MadeTech

Introduction to Exploratory Data Analysis

By James Poulten

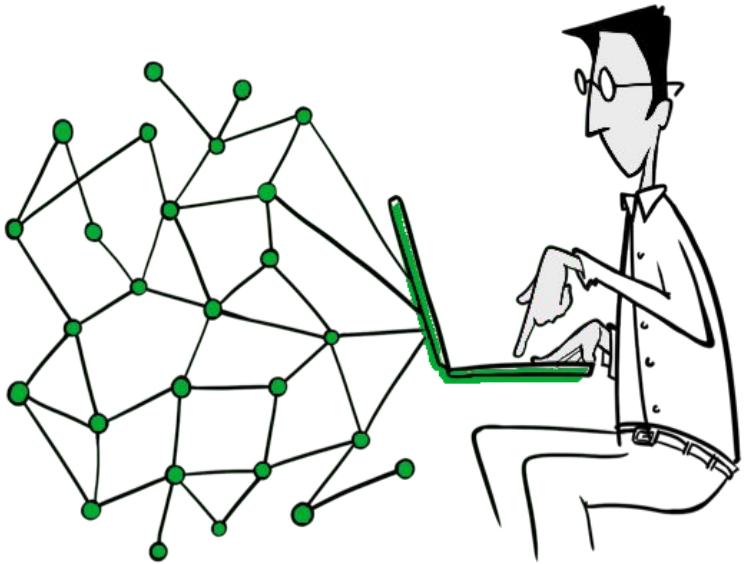
Lead Data Scientist

Why Exploratory Data Analysis?!

Clients hear Data and jump to Analyst.
While the goal is educating them to understand the differences in the data ecosystem, sometimes they will still look to data engineers to provide deeper insight into their data.

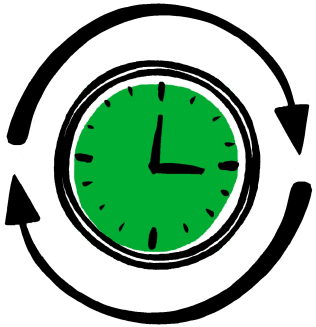
Understanding the data we are given allows us to build better data platforms.
By performing simple interrogations of the data, we can learn where there are quality issues, where there are gaps and where there is room for improvement.

Discover potential within the data.
By better understanding the data we are extracting and loading for your clients, we gain the ability to make meaningful suggestions about .



Outline for the rest of the day

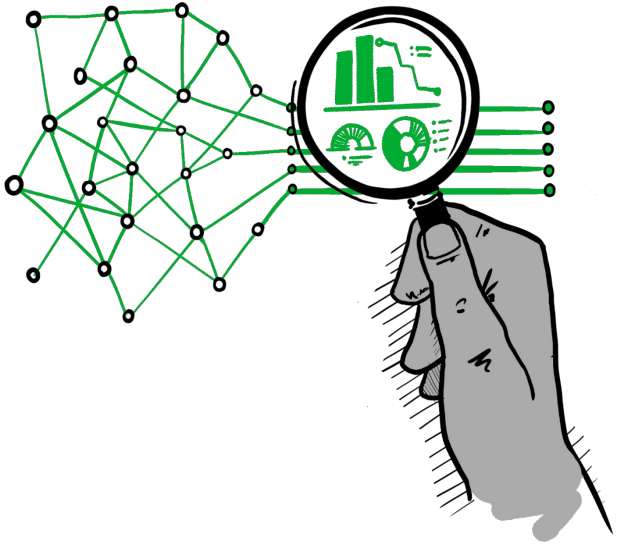
10:00 - 11:30	Introduction to Exploratory Data Analysis (EDA)
11:30 - 12:30	Practical, put the theory to the test
12:30 - 13:30	Lunch
13:30 - 14:00	Visualisations, how to present your data
14:00 - 16:00	Practical, make me some pretty graphs



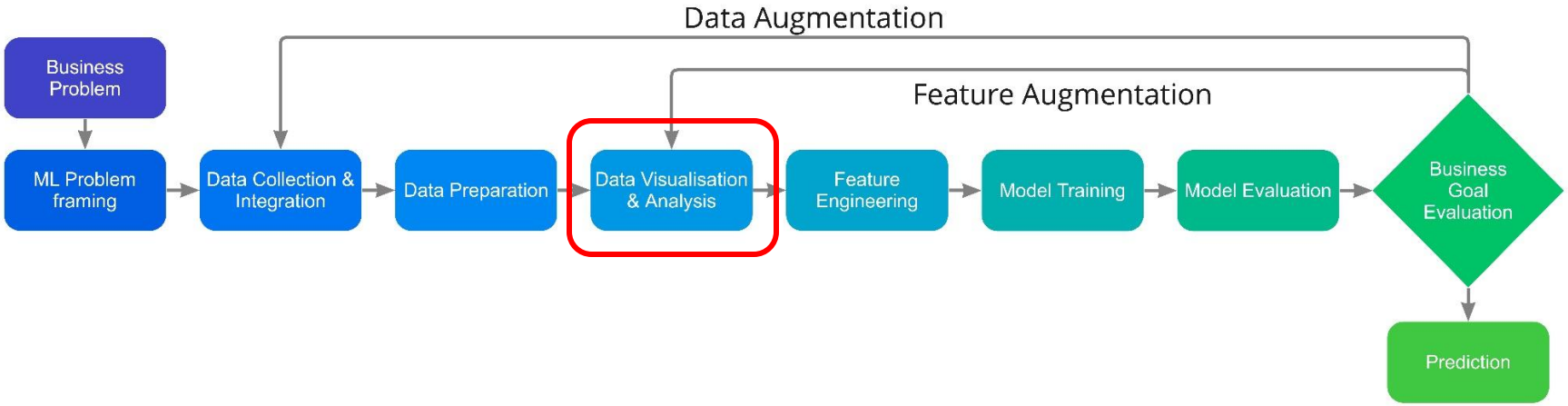
What is Exploratory Data Analysis

Exactly what it says.
An initial or preliminary inspection of the data, looking to ascertain the characteristics, trends and themes of the data set with the goal of better enabling future analysis.

An opportunity to better understand the problem.
After performing exploratory data analysis, you will oftentimes have a better understanding of the clients data than even the analysts on their team. You will be able to advise them on what additions they should make to their pipeline and will likely be able to provide them with insight into their data.



It is a key part to any ML project



https://github.com/madetech/data-101/blob/main/guides/machine_learning_workflow.md

Where do we begin?

You would be surprised at the amount of low hanging fruit.

You are probably the first person to be looking at the data.
We are tasked with setting up data infrastructure and often pull numerous data sources together.
Chances are, nobody else has had the opportunity or time to do that.

So don't over think it and go for the basics first.
It really is that simple

Titanic data set from Kaggle for its worked examples

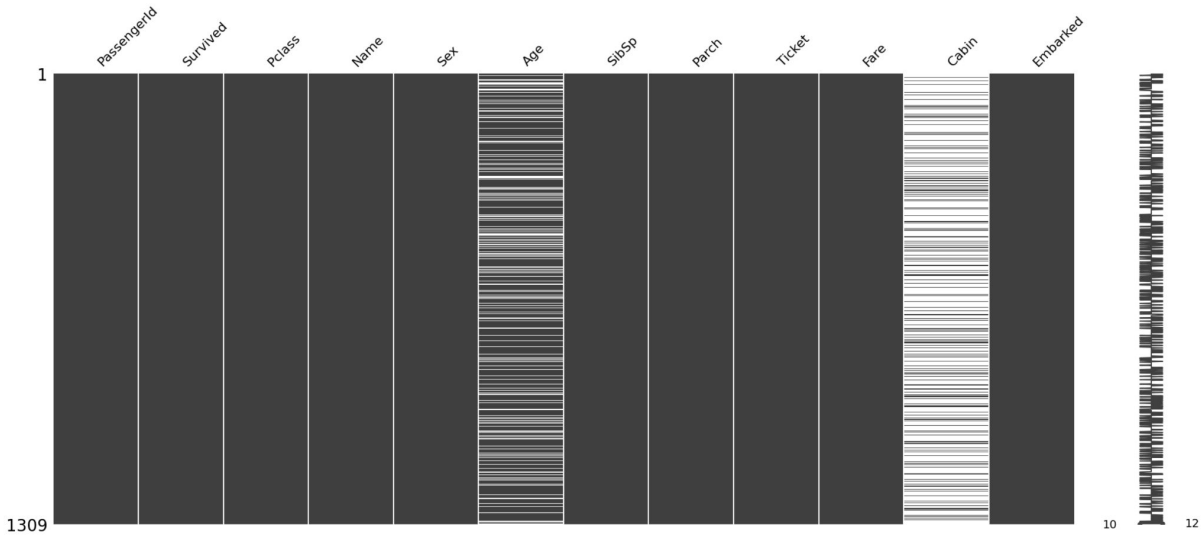
Identifying missing data

There are plenty of ways to do this. But visulisations are excellent at driving the point home

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

```
PassengerId       0
Survived          0
Pclass            0
Name              0
Sex               0
Age             264
SibSp             0
Parch             0
Ticket            0
Fare              1
Cabin          1019
Embarked          2
```

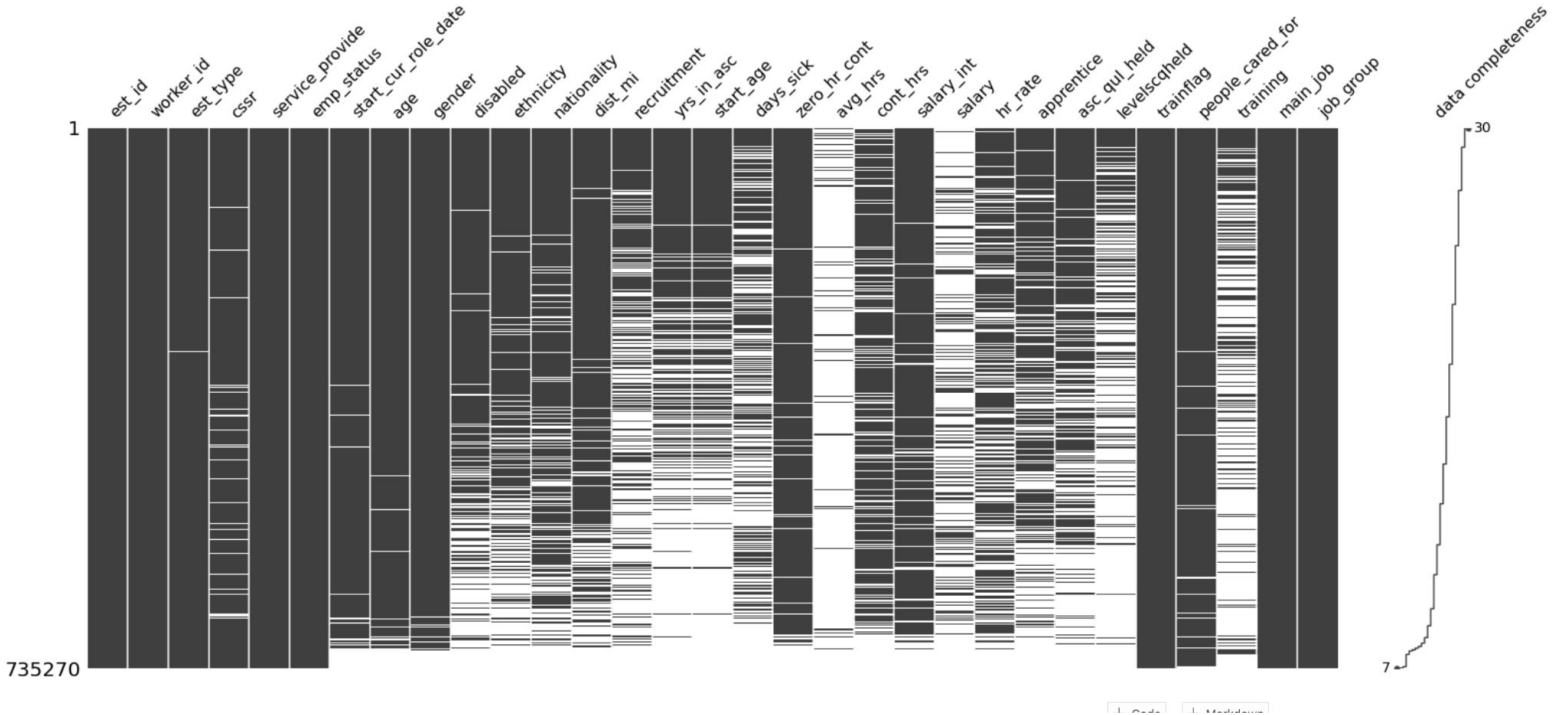
```
import missingno as msno
msno.matrix(df)
```



Titanic data set from Kaggle for its worked examples

Identifying missing data

The client has probably never seen their data like this before!



MadeTech

Check for duplicates

Is there a problem with the unjust process?

```
df.duplicated()
```

This command actually creates a **bool** mask - so let's apply that to the df

```
df.loc[dp.duplicated()]
```

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1309	47	0	3	Lennon, Mr. Denis	male	NaN	1	0	370371	15.5000	NaN	Q
1310	424	0	3	Danbom, Mrs. Ernst Gilbert (Anna Sigrid Maria ...	female	28.0	1	1	347080	14.4000	NaN	S
1311	392	1	3	Jansson, Mr. Carl Olof	male	21.0	0	0	350034	7.7958	NaN	S
1312	54	1	2	Faunthorpe, Mrs. Lizzie (Elizabeth Anne Wilkin...	female	29.0	1	0	2926	26.0000	NaN	S
1313	990	1	3	Braf, Miss. Elin Ester Maria	female	20.0	0	0	347471	7.8542	NaN	S

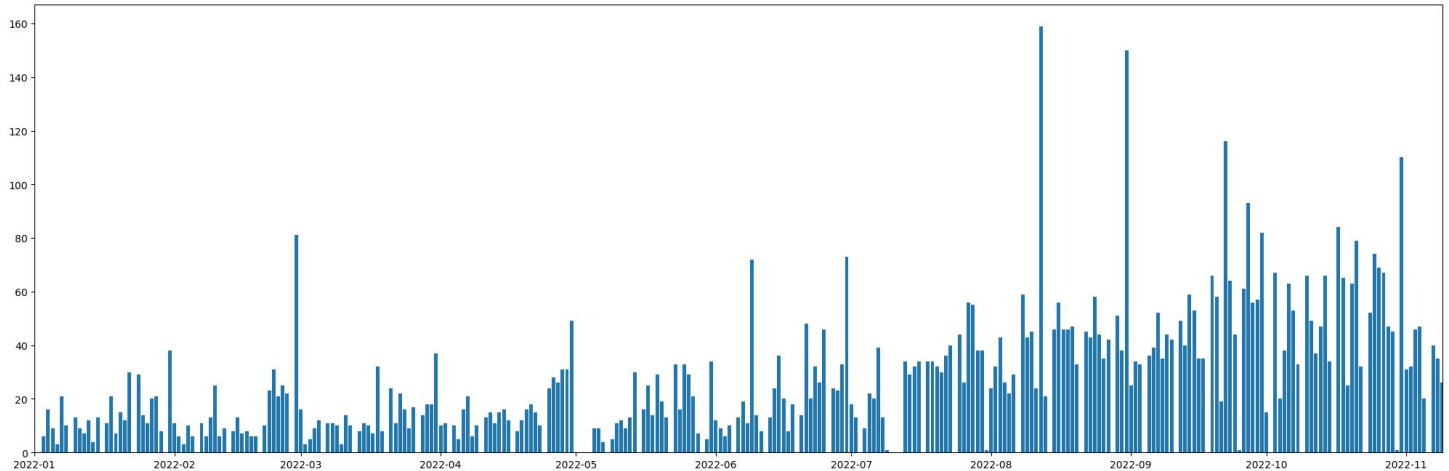
```
df.drop_duplicates()
```

At this point the Data Scientist just kicks it back up to the engineers and plows ahead.
Infact, you may notice a Data Scientist consistently finding problems.

Look at the timestamps!

You can impart so much operational insight just by knowing when your users… use the service!

```
plt.bar(
   x=df["Date"][df["Date"]>"2022-01-01"].value_counts().sort_index().index,
   height=df["Date"][df["Date"]>"2022-01-01"].value_counts().sort_index().values
)
```

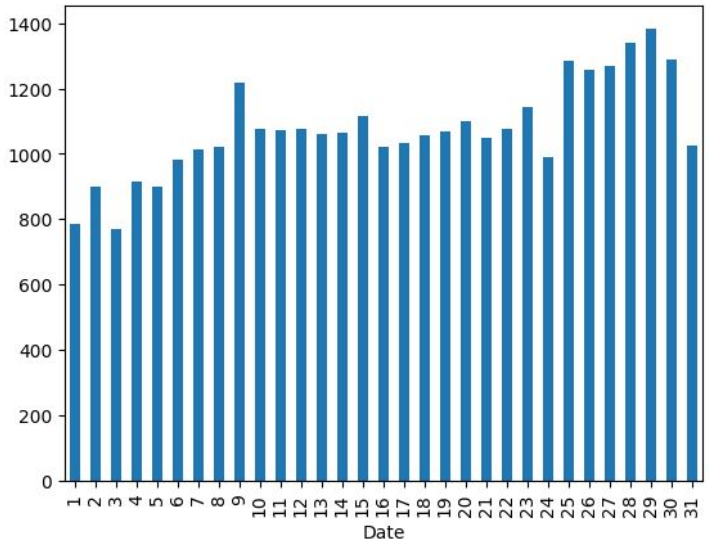


Just by looking at this graph, we know when customers are making more purchases. And when they didn't.

Look at the timestamps!

You can impart so much operational insight just by knowing when your users... use the service!

```
df["Date"].dt.day.value_counts().sort_index().plot(kind="bar")
```

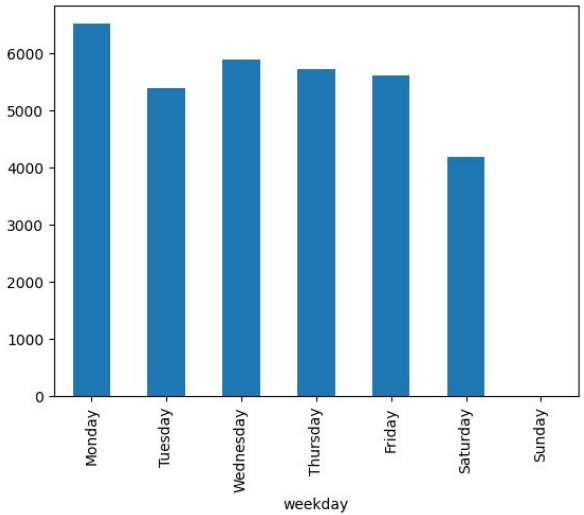


Or when spending happens in the month

Look at the timestamps!

You can impart so much operational insight just by knowing when your users... use the service!

```
week_day_order = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday']
tdf["weekday"].value_counts().reindex(week_day_order).plot(kind="bar")
```



Or on what days of the week.

Instantly, we have questions and insights

You don’t have to have all the answers.

Do we understand why we have missing data?

- Is there an issue with our collection methods?
- Have we imported and joined all the data correctly?
- Do we need to add requirements to our collection process?

Do we know why our users use the product when they do?

- Are there external factors
- Are there trends that were previously unknown

Do we want or need to fix this?

- Is there another data source we can draw upon?
- Is this demonstrating a wider problem with the collection process?
- Is our ETL process working as intended?
- Are there transformations that we can apply that fill the gaps?

Let’s take a break

Back in 10 mins

Let’s dive a little deeper

Get to know the data. Start with the basics

We are going to discuss some basic “types” of data.

But not the generic “types” - Data science types

- Categorical data
- Ordinal data
- Continuous vs Discrete data

How do we explore these different types of data?

What kind of data do you have?

Get to know the data. Start with the basics

```
df.head()
df.info()
df.describe()
```

	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	7.25	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	7.925	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.05	NaN	S

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1309 entries, 0 to 1308
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  1309 non-null   int64
 1   Survived     1309 non-null   int64
 2   Pclass       1309 non-null   object
 3   Name         1309 non-null   object
 4   Sex          1223 non-null   object
 5   Age          1132 non-null   float64
 6   SibSp        1309 non-null   int64
 7   Parch        1309 non-null   object
 8   Ticket       1308 non-null   object
 9   Fare         982 non-null    object
 10  Cabin        622 non-null    object
 11  Embarked     889 non-null    object
dtypes: float64(1), int64(3), object(8)
memory usage: 122.8+ KB
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	1314.000000	1314.000000	1314.000000	1050.000000	1314.000000	1314.000000	1313.000000
mean	653.958904	0.378234	2.296804	29.860638	0.499239	0.384323	33.223181
std	378.271519	0.485131	0.837182	14.391962	1.040131	0.864338	51.674830
min	1.000000	0.000000	1.000000	0.170000	0.000000	0.000000	0.000000
25%	327.250000	0.000000	2.000000	21.000000	0.000000	0.000000	7.895800
50%	653.500000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	981.750000	1.000000	3.000000	39.000000	1.000000	0.000000	31.275000
max	1309.000000	1.000000	3.000000	80.000000	8.000000	9.000000	512.329200

Types to look out for: Categorical

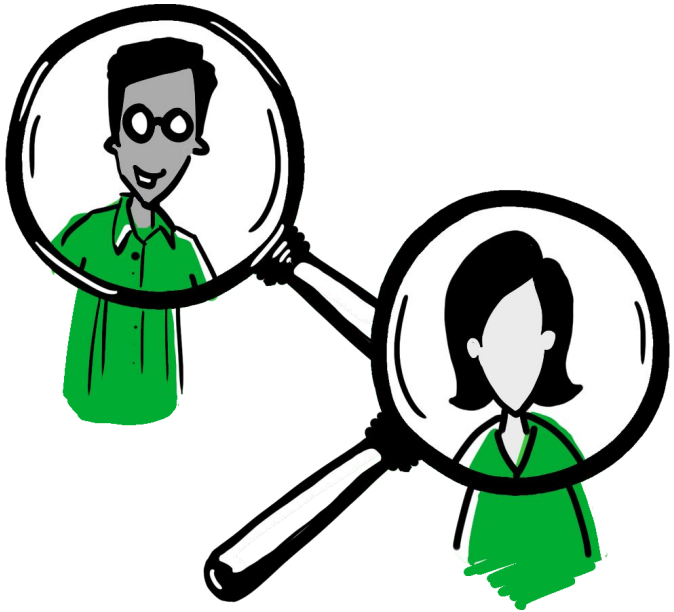
Categorical: the column contains one of a limited number of options.

```
for c in pdf.columns:
    print(f"{c}: {len(pdf[c].unique())}")
```

```
PassengerId: 1309
Survived: 2
Pclass: 3
Name: 1307
Sex: 2
Age: 99
SibSp: 7
Parch: 8
Ticket: 929
Fare: 282
Cabin: 187
Embarked: 4
```

```
pdf["Sex"].unique()
```

```
array(['male', 'female'], dtype=object)
```



Other examples:

- Car make
- Colour
- Species

In each example, there is no intrinsic order to the list of values

Types to look out for: Ordinal

Ordinal: values within the categorical data have a natural order.

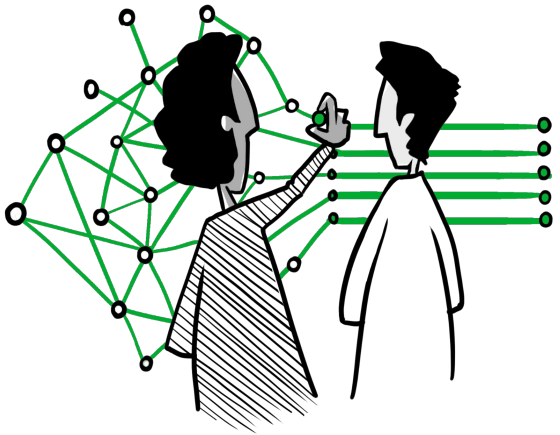
```
df["Pclass"].unique()
```

```
array([3, 1, 2])
```

Other examples:

- Army rank
- Level of education
- Salary band
- Satisfaction rating (number of stars)

The value in the list has an intrinsic relation to other values in the list



Types to look out for: Continuous (float)

Continuous: data that may take on ANY value

We can identify these because they will have a far larger number of distinct values

```
for c in pdf.columns:
   print(f"{c}: {len(pdf[c].unique())}")
```

```
PassengerId: 1309
Survived: 2
Pclass: 3
Name: 1307
Sex: 2
Age: 99
SibSp: 7
Parch: 8
Ticket: 929
Fare: 282
Cabin: 187
Embarked: 4
```

Other examples:

- Area
- Height
- Weight
- Temperature
- Timestamp

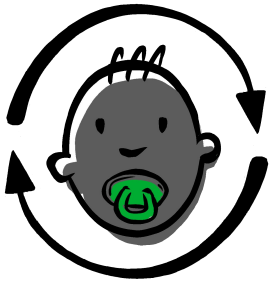
Theoretically, the value may sit anywhere on the scale.

Types to look out for: Discrete (int)

Continuous: data that must take certain values

```
for c in pdf.columns:
   print(f"{c}: {len(pdf[c].unique())}")
```

```
PassengerId: 1309
Survived: 2
Pclass: 3
Name: 1307
Sex: 2
Age: 99
SibSp: 7
Parch: 8
Ticket: 929
Fare: 282
Cabin: 187
Embarked: 4
```



Other examples:

- Age
- Number of days
- Number of children
- Number of sales

Often, it does not make sense to break this into smaller denomination.
You can't have half a child

Univariate analysis

How do individual features “look” how are they distributed?

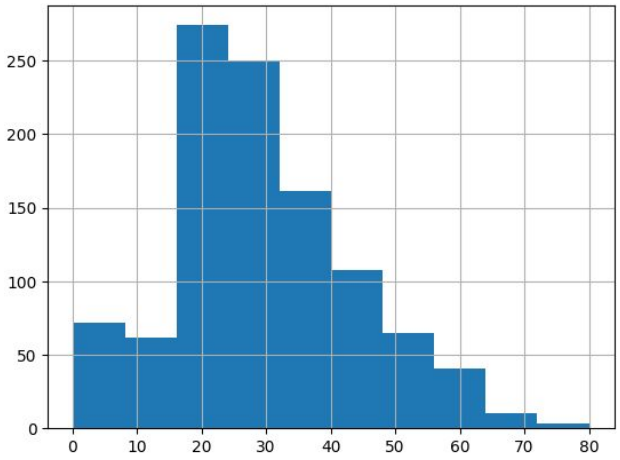
We look at Distributions.

- Histograms - how the values are distributed over the range?
- Box plots - what is the mean values and what are the quartiles?
- Bar plots - how many of each category do we have?

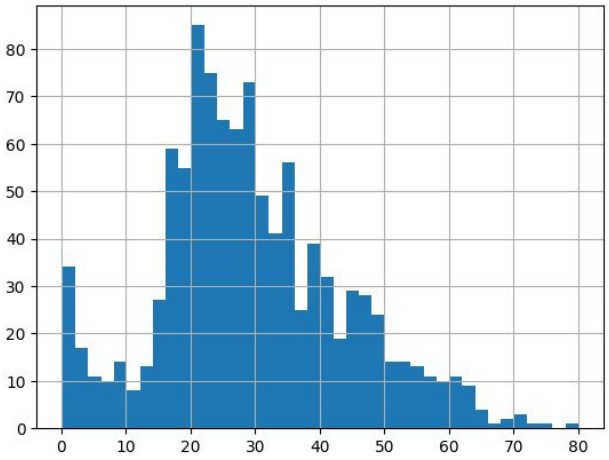
Univariate – Continuous or Discrete data

The Histogram - Values are sorted into bins and counted.

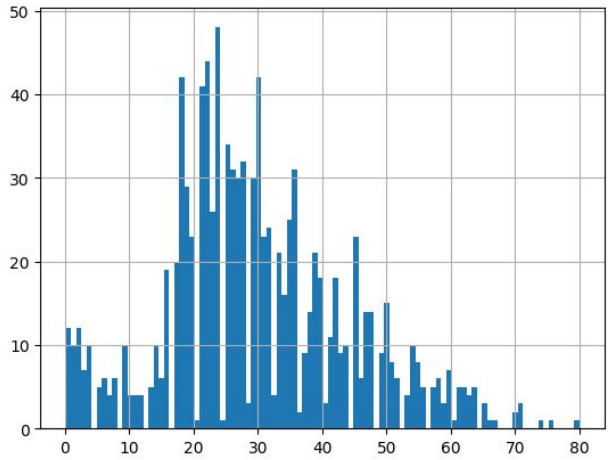
```
df["Age"].hist()
```




```
df["Age"].hist(bins=10)
```




```
df["Age"].hist(bins=40)
```

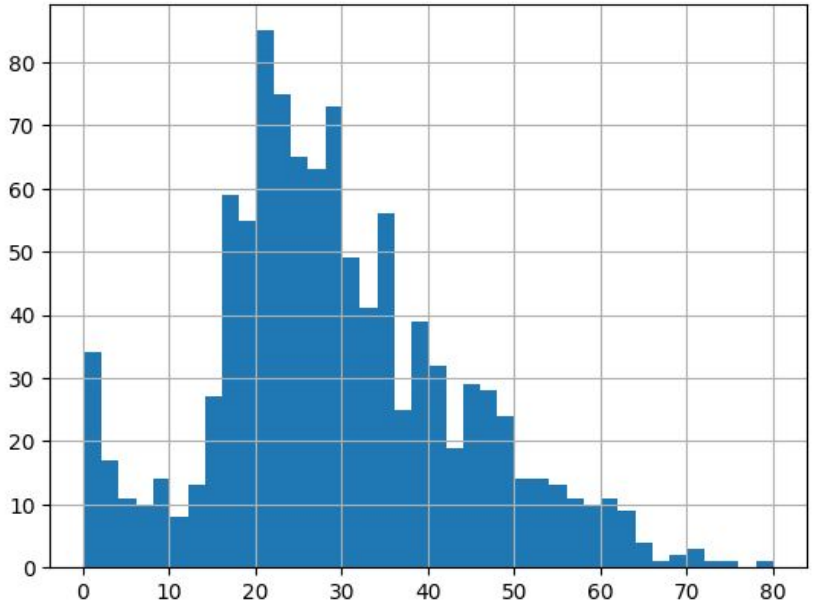



```
df["Age"].hist(bins=100)
```

Univariate – Continuous or Discrete data

The Histogram - Values are sorted into bins and counted.

```
df["Age"].hist()
```

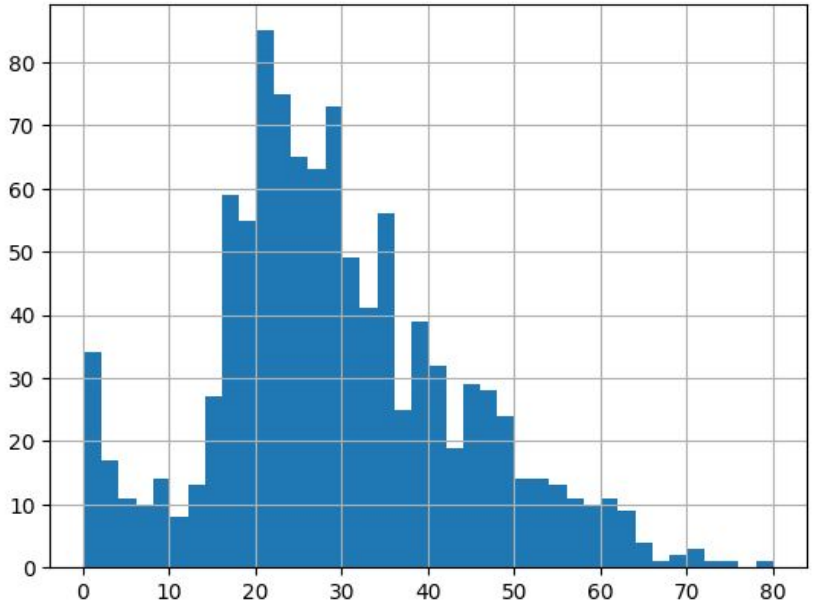



```
df["Age"].hist(bins=40)
```

Univariate – Continuous or Discrete data

The Histogram - Values are sorted into bins and counted.

```
df["Age"].hist()
```



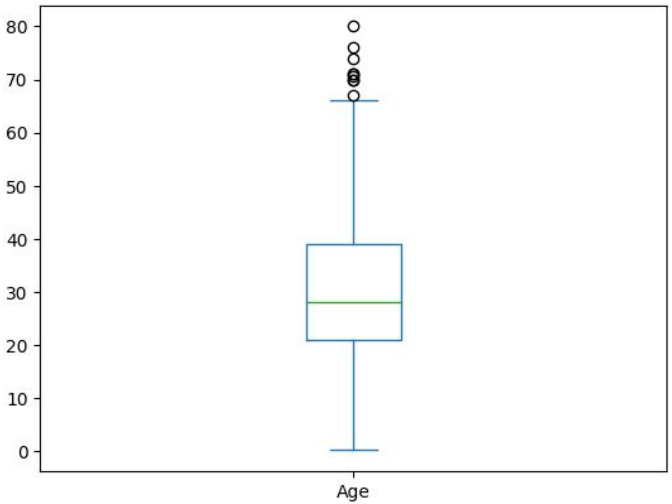
- Most passengers are between 15 and 45
- There were quite a few babies (<3 yo) on the titanic
- The oldest person was 80

```
df["Age"].hist(bins=40)
```

Univariate – Continuous or Discrete data

The Boxplot - The box extends from the 1st quartile to the 3rd quartile, with the mean at the median (2nd quartile). The Whiskers depict the interquartile range

```
df["Age"].plot(kind="box")
```




```
df["Age"].describe()
```

```
count    1046.000000
mean       29.881138
std        14.413493
min         0.170000
25%        21.000000
50%        28.000000
75%        39.000000
max        80.000000
Name: Age, dtype: float64
```

What in the world are quartiles?!

Quartiles divide the data into 4 parts - or quartiles - when ordered smallest to largest.

Q1 - Middle number way between minimum andMedian. 25% of data is below this point.

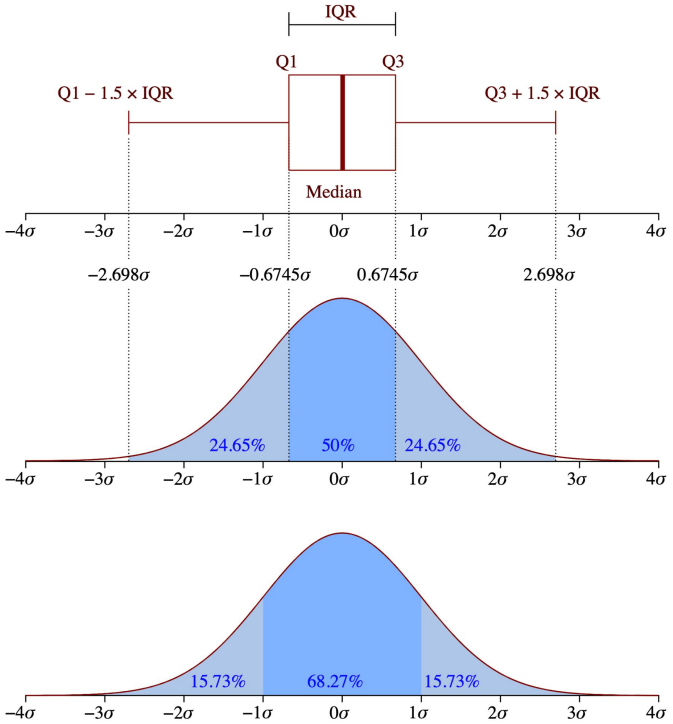
Q2 - The median value. 50% of data is below this point

Q3 - Middle number between the mean and The maximum. 75% of the data is below this point

The IQR (Interquartile Range)
The distance between Q1 and Q3

IQR = Q3 - Q1

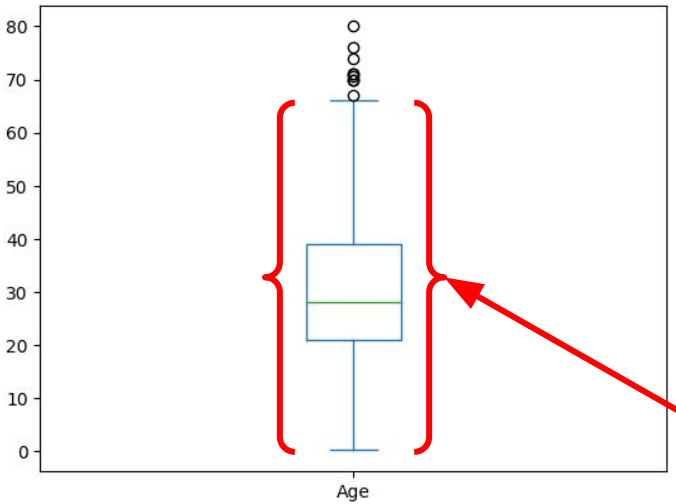
But what does this MEAN!
Well... points outside of the whiskers are **outliers** in the data set.



Univariate analysis – Box plots

The Boxplot - The box extends from the 1st quartile to the 3rd quartile, with the mean at the median (2nd quartile). The Whiskers depict the interquartile range

```
df["Age"].plot(kind="box")
```




```
df["Age"].describe()
```

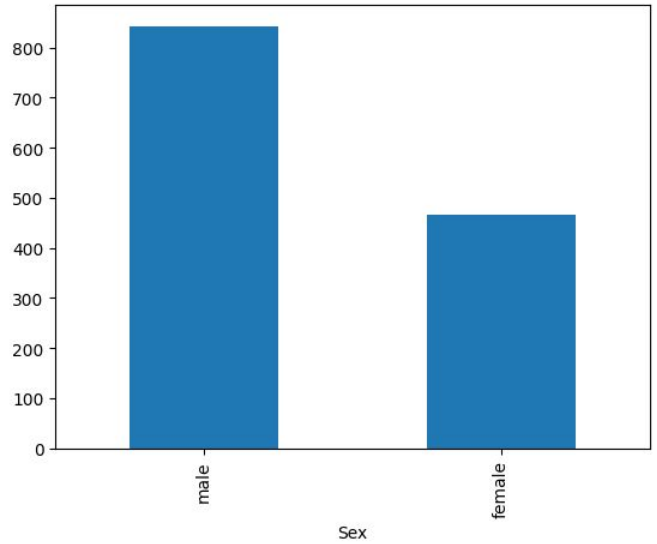
```
count    1046.000000
mean       29.881138
std        14.413493
min         0.170000
25%        21.000000
50%        28.000000
75%        39.000000
max        80.000000
Name: Age, dtype: float64
```

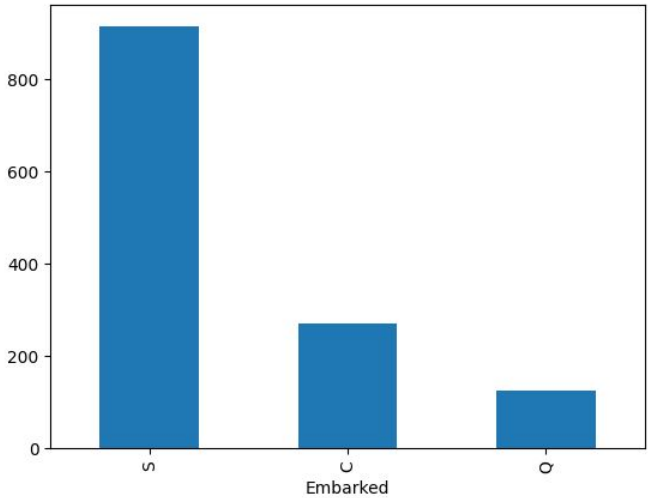
99.3% of you 1046 data points are here!

Univariate – Categorical Data

The Bar plot - Simply, a count of each category, represented as a bar.

```
df["<feature name>"].value_counts().plot(kind="bar")
```

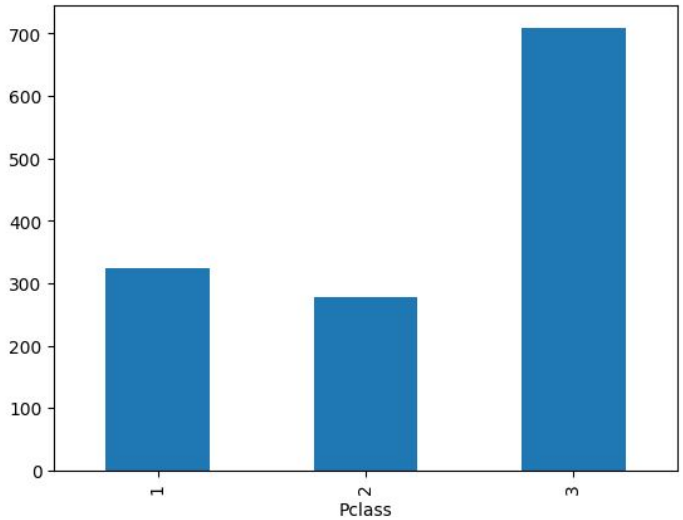




Univariate – Ordinal data

The Bar plot, but ordered - A count of each category, sorted and represented as a bar.

```
df["<feature name>"].value_counts().sort_index().plot(kind="bar")
```



Let’s take a break

Back in 10 mins

Bivariate analysis – when data gets together

How does one feature interact with another?

For Continuous and Discrete data values, we can check their correlations

- Scatter plots - plotting data as xy points on a graph
- Correlation plots

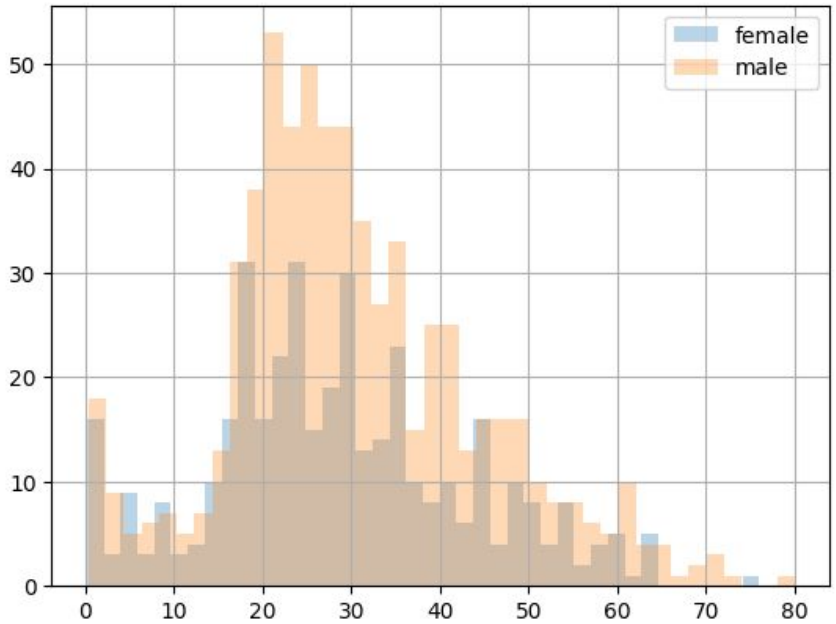
For categorical and ordinal, we use the same plots but add a group by components

For a mixture, we group the data then plot the distributions

Adding categories to distributions

How does the age distribution relate to the sex of the passenger?

```
df.groupby("Sex")["Age"].hist(bins=40, alpha=0.3, legend=True)
```



We can use the describe method to pull out specific numbers

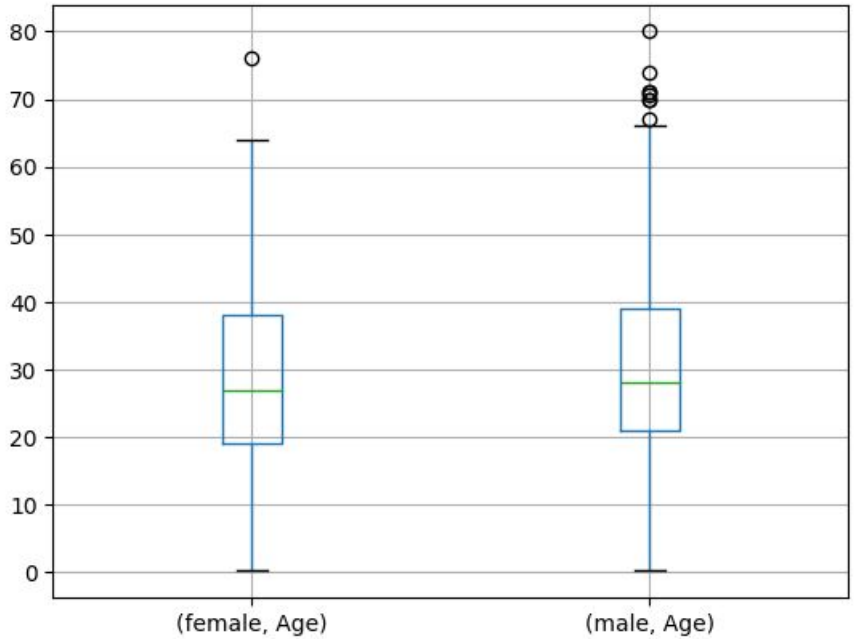
```
df.groupby("Sex")["Age"].describe()
```

Sex	count	mean	std	min	25%	50%	75%	max
female	388.0	28.687088	14.576962	0.17	19.0	27.0	38.0	76.0
male	658.0	30.585228	14.280581	0.33	21.0	28.0	39.0	80.0

Adding categories to distributions – Box plot

How does the age distribution relate to the sex of the passenger?

```
df[["Age", "Sex"]].groupby("Sex").boxplot(subplot=False)
```



We can use the describe method to pull out specific numbers

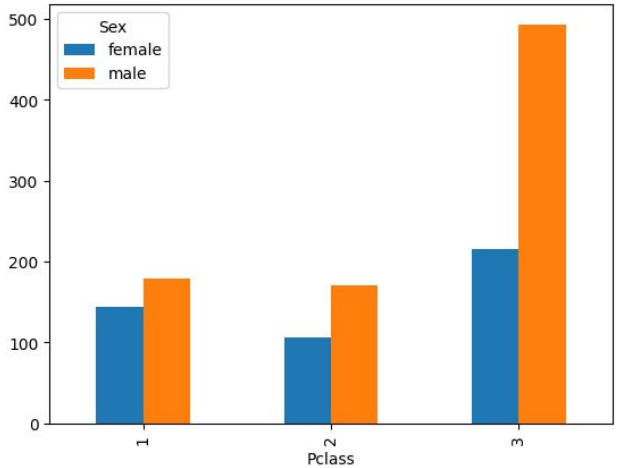
```
df.groupby("Sex")["Age"].describe()
```

Sex	count	mean	std	min	25%	50%	75%	max
female	388.0	28.687088	14.576962	0.17	19.0	27.0	38.0	76.0
male	658.0	30.585228	14.280581	0.33	21.0	28.0	39.0	80.0

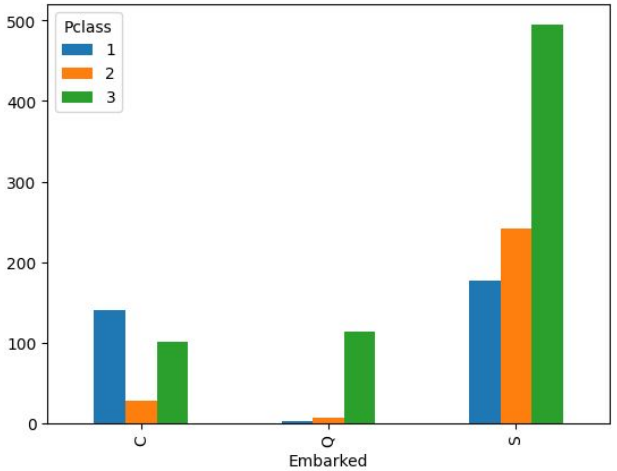
Categorical barcharts

Incredibly useful for direct comparisons - but a bit of a pain to generate

```
df.groupby("Sex")["Pclass"].value_counts().unstack(level= 0).plot(kind="bar")
```



So we can straight away see that 3rd class is more than 2 thirds man



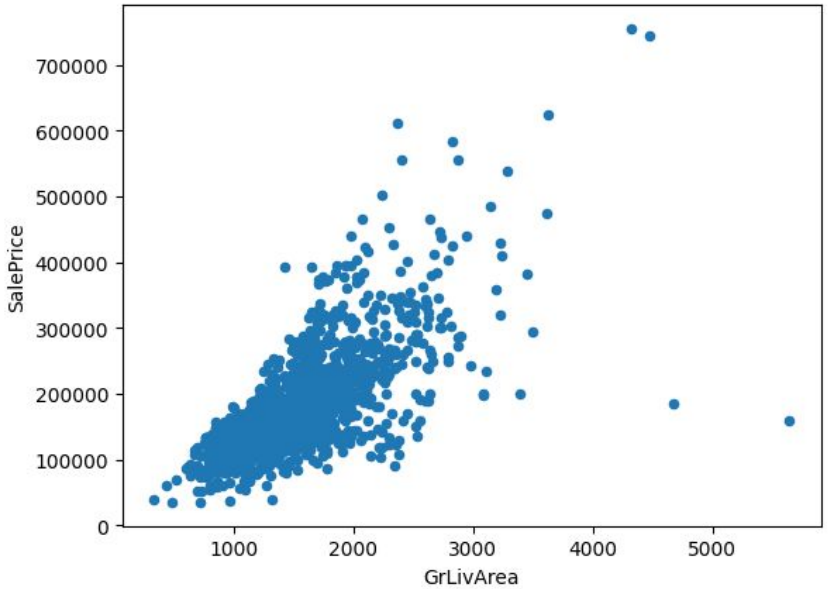
Or that 1st class passengers mostly embarked at Cherbourg or Southampton

Port of Embarkation:
C = Cherbourg
Q = Queenstown
S = Southampton

Two continuous features

No need for fancy maths. We are the hardwired to see patterns.

```
df.plot(kind="scatter", x="GrLivingArea", y="SalePrice")
```



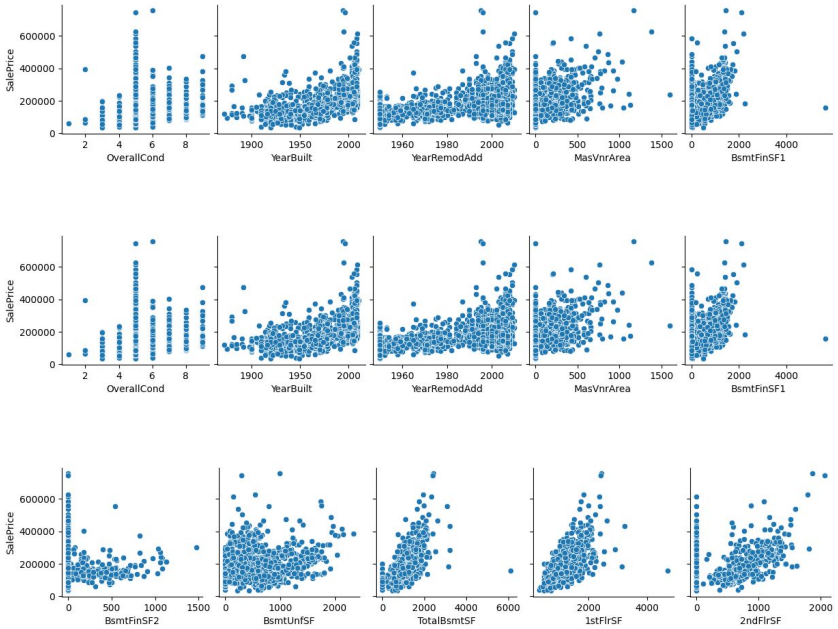
We can see:

- As general living area increases, the cost of the house increases.
- Most houses have less than 2200 sq ft of living space (this is US data - im sorry)
- We can see, very clearly, the outliers in the data

I had to bring in another data set for this example. So this a House Prices regression modeling data set.

Two continuous features

The seaborn library has tons of excellent predefined visualisations!



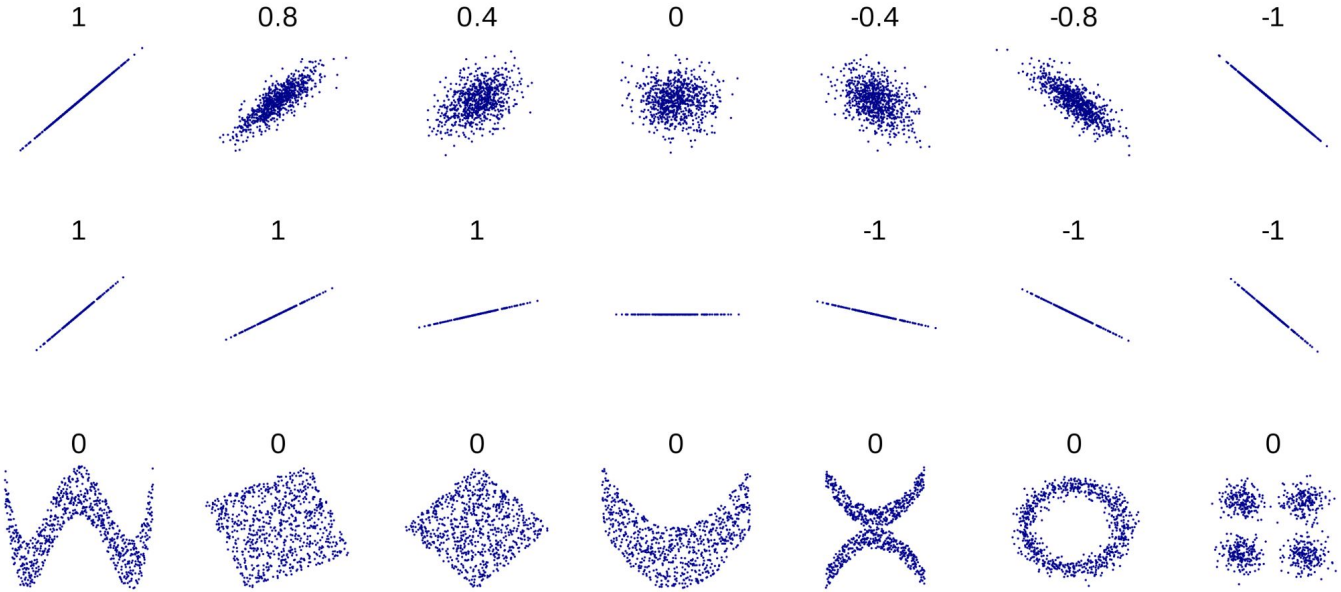

```
import seaborn as sns

df_num = house_df.select_dtypes(
      include = ['float64', 'int64']
)

for i in range(0, len(df_num.columns), 5):
      sns.pairplot(
            data=df_num,
            x vars=df num.columns[i:i+5],
            y_vars=['SalePrice']
      )
```

Correlation between features

These are often described as trends



Correlation between features

The standard pandas output is quite hard to read

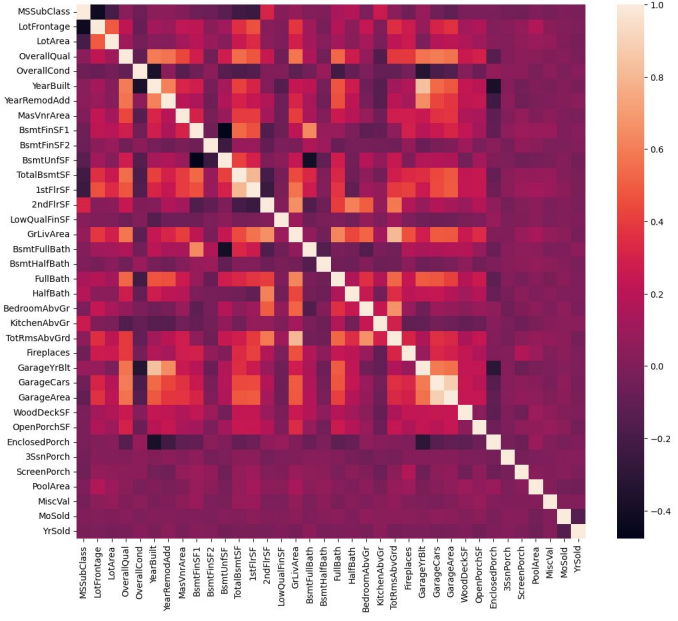
```
df.corr()
```

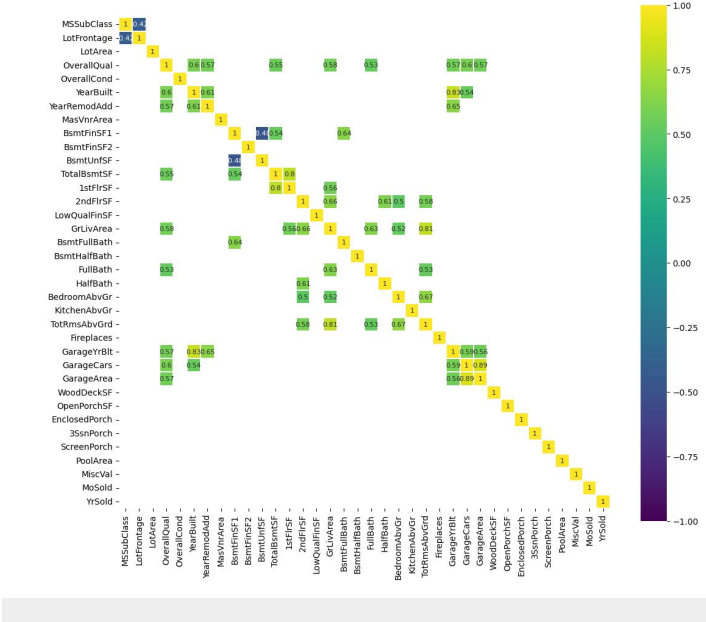
	MSSubClass	LotFrontage	LotArea	OverallQual	OverallCond	YearBuilt	YearRemodAdd	MasVnrArea	BsmtFinSF1	BsmtFinSF2	...	GarageArea	WoodDeckSF	OpenPorchSF	EnclosedPorch	3SsnPorch	ScreenPorch	PoolArea	MiscVal	MoSold	YrSold
MSSubClass	1.000000	-0.417359	-0.201730	0.033638	-0.065625	0.034409	0.043315	0.005433	-0.064311	-0.072530	...	-0.103394	-0.017654	-0.015923	-0.020867	-0.037529	-0.049181	-0.003080	-0.028867	-0.001231	-0.015028
LotFrontage	-0.417359	1.000000	0.489896	0.217645	-0.075508	0.122811	0.091557	0.221079	0.219408	0.047431	...	0.359786	0.122070	0.164896	0.011509	0.028289	0.075858	0.174119	0.044272	0.011254	-0.007917
LotArea	-0.201730	0.489896	1.000000	0.100541	-0.035617	0.024128	0.021612	0.125596	0.194031	0.084059	...	0.213251	0.158045	0.104797	0.020974	0.015995	0.054375	0.093708	0.069029	0.004156	-0.024234
OverallQual	0.033638	0.217645	0.100541	1.000000	-0.093847	0.597554	0.571532	0.432947	0.281810	-0.042771	...	0.565122	0.255317	0.298084	-0.139256	0.018715	0.042910	0.030740	0.005562	0.030405	-0.019614
OverallCond	-0.065625	-0.075508	-0.035617	-0.093847	1.000000	-0.368477	0.047654	-0.136007	-0.050418	0.041501	...	-0.154149	0.020123	-0.068978	0.071044	0.043739	0.043713	-0.016876	0.033956	-0.006256	0.030102
YearBuilt	0.034409	0.122811	0.024128	0.597554	-0.368477	1.000000	0.612235	0.314051	0.279581	-0.027595	...	0.480735	0.229426	0.198554	-0.374073	0.015958	-0.041046	0.002304	-0.010886	0.013938	-0.012344
YearRemodAdd	0.043315	0.091557	0.021612	0.571532	0.047654	0.612235	1.000000	0.196875	0.152126	-0.062153	...	0.376765	0.218513	0.242182	-0.220456	0.037433	-0.046878	-0.011407	-0.003124	0.017693	0.033203
MasVnrArea	0.005433	0.221079	0.125596	0.432947	-0.136007	0.314051	0.196875	1.000000	0.303490	-0.015645	...	0.374061	0.166200	0.144650	-0.111499	0.013612	0.065209	0.004512	0.044811	-0.000117	-0.018510
BsmtFinSF1	-0.064311	0.219408	0.194031	0.281810	-0.050418	0.279581	0.152126	0.303490	1.000000	-0.055045	...	0.310449	0.223492	0.124163	-0.099712	0.050908	0.096823	0.084462	0.093295	-0.000942	0.022556
BsmtFinSF2	-0.072530	0.047431	0.084059	-0.042771	0.041501	-0.027595	-0.062153	-0.015645	-0.055045	1.000000	...	0.003139	0.098399	-0.005876	0.032740	-0.023279	0.063302	0.044524	-0.005139	-0.009593	0.008883
BsmtUnfSF	-0.125994	0.113714	0.021362	0.275175	-0.138202	0.130473	0.165175	0.090163	-0.477404	-0.238241	...	0.164409	-0.039302	0.119764	0.005006	-0.005810	-0.049158	-0.032273	-0.010492	0.022607	-0.038015
TotalBsmtSF	-0.219965	0.354822	0.254138	0.549294	-0.174002	0.408515	0.298107	0.397240	0.536467	0.089410	...	0.486067	0.229600	0.245521	-0.085510	0.037892	0.075363	0.072216	0.084002	0.017888	-0.011184
1stFlrSF	-0.248641	0.458247	0.332460	0.479152	-0.157418	0.310814	0.242245	0.395834	0.458092	0.084330	...	0.492011	0.227347	0.238502	-0.065796	0.044086	0.098381	0.121900	0.093062	0.040143	-0.013442
2ndFlrSF	0.309309	0.026545	0.031515	0.245596	0.005494	0.017588	0.158985	0.121014	-0.162301	-0.097744	...	0.128570	0.089922	0.185387	0.054645	-0.032458	0.011070	0.044503	-0.005299	0.014185	-0.019229
LowQualFinSF	0.026482	0.004894	0.000554	-0.048393	0.009048	-0.144191	-0.060371	-0.057912	-0.066028	-0.004923	...	-0.053551	-0.015622	-0.000692	0.087212	-0.004545	0.006835	0.035177	-0.005973	0.011528	-0.002257
GrLivArea	0.071677	0.382462	0.284519	0.575126	-0.116569	0.242666	0.316972	0.402994	0.211669	-0.017872	...	0.485469	0.251017	0.341907	0.003274	0.006268	0.086284	0.135441	0.067118	0.044198	-0.026919
BsmtFullBath	0.009950	0.113245	0.128349	0.164543	-0.042133	0.211580	0.134947	0.141593	0.638847	0.162835	...	0.184738	0.186107	0.081265	-0.068393	0.027378	0.053119	0.043970	-0.004629	-0.003568	0.045255
BsmtHalfBath	-0.001878	-0.025629	0.026292	-0.040732	0.084181	-0.030282	-0.046285	0.015006	0.078361	0.099485	...	-0.021445	0.051549	-0.034917	-0.009675	0.026861	0.042069	0.066851	0.036902	0.023014	-0.020028
FullBath	0.139140	0.181668	0.125826	0.528483	-0.215504	0.471169	0.457980	0.259777	0.081525	-0.075432	...	0.408487	0.181266	0.260423	-0.118983	0.015192	-0.015910	0.028091	-0.010024	0.046274	-0.004882
HalfBath	0.178750	0.039452	0.034244	0.272668	-0.088577	0.269743	0.211430	0.191950	-0.007311	-0.032448	...	0.178993	0.116701	0.182048	-0.081978	-0.023417	0.035622	0.001424	0.026553	-0.001136	0.001587
BedroomAbvGr	-0.008796	0.234892	0.132801	0.073075	-0.008477	-0.053101	-0.021912	0.078126	-0.113547	-0.031223	...	0.073912	0.031644	0.086212	0.049940	-0.048279	0.007254	0.036544	0.000245	0.055997	-0.020453
KitchenAbvGr	0.260155	0.004676	-0.020854	-0.159325	-0.086700	-0.137614	-0.142431	-0.051389	-0.086354	-0.037779	...	-0.057852	-0.087399	-0.068181	0.027639	-0.021462	-0.056573	-0.013116	0.025078	0.035493	0.035173
TotRmsAbvGrd	0.040509	0.349513	0.213802	0.389761	-0.092027	0.114280	0.198250	0.278228	0.052141	-0.048423	...	0.328687	0.156543	0.238156	0.015345	-0.025764	0.032324	0.072061	0.060903	0.045137	-0.032354
Fireplaces	-0.055151	0.261970	0.261185	0.390753	-0.030999	0.170680	0.134157	0.275195	0.293089	0.065625	...	0.294767	0.227608	0.158621	0.000971	0.018852	0.169784	0.099123	0.008535	0.032352	-0.006752
GarageYrBlt	0.087898	0.076673	-0.008628	0.571803	-0.325849	0.834812	0.652365	0.255112	0.194270	-0.068549	...	0.555836	0.222584	0.231639	-0.300751	0.020699	-0.062320	-0.014467	-0.009203	0.024017	-0.004543
GarageCars	-0.046597	0.310587	0.180434	0.600744	-0.181787	0.538074	0.426022	0.361190	0.255482	-0.014827	...	0.889700	0.240715	0.203544	-0.132846	0.023383	0.043112	0.030424	-0.016934	0.050882	-0.022918
GarageArea	-0.103394	0.359786	0.213251	0.565122	-0.154149	0.480735	0.376765	0.374061	0.310449	0.003139	...	1.000000	0.238075	0.232559	-0.106417	0.029440	0.062389	0.053049	0.008446	0.040335	-0.013451
WoodDeckSF	-0.017654	0.122070	0.158045	0.255317	0.020123	0.229426	0.218513	0.166200	0.223492	0.098399	...	0.238075	1.000000	0.038457	-0.119114	-0.003935	-0.052134	0.094231	0.056883	0.018120	-0.000180
OpenPorchSF	-0.015923	0.164896	0.104797	0.298084	-0.068978	0.198554	0.242182	0.144650	0.124163	-0.005876	...	0.232559	0.038457	1.000000	-0.059672	-0.009392	0.047777	0.064212	0.077357	0.034167	-0.037377
EnclosedPorch	-0.020867	0.011509	0.020974	-0.139256	0.071044	-0.374073	-0.220456	-0.111499	-0.099712	0.032740	...	-0.106417	-0.119114	-0.059672	1.000000	-0.032822	-0.064377	0.092534	0.008654	-0.020976	-0.001134
3SsnPorch	-0.037529	0.028289	0.015995	0.018715	0.043739	0.015958	0.037433	0.013612	0.050908	-0.023279	...	0.029440	-0.003935	-0.009392	-0.032822	1.000000	-0.029546	-0.006525	-0.000788	0.027391	0.022560
ScreenPorch	-0.049181	0.075858	0.054375	0.042910	0.043713	-0.041046	-0.046878	0.065209	0.096823	0.063302	...	0.062389	-0.052134	0.047777	-0.064377	-0.029546	1.000000	0.026319	0.007067	0.028535	-0.006634
PoolArea	-0.003080	0.174119	0.093708	0.030740	-0.016876	0.002304	-0.011407	0.004512	0.084462	0.044524	...	0.053049	0.094231	0.064212	0.092534	-0.006525	0.026319	1.000000	0.011921	-0.042230	-0.052816
MiscVal	-0.028867	0.044272	0.069029	0.005562	0.033956	-0.010886	-0.003124	0.044811	0.093295	-0.005139	...	0.008446	0.056883	0.077357	0.008654	-0.000788	0.007067	0.011921	1.000000	0.007443	0.008445
MoSold	-0.001231	0.011254	0.004156	0.030405	-0.006256	0.013938	0.017693	-0.000117	-0.000942	-0.009593	...	0.040335	0.018120	0.034167	-0.020976	0.027391	0.028535	-0.042230	0.007443	1.000000	-0.153895
YrSold	-0.015028	-0.007917	-0.024234	-0.019614	0.030102	-0.012344	0.033203	-0.018510	0.022556	0.008883	...	-0.013451	-0.000180	-0.037377	-0.001134	0.022560	-0.006634	-0.052816	0.008445	-0.153895	1.000000

Correlation between features

So we use a seaboard heatmap

```
corr = df_num.drop([ 'SalePrice', 'Id'], axis=1).corr()
sns.heatmap(corr)
```





We can even filter this down to only display relationships above or below a certain threshold

```
corr[(corr >= 0.5) | (corr <= -0.4)]
```

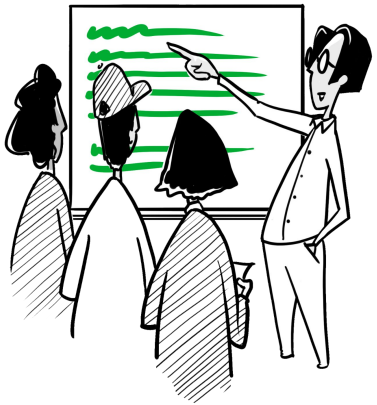
MadeTech

Already we are building up a picture

Using mostly inbuilt functions and next to no maths.

Hopefully it is clear that anyone can very quickly build a picture of the data and start a discussion.

- Where is data missing?
- How is that data distributed?
- How do features relate to one another?



Let’s take a break

Back in 10 mins

Multi-variate

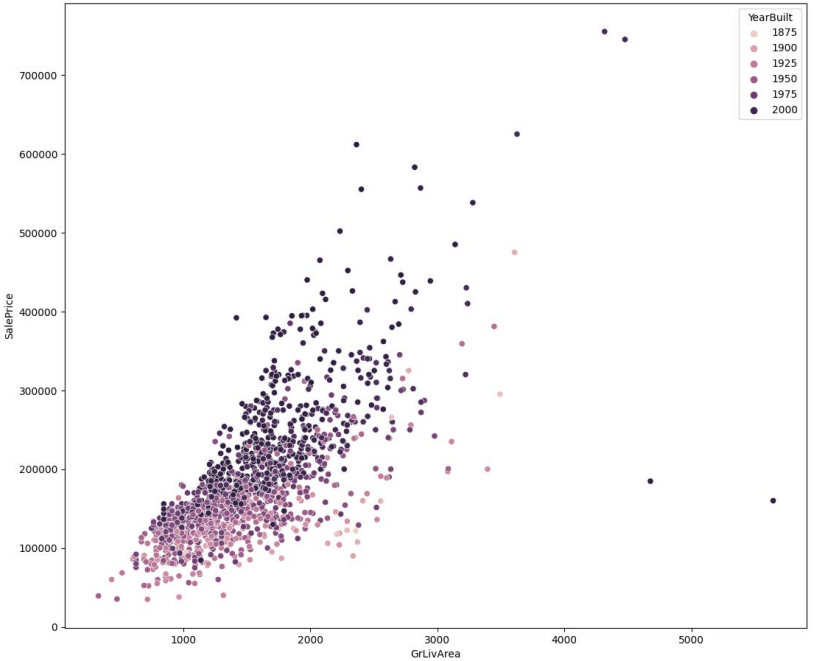
Now things get really fun!

What happens if you add categorical to scatter data

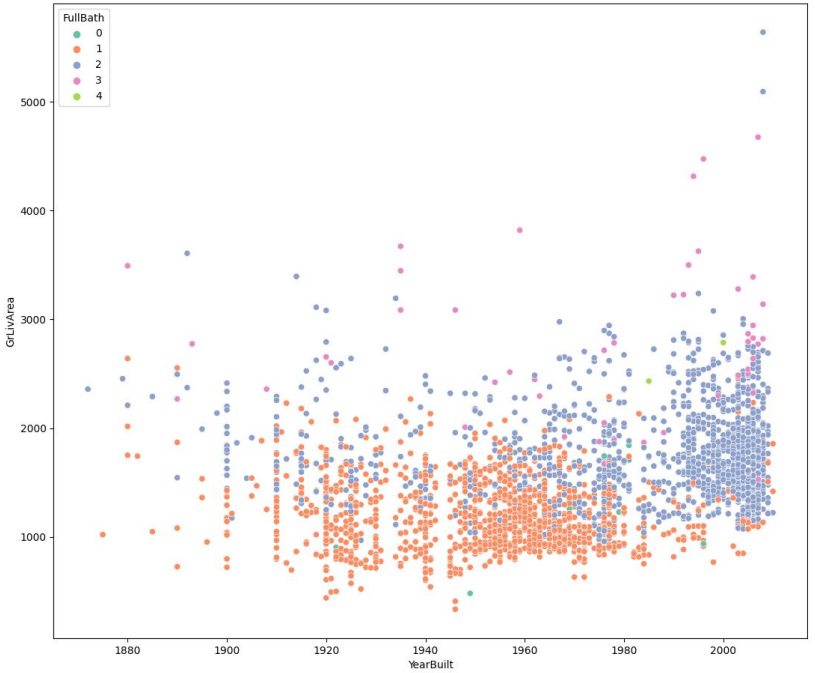
- Where is data missing?
- How is that data distributed?
- How do features relate to one another?

Multivariate – Scatter plots with colours

By simply adding a 3rd “variable”, we can see hidden structures in the data



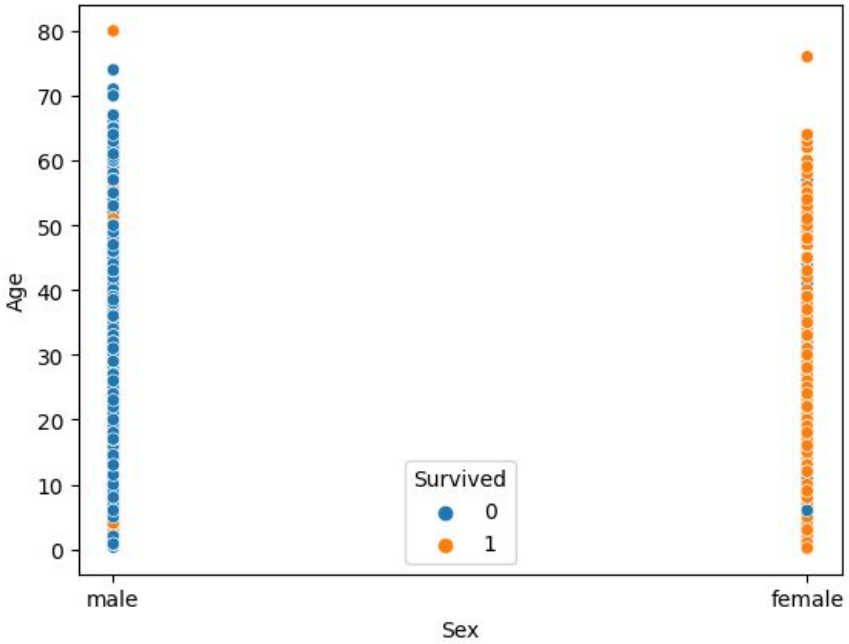
Newer properties seem to be more expensive, and also slightly larger.



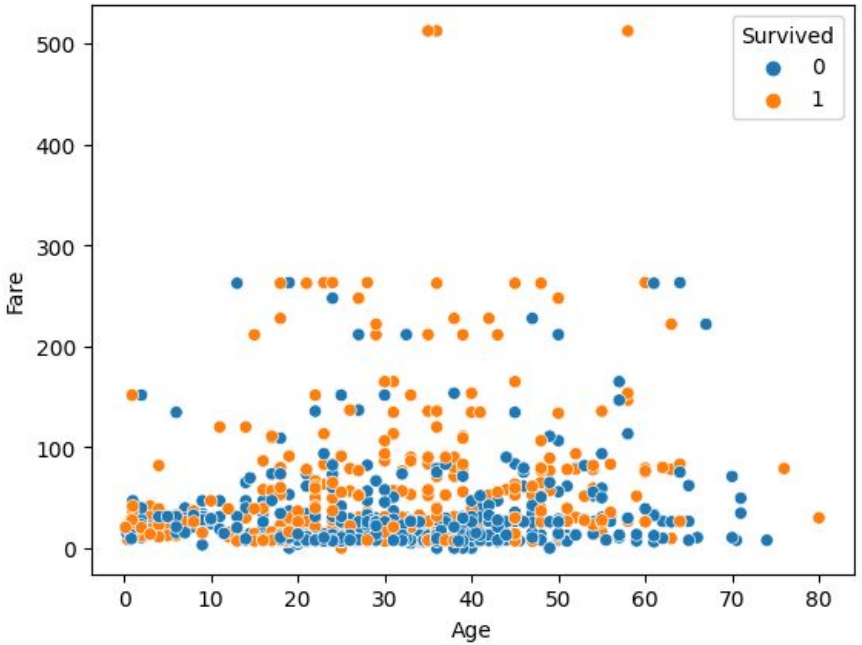
Newer properties typically have more bathrooms

Multivariate – Scatter plots with colours

By simply adding a 3rd “variable”, we can see hidden structures in the data



Depending on your sex.



Or how much you paid for your ticket…

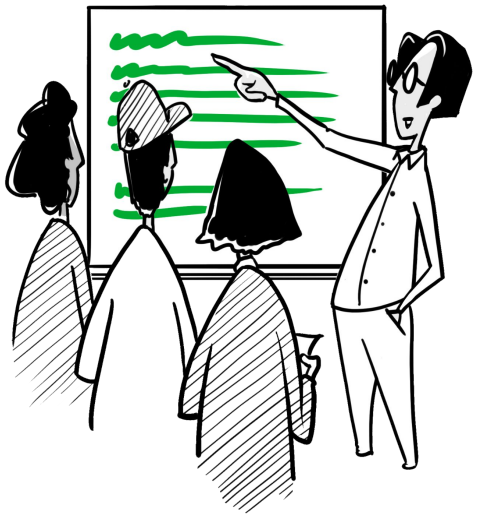
Summary

The point is...

Looking at the data is not very hard. But can add a huge amount of value.
As the first people in. We have to opportunity to really provide meaningful insight for our clients.

Keep it simple. That's often all you need.
The clients are not likely going to be statisticians, they don't want the maths. They want to look at their data and draw meaning from impactful visualisations.

By understanding the data, we can drive the direction of work.
If we are the ones to identify the problem. We are also the first people they will talk to about fixing the problem.
(Even if you made the problem...)



Let’s have lunch

Back at 13:30

Colours Kill

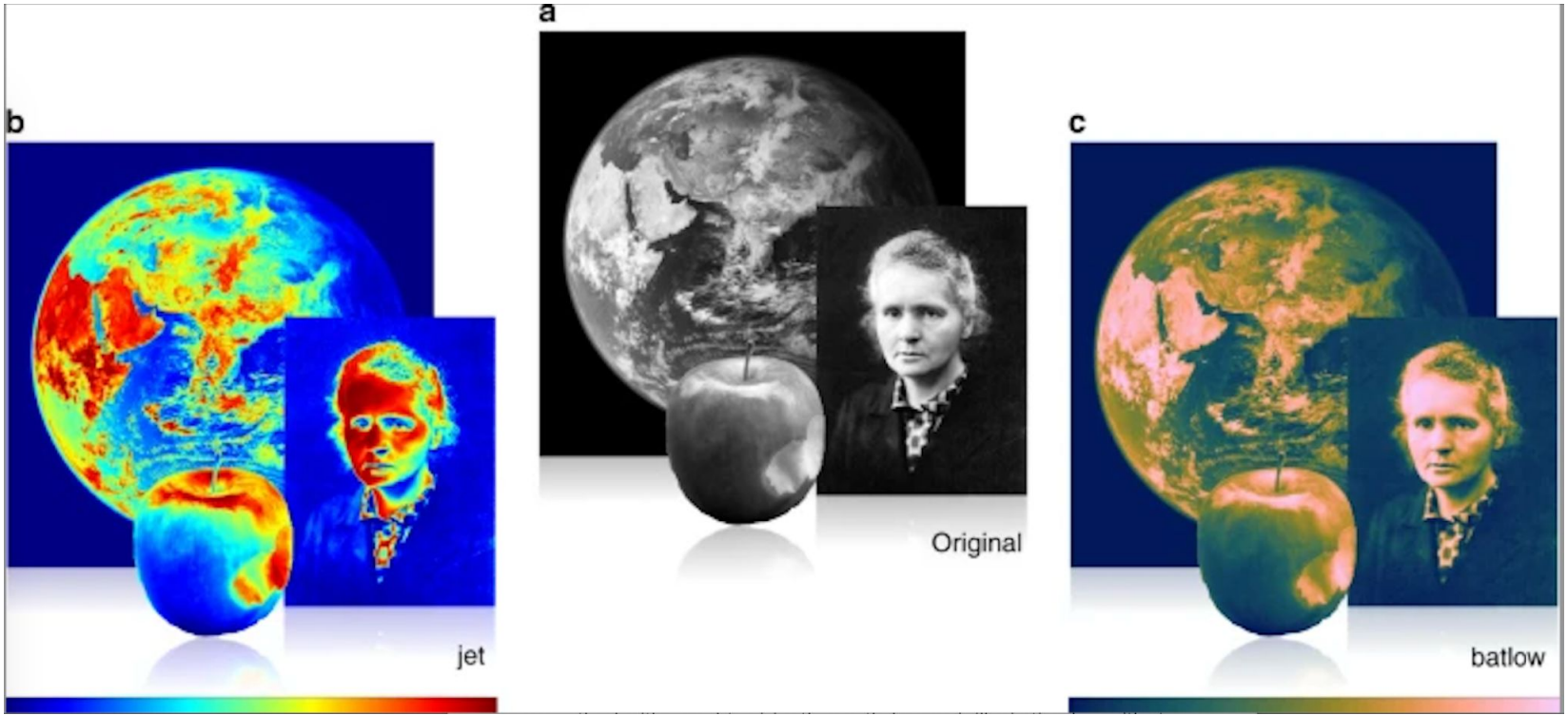
A time when the wrong visualisation had a massive impact

The misuse of colour in science communication
Fabio Crameri et al - https://www.nature.com/articles/s41467-020-19160-7

SciPy conference 2015
https://www.youtube.com/watch?v=xAoljeRJ3lU&list=PLYx7XA2nY5Gcpabmu61kKcToLz0FapmHu&index=2

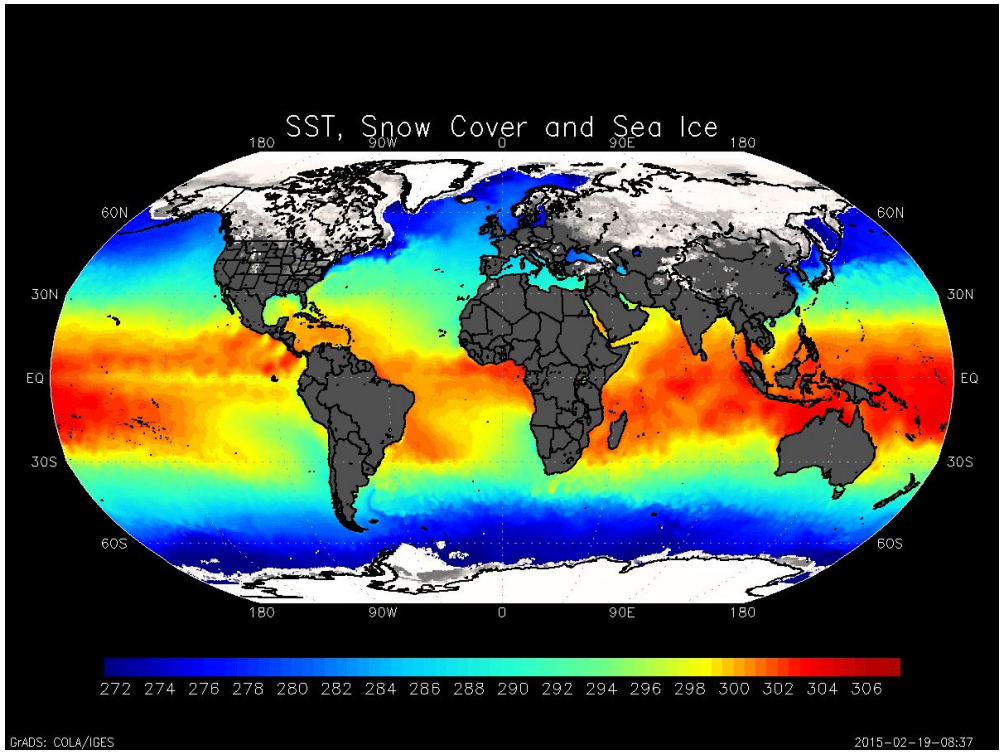
Jet

The designers were right all along!



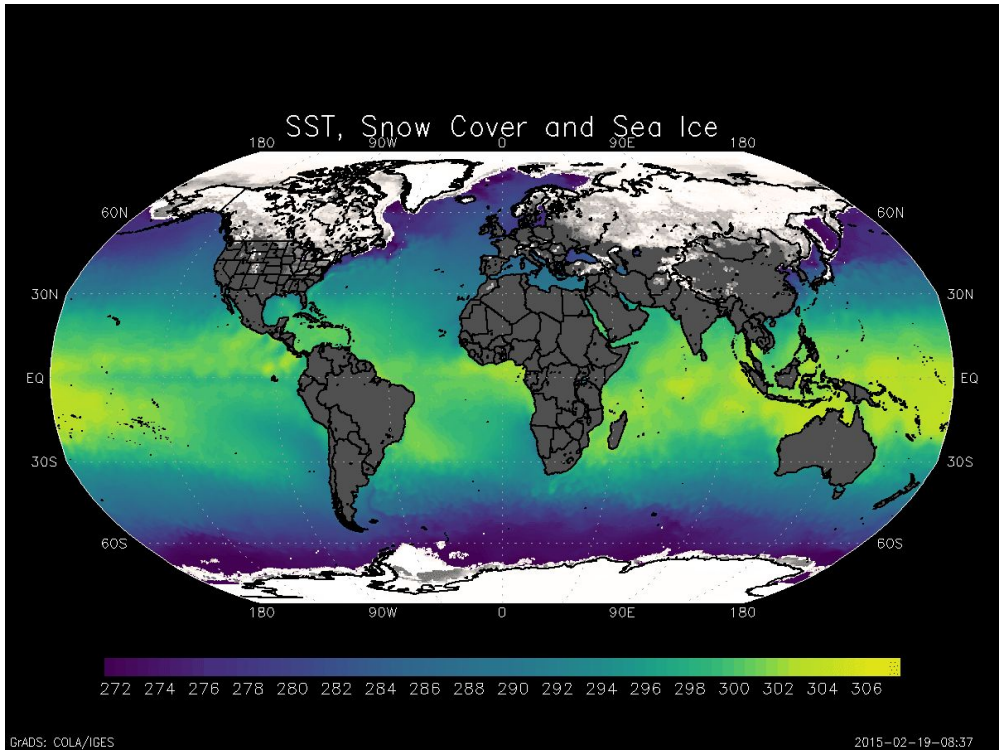
Jet

The designers were right all along!



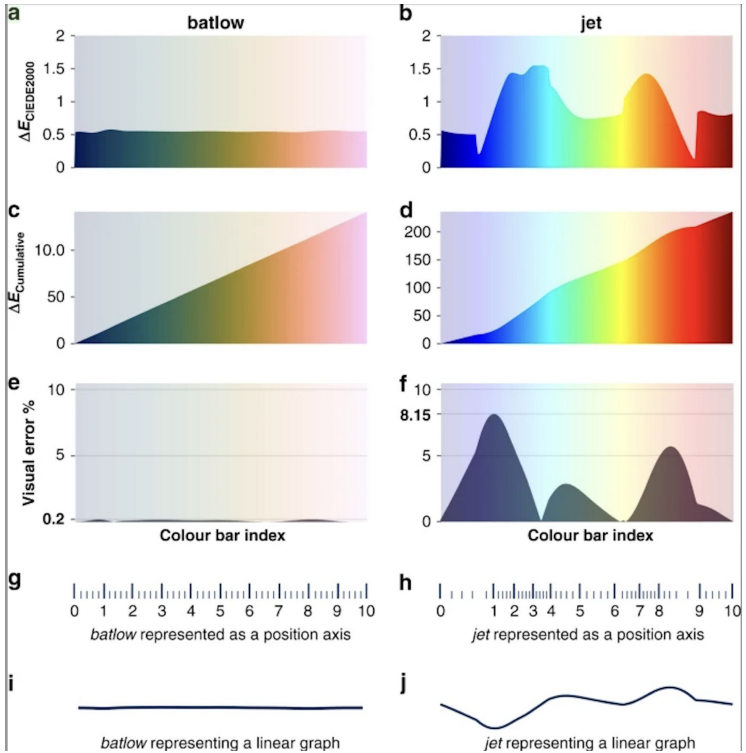
Jet

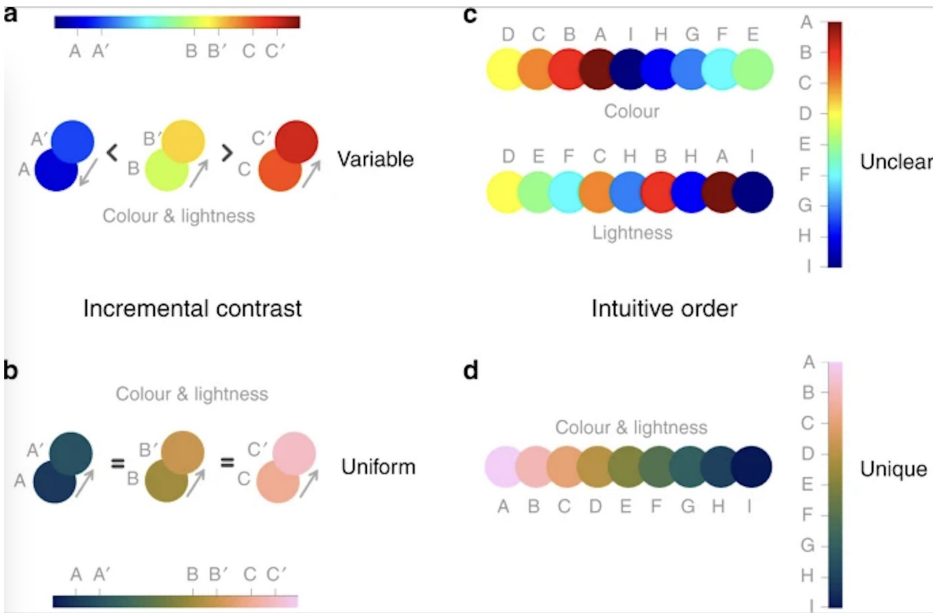
The designers were right all along!



Colours Kill

A time when the wrong visualisation had a massive impact





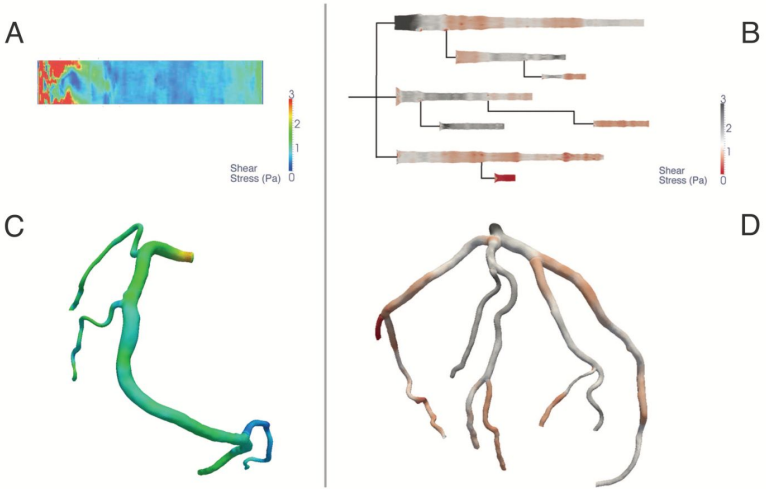
MadeTech

Colours Kill

This was a very real problem

Evaluation of Artery Visualizations for Heart Disease Diagnosis

Michelle A. Borkin, *Student Member, IEEE*, Krzysztof Z. Gajos, Amanda Peters, Dimitrios Mitsouras,
Simone Melchionna, Frank J. Rybicki, Charles L. Feldman, & Hanspeter Pfister, *Senior Member, IEEE*



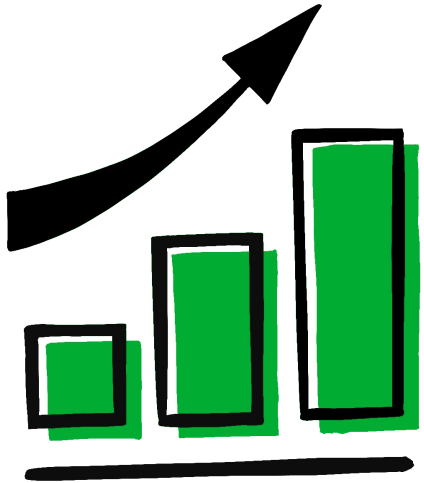
Evaluation of Artery Visualizations for Heart Disease Diagnosis
http://www.eecs.harvard.edu/~kgajos/papers/2011/borkin11-infoviz.pdf

Visualisations are KEY

If you cannot read your graph, then does it even contain data?

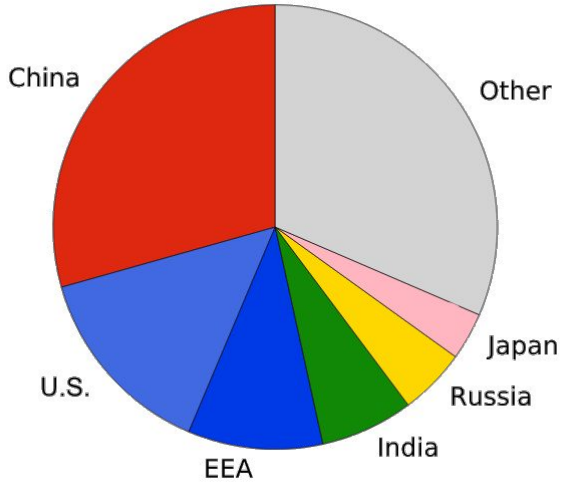
The three pillars of excellent visualisations:

1. **Clear axis** - Meaningful labels, appropriate scales, legible text and values
2. **Sendible colours and markers** - can it still be read in black and white? Is it accessible?
3. **Simplicity** - Do not go for style over substance.



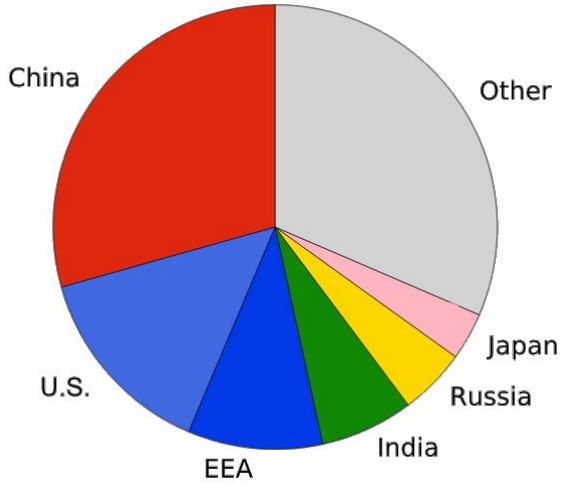
Let’s walk through some examples.

What is wrong with this (shudder) Pie chart?



Let’s walk through some examples.

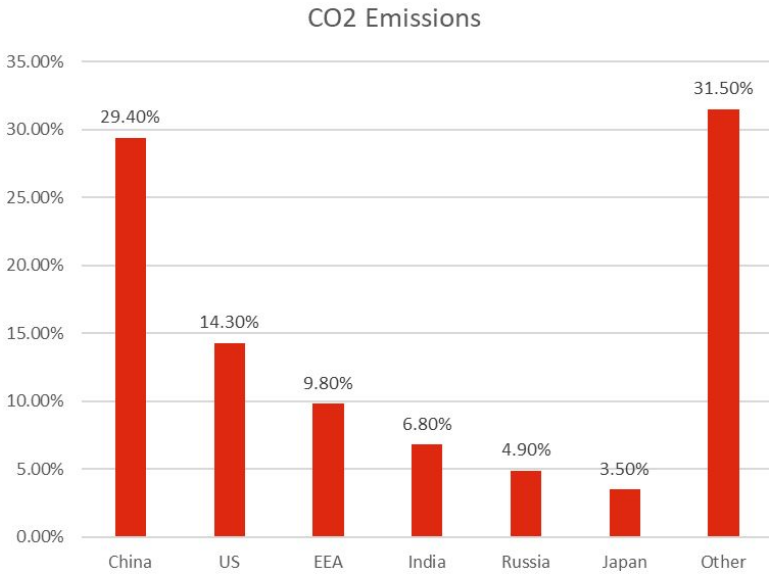
What is wrong with this (shudder) Pie chart?



- Best used for 3 to 4 items (but really not ever)
- Difficult to compare sizes of “China” and “Other”
- Low contrast between “U.S.” and “EEA”
- Hard to discern real values

Let’s walk through some examples.

Better to use a simple bar chart

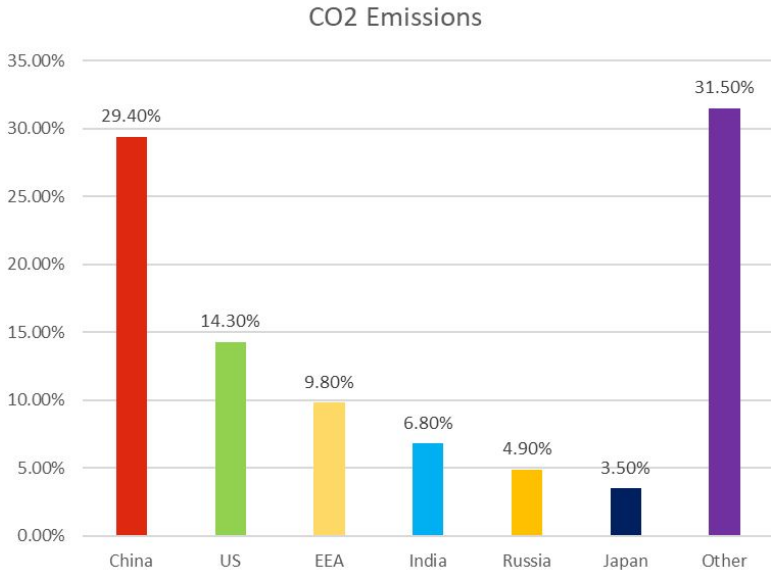


- Easy to read accurate values.
- Easy to compare
- Does not rely on colour (same info in black and white)

```
df["<feature name>"].value_counts(normalize=True).plot(kind="bar")
```

Let’s walk through some examples.

But don’t overload the user with colours!

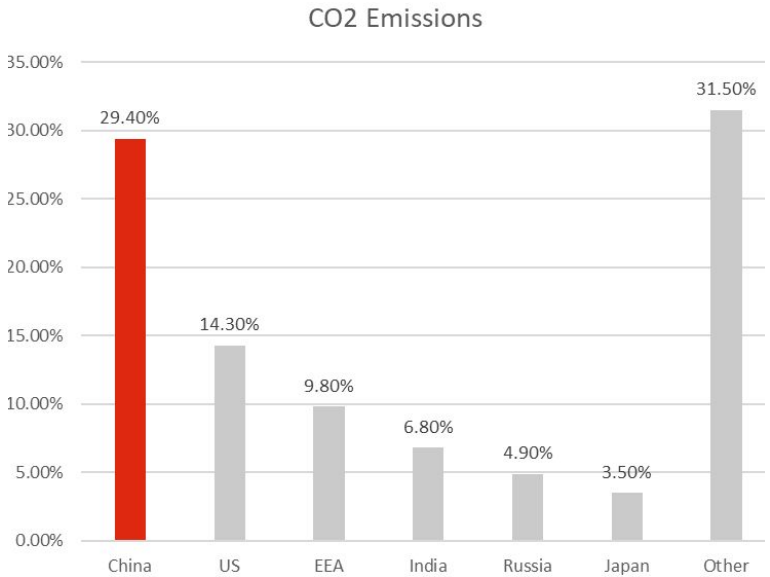


- More colours means more for the user to process
- Is there meaning to the colours?
- Conveys different information in black and white

```
df["<feature name>"].value_counts(normalize=True).plot(kind="bar")
```

Let’s walk through some examples.

If you really want to highlight something with colour, limit yourself



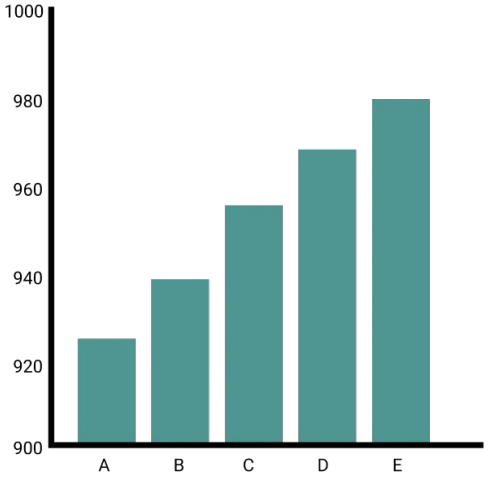
- Draw the users attention
- Clear meaning
- Back to conveying the same information in black and white

```
df["<feature name>"].value_counts(normalize=True).plot(kind="bar")
```

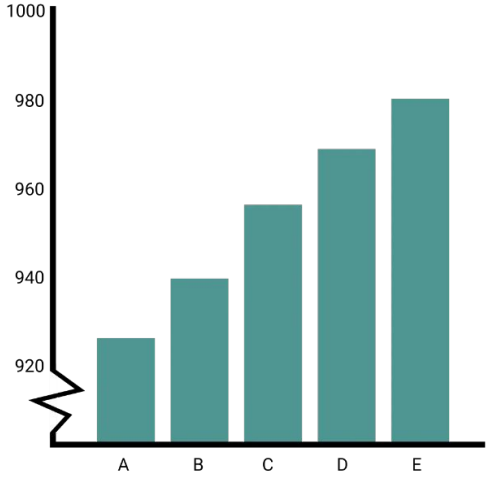
Let’s walk through some examples.

Setting your axis is very important - don’t present a misleading graph

Misleading Graph

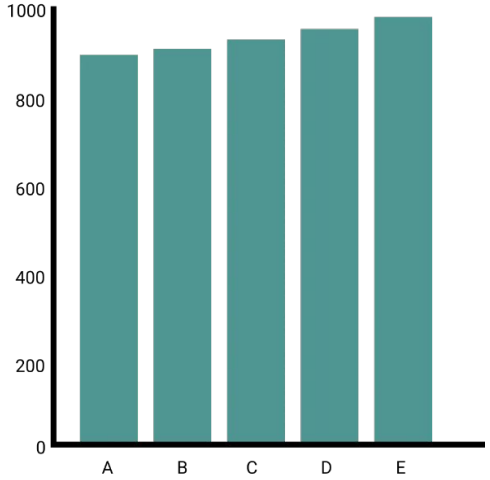


Adding Zero-Break



or

Starting from Zero



See how altering the scale massively alters the presentation of the data

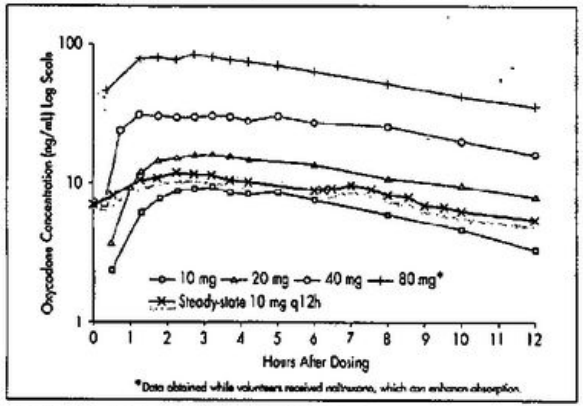
Let’s walk through some examples.

Don’t be like Purdue Phama

For moderate to severe pain when a continuous, around-the-clock analgesic is needed for an extended period of time

Consistent Plasma Levels Over 12 Hours

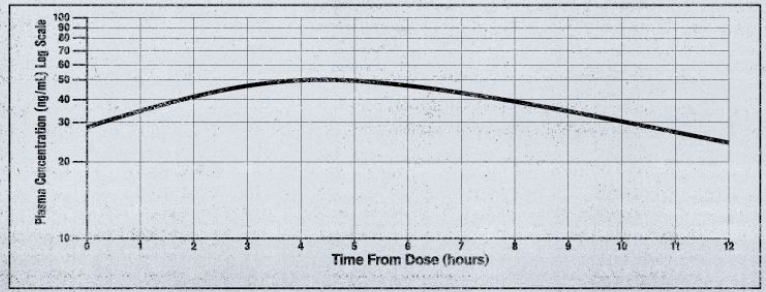
Plasma concentrations (ng/mL) over time of various dosage strengths



- OxyContin® 80 and 160 mg Tablets FOR USE ONLY IN OPIOID-TOLERANT PATIENTS requiring minimum daily oxycodone equivalent dosages of 160 mg and 320 mg, respectively. These tablet strengths may cause fatal respiratory depression when administered to patients not previously exposed to opioids

Steady state achieved within 24 to 36 hours

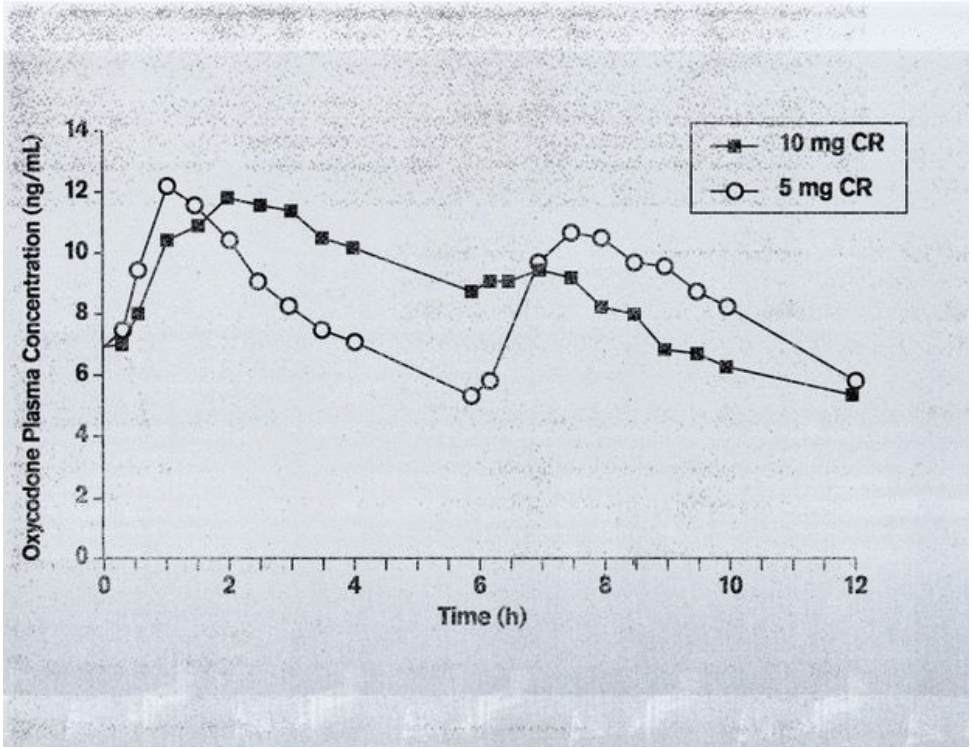
Q12h dosing
provides smooth and
sustained blood levels.



– Fewer “peaks and valleys” than with immediate-release oxycodone

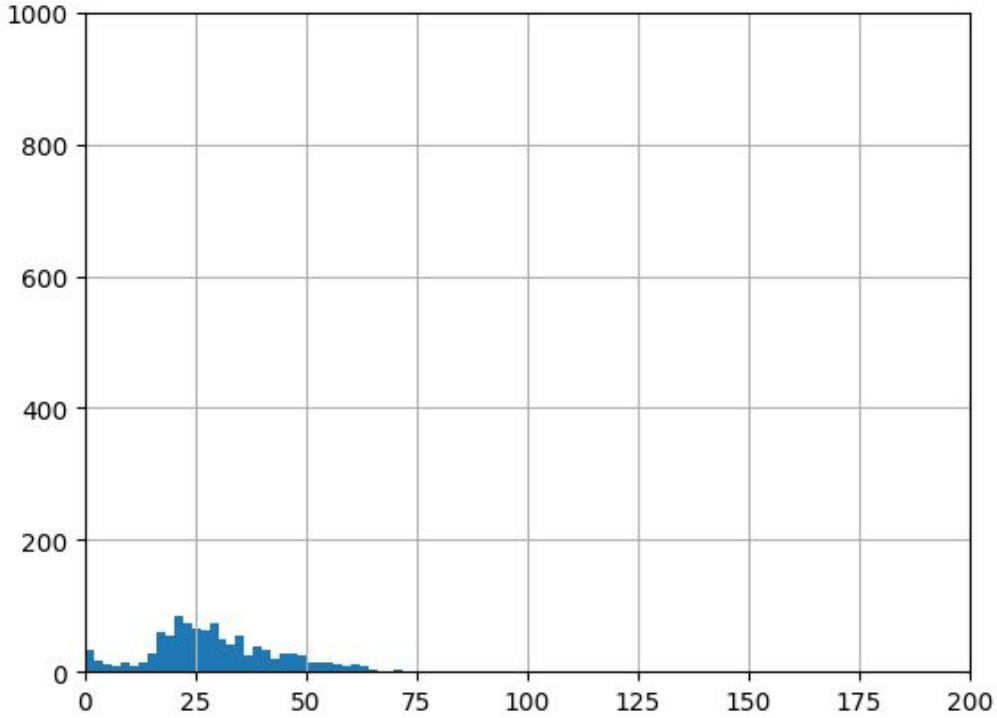
Let’s walk through some examples.

Dont be like Purdue Phama



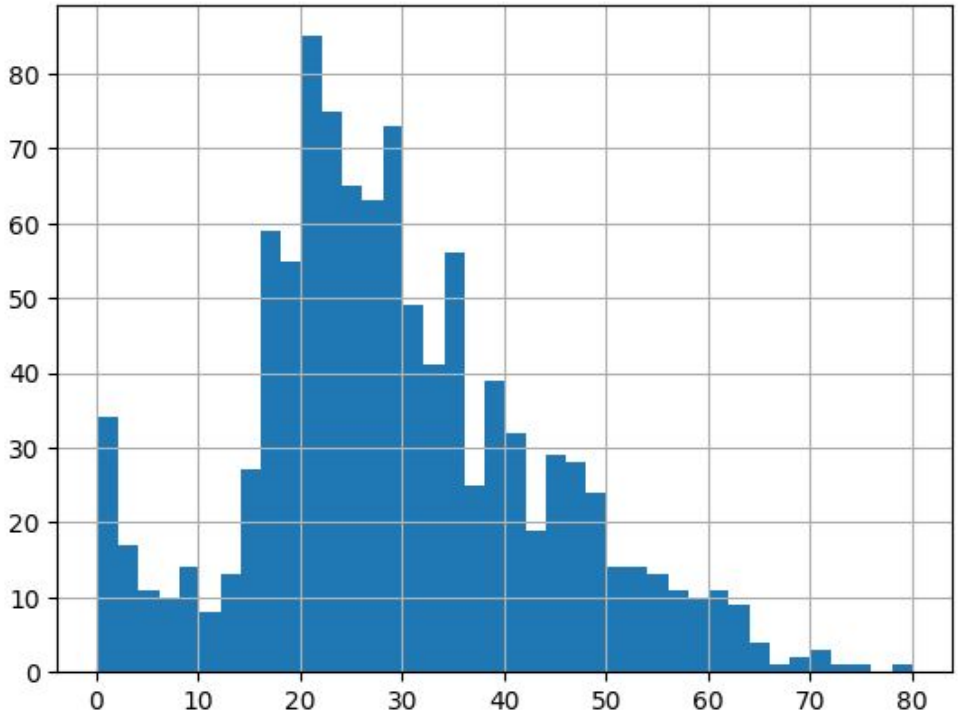
Let’s walk through some examples.

Dont be like Purdue Phama



Let’s walk through some examples.

Dont be like Purdue Phama



Thank you.