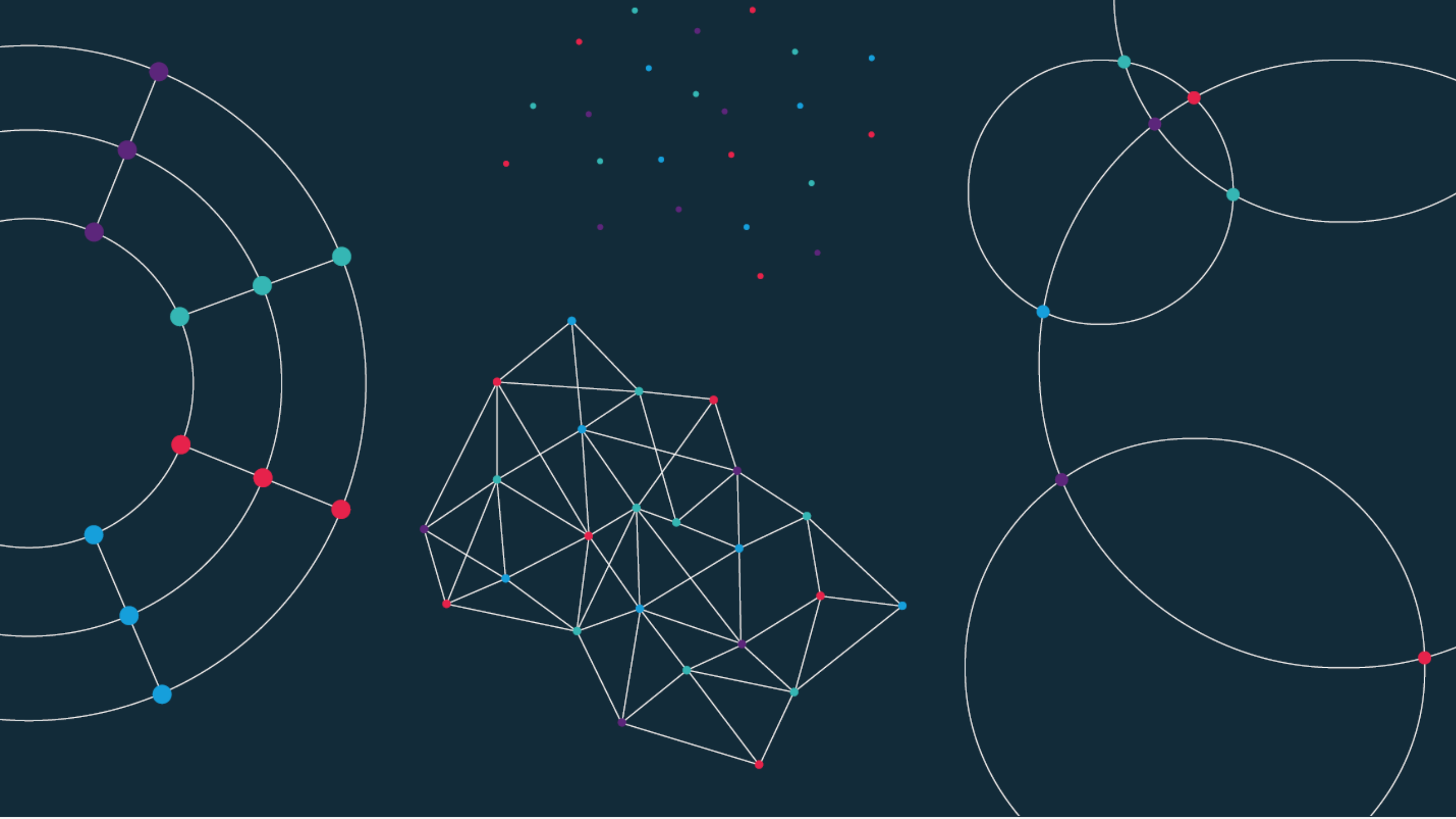
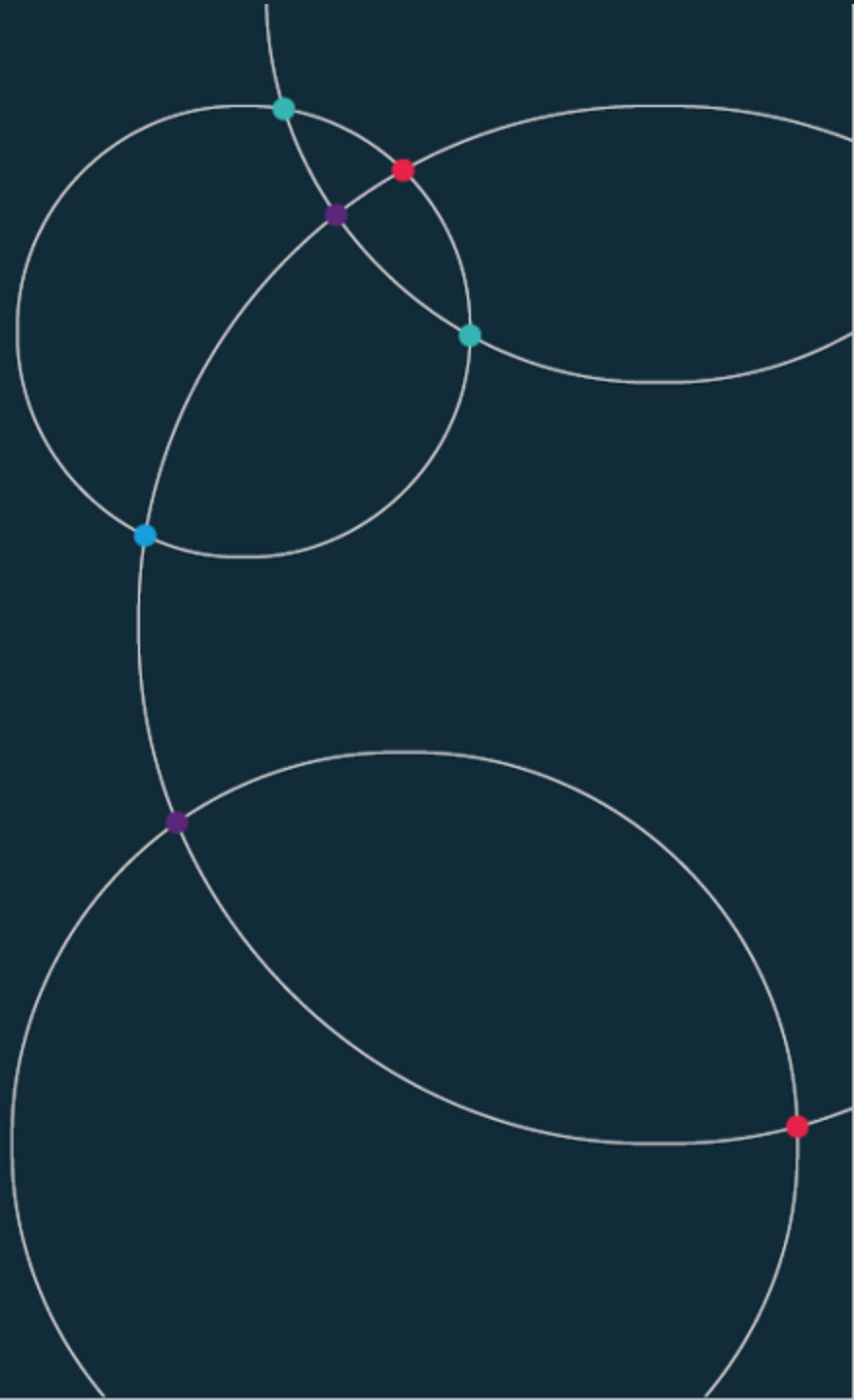


Automated Data Visualisation for Policymaking.

Session 5
Autumn 2024



Nearly halfway...

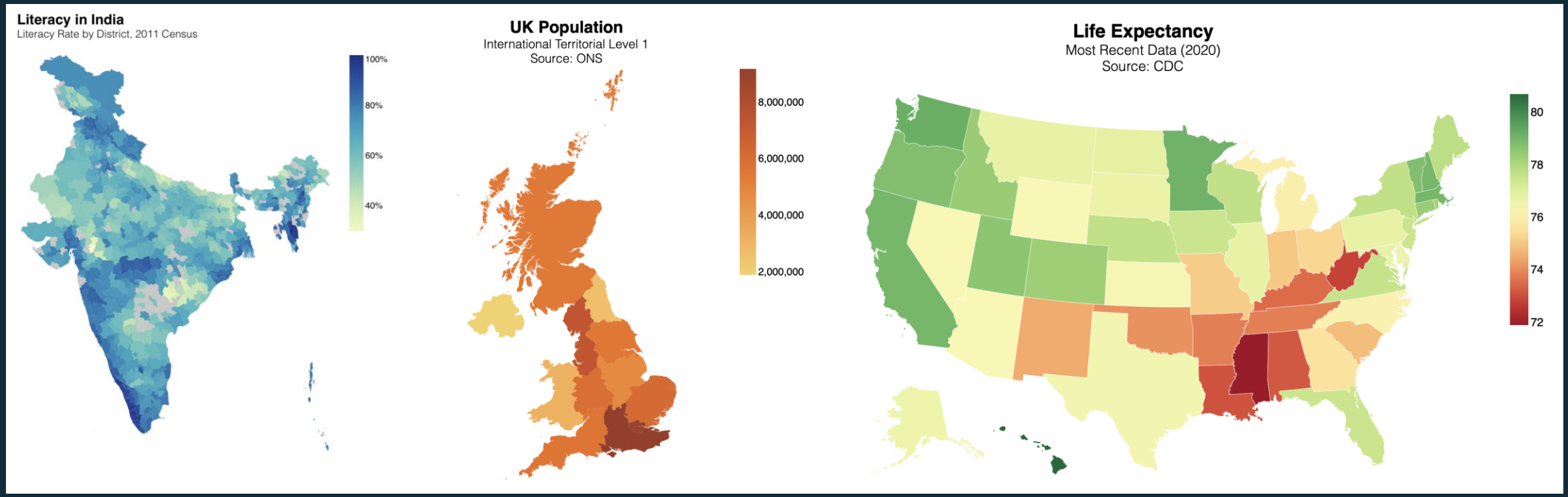
Re- cap, and what's coming next

- W1. Your Live Site. HTML, CSS, JavaScript. GitHub and embedding.
- W2. Data debating – chart design. Raw files, editing data. String and date functions.
- W3. The Language of Data. Why visualisation is vital in policy. Manipulating data with Python.
- W4. Data. Collection, storage and biases. "Tidy" data storage. Data access. APIs. Scraping with Python.
- W5. Programming. Control structures. Loops and conditionals. Combining APIs with loops in Python.
- *W6. Break. No classes. No homework. Tip: Complete all your portfolio work and start on project.*
- 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 7.

After the break

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



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).

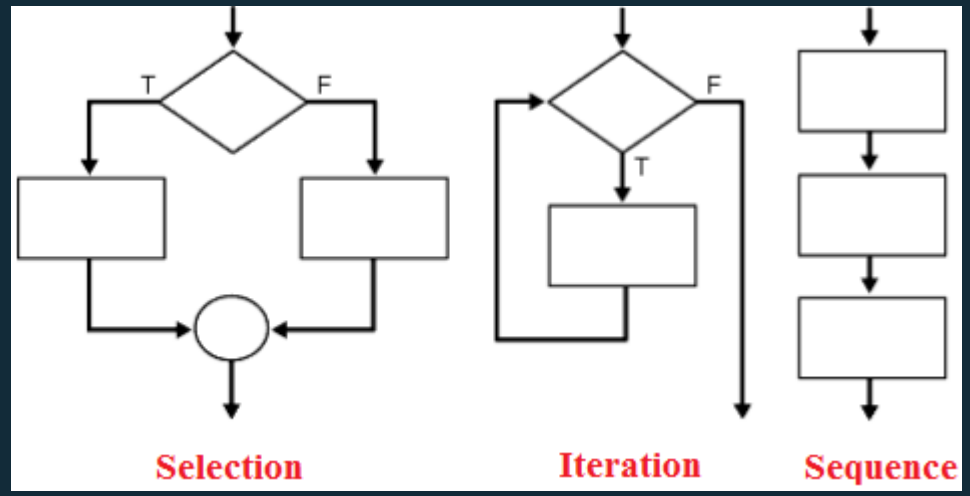
- Do something many times.
  - Dynamic programming / maximisation;
  - Batches of analysis: downloading, cleaning, charting.

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.

History: Joseph Marie Jacquard.

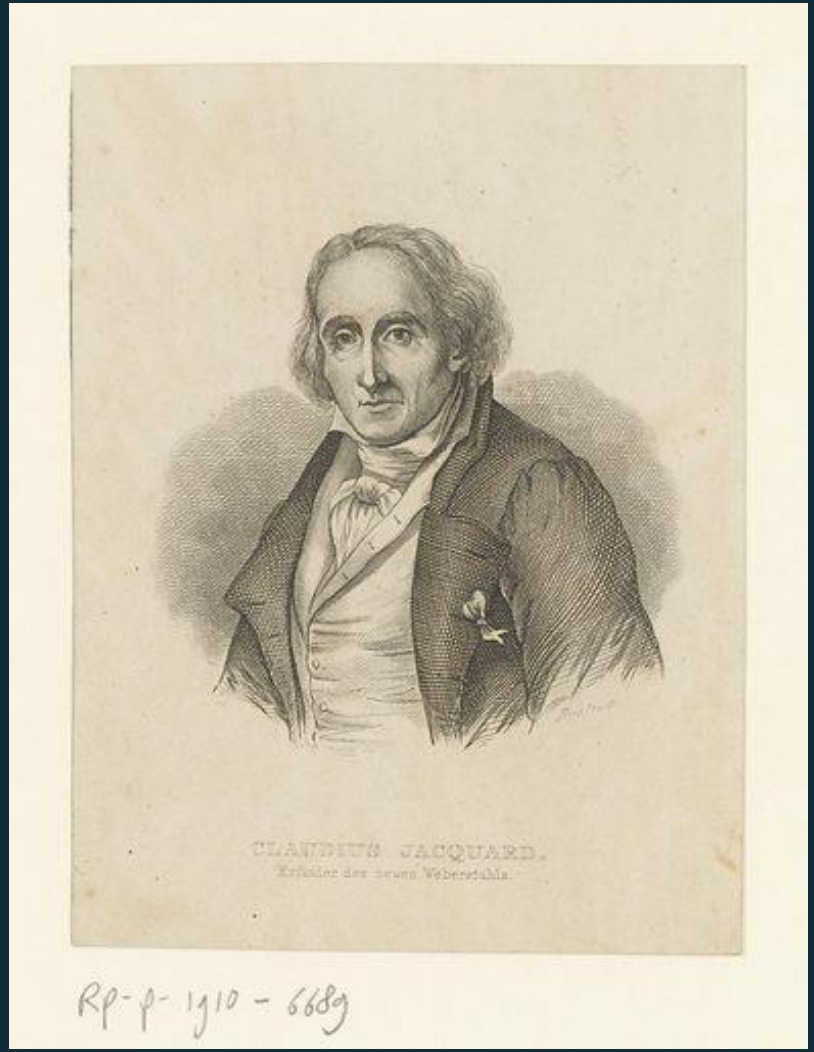
The origins of programming

Jacquard (1752-1834) was a French merchant, weaver, and inventor.

From his base in Lyon – which he defended on the side of the revolutionaries during the French revolution – he developed the Jacquard machine.

It was the first programmable loom.

Though unpopular with textile workers, whose labour it automated, the loom would spread across France, and eventually the industrialising world.



Jacquard

History: Jaquard’s Machine.

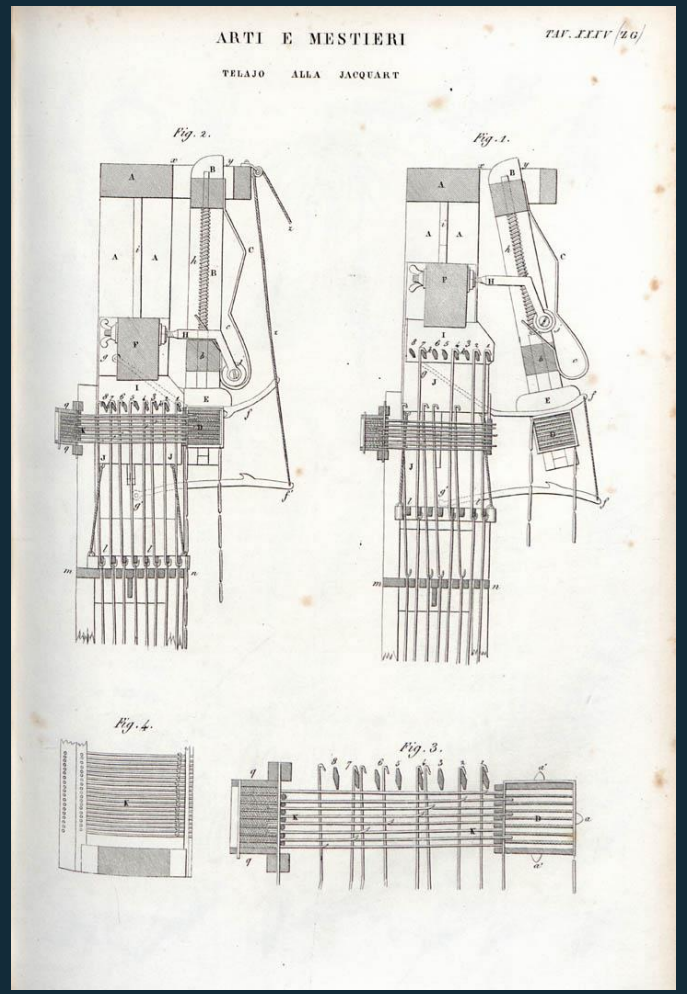
The origins of programming

Invented by Jacquard in 1804, the loom revolutionised textile production. It used punch cards to automate complex patterns.

(Museum of Obsolete Media)

This was a revolutionary change – it could be reprogrammed just by punching holes into a card.

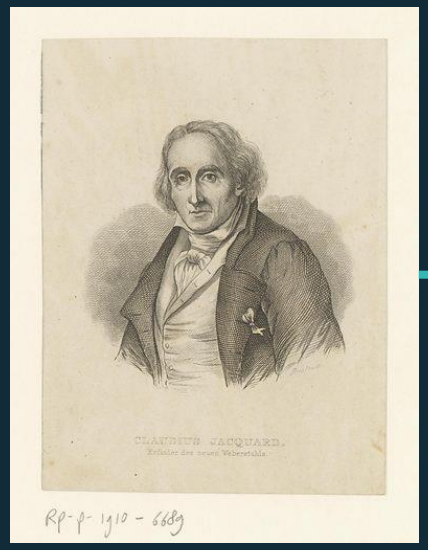
It was not a computer itself but the innovation of the loom’s binary system laid the groundwork for programmable computers.



History: Charles Babbage.

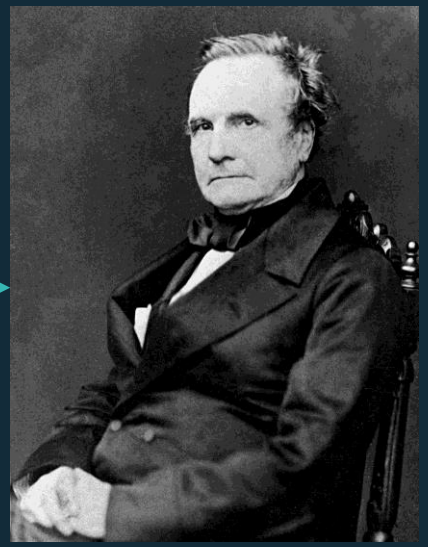
The father of computing

Jacquard's programmable loom would inspire an Englishman, Charles Babbage (1791-1871).
Babbage was a polymath - mathematician, philosopher, engineer – who **designed** the first computer.



Jacquard
Programmable Loom

Punch card
programming



Babbage
First computer design

"the Analytical Engine weaves algebraical patterns just as the Jacquard loom weaves flowers and leaves"
Ada Lovelace

History: Charles Babbage.

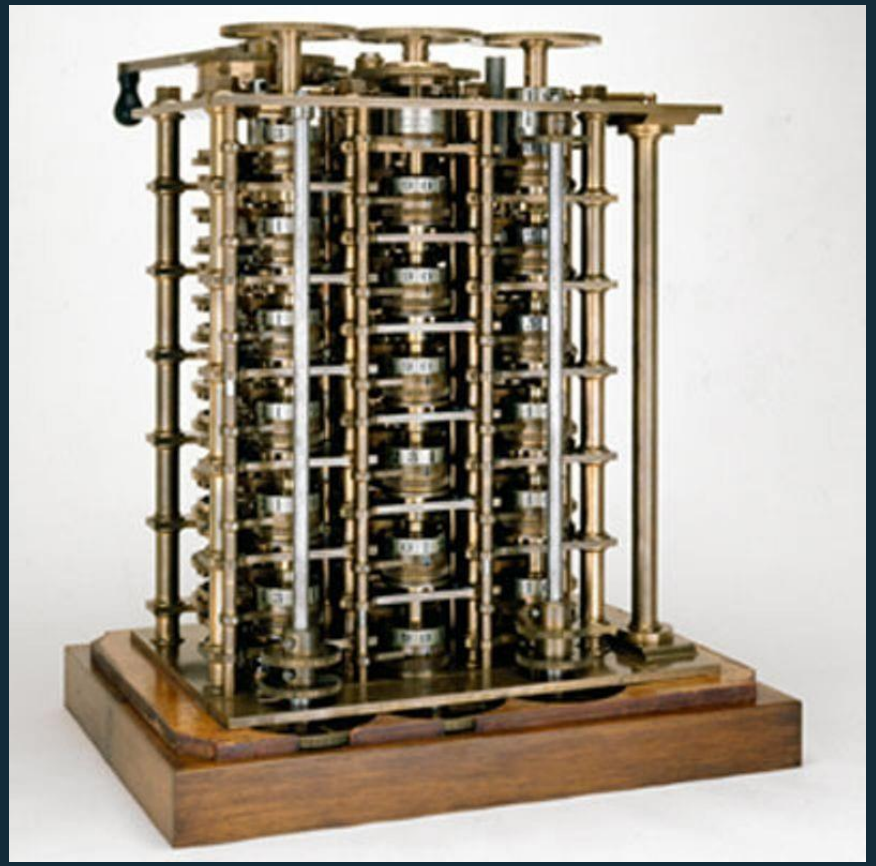
His machines

Charles Babbage, the “father of computing,” designed the Difference Engine and Analytical Engine to automate calculations.

Difference Engine. A mechanical calculator that automates calculations with finite differences, using repeated addition in place of multiplication or division.

Analytical Engine. A general-purpose computing machine concept. It featured punched cards for programming, memory (store) and processing (Mill) units. It would execute conditional operations.

The analytical engine introduced the tools we use today: loops and conditionals, among others. If it were constructed in Babbage’s lifetime, it would have been the first computer.



History: Ada Lovelace.

English mathematician and writer

Augusta Ada King, Countess of Lovelace. 1815-1852

Aged sixteen Ada met Charles Babbage and corresponded about Babbage's Difference Engine and The Analytical Engine.

Translated an article by an Italian engineer (Luigi Menabrea) on Babbage's Analytical Engine to English. The translation took 9 months. An appendix with commentary was three times as long as the translation. The translated notes and commentary were published in 1943.

Ada recognised the ability of Babbage's Analytical Engine to carry out complex sequences of mathematical operation. The notes include an example of how to calculate Bernoulli numbers – this is regarded as the first computer program.

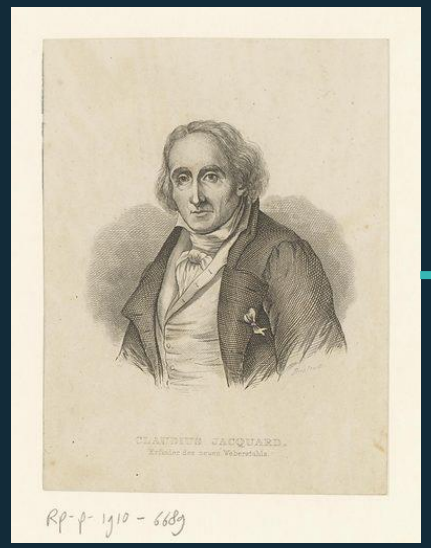
- International Ada Lovelace Day: second Tuesday of October. Celebrates the contributions of women in STEM.

Taylor, R. (Ed.). (1843). Scientific Memoirs, Selected from the Transactions of Foreign Academies of Science and Learned Societies, and from Foreign Journals (Vol. 3, pp. 660–702). R. and J. E. Taylor.

History: Ada Lovelace.

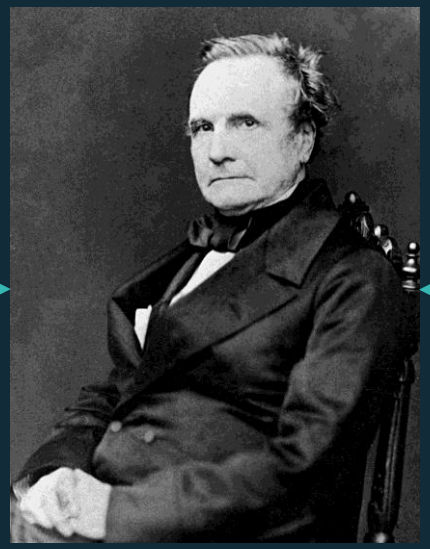
The first programmer

In correspondence with Babbage, Ada Lovelace (1815-1852) would program the first algorithm for the analytical machine, an algorithm to compute Bernoulli numbers.



Jacquard
Programmable Loom

Punch card programming



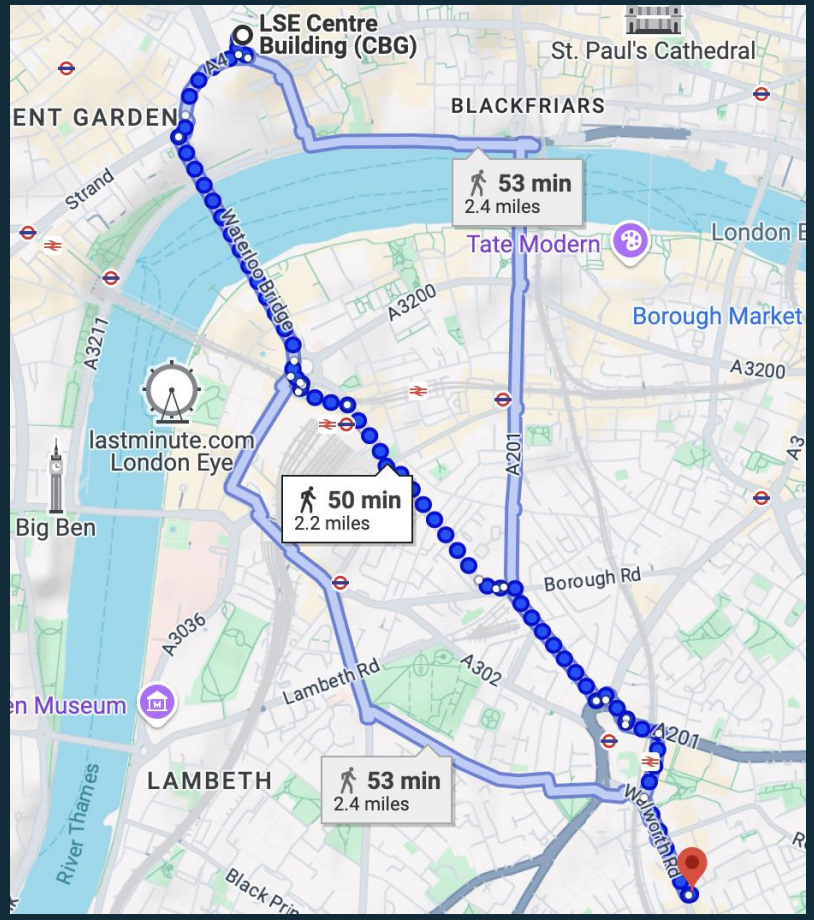
Babbage
First computer design

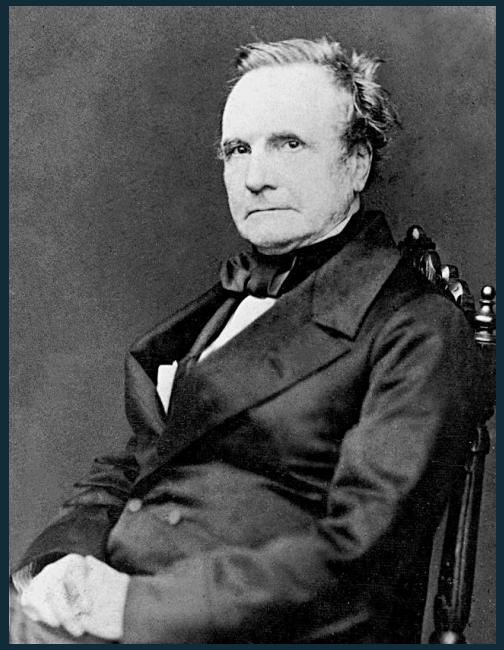
Bernoulli sequence

Lovelace
The first programmer

London tour IV.

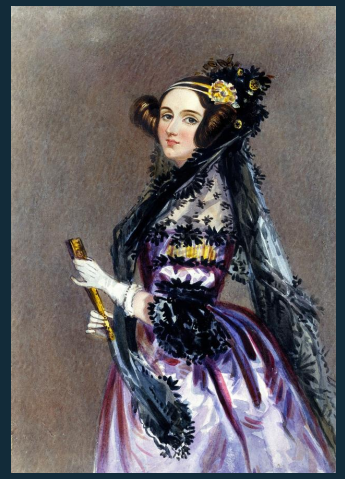
Charles Babbage.
Plaque on Larcom Street. 50 min walk.

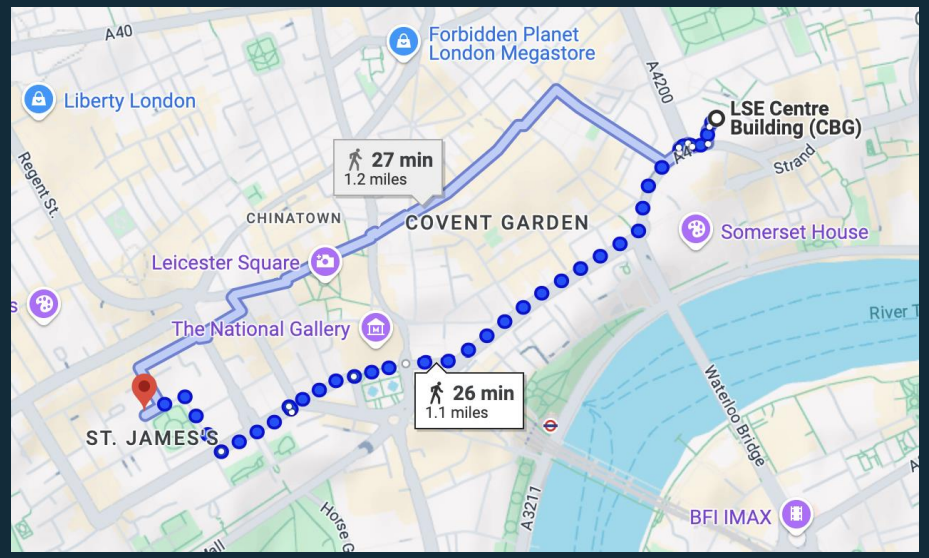


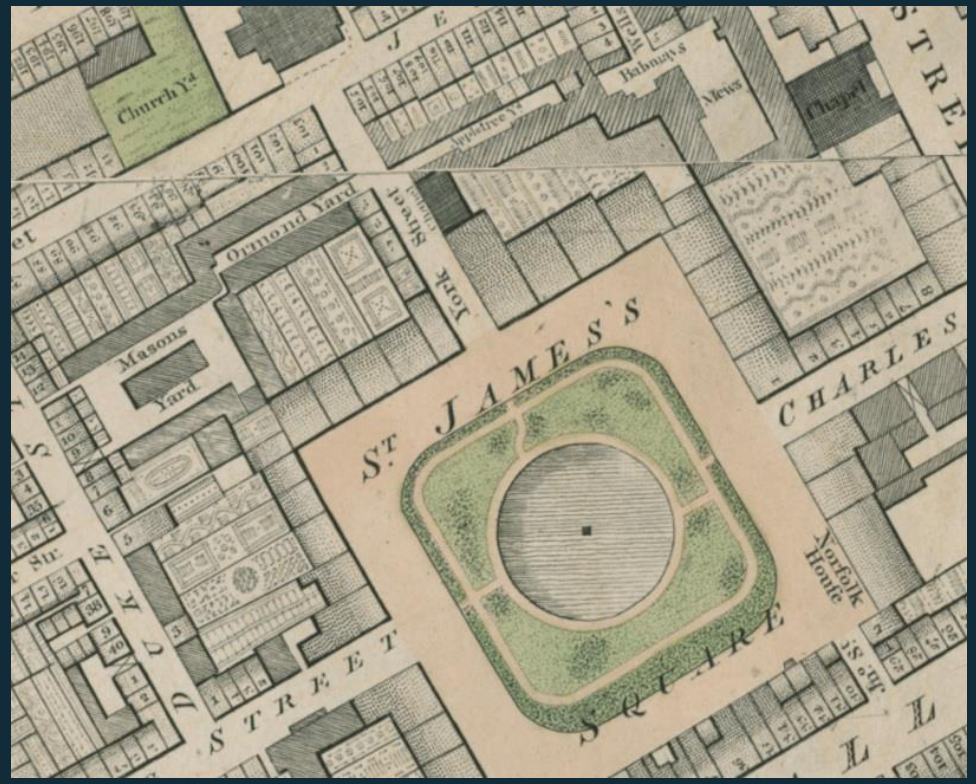


London tour V.

Ada Lovelace
Blue plaque St James Sq. 26m walk.

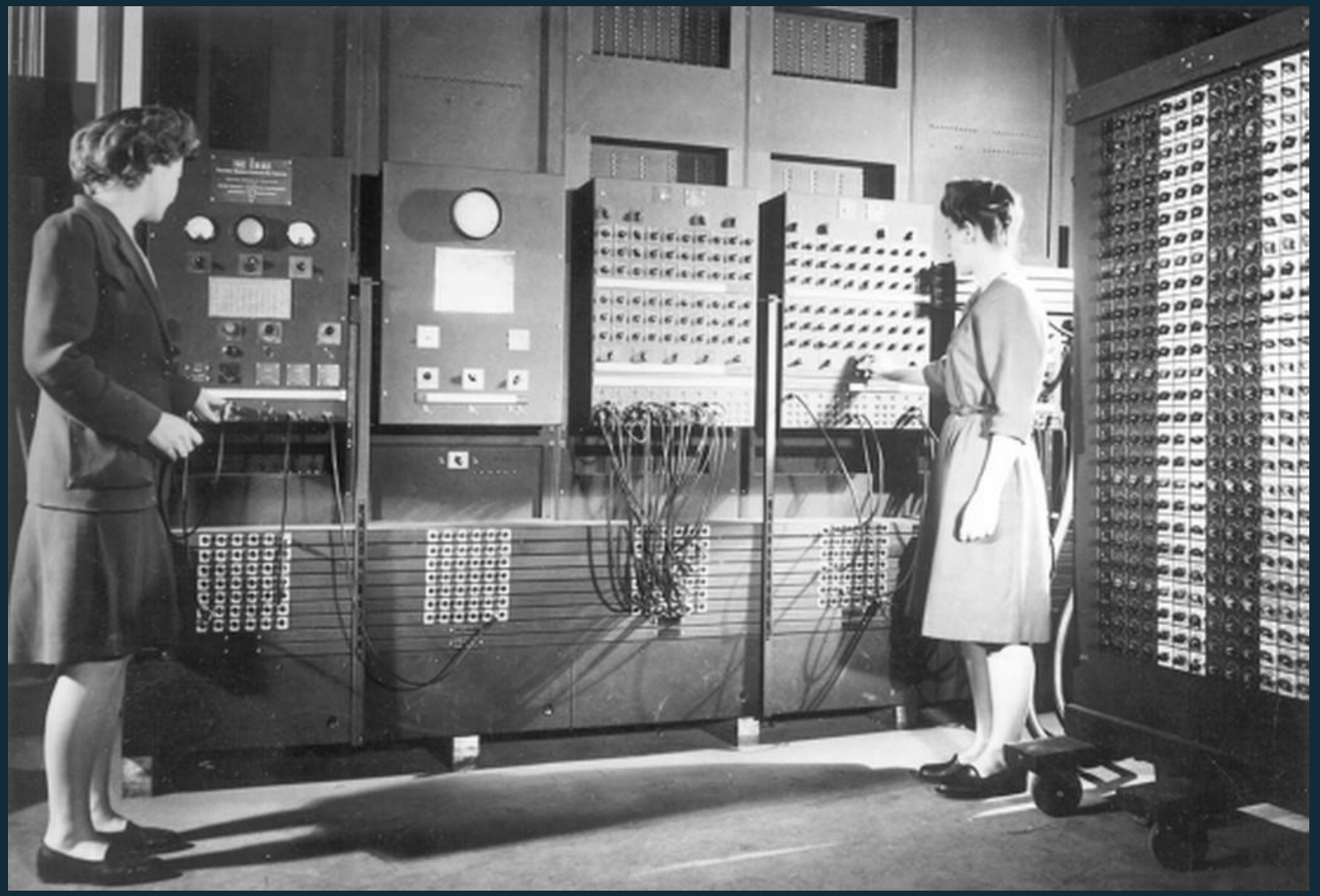






History: ENIAC.

The first computer – conditionals and loops



The Electronic Numerical Integrator and Computer was the first general purpose computer.

Completed in 1945, at University of Pennsylvania's Moore School of Engineering.

It was programmable, electronic and Turing complete.

https://penntoday.upenn.edu/news/worlds-first-general-purpose-computer-turns-75

How many languages?

How many in the room?

Approximately 7,000 spoken languages globally.

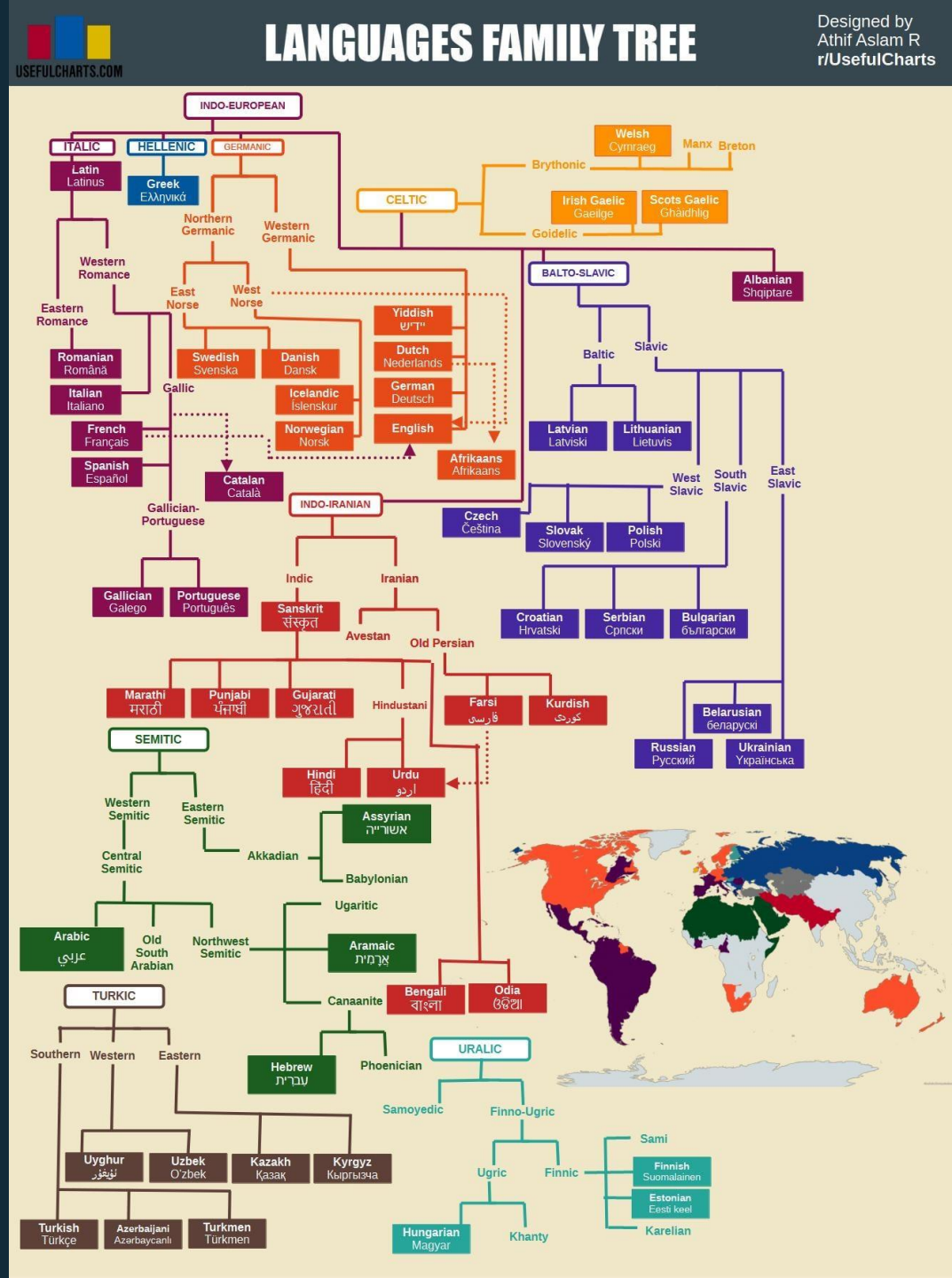
Languages are grouped into families based on common ancestral roots.

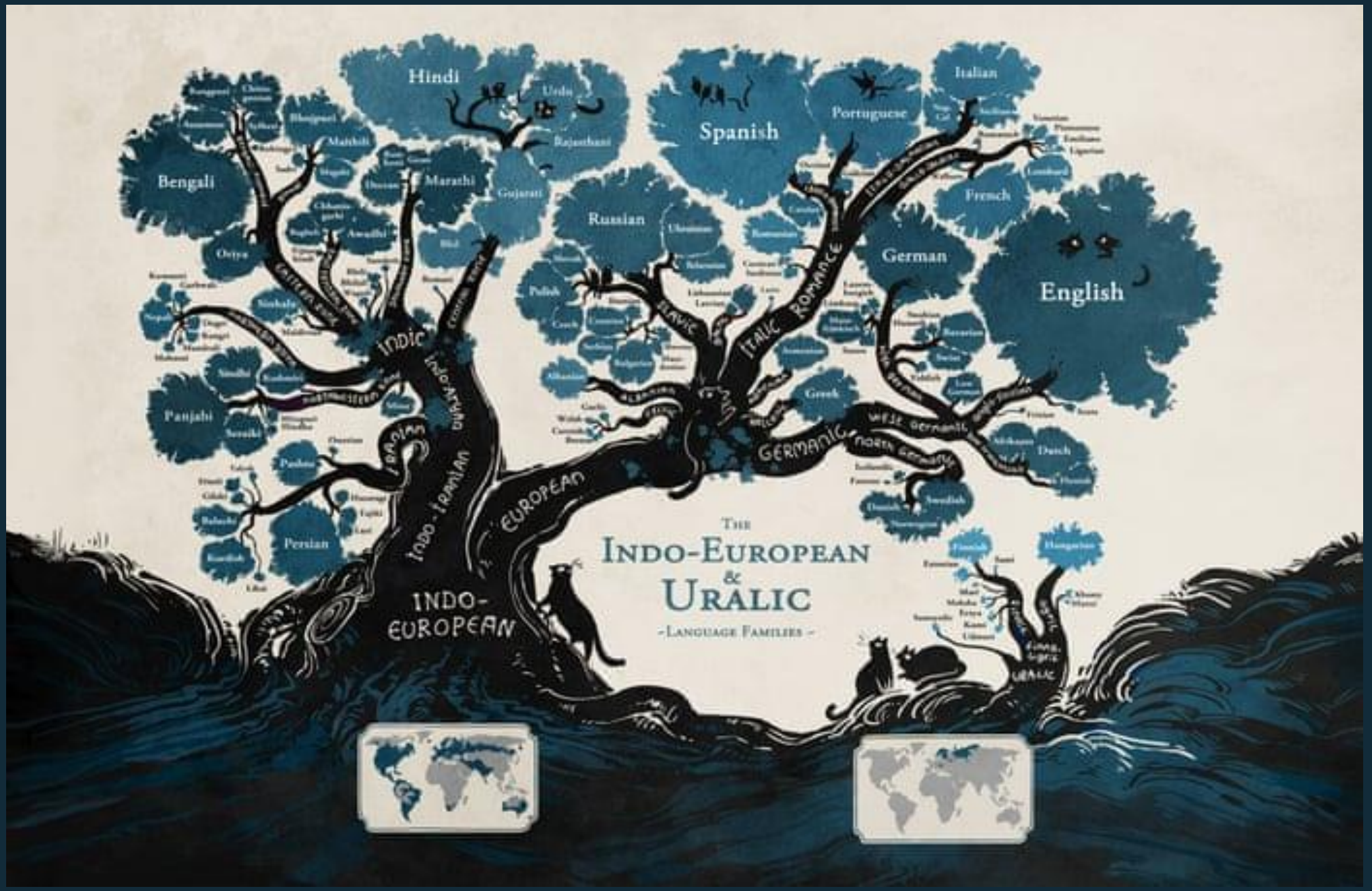
The largest family is Indo-European, covering languages spoken by over 3 billion people, e.g. English, Spanish, Hindi etc.

Languages evolve. Latin developed into entirely new languages (e.g., Italian, French, Spanish).

Around 40% of languages are considered endangered. Most are spoken by fewer than 10,000 people.

Austin, P. K., & Sallabank, J. (Eds.). (2011). The Cambridge Handbook of Endangered Languages. Cambridge: Cambridge University Press.





https://www.theguardian.com/education/gallery/2015/jan/23/a-language-family-tree-in-pictures

How many languages?

Five?

How many languages?

Three?

How many languages?

Most high-level languages have loops that you will be able to understand.

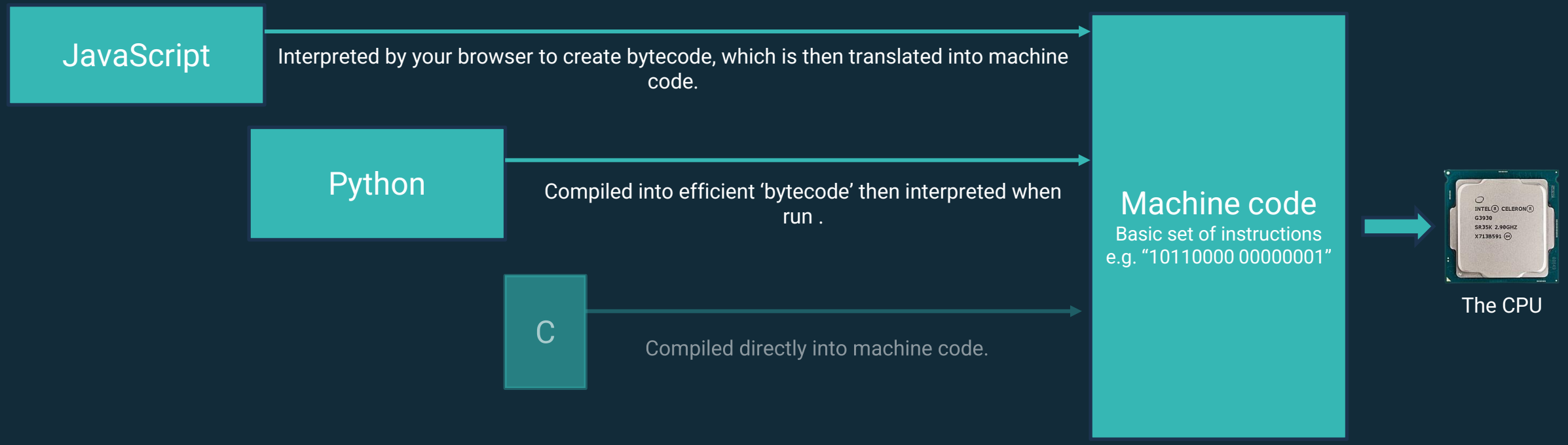
From code to metal.

Many languages, one result

Why are languages so similar?

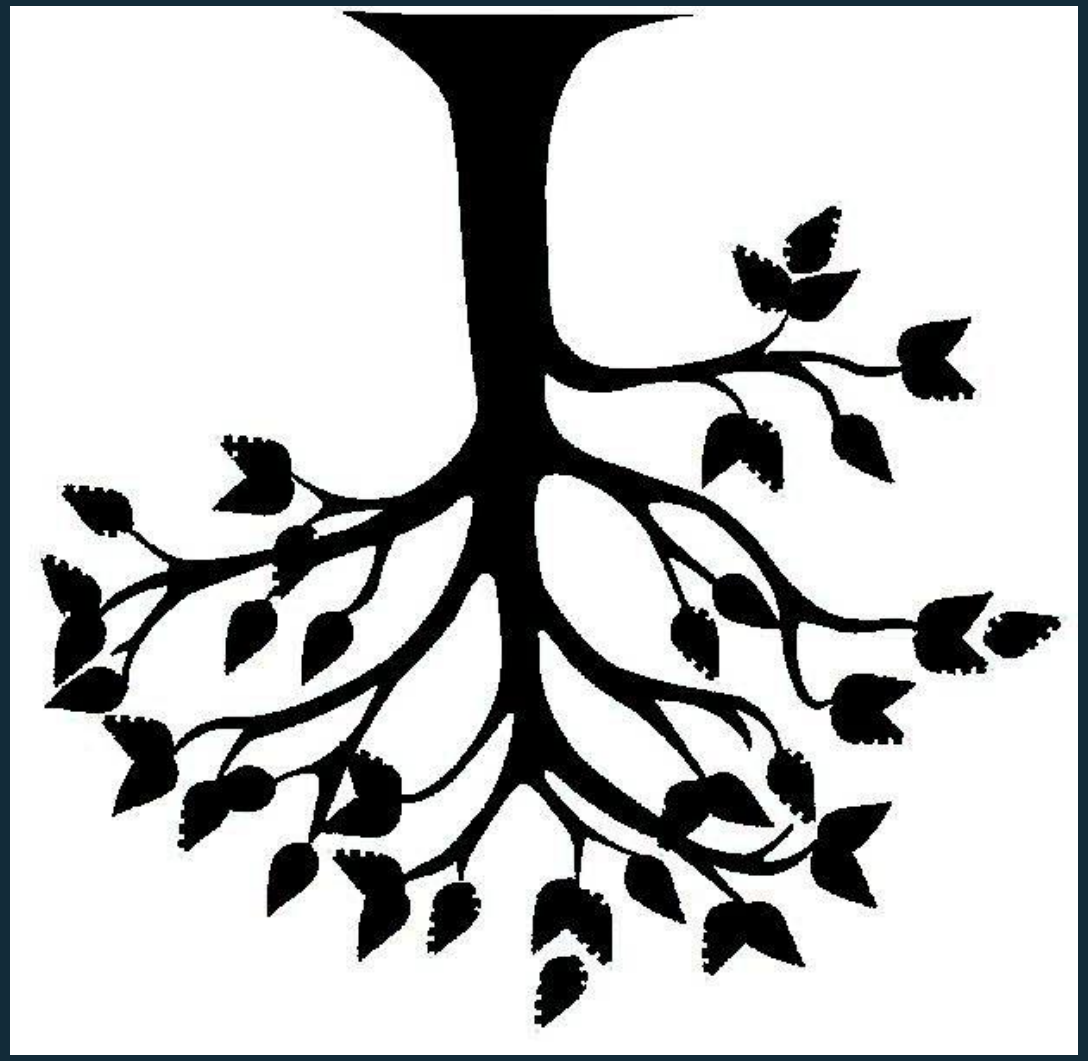
Every language has the same purpose – to turn abstract instructions into 0s and 1s.

They just take different routes.



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 (only in JavaScript).

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)
10	Ultra-Short						
11	0 1/8% Treasury Gilt 2023	GB00BL68HG94	31-Jan-2023	08-Apr-2020	31 Jan/Jul	20-Jan-2023	33,824
12	0¾% Treasury Gilt 2023	GB00BF0HZ991	22-Jul-2023	20-Jul-2017	22 Jan/Jul	12-Jan-2023	34,359
13	2¼% Treasury Gilt 2023	GB00B7Z53659	07-Sep-2023	12-Jun-2013	7 Mar/Sep	24-Feb-2023	35,922
14	0 1/8% Treasury Gilt 2024	GB00BMGR2791	31-Jan-2024	07-Oct-2020	31 Jan/Jul	20-Jan-2023	35,551
15	1% Treasury Gilt 2024	GB00BFWFPL34	22-Apr-2024	25-Jul-2018	22 Apr/Oct	13-Oct-2022	35,638
16	2¾% Treasury Gilt 2024	GB00BHBFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806
17	0¼% Treasury Gilt 2025	GB00BLPK7110	31-Jan-2025	02-Jul-2021	31 Jan/Jul	20-Jan-2023	29,353
18	5% Treasury Stock 2025	GB0030880693	07-Mar-2025	27-Sep-2001	7 Mar/Sep	24-Feb-2023	37,339
19	0 5/8% Treasury Gilt 2025	GB00BK5CVX03	07-Jun-2025	03-Jul-2019	7 Jun/Dec	28-Nov-2022	44,623
20	2% Treasury Gilt 2025	GB00BTHH2R79	07-Sep-2025	20-Mar-2015	7 Mar/Sep	24-Feb-2023	39,934
21	Short						
22	0 1/8% Treasury Gilt 2026	GB00BL68HJ26	30-Jan-2026	03-Jun-2020	30 Jan/Jul	19-Jan-2023	35,316
23	1½% Treasury Gilt 2026	GB00BYZW3G56	22-Jul-2026	18-Feb-2016	22 Jan/Jul	12-Jan-2023	43,651
24	0 3/8% Treasury Gilt 2026	GB00BNNGP668	22-Oct-2026	03-Mar-2021	22 Apr/Oct	13-Oct-2022	32,890
25	4 1/8% Treasury Gilt 2027	GB00BL6C7720	29-Jan-2027	13-Oct-2022	29 Jan/Jul	19-Jan-2023	4,445
26	1¼% 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 us 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:

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

// Problem variable: looks like this:
// "0 5/8% Treasury Gilt 2025"
// We want to ge 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+type22
drop T

/// Genearting a substring, based on the kown position of a chatacter:
replace type22 = substr(type22, 1, strpos(type22, "2") - 1)

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

github.com/RDeconomist/RDeconomist.github.io/blob/main/data/giltsCleaningData.do

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	2¾% Treasury Gilt 2024	GB00BHBFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806	NaN
3	0¼% 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	2¾% Treasury Gilt 2024	GB00BHBFH458	07-Sep-2024	12-Mar-2014	7 Mar/Sep	24-Feb-2023	35,806	NaN
3	0¼% 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

github.com/jhellingsdata/RADataHub/blob/main/misc/cleaning_dmo_data.ipynb

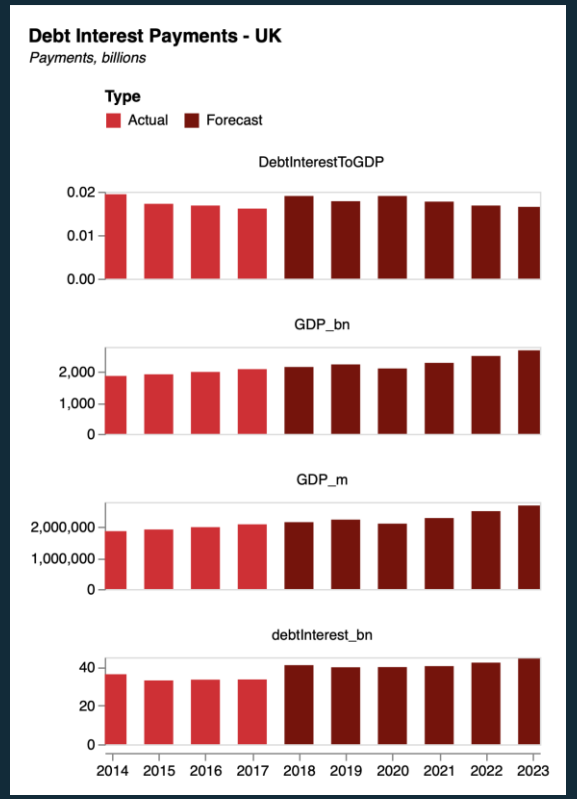
Example 2. In Vega-Lite.

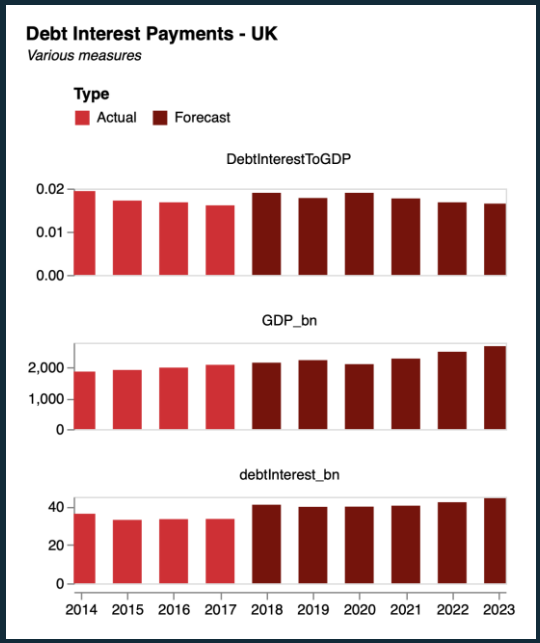
Using conditionals in your charts

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





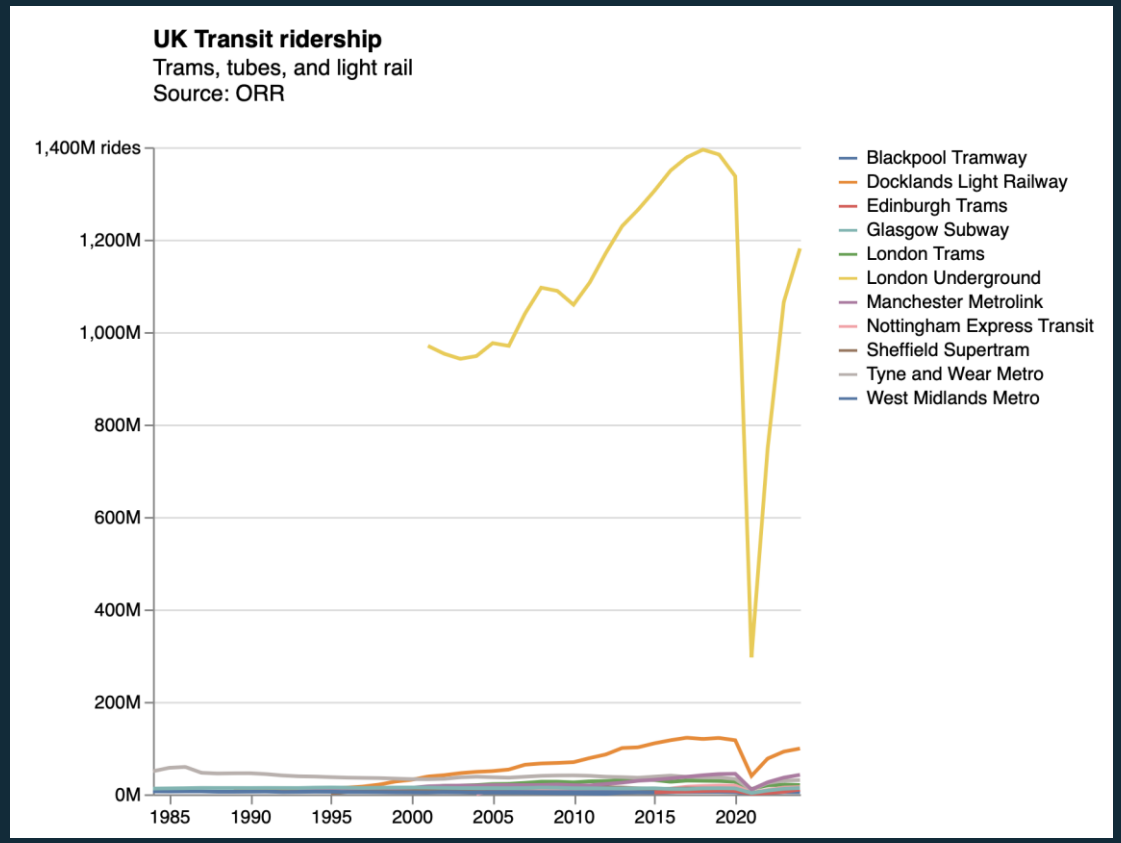

```
"transform": [
  {"filter":{
     "field": "Variable",
     "oneOf":["DebtInterestToGDP", "GDP_bn", "debtInterest_bn"]
  }}
],
```

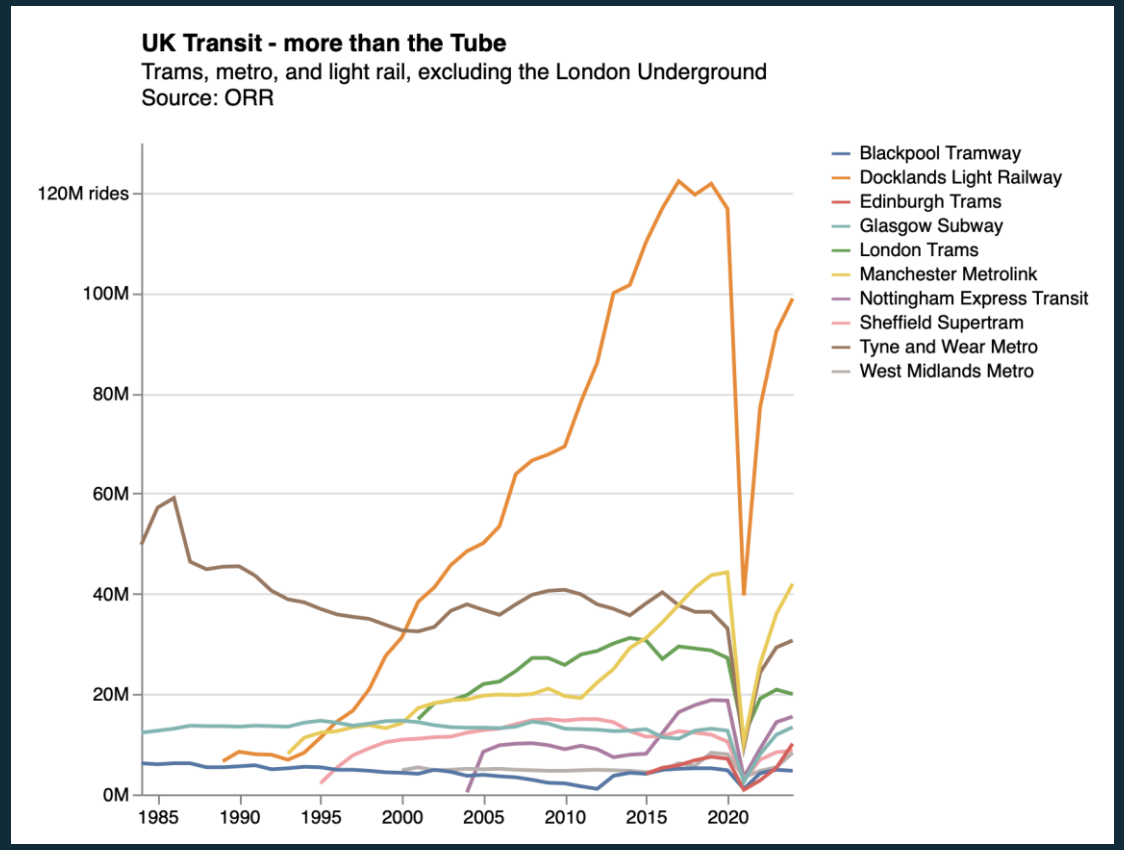
Example 3. Vega-Lite.

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 4. Vega-Lite.

Using conditionals with ternary operators

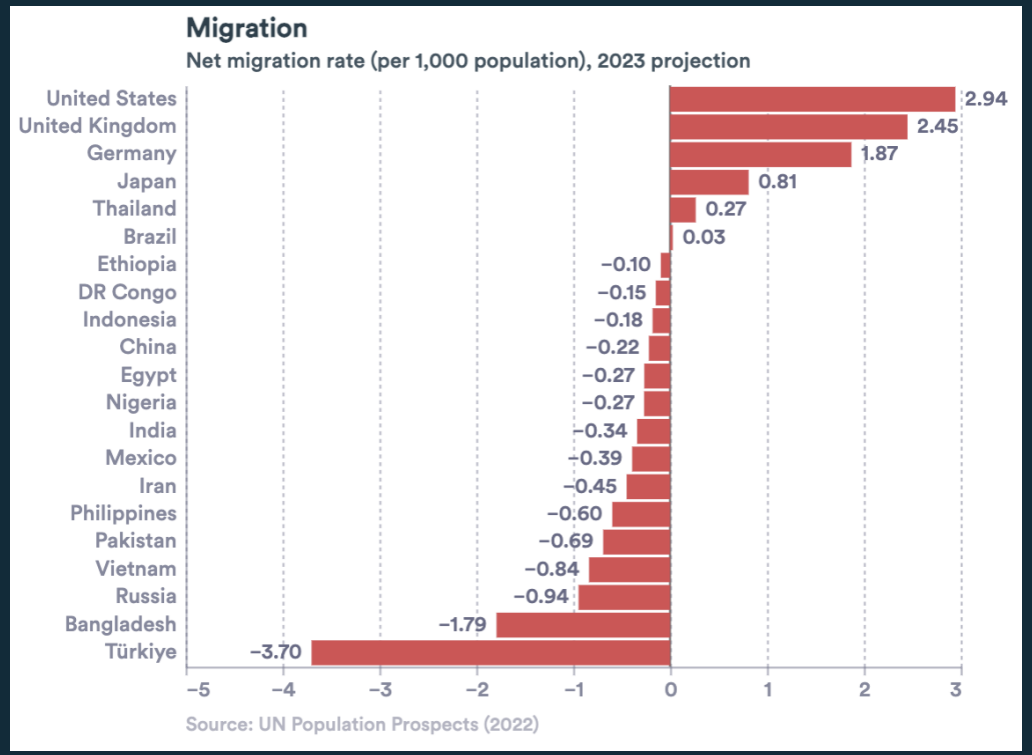
Ternary operators allow us to use normal if-else statements within Vega-Lite.

condition ? expression_if_true : expression_if_false

This is useful in transforms, label expressions and expressions for customising properties.

Here, we use an expression to test the sign of each ‘Net Migration Rate’ value. If positive, we position the text label to the right of the bar. If negative, position to left of the bar.

```
”mark”: {
        ”type”: ”text”,
        ”align”: {
                ”expr”: "datum['Net Migration Rate'] < 0 ? 'right' : 'left’”
        },
        ”dx”: {
                ”expr”: "datum[‘Net Migration Rate’] < 0 ? -5 : 5"
        }
},
```



Example 5. Vega-Lite.

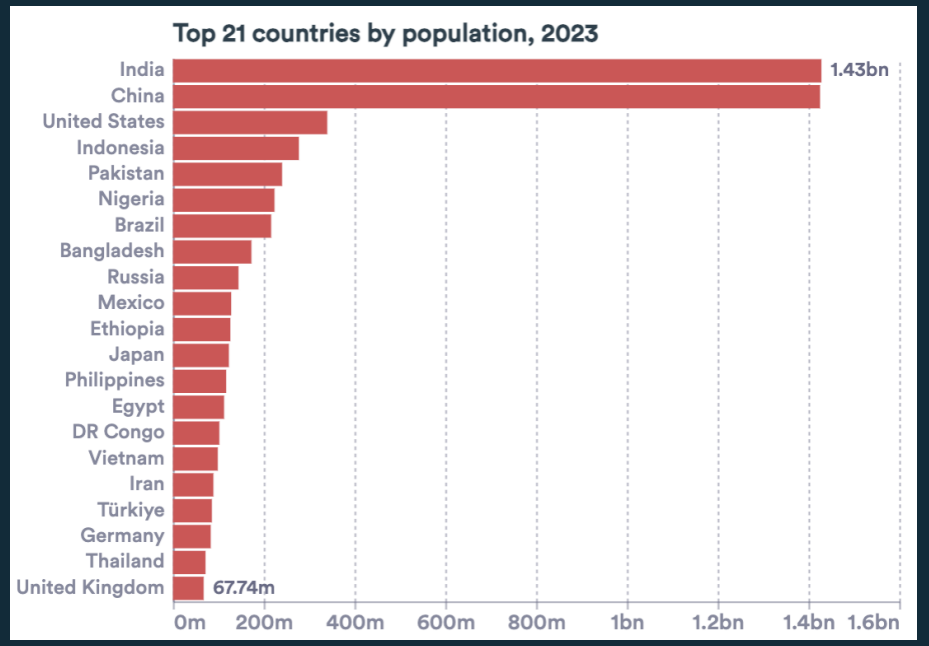
Using conditionals with ternary operators

Ternary operators allow us to use normal if-else statements within Vega-Lite.

In a label expression, the axis label text and values can be accessed through `datum.value` and `datum.label`.

Here, we test whether the label value is below 1 billion (1e9), if yes: we denote as millions with ‘m’, if no: we denote as billions with ‘bn’.

```
"field": "Population",
"type": "quantitative”,
"axis": {
        “labelExpr": "datum.value < 1e9 ? datum.value/1e6 + 'm' : datum.value/1e9 + 'bn’”
},
```

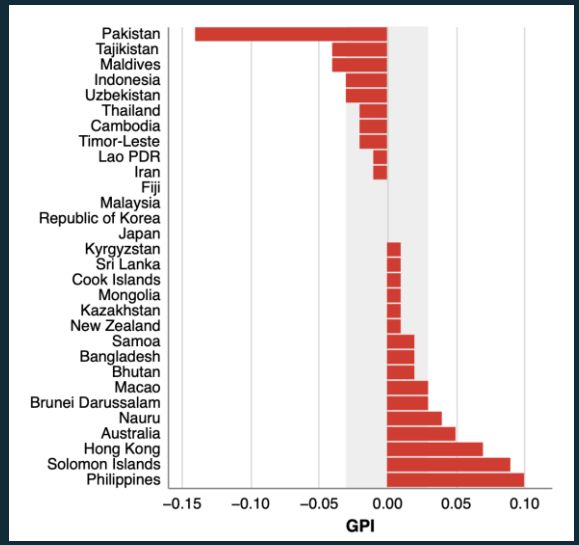


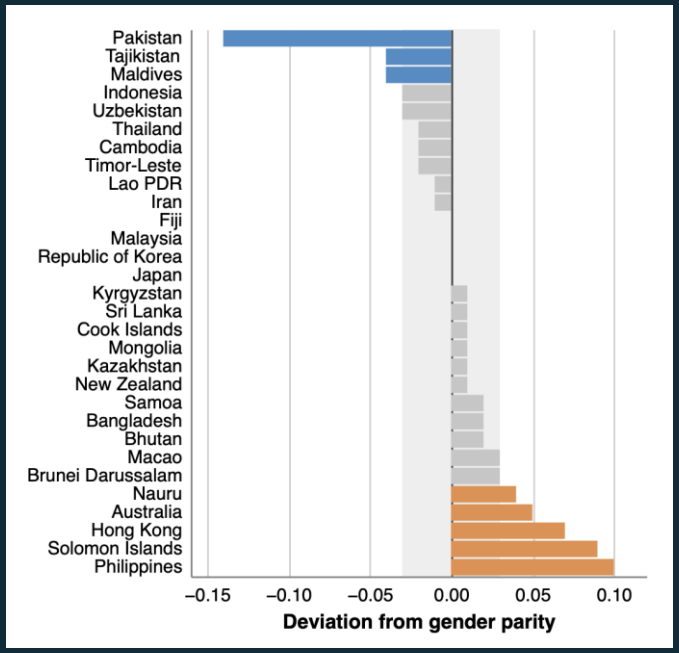
Example 6. In Vega-Lite.

Using conditionals in your charts

With a condition property, 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)}]
```



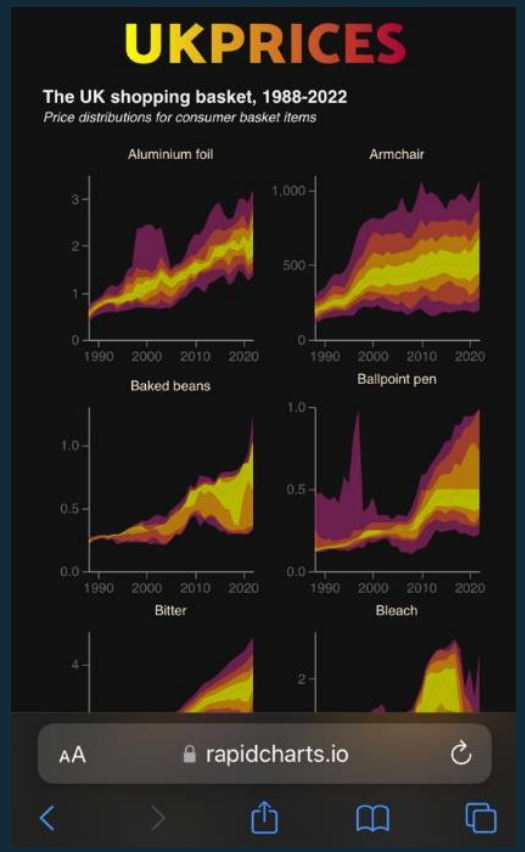


Example 7: screen sizes

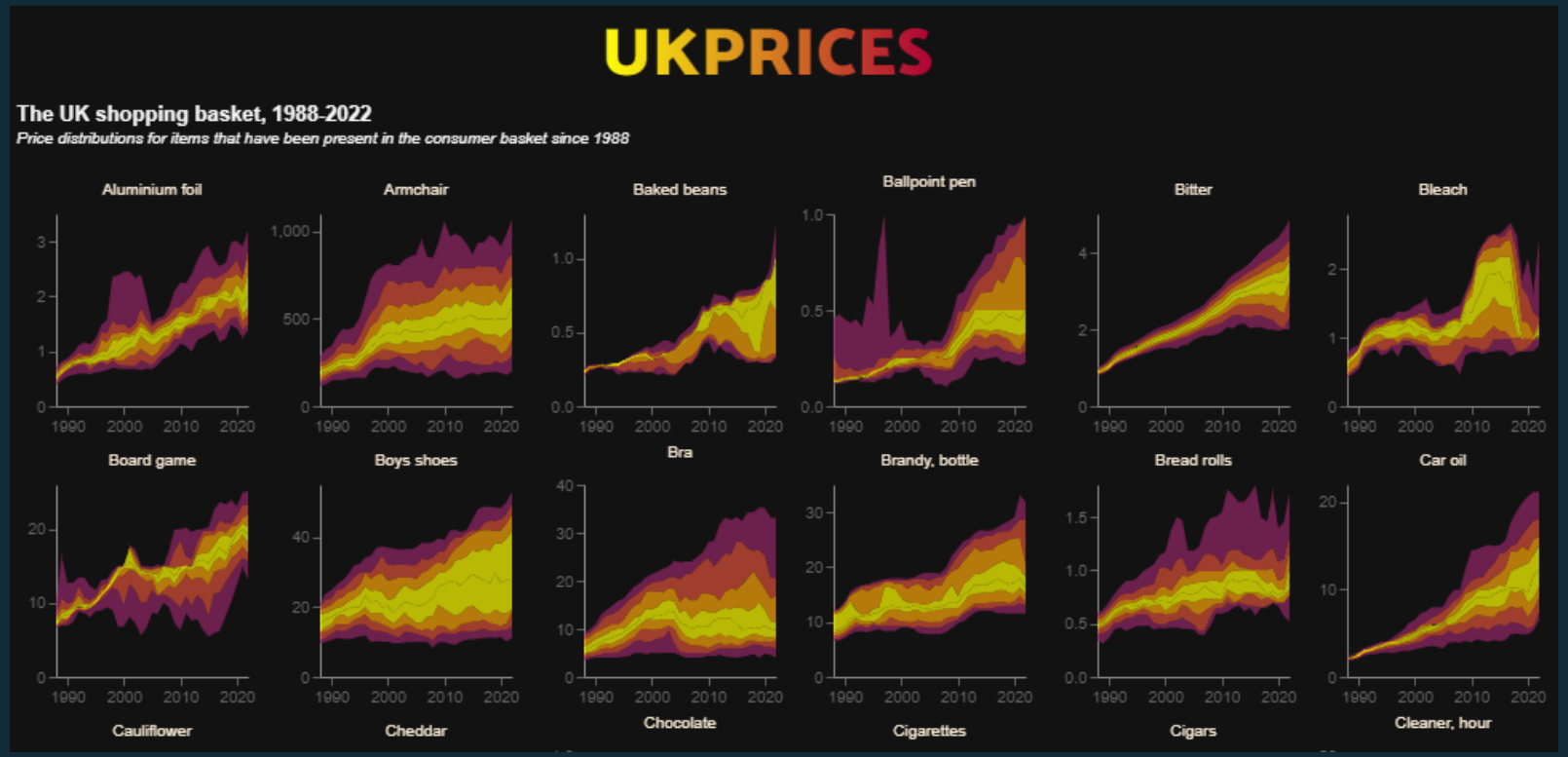
Excel | STATA | Python | JavaScript | CSS

https://rapidcharts.io/prices

Phone



Desktop



Example 7: 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 approprite 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 8 – in CSS.

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

```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:

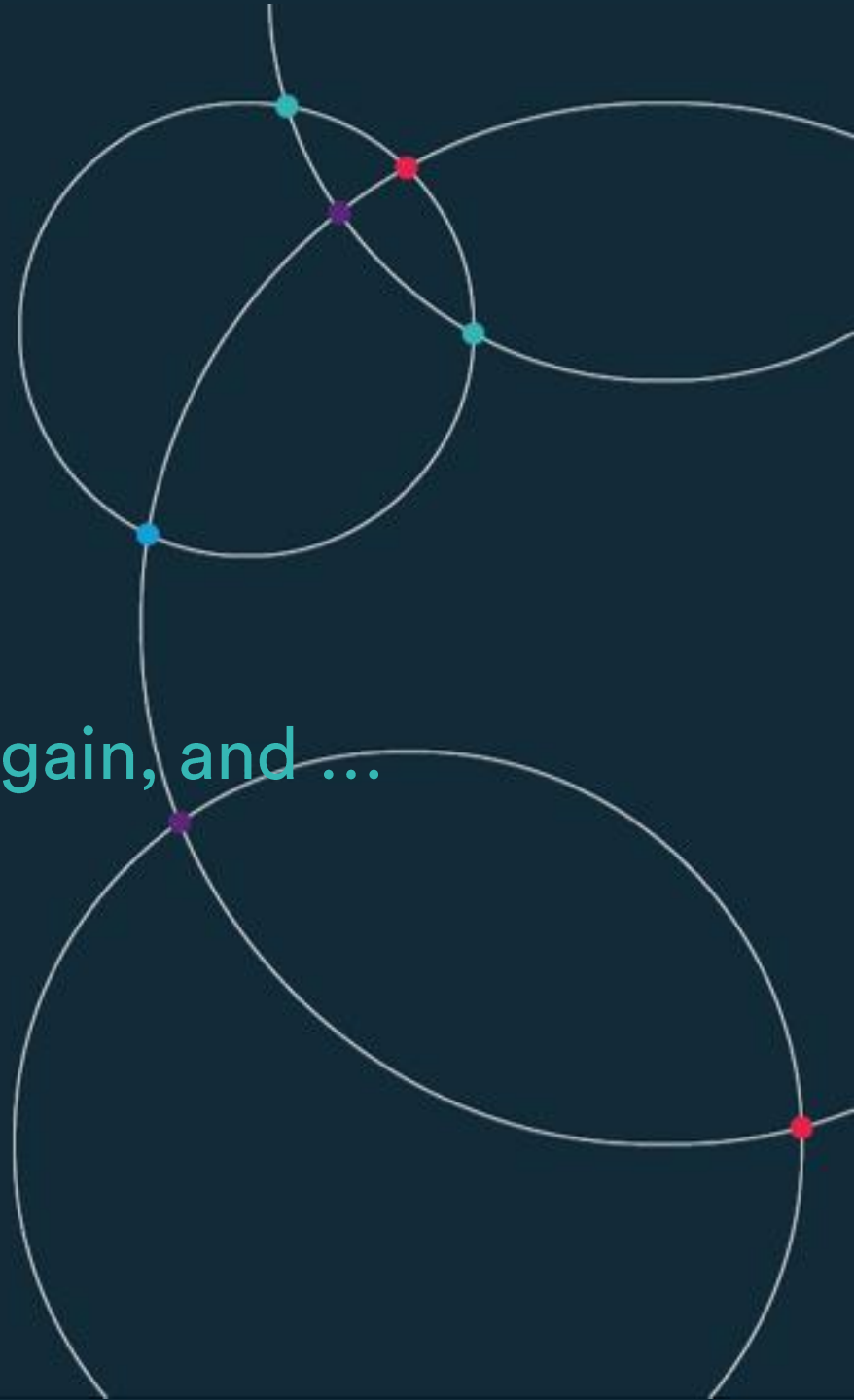
Smartphone?

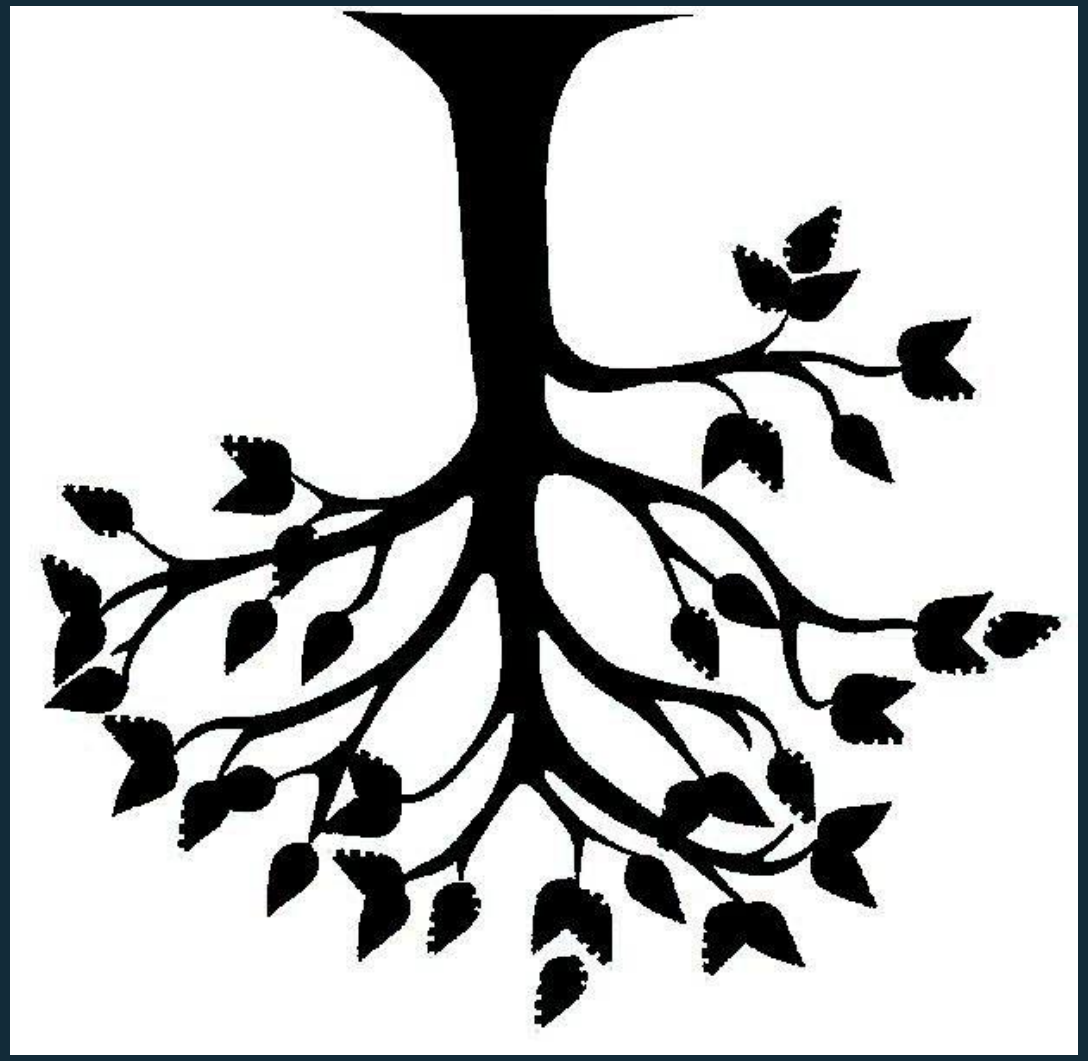
Laptop?

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 8 – in CSS.

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

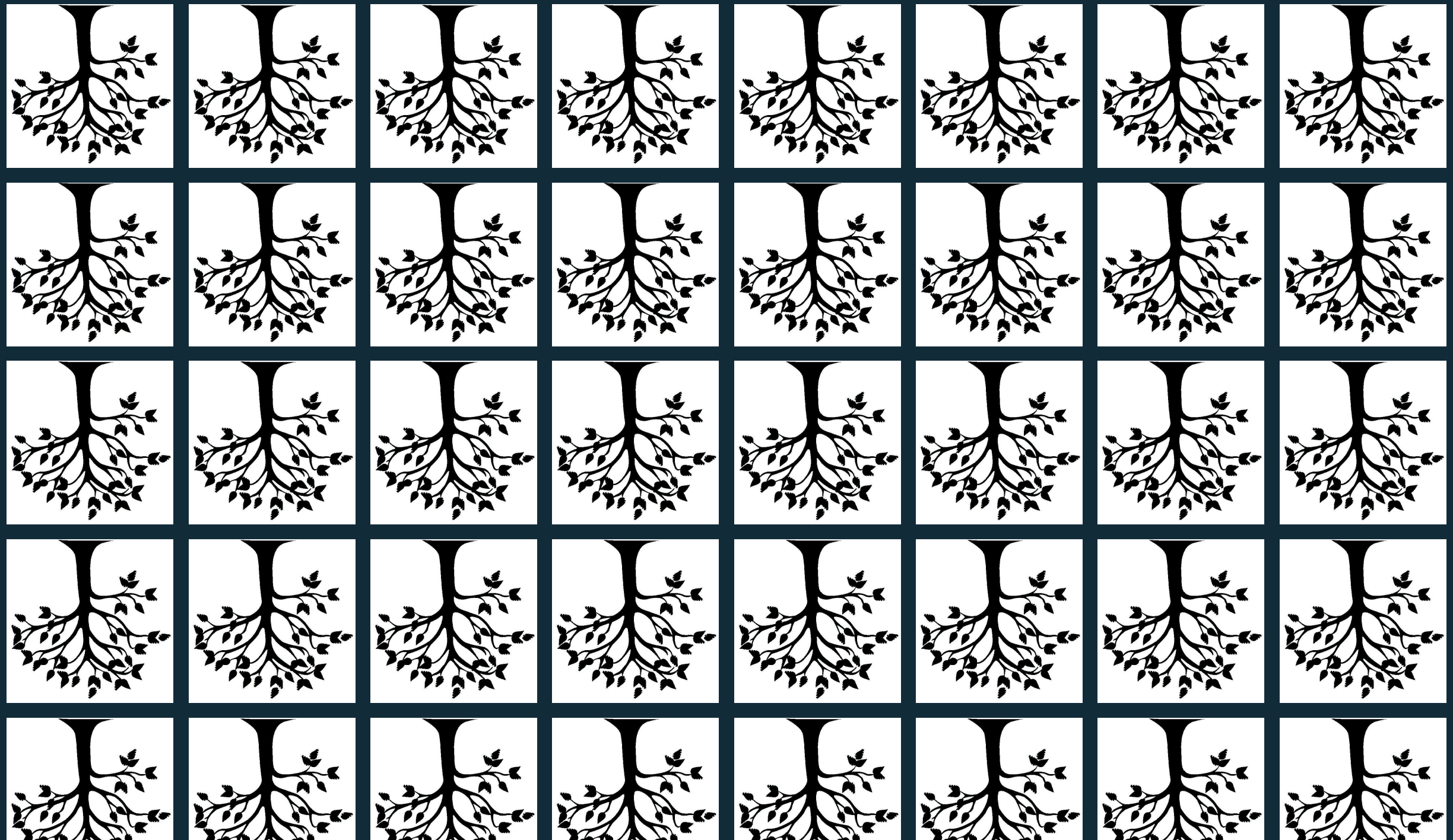
```css
/* Some CSS to alter the colour of my site */
body{
    background-color: white;
}

/* 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

```VB
For counter [ As datatype ] = start To end [ Step step ]
    [ statements ]
    [ Continue For ]
    [ statements ]
    [ Exit For ]
    [ statements ]
Next [ counter ]
```

If you must do this, the code snippet you are looking for is called “For-Next”. It is not like loops in Python, JS or Stata though.

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.
- Statament_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-pulsate2 {
  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-pulsate2 2s infinite;
}
```

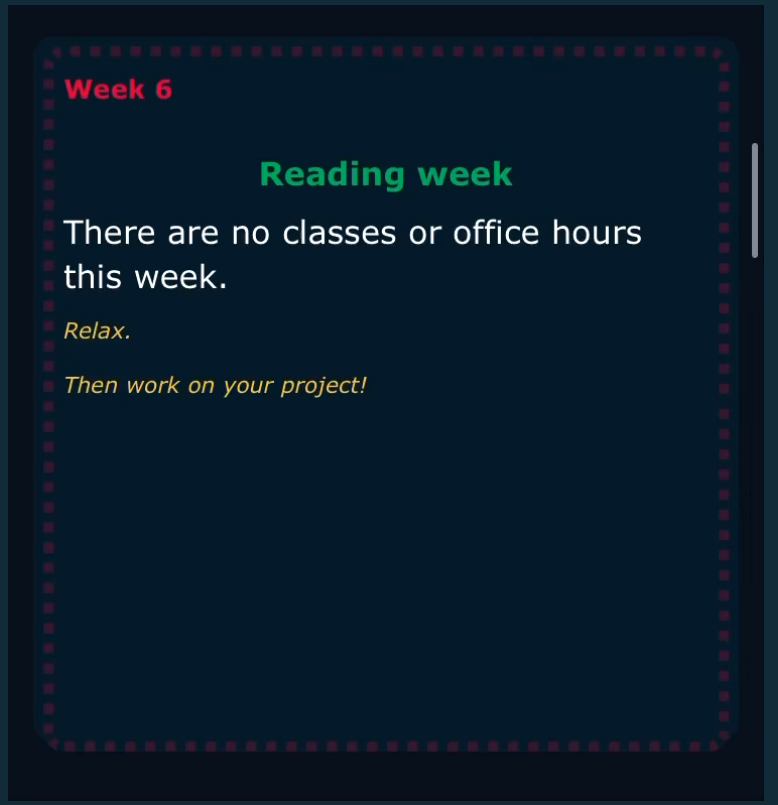
Loops.

Excel | STATA | Python | JavaScript | CSS

Again CSS is not built for looping

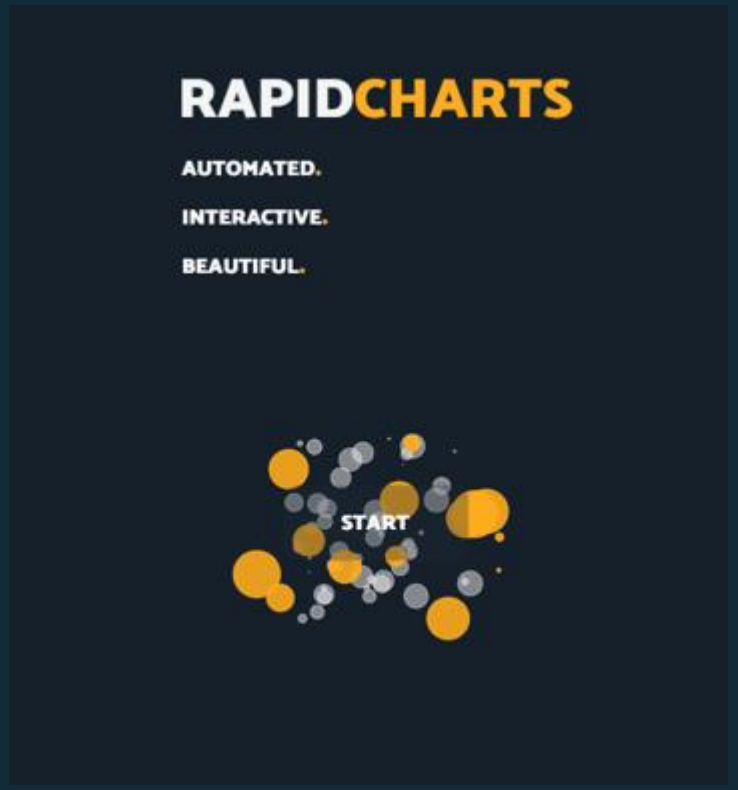
However, it can iterate over certain commands for a given time. One example

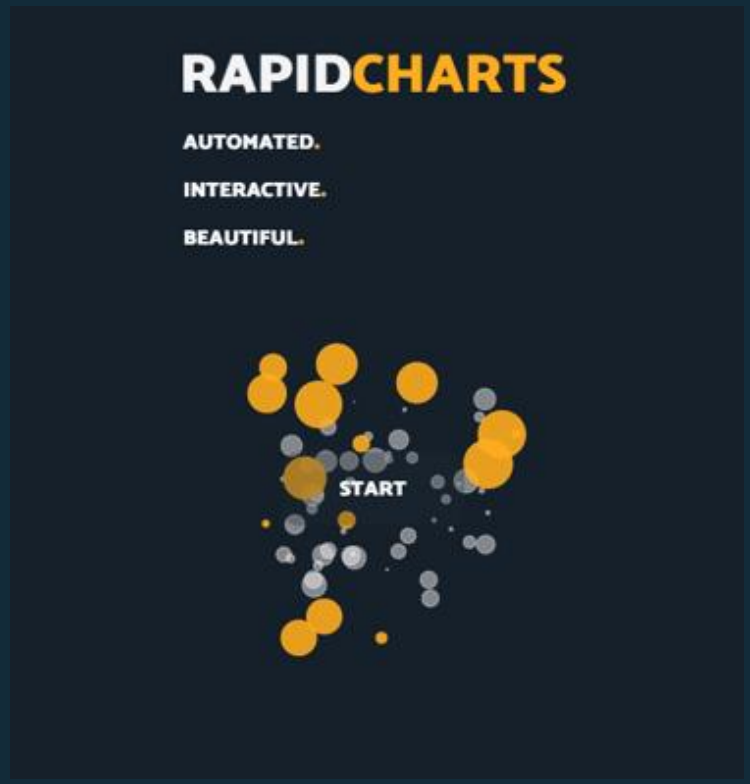
```
@keyframes border-pulsate2 {
  0%   { border-color: #22e68b}
  10%  { border-color: rgba(230, 34, 75, 0)}
  20%  { border-color: rgba(230, 34, 75)}
  30%  { border-color: rgba(230, 34, 75, 0)}
  40%  { border-color: rgba(0, 47, 167)}
  50%  { border-color: rgba(230, 34, 75, 0)}
  60%  { border-color: 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-pulsate2 2s infinite;
}
```



Using loops.

Exploding bubbles





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.

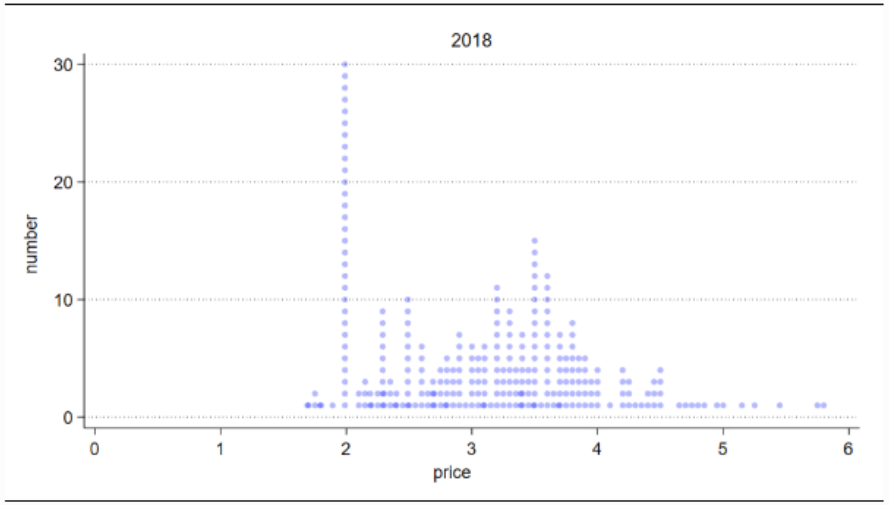
Using loops.

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, inlcuding £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.

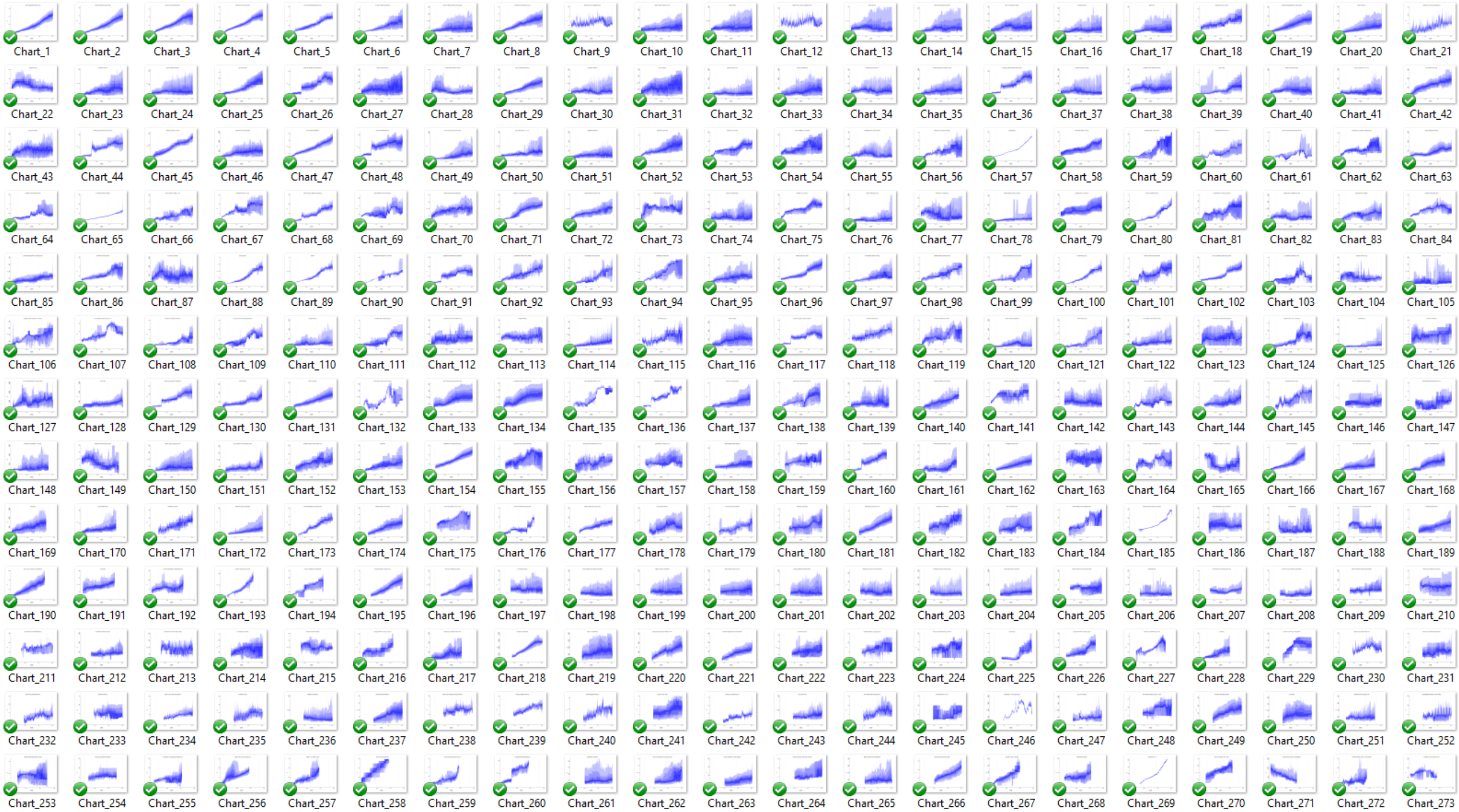


Chart_1 Chart_2 Chart_3 Chart_4 Chart_5 Chart_6 Chart_7 Chart_8 Chart_9 Chart_10 Chart_11 Chart_12 Chart_13 Chart_14 Chart_15 Chart_16 Chart_17 Chart_18 Chart_19 Chart_20 Chart_21
Chart_22 Chart_23 Chart_24 Chart_25 Chart_26 Chart_27 Chart_28 Chart_29 Chart_30 Chart_31 Chart_32 Chart_33 Chart_34 Chart_35 Chart_36 Chart_37 Chart_38 Chart_39 Chart_40 Chart_41 Chart_42
Chart_43 Chart_44 Chart_45 Chart_46 Chart_47 Chart_48 Chart_49 Chart_50 Chart_51 Chart_52 Chart_53 Chart_54 Chart_55 Chart_56 Chart_57 Chart_58 Chart_59 Chart_60 Chart_61 Chart_62 Chart_63
Chart_64 Chart_65 Chart_66 Chart_67 Chart_68 Chart_69 Chart_70 Chart_71 Chart_72 Chart_73 Chart_74 Chart_75 Chart_76 Chart_77 Chart_78 Chart_79 Chart_80 Chart_81 Chart_82 Chart_83 Chart_84
Chart_85 Chart_86 Chart_87 Chart_88 Chart_89 Chart_90 Chart_91 Chart_92 Chart_93 Chart_94 Chart_95 Chart_96 Chart_97 Chart_98 Chart_99 Chart_100 Chart_101 Chart_102 Chart_103 Chart_104 Chart_105
Chart_106 Chart_107 Chart_108 Chart_109 Chart_110 Chart_111 Chart_112 Chart_113 Chart_114 Chart_115 Chart_116 Chart_117 Chart_118 Chart_119 Chart_120 Chart_121 Chart_122 Chart_123 Chart_124 Chart_125 Chart_126
Chart_127 Chart_128 Chart_129 Chart_130 Chart_131 Chart_132 Chart_133 Chart_134 Chart_135 Chart_136 Chart_137 Chart_138 Chart_139 Chart_140 Chart_141 Chart_142 Chart_143 Chart_144 Chart_145 Chart_146 Chart_147
Chart_148 Chart_149 Chart_150 Chart_151 Chart_152 Chart_153 Chart_154 Chart_155 Chart_156 Chart_157 Chart_158 Chart_159 Chart_160 Chart_161 Chart_162 Chart_163 Chart_164 Chart_165 Chart_166 Chart_167 Chart_168
Chart_169 Chart_170 Chart_171 Chart_172 Chart_173 Chart_174 Chart_175 Chart_176 Chart_177 Chart_178 Chart_179 Chart_180 Chart_181 Chart_182 Chart_183 Chart_184 Chart_185 Chart_186 Chart_187 Chart_188 Chart_189
Chart_190 Chart_191 Chart_192 Chart_193 Chart_194 Chart_195 Chart_196 Chart_197 Chart_198 Chart_199 Chart_200 Chart_201 Chart_202 Chart_203 Chart_204 Chart_205 Chart_206 Chart_207 Chart_208 Chart_209 Chart_210
Chart_211 Chart_212 Chart_213 Chart_214 Chart_215 Chart_216 Chart_217 Chart_218 Chart_219 Chart_220 Chart_221 Chart_222 Chart_223 Chart_224 Chart_225 Chart_226 Chart_227 Chart_228 Chart_229 Chart_230 Chart_231
Chart_232 Chart_233 Chart_234 Chart_235 Chart_236 Chart_237 Chart_238 Chart_239 Chart_240 Chart_241 Chart_242 Chart_243 Chart_244 Chart_245 Chart_246 Chart_247 Chart_248 Chart_249 Chart_250 Chart_251 Chart_252
Chart_253 Chart_254 Chart_255 Chart_256 Chart_257 Chart_258 Chart_259 Chart_260 Chart_261 Chart_262 Chart_263 Chart_264 Chart_265 Chart_266 Chart_267 Chart_268 Chart_269 Chart_270 Chart_271 Chart_272 Chart_273

Break.

Then: portfolios, projects, practical.

Break.

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

Resources.

Add your site to the google sheet.

Resource 1: my site:
www.richarddavies.io/data-science

Resource 2: chart library.
www.richarddavies.io/library

Resource 3: course Google sheet.
Google sheet. Link.

Resource 4: course DropBox.
DropBox. Link.

Portfolio.

This week's task

CC5. Scraper. Implement a data scraper of your own.

This week we have discussed the simple ways of scraping data from websites. Using a Google Colab python notebook, scrape a website through one of the presented methods in class (pandas read_html or beautifulsoup). This can be any source, any data.

Then clean and normalise the data and export into TIDY (long form) format.

1. A link to the Google Collab python notebook where you conducted your data analysis (make sure you make it publicly accessible!)

2. A chart built with the data from the previous point. It is up to you what how do you reference and include the data in the Vega-lite chart specification.

3. Comment (not more than 25 words) why did you choose this site.

Projects.

Let us know what you are working on!

5.3 Building dashboards.

Using loops to analyse multiple data series
Looping over a list of countries to create create dynamic API-linked charts.