dm24s1

Topic 04 : Exploratory Data Analysis

Part 01 : EDA

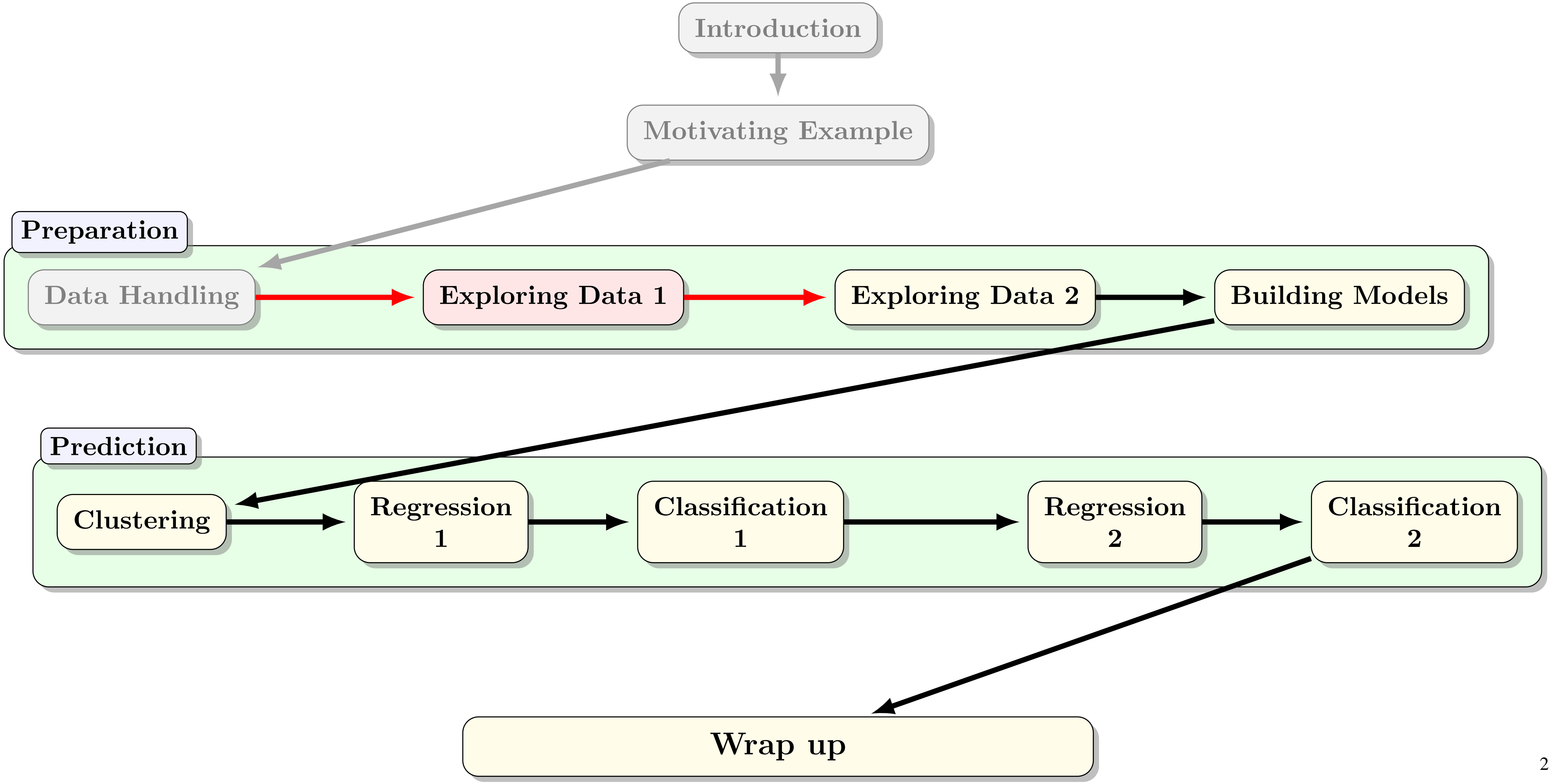
Dr Bernard Butler
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

Autumn Semester, 2024

Outline

- EDA Process
- Datasets = Tips, Titanic and Algae Blooms

Data Mining (Week 4)



EDA — Summary

Acknowledgment

A big thanks to Dr Kieran Murphy, who provided many of the slides for today's lecture.

Introduction

Exploratory Data Analysis (EDA)

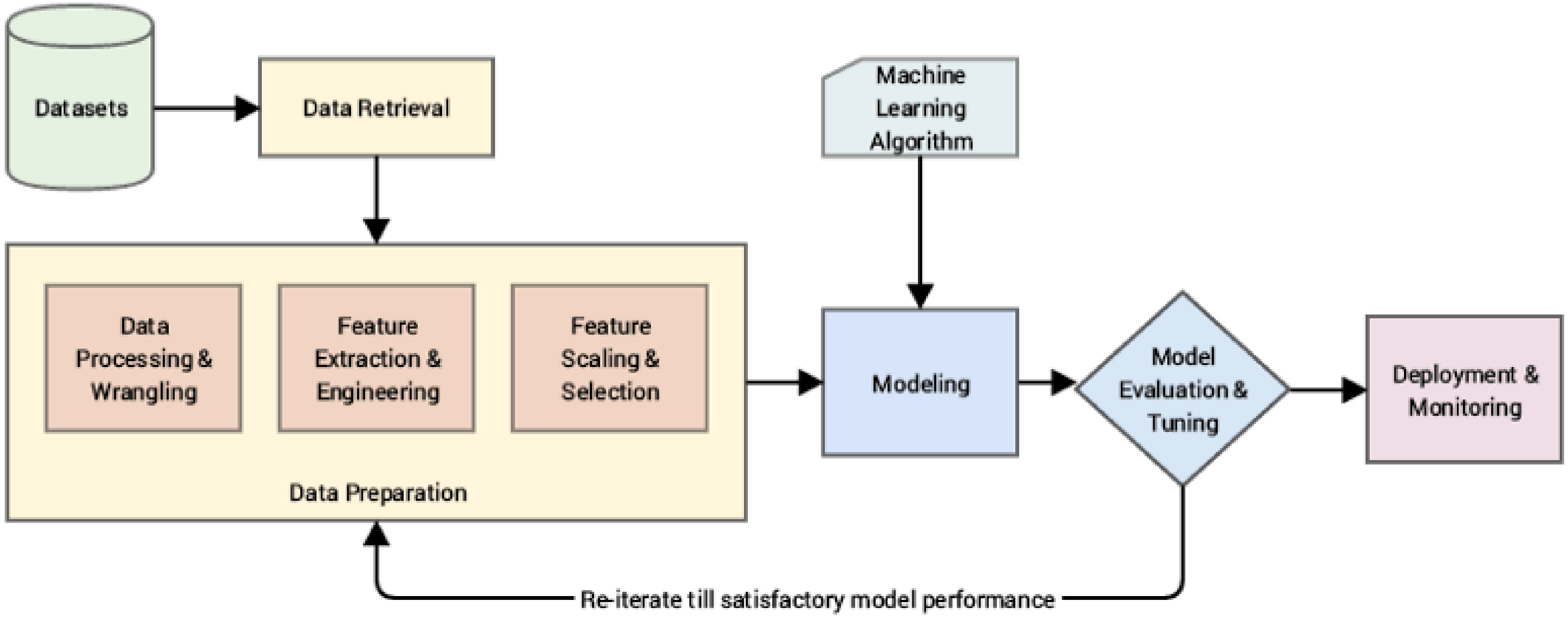
Aim

To understand and summarise a dataset to ensure that the features which are feed to machine learning algorithms are refined and that the results are valid and can correctly interpreted.

Benefits

- Develop insight about the dataset and understanding of the underlying structure.
- Extract important parameters and relationships that hold between them.
- Test underlying assumptions.
- Identify issues that affect model performance — outliers, missing values.

Data Pipeline



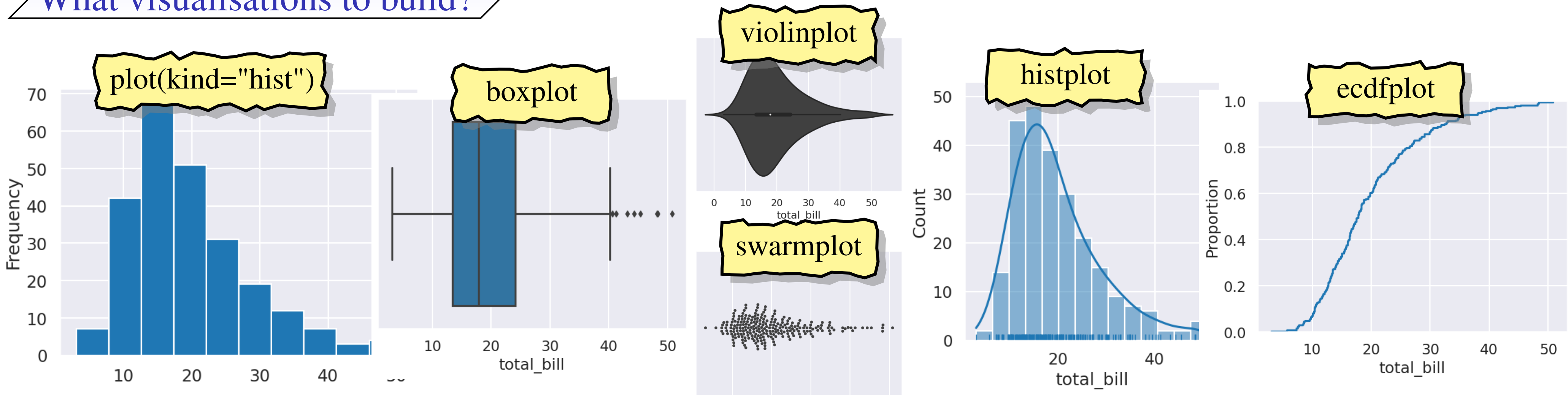
- Data preparation is the core of the data mining pipeline (typical estimates >50% of the time/effort).
- EDA is the data processing and wrangling.
- EDA informs the feature extraction, engineering, transformation and selection.

The Bad News — ‘The curse of choice’

What questions to ask?

Dataset global questions: How many features? How many observations? What is the data type of each feature? Any null values? … Feature specific questions: What is the distribution of each variable? Do there appear to be outliers? What features are related? … Missing value questions: Are null value a result of the way data was recorded? Can we drop the rows with null values without it significantly affecting your analysis? Can we justify filling in the missing values with the mean or median for that variable? If the data is time-series data, can we fill the missing values with interpolation? Are there so many missing values for a variable that we should drop that variable from the dataset? … Outlier questions: Why are outliers present? Do the outliers represent real observations (i.e. not errors)? Should we exclude these observations? If not, should we winsorise the values? … Correlations/Relationships questions: Which variables are most correlated with your target variable? (If applicable) Is there multicollinearity? (Two features that have a correlation > 0.8) How will this affect your model? Do you have variables that represent the same information? Can one be dropped? …

What visualisations to build?



Have a plan, be selective, understand strengths/weaknesses of metrics/visualisations

Terminology / Notation

$n+1$ columns / variables

$\mathbf{X}$

n features / attributes / dimensions

$\mathbf{y}$

target

m observations / instances / cases / rows

PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked	Survived
1	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S	0
2	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C	1
3	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S	1
4	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S	1
5	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S	0
6	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q	0
7	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S	0
8	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S	0
9	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S	1
10	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C	1
11	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S	1

$\mathbf{x}^{(i)}$

$\mathbf{x}_j$

- A labeled dataset consists of m rows $\times$ $(n+1)$ columns / variables.
- Use bold to represent vectors and matrices.
- Use subscripts to indicate particular feature / attribute / column . . . . $\mathbf{x}_j$
- Use superscript in parenthesis to indicate particular observation / instance/ case / row . . . . $\mathbf{x}^{(i)}$
- So $x_j^{(i)}$ (or $x_{i,j}$) is the i-th observation in the j-th feature . . . . $x_j^{(i)}$

Example Datasets

We will use a few datasets today to illustrate the various features:

Tips

- Small dataset of total bills, and tips for different servers with gender, day, time and group size.
- Clean, no missing values, some outliers.
- Task: exploratory data analysis

Titanic

- Classic dataset with passenger information for the Titanic's fatal voyage, and whether they survived.
- Has missing values and information rich text fields (Name, ticket number).
- Task: classification — predict whether a passenger survived.

Algae Blooms

- Water quality study where samples were taken from different rivers over time.
- Recorded levels of (seven) chemical substances and population of (six) algae species and other information on the sample conditions.
- Task: regression — predict algae population level (7 separate populations).

Tips dataset

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

No target column, so mainly just an exploratory data analysis problem. But questions of interest:

- How do factors `sex`, `smoker`, `day`, `time`, or `size` affect tip / percentage tip?
- Does `size` vary with `day`, `time`, `smoker`?

But some questions don't make sense

- What is the relationship between `sex` and `smoker`? — why should they be related?

This is the downside of automatic EDA tools such as `pandas−profiling` — you will drown in statistics / charts.

Algae Blooms dataset

	Season	Size	Speed	max_pH	min_O2	mean_Cl	mean_NO3	mean_NH4	mean_oPO4	mean_PO4	mean_Chlor	a1	a2	a3	a4	a5
0	winter	small	medium	8.00	9.8	60.800	6.238	578.00000	105.00000	170.00000	50.000	0.0	0.0	0.0	0.0	34.2
1	spring	small	medium	8.35	8.0	57.750	1.288	370.00000	428.75000	558.75000	1.300	1.4	7.6	4.8	1.9	6.7
2	autumn	small	medium	8.10	11.4	40.020	5.330	346.66699	125.66700	187.05701	15.600	3.3	53.6	1.9	0.0	0.0
3	spring	small	medium	8.07	4.8	77.364	2.302	98.18200	61.18200	138.70000	1.400	3.1	41.0	18.9	0.0	1.4
4	autumn	small	medium	8.06	9.0	55.350	10.416	233.70000	58.22200	97.58000	10.500	9.2	2.9	7.5	0.0	7.5
5	winter	small	high	8.25	13.1	65.750	9.248	430.00000	18.25000	56.66700	28.400	15.1	14.6	1.4	0.0	22.5
6	summer	small	high	8.15	10.3	73.250	1.535	110.00000	61.25000	111.75000	3.200	2.4	1.2	3.2	3.9	5.8
7	autumn	small	high	8.05	10.6	59.067	4.990	205.66701	44.66700	77.43400	6.900	18.2	1.6	0.0	0.0	5.5
8	winter	small	medium	8.70	3.4	21.950	0.886	102.75000	36.30000	71.00000	5.544	25.4	5.4	2.5	0.0	0.0
9	winter														2.9	0.0
10	spring	small	high	7.70	10.2	8.000	1.527	21.57100	12.75000	20.75000	0.800	16.6	0.0	0.0	0.0	1.2
11	summer	small	high	7.45	11.7	8.690	1.588	18.42900	10.66700	19.00000	0.600	32.1	0.0	0.0	0.0	0.0
12	winter	small	high	7.74	9.6	5.000	1.223	27.28600	12.00000	17.00000	41.000	43.5	0.0	2.1	0.0	1.2
13	summer	small	high	7.72	11.8	6.300	1.470	8.00000	16.00000	15.00000	0.500	31.1	1.0	3.4	0.0	1.9
14	winter	small	high	7.90	9.6	3.000	1.448	46.20000	13.00000	61.60000	0.300	52.2	5.0	7.8	0.0	4.0
15	autumn	small	high	7.55	11.5	4.700	1.320	14.75000	4.25000	98.25000	1.100	69.9	0.0	1.7	0.0	0.0
16	winter	small	high	7.78	12.0	7.000	1.420	34.33300	18.66700	50.00000	1.100	46.2	0.0	0.0	1.2	0.0
17	spring	small	high	7.61	9.8	7.000	1.443	31.33300	20.00000	57.83300	0.400	31.8	0.0	3.1	4.8	7.7
18	summer	small	high	7.35	10.4	7.000	1.718	49.00000	41.50000	61.50000	0.800	50.6	0.0	9.9	4.3	3.6
19	spring	small	medium	7.79	3.2	64.000	2.822	8777.59961	564.59998	771.59998	4.500	0.0	0.0	0.0	44.6	0.0

How well can we predict the (7) different algae population levels using water sample information?

Titanic dataset

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S
5	6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583	NaN	Q
6	7	0	1	McCarthy, Mr. Timothy J	male	54.0	0	0	17463	51.8625	E46	S
7	8	0	3	Palsson, Master. Gosta Leonard	male	2.0	3	1	349909	21.0750	NaN	S
8	9	1	3	Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)	female	27.0	0	2	347742	11.1333	NaN	S
9	10	1	2	Nasser, Mrs. Nicholas (Adele Achem)	female	14.0	1	0	237736	30.0708	NaN	C
10	11	1	3	Sandstrom, Miss. Marguerite Rut	female	4.0	1	1	PP 9549	16.7000	G6	S
11	12	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	C103	S
12	13	0	3	Saundercock, Mr. William Henry	male	20.0	0	0	A/5. 2151	8.0500	NaN	S
13	14	0	3	Andersson, Mr. Anders Johan	male	39.0	1	5	347082	31.2750	NaN	S
14	15	0	3	Vestrom, Miss. Hulda Amanda Adolfina	female	14.0	0	0	350406	7.8542	NaN	S
15	16	1	2	Hewlett, Mrs. (Mary D Kingcome)	female	55.0	0	0	248706	16.0000	NaN	S
16	17	0	3	Rice, Master. Eugene	male	2.0	4	1	382652	29.1250	NaN	Q

How well can we predict a passenger's survival using information at time of departure?

Before we start . . . Loading libraries

We start by loading in the core data science modules. . .

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

`matplotlib` is an excellent visualisation library but some plots needs additional configuration. `seaborn` sits above `matplotlib` and has a collection of visualisations optimised for statistical analysis. . . .

```
import seaborn as sns
```

Next, we import some statistical modules . . .

```
import scipy.stats as stats
import statsmodels.api as sm
import pingouin as pg
```

`scipy.stats` has a large number of distributions, parametric and nonparametric statistical tests, and descriptive statistics.
`statsmodels` is more focused on estimating statistical models.
`pingouin` overlaps with bits of `scipy.stats` and `statsmodels` but generates more details and nicer visualisations.

Finally we set options . . .

```
plt.style.use("seaborn-v0_8-darkgrid")
```

Before we start . . . auto EDA using ydata−profiling

```python
from pandas_profiling import ProfileReport
profile = ProfileReport(df, title="Tips Report", html={"style": {"full_width": True}}, sort="None")
profile
```

Summarize dataset: 100% 21/21 [00:05<00:00, 4.00it/s, Completed]

Generate report structure: 100% 1/1 [00:02<00:00, 2.78s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.72it/s]

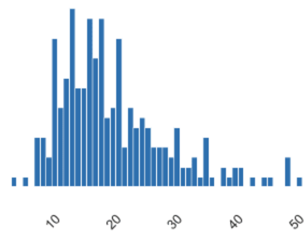
> `ydata−profiling` is nice, but see how slow it is on this tiny dataset. What would happen if we had 100K rows x 100 columns?

Tips Report — Overview | Variables | Interactions | Correlations | Missing values | Sample | Duplicate rows

Variables

total_bill
Real number ($\mathbb{R}_{\geq 0}$)

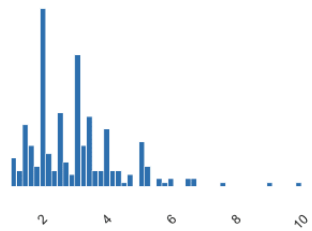
Distinct	229	Mean	19.78594262
Distinct (%)	93.9%	Minimum	3.07
Missing	0	Maximum	50.81
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	1.9 KiB



Toggle details

tip
Real number ($\mathbb{R}_{\geq 0}$)

Distinct	123	Mean	2.998278689
Distinct (%)	50.4%	Minimum	1
Missing	0	Maximum	10
Missing (%)	0.0%	Zeros	0
Infinite	0	Zeros (%)	0.0%
Infinite (%)	0.0%	Memory size	1.9 KiB



Before we start . . . zero-code EDA using `dtale`

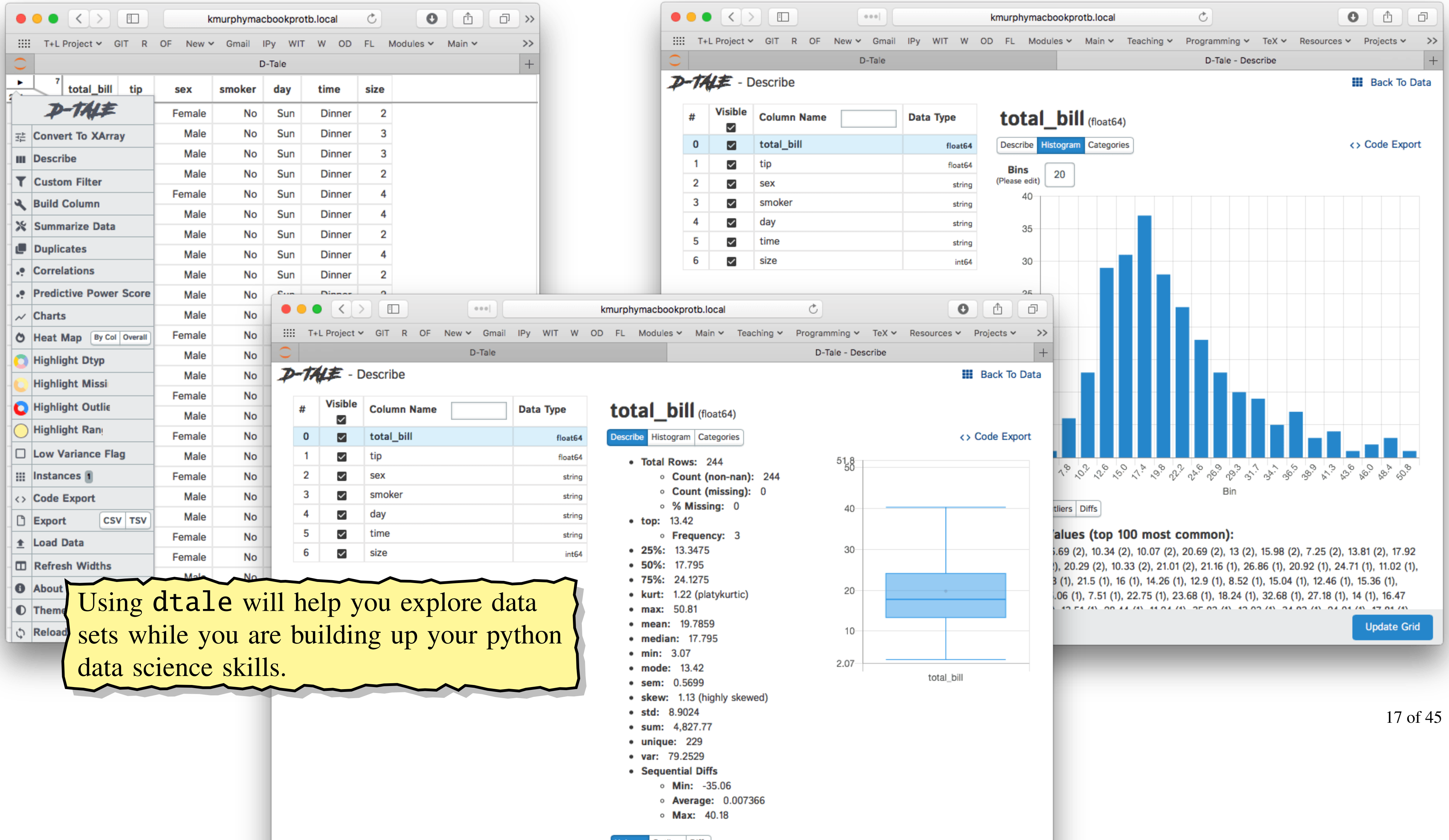
Well, almost zero code....

```
import pandas as pd
import dtale

# Read the Tips data into a dataframe, check it looks OK
df = pd.read_csv('tips.csv')
df.head()

# Run dtale to visualize the structure of the dataframe
dtale.show(df)
```

Before we start . . . zero-code EDA using `dtale`



Using `dtale` will help you explore data sets while you are building up your python data science skills.

First Pass — Load Dataset and Initial Clean

- Load dataset
- Check variables names
- Verify variable types
- Identify (and possibly address) missing values

Tips — Load

```
df = pd.read_csv("data/tips.csv")
print(df.shape)
df.head(10)
```

```
(244, 7)
```

	total_bill	tip	sex	smoker	day	time	size
0	16.99	1.01	Female	No	Sun	Dinner	2
1	10.34	1.66	Male	No	Sun	Dinner	3
2	21.01	3.50	Male	No	Sun	Dinner	3
3	23.68	3.31	Male	No	Sun	Dinner	2
4	24.59	3.61	Female	No	Sun	Dinner	4
5	25.29	4.71	Male	No	Sun	Dinner	4
6	8.77	2.00	Male	No	Sun	Dinner	2
7	26.88	3.12	Male	No	Sun	Dinner	4
8	15.04	1.96	Male	No	Sun	Dinner	2
9	14.78	3.23	Male	No	Sun	Dinner	2

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    object
 3   smoker      244 non-null    object
 4   day         244 non-null    object
 5   time        244 non-null    object
 6   size        244 non-null    int64
dtypes: float64(2), int64(1), object(4)
memory usage: 13.5+ KB
```

Issue: categorical data treated as object (string).

Tips — Fix Data Types I

```
df.sex.unique()
```

```
array(['Female', 'Male'], dtype=object)
```

```
df.sex = pd.Categorical(df.sex)
df.sex.unique()
```

```
['Female', 'Male']
Categories (2, object): ['Female', 'Male']
```

```
df.smoker.unique()
```

```
array(['No', 'Yes'], dtype=object)
```

```
df.smoker = pd.Categorical(df.smoker)
df.smoker.unique()
```

```
['No', 'Yes']
Categories (2, object): ['No', 'Yes']
```

```
df.day.unique()
```

```
array(['Sun', 'Sat', 'Thur', 'Fri'], dtype=object)
```

```
df.day = pd.Categorical(df.day, categories=['Thur', 'Fri','Sun', 'Sat'], ordered=True)
df.day.unique()
```

```
['Sun', 'Sat', 'Thur', 'Fri']
Categories (4, object): ['Thur' < 'Fri' < 'Sun' < 'Sat']
```

`Tips` — fix datatypes II

```
df.time = pd.Categorical(df.time, categories=['Lunch', 'Dinner'], ordered=True)
df.time.unique()
```

```
['Dinner', 'Lunch']
Categories (2, object): ['Lunch' < 'Dinner']
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 244 entries, 0 to 243
Data columns (total 7 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   total_bill  244 non-null    float64
 1   tip         244 non-null    float64
 2   sex         244 non-null    category
 3   smoker      244 non-null    category
 4   day         244 non-null    category
 5   time        244 non-null    category
 6   size        244 non-null    int64
dtypes: category(4), float64(2), int64(1)
memory usage: 7.4 KB
```

Converting to category will:

- Simplify visualisation (order can be preserved).
- Reduce memory usage (not that big a deal for us).
- Speed up I/O (depending on file format).

⇒ Convert to category is a bigger deal for features where the levels have an order.

`Titanic` — load

- Dataset is split into two parts:
  - `train.csv` — 891 rows with `Survived` column, used in EDA and model training.
  - `test.csv` — 418 rows without the `Survived` column, used in competition scoring.

```python
df = pd.read_csv("data/train.csv")
print(df.shape)
df.head(25)
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarke
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
											C123	S
											NaN	S
											NaN	Q
											E46	S
											NaN	S
											NaN	S
											NaN	C
											G6	S
											C103	S
											NaN	S
											NaN	S
											NaN	S
											NaN	S
											NaN	Q
											NaN	S
											NaN	S
19	20	1	3	Masselmani, Mrs. Fatima	female	NaN	0	0	2649	7.2250	NaN	C
20	21	0	2	Fynney, Mr. Joseph J	male	35.0	0	0	239865	26.0000	NaN	S
21	22	1	2	Beesley, Mr. Lawrence	male	34.0	0	0	248698	13.0000	D56	S
22	23	1	3	McGowan, Miss. Anna "Annie"	female	15.0	0	0	330923	8.0292	NaN	Q
23	24	1	1	Sloper, Mr. William Thompson	male	28.0	0	0	113788	35.5000	A6	S
24	25	0	3	Palsson, Miss. Torborg Danira	female	8.0	3	1	349909	21.0750	NaN	S

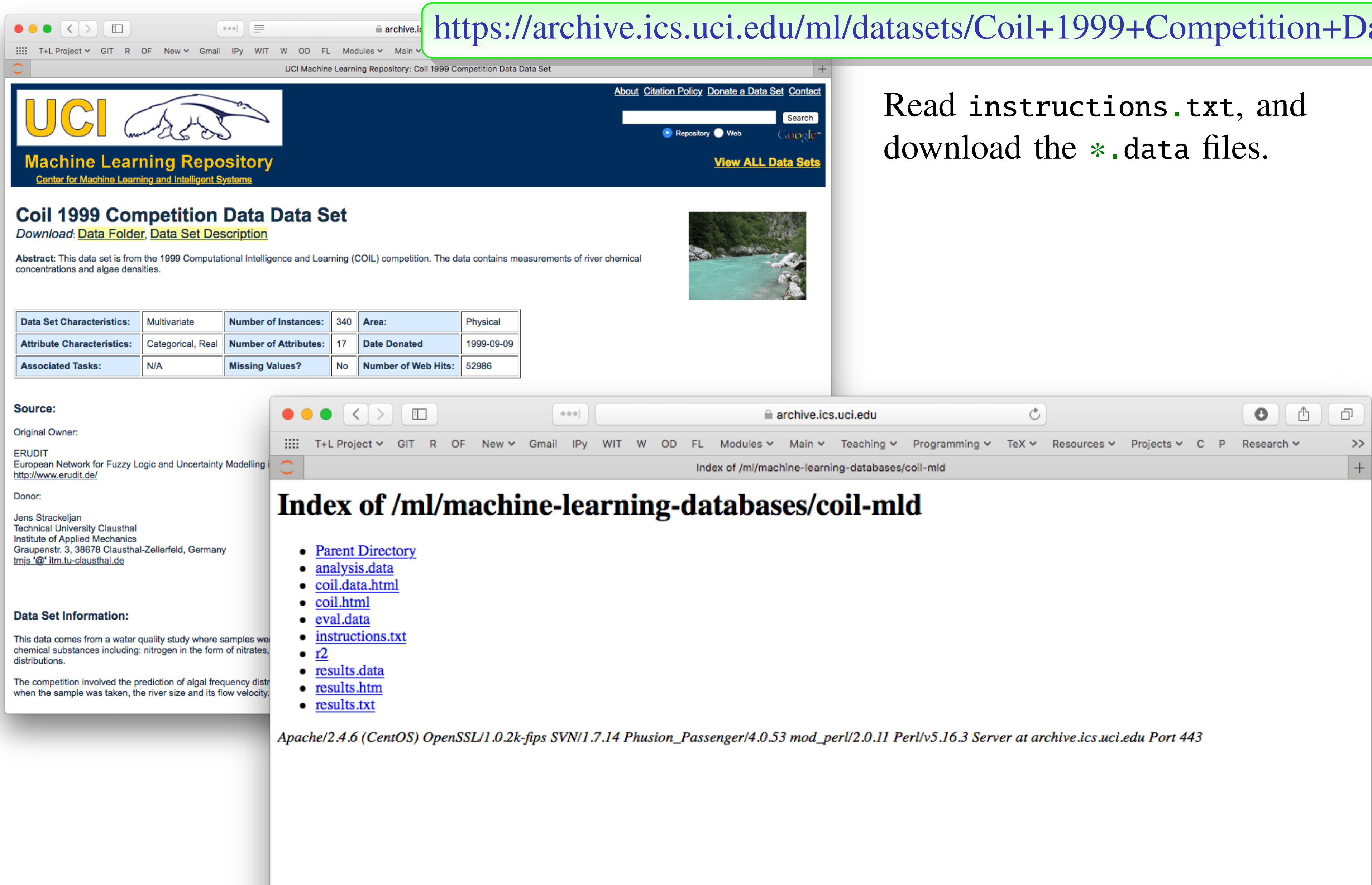
- We could convert `Sex` or `Embarked`, to a category, but since their levels are not ordered there is no big advantage.
- We don't want to convert `Name`, `Ticket` and `Cabin` since we want to perform further text processing on these columns. For example, extracting title (Capt, Mr, Miss, etc.) out of `Name`.
- We have missing values (**that are plausibly linked to target**) that we need to deal with.

```python
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    int64
 3   Name         891 non-null    object
 4   Sex          891 non-null    object
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64
 7   Parch        891 non-null    int64
 8   Ticket       891 non-null    object
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

`Algae_Blooms` – load I

https://archive.ics.uci.edu/ml/datasets/Coil+1999+Competition+Data



Read `instructions.txt`, and download the `*.data` files.

Algae_Blooms — load (1st attempt) II

Pandas function `pd.read_table`, is a more general function than `read_csv`.

```
df = pd.read_table('src/Analysis.txt')
print(df.shape)
df.head()
```

```
(199, 1)
```

	winter small medium 8.00000 9.80000 60.80000 6.23800 578.00000 105.00000 170.00000 50.00000 0.00000 0.00000 0.00000 0.00000 34.20000 8.30000 0.00000
0	spring small medium 8.35000 ...
1	autumn small medium 8.10000 1...
2	spring small medium 8.07000 ...
3	autumn small medium 8.06000 ...
4	winter small high 8.25000 13....

Two problems, first row was treated as column headers, and we need to specify the character(s) used to separate columns

`Algae_Blooms` — load (2nd attempt) III

```
df = pd.read_table('src/Analysis.txt', sep='\s+', header=None)
print(df.shape)
df.head()
```

```
(200, 18)
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
0	winter	small	medium	8.00000	9.80000	60.80000	6.23800	578.00000	105.00000	170.00000	50.00000	0.0	0.0	0.0	0.0	34.2	8.3	0.0
1	spring	small	medium	8.35000	8.00000	57.75000	1.28800	370.00000	428.75000	558.75000	1.30000	1.4	7.6	4.8	1.9	6.7	0.0	2.1
2	autumn	small	medium	8.10000	11.40000	40.02000	5.33000	346.66699	125.66700	187.05701	15.60000	3.3	53.6	1.9	0.0	0.0	0.0	9.7
3	spring	small	medium	8.07000	4.80000	77.36400	2.30200	98.18200	61.18200	138.70000	1.40000	3.1	41.0	18.9	0.0	1.4	0.0	1.4
4	autumn	small	medium	8.06000	9.00000	55.35000	10.41600	233.70000	58.22200	97.58000	10.50000	9.2	2.9	7.5	0.0	7.5	4.1	1.0

- Now, notice that the number of data row changed from 199 to 200 since the first row is now counted as a data row. And now we are using default columns names.
- The "\s+" matches one or more spaces. This is an example of a regex.
- We need to name the columns.

Algae_Blooms — load (3rd attempt) IV

```python
names = ('Season','Size','Speed','max_pH','min_O2','mean_Cl', 'mean_NO3', 'mean_NH4', 'mean_oPO4',
         'mean_PO4', 'mean_Chlor', 'a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7')

df = pd.read_table('src/Analysis.txt', sep='\s+', names=names)
print(df.shape)
df.head()
```

```
(200, 18)
```

	Season	Size	Speed	max_pH	min_O2	mean_Cl	mean_NO3	mean_NH4	mean_oPO4	mean_PO4	mean_Chlor	a1	a2	a3	a4	a5	a6
0	winter	small	medium	8.00000	9.80000	60.80000	6.23800	578.00000	105.00000	170.00000	50.00000	0.0	0.0	0.0	0.0	34.2	8.3
1	spring	small	medium	8.35000	8.00000	57.75000	1.28800	370.0	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
2	autumn	small	medium	8.10000	11.40000	40.02000	5.33000	346.6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
3	spring	small	medium	8.07000	4.80000	77.36400	2.30200	98.18	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]
4	autumn	small	medium	8.06000	9.00000	55.35000	10.41600	233.7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Season      200 non-null    object
 1   Size        200 non-null    object
 2   Speed       200 non-null    object
 3   max_pH      200 non-null    object
 4   min_O2      200 non-null    object
 5   mean_Cl     200 non-null    object
 6   mean_NO3    200 non-null    object
 7   mean_NH4    200 non-null    object
 8   mean_oPO4   200 non-null    object
 9   mean_PO4    200 non-null    object
 10  mean_Chlor  200 non-null    object
 11  a1          200 non-null    float64
 12  a2          200 non-null    float64
```

Dataframe looks a bit better, but why are numeric columns converted as `object`?
Reading `instructions.txt` we see that missing values are indicated by `XXXXXXX`.

`Algae_Blooms` — load (4th attempt)

```python
names = ('Season','Size','Speed','max_pH','min_O2','mean_Cl', 'mean_NO3', 'mean_NH4', 'mean_oPO4',
         'mean_PO4', 'mean_Chlor', 'a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7')

df = pd.read_table('src/Analysis.txt', sep='\s+', names=names, na_values='XXXXXXX')
print(df.shape)
df.head()
```

```
(200, 18)
```

	Season	Size	Speed	max_pH	min_O2	mean_Cl	mean_NO3	mean_NH4	mean_oPO4	mean_PO4	mean_Chlor	a1	a2	a3	a4	a5	a6	a
0	winter	small	medium	8.00	9.8	60.800	6.238	578.00000	105.000	170.00000	50.0	0.0	0.0	0.0	0.0	34.2	8.3	0
1	spring	small	medium	8.35	8.0	57.750	1.288	370.00										2
2	autumn	small	medium	8.10	11.4	40.020	5.330	346.66										9
3	spring	small	medium	8.07	4.8	77.364	2.302	98.182										1
4	autumn	small	medium	8.06	9.0	55.350	10.416	233.70										1

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 200 entries, 0 to 199
Data columns (total 18 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Season      200 non-null    object
 1   Size        200 non-null    object
 2   Speed       200 non-null    object
 3   max_pH      199 non-null    float64
 4   min_O2      198 non-null    float64
 5   mean_Cl     190 non-null    float64
 6   mean_NO3    198 non-null    float64
 7   mean_NH4    198 non-null    float64
 8   mean_oPO4   198 non-null    float64
 9   mean_PO4    198 non-null    float64
 10  mean_Chlor  188 non-null    float64
 11  a1          200 non-null    float64
```

Now some variables have missing values

Also we should convert `Season`, `Size` and `Speed` to category and ensure the levels are ordered.

Algae_Blooms — Fix Data Types II

The three categorical variables have levels with a natural order ⇒ convert to `category` and specify order:

```
df.Season = pd.Categorical(df.Season, categories=['spring', 'summer', 'autumn', 'winter'], ordered=Tr
print(df.Season.unique())
```

```
['winter', 'spring', 'autumn', 'summer']
Categories (4, object): ['spring' < 'summer' < 'autumn' < 'winter']
```

```
df.Size = pd.Categorical(df.Size, categories=['small', 'medium', 'large'], ordered=True)
print(df.Size.unique())
```

```
['small', 'medium', 'large']
Categories (3, object): ['small' < 'medium' < 'large']
```

```
df.Speed = pd.Categorical(df.Speed, categories=['low', 'medium', 'high'], ordered=True)
print(df.Speed.unique())
```

```
['medium', 'high', 'low']
Categories (3, object): ['low' < 'medium' < 'high']
```

`Algae_Blooms` — Identification of Missing Values (NA)

1 Which columns have missing values?

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

```
Season          0
Size            0
Speed           0
max_pH          1
min_O2          2
mean_Cl        10
mean_NO3        2
mean_NH4        2
mean_oPO4       2
mean_PO4        2
mean_Chlor     12
a1              0
a2              0
a3              0
a4              0
a5              0
a6              0
a7              0
dtype: int64
```

- Two columns (features) account for 22 NAs, but cannot just drop them as will lose a lot of information.
- Two rows (observations) account for 12 NAs $\Rightarrow$ remove.
- Removing other rows with a NA will result in a los of 14 rows (7% of the data), instead will impute later.

2 ~~Which rows have missing values?~~
How many NAs per row?

```
df.isna().sum(axis=1).value_counts()
```

```
0    184
1      7
2      7
6      2
dtype: int64
```

4 Rows / ~~Cols~~ to drop?

```
df.loc[df.isna().sum(axis=1)==6]
```

	Season	Size	Speed	max_pH	min_O2	mean_Cl	mean_NO3	mean_NH4	mean_oPO4	mean_PO4	mean_Chlor	a1	a2	a
61	summer	small	medium	6.4	NaN	NaN	NaN	NaN	NaN	14.0	NaN	19.4	0.0	0.
198	winter	large	medium	8.0	7.6	NaN	NaN	NaN	NaN	NaN	NaN	0.0	12.5	3.

```
df = df.loc[df.isna().sum(axis=1)<6].copy()
print(df.shape)
```

```
(198, 18)
```

After Loading and Initial Clean — Where are we? I

Tips

- ✔ Loaded data, corrected dtypes (categorical with order levels)
- ✔ Sanitised column names — not needed, but note column name `size` shadows pandas dataframe function `size` ⇒ so use `df["size"]` instead of `df.size`.
- ✔ No missing values

Titanic

- ✔ Loaded data — no conversion of dtypes needed . . . . (but if you don't plots/crosstab order won't agree)
- ✔ Sanitised column names — not needed,
- Missing values in `Age` (177/891=20%), `Cabin` (687/891=77%), and `Embarked` (2/891=0.2%).
  - A feature with 77% missing values should be considered for deletion, but what if the presence of a missing value actually tells us something? ⇒ convert to a boolean feature.

Algae Blooms

- ✔ Loaded data, corrected dtypes (categorical with ordered levels)
- ✔ Sanitised column names.
- Missing values
  - Removed two rows with 6 NA each, accounted for 12/33=36% of the missing values.
  - Remaining, 21 NAs are concentrated in `mean_CL` (8) and `mean_Chlor` (10). EDA will suggest options.

After Loading and Initial Clean — Where are we? II

Next we might

- Save result of initial clean:
  - To either a CSV (if we don't mind losing `dtype` metadata)

    ```
    df.to_csv('data/Analysis.csv', index=False)
    ```

  - To (say) pickle format (to keep `dtype` metadata)

    ```
    df.to_pickle('data/Analysis.pkl')
    ```

    Later can read dataframe back in using

    ```
    df = pd.read_pickle('data/Analysis.pkl')
    print(df.shape)
    df.head(1)
    ```

- If the dataset is large (>100K rows), save a (reproducible) sample of the dataset for later EDA to speed up calculations (especially visualisations).

  ```
  df.sample(frac=.25, random_state=42).to_pickle('data/Analysis_sample.pkl')
  ```

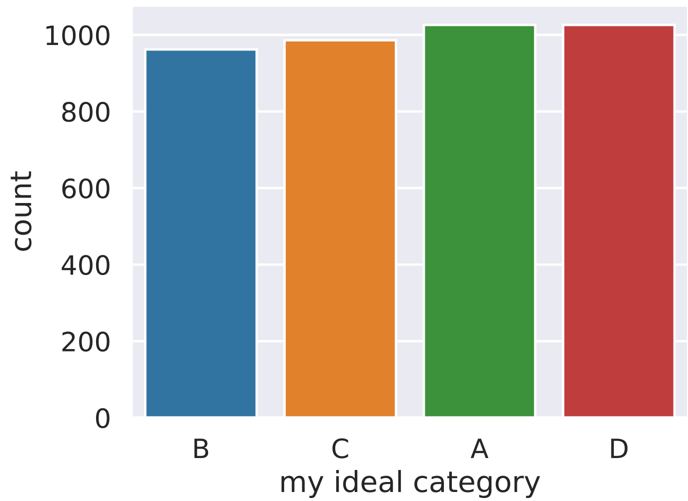
A Selection of Statistical Visualisations and Metrics

		Relationship to Target	
	Feature	Categorical	Numerical
Categorical	`nunique`, `unique`, `describe`, `value_counts`, ... `countplot`, ...	`crosstab`, ... `countplot`, ...	`boxplot`, ... `catplot`, `boxplot`, ...
Numerical	`describe`, ... `histplot`, `boxplot`, `displot`, `qqplot`, ...	`groupby+describe`, ... `catplot`, `boxplot`, ...	correlations, ... `lmplot`, ...

Categorical Variables

The Ideal

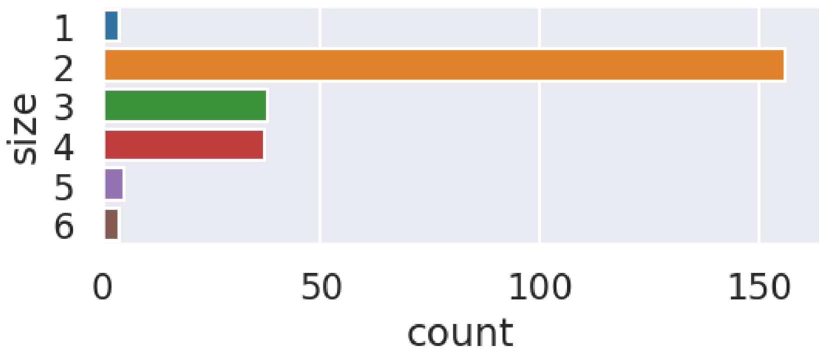
- Each level equally likely.
- Not too many levels: 2–12(ish).



Reality

Tips

```
sns.countplot(y="size", data=df);
```



- If `size` was the target, then most models will train towards the majority class (`size=2`).
- If `size` was a feature, then quality of predictor could vary greatly depending on the feature categorical level.
- Consider merge/drop rare category levels.

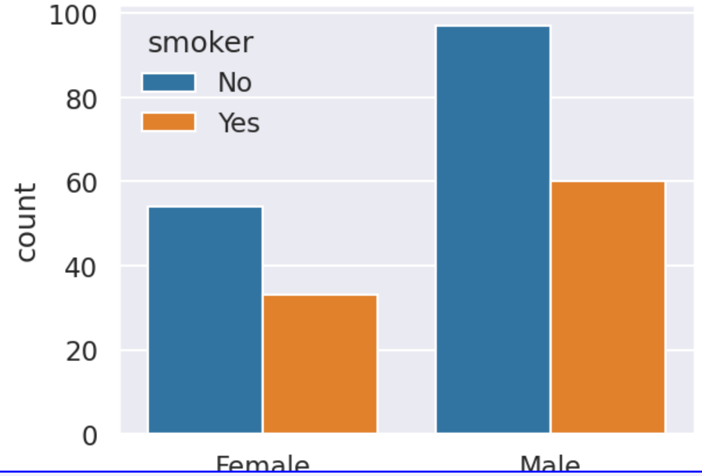
Tools

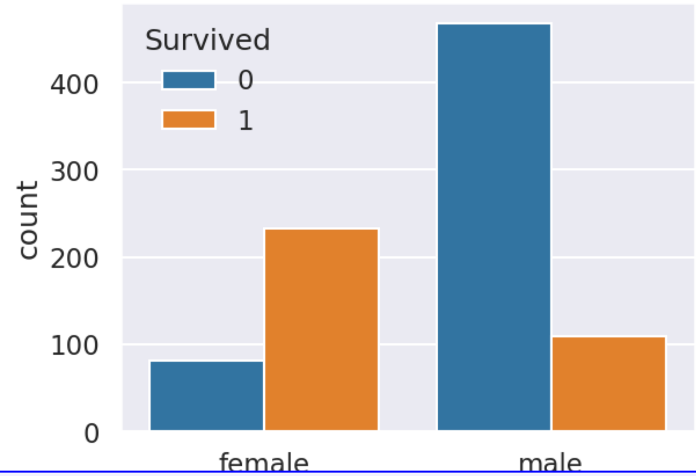
- `nunique`, `unique`, `value_counts`.
- `sns.countplot` — shows the counts of observations in each categorical level using bars.

Categorical Variables — Relationship with (Categorical) Target

feature   target   Titanic

```
sns.countplot(x="Sex", hue="Survived", data=df);
```






```
pd.crosstab(columns=df.Sex, index=df.Survived, margins=True, normalize='columns') \
    .style.format("{:.2%}").background_gradient(cmap='summer_r')
```

sex	Female	Male	All
smoker			
No	62.07%	61.78%	61.89%
Yes	37.93%	38.22%	38.11%

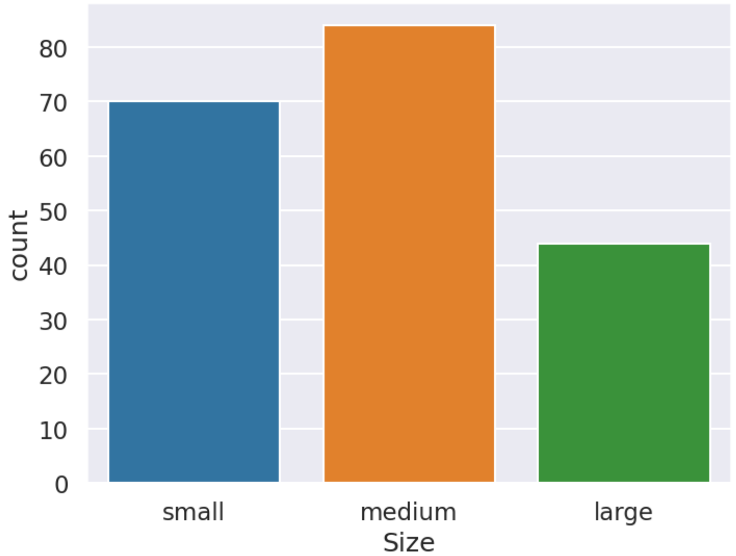
No relationship between `sex` and `smoker`

Sex	female	male	All
Survived			
0	25.80%	81.11%	61.62%
1	74.20%	18.89%	38.38%

Strong relationship between `Sex` and `Survived`

Categorical Variables — Relationship with (Numerical) Target I

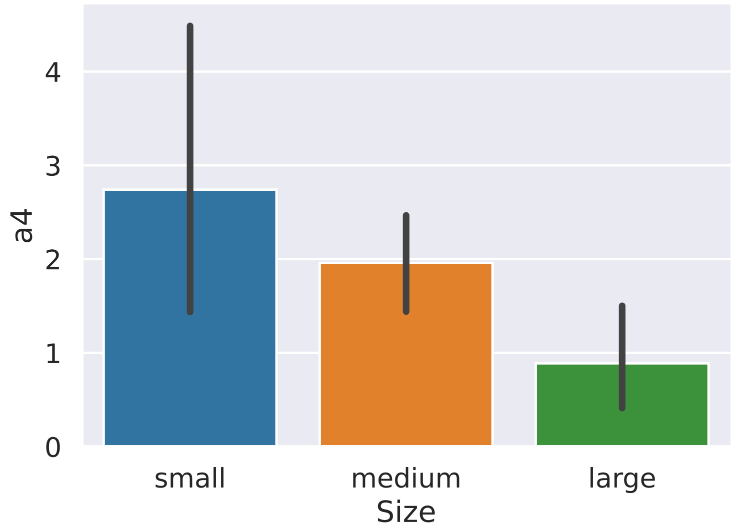
```
sns.countplot(x="Size", data=df);
```



- Shows the counts of observations in each categorical level using bar (height/width).

Is it usable?

```
sns.catplot(x="Size", y="a4", data=df, kind='bar');
```



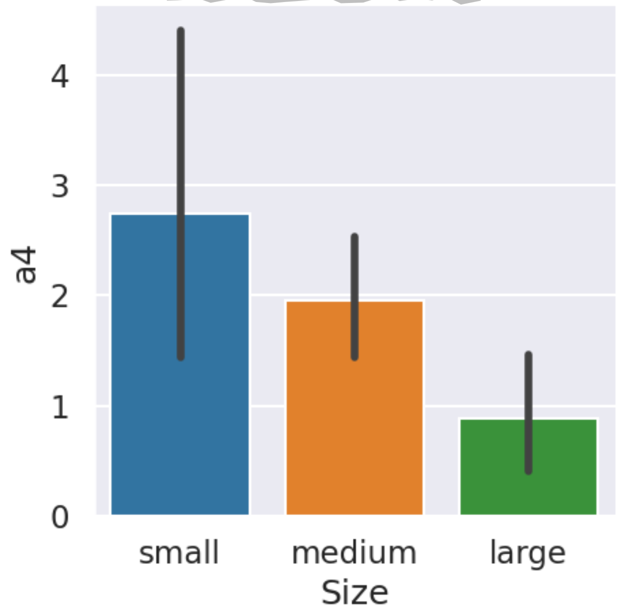
- Shows the average level (mean) and uncertainty (std) of the numerical target (`a4`) in each categorical level of the categorical variable.
- Vertical bar shows 95% confidence interval.

Is it useful?

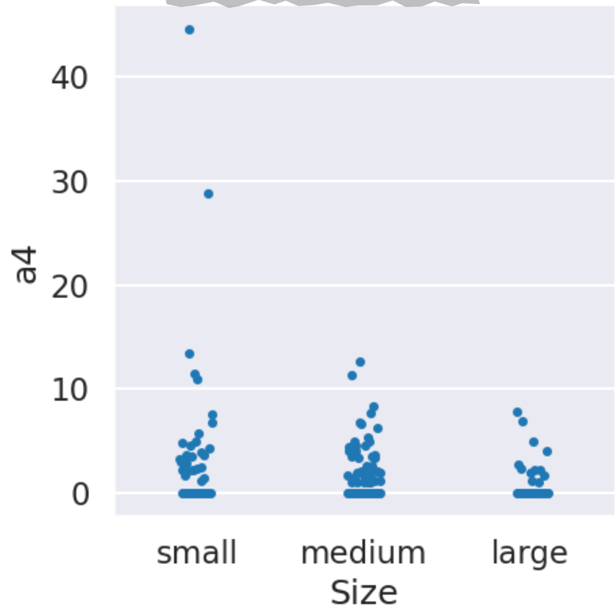
Categorical Variables — Relationship with (Numerical) Target II

The option `kind` in `catplot` can be:

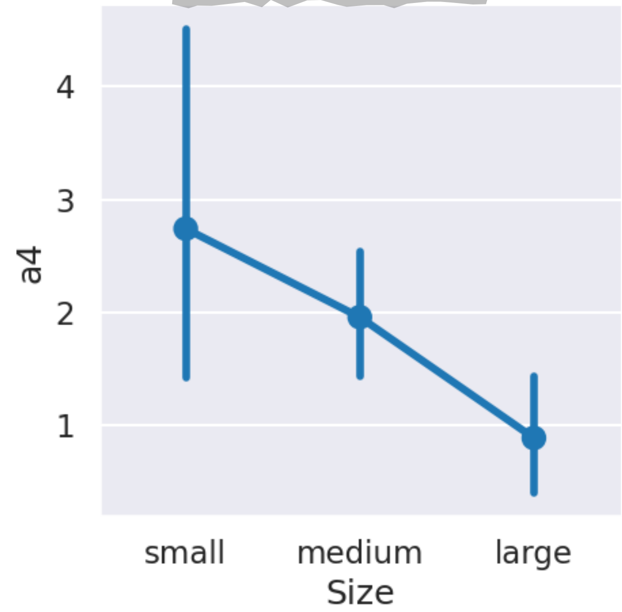
`kind="bar"`



`kind="strip"`

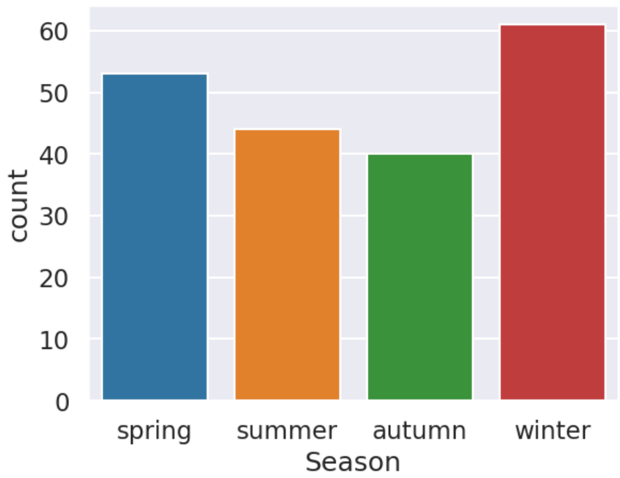


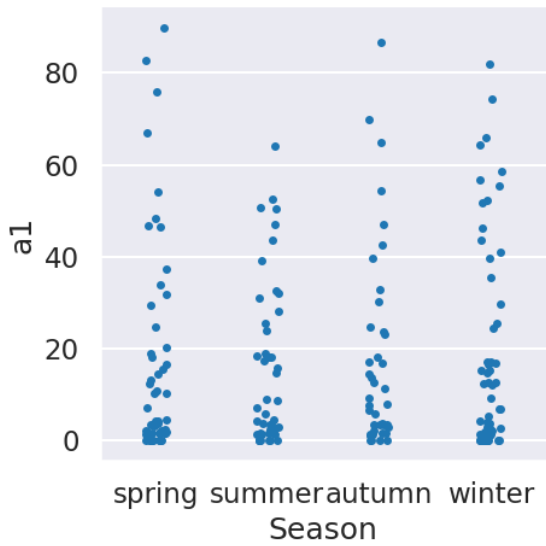
`kind="point"`

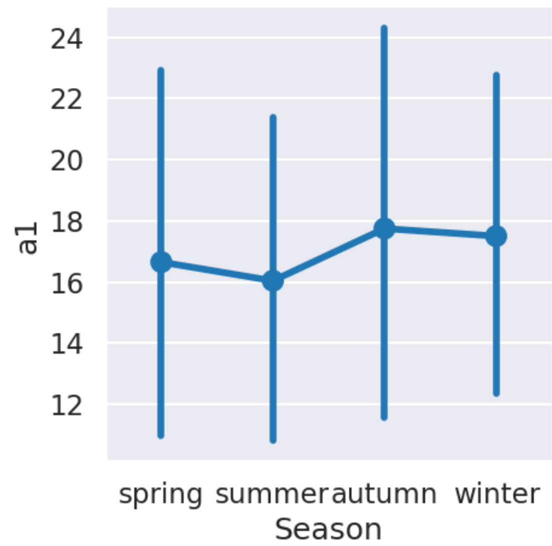


- `bar` and `point` show essentially the same information, but `point` is more compact when comparing multiple categorical features to a continuous target on the same plot.
- `strip` shows individual observations — useful (as in this case) to show that the larger uncertainty in `Size="small"` observations is mainly due to two outliers.

Example — Dataset: `Algae Blooms`, Feature: `Season`, Target: `a1`







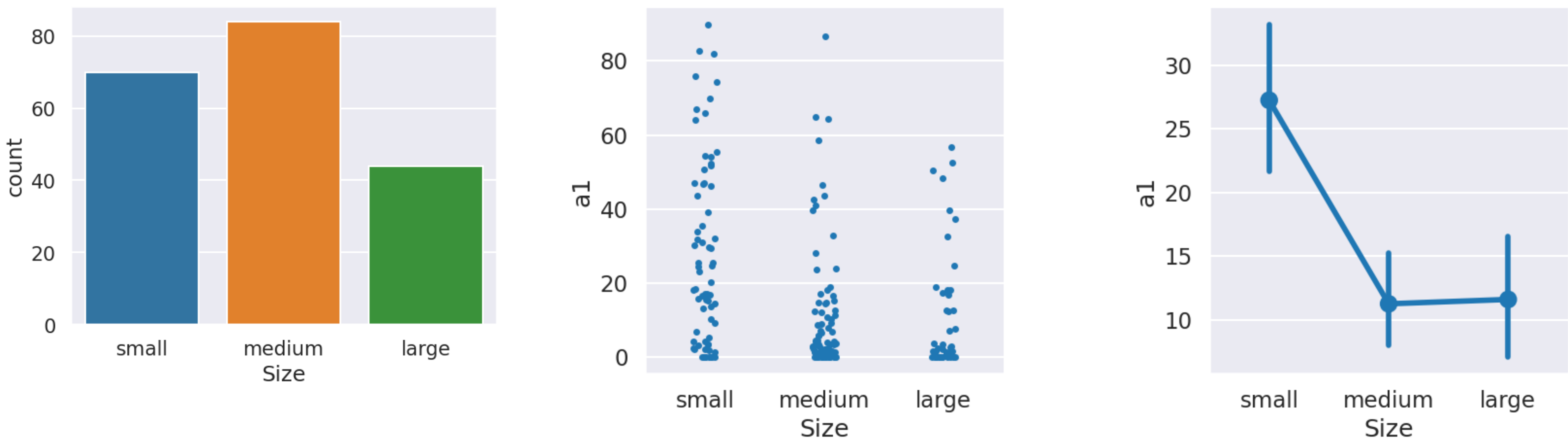

```
df.groupby("Season")["a1"].agg(["mean","count","std"])
```

	mean	count	std
Season	$\bar{x}$	n	σ
spring	16.649057	53	23.093786
summer	16.038636	44	17.920798
autumn	17.745000	40	21.611203
winter	17.498361	61	22.568256

- Countplot shows no issues with feature `Season` — all levels approximately equally represented.
- Catplots show slightly less spread in `a1` for `Season="summer"` observations. (`strip` shows smaller range, `point` shows smaller standard deviation).

⇒ Mean levels of `a1` for different levels of `Season` are well within the 95% confidence intervals ($\bar{x} \pm \sigma 1.96/\sqrt{n}$), so no/weak relationship between categorical feature and numerical target.

Example — Dataset: `Algae Blooms`, Feature: `Size`, Target: `a1`




```
df.groupby("Size")["a1"].agg(["min","max","mean","count","std"])
```

	min	max	mean	count	std
Size			$\bar{x}$	n	σ
small	0.0	89.8	27.255714	70	24.895426
medium	0.0	86.6	11.267857	84	17.163124
large	0.0	56.8	11.611364	44	16.556123

- Countplot shows no issues with feature `Size`.
- Catplot (`point`) shows that levels of `a1` are higher for `Size="small"` observations.

⇒ Confidence interval for `Size="small"` observations do not overlap with CI for other levels, so significant relationship between categorical feature and numerical target.

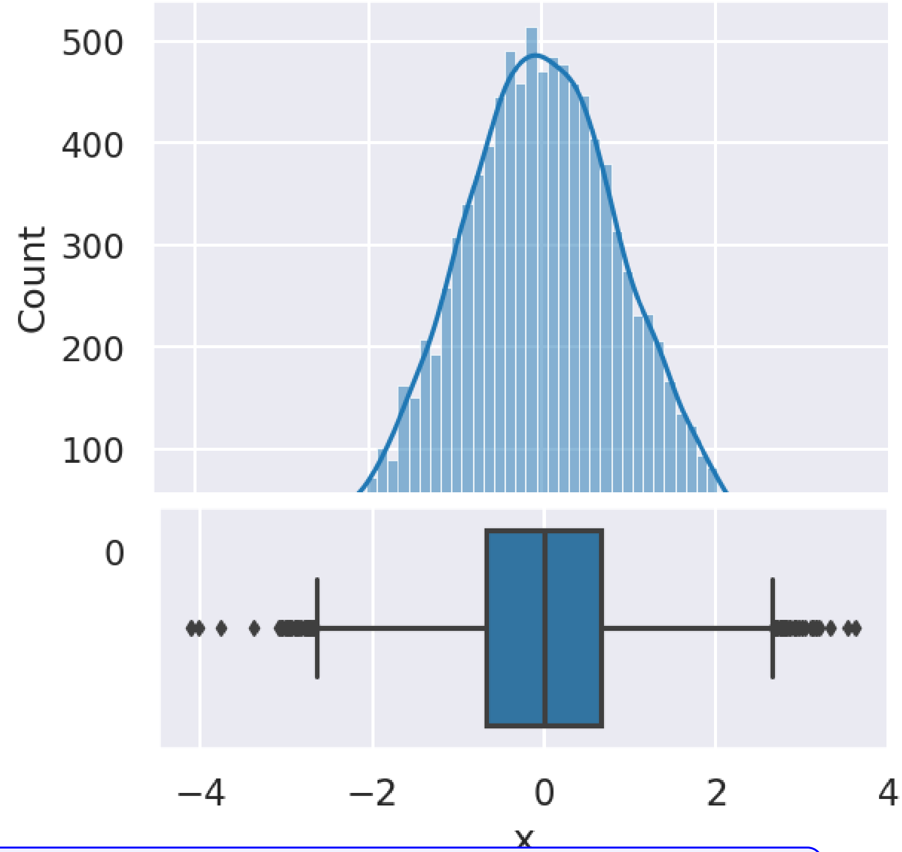
Numerical Variables

Things here are more complicated as a numerical variable could follow many different distributions. Here we look at data following the standard normal distribution. To start we generate 10,000 values and put in to new DataFrame, `df2`.

```
rv = stats.norm()
data = rv.rvs(size=10_000)
df2 = pd.DataFrame(data, columns=["x"])
df2.head(5)
```

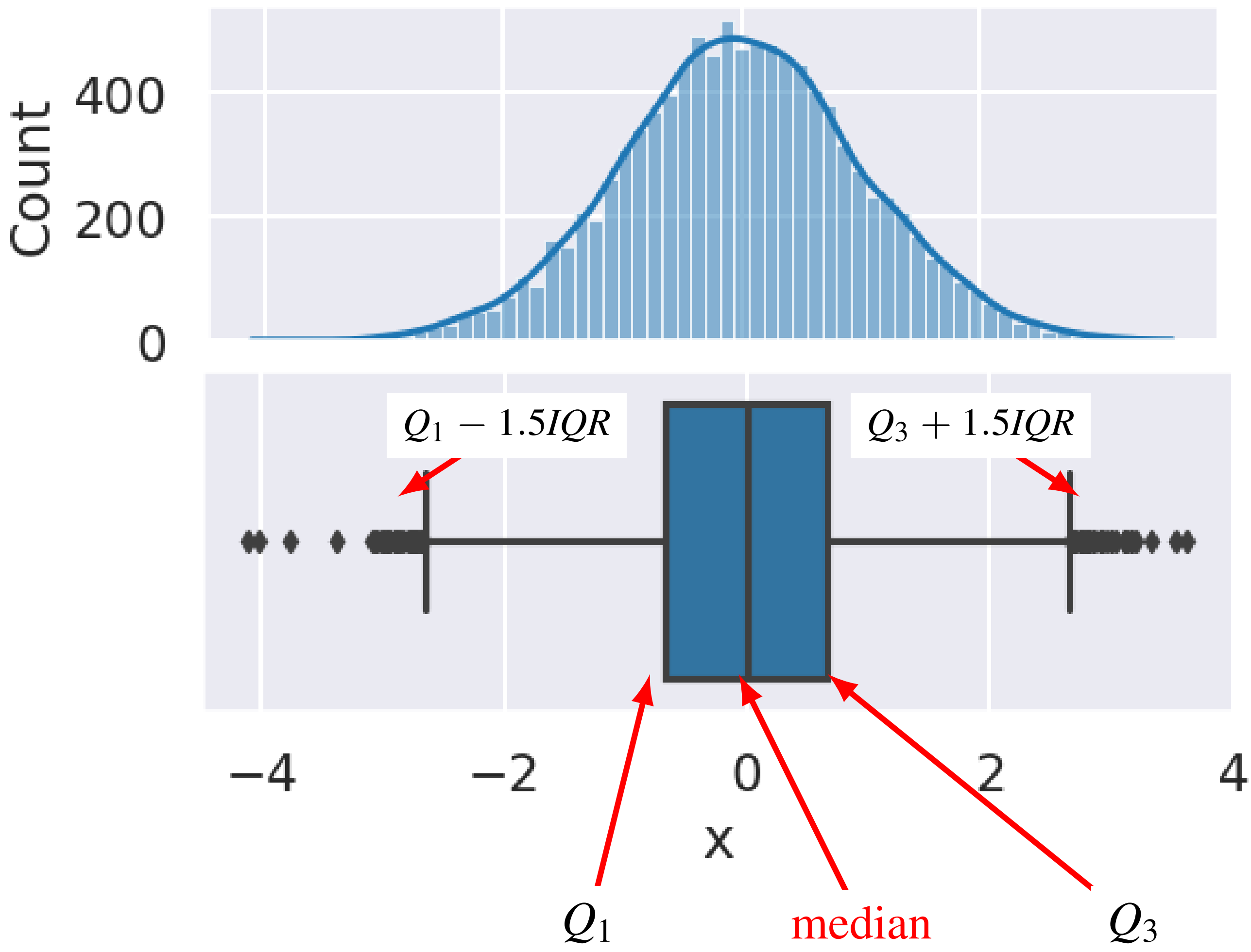
	x
0	0.184386
1	1.033751
2	-1.350789
3	0.326674
4	-1.942126

```
sns.histplot(x="x", data=df2, kde=True);
```




```
sns.boxplot(x="x", data=df2);
```

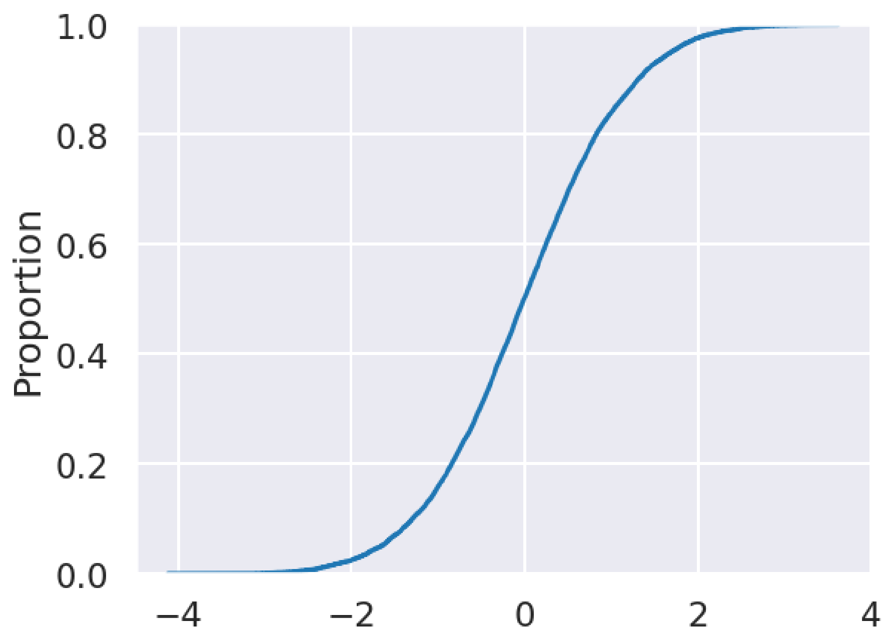
Histplot (Histogram) and Boxplot



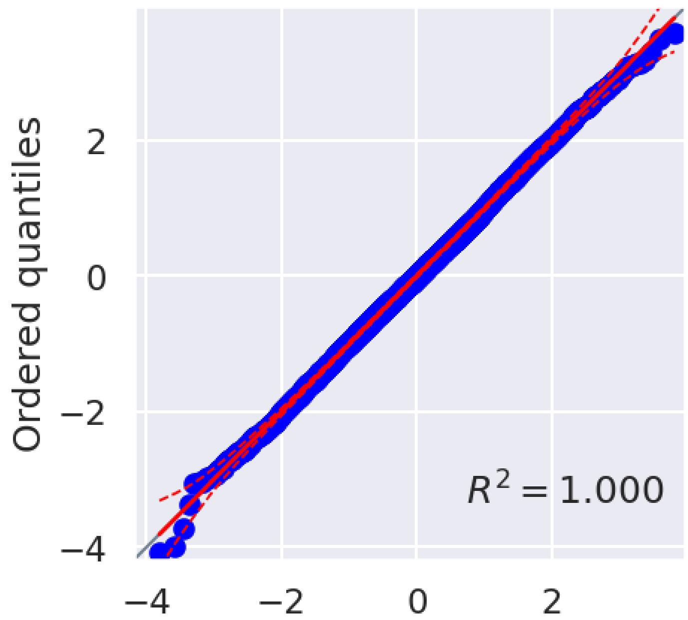
- Histogram is useful in depicting location, spread and shape.
- Curve, is estimate of shape given infinite data and infinite number of bins.
- Boxplots also depicts location, spread and shape, but uses median for estimate of centre, and quartiles for spread.
- Half the data is within the box, data points outside the whiskers (lines) are possible outliers, denoted by circles.

Cumulative Plot and QQ-Plot

```
sns.ecdfplot(data=df2, x="x");
```

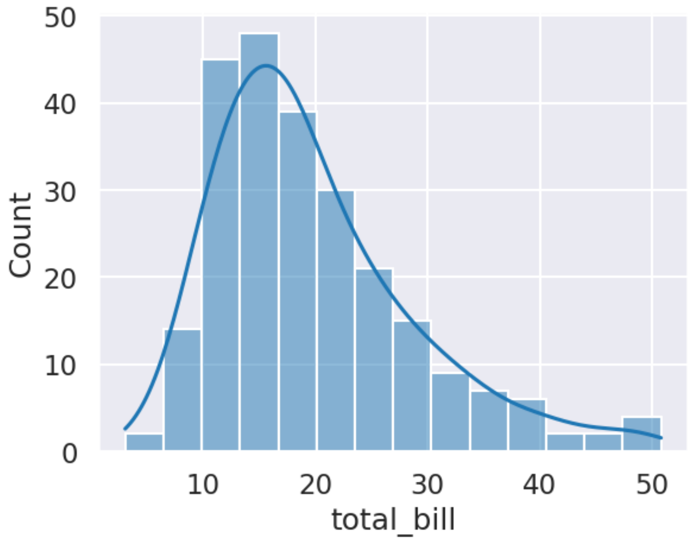


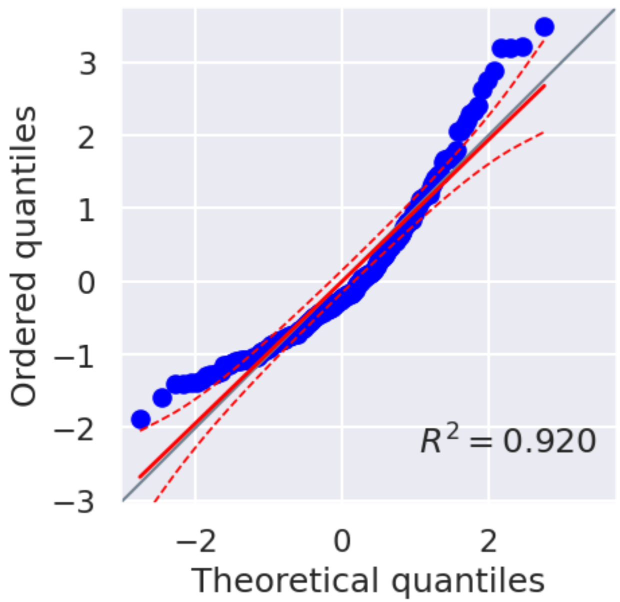

```
import pingouin as pg
pg.qqplot(df2.x);
```



- Represents the proportion of observations less than or equal to given value.
- Plot of observed quantiles against theoretical (assuming normal) quantiles. If both sets of quantiles came from the same distribution, we should see the points forming a line that's roughly straight.

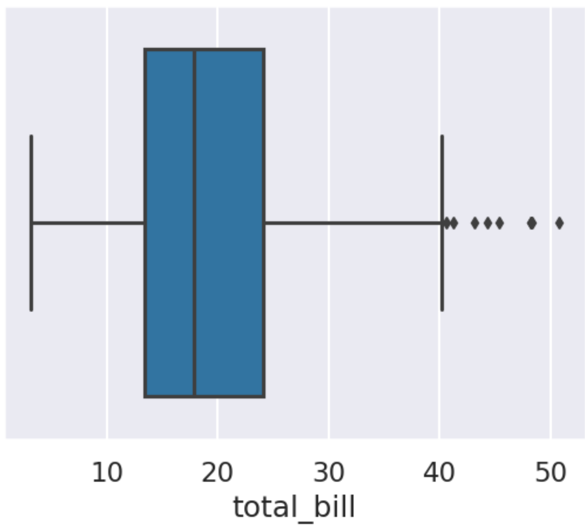
Example — Dataset: `Tips`, Feature: `total_bill`





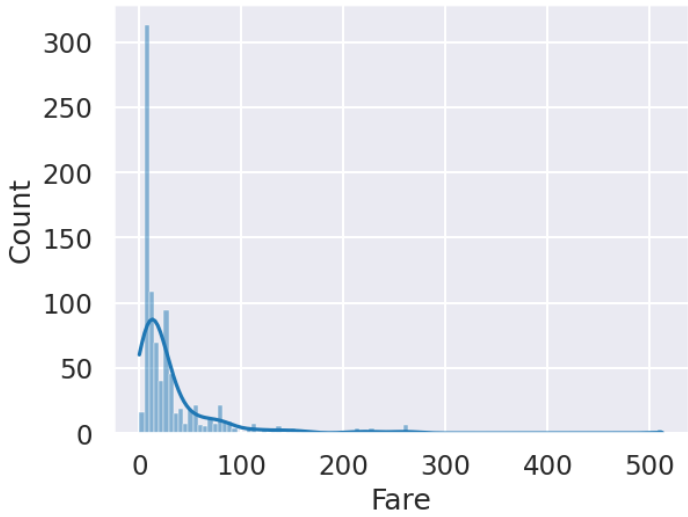

```
df.total_bill.describe()
```

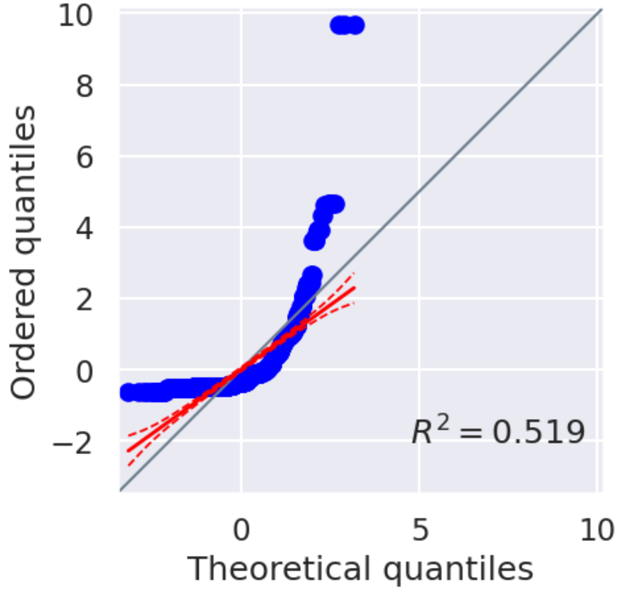
```
count    244.000000
mean      19.785943
std        8.902412
min        3.070000
25%       13.347500
50%       17.795000
75%       24.127500
max       50.810000
Name: total_bill, dtype: float
```



- Data is bell curve shaped, but right skewed (data is more spread out to the right).
- Outliners to the right.
- QQ-Plot indicate that data is not normal, but we could transform it to be more closer to normal.

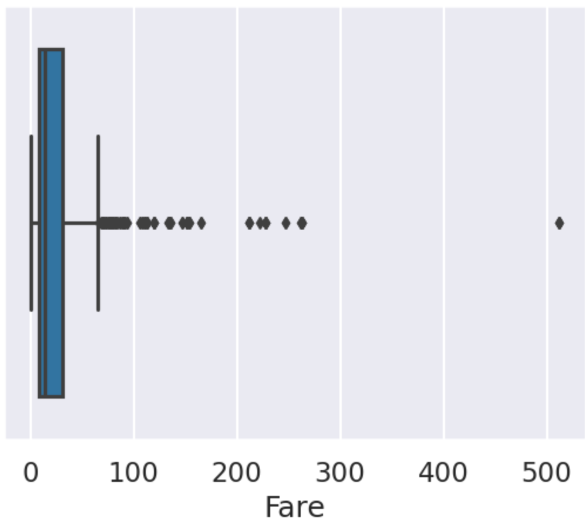
Example — Dataset: `Titanic`, Feature: `Fare`






```
df.Fare.describe()
```

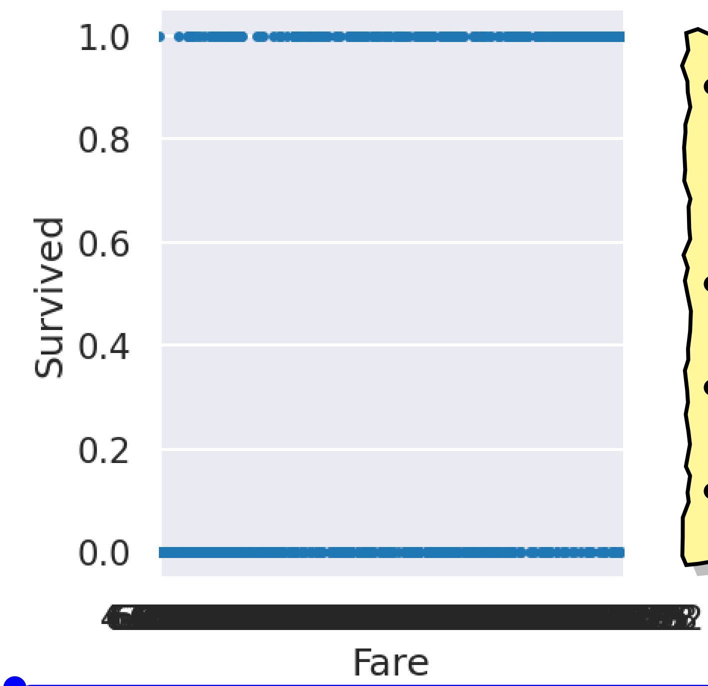
```
count    891.000000
mean      32.204208
std       49.693429
min        0.000000
25%        7.910400
50%       14.454200
75%       31.000000
max      512.329200
Name: Fare, dtype: float64
```



- This variable is more skewed and dominated by its outliers which need to be resolved.

Warning —- Plot Output Depends on Data Assumptions

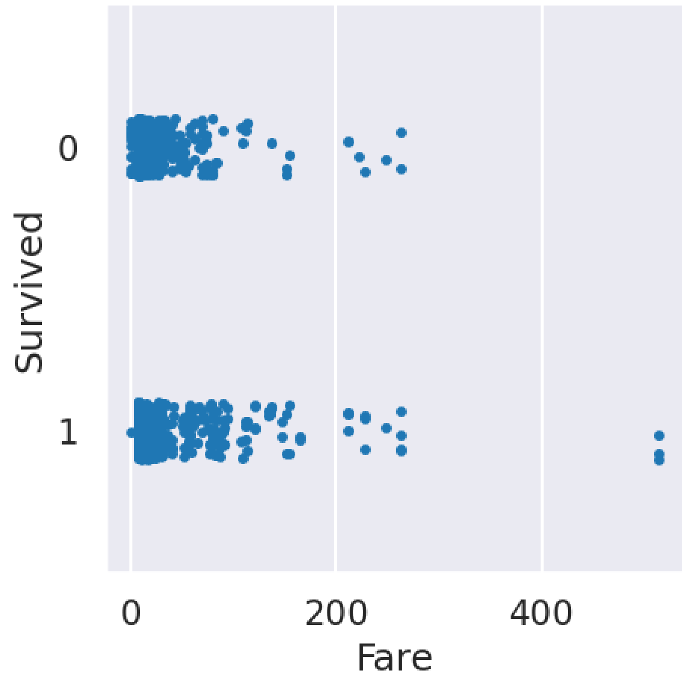
```
df = pd.read_csv("data/train.csv")
sns.catplot(data=df, x="Fare", y="Survived");
```




```
df = pd.read_csv("data/train.csv")
df.Survived = pd.Categorical(df.Survived)
sns.catplot(data=df, x="Fare", y="Survived");
```

- `seaborn` tries to infer the correct graph based on the data values/type, but it does not always get it correct.
- `Survived` stores 0 and 1 and has dtype `int`.
- Converting to a `Categorical` with numeric levels is not enough.
- `astype(str)` converts `0` and `1` to `"0"` and `"1"`.

```
df = pd.read_csv("data/train.csv")
df.Survived = df.Survived.astype(str)
sns.catplot(data=df, x="Fare", y="Survived");
```



Summary

- After reading in the data, exploratory data analysis begins
- Pass 1 is all about assessing the structure and cleanliness of the data
  1. Are the column names descriptive and short, or do we need to rename them?
  2. What datatype is each column - are there any surprises there?
  3. How are missing values handled, and can we standardise this?
- Passes 2 and 3 will examine the data more closely, in a repeatable fashion
- Pandas and seaborn offer easy-to-use ways of visualising columns, noting
  1. their datatype
  2. their cardinality
  3. the visualisation objective: observe distributions or relationships