# MSc - Data Mining

## Topic 03 : Exploratory Data Analysis

## Part 01 : Exploratory Data Analysis

Dr Bernard Butler and Dr Kieran Murphy

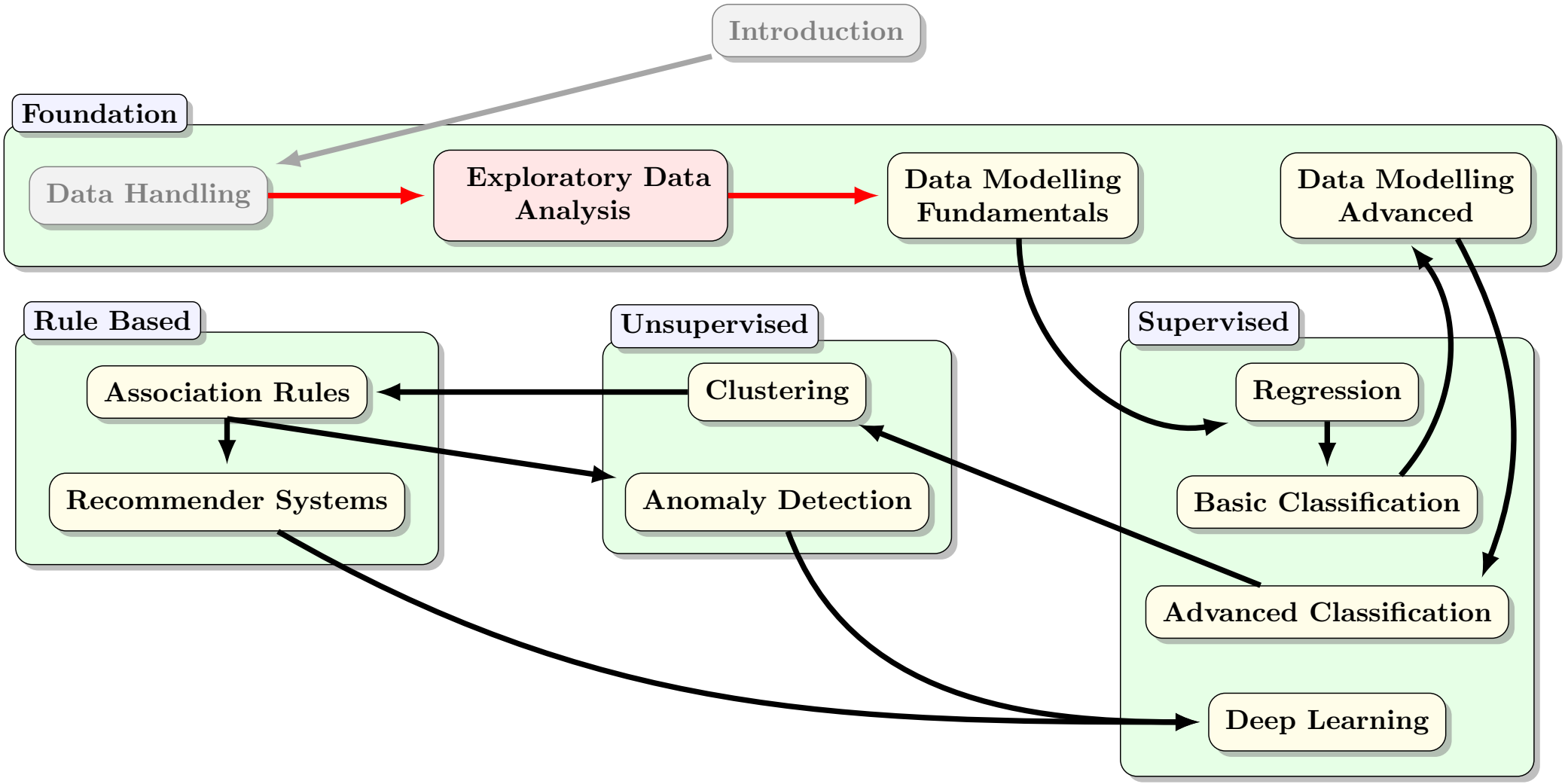
Department of Computing and Mathematics, WIT.
(bbutler@tssg.org and kmurphy@wit.ie)

Spring Semester, 2021

### Outline

- EDA Process
- Datasets = Tips, Titanic and Algae Blooms
- Identifying and resolving issues (missing value, outliers)
- Generating ToDo list for Feature Engineering/Transformation/Selection

# Data Mining (Week 3)



# Exploratory Data Analysis — Summary

# Introduction

# Exploratory Data Analysis (EDA)

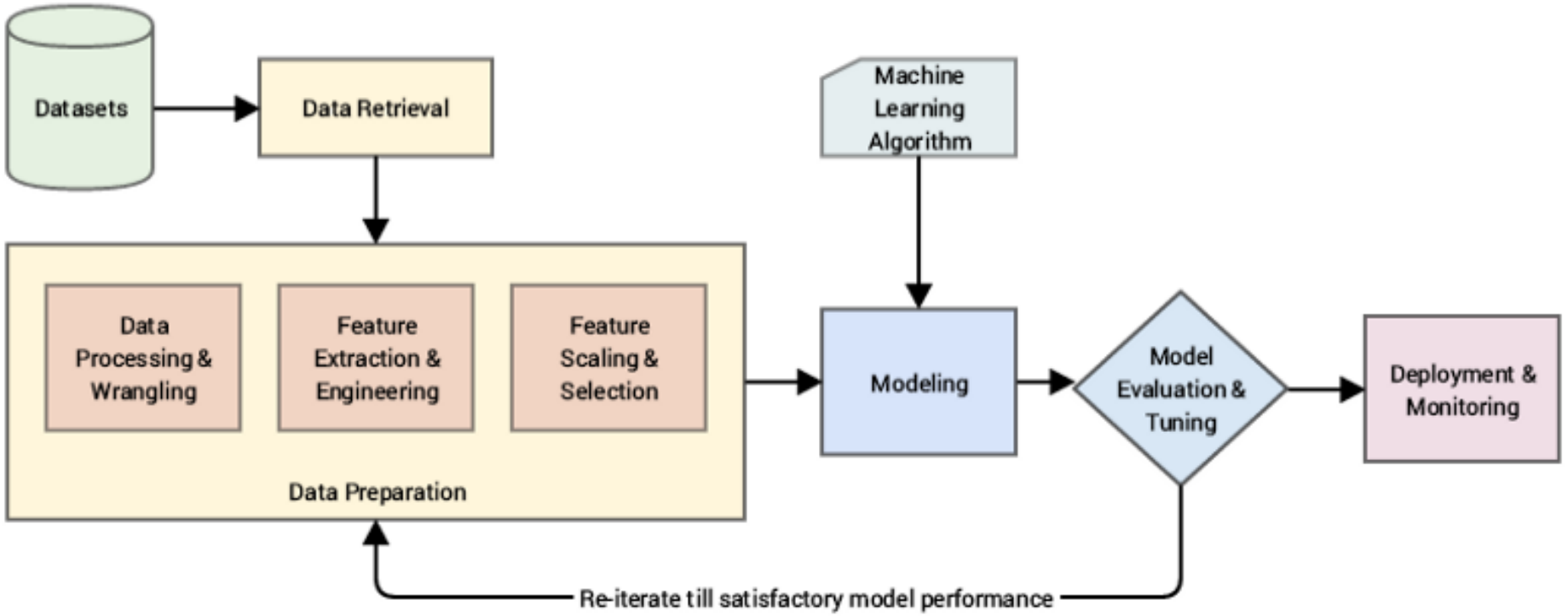
## Aim

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



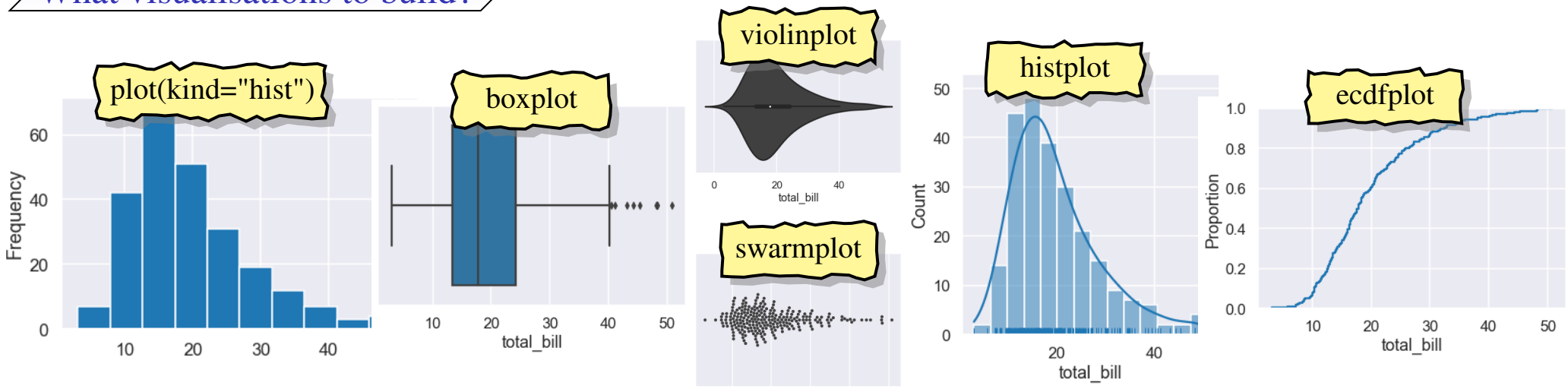

- Data preparation is the core of the data mining pipeline (typical estimates >50% of the time/effort).
- EDA is the data processing and wrangling.
- EDA informs the feature extraction, engineering, transformation and selection.

# The Bad News — 'The curse of choice'

## What questions to ask?

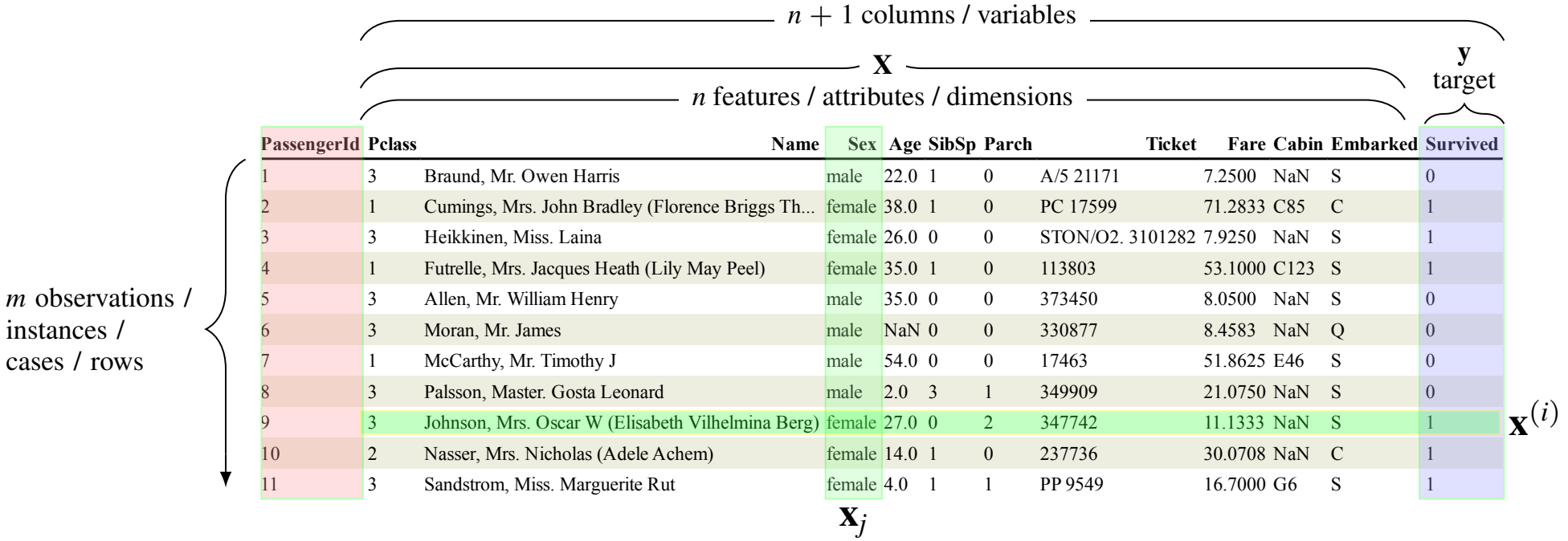
Dataset global questions: How many features? How many observations? What is the data type of each feature? Any null values? ... Feature specific questions: What is the distribution of each variable? Do there appear to be outliers? What features are related? ... Missing value questions: Are null value 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




| 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 |

- 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 drowned 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 | a6 | a7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 | 8.3 | 0.0 |
| **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 | 0.0 | 2.1 |
| **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 | 0.0 | 9.7 |
| **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 | 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 | 4.1 | 1.0 |
| **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 | 12.6 | 2.9 |
| **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 | 6.8 | 0.0 |
| **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 | 8.7 | 0.0 |
| **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 | 0.0 | 0.0 |
| **9** | winter | sm | | | | | | | | | | | | | | | 0.0 | 0.0 | 1.7 |
| **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 | 0.0 | 6.0 |
| **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 | 0.0 | 1.5 |
| **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 | 0.0 | 2.1 |
| **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 | 0.0 | 4.1 |
| **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 | 0.0 | 0.0 |
| **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 | 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 | 0.0 | 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 | 1.4 | 7.2 |
| **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 | 8.2 | 2.2 |
| **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 | 0.0 | 1.4 |

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 | 1 | | | | | | | | | C103 | S |
| **12** | 13 | 0 | | | | | | | | | 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 |
| **17** | 18 | 1 | 2 | Williams, Mr. Charles Eugene | male | NaN | 0 | 0 | 244373 | 13.0000 | NaN | S |
| **18** | 19 | 0 | 3 | Vander Planke, Mrs. Julius (Emelia Maria Vande... | female | 31.0 | 1 | 0 | 345763 | 18.0000 | 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 |

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-darkgrid")
```

# Before we start . . . auto EDA using `pandas-profiling`

```
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]

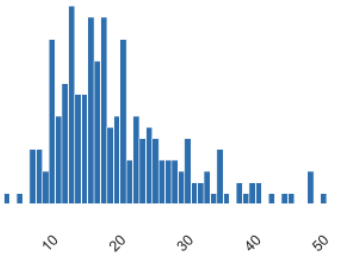
`pandas-profiling` is nice, but see how slow it is on this tiny dataset. What would happen if we had 100K rows x 100 columns?

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

## Variables

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

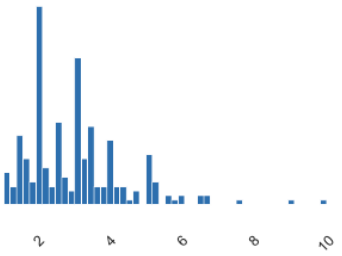
| | | | |
|---|---|---|---|
| Distinct | 229 | Mean | 19.78594262 |
| Distinct (%) | 93.9% | Minimum | 3.07 |
| Missing | 0 | Maximum | 50.81 |
| Missing (%) | 0.0% | Zeros | 0 |
| Infinite | 0 | Zeros (%) | 0.0% |
| Infinite (%) | 0.0% | Memory size | 1.9 KiB |



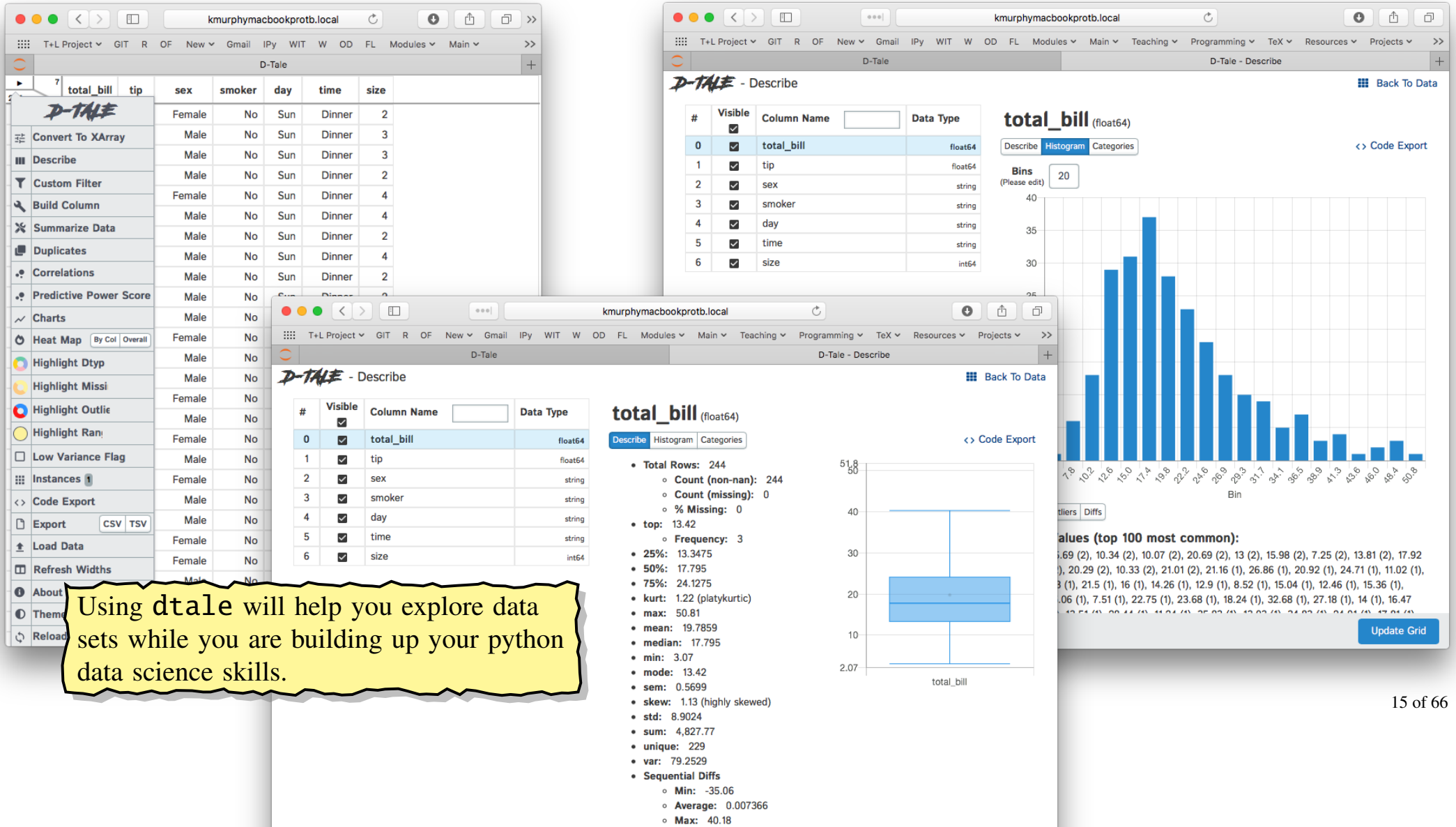

Toggle details

**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`




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("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

```
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: 7.3 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.

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

| | PassengerId | Survived | Pclass | Name | Sex | Age | SibSp | Parch | Ticket | Fare | Cabin | Embark |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 |

- 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.

```
df.info()
```

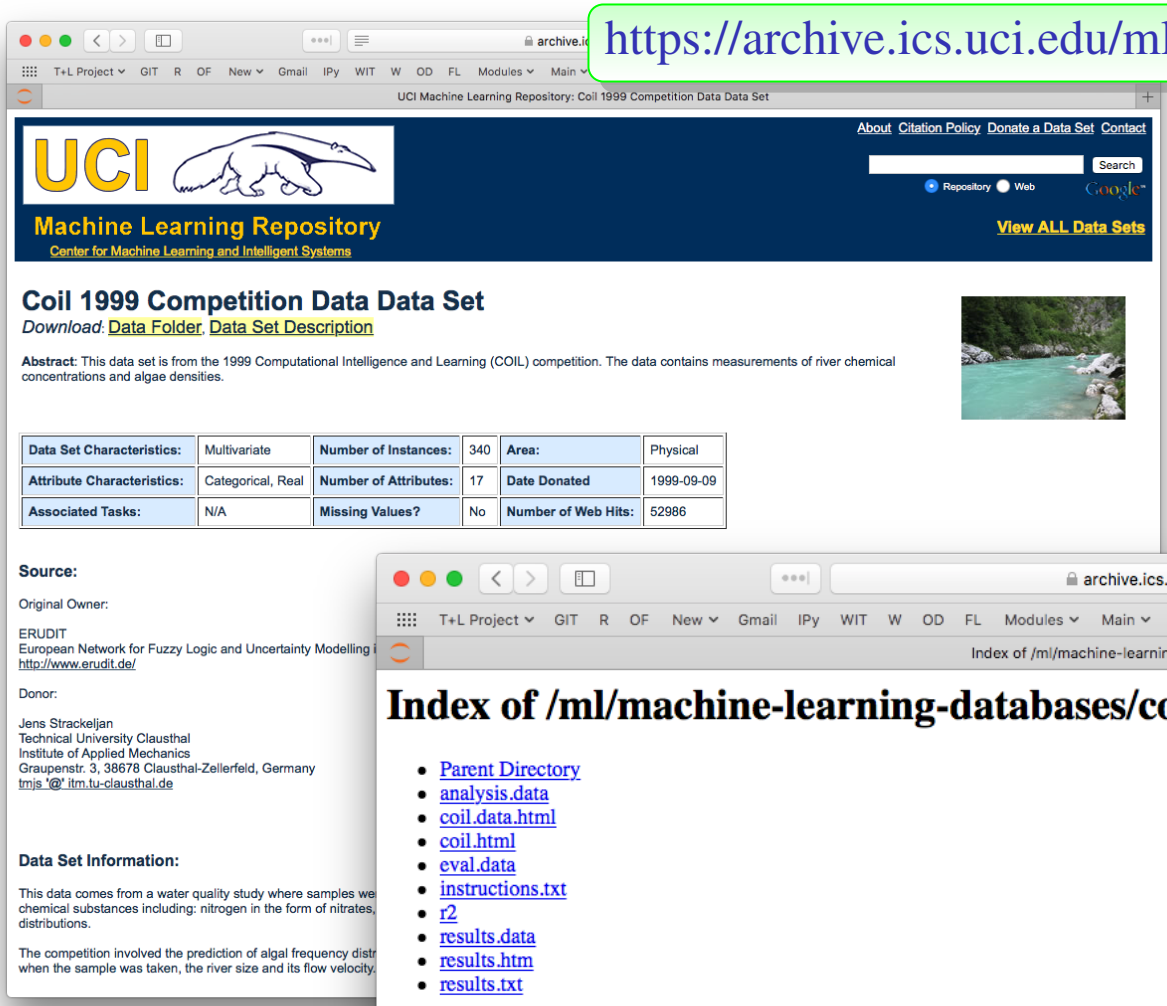
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  891 non-null    int64
 1   Survived     891 non-null    int64
 2   Pclass       891 non-null    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
```

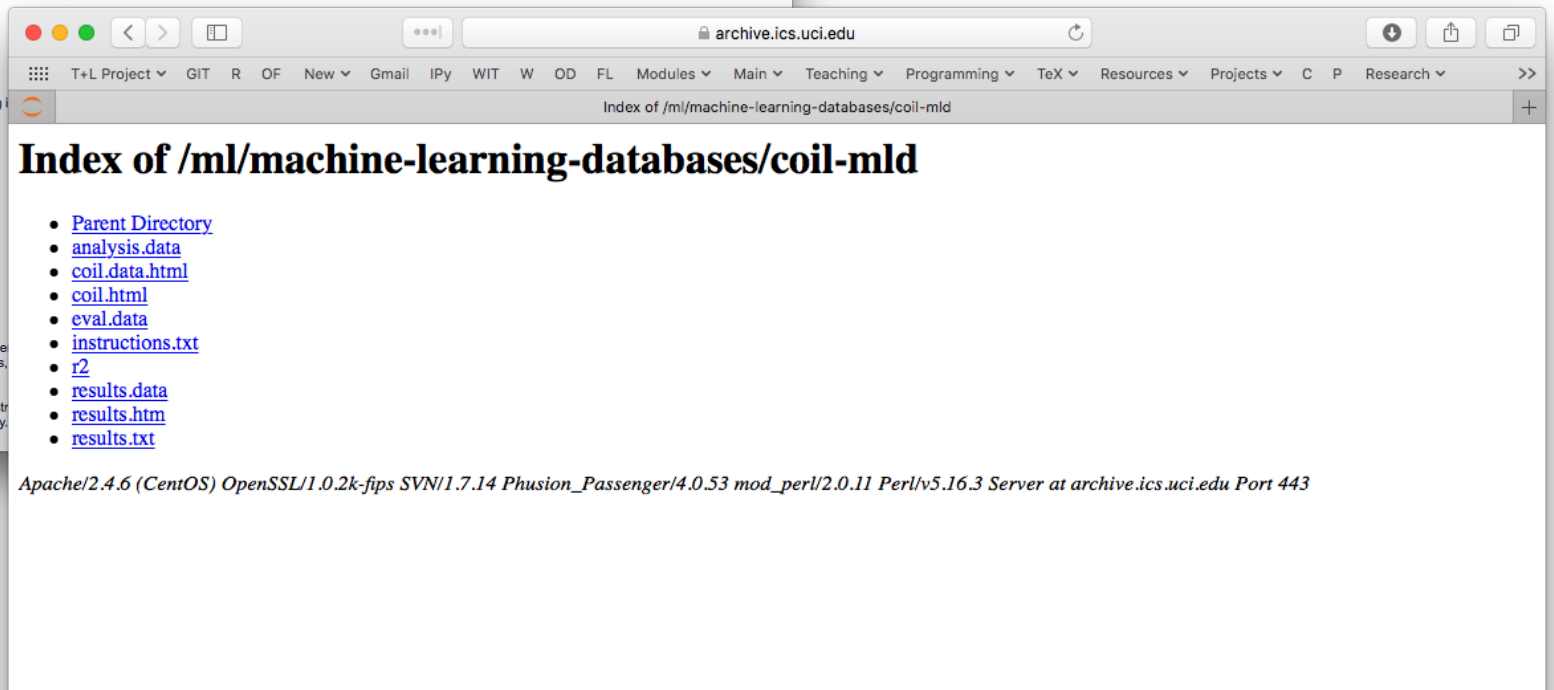
# `Algae_Blooms` – load



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

```
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 | a7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 370.0000 | | | | | | | | | | 1 |
| 2 | autumn | small | medium | 8.10000 | 11.40000 | 40.02000 | 5.33000 | 346.6669 | | | | | | | | | | 7 |
| 3 | spring | small | medium | 8.07000 | 4.80000 | 77.36400 | 2.30200 | 98.18200 | | | | | | | | | | 4 |
| 4 | autumn | small | medium | 8.06000 | 9.00000 | 55.35000 | 10.41600 | 233.7000 | | | | | | | | | | 0 |

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`.

```
<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
```

# Algae_Blooms — load (4th attempt)



```
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 | a7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 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 | 0.0 |
| 1 | spring | small | medium | 8.35 | 8.0 | 57.750 | 1.288 | 370.0000 | | | | | | | | | | 1 |
| 2 | autumn | small | medium | 8.10 | 11.4 | 40.020 | 5.330 | 346.6669 | | | | | | | | | | 7 |
| 3 | spring | small | medium | 8.07 | 4.8 | 77.364 | 2.302 | 98.18200 | | | | | | | | | | 4 |
| 4 | autumn | small | medium | 8.06 | 9.0 | 55.350 | 10.416 | 233.7000 | | | | | | | | | | 0 |

```
<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 ⇒ 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
2      7
1      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 | a3 | a4 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **61** | summer | small | medium | 6.4 | NaN | NaN | NaN | NaN | NaN | 14.0 | NaN | 19.4 | 0.0 | 0.0 | 2.0 |
| **198** | winter | large | medium | 8.0 | 7.6 | NaN | NaN | NaN | NaN | NaN | NaN | 0.0 | 12.5 | 3.7 | 1.0 |

```
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')
```

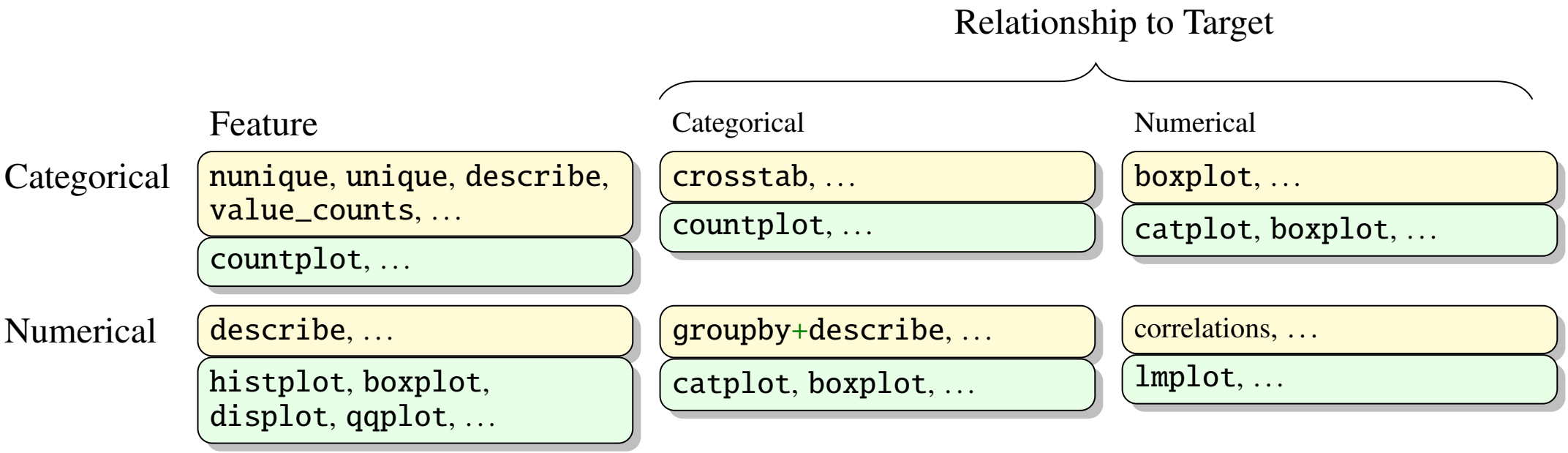
    Later can read dataframe back in using

```
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')
```

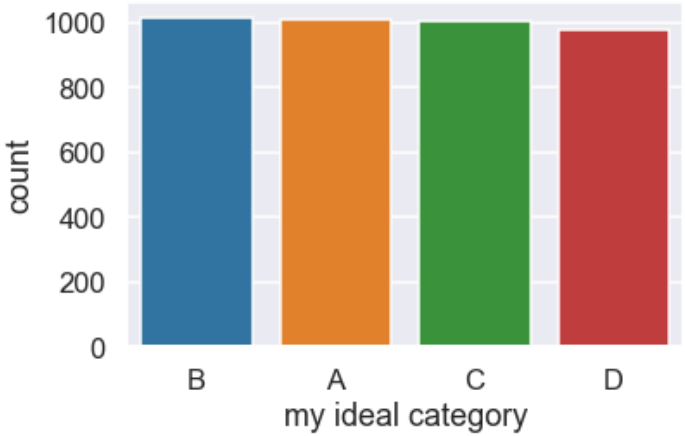
# A Selection of Statistical Visualisations and Metrics



# Categorical Variables

## The Ideal

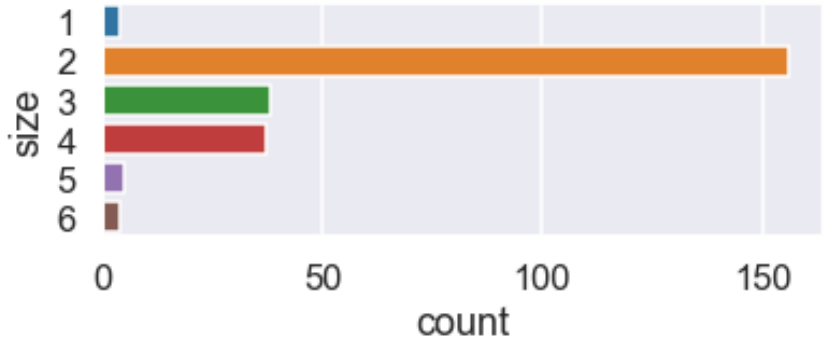
- Each level equally likely.
- Not too many levels: 2–12(ish).




## 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.

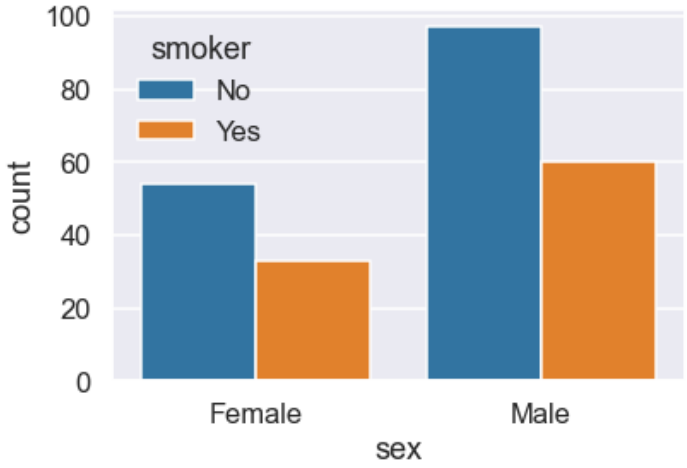
# Categorical Variables — Relationship with (Categorical) Target

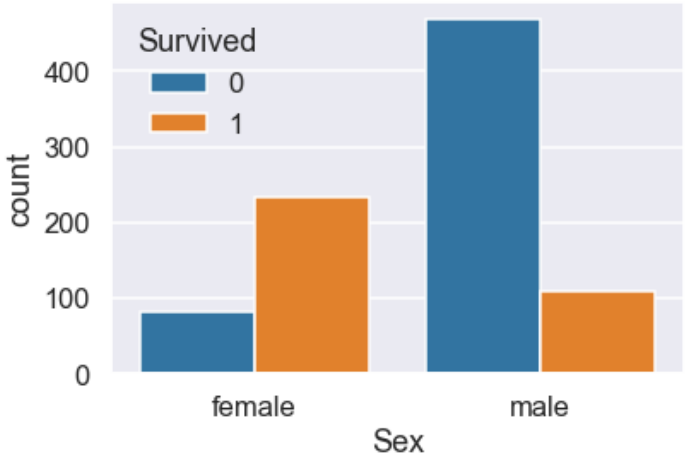
feature

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% |

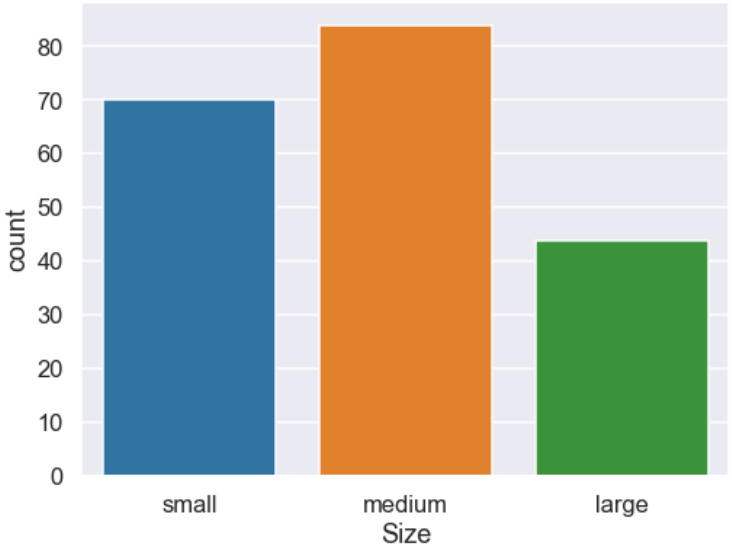
No relationship between `sex` and `smoker`

| 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

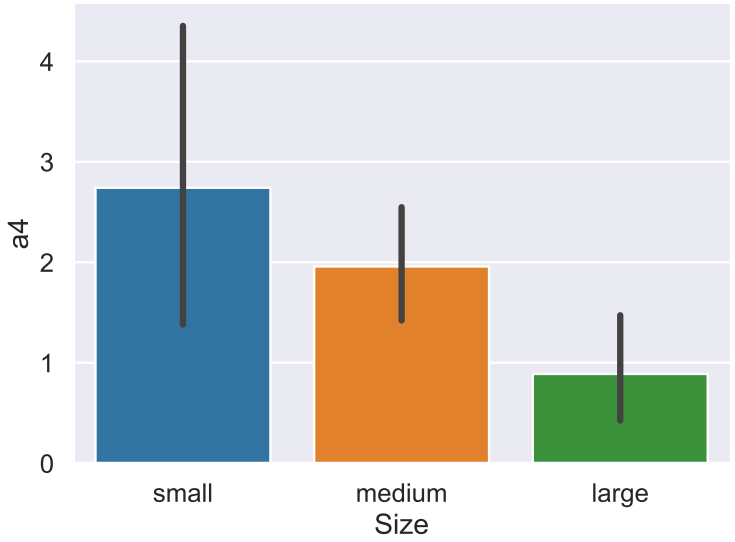
```
sns.countplot(x="Size", data=df);
```




- 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');
```

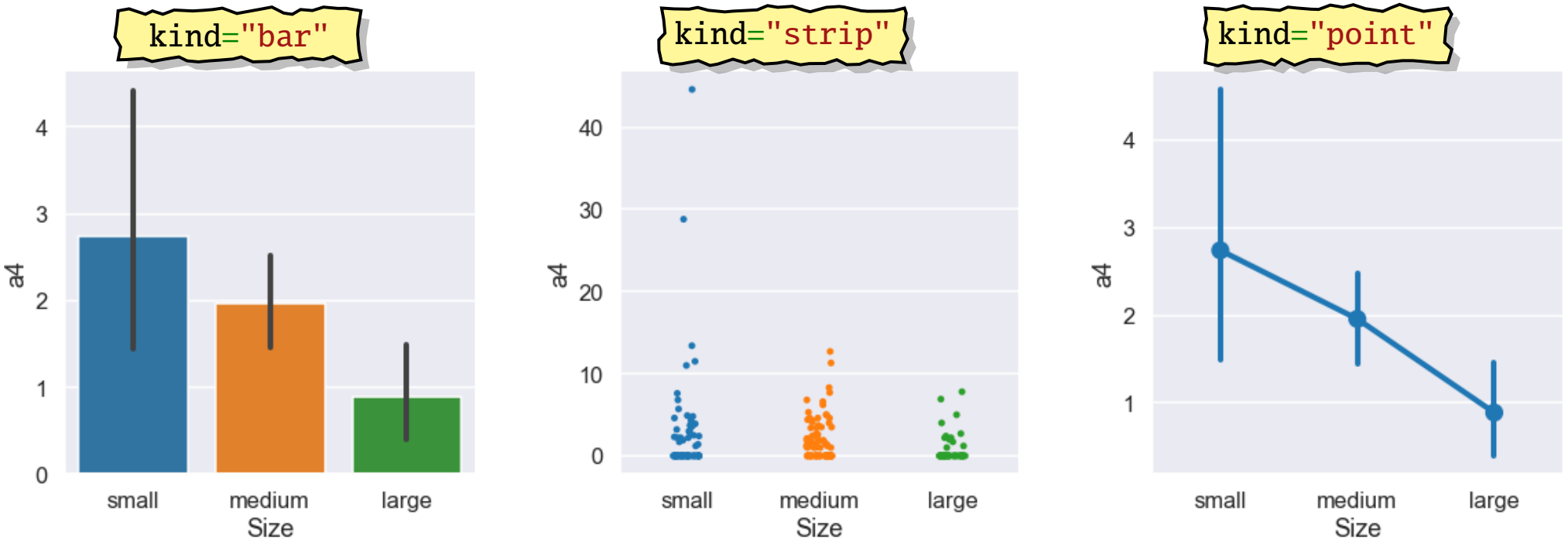



- Shows the average level (mean) and uncertainty (std) of the numerical target (a4) in each categorical level of the categorical variable.
- Vertical bar shows 95% confidence interval.

Is it useful?

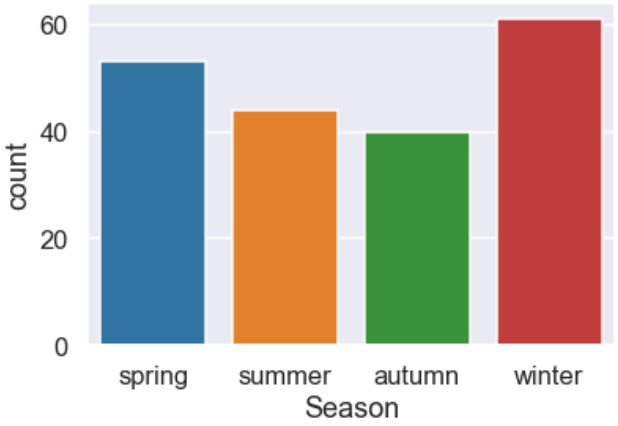
# Categorical Variables — Relationship with (Numerical) Target II

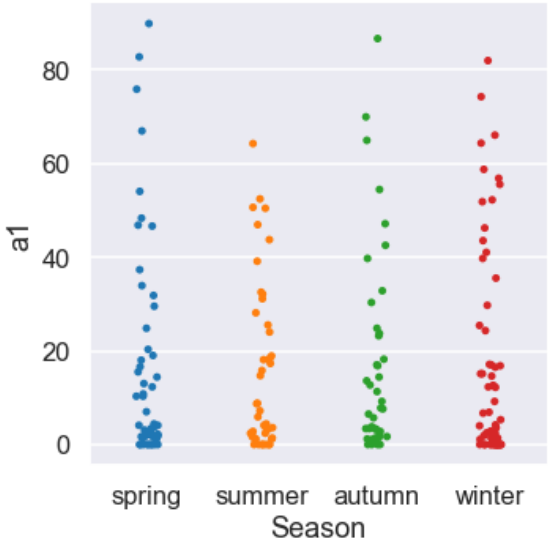
The option `kind` in `catplot` can be:

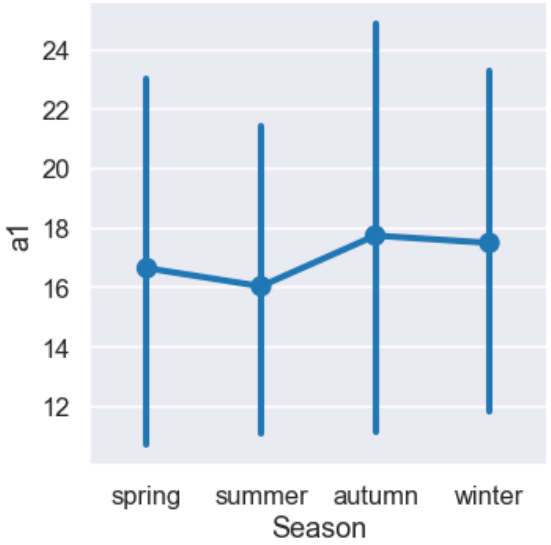



- `bar` and `point` show essentially the same information, but `point` is more compact when comparing multiple categorical features to a continuous target on the same plot.
- `strip` shows individual observations — useful (as in this case) to show that the larger uncertainty in `Size="small"` observations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

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



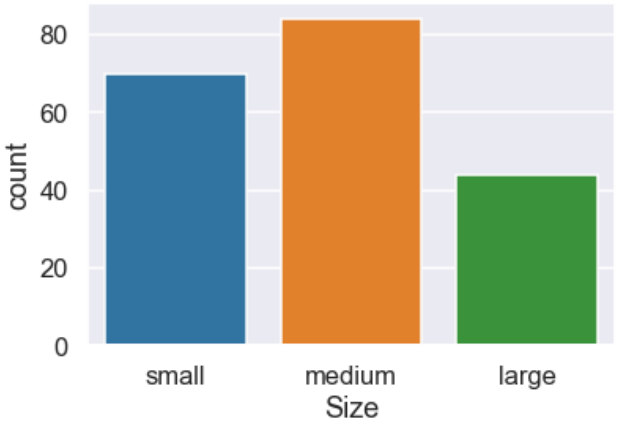

```
df.groupby("Season")["a1"].agg(["mean","count","std"])
```

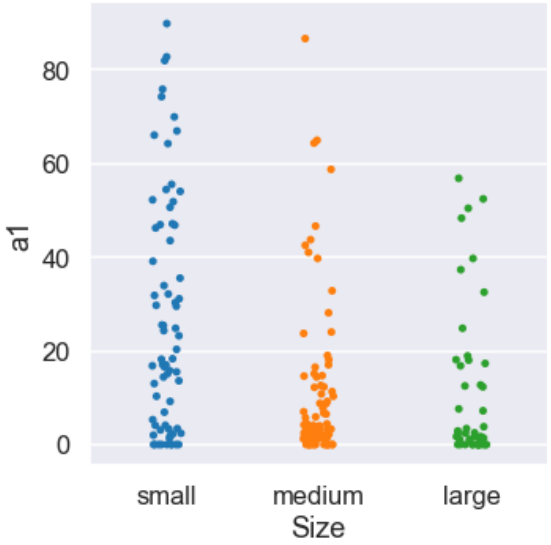
| | mean | count | std |
|---|---|---|---|
| **Season** | $\bar{x}$ | $n$ | $\sigma$ |
| **spring** | 16.649057 | 53 | 23.093786 |
| **summer** | 16.038636 | 44 | 17.920798 |
| **autumn** | 17.745000 | 40 | 21.611203 |
| **winter** | 17.498361 | 61 | 22.568256 |

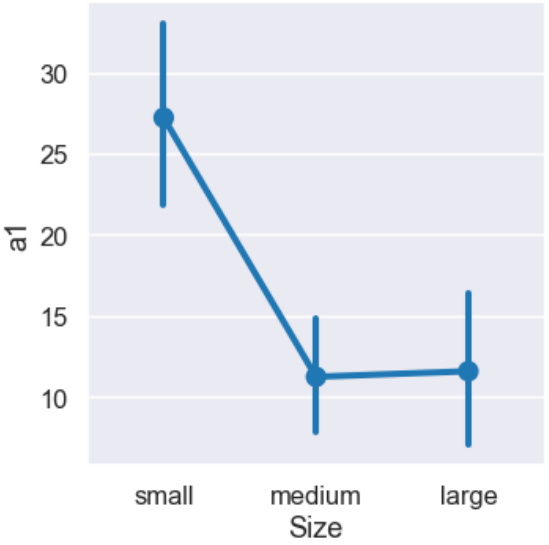
- Countplot shows no issues with feature `Season` — all levels approximately equally represented.
- Catplots show slightly less spread in `a1` for `Season="spring"` observations. (`strip` shows smaller range, `point` shows smaller standard deviation).

$\Rightarrow$ 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** | | | $\bar{x}$ | $n$ | $\sigma$ |
| **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.

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

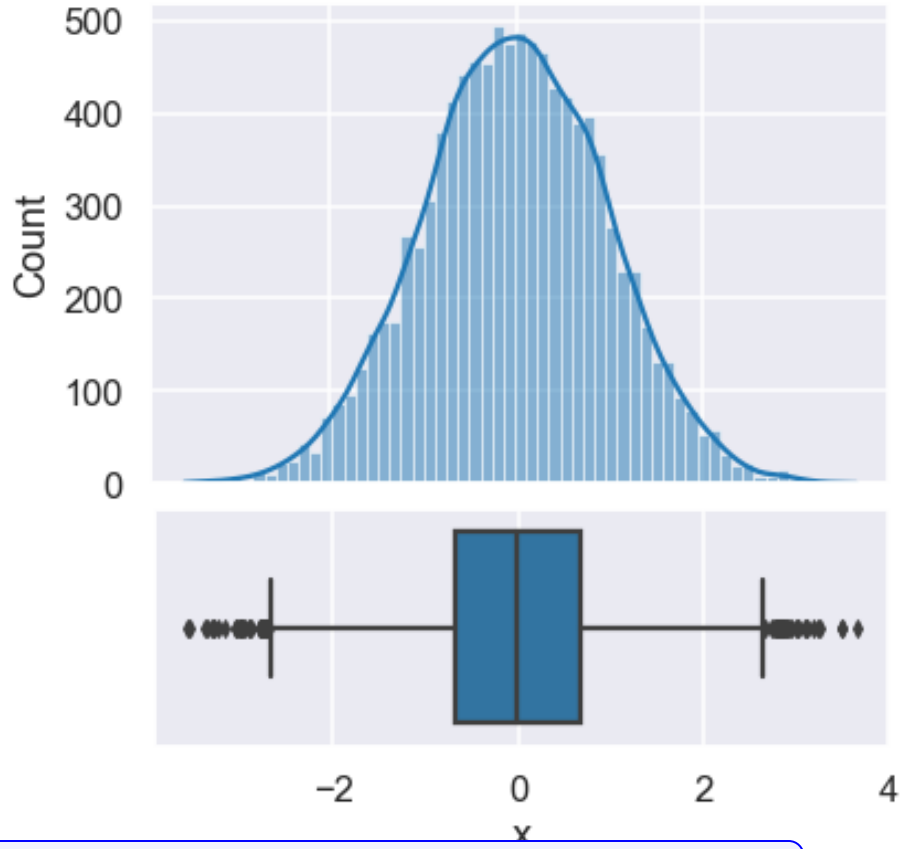
# Numerical Variables

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)
```

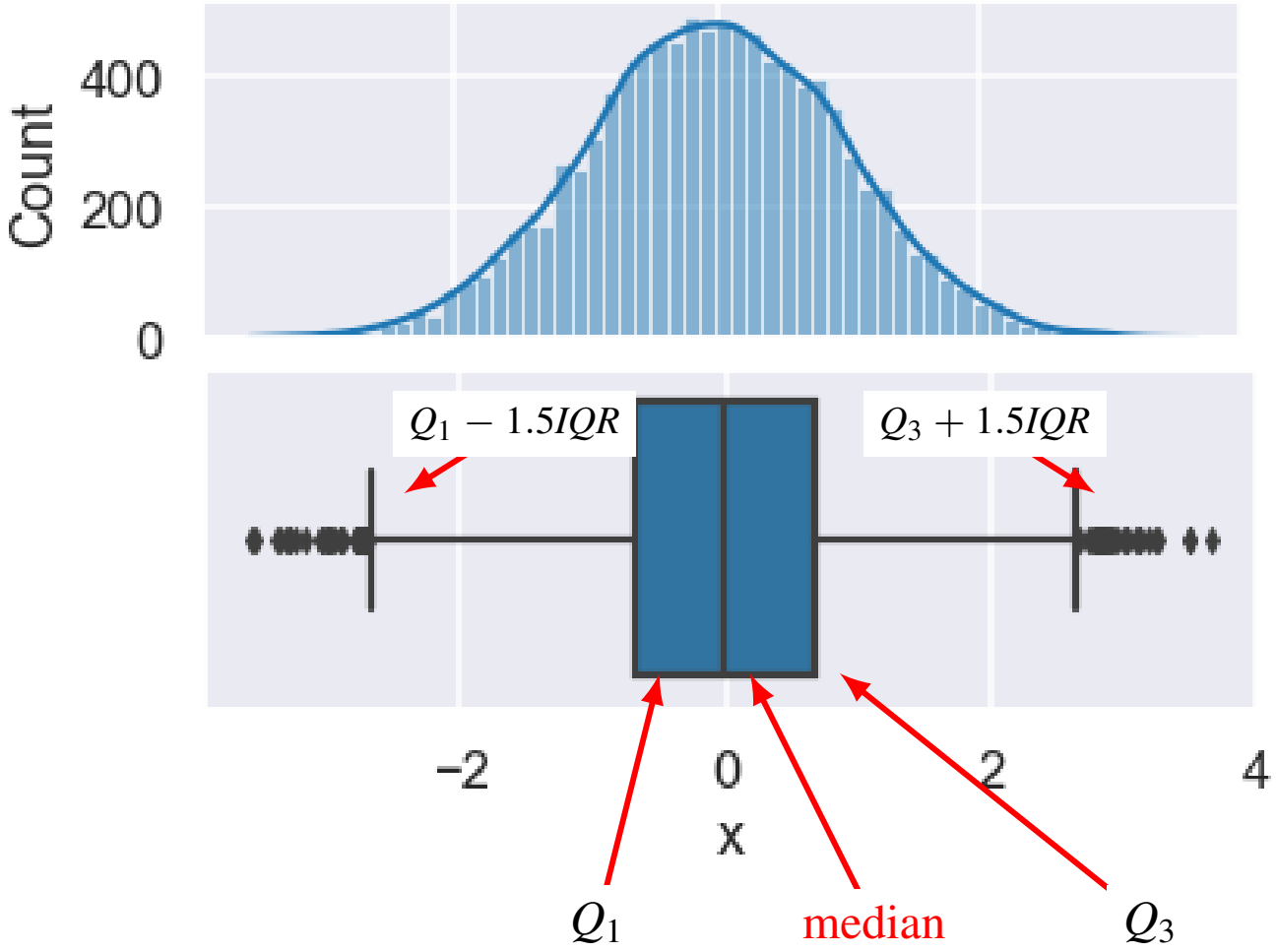
| | x |
|---|---|
| **0** | -0.440553 |
| **1** | 0.507263 |
| **2** | 2.791102 |
| **3** | 0.157782 |
| **4** | -2.740130 |

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




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

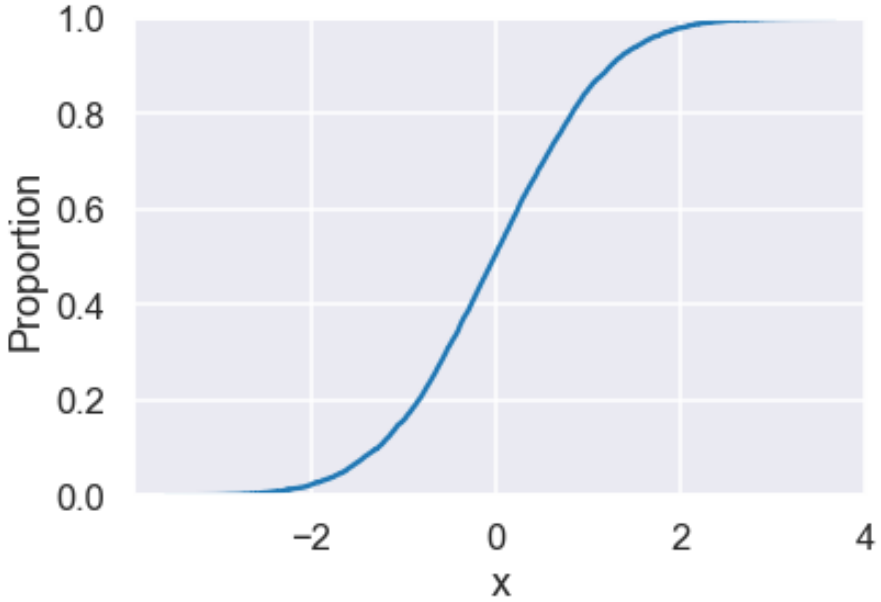
# Histplot (Histogram) and Boxplot




- 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.

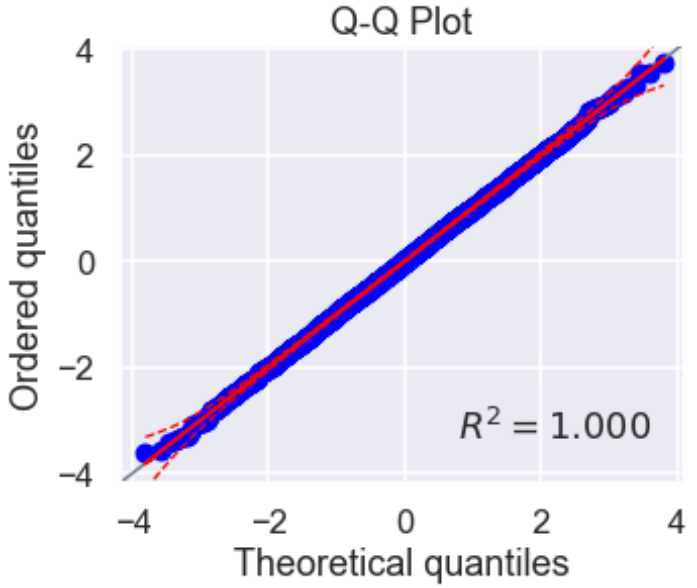
# Cumulative Plot and QQ-Plot

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



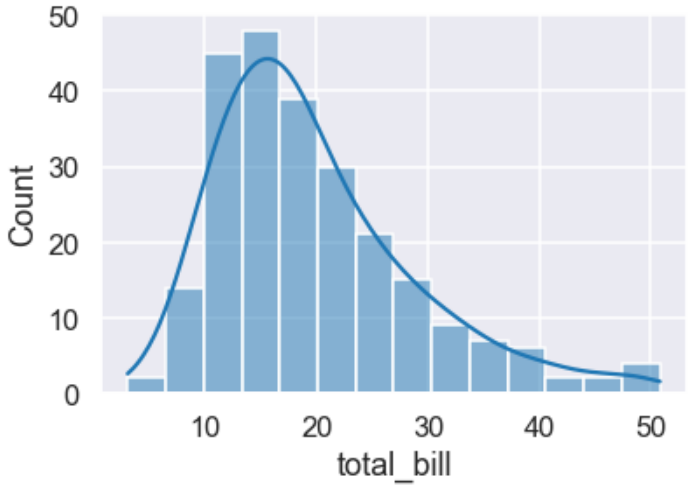

- Represents the proportion of observations less than or equal to given value.

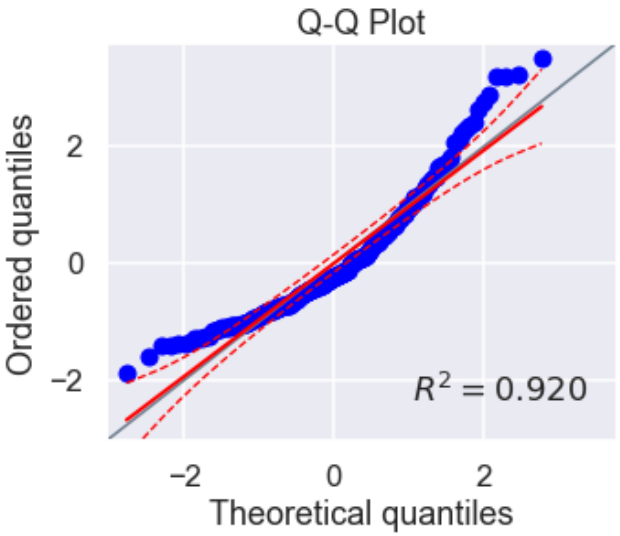
```
import pingouin as pg
pg.qqplot(df2.x);
```




- 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.

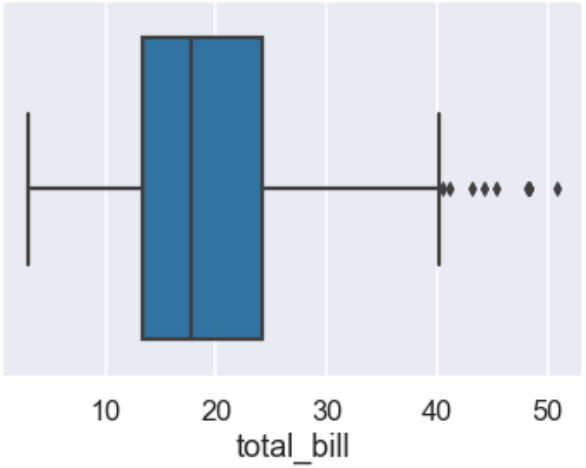
# Example — Dataset: `Tips`, Feature: `total_bill`



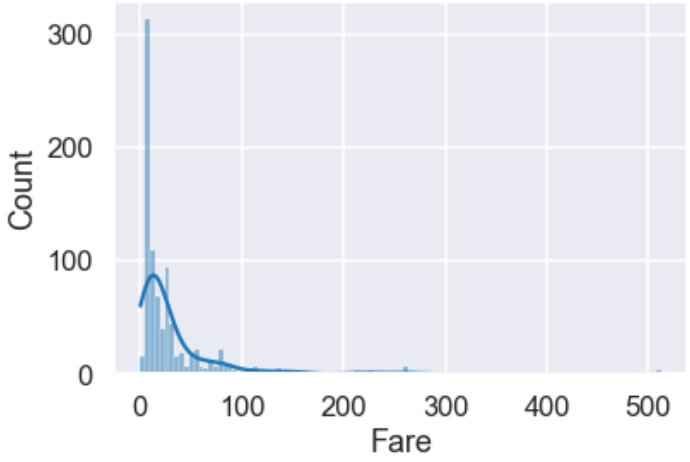

```
df.total_bill.describe()
```

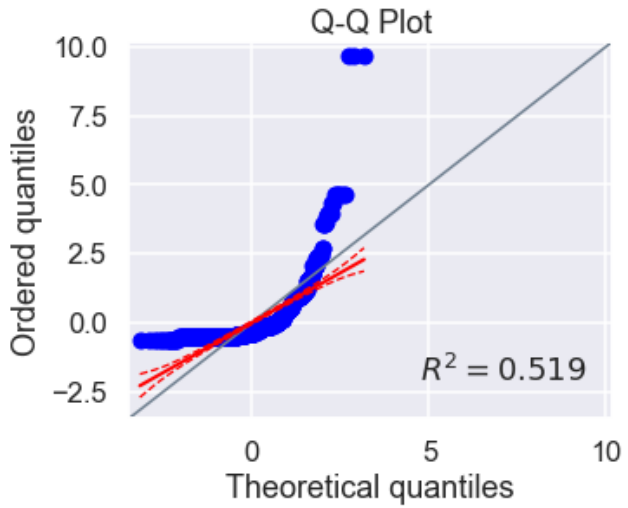
```
count   244.000000
mean     19.785943
std       8.902412
min       3.070000
25%      13.347500
50%      17.795000
75%      24.127500
max      50.810000
Name: total_bill, dtype: float
```




- 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.

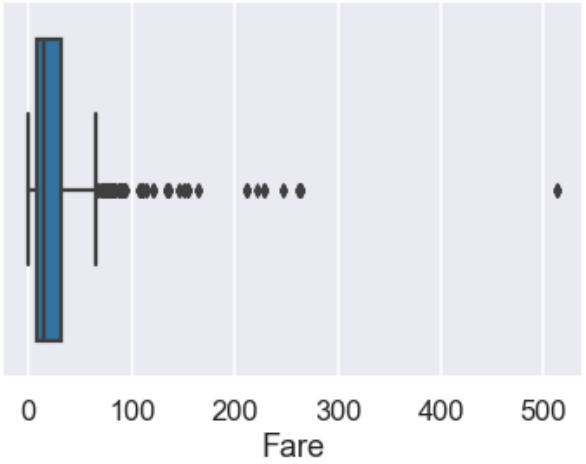
# Example — Dataset: `Titanic`, Feature: `Fare`




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

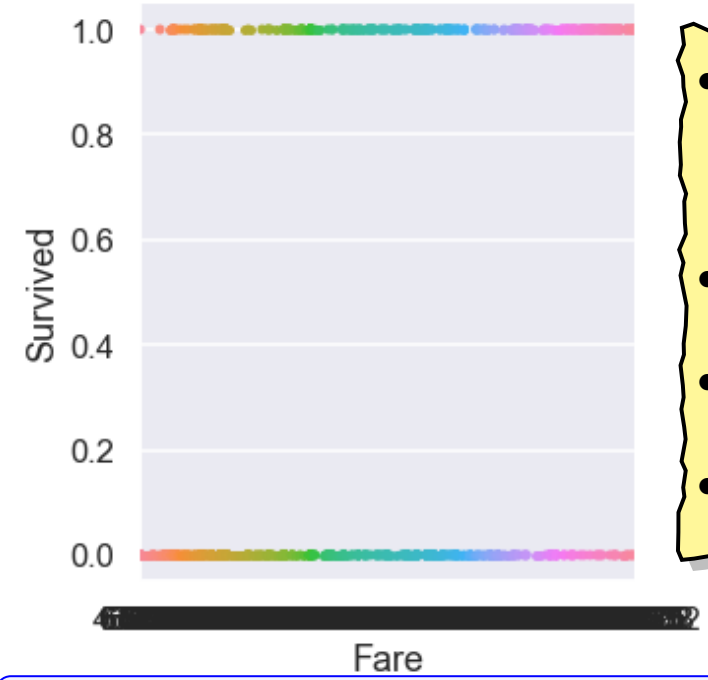
```
count   891.000000
mean     32.204208
std      49.693429
min       0.000000
25%       7.910400
50%      14.454200
75%      31.000000
max     512.329200
Name: Fare, dtype: float64
```




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

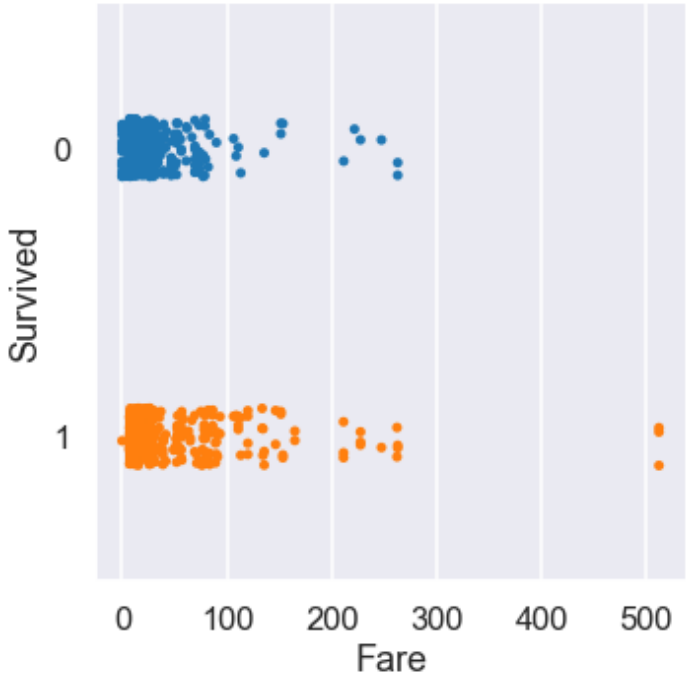
# Warning —- Plot Output Depends on Data Assumptions

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




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

# Second Pass — Individual Features and Target

- Categorical vs numerical target
- Categorical vs numerical features
- Identify (and possibly address) issues

} Is it usable?

- Relationship to target.

} Is it useful?

# Dataset: Titanic, Target: Survived

```
df.Survived.value_counts(normalize=True, dropna=False)
```

```
0    0.616162
1    0.383838
Name: Survived, dtype: float64
```

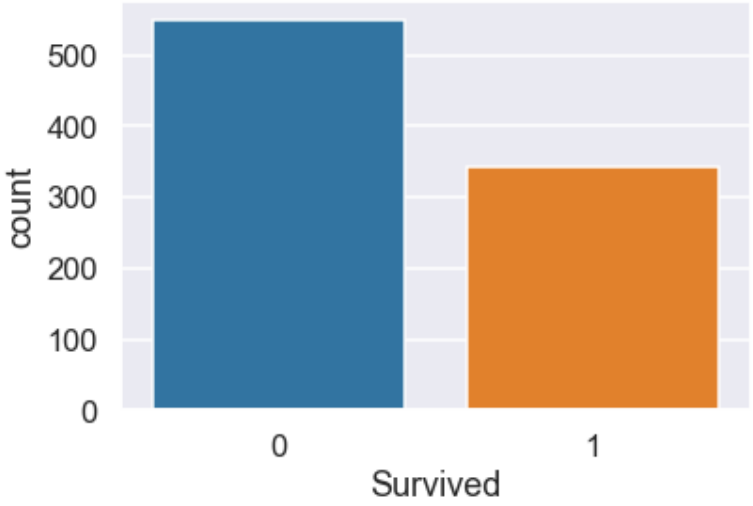
```
df.Survived.describe()
```

```
[0, 1]
Categories (2, int64): [0, 1]
```

```
df.Survived.unique()
```

```
count    891
unique     2
top        0
freq     549
Name: Survived, dtype: int64
```

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




- Simplest classification problem (two classes) with both classes nearly equal frequency.
- In a unbalanced classification problem where the minority class occurs about 20% or lower, models can focus on the majority class.

# Dataset: Algae Blooms, Target: a1,. . . , a7

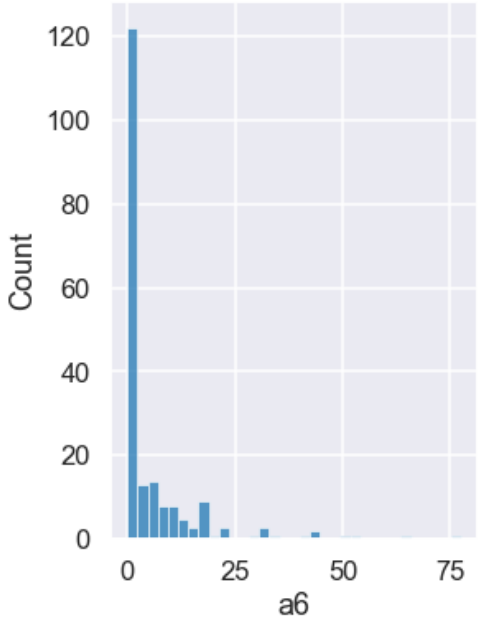
```
targets = [c for c in df.columns if c[0]=="a"]
targets
```

```
['a1', 'a2', 'a3', 'a4', 'a5', 'a6', 'a7']
```

```
df[targets].describe()
```

| | a1 | a2 | a3 | a4 | a5 | a6 | a7 |
|---|---|---|---|---|---|---|---|
| **count** | 198.000000 | 198.000000 | 198.000000 | 198.000000 | 198.000000 | 198.000000 | 198.000000 |
| **mean** | 16.996465 | 7.470707 | 4.334343 | 1.997475 | 5.115657 | 6.004545 | 2.487374 |
| **std** | 21.421713 | 11.065461 | 6.976788 | 4.439205 | 7.511846 | 11.711053 | 5.181536 |
| **min** | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **25%** | 1.525000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 |
| **50%** | 6.950000 | 3.000000 | 1.550000 | 0.000000 | 2.000000 | 0.000000 | 1.000000 |
| **75%** | 24.800000 | 11.275000 | 4.975000 | 2.400000 | 7.500000 | 6.975000 | 2.400000 |
| **max** | 89.800000 | 72.600000 | 42.800000 | 44.600000 | 44.400000 | 77.600000 | 31.600000 |

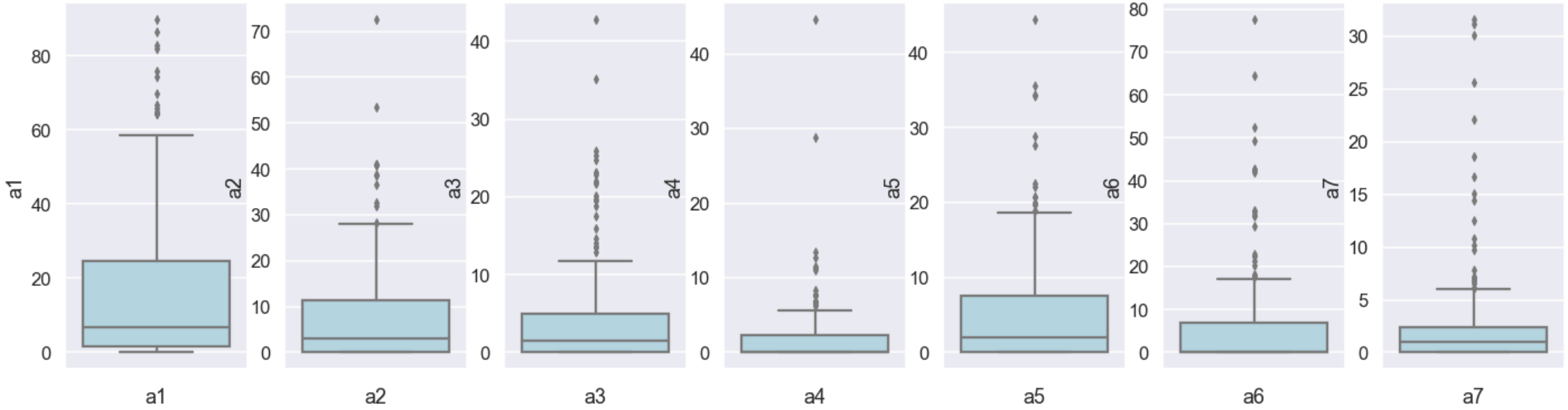
```
plt.figure(figsize=(4,6))
sns.histplot(x="a6", data=df);
```




All distributions are heavily skewed to the right, many with outliers (see next slide). All of the zero measurements are probably due to population levels too low to be measured.

# Dataset: Algae Blooms, Target: a1,. . . , a7

```
fig, axs = plt.subplots(1, 7, figsize=(24,6))
for k, c in enumerate(targets):
    sns.boxplot(data=df, y=c, color="lightblue", ax=axs[k])
    axs[k].set_xlabel(c)
```




The outliers are likely to be true measurements, but their presence can heavily influence the model training — common strategy is to fit two models (one with the case with target outliers and one without) to assess impact of outliers.

# Individual Features

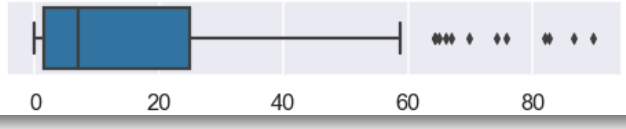
To keep this more manageable we will focus more on the `Algae Blooms` data set ...

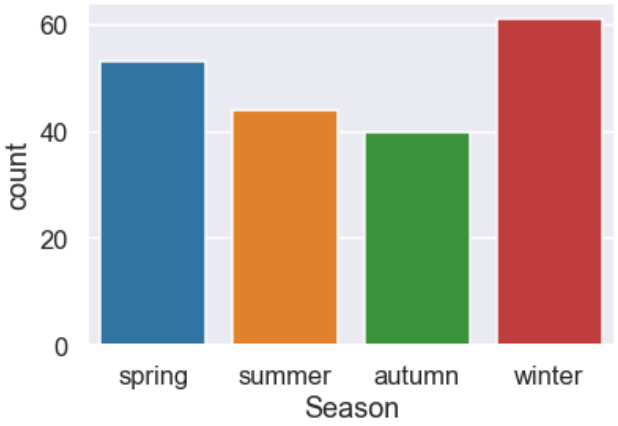
| | Season | Size | Speed | max_pH | min_O2 | mean_Cl | mean_NO3 | mean_NH4 | mean_oPO4 | mean_PO4 | mean_Chlor | a1 | a2 | a3 | a4 | a5 | a6 | a7 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | winter | small | medium | 8.00 | 9.8 | 60.800 | 6.238 | 578.00000 | 105.00000 | 170.00000 | 50.000 | 0.0 | 0.0 | 0.0 | 0.0 | 34.2 | 8.3 | 0.0 |
| **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 | 0.0 | 2.1 |
| **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 | 0.0 | 9.7 |
| **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 | 0.0 | 1.4 |

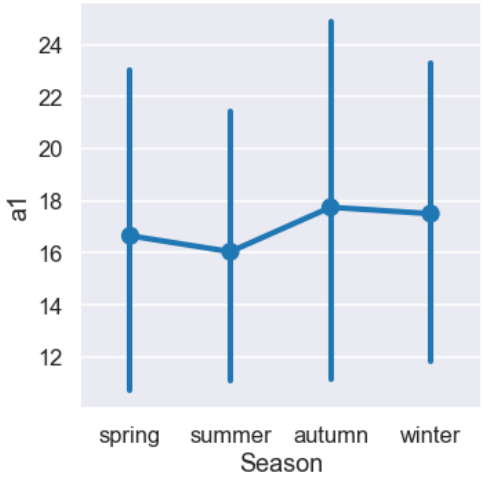
## Sneak perview

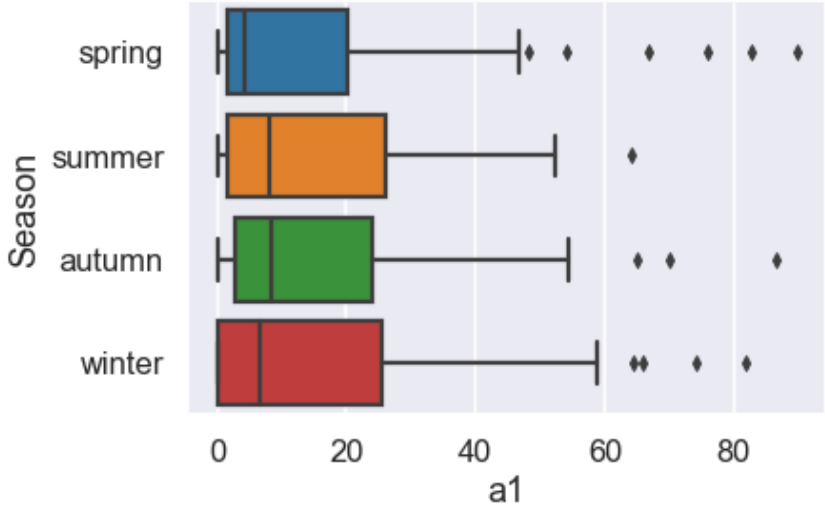
- Three categorical variables `Season`, `Size`, and `Speed`.
  - No missing values
  - No high cardinality, and reasonable balanced.
- Eight numerical variables `max_pH`, ..., `mean_Chlor`
- Missing values present
- Some variables heavily skewed — might need to transform.
- Possibility of features being interrelated — multicollinearity — try principal component analysis.

# Dataset: Algae Blooms, Feature: Season, Target: a1



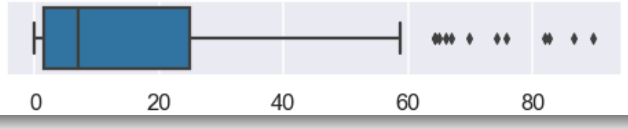



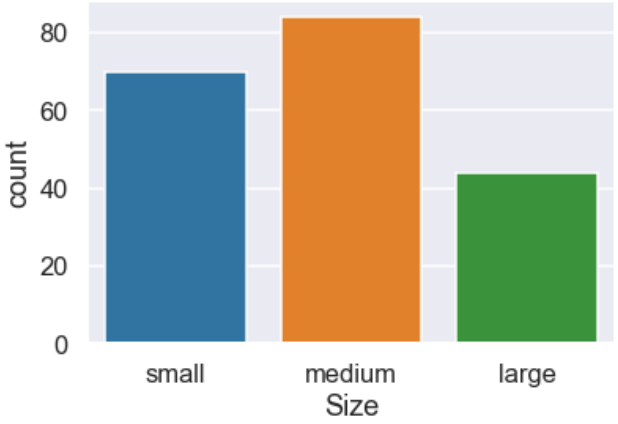

```
df.groupby("Season")["a1"].agg(["min","max","mean","count","std"])
```

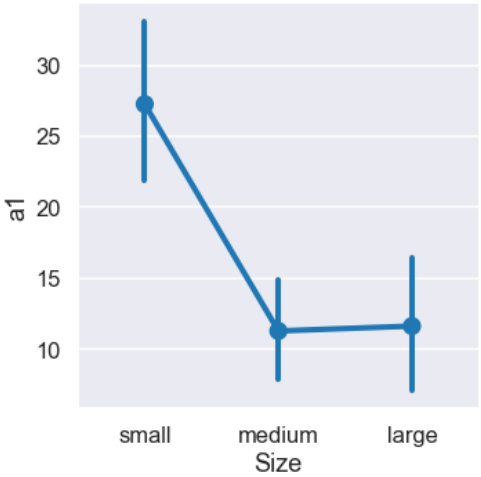
| Season | min | max | mean $\bar{x}$ | count $n$ | std $\sigma$ |
|---|---|---|---|---|---|
| **spring** | 0.0 | 89.8 | 16.649057 | 53 | 23.093786 |
| **summer** | 0.0 | 64.2 | 16.038636 | 44 | 17.920798 |
| **autumn** | 0.0 | 86.6 | 17.745000 | 40 | 21.611203 |
| **winter** | 0.0 | 81.9 | 17.498361 | 61 | 22.568256 |

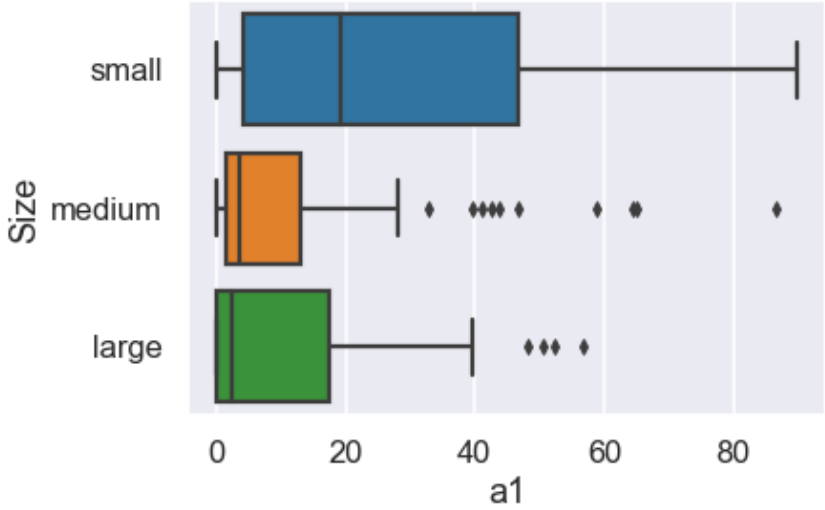
- Countplot shows no issues with feature Season — all levels approximately equally represented.
- Catplots show slightly less spread in a1 for Season="spring" observations.
- No/weak relationship between Season feature and a1 target.

# Dataset: Algae Blooms, Feature: Size, Target: a1



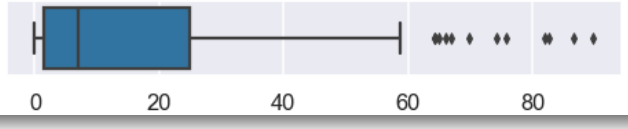



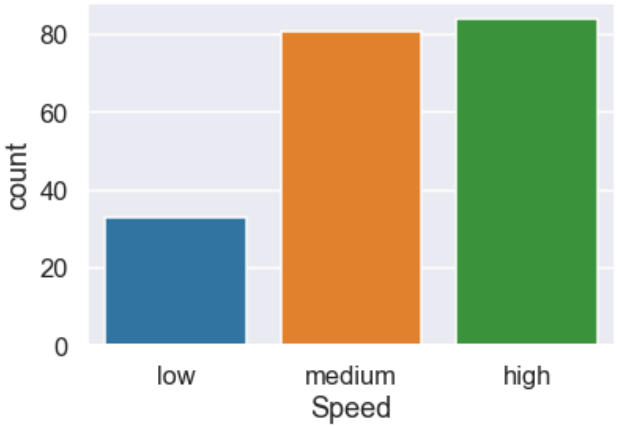

```
df.groupby("Size")["a1"].agg(["min","max","mean","count","std"])
```

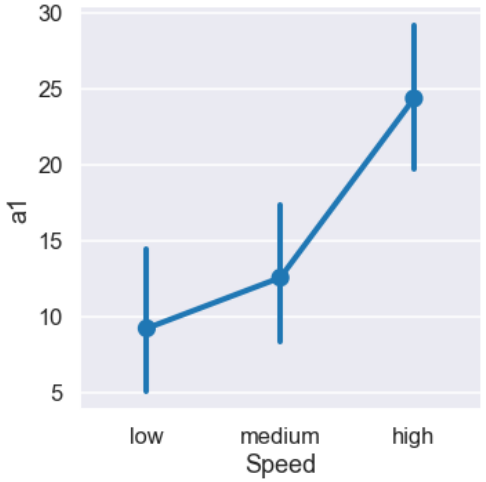
| Size | min | max | mean | count | std |
|---|---|---|---|---|---|
| **small** | 0.0 | 89.8 | 27.255714 | 70 | 24.895426 |
| **medium** | 0.0 | 86.6 | 11.267857 | 84 | 17.163124 |
| **large** | 0.0 | 56.8 | 11.611364 | 44 | 16.556123 |

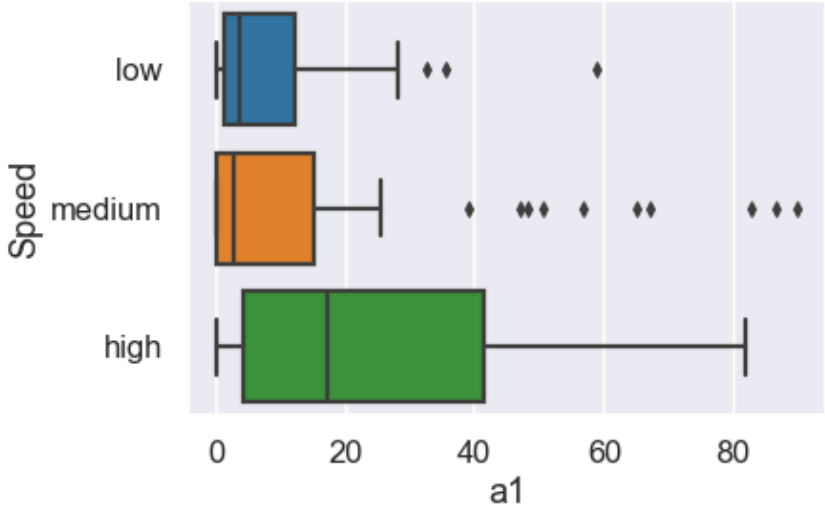
- Countplot shows no issues with feature `Size`.
- `Size="small"` rivers have higher frequencies of `a1` alga ((point) catplot), and observed frequencies for small rivers is much more widespread across the domain of frequencies than for other types of rivers (boxplot).

# Dataset: Algae Blooms, Feature: Speed, Target: a1



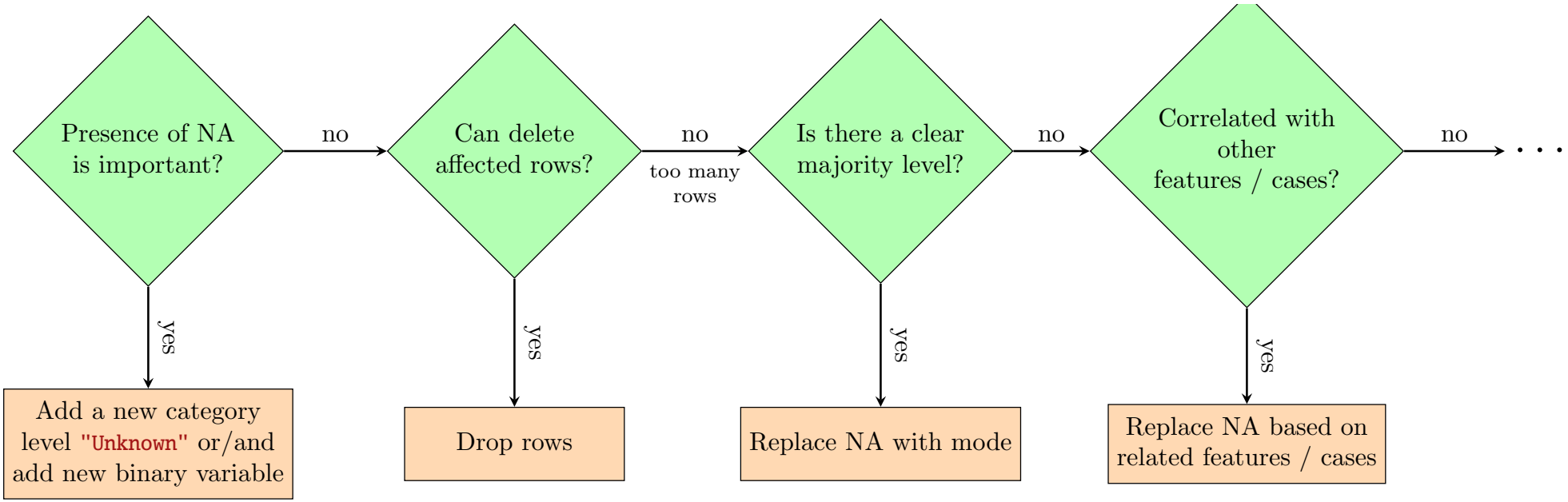




```
df.groupby("Speed")["a1"].agg(["min","max","mean","count","std"])
```

| Speed | min | max | mean | count | std |
|---|---|---|---|---|---|
| **low** | 0.0 | 58.7 | 9.209091 | 33 | 13.164758 |
| **medium** | 0.0 | 89.8 | 12.548148 | 81 | 21.146986 |
| **high** | 0.0 | 81.9 | 24.345238 | 84 | 22.209123 |

- Countplot shows no issues with feature `Size`.
- `Speed="high"` rivers have higher frequencies of `a1` alga ((point) catplot), and observed frequencies for small rivers is much more widespread across the domain of frequencies than for other types of rivers (boxplot).

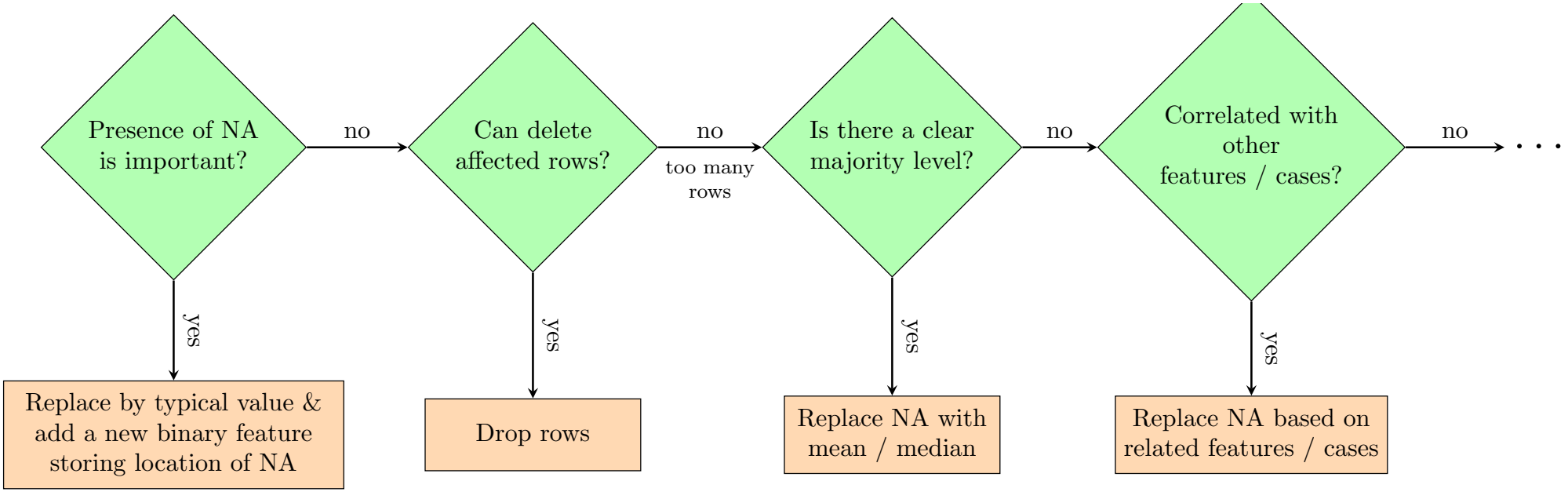
# Categorical Variables — Dealing with Missing Values




In terms of our three datasets, only `Titanic` has missing values in categorical features:

- Location of `cabin`'s missing values are important (1st class passengers were most likely to have a cabin) so add new category level `"Unknown"`.
- Replace `Embarked`'s 2 missing values with mode (`"S"`, 644/891=72%).
  Note: Use `df.Embarked.value_counts(dropna=False)` to include missing values in count tables.

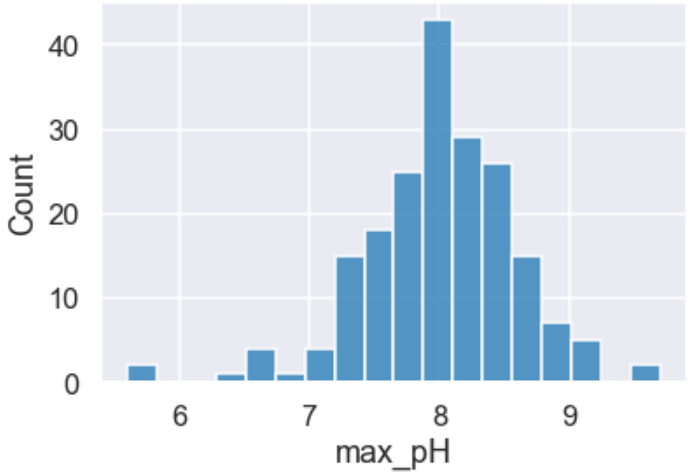
# Numerical Variables — Dealing with Missing Values

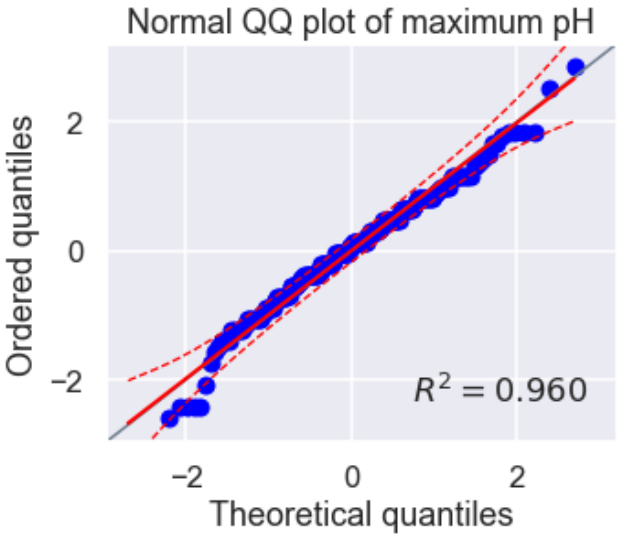



In terms of our three datasets:

- In `Titanic`, feature `Fare` appears to have no missing values, but has 15 zero entries. Are these missing values? or free tickets due to age? . . .
- In `Algae Blooms`, some of the 8 numeric features have NAs . . . next few slides.

# Dataset: `Algae Blooms`, Feature: `max_pH`

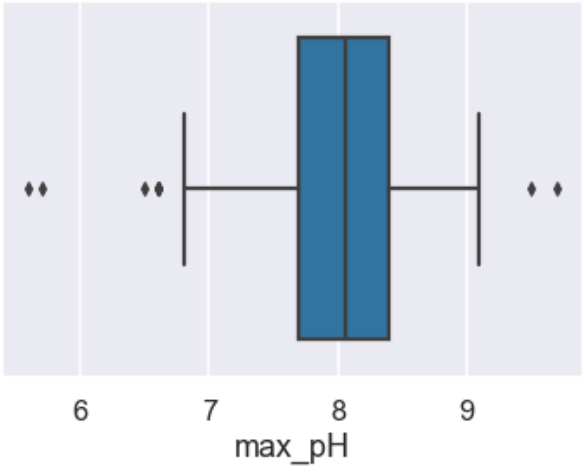



```
df.max_pH.isna().sum()
```

```
1
```

```
count   197.000000
mean      8.019975
std       0.590169
min       5.600000
25%       7.700000
50%       8.060000
75%       8.400000
max       9.700000
Name: max_pH, dtype: float64
```




- Data is relatively normal — minor issue with (left) outliers.

$\Rightarrow$ Will replace (single) NA by mean

```
df.max_pH.fillna(df.max_pH.mean(), inplace=True)
```

# Dataset: Algae Blooms, Feature: max_pH, Target: a1

Is there a relationship between feature max_pH and target a1?

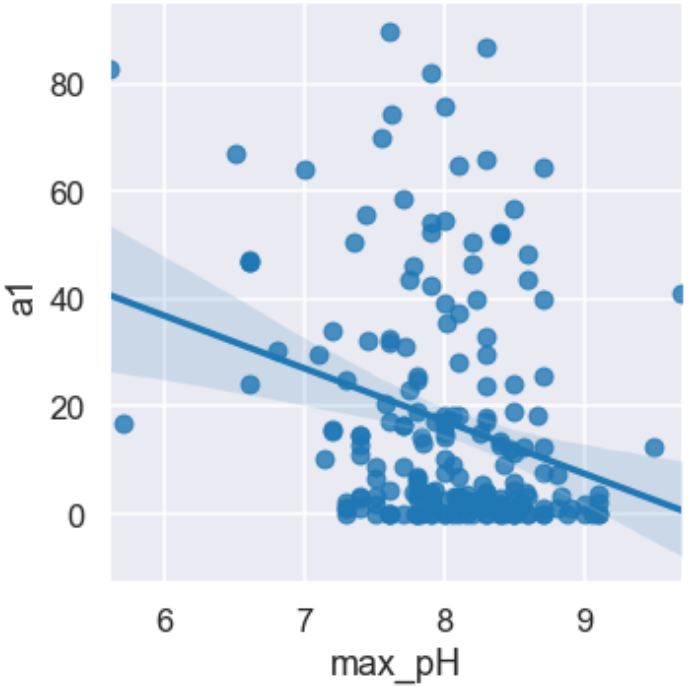
```
df[["max_pH", "a1"]].corr()
```

| | max_pH | a1 |
|---|---|---|
| **max_pH** | 1.000000 | -0.268539 |
| **a1** | -0.268539 | 1.000000 |

(Pearson's) Correlation coefficient, $r$, measures the strength of a **linear** relationship between two numerical variables.

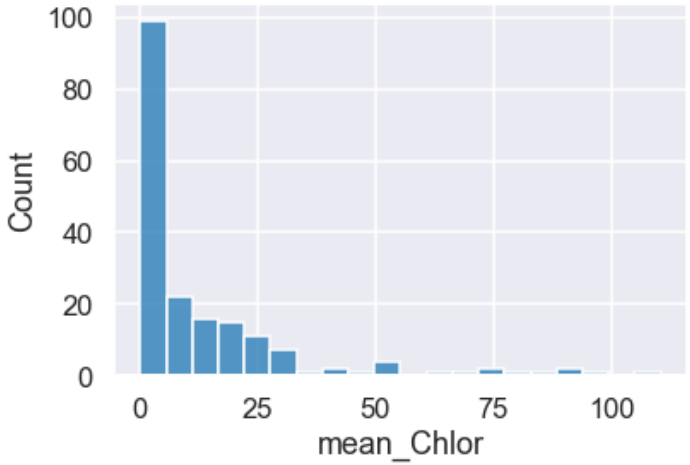
- near zero means no/weak linear relationship.
- near $\pm 1$ zero means strong linear relationship.
- sign indicates direction.

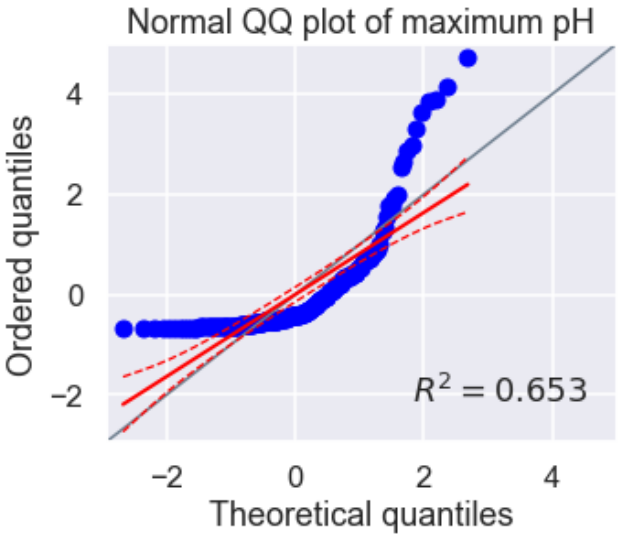
```
sns.lmplot(x="max_pH", y="a1", data=df);
```




- Correlation coefficient, $r = -0.27$, shows (at most) a weak negative linear relationship.
- No obvious relationship visible in scatter plot.

# Dataset: Algae Blooms, Feature: mean_Chlor

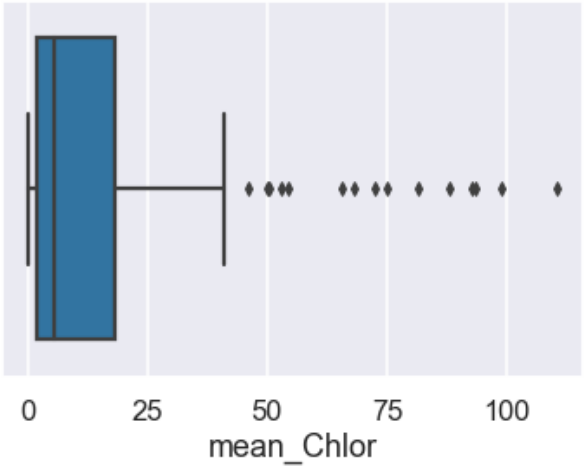



```
df.mean_Chlor.isna().sum()
```

```
10
```

```
count   188.000000
mean     13.971197
std      20.495920
min       0.200000
25%       2.000000
50%       5.475000
75%      18.307500
max     110.456000
Name: mean_Chlor, dtype: float
```




- Data is not normal, heavily skewed to the right ⇒ mean is a poor representative of the central location.

⇒ Will replace (single) NA by median

```
df.mean_Chlor.fillna(df.mean_Chlor.median(), inplace=True)
```

# After Target and Individual Feature Pass — Where are we? I

## Tips

- Reviewed each feature — location, spread, shape, issues.
- No missing values
- `total_bill`, and `total_tip` have possible outliers.

## Titanic

- Reviewed each feature — location, spread, shape, issues.
- Generated ToDo list for for cleaning, feature extraction
  - Identified features that appear to be related to the target.
  - Feature `age` has missing values.
  - Feature `Fare`
    - has 15 measurements with value 0 — decide missing value or not.
    - distribution has large outliers and is skewed — remove/fix outliers and transform.
  - Feature `Name` has could be used to obtain new feature `Title`.
  - ...

## Algae Blooms

- Reviewed each feature — location, spread, shape, issues.
- Imputed missing values using feature distributions (mean/median).
- Identified features that appear to be related to the target.

# Aside: Steps needed to create new feature `Title` from feature `Name`

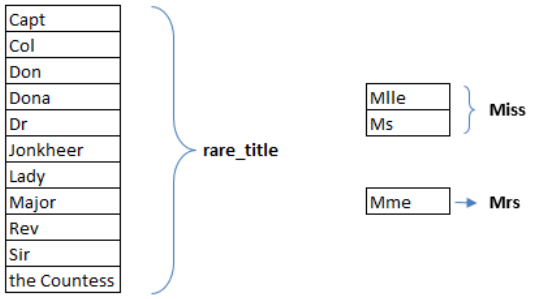
```
df = pd.read_csv('assets/train.csv')
df.head()
```

| | 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 | 7.2 | NaN | S |
| 1 | 2 | 1 | 1 | Cumings, Mrs. John Bradley (Florence Briggs Th... | female | 38.0 | 1 | 0 | PC 17599 | 71.28 | | |
| | | | | Heikkinen, Miss. Laina | female | 26.0 | 0 | 0 | STON/O2. 3101282 | 7.925 | | |
| | | | | Futrelle, Mrs. Jacques Heath (Lily May Peel) | female | 35.0 | 1 | 0 | 113803 | 53.1000 | C123 | S |
| | | | | Allen, Mr. William Henry | male | 35.0 | 0 | 0 | 373450 | 8.0500 | NaN | S |

1 Have a (text) feature representing the passenger's name.

| Title | Count |
|---|---|
| Capt | 1 |
| Col | 4 |
| Don | 1 |
| Dona | 1 |
| Dr | 8 |
| Jonkheer | 1 |
| Lady | 1 |
| Major | 2 |
| Master | 61 |
| Miss | 260 |
| Mlle | 2 |
| Mme | 1 |
| Mr | 757 |
| Mrs | 197 |
| Ms | 2 |
| Rev | 8 |
| Sir | 1 |
| the Countess | 1 |

2 Rare categories can be merged under a single label – 'Dona', 'Lady', 'the Countess', 'Capt', 'Col', etc. .
Duplicate/redundant categories can be merged – the titles 'Mlle', and 'Ms' and can be merged under 'Miss'. 'Mme' can be merged with 'Mrs'.




3 New categorical feature with 5 unique values

| Master | Miss | Mr | Mrs | rare_title |
|---|---|---|---|---|
| 61 | 264 | 757 | 198 | 29 |

# Third Pass — Relationships Between Features (and Target)

- Correlations

# Correlations — Relationship Between two Variables

## Pearson's correlation coefficient, $r$

is a measure of linear correlation between two variables. Its value lies between -1 and +1, -1 indicating total negative linear correlation, 0 indicating no linear correlation and 1 indicating total positive linear correlation.

## Spearman's rank correlation coefficient, $\rho$

is a measure of monotonic correlation between two variables, and is therefore better in catching nonlinear monotonic correlations than Pearson's $r$. Its value also lies between -1 and +1, with values near zero indicating no monotonic relation.

## Kendall rank correlation coefficient, $\tau$

measures ordinal association between two variables. Its value lies between -1 and +1 with values near zero indicating no relation.

## Phi-k, $\phi k$

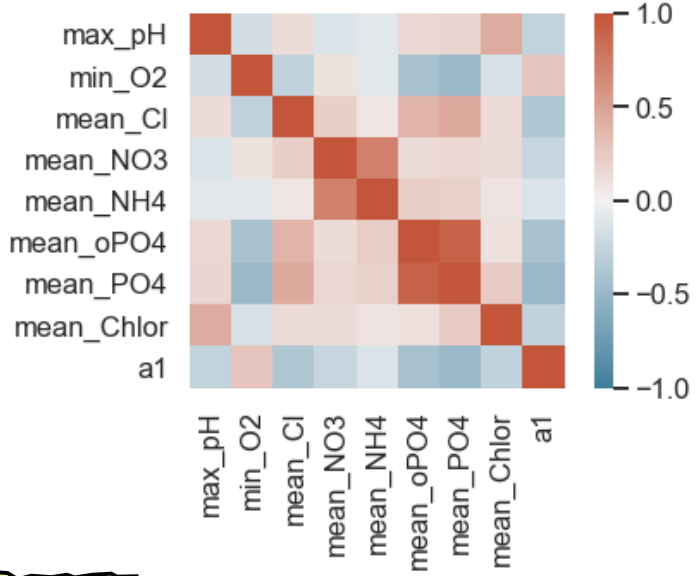
is a new and practical correlation coefficient that works consistently between categorical, ordinal and interval variables, captures non-linear dependency and reverts to the Pearson correlation coefficient in case of a bivariate normal input distribution. Its value also lies between 0 and +1, with values near zero indicating no relation.

# Pearson's Correlation Coefficient — Dataset: Algae Blooms

```
columns = df.columns[:12]
corr = df[columns].corr()
corr
```

```
cmap = sns.diverging_palette(230, 20, as_cmap=True)
sns.heatmap(corr, square=True, vmin=-1, vmax=1, cmap=cmap);
```

| | max_pH | min_O2 | mean_Cl | mean_NO3 | mean_NH4 | mean_oPO4 | mean_PO4 | mean_Chlor | a1 |
|---|---|---|---|---|---|---|---|---|---|
| **max_pH** | 1.000000 | -0.167981 | 0.136369 | -0.130762 | -0.093521 | 0.158769 | 0.179885 | 0.445864 | -0.268539 |
| **min_O2** | -0.167981 | 1.000000 | -0.278333 | 0.099444 | -0.087478 | -0.416163 | -0.487486 | -0.153265 | 0.285564 |
| **mean_Cl** | 0.136369 | -0.278333 | 1.000000 | 0.225041 | 0.071913 | 0.391054 | 0.457449 | 0.149856 | -0.371171 |
| **mean_NO3** | -0.130762 | 0.099444 | 0.225041 | 1.000000 | 0.721444 | 0.144588 | 0.168601 | 0.139679 | -0.241211 |
| **mean_NH4** | -0.093521 | -0.087478 | 0.071913 | 0.721444 | 1.000000 | 0.227237 | 0.208180 | 0.088947 | -0.132656 |
| **mean_oPO4** | 0.158769 | -0.416163 | 0.391054 | 0.144588 | 0.227237 | 1.000000 | 0.914365 | 0.115621 | -0.417358 |
| **mean_PO4** | 0.179885 | -0.487486 | 0.457449 | 0.168601 | 0.208180 | 0.914365 | 1.000000 | 0.253621 | -0.487023 |
| **mean_Chlor** | 0.445864 | -0.153265 | 0.149856 | 0.139679 | 0.088947 | 0.115621 | 0.253621 | 1.000000 | -0.277987 |
| **a1** | -0.268539 | 0.285564 | -0.371171 | -0.241211 | -0.132656 | -0.417358 | -0.487023 | -0.277987 | 1.000000 |



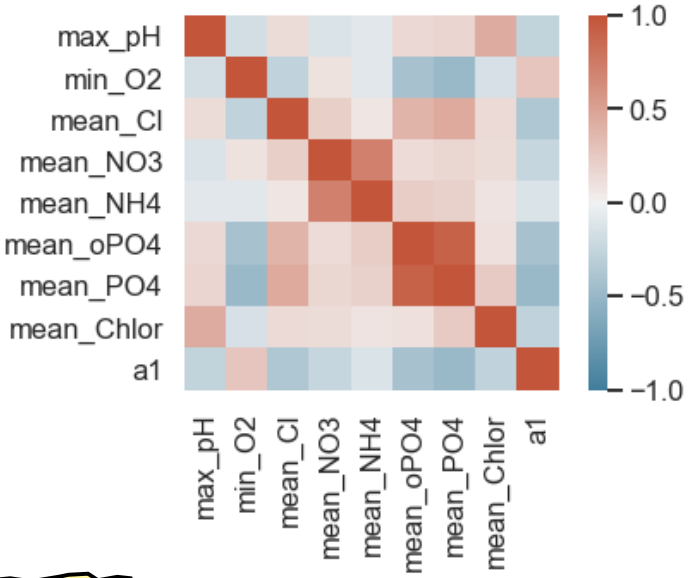

- Categorical variables are not included.
- Suggests best predictors for `a1` are `mean_PO4`, `mean_oPO4`, and `meanCl`.
- `mean_PO4` and `mean_oPO4` are highly correlated (0.91) — could use values of one to estimate missing values of the other.

# Spearman's Rank Correlation Coefficient — Dataset: Algae Blooms

```
columns = df.columns[:12]
coor = df[columns].corr(method='spearman')
coor
```

```
cmap = sns.diverging_palette(230, 20, as_cmap=Tr
sns.heatmap(corr, square=True, vmin=−1, vmax=1,
```

| | max_pH | min_O2 | mean_Cl | mean_NO3 | mean_NH4 | mean_oPO4 | mean_PO4 | mean_Chlor | a1 |
|---|---|---|---|---|---|---|---|---|---|
| **max_pH** | 1.000000 | -0.148676 | 0.159079 | -0.145182 | 0.026160 | 0.290245 | 0.214569 | 0.394813 | -0.247787 |
| **min_O2** | -0.148676 | 1.000000 | -0.405142 | 0.057610 | -0.348226 | -0.457805 | -0.519786 | -0.217714 | 0.283418 |
| **mean_Cl** | 0.159079 | -0.405142 | 1.000000 | 0.530374 | 0.592052 | 0.670399 | 0.713479 | 0.564915 | -0.546845 |
| **mean_NO3** | -0.145182 | 0.057610 | 0.530374 | 1.000000 | 0.425010 | 0.432303 | 0.451272 | 0.346805 | -0.382403 |
| **mean_NH4** | 0.026160 | -0.348226 | 0.592052 | 0.425010 | 1.000000 | 0.603157 | 0.646690 | 0.406656 | -0.449194 |
| **mean_oPO4** | 0.290245 | -0.457805 | 0.670399 | 0.432303 | 0.603157 | 1.000000 | 0.914921 | 0.510930 | -0.671019 |
| **mean_PO4** | 0.214569 | -0.519786 | 0.713479 | 0.451272 | 0.646690 | 0.914921 | 1.000000 | 0.554167 | -0.656670 |
| **mean_Chlor** | 0.394813 | -0.217714 | 0.564915 | 0.346805 | 0.406656 | 0.510930 | 0.554167 | 1.000000 | -0.537823 |
| **a1** | -0.247787 | 0.283418 | -0.546845 | -0.382403 | -0.449194 | -0.671019 | -0.656670 | -0.537823 | 1.000000 |



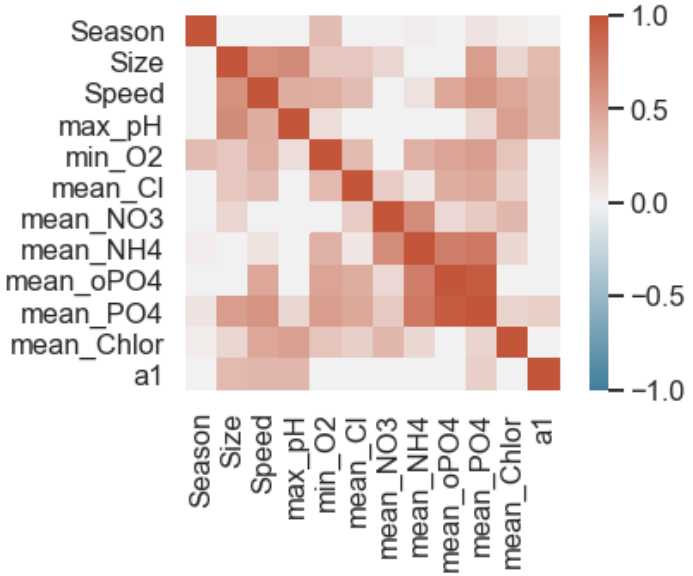

- Now best predictors for `a1` also include `mean_Chlor` and `mean_NH4`.

# Phik Correlation Coefficient — Dataset: Algae Blooms

```
import phik
columns = df.columns[:12]
corr = df[columns].phik_matrix()
corr
```

```
cmap = sns.diverging_palette(230, 20, as_cmap=Tr
sns.heatmap(corr, square=True, vmin=−1, vmax=1,
```

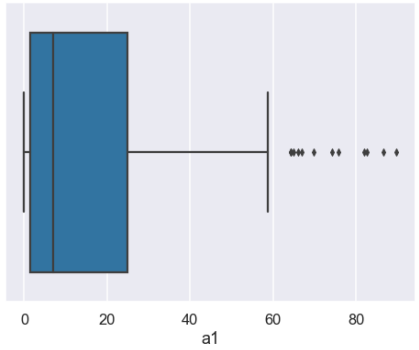
| | Season | Size | Speed | max_pH | min_O2 | mean_Cl | mean_NO3 | mean_NH4 | mean_oPO4 | mean_PO4 | mean_Chlor | a1 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Season** | 1.000000 | 0.000000 | 0.000000 | 0.000000 | 0.343496 | 0.000000 | 0.000000 | 0.034202 | 0.000000 | 0.093199 | 0.045361 | 0.000000 |
| **Size** | 0.000000 | 1.000000 | 0.620101 | 0.655207 | 0.270013 | 0.268198 | 0.182410 | 0.000000 | 0.000000 | 0.531635 | 0.173516 | 0.353390 |
| **Speed** | 0.000000 | 0.620101 | 1.000000 | 0.445096 | 0.437356 | 0.339237 | 0.000000 | 0.101348 | 0.483298 | 0.594480 | 0.479735 | 0.369374 |
| **max_pH** | 0.000000 | 0.655207 | 0.445096 | 1.000000 | 0.125231 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.175105 | 0.528134 | 0.372031 |
| **min_O2** | 0.343496 | 0.270013 | 0.437356 | 0.125231 | 1.000000 | 0.353196 | 0.000000 | 0.416999 | 0.492457 | 0.535996 | 0.296376 | 0.000000 |
| **mean_Cl** | 0.000000 | 0.268198 | 0.339237 | 0.000000 | 0.353196 | 1.000000 | 0.243887 | 0.073692 | 0.443047 | 0.472824 | 0.225583 | 0.000000 |
| **mean_NO3** | 0.000000 | 0.182410 | 0.000000 | 0.000000 | 0.000000 | 0.243887 | 1.000000 | 0.642789 | 0.158463 | 0.259915 | 0.368142 | 0.000000 |
| **mean_NH4** | 0.034202 | 0.000000 | 0.101348 | 0.000000 | 0.416999 | 0.073692 | 0.642789 | 1.000000 | 0.734681 | 0.776197 | 0.167533 | 0.000000 |
| **mean_oPO4** | 0.000000 | 0.000000 | 0.483298 | 0.000000 | 0.492457 | 0.443047 | 0.158463 | 0.734681 | 1.000000 | 0.954601 | 0.000000 | 0.000000 |
| **mean_PO4** | 0.093199 | 0.531635 | 0.594480 | 0.175105 | 0.535996 | 0.472824 | 0.259915 | 0.776197 | 0.954601 | 1.000000 | 0.192920 | 0.221308 |
| **mean_Chlor** | 0.045361 | 0.173516 | 0.479735 | 0.528134 | 0.296376 | 0.225583 | 0.368142 | 0.167533 | 0.000000 | 0.192920 | 1.000000 | 0.000000 |
| **a1** | 0.000000 | 0.353390 | 0.369374 | 0.372031 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.000000 | 0.221308 | 0.000000 | 1.000000 |



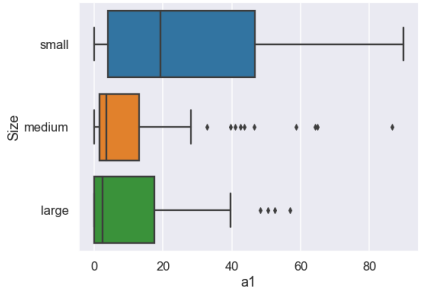

- Now include categorical variables — Season is not related, but Size and Speed are.

# Multi-Relation Plots

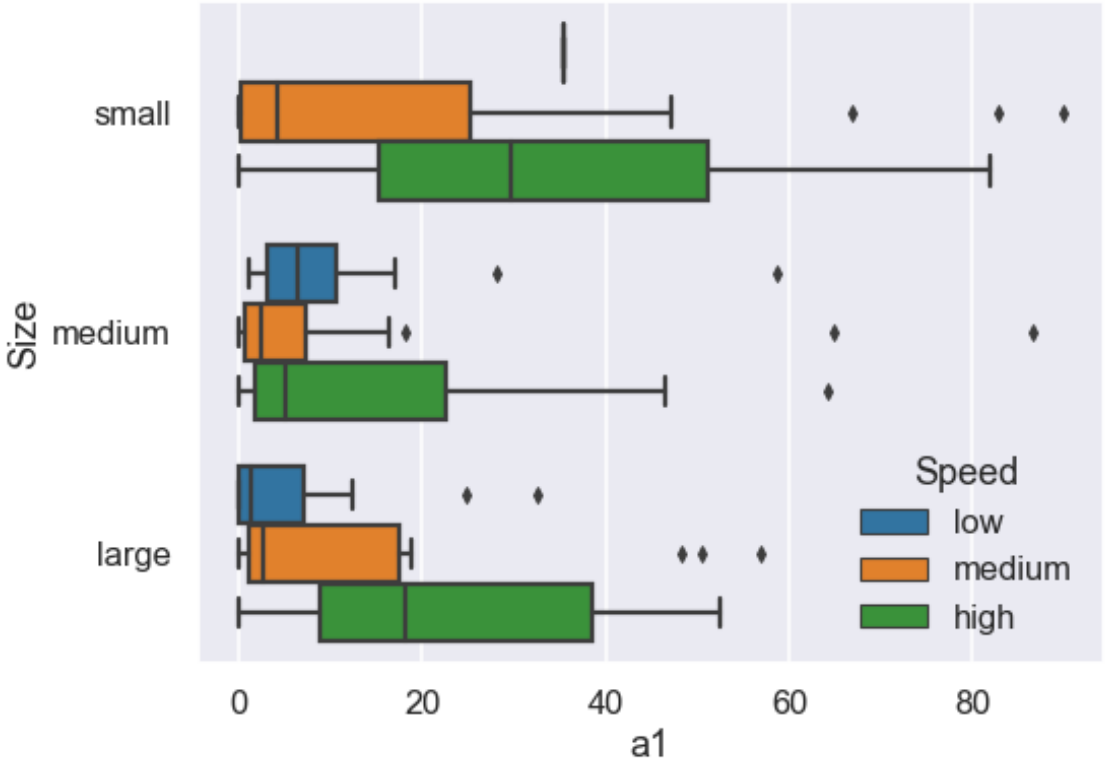
```
sns.boxplot(x="a1", data=df);
```




```
sns.boxplot(x="a1", y="Size", data=df);
```




```
sns.boxplot(x="a1", y="Size", hue="Speed", data=df);
```




- If have $n$ features, then have $n(n-1)$ possible visualisations — gets a bit crazy.

# After Third Pass — Where are we?

- Reviewed each feature — location, spread, shape, issues.
- Identified any correlation among features and with target.
- Located and resolved missing values.
- Generated list of possible feature engineering tasks.

# Resources

# Resources

## Guides

- 1 hour, Youtube on generating seaborn plots — excellent (but wrong on interpretation of box plot)
  `www.youtube.com/watch?v=6GUZXDef2U0&t=1363s`

## Articles on Exploratory Data Analysis

- Exploratory Data Analysis (EDA) and Data Visualization with Python
  `www.kite.com/blog/python/data-analysis-visualization-python/`
- Titanic Survival Dataset Part 1/2: Exploratory Data Analysis (9 min read)
  `www.kaggle.com/mcromao/titanic-exploratory-data-analysis`
- Titanic - Exploratory Data Analysis
  `becominghuman.ai/titanic-survival-dataset-part-1-2-exploratory-data-analysis-5b98f7917913`
- When Should You Delete Outliers from a Data Set?
  `humansofdata.atlan.com/2018/03/when-delete-outliers-dataset`