dm25s1

Topic 04 : Exploratory Data Analysis

Part 02 : EDA Pass2

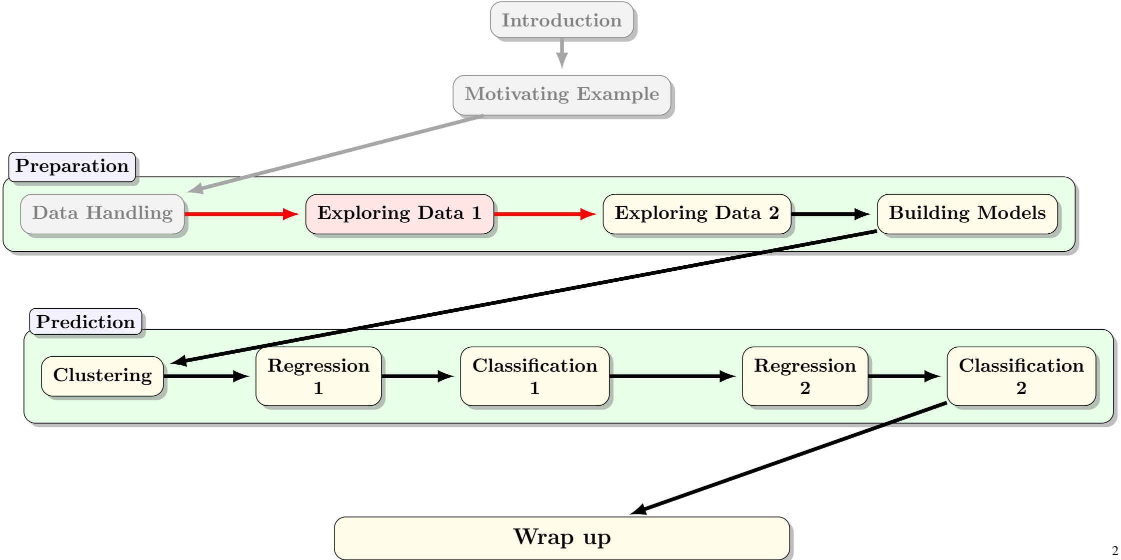
Dr Bernard Butler
Department of Computing and Mathematics, WIT.
(bernard.butler@setu.ie)

Autumn Semester, 2025

Outline

- Feature and target distributions
- Plotting
- After EDA Pass 2

Data Mining (Week 4)



EDA Pass2 — Summary

Acknowledgment

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

Introducing EDA Pass2

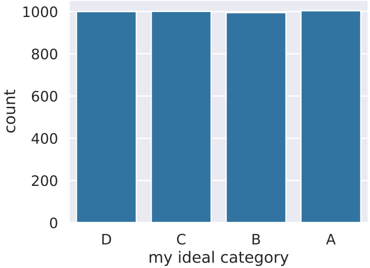
A Selection of Statistical Visualisations and Metrics

		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).



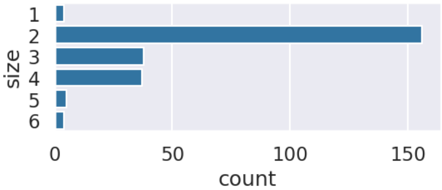
Tools

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

Reality

Tips

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

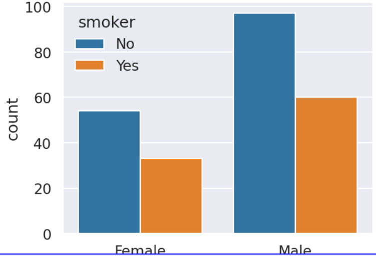


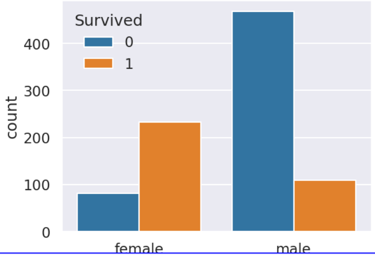
- If `size` was the target, then most models will train towards the majority class (`size=2`).
- If `size` was a feature, then quality of predictor could vary greatly depending on the feature categorical level.
- Consider merge/drop rare category levels.

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%

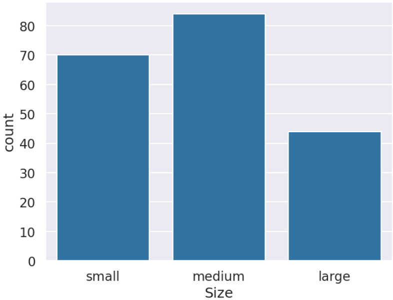
No relationship between `sex` and `smoker`

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`

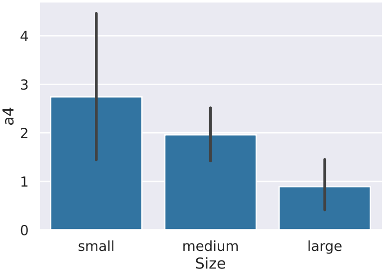
Categorical Variables — Relationship with (Numerical) Target I

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



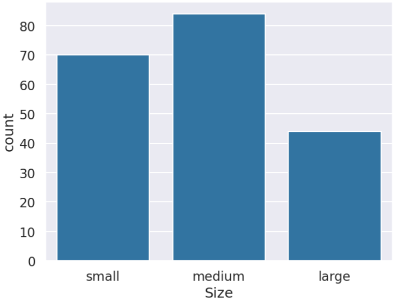
- Shows the counts of observations in each categorical level using bar (height/width).

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



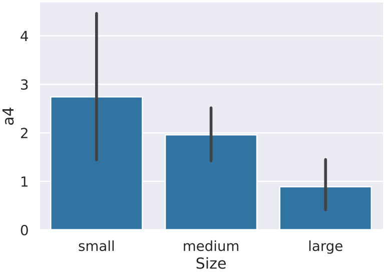
Categorical Variables — Relationship with (Numerical) Target I

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



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

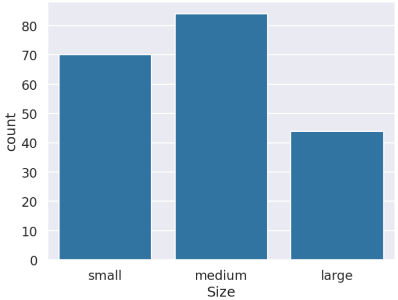
```
sns.catplot(x="Size", y="a4", data=df, kind='bar');
```



- 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.

Categorical Variables — Relationship with (Numerical) Target I

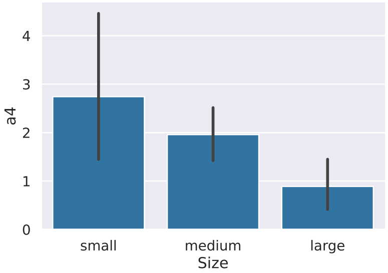
```
sns.countplot(x="Size", data=df);
```



- 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');
```

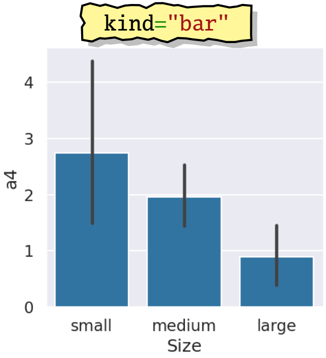


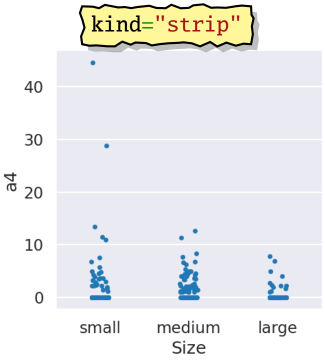
- Shows the average level (mean) and uncertainty (std) of the numerical target (`a4`) in each categorical level of the categorical variable.
- Vertical bar shows 95% confidence interval.

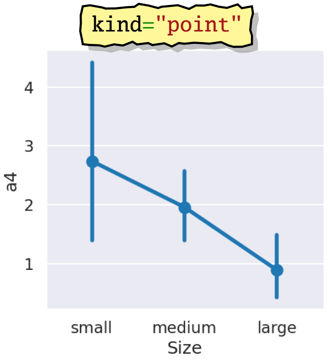
Is it useful?

Categorical Variables — Relationship with (Numerical) Target II

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



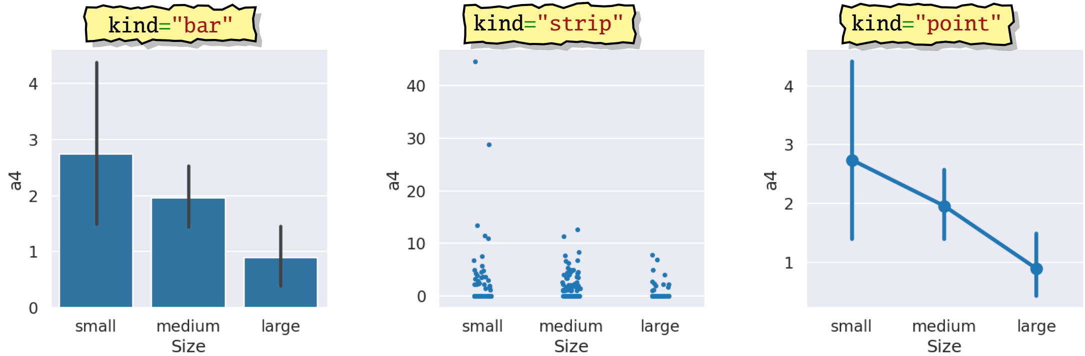




- `bar` and `point` show essentially the same information, but `point` is more compact when comparing multiple categorical features to a continuous target on the same plot.

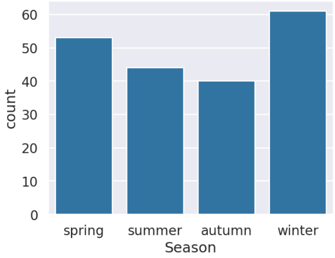
Categorical Variables — Relationship with (Numerical) Target II

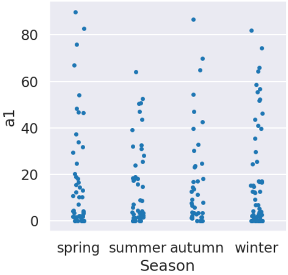
The option `kind` in `catplot` can be:

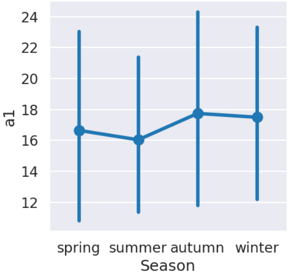


- `bar` and `point` show essentially the same information, but `point` is more compact when comparing multiple categorical features to a continuous target on the same plot.
- `strip` shows individual observations — useful (as in this case) to show that the larger uncertainty in `Size="small"` observations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

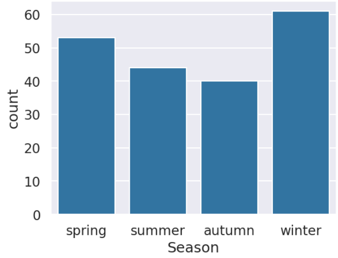
Example — Dataset: `Algae Blooms`, Feature: `Season`, Target: `a1`

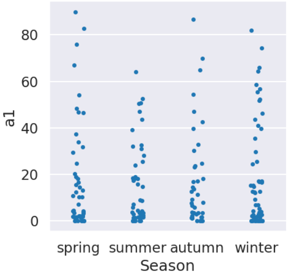


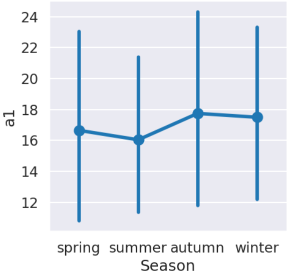




Example — Dataset: `Algae Blooms`, Feature: `Season`, Target: `a1`



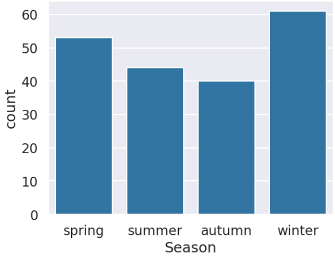


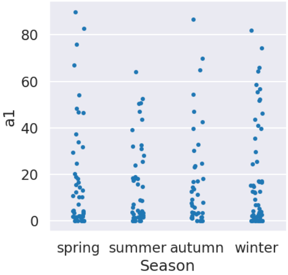


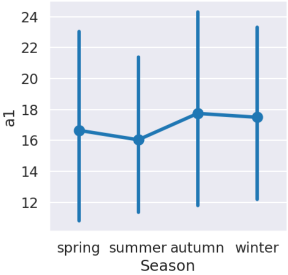

```
df.groupby("Season")["a1"].agg(["mean","count","std"])
```

	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

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







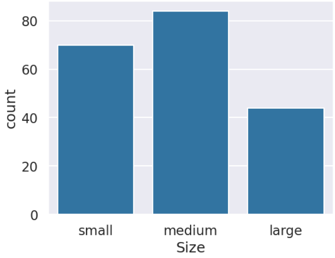

```
df.groupby("Season")["a1"].agg(["mean","count","std"])
```

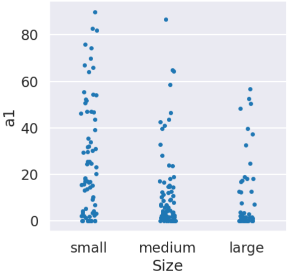
	mean	**count**	**std**
Season	$\bar{x}$	n	σ
spring	16.649057	53	23.093786
summer	16.038636	44	17.920798
autumn	17.745000	40	21.611203
winter	17.498361	61	22.568256

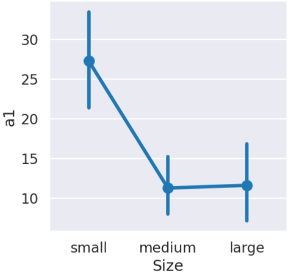
- Countplot shows no issues with feature `Season` — all levels approximately equally represented.
- Catplots show slightly less spread in `a1` for `Season="summer"` observations. (`strip` shows smaller range, `point` shows smaller standard deviation).

$\Rightarrow$ 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.

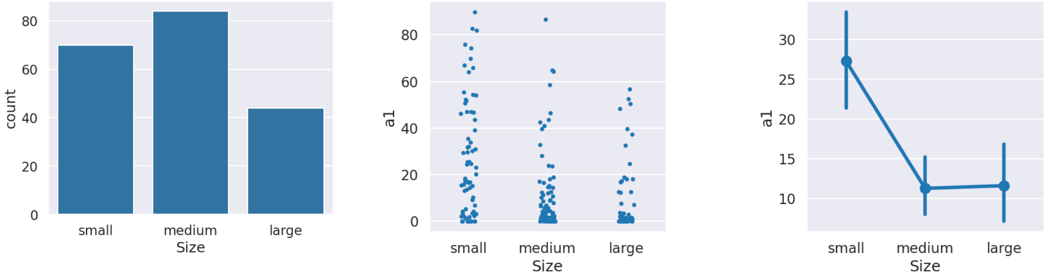
Example — Dataset: `Algae Blooms`, Feature: `Size`, Target: `a1`







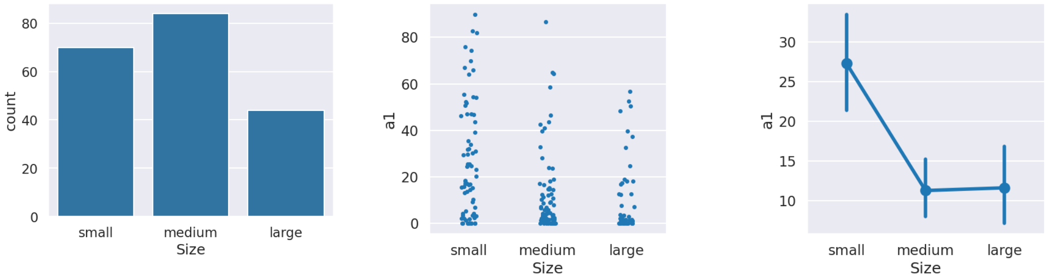
Example — Dataset: `Algae Blooms`, Feature: `Size`, Target: `a1`




```
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

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.

$\Rightarrow$ 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	-1.915795
1	0.641175
2	0.040114
3	0.725531
4	0.252675

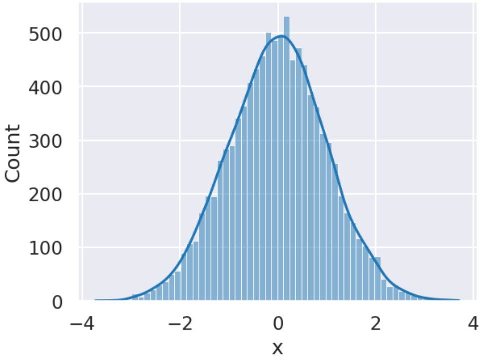
Numerical Variables

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	-1.915795
1	0.641175
2	0.040114
3	0.725531
4	0.252675

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



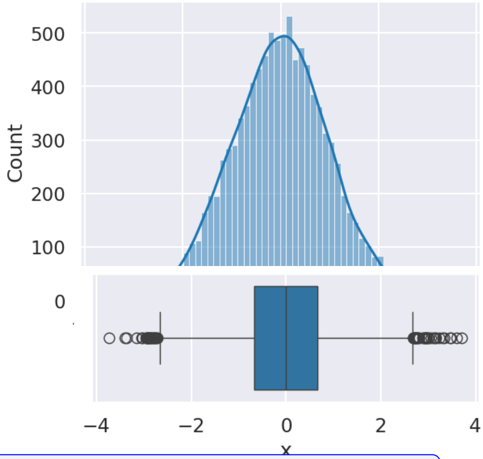
Numerical Variables

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)
```

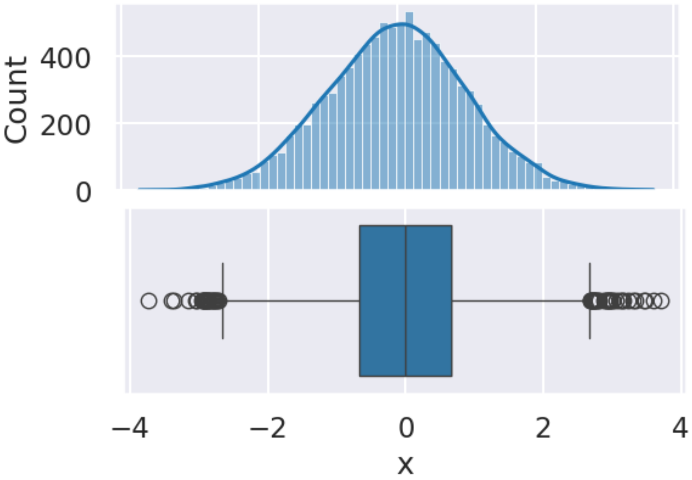
	x
0	-1.915795
1	0.641175
2	0.040114
3	0.725531
4	0.252675

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

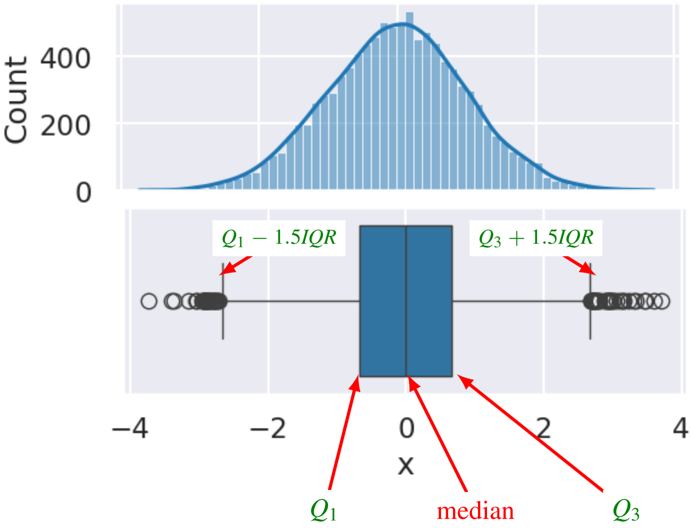



```
sns.boxplot(x="x", data=df2);
```

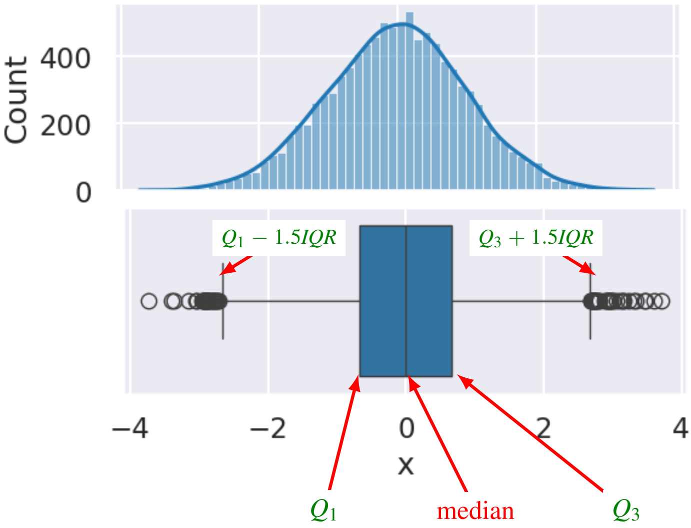
Histplot (Histogram) and Boxplot



Histplot (Histogram) and Boxplot

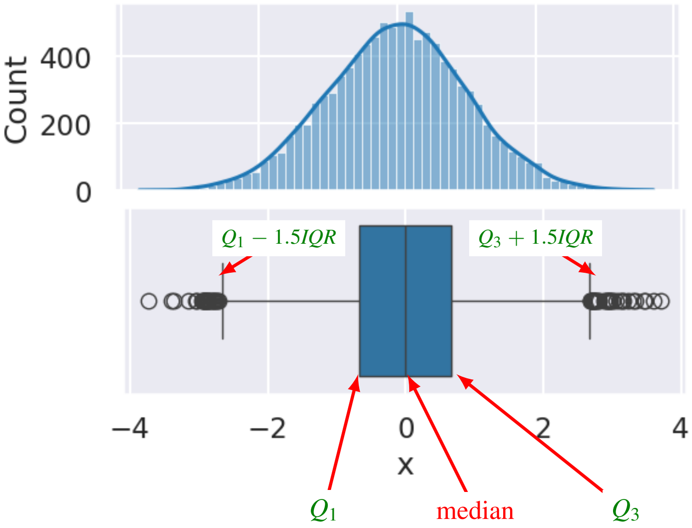


Histplot (Histogram) and Boxplot



- 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.

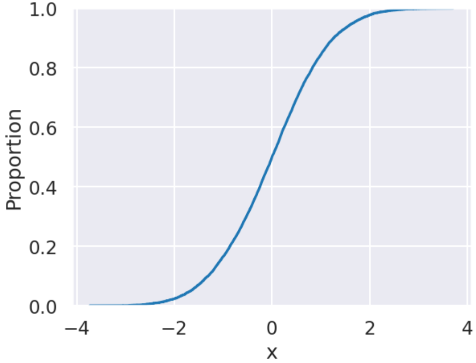
Histplot (Histogram) and Boxplot



- 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.

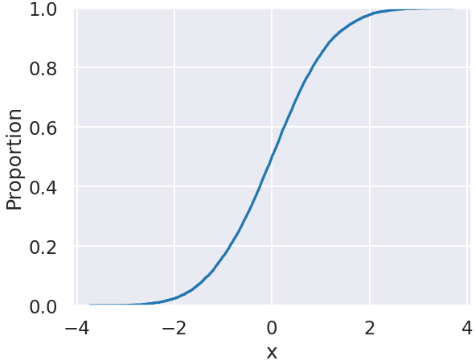
Cumulative Plot and QQ-Plot

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



Cumulative Plot and QQ-Plot

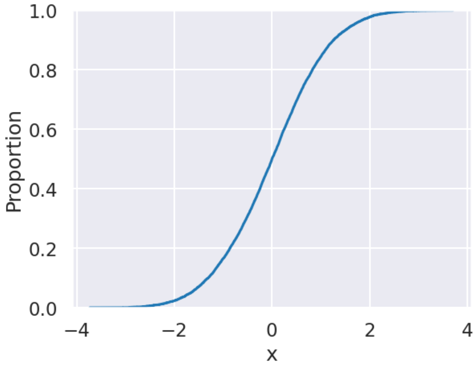
```
sns.ecdfplot(data=df2, x="x");
```



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

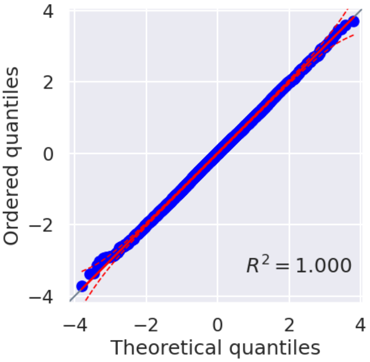
Cumulative Plot and QQ-Plot

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



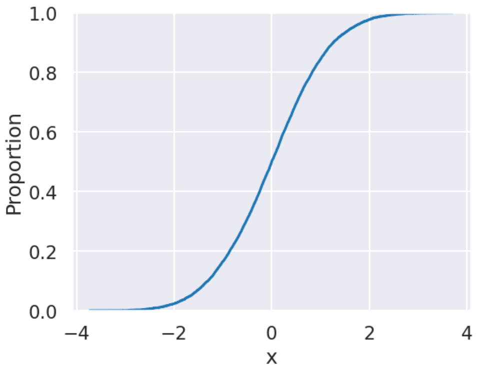
- Represents the proportion of observations less than or equal to given value.

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



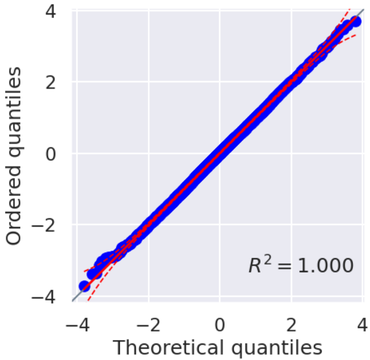
Cumulative Plot and QQ-Plot

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



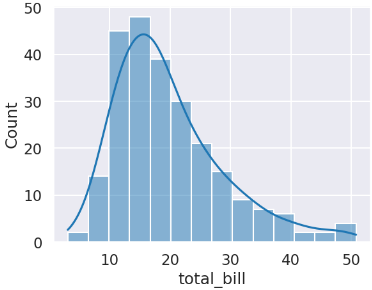
- Represents the proportion of observations less than or equal to given value.

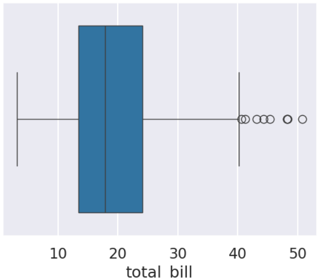
```
import pingouin as pg
pg.qqplot(df2.x);
```

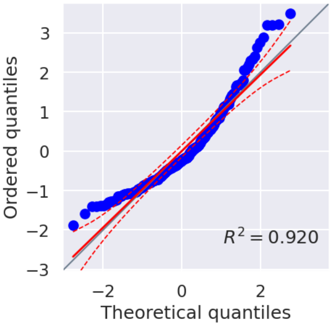


- Plot of observed quantiles against theoretical (assumed normal) quantiles. If both sets of quantiles came from the same distribution, the points lie on a line (approx.).

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



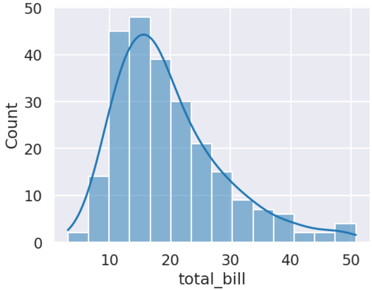


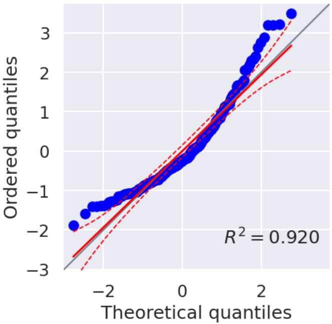



```
df.total_bill.describe()
```

```
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
```

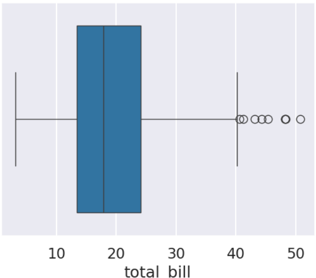
Example — Dataset: `Tips`, Feature: `total_bill`





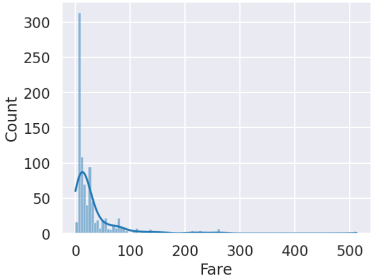

```
df.total_bill.describe()
```

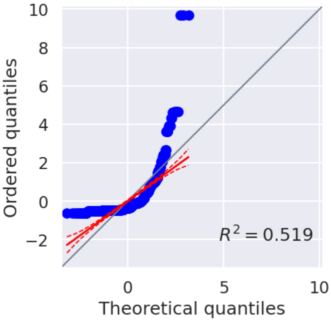
```
count    244.000000
mean      19.785943
std        8.902412
min        3.070000
25%       13.347500
50%       17.795000
75%       24.127500
max       50.810000
Name: total_bill, dtype: float
```

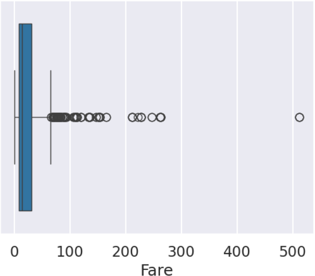


- Data is bell curve shaped, but right skewed (data is more spread out to the right).
- Outliers to the right.
- QQ-Plot indicate that data is not normal, but we could transform it to be more closer to normal.

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



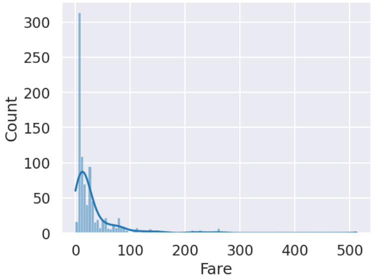


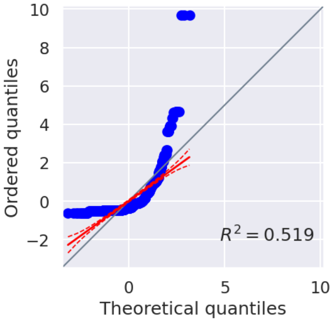


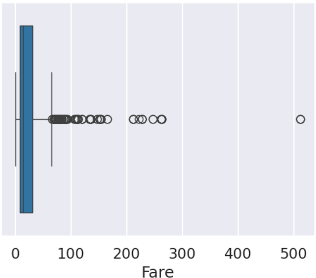

```
df.Fare.describe()
```

```
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
```

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







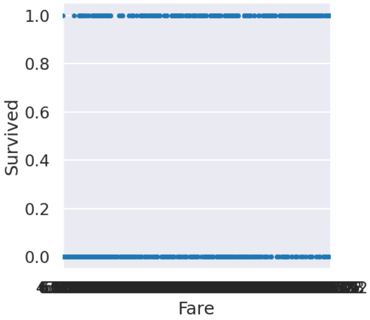

```
df.Fare.describe()
```

```
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

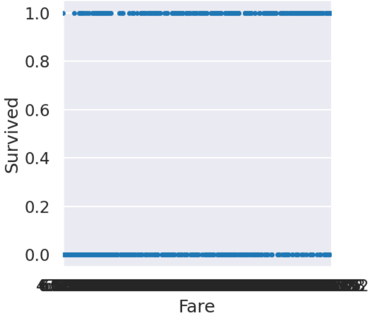
```
df = pd.read_csv("data/train.csv")
sns.catplot(data=df, x="Fare", y="Survived");
```

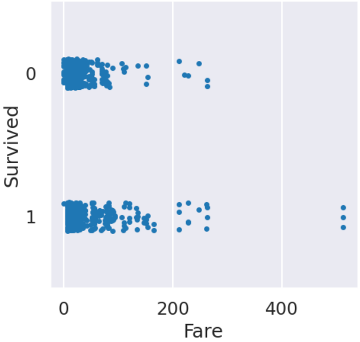


Warning —- Plot Output Depends on Data Assumptions

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

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

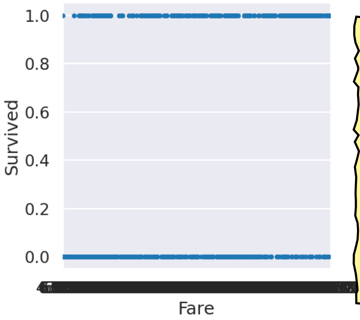




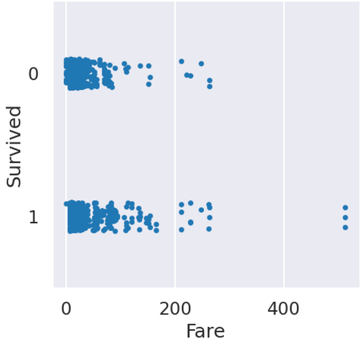
Warning —- Plot Output Depends on Data Assumptions

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

```
df = pd.read_csv("data/train.csv")
df.Survived = 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"`.



Summary

- After reading in the data, exploratory data analysis begins

Summary

- After reading in the data, exploratory data analysis begins
- Pass 1 is all about assessing the structure and cleanliness of the data

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?

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?

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?

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?
- Pass 2 examines the data more closely, in a repeatable fashion

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?
- Pass 2 examines the data more closely, in a repeatable fashion
    1. How are (subsets of) the features and target(s) distributed?

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?
- Pass 2 examines the data more closely, in a repeatable fashion
  1. How are (subsets of) the features and target(s) distributed?
- Pandas and seaborn offer easy-to-use ways of visualising columns, noting

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?
- Pass 2 examines the data more closely, in a repeatable fashion
    1. How are (subsets of) the features and target(s) distributed?
- Pandas and seaborn offer easy-to-use ways of visualising columns, noting
    1. their datatype

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?
- Pass 2 examines the data more closely, in a repeatable fashion
  1. How are (subsets of) the features and target(s) distributed?
- Pandas and seaborn offer easy-to-use ways of visualising columns, noting
  1. their datatype
  2. their cardinality

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?
- Pass 2 examines the data more closely, in a repeatable fashion
  1. How are (subsets of) the features and target(s) distributed?
- 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 within a column