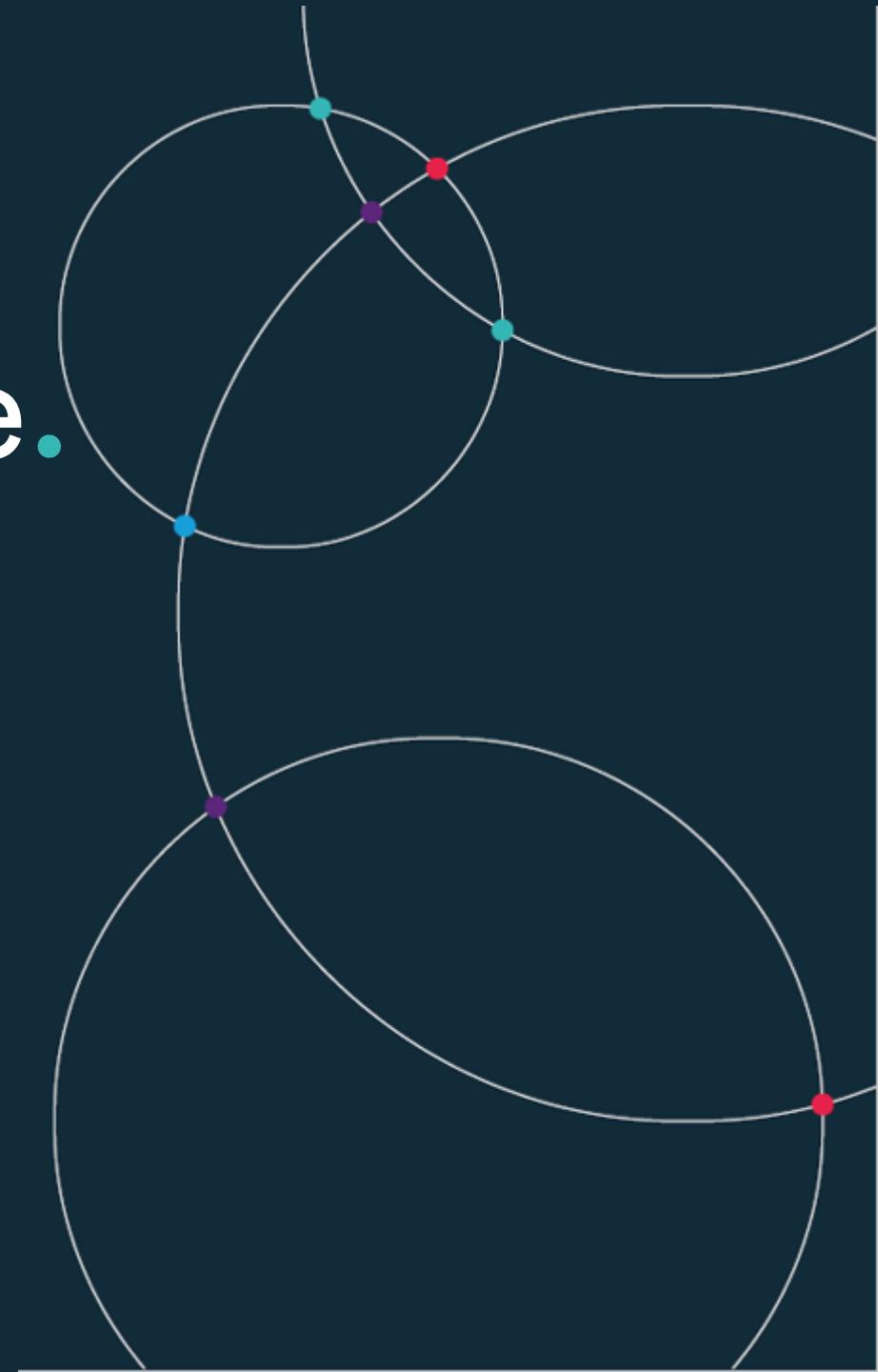


# Practical Data Science.

*For economists*

## Week 5.

University of Bristol  
Autumn 2022



# Nearly half way...

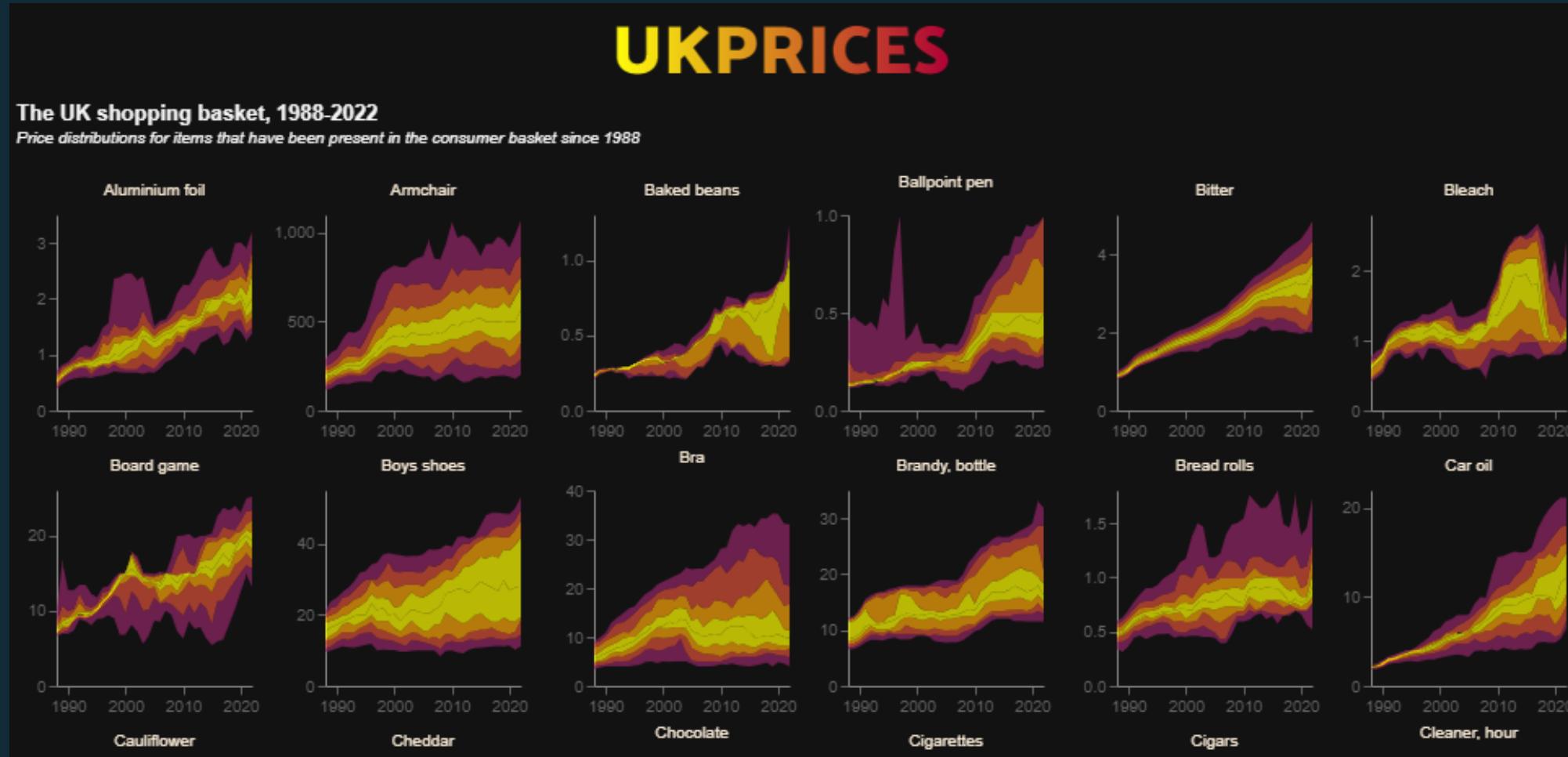
*Re-cap, and what's coming next*

- W1. Motivation, HTML, CSS. GitHub and embedding.
- W2. Chart design. Raw files, editing data. String and date functions.
- W3. Data: models and visualisation types. Manipulating data with Python.
- W4. Data: storage types. Tidy data storage. APIs. Scraping with Python.
- W5. Control structures. APIs and loops with Python.
- W6. Break.
- W7. Maps. Cartography history. Projections. Base maps. Putting data on maps.
- W8. Advanced analytics. Transforming data. Investigating relationships.
- W9. Big data. Challenges with scale. A big data cookbook.
- W10. Machine Learning. History. Regressions and classification. Clustering and dimensionality reduction.
- W11. Large Language Models. Gathering, cleaning, visualising data. Auditing output.

# Week 5.

*This week*

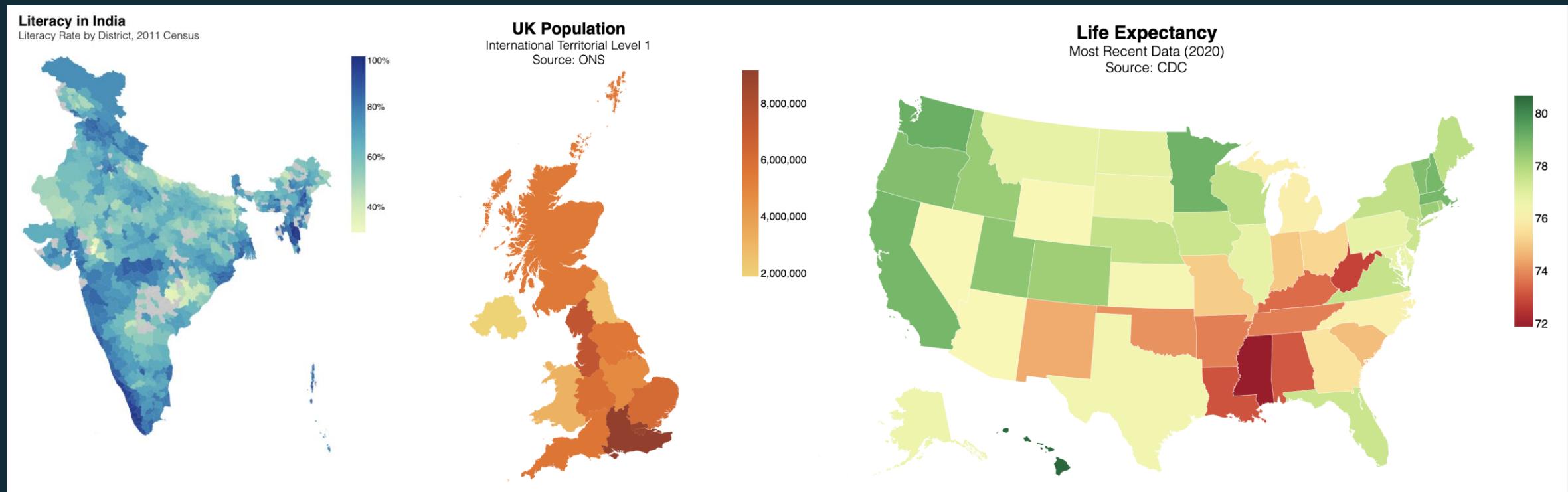
APIs and loops with Python. Making your code do the work for you.



# Week 6.

*Next week*

Cartography history. Projections. Base maps. Putting your data on maps.



# Tell us about your projects

*Remember, this is NOT for grading, it is so that we can help you.*



Google  
Sheet

Please set out the working title of your project on the sheet.

This is simply so that we can see, roughly, what you are working on.

You can change project and title as you wish.

# It's time to complete your unit evaluation surveys



**[bristol.ac.uk/unit-evaluation](http://bristol.ac.uk/unit-evaluation)**

# 5. Programming.

Making your code do the work for you



# Week 5.

- Programming. Control structures.
- Conditionals. Action/selection based on a criterion.
- Loops. Iteration.
- Dashboards. Using this to build a dashboard.

# 5.1 Control Structures.

Concepts, languages



# The idea.

Control structures | Flow control | Control statements

We want our programs/analysis to take decisions for us. Not to continue doing the same thing again and again, but to be able to decide what to do next.

Without control structures (AKA ‘flow control’) programs don’t do much.

What might you want a program to do for you?

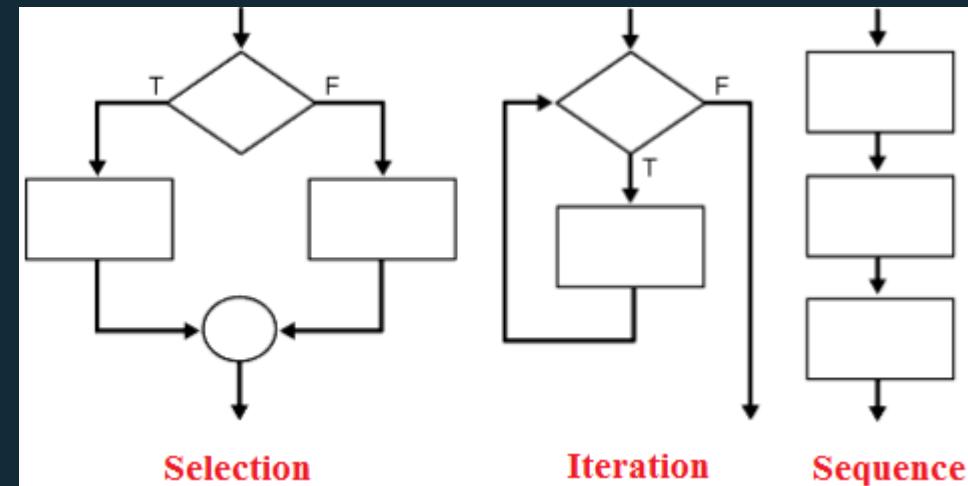
- Stop or start.
- Take a decision on what to do next. Do different things in different conditions:
  - Time of day, or days of the week;
  - If data has certain properties: (stock market alert).
- Do something many times.
  - Dynamic programming / maximisation;
  - Batches of analysis: downloading, cleaning, charting.

# The big three.

Sequence | Iteration | Conditionality/Selection

Programming languages typically feature three types of control:

- **Sequence.** Tells the program the order to do things in.
- **Selection/conditionality.** Makes a decision based on a testable condition.
- **Iteration.** Repeats a command over and over again, until a condition is met. Then stop and continue the code.



# Control in data science.

If-else | Loops

In practical terms during a career in data you are going to make daily use two particular examples of these general ideas.

- **If-else.** Test some condition in your data. Based on the results of this test, take a number of different actions.
- **Loops.** Do the same thing to many pieces of data, many variables, many data sets. Or do the same thing on a number of different days.

These can be combined all possible ways.

- An if statement inside an if (AKA “nested”)
- A loop withing a loop.
- If inside a loop
- Loop inside an if.

# How many languages?

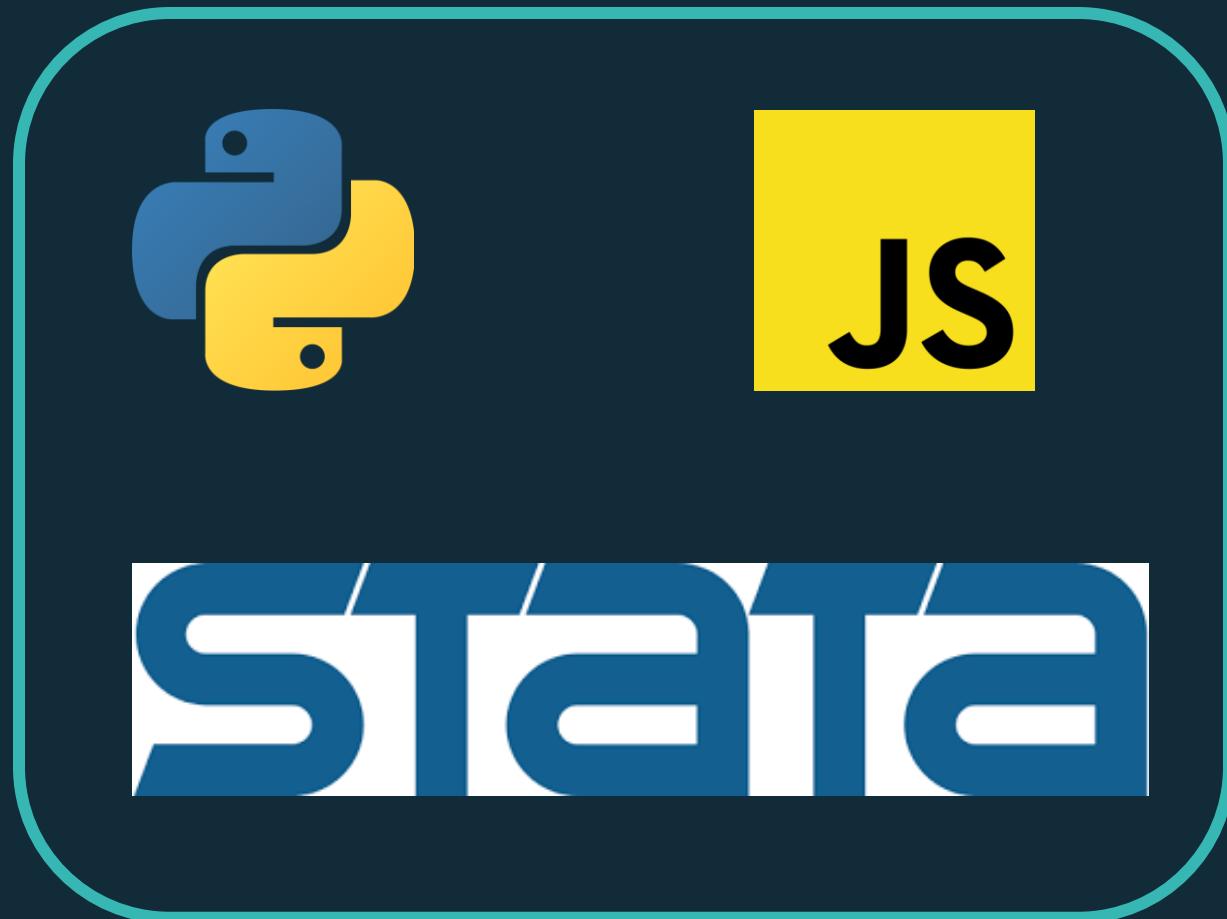
Five?



**Stata**

# How many languages?

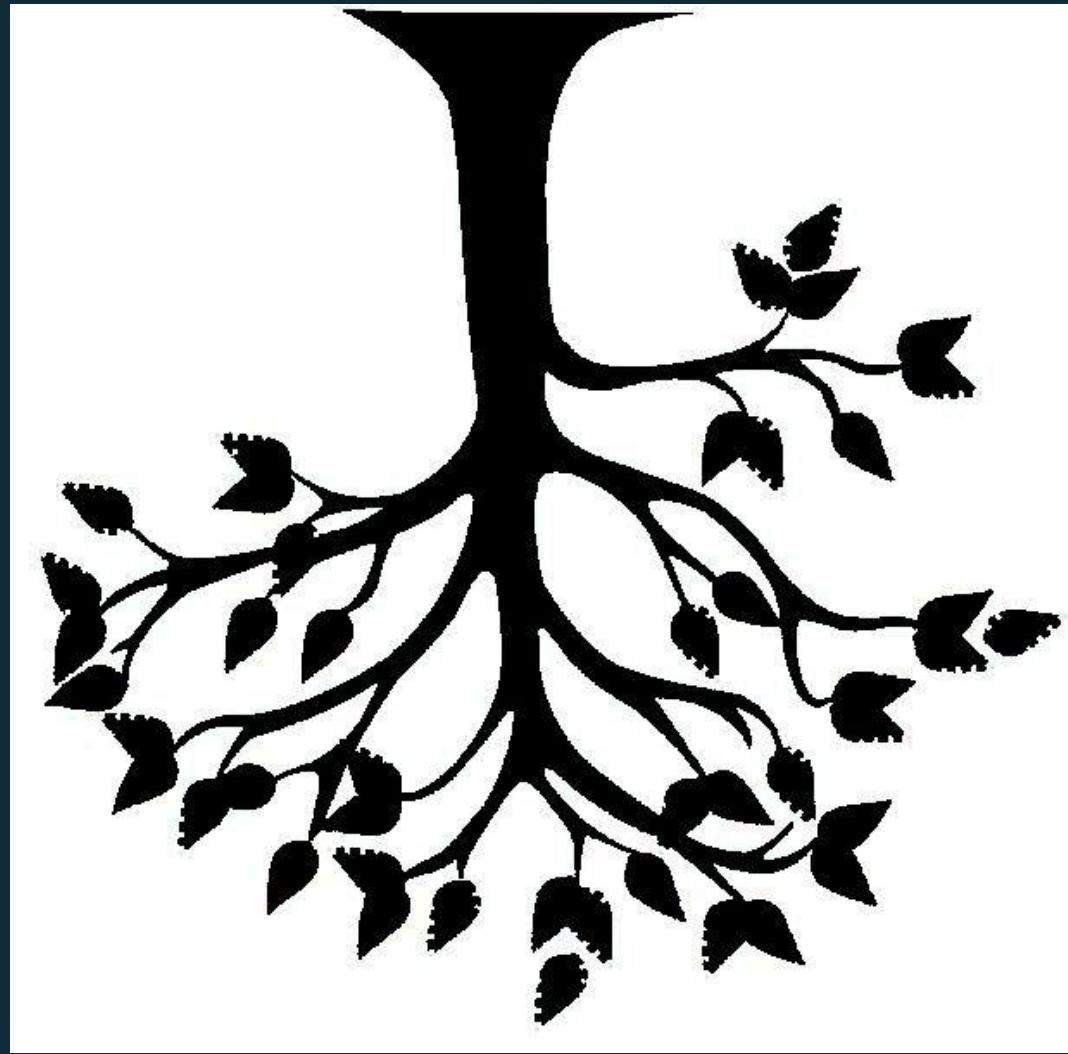
Three?



## 5.2 Conditionals.

A fork in the road





Code as a tree |  
Conditional statements  
influence which branch  
your code goes along.

# If-else.

Excel | STATA | Python | JavaScript | CSS

Syntax:

IF({CONDITION}, {VALUE IF TRUE}, {VALUE IF FALSE})

*Example:*

=if(A1="Richard", "Yes", "No")

# If-else.

Excel | STATA | Python | JavaScript | CSS

Practical usage:

## Data cleaning

*You have two data sets on countries that you want to match.*

*But the names are not consistent.*

*You want to create a column that provides a 1:1 correspondence between the data sets*

=if(A1="Côte d'Ivoire", "Ivory Coast", A1)

## Categorisation

*You have data on firm size, measured by number of employees...*

*You want to analyse based on two types, large firms and SMEs..*

*You define SME as anything with up to 250 employees..*

=if(A1>250, "Large", "SME")

# If-else – what does equal mean?

Excel | STATA | Python | JavaScript | CSS

=  
==  
=====

?  
.

# =, == and ===

Three uses of equals signs: [Stata](#) | [JS](#) | [Python](#)

= **Assigns value** | i.e. it makes the thing on the left equal the thing on the right.

== **Tests value** | It checks whether the thing on the left is equal to the thing on the right.

=== **Tests value and type** | Is the thing on the left the same value and the same type (number, word, etc) as the thing on the right.

# If-else – what does equal mean?

Excel | STATA | Python | JavaScript | CSS

Joe = President

Joe == President

Joe === President

# If-else.

Excel | STATA | Python | JavaScript | CSS

```
// Changing/cleaning part of an variable  
replace countryName ="Ivory Coast" if countryName=="Côte d'Ivoire"  
  
// Changing the way missing values are reported:  
replace value = . if value=="--"  
  
// Categorising firm size:  
gen firmSize = ""  
replace firmSize = "Small" if inrange(employees, 0, 24)  
replace firmSize = "Medium" if inrange(employees, 25, 249)  
replace firmSize = "Large" if employees >249
```

# If-else.

Excel | STATA | Python | JavaScript | CSS

```
# Simple if:  
if CONDITION:  
    #Code to run if true  
    #Can be more than one line  
  
# If else:  
if CONDITION:  
    # Code to run if true  
else  
    # Code to run if condition is FALSE  
  
# Elif  
if CONDITION_A:  
    # Code to run if conditionA TRUE.  
elif CONDITION_B:  
    # Code to run if conditionB TRUE.  
else:  
    # Code to run if BOTH conditions are FALSE.
```

# If-else.

Excel | STATA | Python | JavaScript | CSS

```
# Example: how big are firms in our dataset?

firm1 = 14
firm2 = 250

# Comparing values
if firm1 > firm2:
    print("Firm 1 has more employees than Firm 2")
else:
    print("Firm 2 has more employees than Firm 1")

# Assessing size
if firm1 > 249:
    print("Firm 1 is large")
else:
    print("Firm 1 is an SME")
```

# If-else.

Excel | STATA | Python | JavaScript | CSS

```
//Simple if condition
if (condition) {
    // Add code here--can be many lines, to run if the condition is TRUE.
    // If the condition is FALSE then nothing happens
}

//If else
if (condition) {
    // Code to run if condition TRUE.
} else {
    // Code to run if condition FALSE.
}

//If-elif-else:
if (conditionA) {
    // Code to run if conditionA TRUE.
} else if (conditionB) {
    // Code to run if conditionB TRUE.
} else {
    // Code to run if BOTH conditionA and conditionB are FALSE.
}
```

# If-else.

Excel | STATA | Python | JavaScript | **CSS**

CSS does not officially support conditionals.

But there is a way we can make CSS react in different ways in different situations?

**Media Queries (@media)** allow you to change the look based on screen size:

- With these you are saying:
- If the screen size is {CONDITION} then do this {ACTION}
- So CSS does allow conditionals.
- Understanding this is the way to make your site have two looks: one for mobile, one for laptop/desktop.

# Examples.

Why are conditionals so useful?



# Example 1. Cleaning. UK Gilt Market

Excel | STATA | Python | JavaScript | CSS

<https://dmo.gov.uk/data/gilt-market/gilts-in-issue/>

A	B	C	D	E	F	G	
1	Data Date: 20-Oct-2022	GILT MARKET	GILTS IN ISSUE ON 19 OCTOBER 2022				
2							
3							
4							
5							
6	Total Amount Outstanding (including inflation uplift for index-linked gilts) = £2,234.74 billion nominal						
7							
8							
9	Conventional Gilts	ISIN Code	Redemption Date	First Issue Date	Dividend Dates	Current/Next Ex-dividend Date	Total Amount in Issue (£ million nominal)
0	Ultra-Short						
1	0 1/8% Treasury Gilt 2023	GB00BL68HG94	31-Jan-2023	08-Apr-2020	31 Jan/Jul	20-Jan-2023	33,824
2	0 3/4% Treasury Gilt 2023	GB00BF0HZ991	22-Jul-2023	20-Jul-2017	22 Jan/Jul	12-Jan-2023	34,359
3	2 1/4% Treasury Gilt 2023	GB00B7Z53659	07-Sep-2023	12-Jun-2013	7 Mar/Sep	24-Feb-2023	35,922
4	0 1/8% Treasury Gilt 2024	GB00BMGR2791	31-Jan-2024	07-Oct-2020	31 Jan/Jul	20-Jan-2023	35,551
5	1% Treasury Gilt 2024	GB00BFWFPL34	22-Apr-2024	25-Jul-2018	22 Apr/Oct	13-Oct-2022	35,638
6	2 3/4% Treasury Gilt 2024	GB00BHBHFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806
7	0 1/4% Treasury Gilt 2025	GB00BLPK7110	31-Jan-2025	02-Jul-2021	31 Jan/Jul	20-Jan-2023	29,353
8	5% Treasury Stock 2025	GB0030880693	07-Mar-2025	27-Sep-2001	7 Mar/Sep	24-Feb-2023	37,339
9	0 5/8% Treasury Gilt 2025	GB00BK5CVX03	07-Jun-2025	03-Jul-2019	7 Jun/Dec	28-Nov-2022	44,623
0	2% Treasury Gilt 2025	GB00BTHH2R79	07-Sep-2025	20-Mar-2015	7 Mar/Sep	24-Feb-2023	39,934
1	Short						
2	0 1/8% Treasury Gilt 2026	GB00BL68HJ26	30-Jan-2026	03-Jun-2020	30 Jan/Jul	19-Jan-2023	35,316
3	1 1/2% Treasury Gilt 2026	GB00BYZW3G56	22-Jul-2026	18-Feb-2016	22 Jan/Jul	12-Jan-2023	43,651
4	0 3/8% Treasury Gilt 2026	GB00BNNGP668	22-Oct-2026	03-Mar-2021	22 Apr/Oct	13-Oct-2022	32,890
5	4 1/8% Treasury Gilt 2027	GB00BL6C7720	29-Jan-2027	13-Oct-2022	29 Jan/Jul	19-Jan-2023	4,445
6	1 1/4% Treasury Gilt 2027	GB00BDRHNP05	22-Jul-2027	15-Mar-2017	22 Jan/Jul	12-Jan-2023	40,987

# Example 1. UK Gilt Market

How would you clean this data?

0 1/8% Treasury Gilt 2023  
0¾% Treasury Gilt 2023  
2¼% Treasury Gilt 2023  
5% Treasury Stock 2025  
1 5/8% Treasury Gilt 2028  
6% Treasury Stock 2028  
0¼% Treasury Gilt 2031  
1% Treasury Gilt 2032  
4¼% Treasury Stock 2032  
0 7/8% Green Gilt 2033  
4½% Treasury Gilt 2034  
1 1/8% Treasury Gilt 2073  
0 1/8% Index-linked Treasury Gilt 2046  
0¾% Index-linked Treasury Gilt 2047  
0 1/8% Index-linked Treasury Gilt 2048  
0½% Index-linked Treasury Gilt 2050  
4 1/8% Index-linked Treasury Stock 2030  
2% Index-linked Treasury Stock 2035

Cleaning data is often a matter of:

1. Spotting patterns in the data. Where in the mess of string does the number that you want appear?
2. Using string functions to split, separate and change the data in other ways.

You will very often use programming logic – if statements – to do this.

# Example 1. UK Gilt Market

Excel | STATA | Python | JavaScript | CSS

<https://github.com/RDeconomist/RDeconomist.github.io/blob/main/data/GiltsInIssueRaw.csv>

44	Long	
45	4¾% Treasury Stock 2038	GB00B00NY175
46	1 1/8% Treasury Gilt 2039	GB00BLPK7334
47	4¼% Treasury Gilt 2039	GB00B3KJDS62
48	4¼% Treasury Gilt 2040	GB00B6460505
49	1¼ % Treasury Gilt 2041	GB00BJQWYH73
50	4½% Treasury Gilt 2042	GB00B1VWPJ53
51	3¼% Treasury Gilt 2044	GB00B84Z9V04
52	3½% Treasury Gilt 2045	GB00BN65R313
53	0 7/8% Treasury Gilt 2046	GB00BNNGP775
54	4¼% Treasury Gilt 2046	GB00B128DP45
55	1½% Treasury Gilt 2047	GB00BDCHBW80
56	1¾% Treasury Gilt 2049	GB00BFWFPP71
57	4¼% Treasury Gilt 2049	GB00B39R3707
58	0 5/8% Treasury Gilt 2050	GB00BMBL1F74
59	1¼% Treasury Gilt 2051	GB00BLH38158
60	3¾% Treasury Gilt 2052	GB00B6RNH572
61	1½% Green Gilt 2053	GB00BM8Z2V59

# Example 1. UK Gilt Market

Excel | STATA | Python | JavaScript | CSS

With a combination of string functions,  
we can clean the raw data with Stata:

- Upper, lower
- Split:
  - At a fixed position.
  - At a character.
- Concatenate
- Measuring:
  - Find the Length
  - Find the position of a character
- Substring
- Trim

```
[...]
///////////////////////////////
//2. USING STRING FUNCTIONS TO CLEAN THE DATA:

// Destrинг:
destring issueAmount baseRPI amountINCinfUplift, ignore(",") replace

// Problem variable: looks like this:
// "0 5/8% Treasury Gilt 2025"
// We want to get the type, and the coupon.

/// String split, based on a character:
split type, parse("T") generate(type2)
order type*

/// String concatenate:
** This adds back the missing "T".
gen T = "T"
replace type22 = T+type2
drop T

/// Generating a substring, based on the known position of a character:
replace type22 = substr(type22, 1, strpos(type22, "2") - 1)

/// Trimming the leading and trailing spaces from around a variable:
replace type22 = strtrim(type22)
[...]
```

# Example 1. UK Gilt Market

Excel | STATA | Python | JavaScript | CSS

And do the same thing with Python,  
using built-in Pandas methods:

- Upper, lower: str.upper, str.lower
- Split: str.split()
- Length: str.len()
- Substring: str[x:y]
- Trim: str.strip()

## Step 3: Splitting the type column

We split the type column to extract the Gilt Type and coupon information.

```
# Split the 'type' column and clean up
df[['rawType', 'extra']] = df['type'].str.split("T", n=1, expand=True)
df['giltType'] = df['rawType'].str.strip()
df.head()
```

	type	code	redemptionDate	issueDate	divDates	divNextDate	issueAmount	baseRPI
0	0 1/8% Treasury Gilt 2024	GB00BMGR2791	31-Jan-2024	07-Oct-2020	31 Jan/Jul	20-Jan-2023	35,551	NaN
1	1% Treasury Gilt 2024	GB00BFWFPL34	22-Apr-2024	25-Jul-2018	22 Apr/Oct	13-Oct-2022	35,638	NaN
2	2 3/4% Treasury Gilt 2024	GB00BHFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806	NaN
3	0 1/4% Treasury Gilt 2025	GB00BLPK7110	31-Jan-2025	02-Jul-2021	31 Jan/Jul	20-Jan-2023	29,353	NaN
4	5% Treasury Stock 2025	GB0030880693	07-Mar-2025	27-Sep-2001	7 Mar/Sep	24-Feb-2023	37,339	NaN

## Step 4: Processing the giltType

We fill missing values in giltType with "Green Gilt" and adjust for index-linked gilts.

```
# Add "T" back into the second part of the split if necessary
df['giltType'] = df['giltType'].replace("", "Green Gilt")
df['giltType'] = df.apply(lambda x: "Index Linked" if pd.notna(x['baseRPI']) else x['giltType'], axis=1)
df.head()
```

	type	code	redemptionDate	issueDate	divDates	divNextDate	issueAmount	baseRPI
0	0 1/8% Treasury Gilt 2024	GB00BMGR2791	31-Jan-2024	07-Oct-2020	31 Jan/Jul	20-Jan-2023	35,551	NaN
1	1% Treasury Gilt 2024	GB00BFWFPL34	22-Apr-2024	25-Jul-2018	22 Apr/Oct	13-Oct-2022	35,638	NaN
2	2 3/4% Treasury Gilt 2024	GB00BHFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806	NaN
3	0 1/4% Treasury Gilt 2025	GB00BLPK7110	31-Jan-2025	02-Jul-2021	31 Jan/Jul	20-Jan-2023	29,353	NaN
4	5% Treasury Stock 2025	GB0030880693	07-Mar-2025	27-Sep-2001	7 Mar/Sep	24-Feb-2023	37,339	NaN

# Example 2. In Vega-Lite.

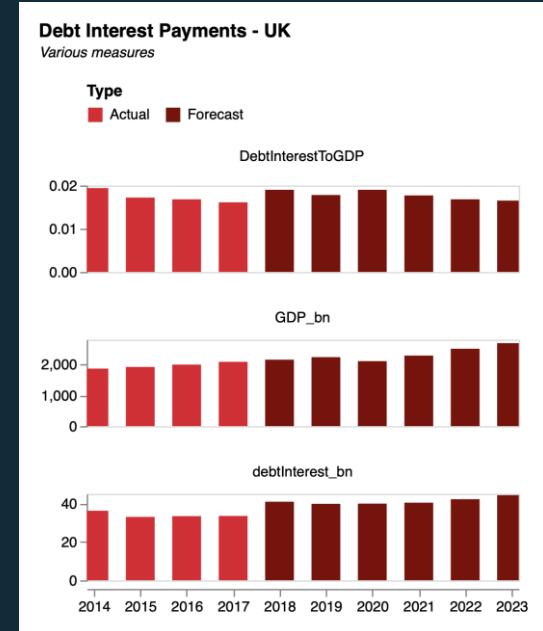
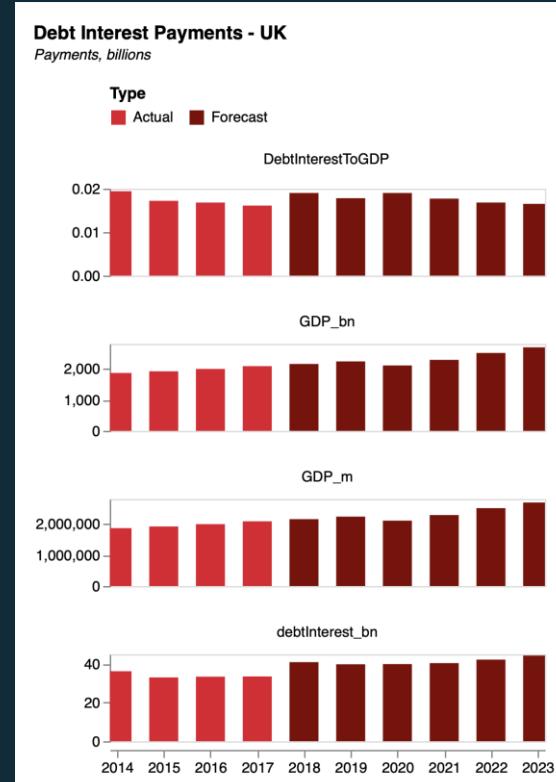
Using conditionals in your charts

With a Vega-lite filter transform, we can restrict the data to just the variables or values we care about.

Recall our debt example from earlier in the course.

For code see: [richarddavies.io/library](http://richarddavies.io/library)

```
"transform": [  
    {"filter":{  
        "field": "Variable",  
        "oneOf": ["DebtInterestToGDP", "GDP_bn", "debtInterest_bn"]  
    }}  
,
```

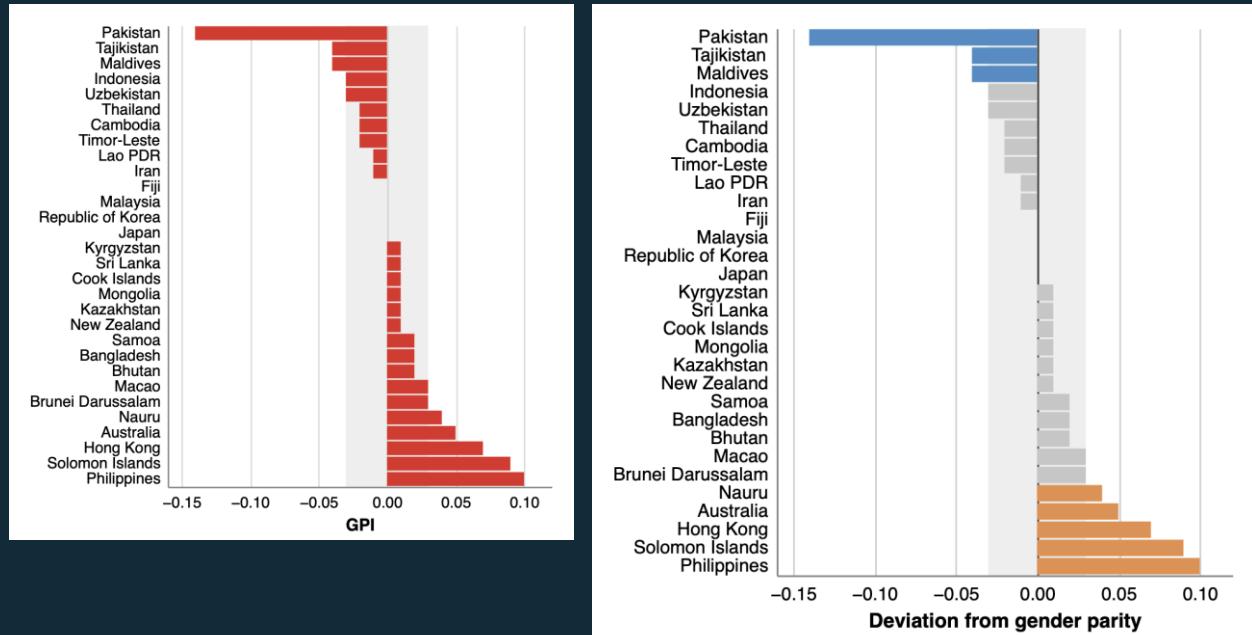


# Example 3. In Vega-Lite.

Using conditionals in your charts

With a `condition` objects, we can define rules for encodings. For example, we can highlight points or rows by value.

```
"color": {  
  "condition": [  
    {  
      "test": "datum.diff < -0.035",  
      "value": "rgb(69,141,198)"  
    },  
    {  
      "test": "datum.diff > -0.035 && datum.diff <= 0.035",  
      "value": "rgb(198,198,198)"  
    },  
    {  
      "test": "datum.diff > 0.035",  
      "value": "rgb(230,142,72)"}]
```

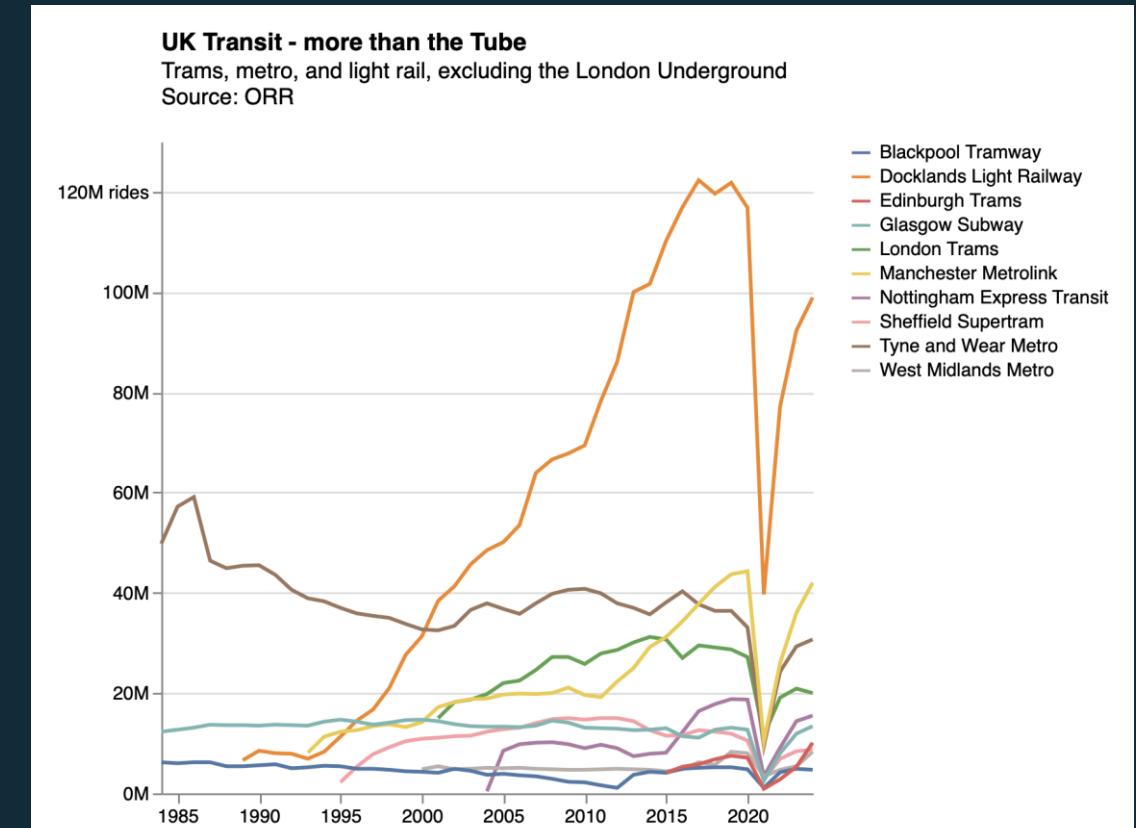
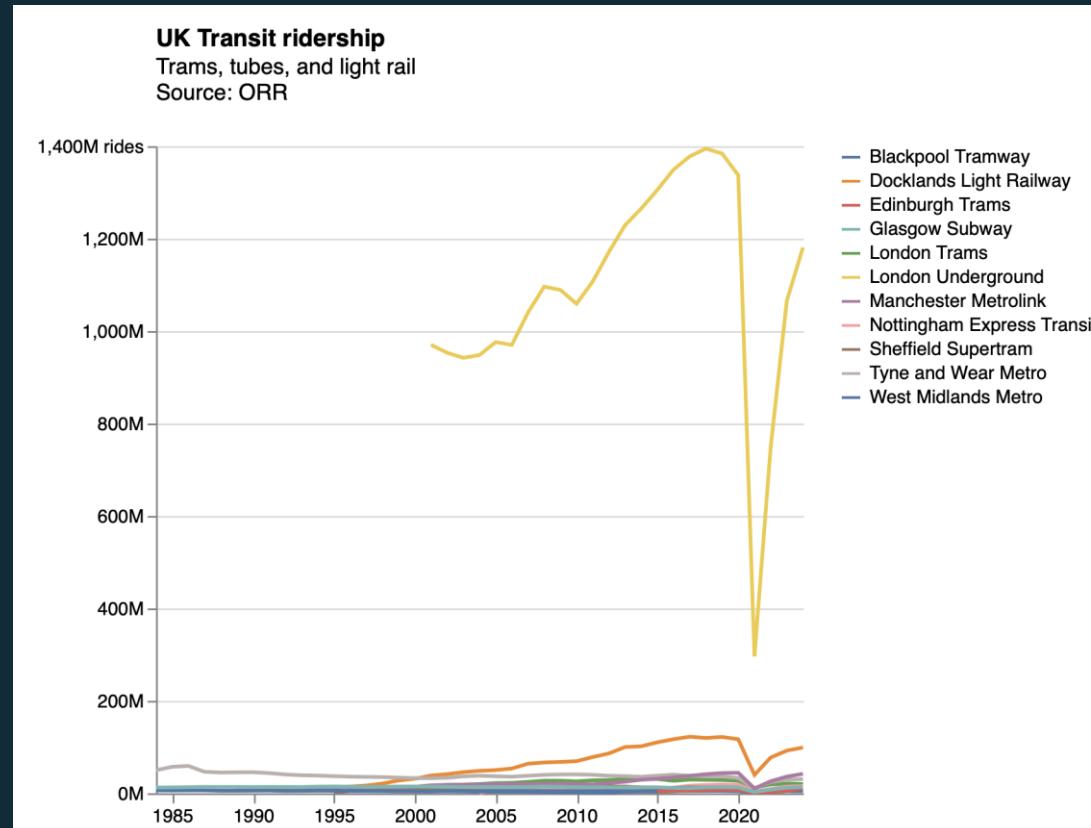


# Example 4. Vega-Lite.

Using conditionals to filter extreme values

```
"transform": [
  {
    "filter": "datum.value < 250"
  }
],
```

We can filter out the most extreme values to see what's happening in the rest of the data:

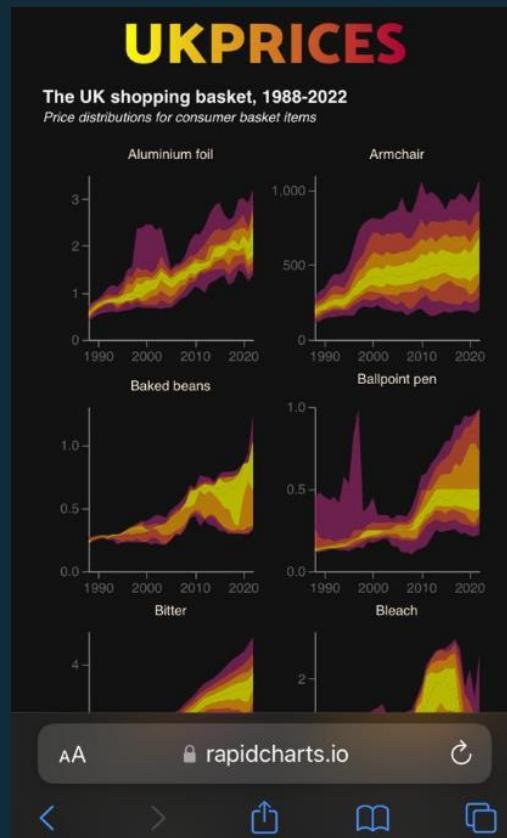


# Example 5: screen sizes

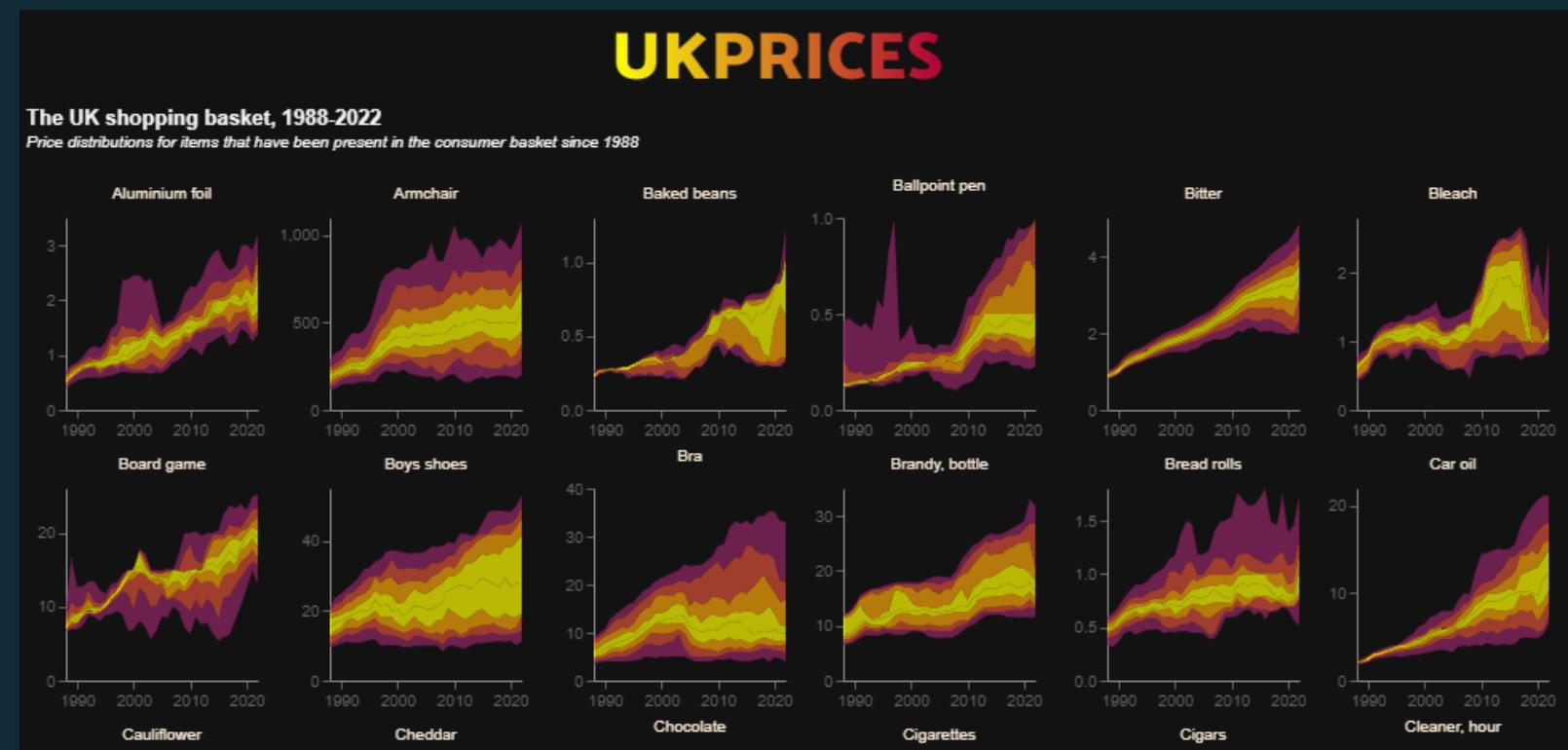
Excel | STATA | Python | JavaScript | CSS

<https://rapidcharts.io/prices>

*Phone*



*Desktop*



# Example 5: screen sizes

Excel | STATA | Python | JavaScript | CSS

```
<!-- CONDITIONAL SCRIPT TO EMBED BASED ON SCREEN WIDTH -->
<script>

// Find the current screen width:
let width = screen.width;

// Use an if function to pick the appropriate visualisation:
if (width > 950) {
  prices1 = "charts/ONSinflation/distributionsPerrenials_DarkWide.json";
} else if (width > 450) {
  prices1 = "charts/ONSinflation/distributionsPerrenials_DarkMedium.json";
} else {
  prices1 = "charts/ONSinflation/distributionsPerrenials_DarkNarrow.json";
}

// Now embed the chart, which will vary based on screen width:
vegaEmbed('#chart1', prices1, {"actions": false});

</script>
```

# Example 6 – in CSS.

Excel | STATA | Python | JavaScript | **CSS**

```
/* Some CSS to alter the colour of my site */
body{
    background-color: white;
}

/* Screen size 1 */
@media screen and (max-width: 450px) {
    body {
        background-color: lightblue;
    }
}

/* Screen size 2 */
@media screen and (max-width: 600px) {
    body {
        background-color: blue;
    }
}

/* Screen size 3 */
@media screen and (max-width: 800px) {
    body {
        background-color: pink;
    }
}
```

What colour on:

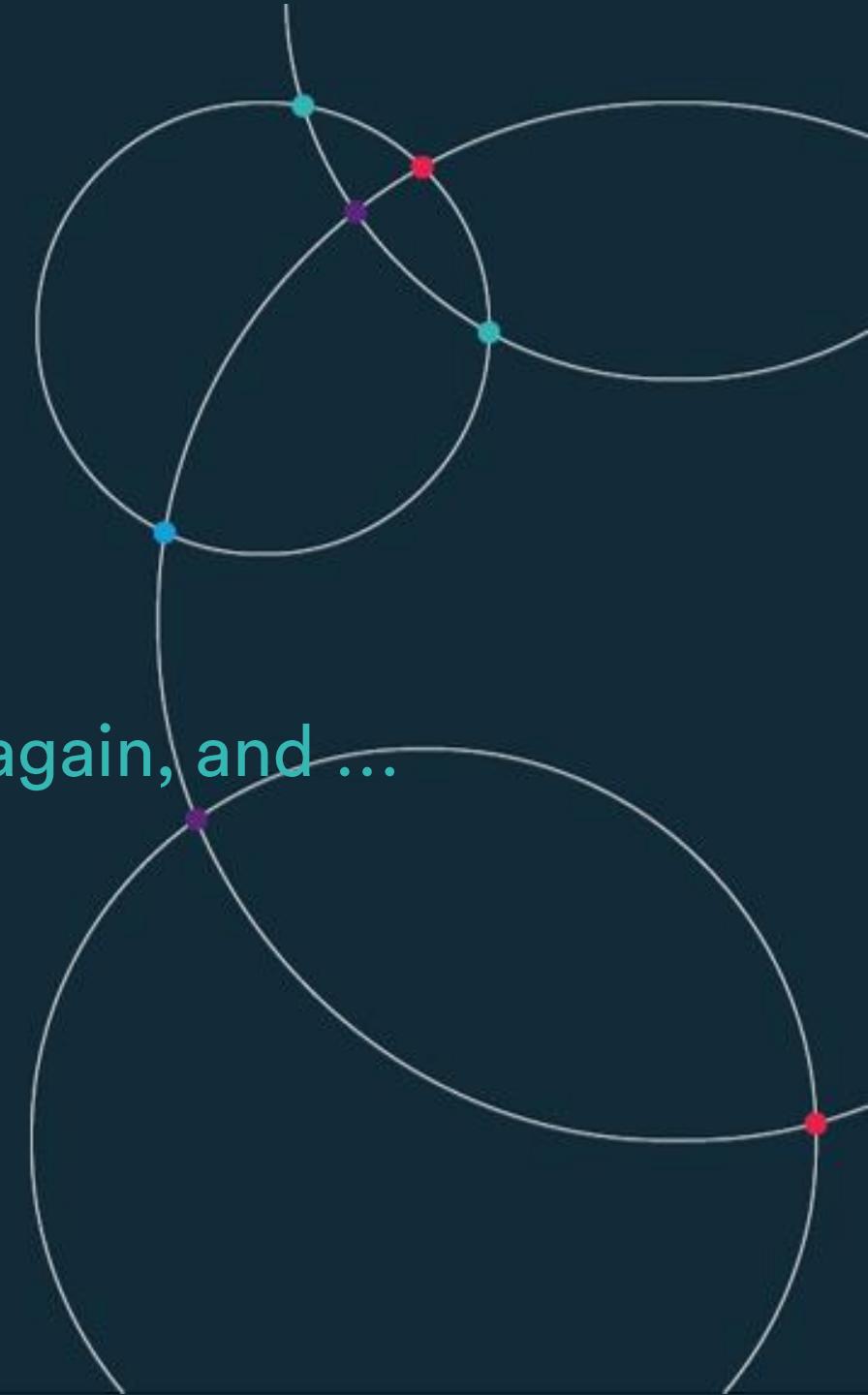
Smartphone?

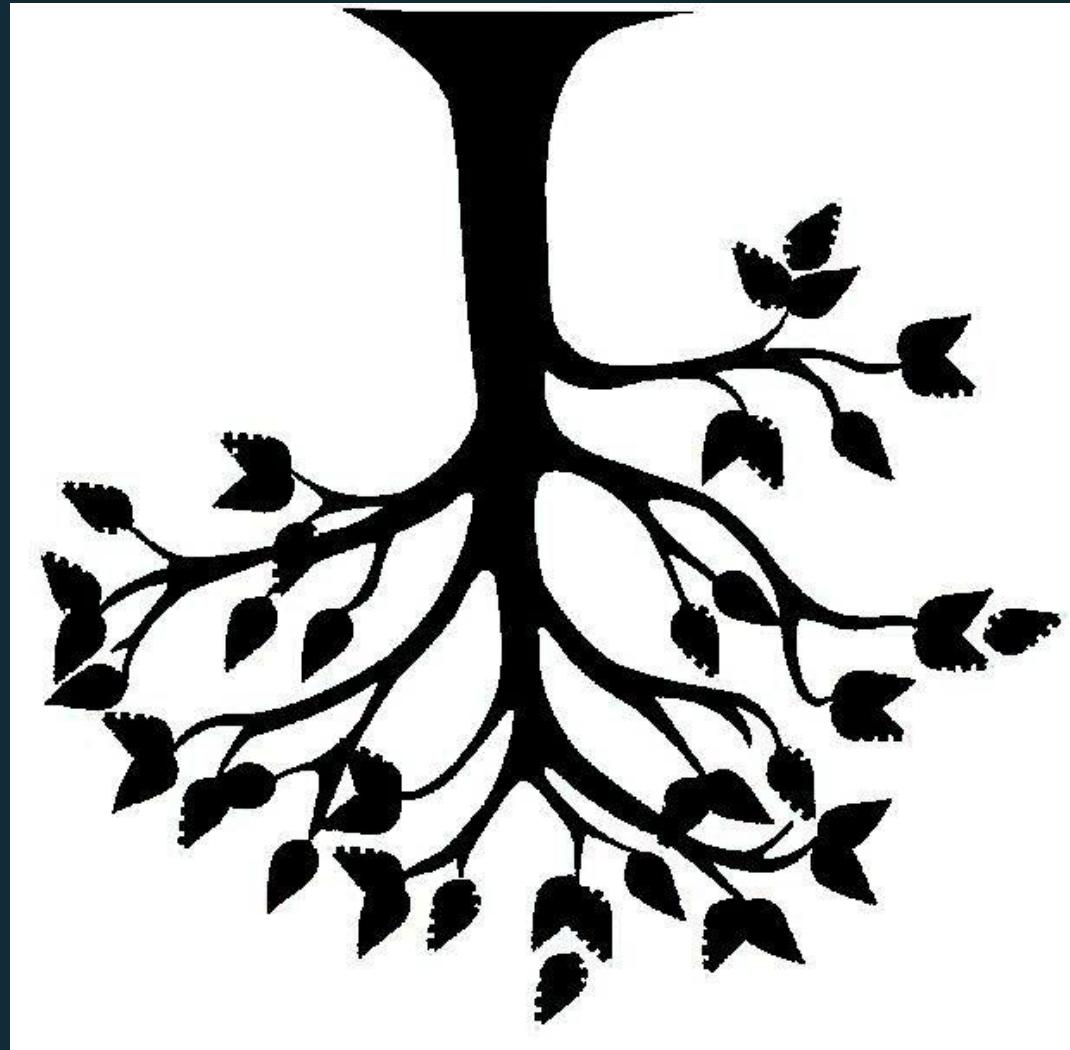
Laptop?

iPad?

## 5.2 Loops.

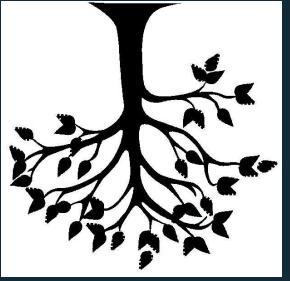
Doing a task again and again, and again and again, and ...





Code as a tree | Conditional statements influence which branch your code goes along.

Now your code is working nicely, for one country, or one company, or one year.



But you want to look at all the countries, all the companies, all the years.

# Example 6 – in CSS.

Excel | STATA | Python | JavaScript | **CSS**

```
/* Some CSS to alter the colour of my site */
body{
    background-color: white;
}

/* Screen size 1 */
@media screen and (max-width: 450px) {
    body {
        background-color: lightblue;
    }
}

/* Screen size 2 */
@media screen and (max-width: 600px) {
    body {
        background-color: blue;
    }
}

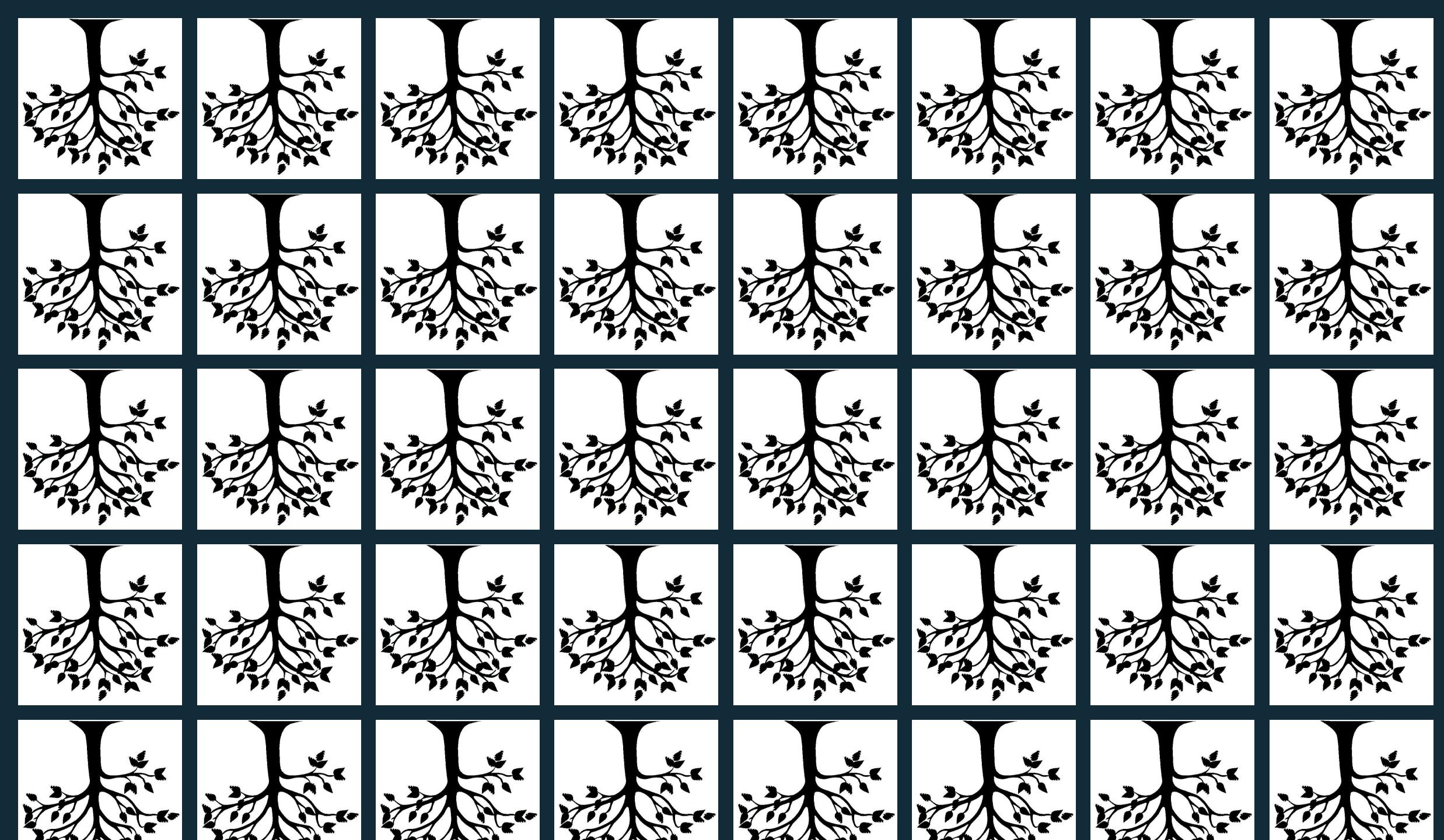
/* Screen size 3 */
@media screen and (max-width: 800px) {
    body {
        background-color: pink;
    }
}
```

What colour on:

Smartphone?

Laptop?

iPad?



# Loops.

Excel | STATA | Python | JavaScript | CSS

The Excel GUI is not built for iteration.

You can write loops in its background language VBA

This is fiddly



# Loops.

Excel | STATA | Python | JavaScript | CSS

```
// LOOPS

// For loops - two types:

forvalues i=1/100{
    display `i'
}

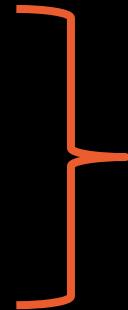
forvalues i=1(10)100{
    display `i'
}

// Generating ratios of variables to GDP
foreach i in debt deficit currentAccount investment consumption{
    gen `i'_ratio = `i'/gdp
}
```

# Loops.

Excel | STATA | Python | JavaScript | CSS

```
for i in range (1, 100):  
    print i  
  
for i in range (1, 10, 100):  
    print i  
  
for i in range (1, 100, 10):  
    print i
```



```
variables = ["debt", "deficit", "GDP", "inflation"]  
for i in variables:  
    print(i)
```

An example of subtle differences between languages / functions.

# Loops.

Excel | STATA | Python | JavaScript | CSS

```
for (statement_1; statement_2; statement_3) {  
    // Code block to run  
}
```

What happens here:

- Statement\_1 runs once, before the code block starts.
- Statement\_2 defines a condition that must hold for the code block to run.
- Statement\_3 runs each time the code block has been executed.

# Loops.

Excel | STATA | Python | JavaScript | CSS

```
for (let i = 1; i < 101; i++) {  
    console.log(i);  
}
```

```
for (let i = 1; i < 101; i+10) {  
    console.log(i);  
}
```

# Loops.

Excel | STATA | Python | JavaScript | CSS

```
// Set a list of variables:  
variables = ["debt", "deficit", "GDP", "inflation"]  
  
// We can index these:  
console.log(variables)  
console.log(variables[0])  
console.log(variables[3])  
  
// Work out how long this thing is:  
len = variables.length  
  
// Iterate though it, printing out each particular variable  
for (let i=0; i<len; i++) {  
  x = variables[i]  
  console.log(x)  
}
```

Example in this week's DropBox folder

# Loops.

Excel | STATA | Python | JavaScript | CSS

Again CSS is not really built for looping

However, it can repeat commands. An example: `@keyframes` (an At-Rule).

```
@keyframes border-pulse2 {  
 0%   { border-color: #22e68b}  
 10%  { border-color: rgba(230, 34, 75, 0)}  
 20%  { border-color: rgba(230, 34, 75)}  
 30%  { border-color: rgba(230, 34, 75, 0)}  
 40%  { border-color: rgba(0, 47, 167)}  
 50%  { border-color: rgba(230, 34, 75, 0)}  
 60%  { border-color: rgba(250, 250, 0)}  
 70%  { border-color: rgba(230, 34, 75, 0)}  
 80% { border-color: #22e68b}  
}  
  
.readingWeek {  
background-color: #122b39;  
border-radius: 10px;  
border-style: dotted;  
border-width: 5px;  
height:100%;  
animation: border-pulse2 2s infinite;  
}
```

# Loops.

Excel | STATA | Python | JavaScript | CSS

Again CSS is not built for looping

However, it can iterate over certain commands for a given time. One example

```
@keyframes border-pulse2 {  
 0%   { border-color: #22e68b}  
 10%  { border-color: rgba(230, 34, 75, 0)}  
 20%  { border-color: rgba(230, 34, 75)}  
 30%  { border-color: rgba(230, 34, 75, 0)}  
 40%  { border-color: rgba(0, 47, 167)}  
 50%  { border-color: rgba(230, 34, 75, 0)}  
 60%  { border-color: rgba(250, 250, 0)}  
 70%  { border-color: rgba(230, 34, 75, 0)}  
 80% { border-color: #22e68b}  
}  
  
.readingWeek {  
  background-color: #122b39;  
  border-radius: 10px;  
  border-style: dotted;  
  border-width: 5px;  
  height:100%;  
  animation: border-pulse2 2s infinite;  
}
```



**Week 6**

**Reading week**

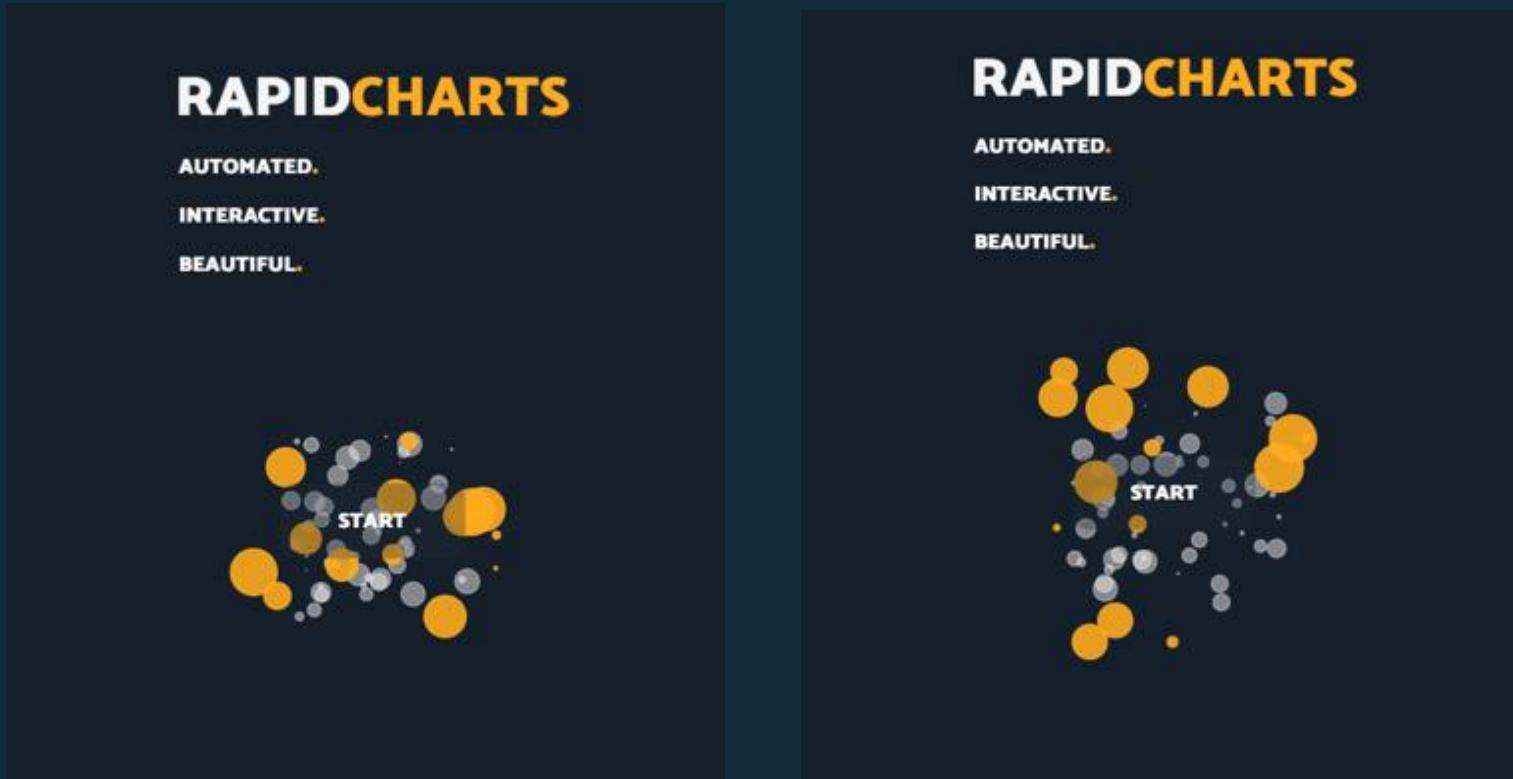
There are no classes or office hours this week.

*Relax.*

*Then work on your project!*

# Using loops.

Exploding bubbles



[https://rdeconomist.github.io/index\\_old](https://rdeconomist.github.io/index_old)

Code intuition:

For n bubbles, pick a size, x direction and y direction. Use random numbers so different each time.

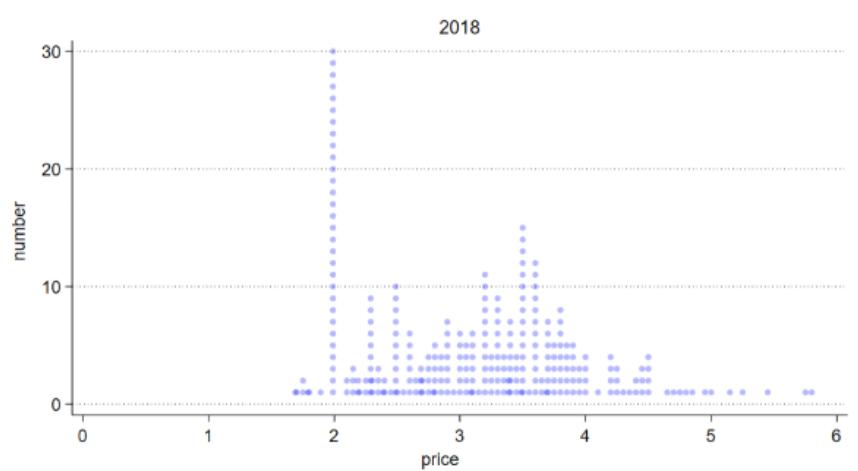
# Using loops.

## Stop frame animation

Example: animation: pints of bitter

The humble pint of bitter is one of the most sampled prices in the UK CPI. Here is how we went from 80p a pint in 1988 to a huge range of prices, including £6 a pint today. Note the peak that emerges at £2, which a knowledgeable drinker tells me is the 'Wetherspoons Effect'.

The video is created using simple stop frame animation (running a loop to create 395 monthly charts).



Watch the video here:  
<https://richarddavies.io/research/prices>

Code intuition:

For dates i, draw histogram if date==i. Save chart as png.

# Using loops.

Batching analysis

**Problem:** We often have too much data

*How can visualising help us understand the data in the early stage of a project?*

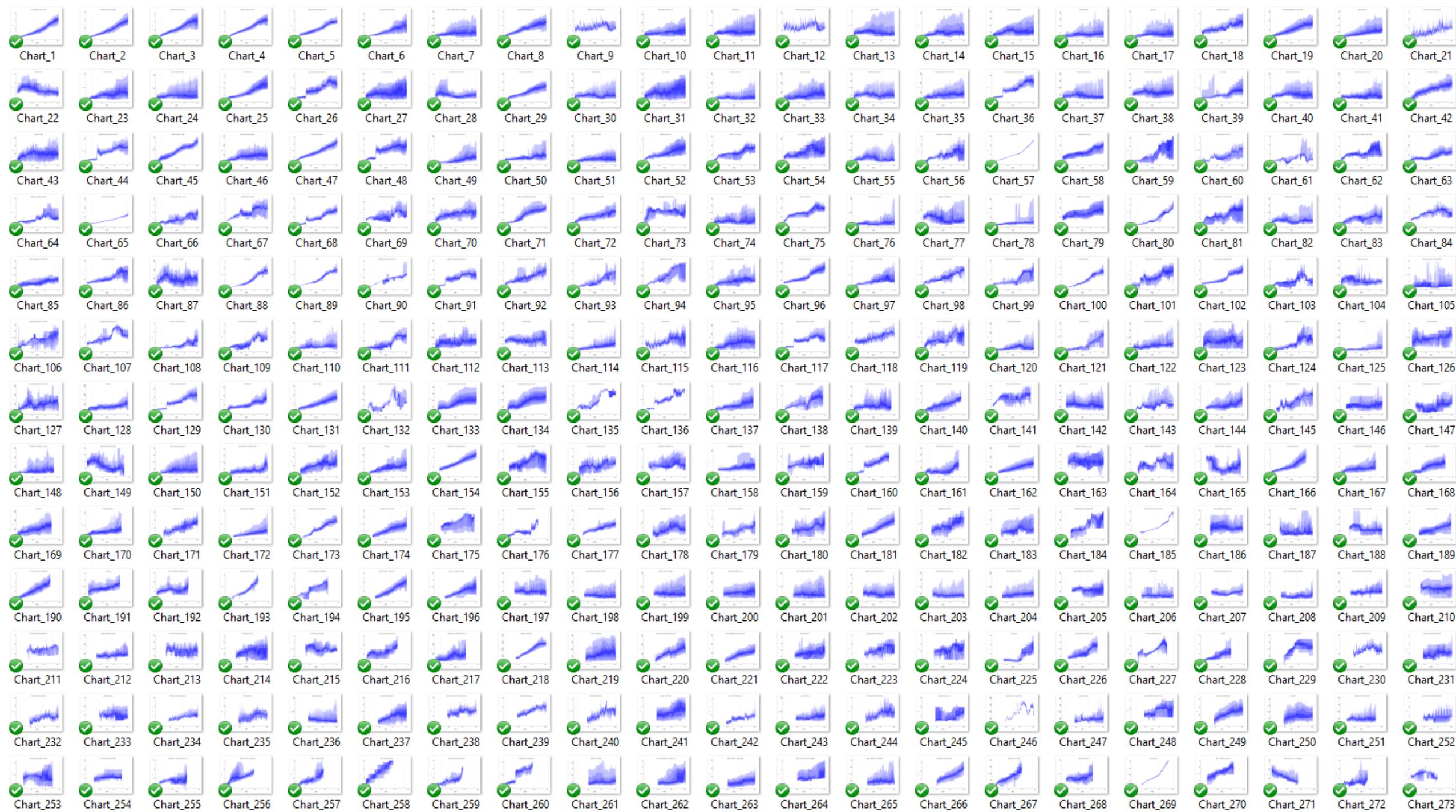
Code intuition:

- Think of a chart that summarises the data in some useful way:

[Useful options: histogram, percentiles as a swathe.]

- Code this chart for one part (one firm, one year, one product).
- Use a loop: for each product, draw the chart

Example: UK CPI data, which has ~1300 products.



# It's time to complete your unit evaluation surveys



**[bristol.ac.uk/unit-evaluation](http://bristol.ac.uk/unit-evaluation)**

## 5.3 Building dashboards.

Using loops and conditionals to analyse data



# Break.

Find two series on FRED that you are interested in.  
Make a note of their codes:  
<https://fred.stlouisfed.org/>



# Practical.

Looping over a list of countries to create dynamic API-linked charts.

