Addressing missing data

EXPLORATORY DATA ANALYSIS IN PYTHON

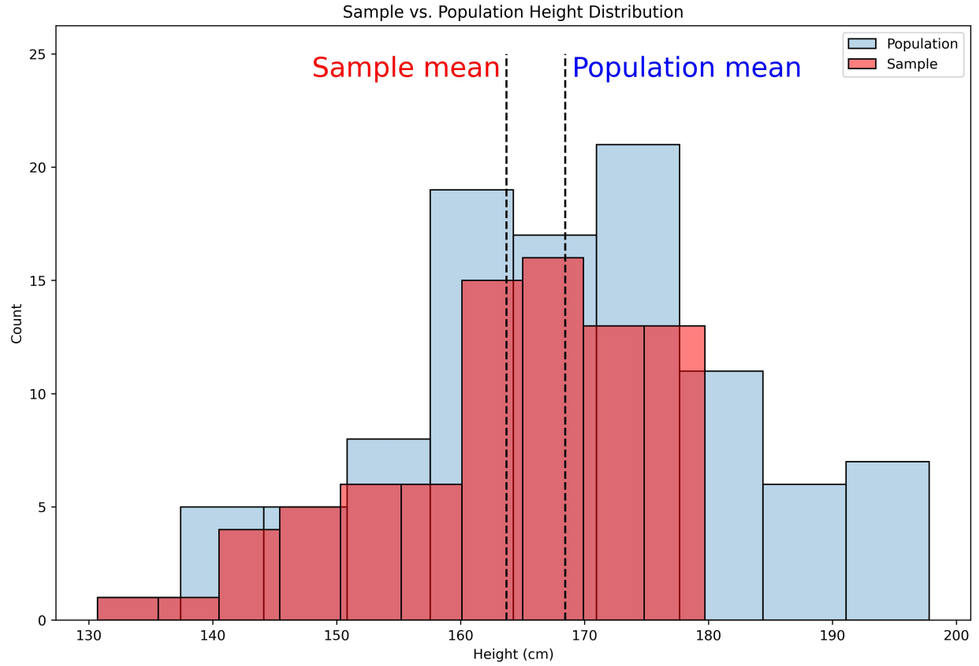


George Boorman
Curriculum Manager, DataCamp

datacamp

Why is missing data a problem?

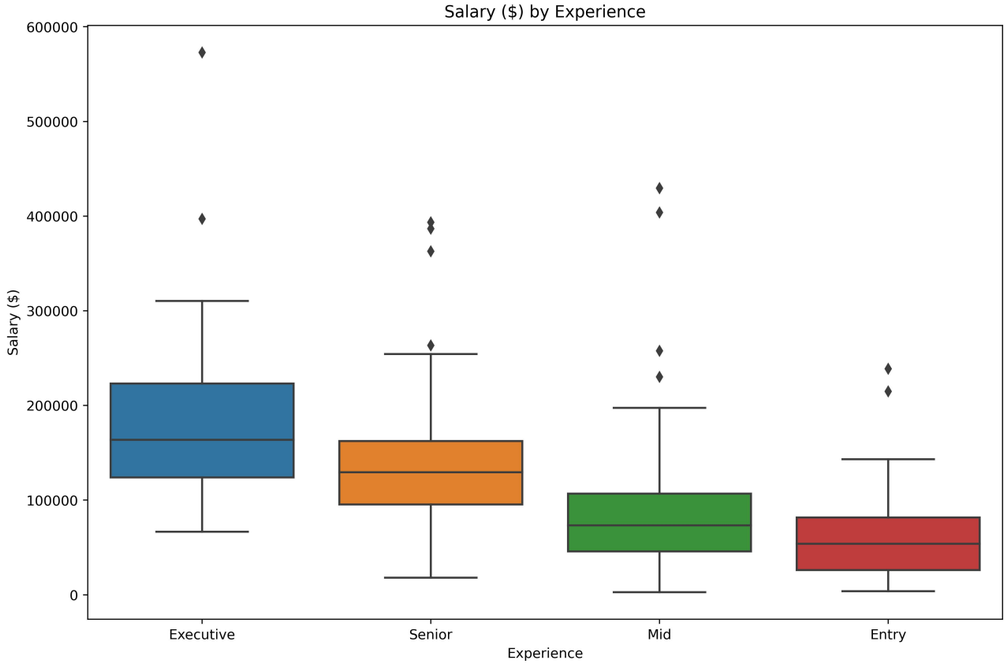
- Affects distributions
  - Missing heights of taller students
- Less representative of the population
  - Certain groups disproportionately represented, e.g., lacking data on oldest students
- Can result in drawing incorrect conclusions

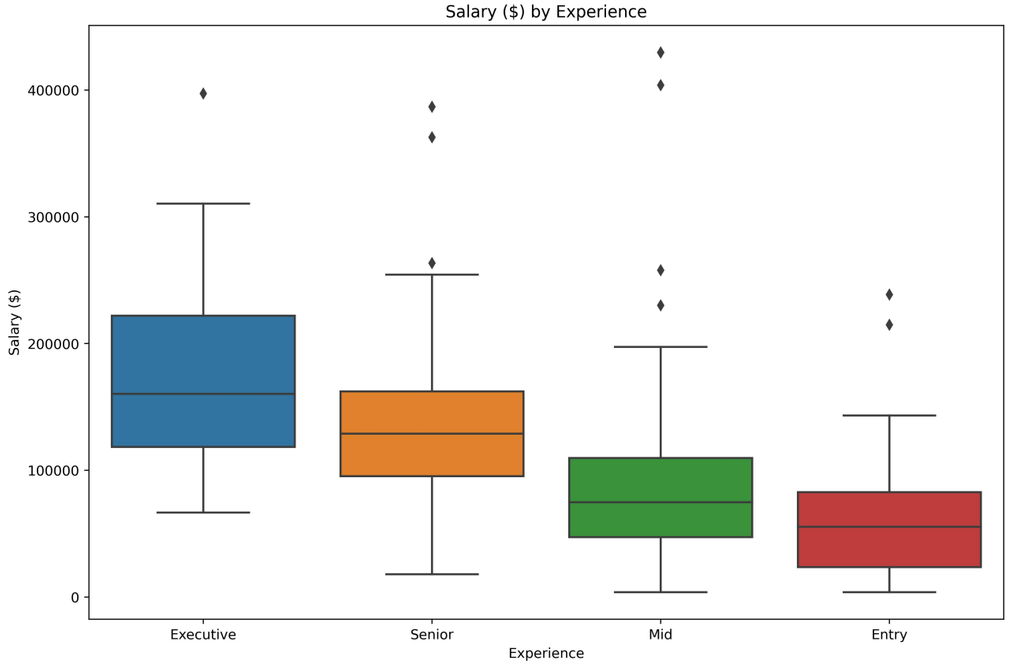


Data professionals' job data

Column	Description	Data type
`Working_Year`	Year the data was obtained	Float
`Designation`	Job title	String
`Experience`	Experience level e.g., `"Mid"`, `"Senior"`	String
`Employment_Status`	Type of employment contract e.g., `"FT"`, `"PT"`	String
`Employee_Location`	Country of employment	String
`Company_Size`	Labels for company size e.g., `"S"`, `"M"`, `"L"`	String
`Remote_Working_Ratio`	Percentage of time working remotely	Integer
`Salary_USD`	Salary in US dollars	Float

Salary by experience level





Checking for missing values

```
print(salaries.isna().sum())
```

```
Working_Year              12
Designation               27
Experience                33
Employment_Status         31
Employee_Location         28
Company_Size              40
Remote_Working_Ratio      24
Salary_USD                60
dtype: int64
```

Strategies for addressing missing data

- Drop missing values
  - 5% or less of total values
- Impute mean, median, mode
  - Depends on distribution and context
- Impute by sub-group
  - Different experience levels have different median salary

Dropping missing values

```
threshold = len(salaries) * 0.05
print(threshold)
```

```
30
```

Dropping missing values

```
cols_to_drop = salaries.columns[salaries.isna().sum() <= threshold]
print(cols_to_drop)
```

```
Index(['Working_Year', 'Designation', 'Employee_Location',
       'Remote_Working_Ratio'],
      dtype='object')
```

```
salaries.dropna(subset=cols_to_drop, inplace=True)
```

Imputing a summary statistic

```
cols_with_missing_values = salaries.columns[salaries.isna().sum() > 0]
print(cols_with_missing_values)
```

```
Index(['Experience', 'Employment_Status', 'Company_Size', 'Salary_USD'],
    dtype='object')
```

```
for col in cols_with_missing_values[:-1]:
    salaries[col].fillna(salaries[col].mode()[0])
```

Checking the remaining missing values

```
print(salaries.isna().sum())
```

```
Working_Year              0
Designation               0
Experience                0
Employment_Status         0
Employee_Location         0
Company_Size              0
Remote_Working_Ratio      0
Salary_USD               41
```

Imputing by sub-group

```
salaries_dict = salaries.groupby("Experience")["Salary_USD"].median().to_dict()
print(salaries_dict)
```

```
{'Entry': 55380.0, 'Executive': 135439.0, 'Mid': 74173.5, 'Senior': 128903.0}
```

Imputing by sub-group

```
salaries["Salary_USD"] = salaries["Salary_USD"].fillna(salaries["Experience"].map(salaries_dict))
```

No more missing values!

```
print(salaries.isna().sum())
```

```
Working_Year              0
Designation               0
Experience                0
Employment_Status         0
Employee_Location         0
Company_Size              0
Remote_Working_Ratio      0
Salary_USD                0
dtype: int64
```

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

datacamp

Converting and analyzing categorical data

EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman
Curriculum Manager, DataCamp

datacamp

Previewing the data

```
print(salaries.select_dtypes("object").head())
```

```
  Designation                Experience    Employment_Status    Employee_Location    Company_Size
0 Data Scientist             Mid           FT                   DE                   L
1 Machine Learning Scientist Senior        FT                   JP                   S
2 Big Data Engineer          Senior        FT                   GB                   M
3 Product Data Analyst       Mid           FT                   HN                   S
4 Machine Learning Engineer  Senior        FT                   US                   L
```

Job titles

```
print(salaries["Designation"].value_counts())
```

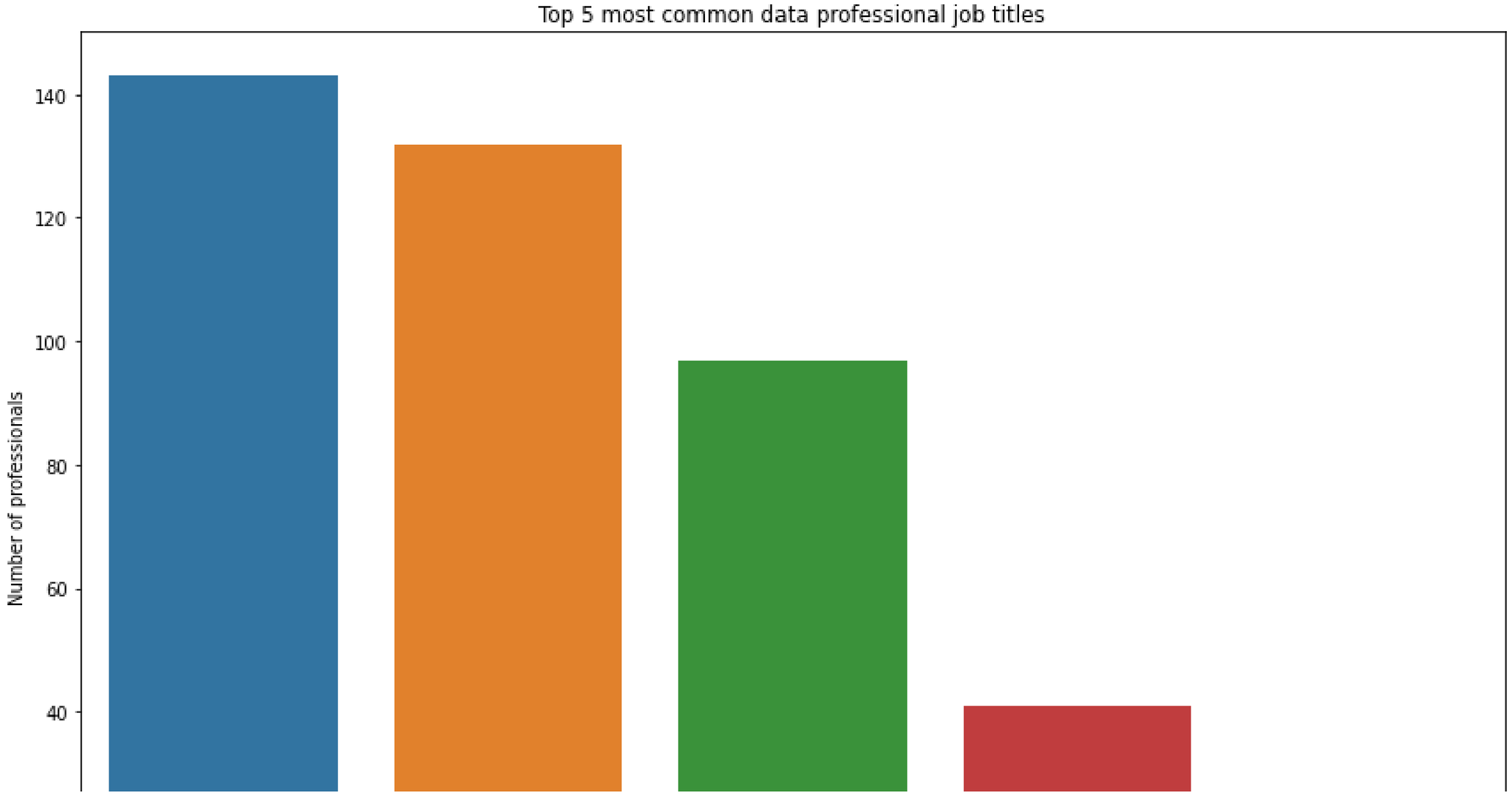
```
Data Scientist                    143
Data Engineer                     132
Data Analyst                       97
Machine Learning Engineer          41
Research Scientist                 16
Data Science Manager               12
Data Architect                     11
Big Data Engineer                   8
Machine Learning Scientist          8
...
```

Job titles

```
print(salaries["Designation"].nunique())
```

```
50
```

Job titles


Top 5 most common data professional job titles
Number of professionals
140
120
100
80
60
40

Extracting value from categories

- Current format limits our ability generate insights
- `pandas.Series.str.contains()`
  - Search a column for a specific string or multiple strings

```
salaries["Designation"].str.contains("Scientist")
```

```
0       True
1       True
2      False
3      False
       ...
604    False
605    False
606     True
Name: Designation, Length: 607, dtype: bool
```

Finding multiple phrases in strings

- Words of interest: Machine Learning **or** AI

```
salaries["Designation"].str.contains("Machine Learning|AI")
```

```
0      False
1       True
2      False
3      False
       ...
604    False
605    False
606     True
Name: Designation, Length: 607, dtype: bool
```

Finding multiple phrases in strings

- Words of interest: Any that start with Data

```
salaries["Designation"].str.contains("^Data")
```

```
0       True
1      False
2      False
3      False
       ...
604     True
605     True
606    False
Name: Designation, Length: 607, dtype: bool
```

Finding multiple phrases in strings

```
job_categories = ["Data Science", "Data Analytics",
                  "Data Engineering", "Machine Learning",
                  "Managerial", "Consultant"]
```

Finding multiple phrases in strings

```
data_science = "Data Scientist|NLP"
data_analyst = "Analyst|Analytics"
data_engineer = "Data Engineer|ETL|Architect|Infrastructure"
ml_engineer = "Machine Learning|ML|Big Data|AI"
manager = "Manager|Head|Director|Lead|Principal|Staff"
consultant = "Consultant|Freelance"
```

Finding multiple phrases in strings

```
conditions = [
    (salaries["Designation"].str.contains(data_science)),
    (salaries["Designation"].str.contains(data_analyst)),
    (salaries["Designation"].str.contains(data_engineer)),
    (salaries["Designation"].str.contains(ml_engineer)),
    (salaries["Designation"].str.contains(manager)),
    (salaries["Designation"].str.contains(consultant))
]
```

Creating the categorical column

```
salaries["Job_Category"] =
```

Creating the categorical column

```
salaries["Job_Category"] = np.select(conditions,
```

Creating the categorical column

```
salaries["Job_Category"] = np.select(conditions,
                                     job_categories,
```

Creating the categorical column

```
salaries["Job_Category"] = np.select(conditions,
                                     job_categories,
                                     default="Other")
```

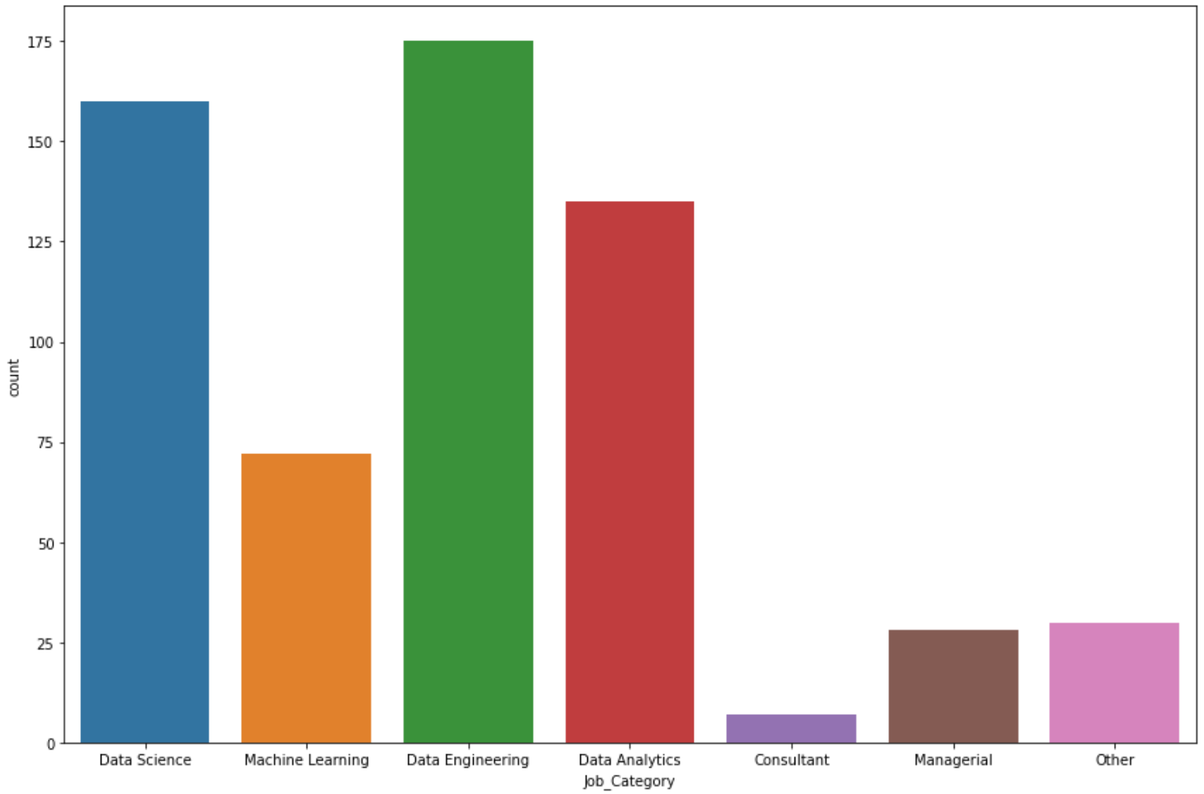
Previewing job categories

```
print(salaries[["Designation", "Job_Category"]].head())
```

```
                  Designation      Job_Category
0              Data Scientist      Data Science
1  Machine Learning Scientist  Machine Learning
2           Big Data Engineer  Data Engineering
3         Product Data Analyst    Data Analytics
4   Machine Learning Engineer  Machine Learning
```

Visualizing job category frequency

```
sns.countplot(data=salaries, x="Job_Category")
plt.show()
```



Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

datacamp

Working with numeric data

EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman
Curriculum Manager, DataCamp

datacamp

The original salaries dataset

```
print(salaries.info())
```

The original salaries dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 594 entries, 0 to 593
Data columns (total 9 columns):
 #   Column               Non-Null Count  Dtype
--   ------               --------------  -----
 0   Working_Year         594 non-null    int64
 1   Designation          567 non-null    object
 2   Experience           561 non-null    object
 3   Employment_Status    563 non-null    object
 4   Salary_In_Rupees     566 non-null    object
 5   Employee_Location    554 non-null    object
 6   Company_Location     570 non-null    object
 7   Company_Size         535 non-null    object
 8   Remote_Working_Ratio  571 non-null    float64
dtypes: float64(1), int64(1), object(7)
memory usage: 41.9+ KB
None
```

Salary in rupees

```
print(salaries["Salary_In_Rupees"].head())
```

```
0    20,688,070.00
1     8,674,985.00
2     1,591,390.00
3    11,935,425.00
4     5,729,004.00
Name: Salary_In_Rupees, dtype: object
```

Converting strings to numbers

- Remove comma values in `Salary_In_Rupees`

- Convert the column to `float` data type

- Create a new column by converting the currency



Converting strings to numbers

```
pd.Series.str.replace("characters to remove", "characters to replace them with")
```

```
salaries["Salary_In_Rupees"] = salaries["Salary_In_Rupees"].str.replace(",", "")
print(salary["Salary_In_Rupees"].head())
```

```
1    20688070.00
2     8674985.00
3     1591390.00
4    11935425.00
5     5729004.00
Name: Salary_In_Rupees, dtype: object
```

Converting strings to numbers

```
salaries["Salary_In_Rupees"] = salaries["Salary_In_Rupees"].astype(float)
```

- 1 Indian Rupee = 0.012 US Dollars

```
salaries["Salary_USD"] = salaries["Salary_In_Rupees"] * 0.012
```

Previewing the new column

```
print(salaries[["Salary_In_Rupees", "Salary_USD"]].head())
```

```
   Salary_In_Rupees  Salary_USD
0        20688070.0  248256.840
1         8674985.0  104099.820
2         1591390.0   19096.680
3        11935425.0  143225.100
4         5729004.0   68748.048
```

Adding summary statistics into a DataFrame

```
salaries.groupby("Company_Size")["Salary_USD"].mean()
```

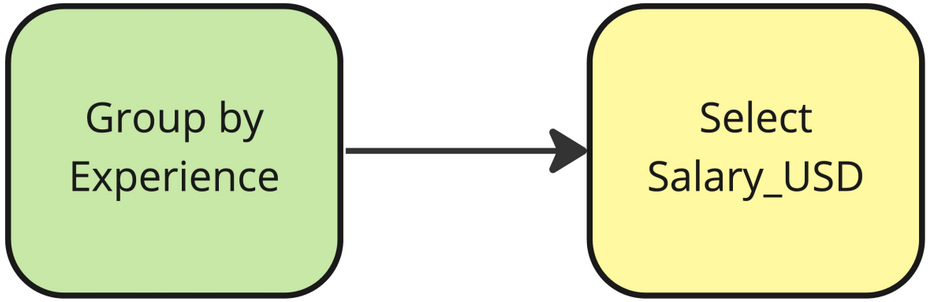
```
Company_Size
L    111934.432174
M    110706.628527
S     69880.980179
Name: Salary_USD, dtype: float64
```

Adding summary statistics into a DataFrame




```
salaries["std_dev"] = salaries.groupby("Experience")
```

Adding summary statistics into a DataFrame



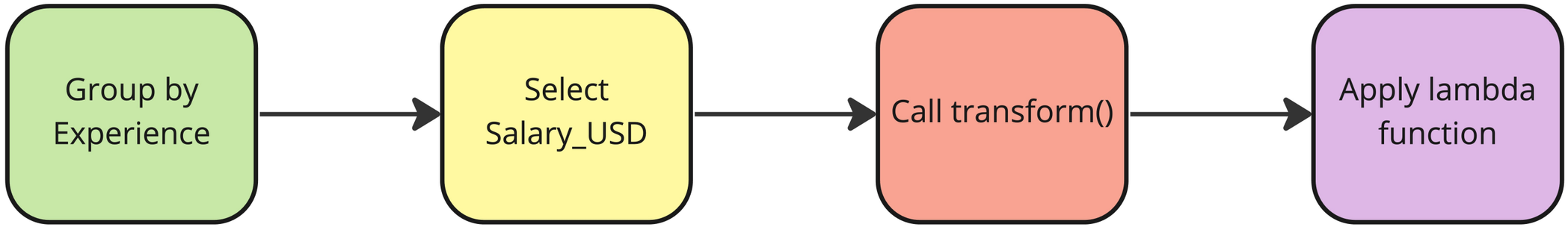

```
salaries["std_dev"] = salaries.groupby("Experience")["Salary_USD"]
```

Adding summary statistics into a DataFrame




```
salaries["std_dev"] = salaries.groupby("Experience")["Salary_USD"].transform(
```

Adding summary statistics into a DataFrame




```
salaries["std_dev"] = salaries.groupby("Experience")["Salary_USD"].transform(lambda x: x.std())
```

Adding summary statistics into a DataFrame

```
print(salaries[["Experience", "std_dev"]].value_counts())
```

```
Experience          std_dev
SE                  52995.385395        257
MI                  63217.397343        197
EN                  43367.256303         83
EX                  86426.611619         24
```

Adding summary statistics into a DataFrame

```
salaries["median_by_comp_size"] = salaries.groupby("Company_Size") \
                                          ["Salary_USD"].transform(lambda x: x.median())
```

```
print(salaries[["Company_Size", "median_by_comp_size"]].head())
```

```
     Company_Size     median_by_comp_size
0    S                60833.424
1    M                105914.964
2    S                60833.424
3    L                95483.400
4    L                95483.400
```

Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

datacamp

Handling outliers

EXPLORATORY DATA ANALYSIS IN PYTHON



George Boorman
Curriculum Manager, DataCamp

datacamp

What is an outlier?

- An observation far away from other data points
  - Median house price: $400,000
  - Outlier house price: $5,000,000

- Should consider why the value is different:
  - Location, number of bedrooms, overall size etc



[1] Image credit: https://unsplash.com/@ralphkayden

Using descriptive statistics

```
print(salaries["Salary_USD"].describe())
```

```
count       518.000
mean     104905.826
std       62660.107
min        3819.000
25%       61191.000
50%       95483.000
75%      137496.000
max      429675.000
Name: Salary_USD, dtype: float64
```

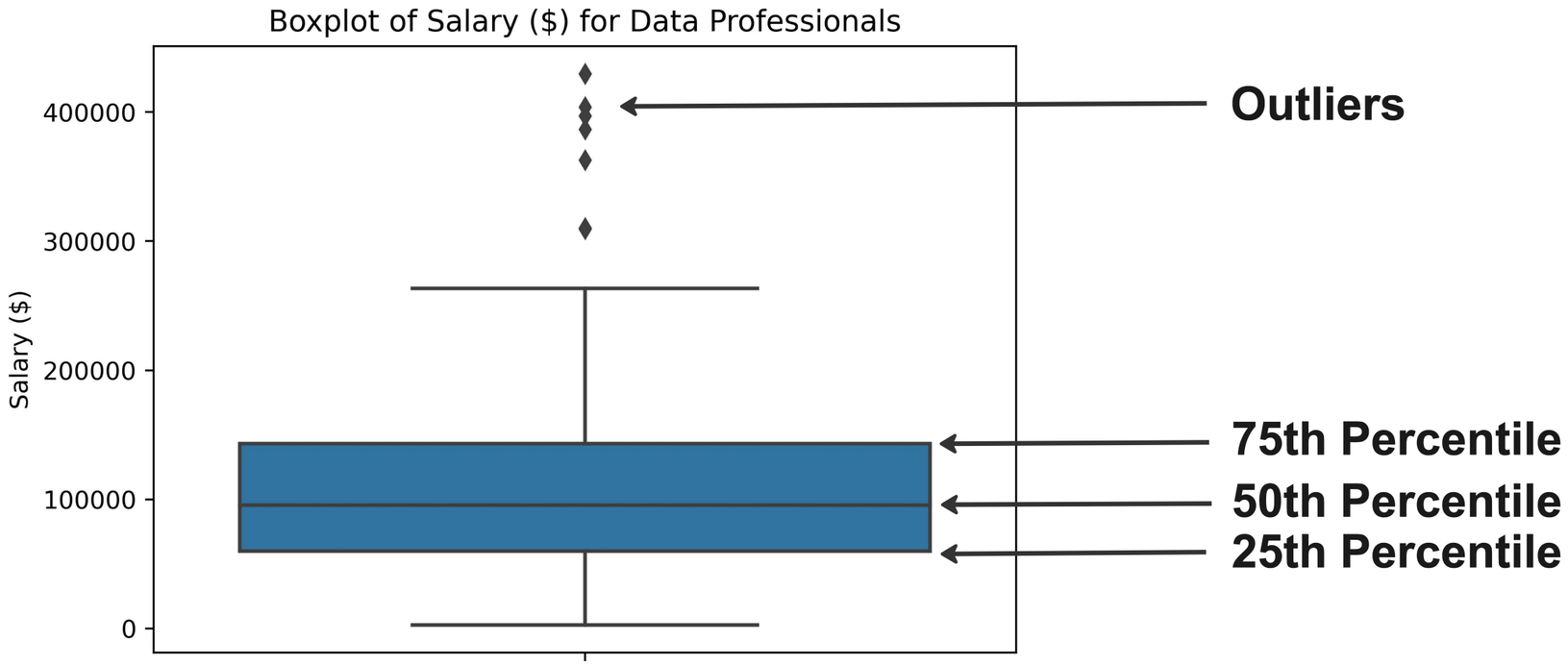
Using the interquartile range

Interquartile range (IQR)

- IQR = 75th - 25th percentile

IQR in box plots

```
sns.boxplot(data=salaries,
            y="Salary_USD")
plt.show()
```



Using the interquartile range

Interquartile range (IQR)

- IQR = 75th - 25th percentile
- Upper Outliers > 75th percentile + (1.5 * IQR)
- Lower Outliers < 25th percentile - (1.5 * IQR)

Identifying thresholds

```
# 75th percentile
seventy_fifth = salaries["Salary_USD"].quantile(0.75)

# 25th percentile
twenty_fifth = salaries["Salary_USD"].quantile(0.25)

# Interquartile range
salaries_iqr = seventy_fifth - twenty_fifth

print(salaries_iqr)
```

```
76305.0
```

Identifying outliers

```
# Upper threshold
upper = seventy_fifth + (1.5 * salaries_iqr)

# Lower threshold
lower = twenty_fifth - (1.5 * salaries_iqr)

print(upper, lower)
```

```
251953.5 -53266.5
```

Subsetting our data

```
salaries[(salaries["Salary_USD"] < lower) | (salaries["Salary_USD"] > upper)] \
        [["Experience", "Employee_Location", "Salary_USD"]]
```

	Experience	Employee_Location	Salary_USD
29	Mid	US	429675.0
67	Mid	US	257805.0
80	Senior	US	263534.0
83	Mid	US	429675.0
133	Mid	US	403895.0
410	Executive	US	309366.0
441	Senior	US	362837.0
445	Senior	US	386708.0
454	Senior	US	254368.0

Why look for outliers?

- Outliers are extreme values
  - may not accurately represent our data
- Can change the mean and standard deviation
- Statistical tests and machine learning models need normally distributed data

What to do about outliers?

Questions to ask:

- Why do these outliers exist?
  - More senior roles / different countries pay more
  - Consider leaving them in the dataset

- Is the data accurate?
  - Could there have been an error in data collection?
    - If so, remove them

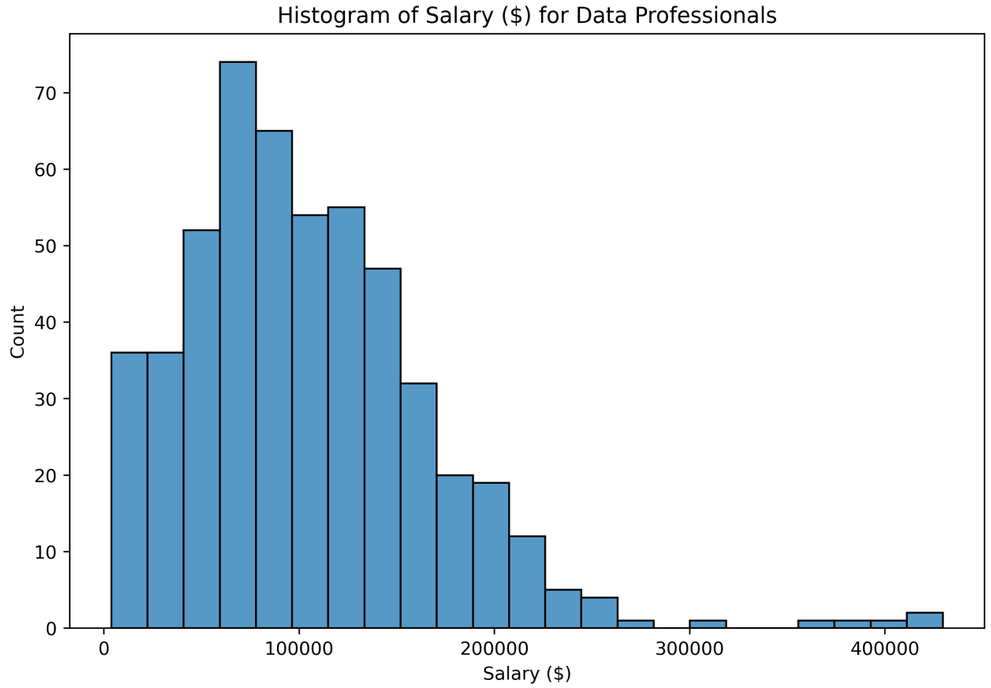
Dropping outliers

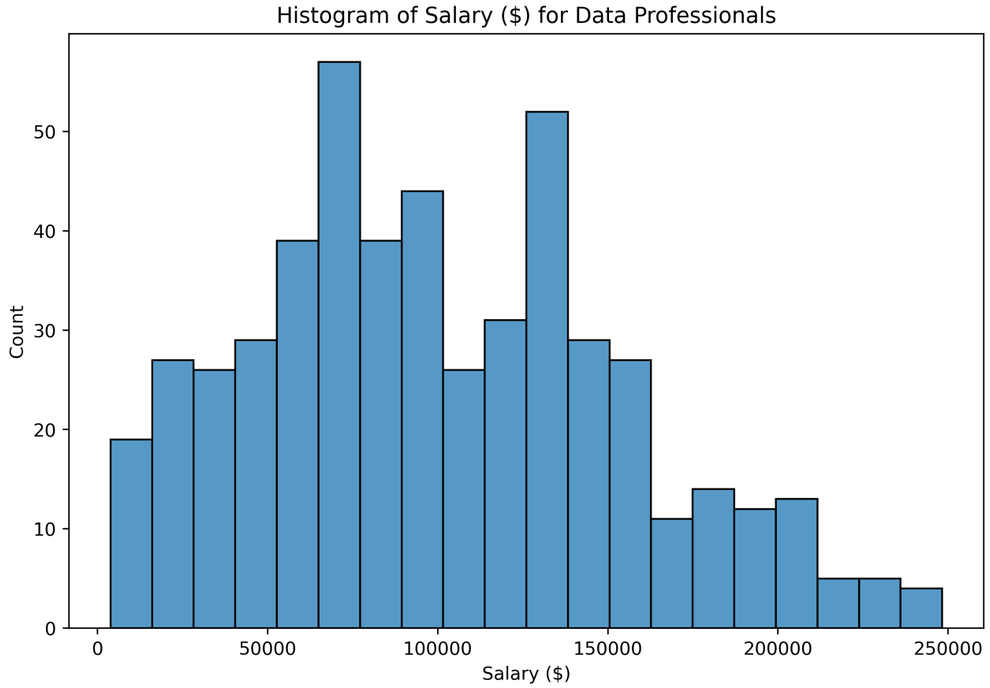
```
no_outliers = salaries[(salaries["Salary_USD"] > lower) & (salaries["Salary_USD"] < upper)]
```

```
print(no_outliers["Salary_USD"].describe())
```

```
count       509.000000
mean     100674.567780
std       53643.050057
min        3819.000000
25%       60928.000000
50%       95483.000000
75%      134059.000000
max      248257.000000
Name: Salary_USD, dtype: float64
```

Distribution of salaries





Let's practice!

EXPLORATORY DATA ANALYSIS IN PYTHON

datacamp