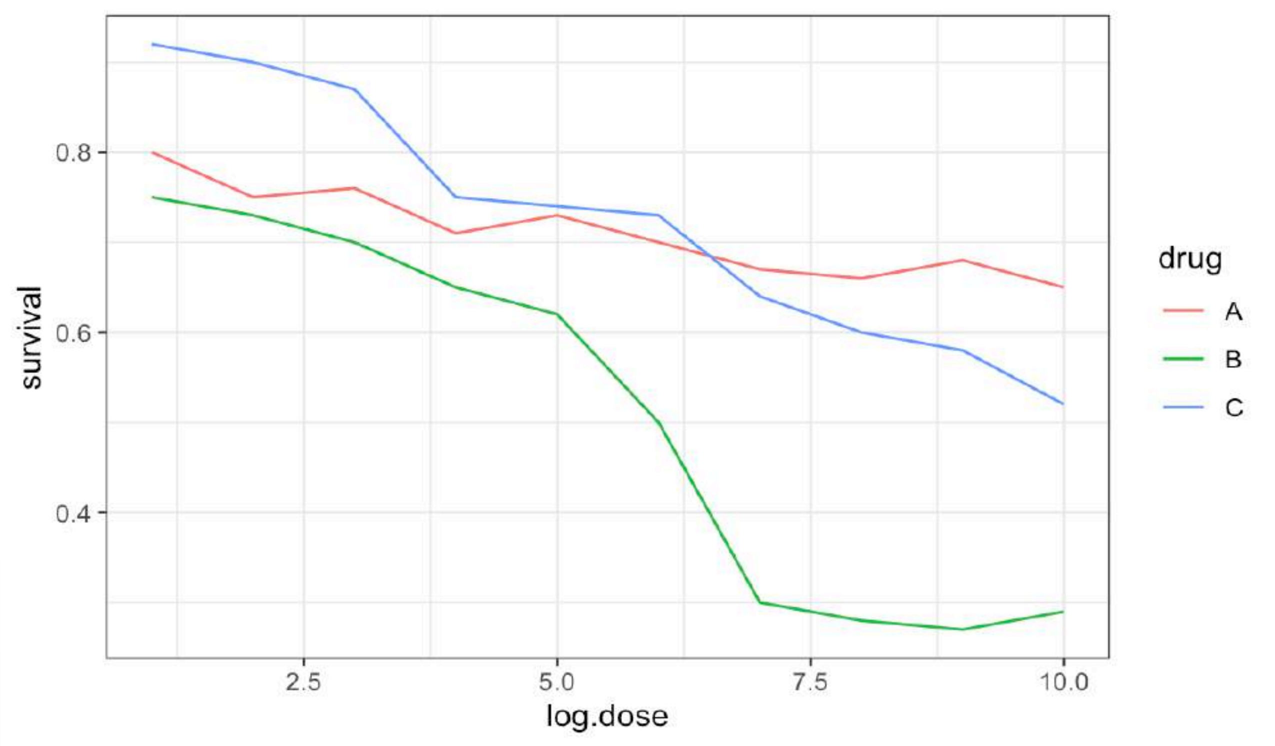
Principles of Data Visualization: **6. Avoid pseudo-three-dimensional plots**

Principles of Data Visualization: **6. Avoid pseudo-three-dimensional plots**



Principles of Data Visualization: **6. Avoid pseudo-three-dimensional plots**





Principles of Data Visualization: **7. Avoid too many significant digits**

Principles of Data Visualization: **7. Avoid too many significant digits**

state	year	Measles	Pertussis	Polio
California	1940	37.8826320	18.3397861	0.8266512
California	1950	13.9124205	4.7467350	1.9742639
California	1960	14.1386471	NA	0.2640419
California	1970	0.9767889	NA	NA
California	1980	0.3743467	0.0515466	NA

Principles of Data Visualization: **7. Avoid too many significant digits**

state	year	Measles	Pertussis	Polio
California	1940	37.8826320	18.3397861	0.8266512
California	1950	13.9124205	4.7467350	1.9742639
California	1960	14.1386471	NA	0.2640419
California	1970	0.9767889	NA	NA
California	1980	0.3743467	0.0515466	NA

state	year	Measles	Pertussis	Polio
California	1940	37.9	18.3	0.8
California	1950	13.9	4.7	2.0
California	1960	14.1	NA	0.3
California	1970	1.0	NA	NA
California	1980	0.4	0.1	NA

Principles of Data Visualization: **8. Place values being compared on columns rather than on rows**

Principles of Data Visualization: **8. Place values being compared on columns rather than on rows**

state	year	Measles	Pertussis	Polio
California	1940	37.9	18.3	0.8
California	1950	13.9	4.7	2.0
California	1960	14.1	NA	0.3
California	1970	1.0	NA	NA
California	1980	0.4	0.1	NA

Principles of Data Visualization: **8. Place values being compared on columns rather than on rows**

state	year	Measles	Pertussis	Polio
California	1940	37.9	18.3	0.8
California	1950	13.9	4.7	2.0
California	1960	14.1	NA	0.3
California	1970	1.0	NA	NA
California	1980	0.4	0.1	NA

state	disease	1940	1950	1960	1970	1980
California	Measles	37.9	13.9	14.1	1	0.4
California	Pertussis	18.3	4.7	NA	NA	0.1
California	Polio	0.8	2.0	0.3	NA	NA

Robust summary

Robust summary

- Robustness against **Outliers**

Robust summary

- Robustness against **Outliers**

- **Examples:** Old machine giving out nonsensical measurements; human error in expressing distance in miles instead of kilometers or putting decimal point in wrong places

Robust summary

- Robustness against **Outliers**

- **Examples:** Old machine giving out nonsensical measurements; human error in expressing distance in miles instead of kilometers or putting decimal point in wrong places

- **How to detect outliers?**

Robust summary

- Robustness against **Outliers**

- **Examples:** Old machine giving out nonsensical measurements; human error in expressing distance in miles instead of kilometers or putting decimal point in wrong places

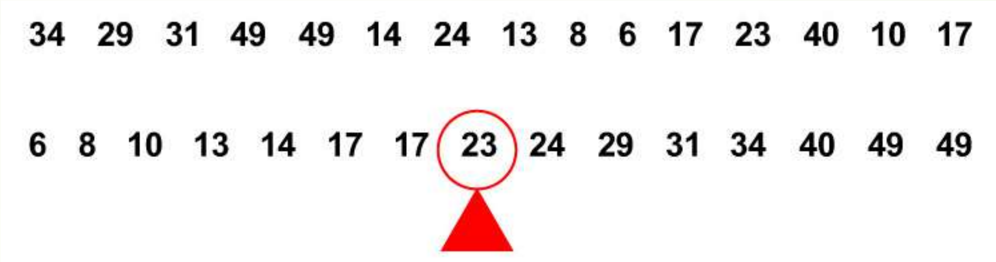
- **How to detect outliers?**
  - **Boxplot:** briefly introduced before
  - Need notions of **Median, IQR**..

- $Q1 - 1.5 \times (Q3 - Q1), Q3 + 1.5 \times (Q3 - Q1)]$.

- **Median:** 50 percentile.

- $Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)]$.

- **Median:** 50 percentile.



- $Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)]$.

- **Median:** 50 percentile.
- **Inter-quartile range (IQR):** 50% data within the range
- $Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)]$.

- **Median:** 50 percentile.
- **Inter-quartile range (IQR):** 50% data within the range
  - The difference between the **3rd and 1st quartile** (or **75th and 25th percentiles**)
- $$Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)].$$

- $QQ1-1.5\times(QQ3-QQ1),QQ3+1.5\times(QQ3-QQ1)].$
- *Median*: 50 percentile.
- **Inter-quartile range (IQR):** 50% data within the range
  - The difference between the **3rd and 1st quartile** (or **75th and 25th percentiles**)
- **Tukey's definition of an outlier:** $[Q1-1.5\times(Q3-Q1), Q3+1.5\times(Q3-Q1)]$.

- $QQ1-1.5\times(QQ3-QQ1),QQ3+1.5\times(QQ3-QQ1)].$
- *Median*: 50 percentile.

- **Inter-quartile range (IQR):** 50% data within the range
  - The difference between the **3rd and 1st quartile** (or **75th and 25th percentiles**)

- **Tukey's definition of an outlier:** $[Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)]$.

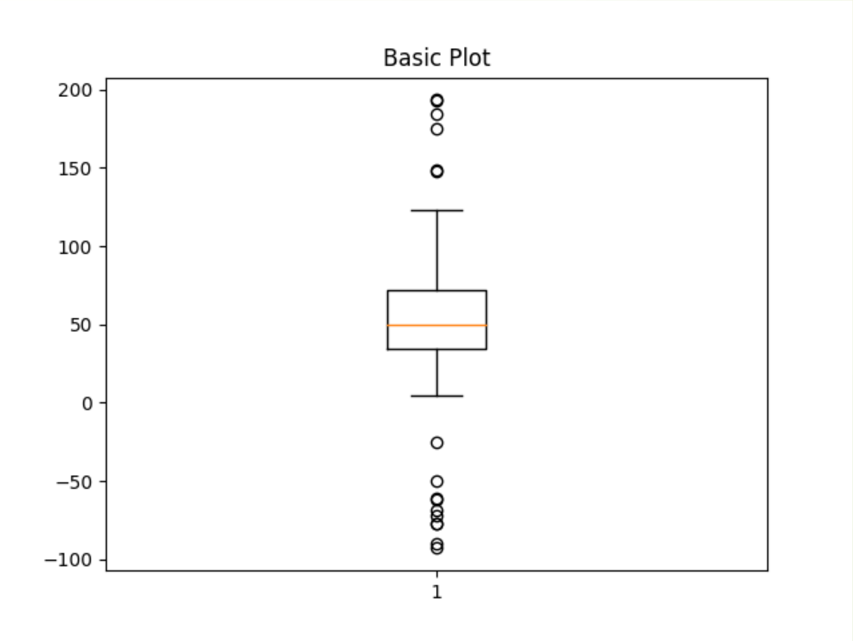
```
import numpy as np
import matplotlib.pyplot as plt
# Fixing random state for reproducibility
np.random.seed(19680801)
# create random data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```

- *QQ1−1.5×(QQ3−QQ1),QQ3+1.5×(QQ3−QQ1)].*
- *Median*: 50 percentile.

- **Inter-quartile range (IQR):** 50% data within the range
  - The difference between the **3rd and 1st quartile** (or **75th and 25th percentiles**)

- **Tukey's definition of an outlier:** $[Q1 - 1.5\times(Q3 - Q1), Q3 + 1.5\times(Q3 - Q1)]$.

```
import numpy as np
import matplotlib.pyplot as plt
# Fixing random state for reproducibility
np.random.seed(19680801)
# create random data
spread = np.random.rand(50) * 100
center = np.ones(25) * 50
flier_high = np.random.rand(10) * 100 + 100
flier_low = np.random.rand(10) * -100
data = np.concatenate((spread, center, flier_high, flier_low))
fig1, ax1 = plt.subplots()
ax1.set_title('Basic Plot')
ax1.boxplot(data)
```



Reshaping data with Pandas

```
import pandas as pd
```

```
full_data = pd.read_excel('Sample - Superstore.xls', sheet_name='Orders')
```

```
full_data.head(5)
```

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...	Postal Code	Region	Product ID	Category	Sub-Category	Product Name	Sales	Quantity	Discount
0	1	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-BO-10001798	Furniture	Bookcases	Bush Somerset Collection Bookcase	261.9600	2	0.00
1	2	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...	42420.0	South	FUR-CH-10000454	Furniture	Chairs	Hon Deluxe Fabric Upholstered Stacking Chairs,...	731.9400	3	0.00
2	3	CA-2017-138688	2017-06-12	2017-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...	90036.0	West	OFF-LA-10000240	Office Supplies	Labels	Self-Adhesive Address Labels for Typewriters b...	14.6200	2	0.00
3	4	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311.0	South	FUR-TA-10000577	Furniture	Tables	Bretford CR4500 Series Slim Rectangular Table	957.5775	5	0.45
4	5	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...	33311.0	South	OFF-ST-10000760	Office Supplies	Storage	Eldon Fold 'N Roll Cart System	22.3680	2	0.20

Reshaping data with Pandas

Reshaping data with Pandas

- Shape of data: ***full_data.shape***

Reshaping data with Pandas

Reshaping data with Pandas

Observation

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...
0	1	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...
1	2	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...
2	3	CA-2017-138688	2017-06-12	2017-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...
3	4	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...
4	5	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...

	Row ID	Order ID	Order Date	Ship Date	Ship Mode	Customer ID	Customer Name	Segment	Country	City	...
0	1	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...
1	2	CA-2017-152156	2017-11-08	2017-11-11	Second Class	CG-12520	Claire Gute	Consumer	United States	Henderson	...
2	3	CA-2017-138688	2017-06-12	2017-06-16	Second Class	DV-13045	Darrin Van Huff	Corporate	United States	Los Angeles	...
3	4	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...
4	5	US-2016-108966	2016-10-11	2016-10-18	Standard Class	SO-20335	Sean O'Donnell	Consumer	United States	Fort Lauderdale	...

Variable

Reshaping data with Pandas

Reshaping data with Pandas

- **Pivot table**

Reshaping data with Pandas

- **Pivot table**

```
selective_df=pd.DataFrame(full_data_df, columns= ['Order ID','Order Date','Product ID','Ship Mode','Segment','Country','State','Region','Category',\
                                                  'Sub-Category', 'Sales','Quantity','Discount','Profit'])
```

```
selective_df.head(5)
```

	Order ID	Order Date	Product ID	Ship Mode	Segment	Country	State	Region	Category	Sub-Category	Sales	Quantity	Discount	Profit
0	CA-2017-152156	2017-11-08	FUR-BO-10001798	Second Class	Consumer	United States	Kentucky	South	Furniture	Bookcases	261.9600	2	0.00	41.9136
1	CA-2017-152156	2017-11-08	FUR-CH-10000454	Second Class	Consumer	United States	Kentucky	South	Furniture	Chairs	731.9400	3	0.00	219.5820
2	CA-2017-138688	2017-06-12	OFF-LA-10000240	Second Class	Corporate	United States	California	West	Office Supplies	Labels	14.6200	2	0.00	6.8714
3	US-2016-108966	2016-10-11	FUR-TA-10000577	Standard Class	Consumer	United States	Florida	South	Furniture	Tables	957.5775	5	0.45	-383.0310
4	US-2016-108966	2016-10-11	OFF-ST-10000760	Standard Class	Consumer	United States	Florida	South	Office Supplies	Storage	22.3680	2	0.20	2.5164

Reshaping data with Pandas

- **Pivot table**
- pivot_table() operates on an existing data-frame and **accepts certain parameters across which the aggregation is to be done.**

Reshaping data with Pandas

```
pivot_table_df=pd.pivot_table(selective_df,index=['Region','Segment'])
```

```
pivot_table_df
```

Region	Segment	Discount	Profit	Quantity	Sales
Central	Consumer	0.252030	7.066046	3.728548	207.946728
	Corporate	0.239822	27.791831	3.869242	234.763466
	Home Office	0.208858	28.398202	3.783105	208.248046
East	Consumer	0.147447	28.040153	3.639891	238.875539
	Corporate	0.144356	26.935666	3.828962	228.516929
	Home Office	0.141036	53.205611	3.810757	253.911805
South	Consumer	0.142124	32.116435	3.792363	233.390180
	Corporate	0.157745	29.833771	3.952941	238.992025
	Home Office	0.143382	16.987626	3.731618	272.996329
West	Consumer	0.107506	34.360409	3.873804	217.033955
	Corporate	0.113958	35.872323	3.781250	235.265911
	Home Office	0.106918	28.949939	3.781086	239.442692

Reshaping data with Pandas

- **Pivot table**
- pivot_table() operates on an existing data-frame and **accepts certain parameters across which the aggregation is to be done.**
- **Pivot_table_df.shape: (12,4)**

Reshaping data with Pandas

```
pivot_table_df.reset_index(inplace=True)
```

```
pivot_table_df
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	Central	Consumer	0.252030	7.066046	3.728548	207.946728
1	Central	Corporate	0.239822	27.791831	3.869242	234.763466
2	Central	Home Office	0.208858	28.398202	3.783105	208.248046
3	East	Consumer	0.147447	28.040153	3.639891	238.875539
4	East	Corporate	0.144356	26.935666	3.828962	228.516929
5	East	Home Office	0.141036	53.205611	3.810757	253.911805
6	South	Consumer	0.142124	32.116435	3.792363	233.390180
7	South	Corporate	0.157745	29.833771	3.952941	238.992025
8	South	Home Office	0.143382	16.987626	3.731618	272.996329
9	West	Consumer	0.107506	34.360409	3.873804	217.033955
10	West	Corporate	0.113958	35.872323	3.781250	235.265911
11	West	Home Office	0.106918	28.949939	3.781086	239.442692

```
pivot_table_df.shape
```

```
(12, 6)
```

Reshaping data with Pandas

Reshaping data with Pandas

- **Q: How to find the Region and Segment which has the highest mean profit?**

Reshaping data with Pandas

- **Q: How to find the Region and Segment which has the highest mean profit?**

```
pivot_table_df.sort_values('Profit', ascending=False).reset_index()
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	East	Home Office	0.141036	53.205611	3.810757	253.911805
1	West	Corporate	0.113958	35.872323	3.781250	235.265911
2	West	Consumer	0.107506	34.360409	3.873804	217.033955
3	South	Consumer	0.142124	32.116435	3.792363	233.390180
4	South	Corporate	0.157745	29.833771	3.952941	238.992025
5	West	Home Office	0.106918	28.949939	3.781086	239.442692
6	Central	Home Office	0.208858	28.398202	3.783105	208.248046
7	East	Consumer	0.147447	28.040153	3.639891	238.875539
8	Central	Corporate	0.239822	27.791831	3.869242	234.763466
9	East	Corporate	0.144356	26.935666	3.828962	228.516929
10	South	Home Office	0.143382	16.987626	3.731618	272.996329
11	Central	Consumer	0.252030	7.066046	3.728548	207.946728

Reshaping data with Pandas

- **Q: How to find the Region and Segment which has the highest mean profit?**
- **Aggregating data**

Reshaping data with Pandas

- **Q: How to find the Region and Segment which has the highest mean profit?**
- **Aggregating data**

```
pivot_table_df_agg=pd.pivot_table(selective_df,index=['Region','Segment'],aggfunc=['mean', 'sum'])
pivot_table_df_agg
```

		mean				sum			
		Discount	Profit	Quantity	Sales	Discount	Profit	Quantity	Sales
Region	Segment								
Central	Consumer	0.252030	7.066046	3.728548	207.946728	305.46	8564.0481	4519	252031.4340
	Corporate	0.239822	27.791831	3.869242	234.763466	161.40	18703.9020	2604	157995.8128
	Home Office	0.208858	28.398202	3.783105	208.248046	91.48	12438.4124	1657	91212.6440
East	Consumer	0.147447	28.040153	3.639891	238.875539	216.60	41190.9843	5347	350908.1670
	Corporate	0.144356	26.935666	3.828962	228.516929	126.60	23622.5789	3358	200409.3470
	Home Office	0.141036	53.205611	3.810757	253.911805	70.80	26709.2168	1913	127463.7260
South	Consumer	0.142124	32.116435	3.792363	233.390180	119.10	26913.5728	3178	195580.9710
	Corporate	0.157745	29.833771	3.952941	238.992025	80.45	15215.2232	2016	121885.9325
	Home Office	0.143382	16.987626	3.731618	272.996329	39.00	4620.6343	1015	74255.0015
West	Consumer	0.107506	34.360409	3.873804	217.033955	179.75	57450.6040	6477	362880.7730
	Corporate	0.113958	35.872323	3.781250	235.265911	109.40	34437.4299	3630	225855.2745
	Home Office	0.106918	28.949939	3.781086	239.442692	61.05	16530.4150	2159	136721.7770

Reshaping data with Pandas

- **Q: How to find the Region and Segment which has the highest mean profit?**
- **Aggregating data**

Reshaping data with Pandas

Reshaping data with Pandas

- **Melt:** pass a list as *identifier variables i.e. id_vars* and all other columns are considered as the *measure variables* which get listed under the *variable* and *value* columns

Reshaping data with Pandas

- **Melt:** pass a list as *identifier variables i.e. id_vars* and all other columns are considered as the *measure variables* which get listed under the *variable* and *value* columns

```
pivot_table_df.reset_index(inplace=True)
```

```
pivot_table_df
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	Central	Consumer	0.252030	7.066046	3.728548	207.946728
1	Central	Corporate	0.239822	27.791831	3.869242	234.763466
2	Central	Home Office	0.208858	28.398202	3.783105	208.248046
3	East	Consumer	0.147447	28.040153	3.639891	238.875539
4	East	Corporate	0.144356	26.935666	3.828962	228.516929
5	East	Home Office	0.141036	53.205611	3.810757	253.911805
6	South	Consumer	0.142124	32.116435	3.792363	233.390180
7	South	Corporate	0.157745	29.833771	3.952941	238.992025
8	South	Home Office	0.143382	16.987626	3.731618	272.996329
9	West	Consumer	0.107506	34.360409	3.873804	217.033955
10	West	Corporate	0.113958	35.872323	3.781250	235.265911
11	West	Home Office	0.106918	28.949939	3.781086	239.442692

```
pivot_table_df.shape
```

```
(12, 6)
```

Reshaping data with Pandas

- **Melt:** pass a list as *identifier variables i.e. id_vars* and all other columns are considered as the *measure variables* which get listed under the *variable* and *value* columns

```
pivot_table_df.reset_index(inplace=True)
```

```
pivot_table_df
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	Central	Consumer	0.252030	7.066046	3.728548	207.946728
1	Central	Corporate	0.239822	27.791831	3.869242	234.763466
2	Central	Home Office	0.208858	28.398202	3.783105	208.248046
3	East	Consumer	0.147447	28.040153	3.639891	238.875539
4	East	Corporate	0.144356	26.935666	3.828962	228.516929
5	East	Home Office	0.141036	53.205611	3.810757	253.911805
6	South	Consumer	0.142124	32.116435	3.792363	233.390180
7	South	Corporate	0.157745	29.833771	3.952941	238.992025
8	South	Home Office	0.143382	16.987626	3.731618	272.996329
9	West	Consumer	0.107506	34.360409	3.873804	217.033955
10	West	Corporate	0.113958	35.872323	3.781250	235.265911
11	West	Home Office	0.106918	28.949939	3.781086	239.442692

```
pivot_table_df.shape
```

```
(12, 6)
```

```
pivot_table_df.melt(id_vars=['Region','Segment'])
```

	Region	Segment	variable	value
0	Central	Consumer	Discount	0.252030
1	Central	Corporate	Discount	0.239822
2	Central	Home Office	Discount	0.208858
3	East	Consumer	Discount	0.147447
4	East	Corporate	Discount	0.144356
5	East	Home Office	Discount	0.141036
6	South	Consumer	Discount	0.142124
7	South	Corporate	Discount	0.157745
8	South	Home Office	Discount	0.143382
9	West	Consumer	Discount	0.107506
10	West	Corporate	Discount	0.113958
11	West	Home Office	Discount	0.106918
12	Central	Consumer	Profit	7.066046
13	Central	Corporate	Profit	27.791831
14	Central	Home Office	Profit	28.398202
15	East	Consumer	Profit	28.040153
16	East	Corporate	Profit	26.935666
17	East	Home Office	Profit	53.205611
18	South	Consumer	Profit	32.116435
19	South	Corporate	Profit	29.833771

Reshaping data with Pandas

Reshaping data with Pandas

- **Unstack:**

Reshaping data with Pandas

- **Unstack:**

```
pivot_table_df=pd.pivot_table(selective_df,index=['Region','Segment'])
```

```
pivot_table_df
```

		Discount	Profit	Quantity	Sales
Region	**Segment**				
Central	**Consumer**	0.252030	7.066046	3.728548	207.946728
	Corporate	0.239822	27.791831	3.869242	234.763466
	Home Office	0.208858	28.398202	3.783105	208.248046
East	**Consumer**	0.147447	28.040153	3.639891	238.875539
	Corporate	0.144356	26.935666	3.828962	228.516929
	Home Office	0.141036	53.205611	3.810757	253.911805
South	**Consumer**	0.142124	32.116435	3.792363	233.390180
	Corporate	0.157745	29.833771	3.952941	238.992025
	Home Office	0.143382	16.987626	3.731618	272.996329
West	**Consumer**	0.107506	34.360409	3.873804	217.033955
	Corporate	0.113958	35.872323	3.781250	235.265911
	Home Office	0.106918	28.949939	3.781086	239.442692

Reshaping data with Pandas

- **Unstack:**

```
pivot_table_df=pd.pivot_table(selective_df,index=['Region','Segment'])
```

```
pivot_table_df.unstack()
```

	Discount			Profit			Quantity			Sales		
Segment	**Consumer**	**Corporate**	**Home Office**	**Consumer**	**Corporate**	**Home Office**	**Consumer**	**Corporate**	**Home Office**	**Consumer**	**Corporate**	**Home Office**
Region												
Central	0.252030	0.239822	0.208858	7.066046	27.791831	28.398202	3.728548	3.869242	3.783105	207.946728	234.763466	208.248046
East	0.147447	0.144356	0.141036	28.040153	26.935666	53.205611	3.639891	3.828962	3.810757	238.875539	228.516929	253.911805
South	0.142124	0.157745	0.143382	32.116435	29.833771	16.987626	3.792363	3.952941	3.731618	233.390180	238.992025	272.996329
West	0.107506	0.113958	0.106918	34.360409	35.872323	28.949939	3.873804	3.781250	3.781086	217.033955	235.265911	239.442692

Reshaping data with Pandas

- **Stack:**

Reshaping data with Pandas

- **Stack:**

```
unstacked_df=pivot_table_df.unstack()
```

```
unstacked_df.stack()
```

Region	Segment	Discount	Profit	Quantity	Sales
Central	**Consumer**	0.252030	7.066046	3.728548	207.946728
	Corporate	0.239822	27.791831	3.869242	234.763466
	Home Office	0.208858	28.398202	3.783105	208.248046
East	**Consumer**	0.147447	28.040153	3.639891	238.875539
	Corporate	0.144356	26.935666	3.828962	228.516929
	Home Office	0.141036	53.205611	3.810757	253.911805
South	**Consumer**	0.142124	32.116435	3.792363	233.390180
	Corporate	0.157745	29.833771	3.952941	238.992025
	Home Office	0.143382	16.987626	3.731618	272.996329
West	**Consumer**	0.107506	34.360409	3.873804	217.033955
	Corporate	0.113958	35.872323	3.781250	235.265911
	Home Office	0.106918	28.949939	3.781086	239.442692

Reshaping data with Pandas

Reshaping data with Pandas

- **Pivot**: reshaping (pivoting) without aggregating

Reshaping data with Pandas

- **Pivot**: reshaping (pivoting) without aggregating
- **Ensure that our data does not have rows with duplicate values for the specified columns**

Reshaping data with Pandas

- **Pivot**: reshaping (pivoting) without aggregating
- **Ensure that our data does not have rows with duplicate values for the specified columns**

```
pivot_table_df
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	Central	Consumer	0.252030	7.066046	3.728548	207.946728
1	Central	Corporate	0.239822	27.791831	3.869242	234.763466
2	Central	Home Office	0.208858	28.398202	3.783105	208.248046
3	East	Consumer	0.147447	28.040153	3.639891	238.875539
4	East	Corporate	0.144356	26.935666	3.828962	228.516929
5	East	Home Office	0.141036	53.205611	3.810757	253.911805
6	South	Consumer	0.142124	32.116435	3.792363	233.390180
7	South	Corporate	0.157745	29.833771	3.952941	238.992025
8	South	Home Office	0.143382	16.987626	3.731618	272.996329
9	West	Consumer	0.107506	34.360409	3.873804	217.033955
10	West	Corporate	0.113958	35.872323	3.781250	235.265911
11	West	Home Office	0.106918	28.949939	3.781086	239.442692

Reshaping data with Pandas

- **Pivot**: reshaping (pivoting) without aggregating
- **Ensure that our data does not have rows with duplicate values for the specified columns**

```
pivot_table_df
```

	Region	Segment	Discount	Profit	Quantity	Sales
0	Central	Consumer	0.252030	7.066046	3.728548	207.946728
1	Central	Corporate	0.239822	27.791831	3.869242	234.763466
2	Central	Home Office	0.208858	28.398202	3.783105	208.248046
3	East	Consumer	0.147447	28.040153	3.639891	238.875539
4	East	Corporate	0.144356	26.935666	3.828962	228.516929
5	East	Home Office	0.141036	53.205611	3.810757	253.911805
6	South	Consumer	0.142124	32.116435	3.792363	233.390180
7	South	Corporate	0.157745	29.833771	3.952941	238.992025
8	South	Home Office	0.143382	16.987626	3.731618	272.996329
9	West	Consumer	0.107506	34.360409	3.873804	217.033955
10	West	Corporate	0.113958	35.872323	3.781250	235.265911
11	West	Home Office	0.106918	28.949939	3.781086	239.442692

```
Region_Segment_Sales = pivot_table_df.pivot(index="Region", columns="Segment", values="Sales")
Region_Segment_Sales
```

Segment	Consumer	Corporate	Home Office
Region			
Central	207.946728	234.763466	208.248046
East	238.875539	228.516929	253.911805
South	233.390180	238.992025	272.996329
West	217.033955	235.265911	239.442692

```
Region_Segment_Profit = pivot_table_df.pivot(index="Region", columns="Segment", values="Profit")
Region_Segment_Profit
```

Segment	Consumer	Corporate	Home Office
Region			
Central	7.066046	27.791831	28.398202
East	28.040153	26.935666	53.205611
South	32.116435	29.833771	16.987626
West	34.360409	35.872323	28.949939