Effective Data Visualization with Python

Carlos Afonso -- Data Science Instructor

June 23, 2021

NYC Data Science Academy

Objectives

Objectives

1. Learn data visualization principles
2. Learn what are the appropriate data visualizations to use for common data tasks.
3. Learn how to create those visualization with `python` (`pandas` and `matplotlib`).

Notes

- *Think before you code!*
  - Focus on data visualization principles.
  - Don't get distracted with the code.
  - You'll get this presentation after the workshop (including the code).
- I'll be asking you questions:
  - Try answering the questions by yourself.
  - If you want, you can share your answers on the Zoom chat.

Outline

Introduction

- Data Science
- Data Visualization

Common tasks with examples

- Univariate
    - Categorical
    - Numerical
- Multivariate (Bivariate)
    - Categorical vs Numerical
    - Categorical vs Categorical
    - Numerical vs Numerical

Real world data visualization project walkthrough

- Visualizing the 2019 Measles Outbreak (in NYC)

Libraries

In [1]:

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

Data

Data - Read original data

IBM HR Analytics Employee Attrition & Performance:

https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset

In [2]:

```
full_df = pd.read_csv("WA_Fn-UseC_-HR-Employee-Attrition.csv")
full_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Age                       1470 non-null   int64
 1   Attrition                 1470 non-null   objec
t
 2   BusinessTravel            1470 non-null   objec
t
 3   DailyRate                 1470 non-null   int64
 4   Department                1470 non-null   objec
t
 5   DistanceFromHome          1470 non-null   int64
 6   Education                 1470 non-null   int64
```

```
 7   EducationField              1470 non-null   objec
t
 8   EmployeeCount               1470 non-null   int64
 9   EmployeeNumber              1470 non-null   int64
 10  EnvironmentSatisfaction     1470 non-null   int64
 11  Gender                      1470 non-null   objec
t
 12  HourlyRate                  1470 non-null   int64
 13  JobInvolvement              1470 non-null   int64
 14  JobLevel                    1470 non-null   int64
 15  JobRole                     1470 non-null   objec
t
 16  JobSatisfaction             1470 non-null   int64
 17  MaritalStatus               1470 non-null   objec
t
 18  MonthlyIncome               1470 non-null   int64
 19  MonthlyRate                 1470 non-null   int64
 20  NumCompaniesWorked          1470 non-null   int64
 21  Over18                      1470 non-null   objec
t
 22  OverTime                    1470 non-null   objec
t
 23  PercentSalaryHike           1470 non-null   int64
 24  PerformanceRating           1470 non-null   int64
```

```
 25  RelationshipSatisfaction  1470 non-null   int64
 26  StandardHours             1470 non-null   int64
 27  StockOptionLevel          1470 non-null   int64
 28  TotalWorkingYears         1470 non-null   int64
 29  TrainingTimesLastYear     1470 non-null   int64
 30  WorkLifeBalance           1470 non-null   int64
 31  YearsAtCompany            1470 non-null   int64
 32  YearsInCurrentRole        1470 non-null   int64
 33  YearsSinceLastPromotion   1470 non-null   int64
 34  YearsWithCurrManager      1470 non-null   int64
dtypes: int64(26), object(9)
memory usage: 402.1+ KB
```

End of slide

Data - Select data to use

In [3]:

```
categorical_cols = ["Attrition", "Gender", "JobLevel", "JobRole"]
numerical_cols = ["Age", "MonthlyIncome", "YearsInCurrentRole"]
all_cols = sorted(categorical_cols + numerical_cols)
df = full_df[all_cols].copy()
for col in categorical_cols:
    df[col] = df[col].astype("category")
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 7 columns):
 #   Column                Non-Null Count  Dtype   
---  ------                --------------  -----   
 0   Age                   1470 non-null   int64   
 1   Attrition             1470 non-null   category
 2   Gender                1470 non-null   category
 3   JobLevel              1470 non-null   category
 4   JobRole               1470 non-null   category
 5   MonthlyIncome         1470 non-null   int64   
 6   YearsInCurrentRole    1470 non-null   int64   
dtypes: category(4), int64(3)
memory usage: 41.1 KB
```

Notes:

- Converting categorical variables to `category` type facilitates the analysis.
- There are no missing values.

In [4]:

```
df.isna().sum()
```

Out[4]:

```
Age                    0
Attrition              0
Gender                 0
JobLevel               0
JobRole                0
MonthlyIncome          0
YearsInCurrentRole     0
dtype: int64
```

End of slide

Introduction

Introduction - Data Science

Why? -- Why do Data Science? -- What is the **goal** of Data Science?

To find useful/valuable insights for the problem/question at hand.

How? -- How to do Data Science? -- What is the Data Science **process**? (In approx. 5 steps)

(1) Question $\longleftrightarrow$ (2) Data $\longleftrightarrow$ (3) Analysis $\longleftrightarrow$ (4) Deliverable $\longleftrightarrow$ (5) Action

- **(1) Question:** Identify a valuable problem / Ask a valuable question.
- **(2) Data:** Get and process the data.
- **(3) Analysis:** Analyse the data to find valuable insights (may involve modelling).
- **(4) Deliverable:** Deliver/Present valuable insights, and/or products (e.g., dashboard, model).
- **(5) Action:** Use deliverable(s) to make decisions and take action.

Notes:

- Iterative process
- *Garbage in, garbage out!*
- *One for all, all for one!*
    - *Mess one, mess all!*

End of slide

Introduction - Data Visualization

Data Science (from previous slide)

- ***Why?*** -- To find useful/valuable insights for the problem/question at hand.
- ***How?*** -- (1) Question $\longleftrightarrow$ (2) Data $\longleftrightarrow$ (3) Analysis $\longleftrightarrow$ (4) Deliverable $\longleftrightarrow$ (5) Action

Why? -- What are the **purposes** of Data Visualization (within Data Science)?

- **Exploratory:** Explore data in search of insights
  - Exploratory Data Analysis (EDA)
  - *Dashboards*
- **Explanatory:** Communicate insights
  - Clarity, Simplicity, Context, Audience

Notes:

- dev / prod
- static / dynamic / interactive

Common Tasks

Univariate - Categoric

Univariate - Categoric - Introduction

Question: How do you summarize / visualize this data?

In [5]:

```
df[["Gender"]]
```

Out[5]:

	Gender
0	Female
1	Male
2	Male
3	Female
4	Male
...	...
1465	Male
1466	Male
1467	Male
1468	Male

	Gender
1469	Male

1470 rows × 1 columns

Univariate - Categoric - Table - `describe`

Summary table created by the `describe` method (from `pandas`):

In [6]:

```
df[["Gender"]].describe()
```

Out[6]:

	Gender
count	1470
unique	2
top	Male
freq	882

Question: Is this sufficient? Can we do something more?

Univariate - Categoric - Table - Counts & Ratios

In [7]:

```
counts = df["Gender"].value_counts()
counts
```

Out[7]:

```
Male      882
Female    588
Name: Gender, dtype: int64
```

In [8]:

```
ratios = df["Gender"].value_counts(normalize=True)
ratios
```

Out[8]:

```
Male      0.6
Female    0.4
Name: Gender, dtype: float64
```

In [9]:

```
# Useful to create plots
gender_df = pd.DataFrame({"Count": counts, "Ratio": ratios}).rename_axis("Gender")
gender_df
```

Out[9]:

Gender	Count	Ratio

Gender	Count	Ratio
Male	882	0.6
Female	588	0.4

In [10]:

```
df[["Gender"]].describe()
```

Out[10]:

	Gender
count	1470
unique	2
top	Male
freq	882

End of slide

Univariate - Categoric - Plot - Bar

In [11]:

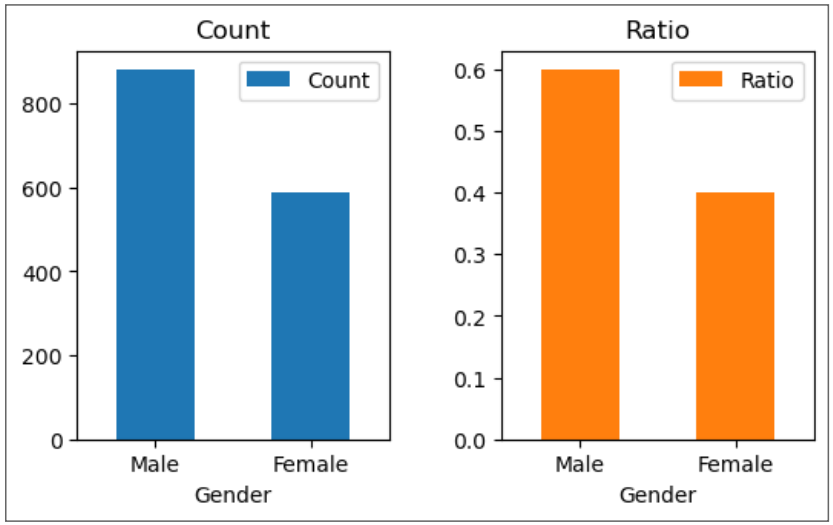
```
gender_df
```

Out[11]:

Gender	Count	Ratio
Male	882	0.6
Female	588	0.4

In [12]:

```
bar_fig, ax = plt.subplots(nrows=1, ncols=2, dpi=100)
gender_df.plot.bar(ax=ax, subplots=True, rot=0)
plt.tight_layout(pad=3)
plt.show()
```



End of slide

Univariate - Categoric - Plot - Bar - Stacked

In [13]:

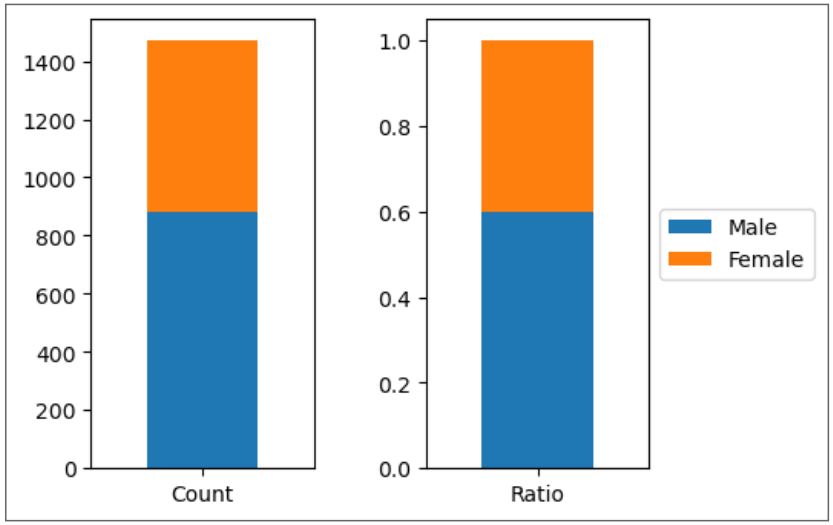
```
gender_df
```

Out[13]:

Gender	Count	Ratio
Male	882	0.6
Female	588	0.4

In [14]:

```
stacked_bar_fig, (ax1, ax2) = plt.subplots(nrows=1, ncols=2, dpi=100)
gender_df[["Count"]].transpose().plot.bar(ax=ax1, stacked=True, rot=0, legend=False)
gender_df[["Ratio"]].transpose().plot.bar(ax=ax2, stacked=True, rot=0)
plt.legend(loc='center left', bbox_to_anchor=(1.0, 0.5))
plt.tight_layout(pad=3)
plt.show()
```

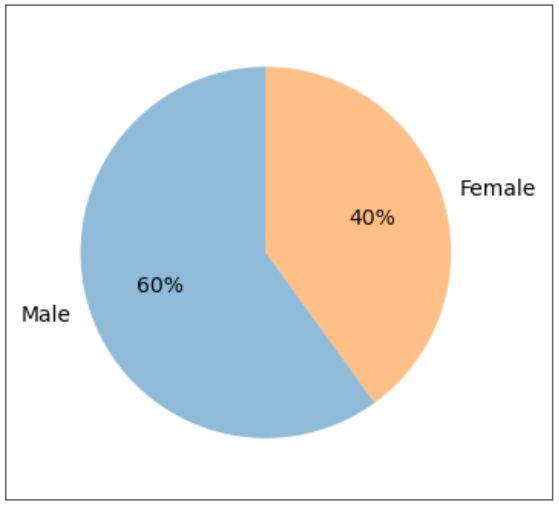

1400
1200
1000
800
600
400
200
0
Count
1.0
0.8
0.6
0.4
0.2
0.0
Ratio
Male
Female

End of slide

Univariate - Categoric - Plot - Pie

In [15]:

```
pie_fig, ax = plt.subplots(dpi=100)
gender_df.plot.pie(ax=ax, y="Count", autopct="%1.0f%%", legend=False, startangle=90, wedgeprops={'alpha':0.5})
plt.ylabel("")
plt.show()
```



Univariate - Categoric - Summary

In [16]:

```
df[["Gender"]].describe()
```

Out[16]:

	Gender
count	1470
unique	2
top	Male
freq	882

In [17]:

```
gender_df
```

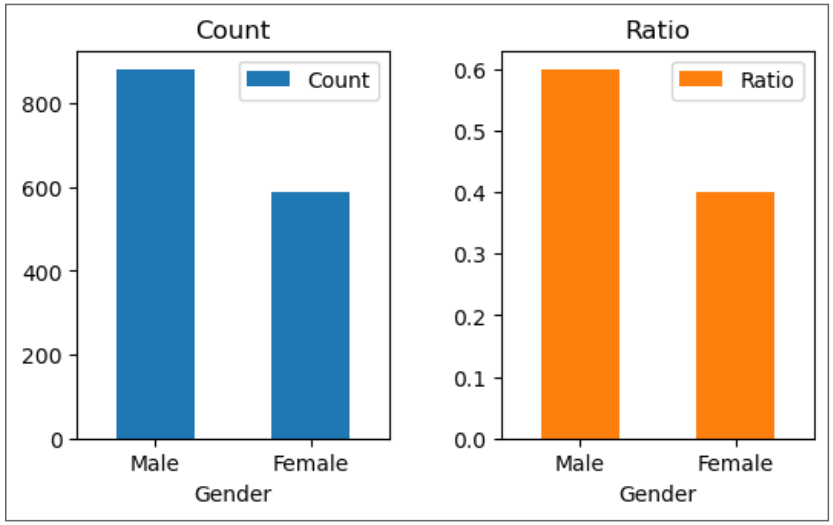
Out[17]:

Gender	Count	Ratio
Male	882	0.6
Female	588	0.4

In [18]:

```
bar_fig
```

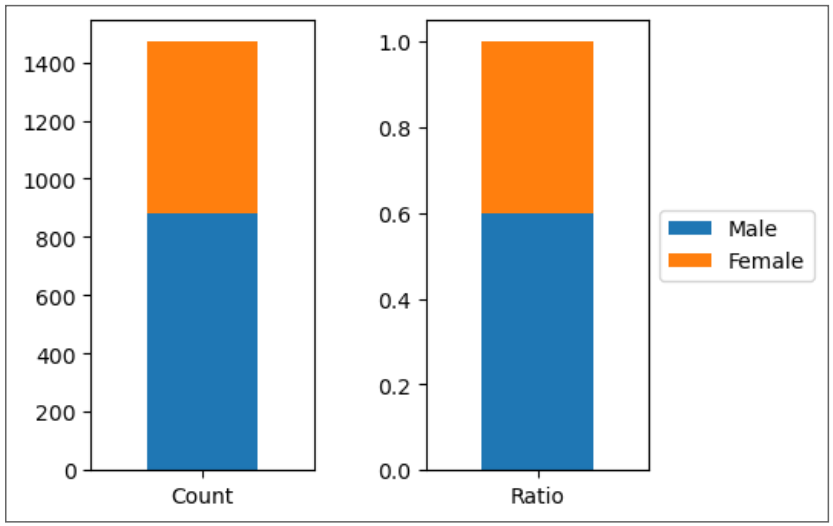
Out[18]:



In [19]:

```
stacked_bar_fig
```

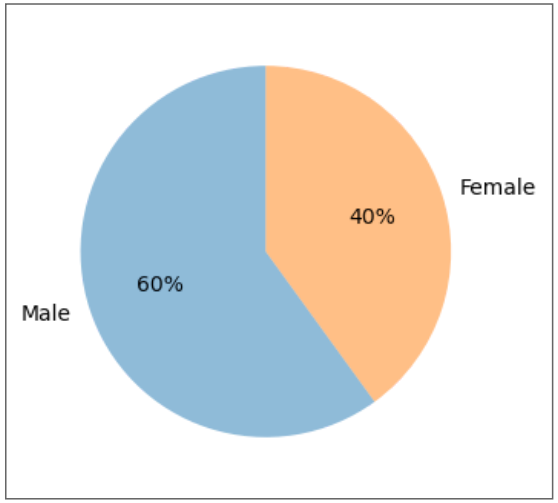
Out[19]:



In [20]:

```
pie_fig
```

Out[20]:



End of slide

Univariate - Numeric

Univariate - Numeric - Introduction

Question: How do you visualize / summarize this data?

In [21]:

```
df[["Age"]]
```

Out[21]:

	Age
0	41
1	49
2	37
3	33
4	27
...	...
1465	36
1466	39
1467	27
1468	49

	Age
1469	34

1470 rows × 1 columns

Univariate - Numeric - Table

Summary table created by the `describe` method (from `pandas`):

In [22]:

```
df[["Age"]].describe()
```

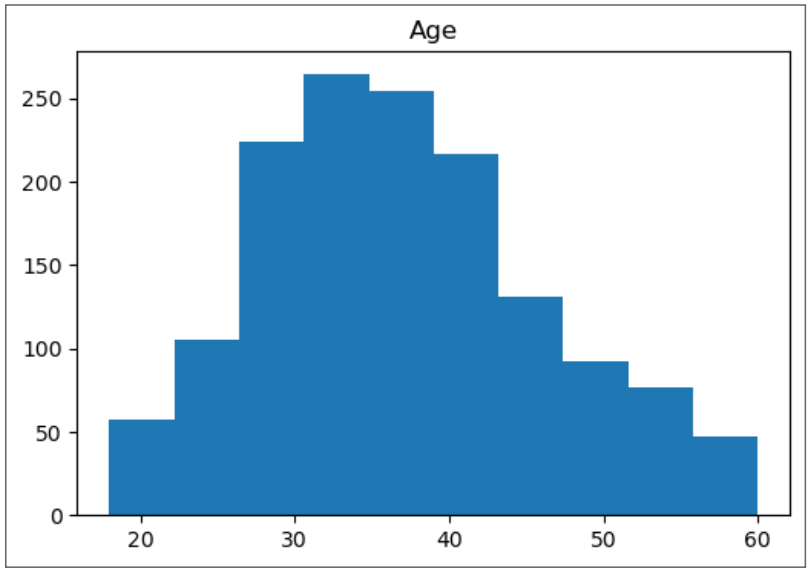
Out[22]:

	Age
count	1470.000000
mean	36.923810
std	9.135373
min	18.000000
25%	30.000000
50%	36.000000
75%	43.000000
max	60.000000

Univariate - Numeric - Plot - Histogram

In [23]:

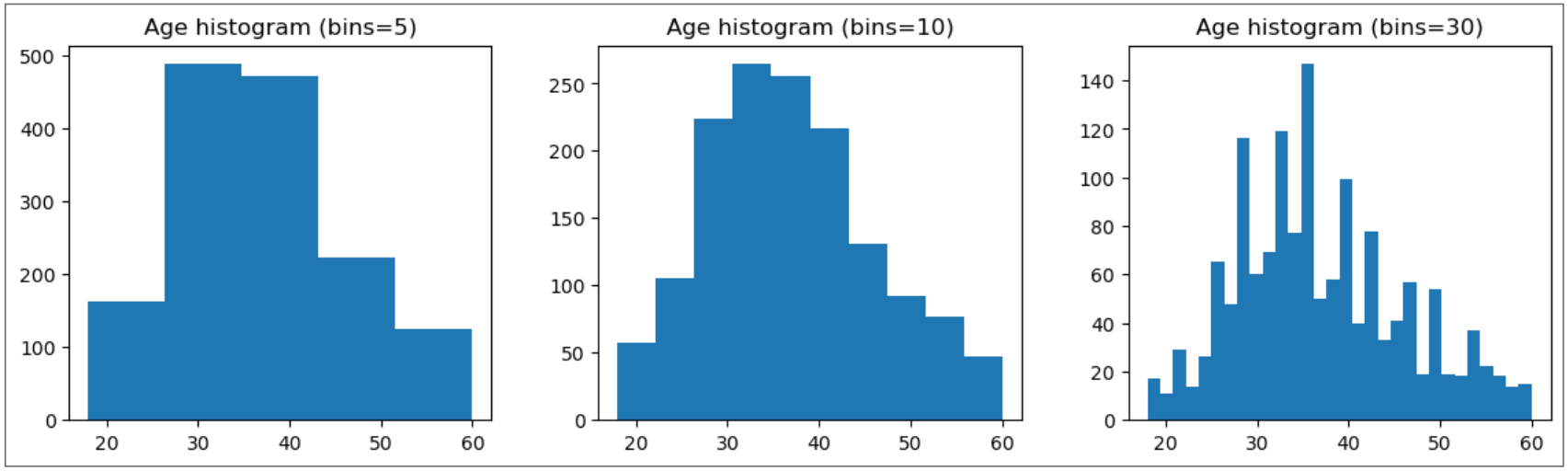
```
hist_fig, ax = plt.subplots(dpi=100)
df[["Age"]].hist(ax=ax, grid=False)
plt.show()
```



Note: Histogram plot depends on the number of bins parameter (`bins`) used:

In [24]:

```
hist3_fig, axs = plt.subplots(nrows=1, ncols=3, dpi=100, figsize=(12, 4))
for (ax, bins) in zip(axs, [5, 10, 30]): # 10 is the default
    df[["Age"]].hist(ax=ax, bins=bins, grid=False)
    ax.set_title("Age histogram (bins=" + str(bins) + ")")
plt.tight_layout(pad=3)
plt.show()
```

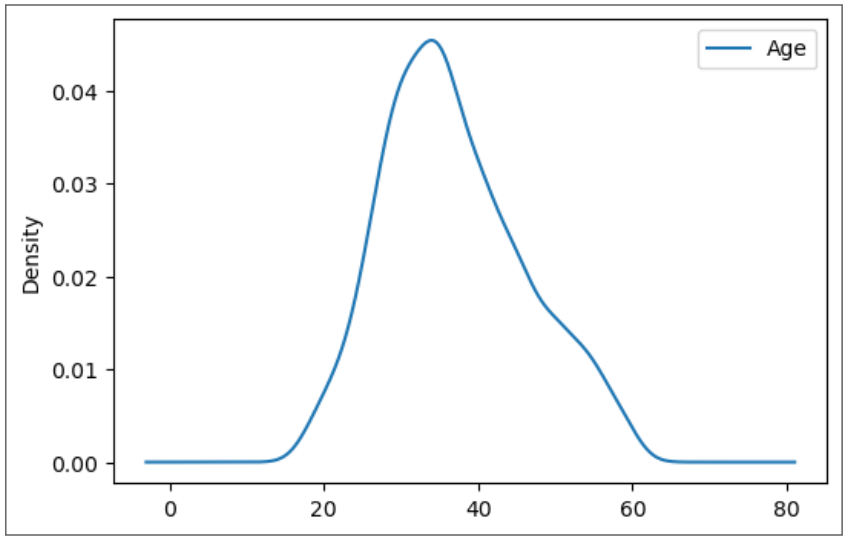

Age histogram (bins=5)
Age histogram (bins=10)
Age histogram (bins=30)

End of slide

Univariate - Numeric - Plot - Density

In [25]:

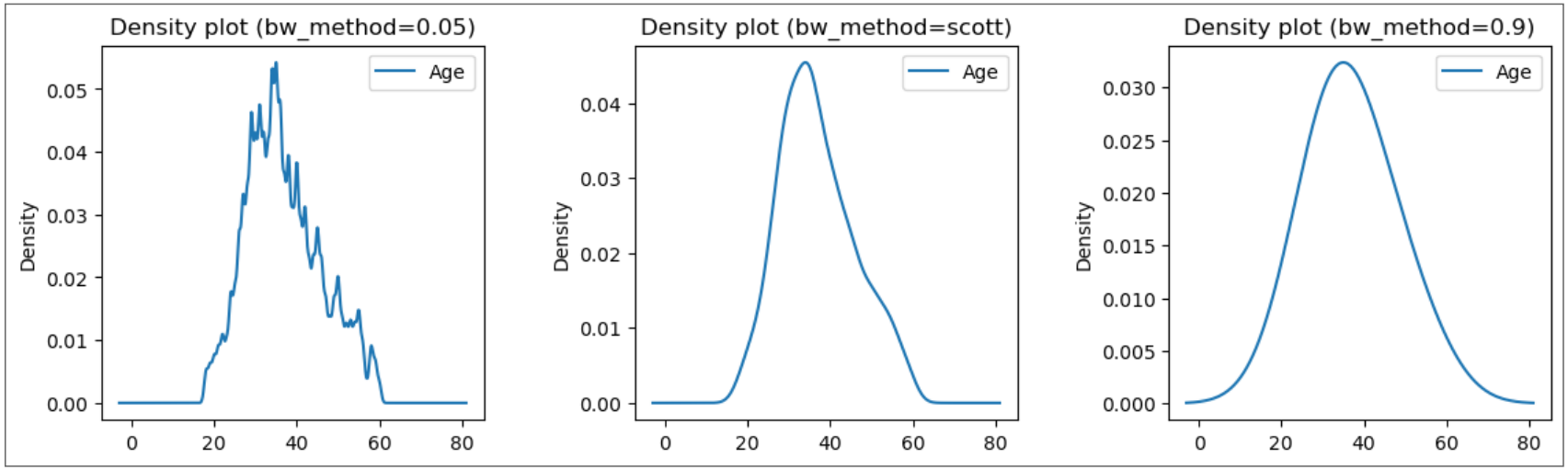
```
density_fig, ax = plt.subplots(dpi=100)
df[["Age"]].plot.density(ax=ax)
plt.show()
```



Note: Density plot depends on the bandwith parameter (`bw_method`) used:

In [26]:

```
density3_fig, axs = plt.subplots(nrows=1, ncols=3, dpi=100, figsize=(12, 4))
for (ax, bwm) in zip(axs, [0.05, "scott", 0.9]): # "scott" is the default
    df[["Age"]].plot.density(ax=ax, bw_method=bwm)
    ax.set_title("Density plot (bw_method=" + str(bwm) + ")")
plt.tight_layout(pad=3)
plt.show()
```

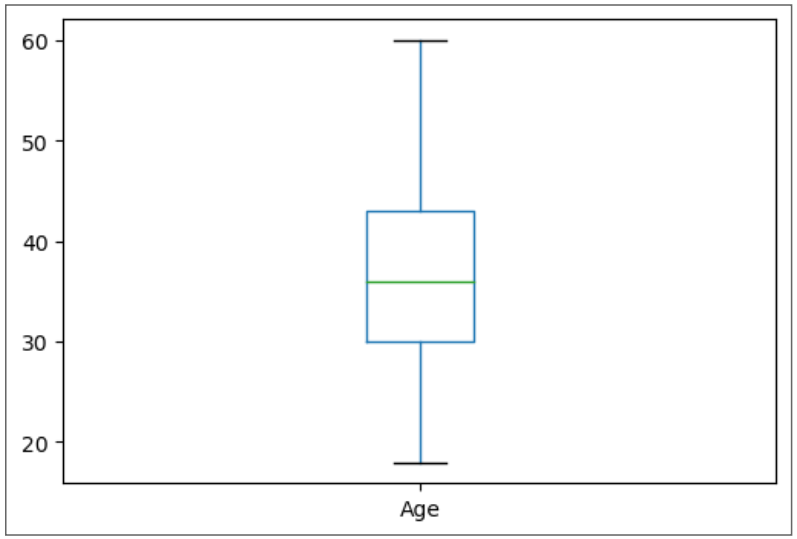

Density plot (bw_method=0.05)
Age
Density
0.05
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
Density plot (bw_method=scott)
Age
Density
0.04
0.03
0.02
0.01
0.00
0
20
40
60
80
Density plot (bw_method=0.9)
Age
Density
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80

End of slide

Univariate - Numeric - Plot - Box

In [27]:

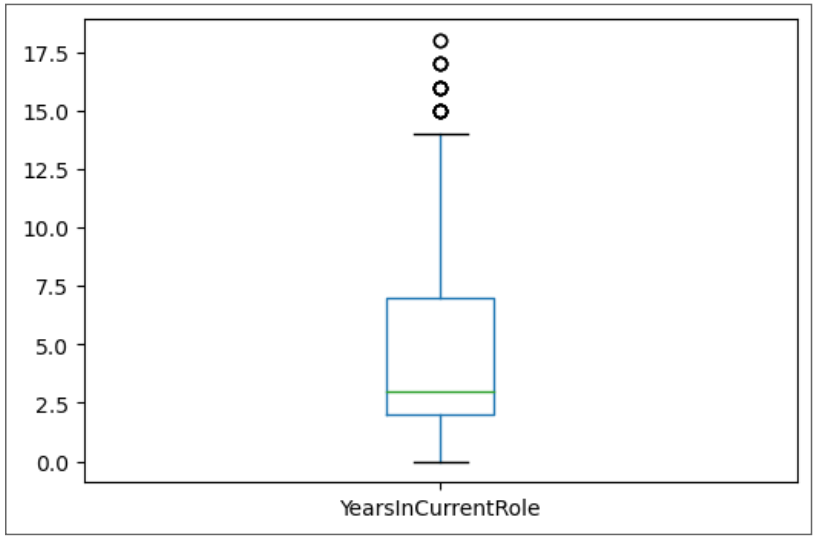
```
boxplot_fig, ax = plt.subplots(dpi=100)
df[["Age"]].boxplot(ax=ax, grid=False)
plt.show()
```



Example with Outliers

In [28]:

```
boxplot_fig_2, ax = plt.subplots(dpi=100)
df[["YearsInCurrentRole"]].boxplot(ax=ax, grid=False)
plt.show()
```



Box plot components

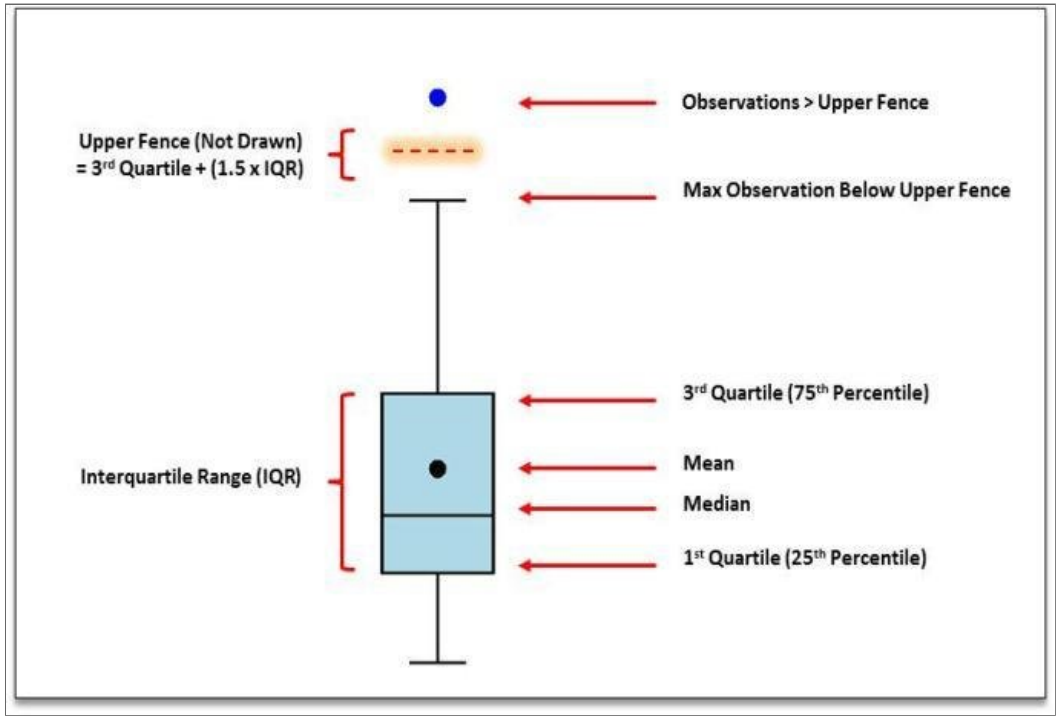


Image Source: **Basnet, Nabin & Hauck, Larry (2013)**

End of slide

Univariate - Numeric - Summary

In [29]:

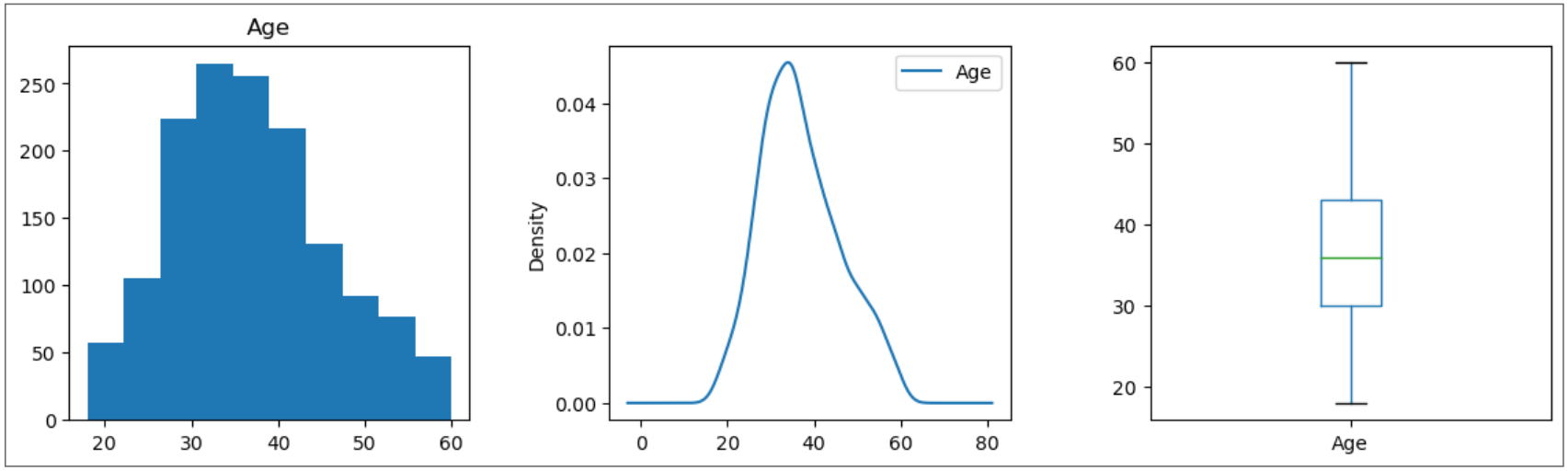
```
df[["Age"]].describe()
```

Out[29]:

	Age
count	1470.000000
mean	36.923810
std	9.135373
min	18.000000
25%	30.000000
50%	36.000000
75%	43.000000
max	60.000000

In [30]:

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, dpi=100, figsize=(12, 4))
df[["Age"]].hist(ax=ax1, grid=False)
df[["Age"]].plot.density(ax=ax2)
df[["Age"]].boxplot(ax=ax3, grid=False)
plt.tight_layout(pad=3)
plt.show()
```


Age
250
200
150
100
50
0
20
30
40
50
60
Age
0.04
0.03
0.02
0.01
0.00
Density
0
20
40
60
80
60
50
40
30
20
Age

End of slide

Multivariate - Categoric vs Numeric

Multivariate - Categoric vs Numeric - Introduction

Question: How do you summarize / visualize the (potential) relation between "Job Level" (categorical) and "Monthly Income" (numerical)?

In [31]:

```
df[["JobLevel", "MonthlyIncome"]]
```

Out[31]:

	JobLevel	MonthlyIncome
0	2	5993
1	2	5130
2	1	2090
3	1	2909
4	1	3468
...	...	...
1465	2	2571
1466	3	9991
1467	2	6142

	JobLevel	MonthlyIncome
1468	2	5390
1469	2	4404

1470 rows × 2 columns

Multivariate - Categoric vs Numeric - Table

In [32]:

```
df.groupby("JobLevel")[["MonthlyIncome"]].describe()
```

Out[32]:

JobLevel	count	mean	std	min	25%	50%	75%	MonthlyIncom ma
1	543.0	2786.915285	748.634767	1009.0	2306.0	2670.0	3207.00	4968.0
2	534.0	5502.277154	1410.029686	2042.0	4544.0	5340.0	6273.50	9998.0
3	218.0	9817.252294	1805.999233	5210.0	8383.0	9980.0	10814.50	13757.0
4	106.0	15503.783019	1816.239003	11103.0	13761.0	16154.0	17036.25	17924.0
5	69.0	19191.826087	512.383127	18041.0	18880.0	19232.0	19586.00	19999.0

In [33]:

```
df[["JobLevel", "MonthlyIncome"]].describe(include="all")
```

Out[33]:

	JobLevel	MonthlyIncome
count	1470.0	1470.000000
unique	5.0	NaN
top	1.0	NaN

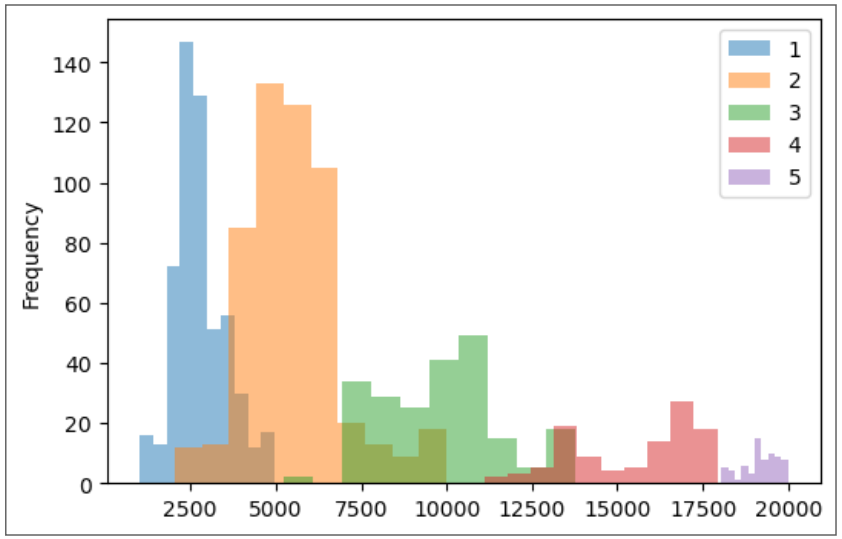
	JobLevel	MonthlyIncome
freq	543.0	NaN
mean	NaN	6502.931293
std	NaN	4707.956783
min	NaN	1009.000000
25%	NaN	2911.000000
50%	NaN	4919.000000
75%	NaN	8379.000000
max	NaN	19999.000000

End of slide

Multivariate - Categoric vs Numeric - Plot - Histogram

In [34]:

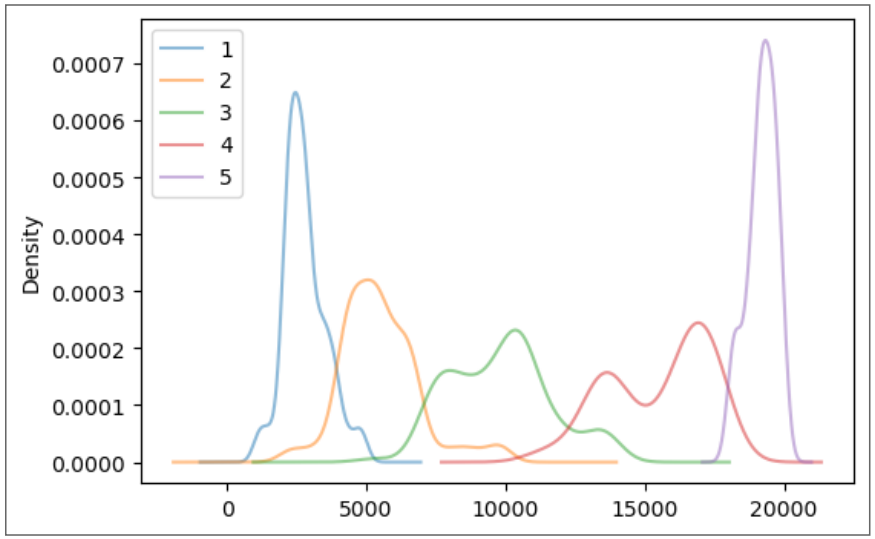
```
fig, ax = plt.subplots(dpi=100)
df.groupby("JobLevel")["MonthlyIncome"].plot.hist(ax=ax, alpha=0.5, legend=True)
plt.show()
```



Multivariate - Categoric vs Numeric - Plot - Density

In [35]:

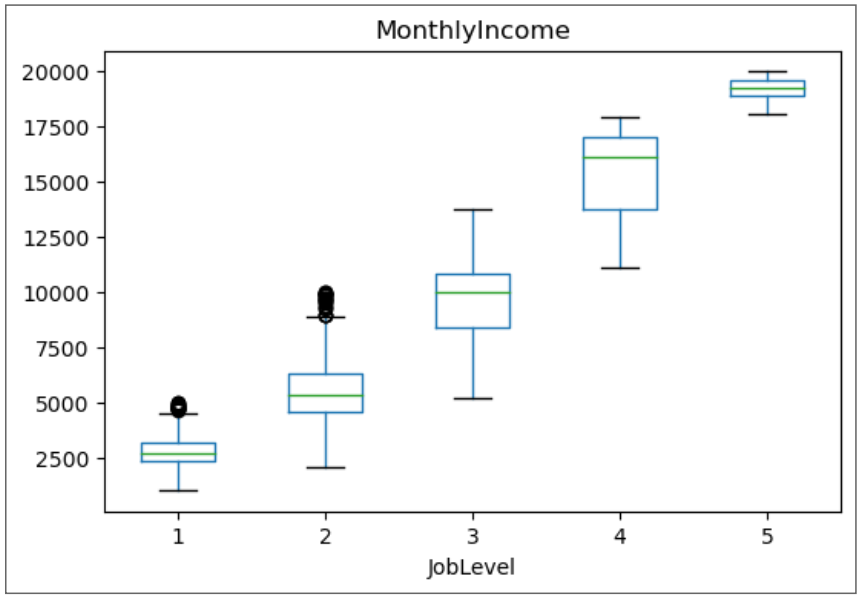
```
fig, ax = plt.subplots(dpi=100)
df.groupby("JobLevel")["MonthlyIncome"].plot.density(ax=ax, alpha=0.5, legend=True)
plt.show()
```



Multivariate - Categoric vs Numeric - Plot - Box

In [36]:

```
boxplot_fig_2, ax = plt.subplots(dpi=100)
df.boxplot(ax=ax, column="MonthlyIncome", by="JobLevel", grid=False)
plt.suptitle("")
plt.show()
```



Multivariate - Categoric vs Numeric - Summary

In [37]:

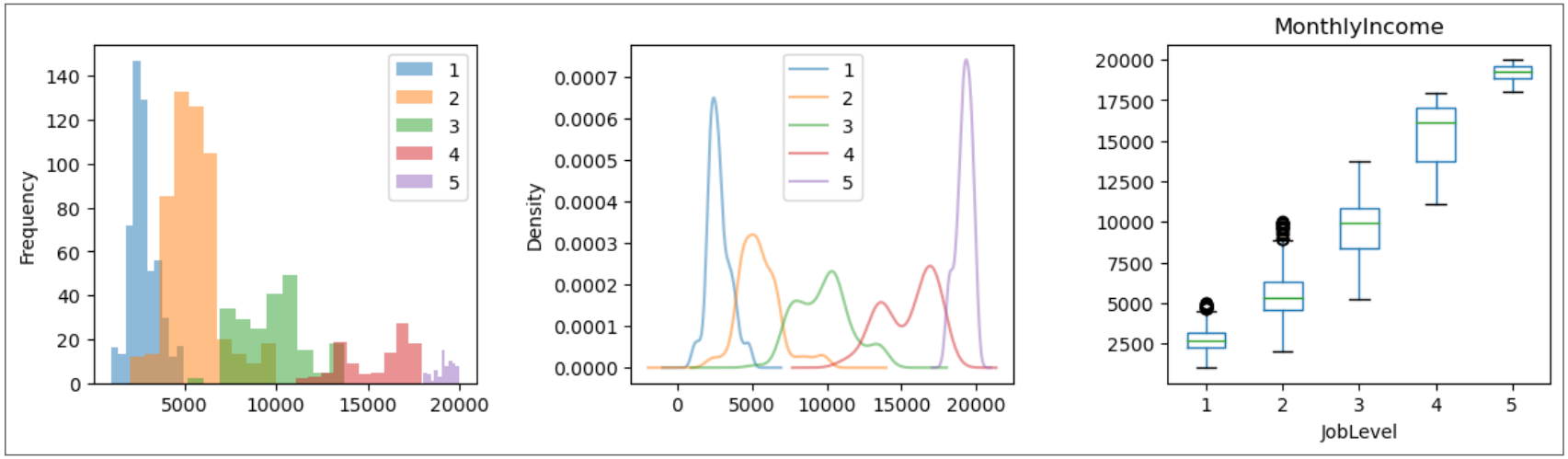
```
df.groupby("JobLevel")[["MonthlyIncome"]].describe()
```

Out[37]:

	MonthlyIncom							
	count	mean	std	min	25%	50%	75%	ma
JobLevel								
1	543.0	2786.915285	748.634767	1009.0	2306.0	2670.0	3207.00	4968.0
2	534.0	5502.277154	1410.029686	2042.0	4544.0	5340.0	6273.50	9998.0
3	218.0	9817.252294	1805.999233	5210.0	8383.0	9980.0	10814.50	13757.0
4	106.0	15503.783019	1816.239003	11103.0	13761.0	16154.0	17036.25	17924.0
5	69.0	19191.826087	512.383127	18041.0	18880.0	19232.0	19586.00	19999.0

In [38]:

```
fig, (ax1, ax2, ax3) = plt.subplots(nrows=1, ncols=3, dpi=100, figsize=(12, 4))
df.groupby("JobLevel")["MonthlyIncome"].plot.hist(ax=ax1, alpha=0.5, legend=True)
df.groupby("JobLevel")["MonthlyIncome"].plot.density(ax=ax2, alpha=0.5, legend=True)
df.boxplot(ax=ax3, column="MonthlyIncome", by="JobLevel", grid=False)
plt.suptitle("")
plt.tight_layout(pad=2)
plt.show()
```



End of slide

Multivariate - Categoric vs Categoric

Multivariate - Categoric vs Categoric - Introduction

Question: How do you summarize / visualize the (potential) relation between "Attrition" and "Gender"?

In [39]:

```
df[["Attrition", "Gender"]]
```

Out[39]:

	Attrition	Gender
0	Yes	Female
1	No	Male
2	Yes	Male
3	No	Female
4	No	Male
...	...	...
1465	No	Male
1466	No	Male
1467	No	Male

	Attrition	Gender
1468	No	Male
1469	No	Male

1470 rows × 2 columns

Multivariate - Categoric vs Categoric - Table

Contingency table (counts):

In [40]:

```
pd.crosstab(df["Attrition"], df["Gender"], margins=True)
```

Out[40]:

Attrition \ Gender	Female	Male	All
No	501	732	1233
Yes	87	150	237
All	588	882	1470

Contingency table (ratios):

In [41]:

```
pd.crosstab(df["Attrition"], df["Gender"], margins=True, normalize="columns")
```

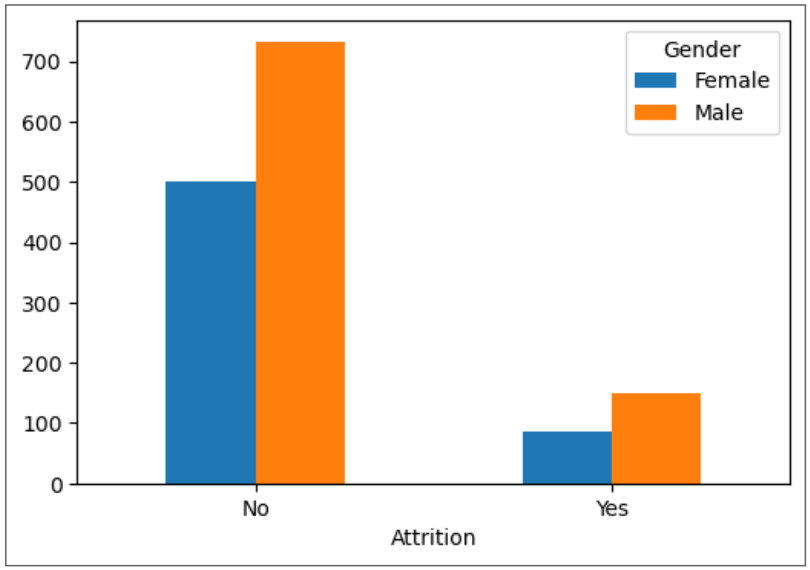
Out[41]:

Attrition \ Gender	Female	Male	All
No	0.852041	0.829932	0.838776
Yes	0.147959	0.170068	0.161224

Multivariate - Categoric vs Categoric - Plot - Bar - Paired

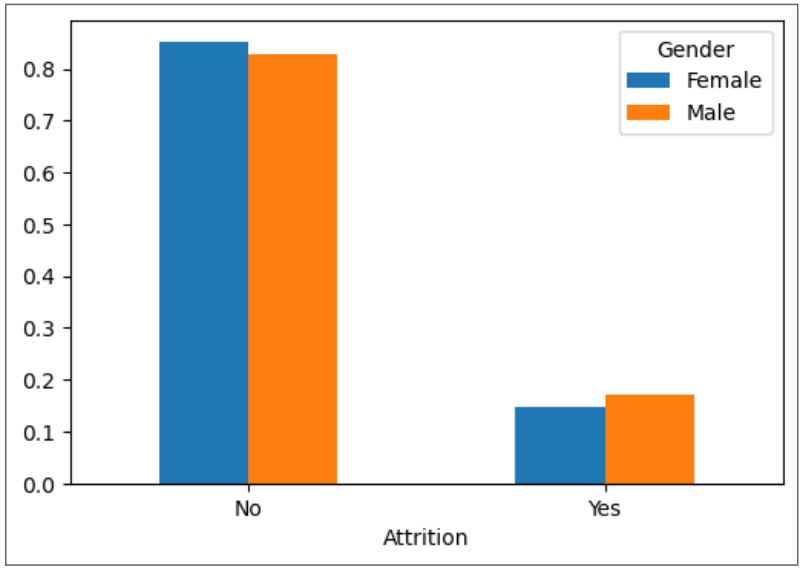
In [42]:

```
fig, ax = plt.subplots(dpi=100)
pd.crosstab(df["Attrition"], df["Gender"]).plot.bar(ax=ax, rot=0)
plt.show()
```



In [43]:

```
fig, ax = plt.subplots(dpi=100)
pd.crosstab(df["Attrition"], df["Gender"], normalize="columns").plot.bar(ax=ax, rot=0)
plt.show()
```

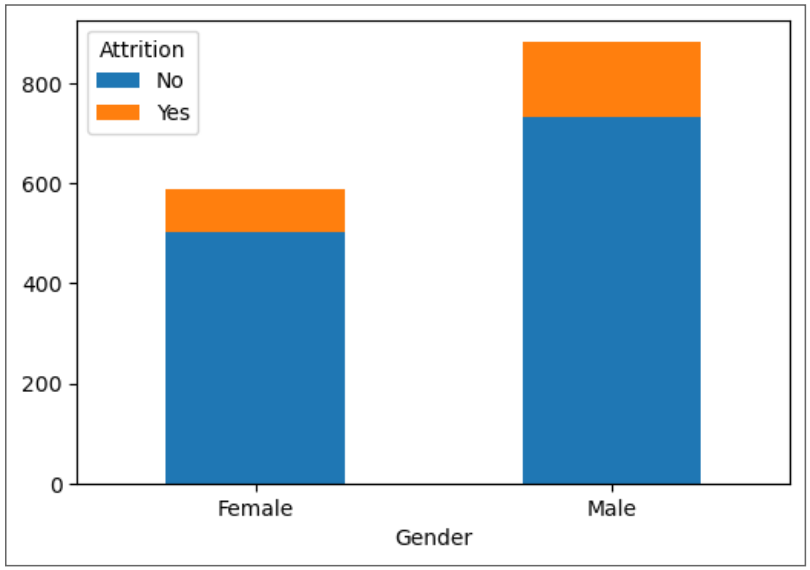


End of slide

Multivariate - Categoric vs Categoric - Plot - Bar - Stacked

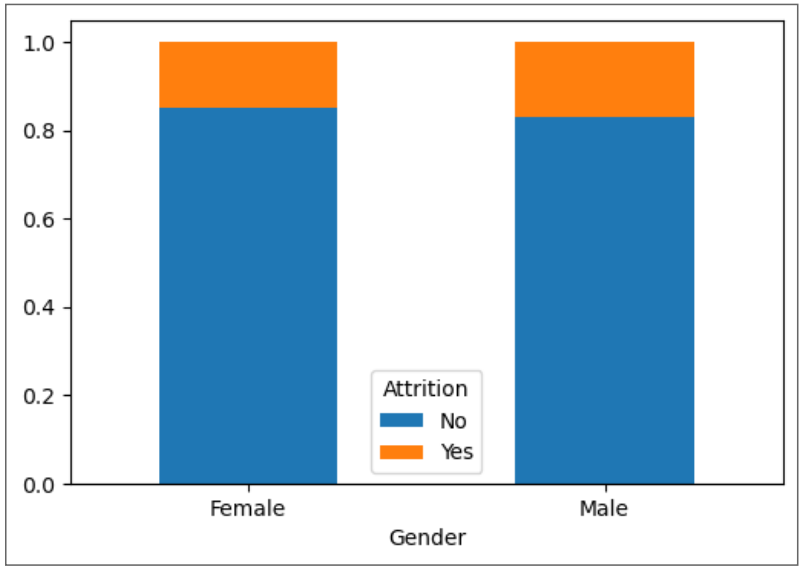
In [44]:

```
fig, ax = plt.subplots(dpi=100)
pd.crosstab(df["Gender"], df["Attrition"]).plot.bar(ax=ax, stacked=True, rot=0)
plt.show()
```



In [45]:

```
fig, ax = plt.subplots(dpi=100)
pd.crosstab(df["Gender"], df["Attrition"], normalize="index").plot.bar(ax=ax, stacked=True, rot=0)
plt.show()
```



End of slide

Multivariate - Numeric vs Numeric

Multivariate - Numeric vs Numeric - Introduction

Question: How do you summarize / visualize the (potential) relation between "Age" and "Monthly Income"?

In [46]:

```
df[["Age", "MonthlyIncome"]]
```

Out[46]:

	Age	MonthlyIncome
0	41	5993
1	49	5130
2	37	2090
3	33	2909
4	27	3468
...	...	...
1465	36	2571
1466	39	9991
1467	27	6142

	Age	MonthlyIncome
1468	49	5390
1469	34	4404

1470 rows × 2 columns

Multivariate - Numeric vs Numeric - Table

In [47]:

```
df[["Age", "MonthlyIncome"]].corr()
```

Out[47]:

	Age	MonthlyIncome
Age	1.000000	0.497855
MonthlyIncome	0.497855	1.000000

In [48]:

```
df[["Age", "MonthlyIncome"]].describe()
```

Out[48]:

	Age	MonthlyIncome
count	1470.000000	1470.000000
mean	36.923810	6502.931293
std	9.135373	4707.956783
min	18.000000	1009.000000
25%	30.000000	2911.000000
50%	36.000000	4919.000000

	Age	MonthlyIncome
75%	43.000000	8379.000000
max	60.000000	19999.000000

Notes:

- Should still use `describe` to understand each individual column
- These methods (`corr` and `describe`) work for all numeric columns in the dataframe

In [49]:

```
df.corr()
```

Out[49]:

	Age	MonthlyIncome	YearsInCurrentRole
Age	1.000000	0.497855	0.212901
MonthlyIncome	0.497855	1.000000	0.363818
YearsInCurrentRole	0.212901	0.363818	1.000000

In [50]:

```
df.describe()
```

Out[50]:

	Age	MonthlyIncome	YearsInCurrentRole
count	1470.000000	1470.000000	1470.000000
mean	36.923810	6502.931293	4.229252

	Age	MonthlyIncome	YearsInCurrentRole
std	9.135373	4707.956783	3.623137
min	18.000000	1009.000000	0.000000
25%	30.000000	2911.000000	2.000000
50%	36.000000	4919.000000	3.000000
75%	43.000000	8379.000000	7.000000
max	60.000000	19999.000000	18.000000

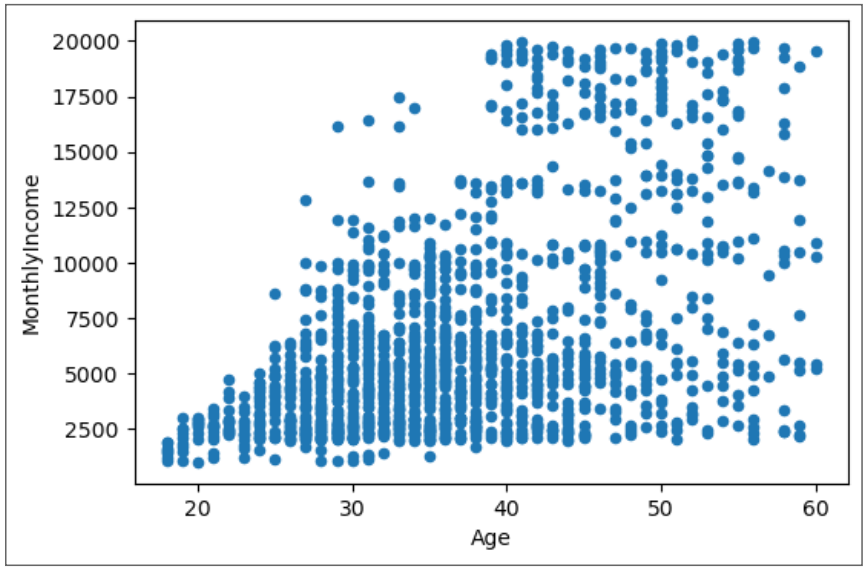
End of slide

Multivariate - Numeric vs Numeric - Plot - Scatter

Question: What do you see here? How do you interpret this plot?

In [51]:

```
fig, ax = plt.subplots(dpi=100)
df.plot.scatter(ax=ax, x="Age", y="MonthlyIncome")
plt.show()
```



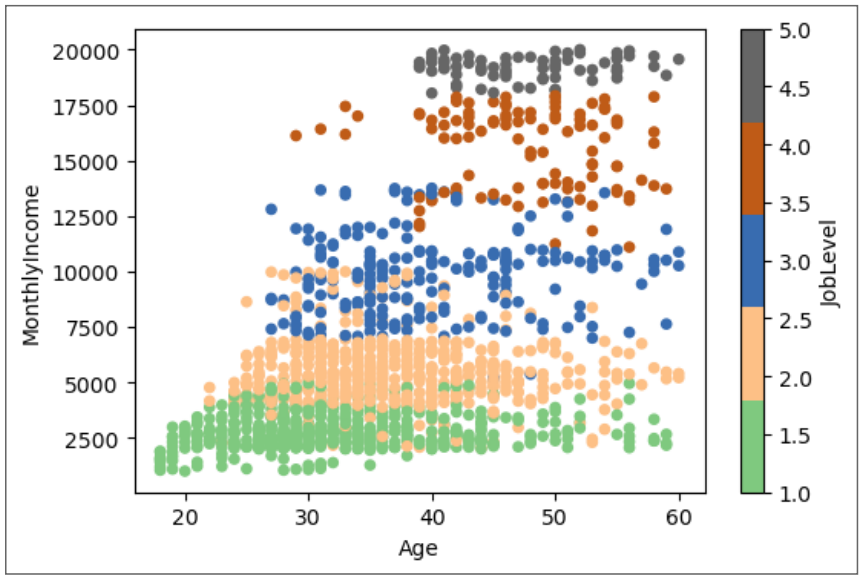
Extra

Questions:

1. How many variables are being plotted here?
2. What do you see here? How do you interpret this plot?

In [52]:

```
fig, ax = plt.subplots(dpi=100)
df.plot.scatter(ax=ax, x="Age", y="MonthlyIncome", c="JobLevel", cmap=plt.get_cmap("Accent", 5))
plt.show()
```



End of slide

Summary

Number of Variables	Variable Type(s)	Tables	Plots
Univariate (1)	Categorical	`describe`, Counts, Ratios	Bar, Stacked Bar, Pie
Univariate (1)	Numerical	`describe`	Histogram, Density, Box
Multivariate (2+)	Categorical vs Numerical	`groupby.describe`	Histogram, Density, Box
Multivariate (2+)	Categorical vs Categorical	Contingency (Counts & Ratios)	Paired Bar, Stacked Bar
Multivariate (2+)	Numerical vs Numerical	Correlation (`corr`)	Scatter

Real world data visualization project walkthrough

Visualizing the 2019 Measles Outbreak (in NYC):

- Presentation (GitHub Pages): **https://carlos-afonso.github.io/measles/**
- Repository (GitHub): **https://github.com/carlos-afonso/measles**

Data Visualization Principles:

- Clarity
- Simplicity
- Context
- Audience

The End