



CLEANING DATA IN PYTHON

# **Diagnose data for cleaning**



# Cleaning data

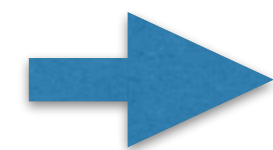
- Prepare data for analysis
- Data almost never comes in clean
- Diagnose your data for problems

# Common data problems

- Inconsistent column names
- Missing data
- Outliers
- Duplicate rows
- Untidy
- Need to process columns
- Column types can signal unexpected data values



# Unclean data



	Continent	Country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
2	NAM	USA	99.0	2.077	3.040600e+08
3	ASI	Indonésie	88.8	2.132	2.273451e+08
4	LAT	Brésil	90.2	1.827	NaN

- Column name inconsistencies
- Missing data
- Country names are in French



# Load your data

```
In [1]: import pandas as pd
```

```
In [2]: df = pd.read_csv('literary_birth_rate.csv')
```



# Visually inspect

```
In [3]: df.head()
```

```
Out[3]:
```

	Continent	Country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
2	NAM	USA	99.0	2.077	3.040600e+08
3	ASI	Indonésie	88.8	2.132	2.273451e+08
4	LAT	Brésil	90.2	1.827	NaN

```
In [4]: df.tail()
```

```
Out[4]:
```

	Continent	Country	female literacy	fertility	population
0	AF	Sao Tomé-et-Principe	90.5	1.769	1.324655e+09
1	LAT	Aruba	50.8	2.682	1.139965e+09
2	ASI	Tonga	99.0	2.077	3.040600e+08
3	OCE	Australia	88.8	2.132	2.273451e+08
4	OCE	Sweden	90.2	1.827	NaN



# Visually inspect

```
In [5]: df.columns
Out[5]: Index(['Continent', 'Country', 'female literacy',
              'fertility', 'population'], dtype='object')
```

```
In [6]: df.shape
Out[6]: (164, 5)
```

```
In [7]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 5 columns):
Continent      164 non-null object
Country        164 non-null object
female literacy 164 non-null float64
fertility       164 non-null object
population     122 non-null float64
dtypes float64(2), object(3)
memory usage: 6.5+ KB
```



CLEANING DATA IN PYTHON

**Let's practice!**





CLEANING DATA IN PYTHON

# Exploratory data analysis



# Frequency counts

- Count the number of unique values in our data



# Data type of each column

```
In [1]: df.info()
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 164 entries, 0 to 163
Data columns (total 5 columns):
continent                164 non-null object
country                  164 non-null object
female literacy          164 non-null float64
fertility                 164 non-null object
population               122 non-null float64
dtypes float64(2), object(3)
memory usage: 6.5+ KB
```



# Frequency counts: continent

```
In [2]: df.continent.value_counts(dropna=False)
```

```
Out[2]:
```

```
AF      49
```

```
ASI     47
```

```
EUR     36
```

```
LAT     24
```

```
OCE      6
```

```
NAM      2
```

```
Name: continent, dtype: int64
```



# Frequency counts: continent

```
In [3]: df['continent'].value_counts(dropna=False)
```

```
Out[3]:
```

```
AF      49
```

```
ASI     47
```

```
EUR     36
```

```
LAT     24
```

```
OCE      6
```

```
NAM      2
```

```
Name: continent, dtype: int64
```



# Frequency counts: country

```
In [4]: df.country.value_counts(dropna=False).head()
```

```
Out[4]:
```

```
Sweden    2
```

```
Algerie    1
```

```
Germany    1
```

```
Angola     1
```

```
Indonésie  1
```

```
Name: country, dtype: int64
```



# Frequency counts: fertility

```
In [5]: df.fertility.value_counts(dropna=False).head()
```

```
Out[5]:
```

```
missing    5
```

```
1.854      2
```

```
1.93       2
```

```
1.841      2
```

```
1.393      2
```

```
Name: fertility, dtype: int64
```



# Frequency counts: population

```
In [6]: df.population.value_counts(dropna=False).head()
```

```
Out[6]:
```

NaN	42
-----	----

5.667325e+06	1
--------------	---

3.773100e+06	1
--------------	---

1.333388e+06	1
--------------	---

1.661115e+08	1
--------------	---

```
Name: population, dtype: int64
```





# Summary statistics

- Numeric columns
- Outliers
  - Considerably higher or lower
  - Require further investigation



# Summary statistics: Numeric data

```
In [7]: df.describe()
```

```
Out[7]:
```

	female_literacy	population
count	164.000000	1.220000e+02
mean	80.301220	6.345768e+07
std	22.977265	2.605977e+08
min	12.600000	1.035660e+05
25%	66.675000	3.778175e+06
50%	90.200000	9.995450e+06
75%	98.500000	2.642217e+07
max	100.000000	2.313000e+09



CLEANING DATA IN PYTHON

**Let's practice!**



CLEANING DATA IN PYTHON

# **Visual exploratory data analysis**



# Data visualization

- Great way to spot outliers and obvious errors
- More than just looking for patterns
- Plan data cleaning steps



# Summary statistics

```
In [1]: df.describe()
```

```
Out[1]:
```

	female_literacy	fertility	population
count	164.000000	163.000000	1.220000e+02
mean	80.301220	2.872853	6.345768e+07
std	22.977265	1.425122	2.605977e+08
min	12.600000	0.966000	1.035660e+05
25%	66.675000	1.824500	3.778175e+06
50%	90.200000	2.362000	9.995450e+06
75%	98.500000	3.877500	2.642217e+07
max	100.000000	7.069000	2.313000e+09



# Bar plots and histograms

- Bar plots for discrete data counts
- Histograms for continuous data counts
- Look at frequencies



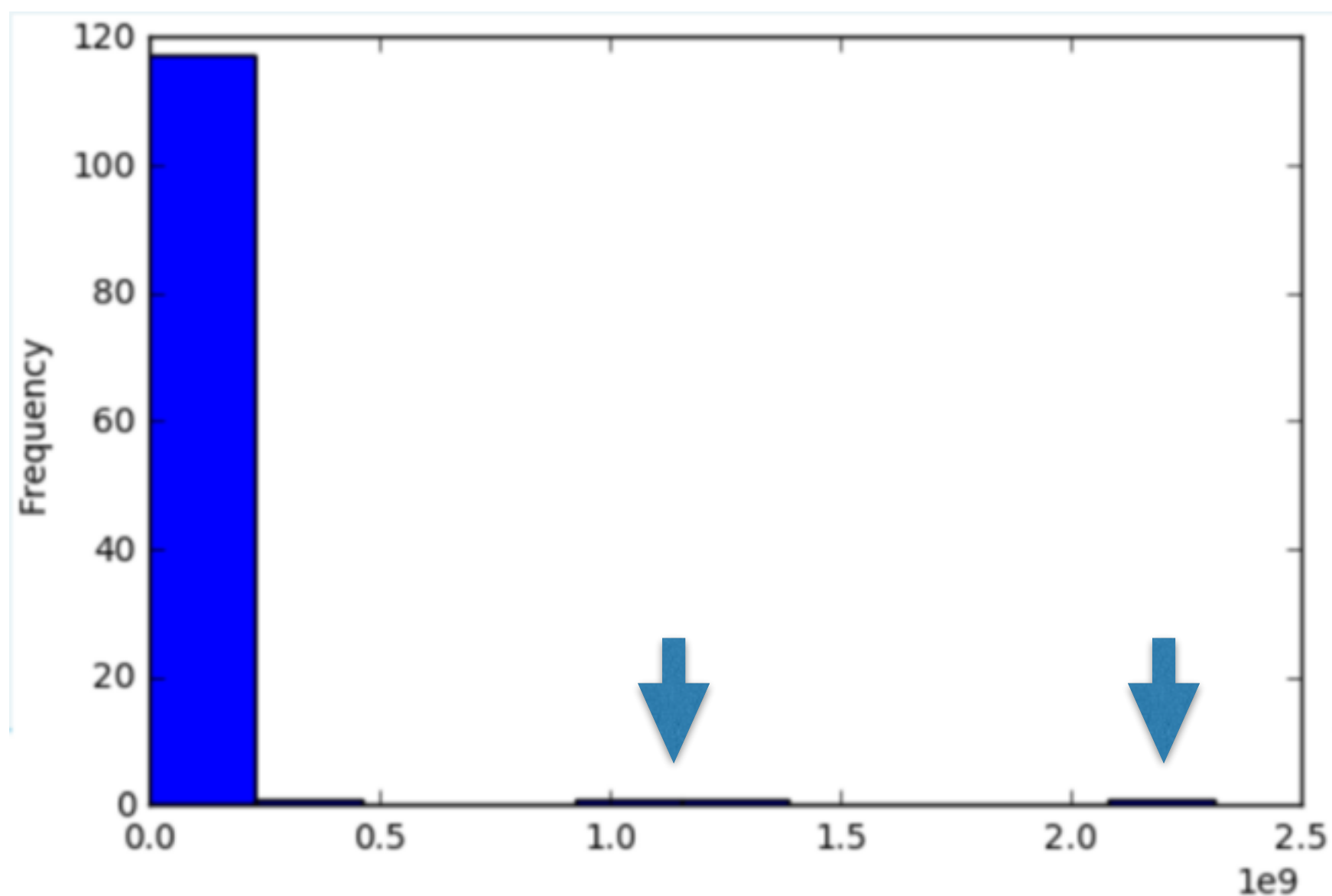
# Histogram

```
In [2]: df.population.plot('hist')
```

```
Out[2]: <matplotlib.axes._subplots.AxesSubplot at 0x7f78e4abafd0>
```

```
In [3]: import matplotlib.pyplot as plt
```

```
In [4]: plt.show()
```







# Identifying the error

```
In [5]: df[df.population > 10000000000]
```

```
Out[5]:
```

	continent	country	female literacy	fertility	population
0	ASI	Chine	90.5	1.769	1.324655e+09
1	ASI	Inde	50.8	2.682	1.139965e+09
162	OCE	Australia	96.0	1.930	2.313000e+09

- Not all outliers are bad data points
- Some can be an error, but others are valid values



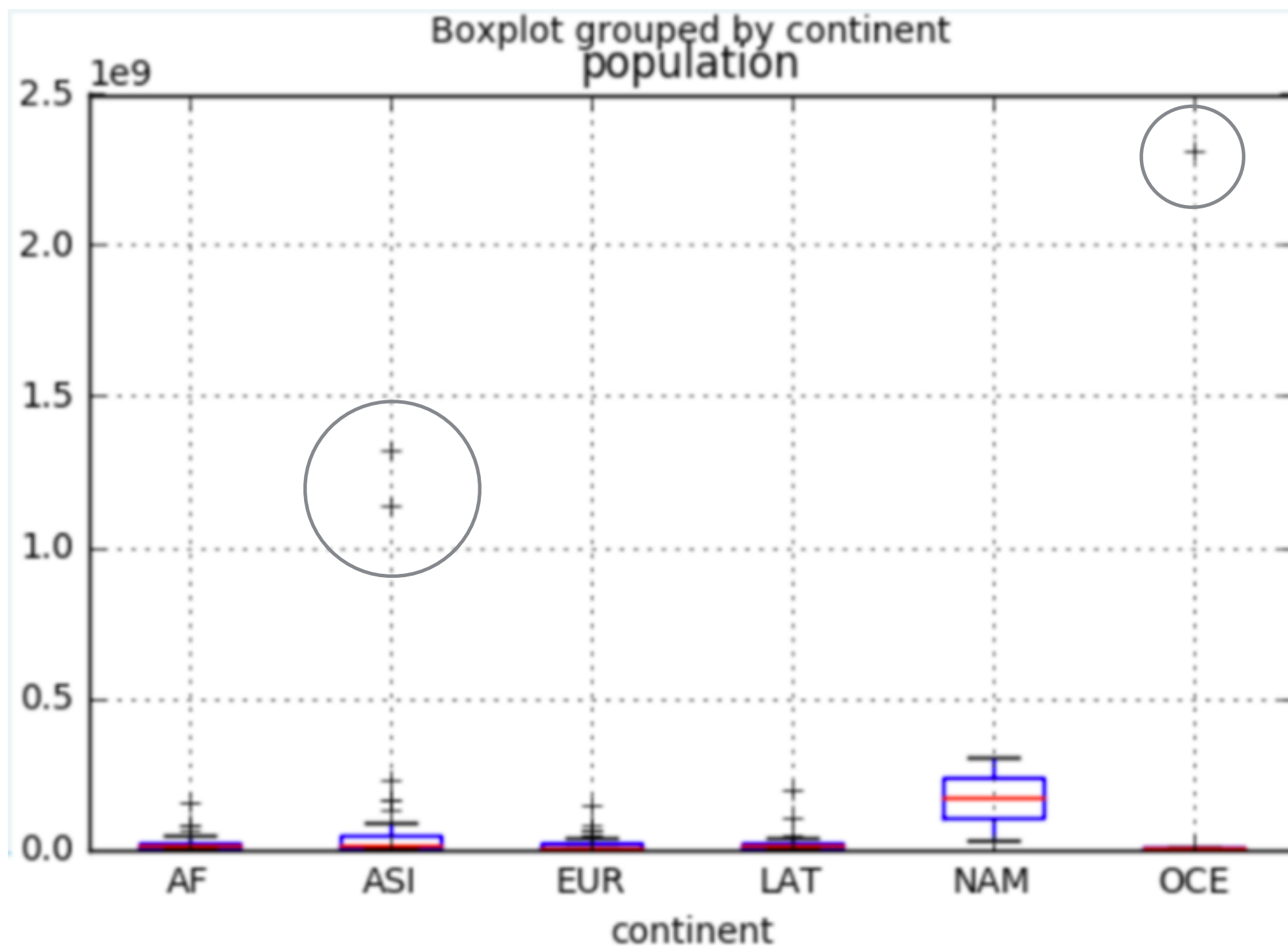
# Box plots

- Visualize basic summary statistics
  - Outliers
  - Min/max
  - 25th, 50th, 75th percentiles



# Box plot

```
In [6]: df.boxplot(column='population', by='continent')  
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x7ff5581bb630>  
  
In [7]: plt.show()
```





# Scatter plots

- Relationship between 2 numeric variables
- Flag potentially bad data
  - Errors not found by looking at 1 variable



CLEANING DATA IN PYTHON

**Let's practice!**