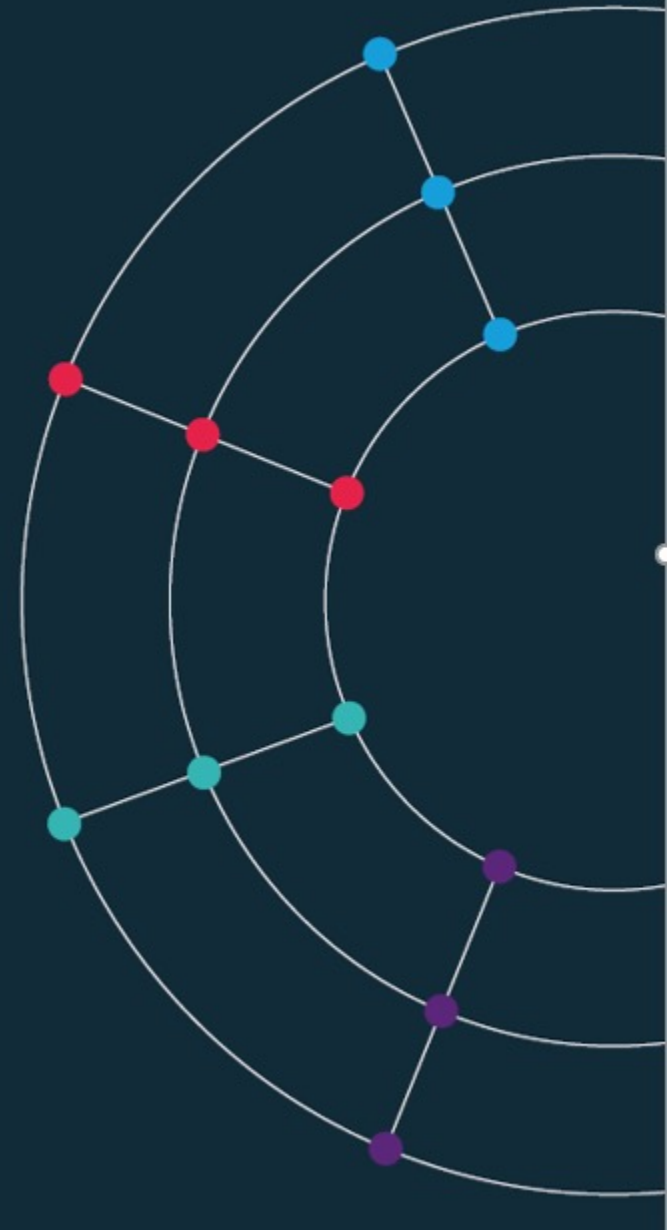




Data Masterclass.

Professor Richard Davies + ECO Data Unit Team

Bank of England
Thursday 14 March



1.0 Motivation.

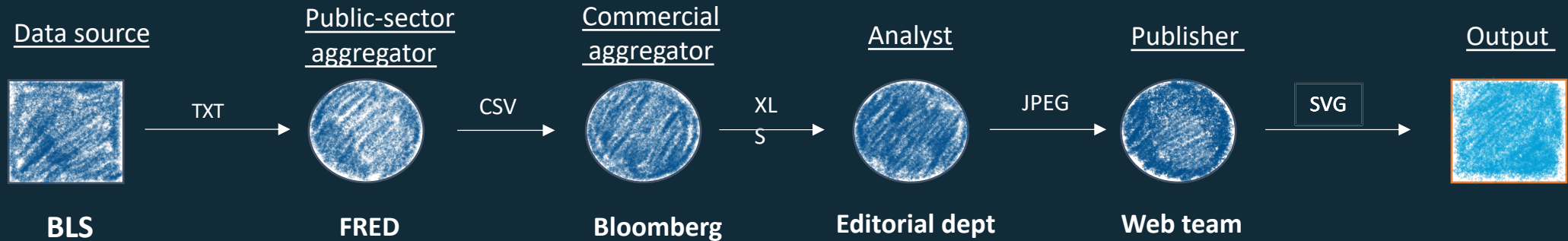
Problems, solutions, and plan for today



Where we are.

The state of play

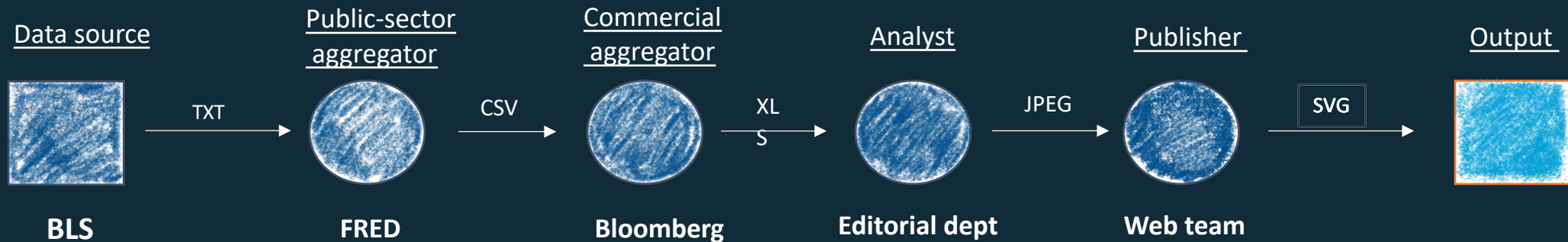
- Analysts will tend to have five steps or more between their raw data and their output.
- Example: a journalist, consultant or academic making a chart of GDP in America



Where we are.

The state of play

- Analysts will tend to have five steps or more between their raw data and their output.
- Example: a journalist, consultant or academic making a chart of GDP in America



Note how this system works:

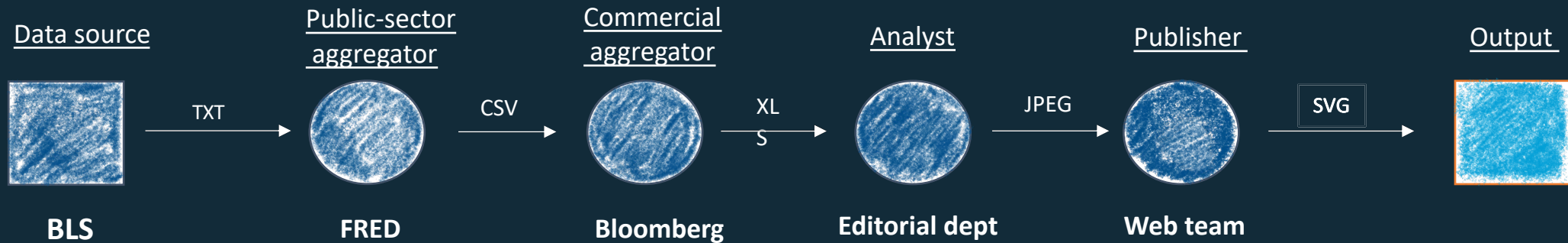
1. **Repetition.** Many players, each adding delay or cost. Slow and expensive.
2. **Errors.** Each link can, and does, break. Each one adds to the chance of a human error.
3. **Compatibility.** Many different file types. Delay and compatibility problems
4. **Storage.** Huge data storage requirement, with files stored in each silo along the way. Resource and environmental costs

This system is slow, costly and inaccurate.

Where we are: **slow, costly, error prone.**

The problems we will solve

- Analysts will tend to have five steps or more between their raw data and their output.
- Example: a journalist, consultant or academic making a chart of GDP in America



Note how this system works:

1. **Repetition.** Many players, each adding delay or cost. Slow and expensive.
2. **Errors.** Each link can, and does, break. Each one adds to the chance of a human error.
3. **Compatibility.** Many different file types. Delay and compatibility problems
4. **Storage.** Huge data storage requirement, with files stored in each silo along the way. Resource and environmental costs

This system is slow, costly and inaccurate.

Where we are going: fast, cheap, accurate

The solutions we will offer

- The modern approach used our site uses is different.
- We are building a single secure channel that links an analyst's output right back to the raw data.



Note the differences with the new system:

1. **Automated**. Your chart is always up to date. When the raw data updates, it flows through the channel instantly and automatically.
2. **Efficient**. The number of players collapses, as do associated costs. There are no data silos – the data shown on your site **is** the raw data, not a copy of them.
3. **Verifiable**. A one-to-one link back to the original data provider means fact checks can be swift.

The result is analysis that is faster, cheaper and more accurate.

Why it matters...

...and why Britain's economists should lead the way

The problems above matter in many fields. They are acute in economics and public policy:

Regular updates. Economic data is high-frequency. This means that the 'repeated analysis' problem is particularly strong.

Descriptive analysis. Economists and analysts are aiming to clarify. We need our work to be replicable (someone else can run the code and get the same result) and verifiable.

Normative and policy analysis. Economic data is hard-wired into policy decisions. Getting it right affects people's lives.

What we are doing.

Data at the Economics Observatory

To respond to these issues, we have developed the **Economics Observatory Data Hub**. This contains three connected tools:

Explore. A new API. This will unify and simplify existing data sources into one stream. The ECO API offers simple URLs that put the user's needs first, allowing them to access a wide range of metrics from multiple international sources

Build. With improved data access comes better analytics. With our new Chart Builder, users take the latest data directly from source, and display this live in a webpage. No coding skills are needed, but modern tools are being used in the background: Excel and JPEG are out, JavaScript Object Notation (JSON), D3.js and Vega-Lite are in.

Share. Robust analysis deserves to be shared. Users can share their visualisations on traditional platforms like Twitter and Instagram, or can post their work to our new timeline for academics, journalists, policy-makers and students to post and discuss their charts.

What we are doing.

Data at the Economics Observatory

Explore

Data Explorer

Select country Select variable

Your ECO API custom link will appear here

Data Explorer

Germany

Your ECO API custom link will appear here

✓ Select variable

- Exports
- GDP
- Imports
- Inflation**
- Labour participation rate
- Population
- Unemployment
- Debt

Data Explorer

✓ Select country

- Australia
- Bangladesh
- Brazil
- Canada
- China
- DR Congo
- Germany**

Select variable

Your ECO API custom link will appear here

```
{
  "author": "Economics Observatory",
  "source": "Destatis",
  "url": "https://raw.githubusercontent.com/EconomicsObservatory/api/main/cached_data/deu/infl.json",
  "data": [
    {
      "date": "1960-01-01",
      "value": 1.53661234295201
    },
    {
      "date": "1961-01-01",
      "value": 2.29369500700881
    },
    {
      "date": "1962-01-01",
      "value": 2.84327019799063
    },
    {
      "date": "1963-01-01",
      "value": 2.96695977587712
    },
    {
      "date": "1964-01-01",
      "value": 3.08064811111111
    }
  ]
}
```

What we are doing.

Data at the Economics Observatory

Create

Data

Country

Variable

Variables

date

Type

Label X

value

Type

Label Y

Appearance

Inflation

Size

Colour ☒

CPI, % |Source: Destatis through ECO API

Size

Colour ☒

Type

2

Height

Colour ☒

Width

Page ☒

Canvas

☒

Chart

Data

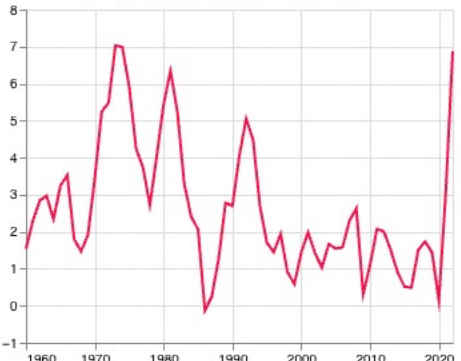
Code

SHARE

Your visualisation:

Inflation

CPI, % |Source: Destatis through ECO API

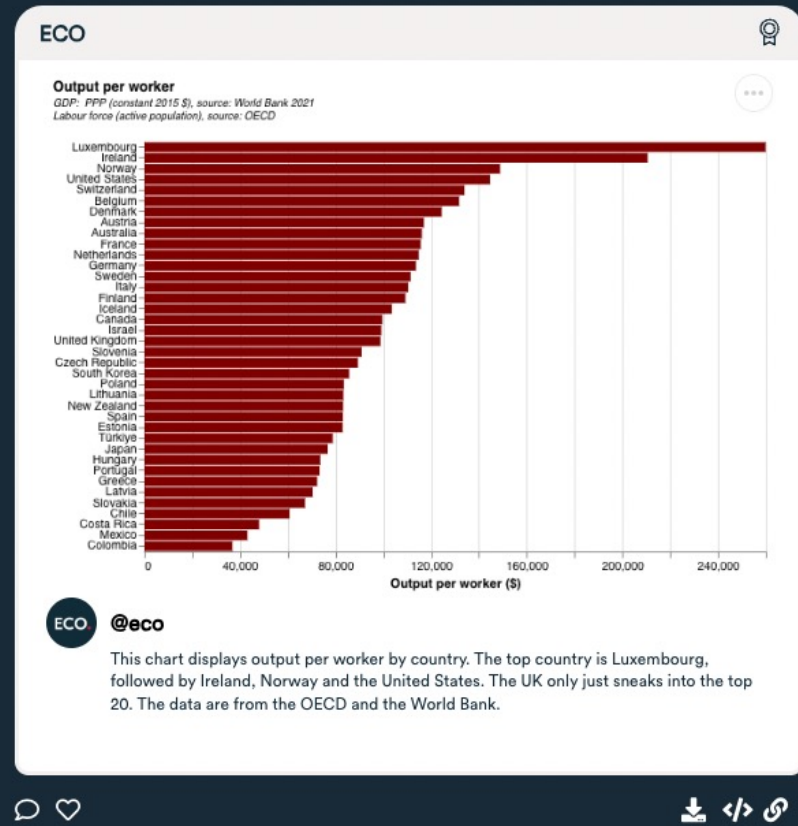
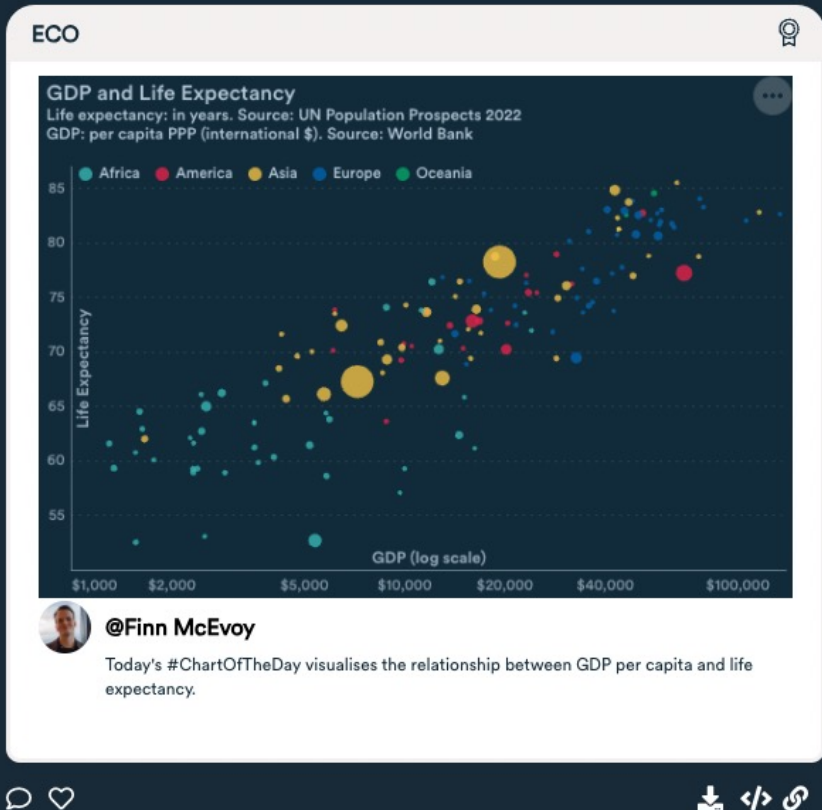


Description

What we are doing.

Data at the Economics Observatory

Share



Masterclass outline.

Teaching team + more on ECO

Richard Davies | Director

Charlie Meyrick | Research Fellow

Dénes Csala | Data Editor

Finn McEvoy | Data Scientist

Website: www.economicsobservatory.com

Newsletter: www.economicsobservatory.com/join-us

Data Hub: www.economicsobservatory.com/data-hub

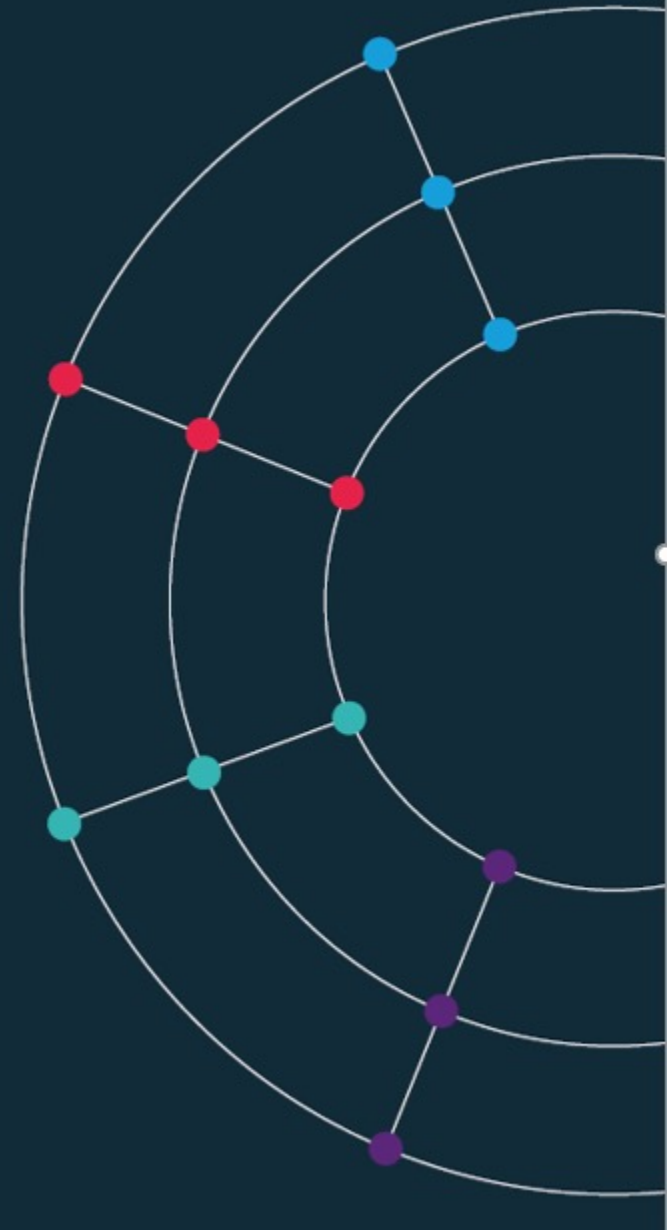
Masterclass outline.

What we are covering today

| <i>Time</i> | <i>Section</i> | <i>Details</i> |
|---------------|----------------|---|
| 09.00 - 09.30 | | Welcome coffee. Checking computer set ups. |
| 09.30 - 10.30 | 1 | Charts as data – introducing Vega-Lite |
| 10.30 - 11.00 | <i>Break</i> | <i>Coffee/tea (team available to fix bugs)</i> |
| 11:00 - 12.30 | 2 | My first website – GitHub, HMTL, CSS and JavaScript |
| 12.30 - 13.15 | <i>Lunch</i> | <i>Lunch (team available to fix bugs)</i> |
| 13.15 - 14.15 | 3 | Programming – APIs, if statements, and loops. |
| 14.15 - 15.15 | 4 | Advanced visualisations – beyond two dimensions |
| 15:15 - 15:30 | <i>Break</i> | <i>Coffee/tea (team available to fix bugs)</i> |
| 15.30 - 16.30 | 5 | Scraping – fetching data with Python |

1.1 Charts as data.

How a new datatype will save you time and errors



JSON data.

- **JavaScript Object Notation.** An important data type. **Why use it?**
 - It is the way computers share data.
 - Most importantly it is what many APIs will deliver – your data will arrive to you in this way.
 - It looks complicated at first, but is easy to convert into access, change and chart.
- Formatted as **key : value** pairs:

```
'{"name": "David Beckham", "yearBorn": 1975, "team": "Manchester United"}'
```
- **Keys** are strings, always in double quotes.
- **Values** can be many things: string, number, arrays, objects.
- Useful tools:
 - www.jsonlint.com. Test and format JSON.
 - **JSON formatter**. Chrome extension.

Raw.

```
{
  "query": {
    "apikey": "b8bb8050-f3d3-11eb-808b-334144de1112",
    "season_id": 352,
    "date_from": "2020-09-19",
    "data": [
      {
        "match_id": 137294,
        "status_code": 3,
        "status": "finished",
        "match_start": "2020-09-19 11:30:00",
        "match_start_iso": "2020-09-19T11:30:00+00:00",
        "minute": null,
        "league_id": 237,
        "season_id": 352,
        "stage": {
          "stage_id": 1,
          "name": "Regular Season"
        },
        "group": {
          "group_id": 103,
          "group_name": "Premier League"
        },
        "round": {
          "round_id": 17515,
          "name": "2",
          "is_current": null,
          "referee_id": 196,
          "home_team": {
            "team_id": 2516,
            "name": "Everton FC",
            "short_code": "EVE",
            "common_name": "",
            "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/10.png",
            "country": {
              "country_id": 42,
              "name": "England",
              "country_code": "en",
              "continent": "Europe"
            },
            "away_team": {
              "team_id": 2544,
              "name": "West Bromwich Albion",
              "short_code": "WBA",
              "common_name": "",
              "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/272.png",
              "country": {
                "country_id": 42,
                "name": "England",
                "country_code": "en",
                "continent": "Europe"
              },
              "stats": {
                "home_score": 5,
                "away_score": 2,
                "ht_score": "2-1",
                "ft_score": "5-2",
                "et_score": null,
                "ps_score": null
              },
              "venue": {
                "venue_id": 1208,
                "name": "Goodison Park",
                "capacity": 39571,
                "city": "Liverpool",
                "country_id": 42
              },
              "match_id": 137340,
              "status_code": 3,
              "status": "finished",
              "match_start": "2020-09-19 14:00:00",
              "match_start_iso": "2020-09-19T14:00:00+00:00",
              "minute": null,
              "league_id": 237,
              "season_id": 352,
              "stage": {
                "stage_id": 1,
                "name": "Regular Season"
              },
              "group": {
                "group_id": 103,
                "group_name": "Premier League"
              },
              "round": {
                "round_id": 17515,
                "name": "2",
                "is_current": null,
                "referee_id": 145,
                "home_team": {
                  "team_id": 2546,
                  "name": "Leeds United",
                  "short_code": "LU",
                  "common_name": "",
                  "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/274.png",
                  "country": {
                    "country_id": 42,
                    "name": "England",
                    "country_code": "en",
                    "continent": "Europe"
                  },
                  "away_team": {
                    "team_id": 12429,
                    "name": "Fulham FC",
                    "short_code": "FUL",
                    "common_name": "Fulham (R)",
                    "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/6214.png",
                    "country": {
                      "country_id": 42,
                      "name": "England",
                      "country_code": "en",
                      "continent": "Europe"
                    },
                    "stats": {
                      "home_score": 4,
                      "away_score": 3,
                      "ht_score": "2-1",
                      "ft_score": "4-3",
                      "et_score": null,
                      "ps_score": null
                    },
                    "venue": {
                      "venue_id": 1225,
                      "name": "Eiland Road",
                      "capacity": 39460,
                      "city": "Leeds",
                      "country_id": 42
                    },
                    "match_id": 137387,
                    "status_code": 3,
                    "status": "finished",
                    "match_start": "2020-09-19 16:30:00",
                    "match_start_iso": "2020-09-19T16:30:00+00:00",
                    "minute": null,
                    "league_id": 237,
                    "season_id": 352,
                    "stage": {
                      "stage_id": 1,
                      "name": "Regular Season"
                    },
                    "group": {
                      "group_id": 103,
                      "group_name": "Premier League"
                    },
                    "round": {
                      "round_id": 17515,
                      "name": "2",
                      "is_current": null,
                      "referee_id": 195,
                      "home_team": {
                        "team_id": 2523,
                        "name": "Manchester United",
                        "short_code": "MU",
                        "common_name": "",
                        "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/19.png",
                        "country": {
                          "country_id": 42,
                          "name": "England",
                          "country_code": "en",
                          "continent": "Europe"
                        },
                        "away_team": {
                          "team_id": 2515,
                          "name": "Crystal Palace",
                          "short_code": "PAL",
                          "common_name": "",
                          "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/9.png",
                          "country": {
                            "country_id": 42,
                            "name": "England",
                            "country_code": "en",
                            "continent": "Europe"
                          },
                          "stats": {
                            "home_score": 1,
                            "away_score": 3,
                            "ht_score": "0-1",
                            "ft_score": "1-3",
                            "et_score": null,
                            "ps_score": null
                          },
                          "venue": {
                            "venue_id": 1204,
                            "name": "Old Trafford",
                            "capacity": 75635,
                            "city": "Manchester",
                            "country_id": 42
                          },
                          "match_id": 137428,
                          "status_code": 3,
                          "status": "finished",
                          "match_start": "2020-09-19 19:00:00",
                          "match_start_iso": "2020-09-19T19:00:00+00:00",
                          "minute": null,
                          "league_id": 237,
                          "season_id": 352,
                          "stage": {
                            "stage_id": 1,
                            "name": "Regular Season"
                          },
                          "group": {
                            "group_id": 103,
                            "group_name": "Premier League"
                          },
                          "round": {

```

API returns. The results of request to Sport Data API.

Parsed.

```
},
  "away_team": {
    "team_id": 2544,
    "name": "West Bromwich Albion",
    "short_code": "WBA",
    "common_name": "",
    "logo": "https://cdn.sportdataapi.com/images/soccer/teams/100/272.png",
    "country": {
      "country_id": 42,
      "name": "England",
      "country_code": "en",
      "continent": "Europe"
    }
  },
  "stats": {
    "home_score": 5,
    "away_score": 2,
    "ht_score": "2-1",
    "ft_score": "5-2",
    "et_score": null,
    "ps_score": null
  },
  "venue": {
    "venue_id": 1208,
    "name": "Goodison Park",
    "capacity": 39571,
    "city": "Liverpool",
    "country_id": 42
  }
}
```

Why data is great.

Some important aspects of the data revolution

Data is great for a host of reasons:

- Transparency
- Verification
- Replication
- Sharing
- Comparing

Why charts are terrible.

The problem we face

Charts can be terrible for many reasons:

- Opaque
- Repetitious
- Inflexible (pixel problems when changing sizes)

During the break: examples of Chart Junk, and Graph Crimes

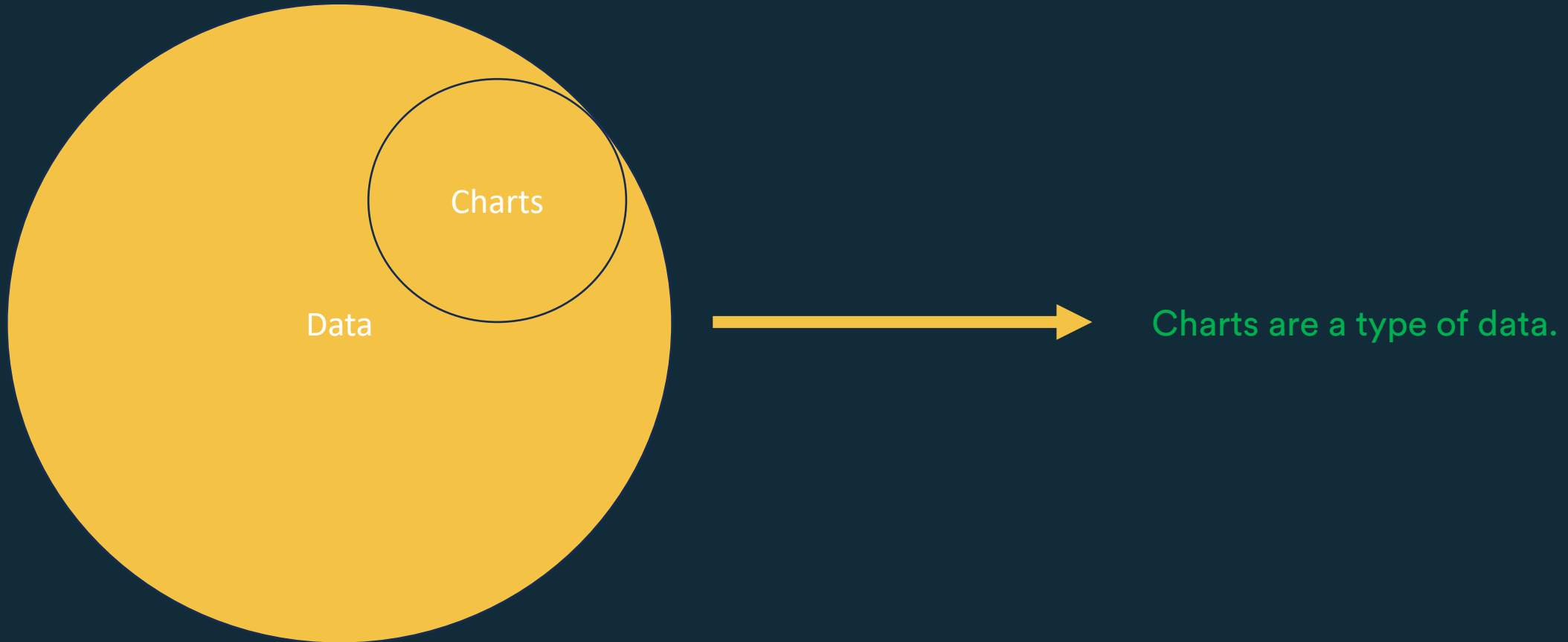
The state of play.

Analysis in 2024



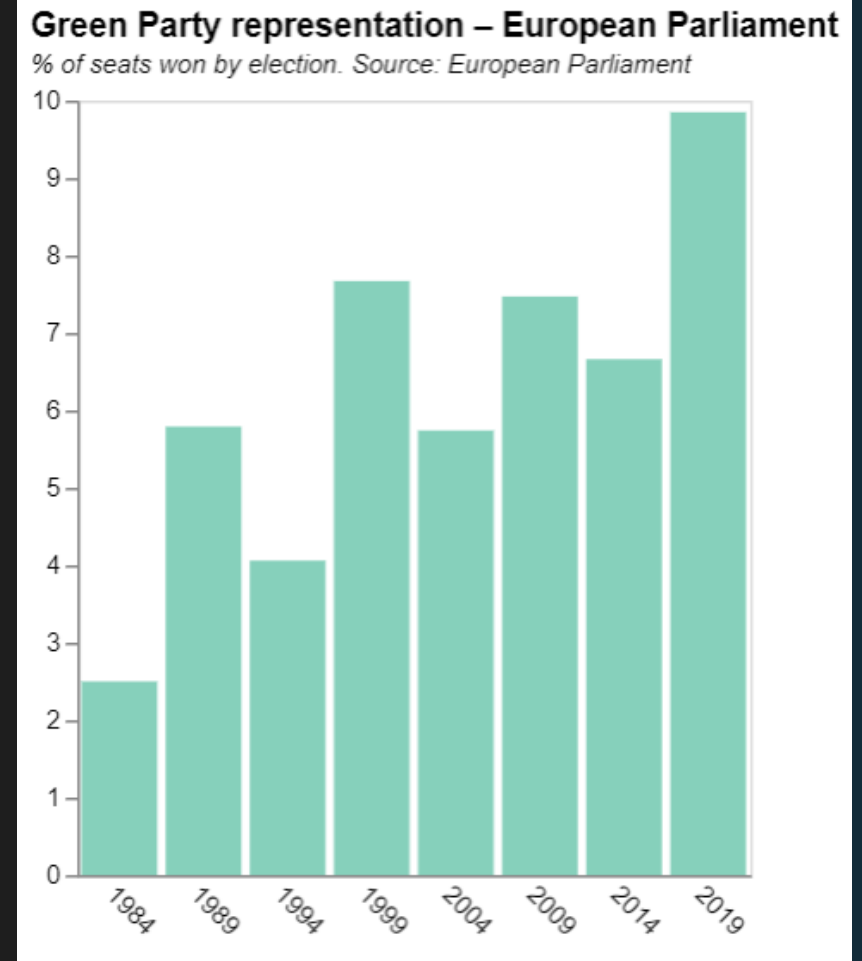
The big idea for today.

The core philosophy of our course



Charts as JSON data.

```
{ "$schema": "https://vega.github.io/schema/vega-lite/v5.json",  
  
  "title": {"text": "Green Party representation - European Parliament"},  
  
  "data": {"url": "https://raw.githubusercontent.com/RDeconomist/RDeconomist.github.io/main/data/chartENV19.csv"},  
  
  "height": 300,  
  "width": 260,  
  
  "mark": {  
    "type": "bar",  
    "color": "#86d0bb"},  
  
  "encoding": {  
    "x": {  
      "field": "Time",  
      "type": "nominal",  
      "axis": {  
        "title": null,  
        "grid": false,  
        "ticks": false,  
        "labelAngle": 45}},  
    "y": {  
      "field": "Value",  
      "type": "quantitative",  
      "title": "",  
      "axis": {"grid": false}}  }  
}
```



1.2 Delivering data.

Three ways to inject numbers into your chart



Delivering data.

Where does the data in your chart come from?

A vital input to your chart is the data that is plotted.

We will meet three:

1. Embedding data in your chart code
2. Delivering data from GitHub
3. Delivering data from an API


```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "title": {"text": "GDP Per Capita"},
  "width": 400,
  "height": 300,
  "data": {
    "values": [
      {"Country": "China", "GDP per capita": 21482},
      {"Country": "DR Congo.", "GDP per capita": 1337},
      {"Country": "UK", "GDP per capita": 54929},
      {"Country": "India", "GDP per capita": 8400},
      {"Country": "Nigeria", "GDP per capita": 5862},
      {"Country": "USA", "GDP per capita": 76329}]
    },
  "mark": {
    "type": "bar",
    "color": "red"},
  "encoding": {
    "x": {"field": "Country"},
    "y": {
      "field": "GDP per capita",
      "type": "quantitative"}}
}
```

Delivering data 1.

Embedding the data in your chart

Delivering data 1.

Embedding the data in your chart

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "title": {"text": "GDP Per Capita"},
  "width": 400,
  "height": 300,
  "data": {
    "values": [
      {"Country": "China", "GDP per capita": 21482},
      {"Country": "DR Congo.", "GDP per capita": 1337},
      {"Country": "UK", "GDP per capita": 54929},
      {"Country": "India", "GDP per capita": 8400},
      {"Country": "Nigeria", "GDP per capita": 5862},
      {"Country": "USA", "GDP per capita": 76329}]
    },
  "mark": {
    "type": "bar",
    "color": "red"},
  "encoding": {
    "x": {"field": "Country"},
    "y": {
      "field": "GDP per capita",
      "type": "quantitative"}}
}
```

← The dataset can be seen here.

```
{  
"$schema": "https://vega.github.io/schema/vega-lite/v5.json",
```

```
"title": {"text": "GDP Per Capita"},
```

```
"width": 400,
```

```
"height": 300,
```

```
"data": {
```

```
  "values": [  
    {"Country": "China", "GDP per capita": 21482},  
    {"Country": "DR Congo.", "GDP per capita": 1337},  
    {"Country": "UK", "GDP per capita": 54929},  
    {"Country": "India", "GDP per capita": 8400},  
    {"Country": "Nigeria", "GDP per capita": 5862},  
    {"Country": "USA", "GDP per capita": 76329}]  
  },
```

```
"mark": {
```

```
  "type": "bar",
```

```
  "color": "red"},
```

```
"encoding": {
```

```
  "x": {"field": "Country"},
```

```
  "y": {
```

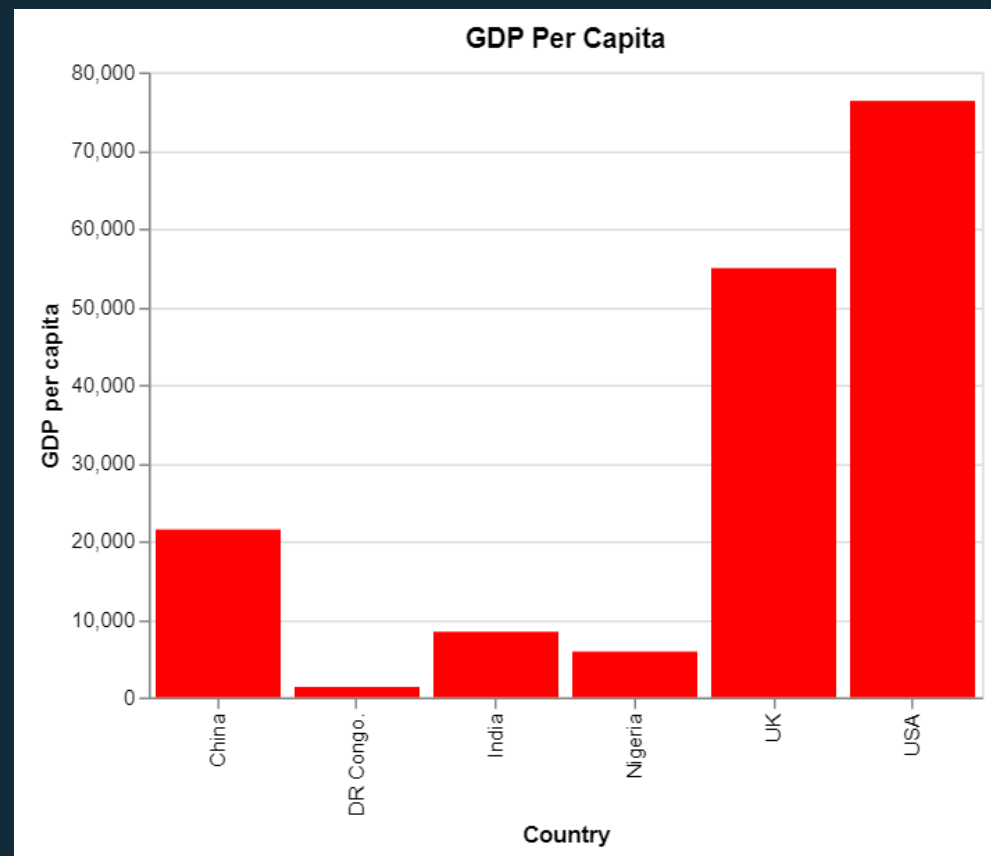
```
    "field": "GDP per capita",
```

```
    "type": "quantitative"}  
}
```



Delivering data 1.

Embedding the data in your chart



Delivering data 2.

Using GitHub to store and deliver your data

```
{  
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",  
  
  "title": {  
    "text": "Real GDP Per Capita",  
    "subtitle": ["Current International USD, PPP", "Source: World Bank"]},  
  
  "width": 400,  
  "height": 300,  
  
  "data": {"url": "https://raw.githubusercontent.com/EconomicsObservatory/courses/main/1/example\_data/chart2\_GDP\_pc\_6\_countries.csv"},  
  
  "mark": "line",  
  
  "encoding": {  
    "x": {"field": "Year", "type": "temporal", "title": null},  
    "y": {"field": "GDP pc", "type": "quantitative", "title": null},  
    "color": {"field": "Country Name", "type": "nominal"}  
  }  
}
```

Delivering data 2.

Using GitHub to store and deliver your data

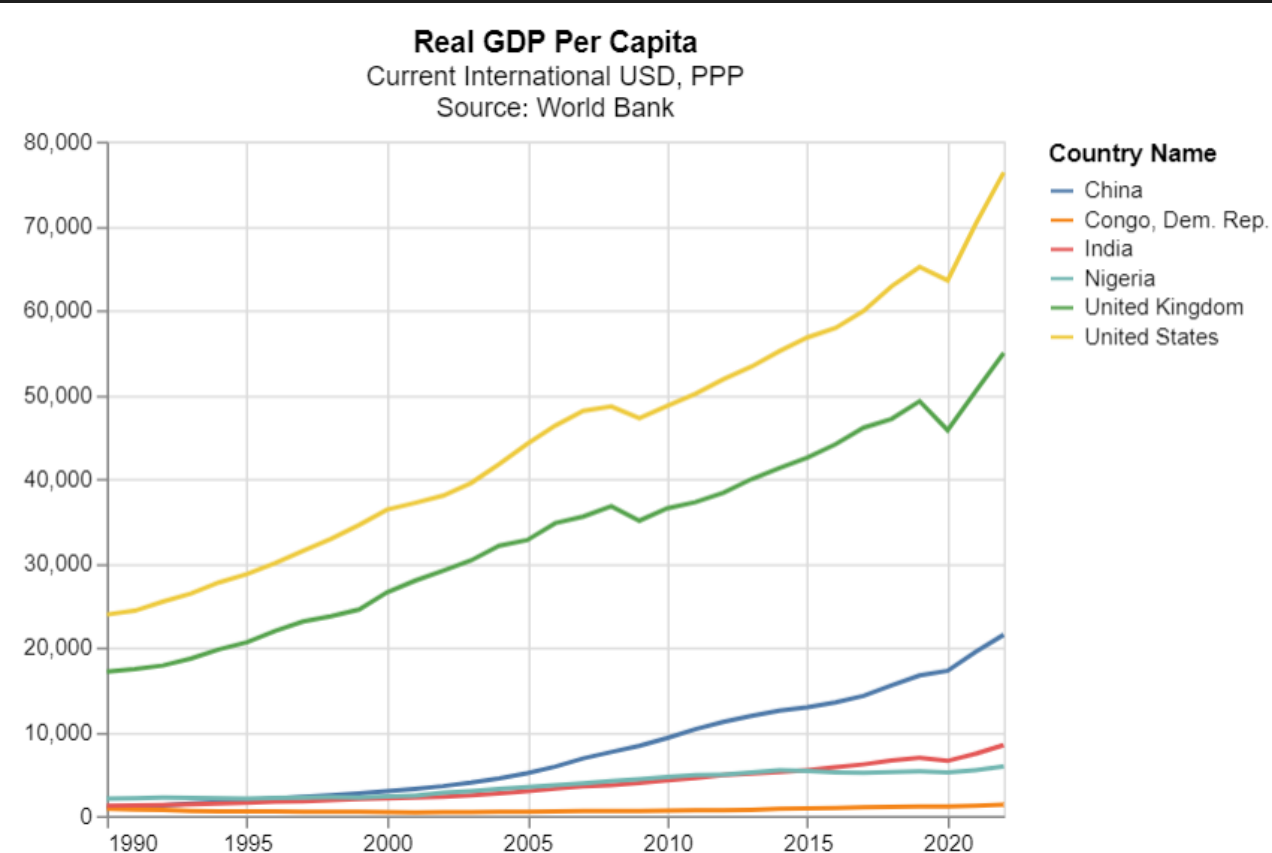
The data are stored elsewhere on GitHub, via this link

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",
  "title": {
    "text": "Real GDP Per Capita",
    "subtitle": ["Current International USD, PPP", "Source: World Bank"]},
  "width": 400,
  "height": 300,
  "data": {"url": "https://raw.githubusercontent.com/EconomicsObservatory/courses/main/1/example\_data/chart2\_GDP\_pc\_6\_countries.csv"},
  "mark": "line",
  "encoding": {
    "x": {"field": "Year", "type": "temporal", "title": null},
    "y": {"field": "GDP pc", "type": "quantitative", "title": null},
    "color": {"field": "Country Name", "type": "nominal"}
  }
}
```

Delivering data 2.

Using GitHub to store and deliver your data

```
{  
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",  
  
  "title": {  
    "text": "Real GDP Per Capita",  
    "subtitle": ["Current International USD, PPP", "Source: World Bank"]  
  },  
  
  "width": 400,  
  "height": 300,  
  
  "data": {"url": "https://raw.githubusercontent.com/EconomicsObserved"},  
  
  "mark": "line",  
  
  "encoding": {  
    "x": {"field": "Year", "type": "temporal", "title": null},  
    "y": {"field": "GDP pc", "type": "quantitative", "title": null},  
    "color": {"field": "Country Name", "type": "nominal"}  
  }  
}
```



Delivering data 2.

Using GitHub to store and deliver your data

In our spec we have the following:

```
"data": {"url":  
"https://raw.githubusercontent.com/  
EconomicsObservatory/courses/main  
/1/example_data/chart2_GDP_pc_6_co  
untries.csv"},
```

You can open the CSV link in your
browser to see the underlying numbers

Delivering data 2.

Using GitHub to store and deliver your data

In our spec we have the following:

```
"data": {"url":  
  "https://raw.githubusercontent.com/  
  /EconomicsObservatory/courses/main  
  /1/example_data/chart2_GDP_pc_6_co  
  untries.csv"},
```

You can open the CSV link in your
browser to see the underlying numbers

```
Country Name,Country Code,Year,GDP pc  
United Kingdom,GBR,1990-01-01,17091.3051155726  
"Korea, Rep.",KOR,1990-01-01,8355.3327739747  
United States,USA,1990-01-01,23888.6000088133  
United Kingdom,GBR,1991-01-01,17420.4212113977  
"Korea, Rep.",KOR,1991-01-01,9474.64259647045  
United States,USA,1991-01-01,24342.2589048189  
United Kingdom,GBR,1992-01-01,17840.5510277811  
"Korea, Rep.",KOR,1992-01-01,10184.8556645893  
United States,USA,1992-01-01,25418.9907763319  
United Kingdom,GBR,1993-01-01,18673.348462135  
"Korea, Rep.",KOR,1993-01-01,11030.7119484601  
United States,USA,1993-01-01,26387.2937338171  
United Kingdom,GBR,1994-01-01,19755.2550837302  
"Korea, Rep.",KOR,1994-01-01,12187.2549659273  
United States,USA,1994-01-01,27694.853416234  
United Kingdom,GBR,1995-01-01,20595.7082090064  
"Korea, Rep.",KOR,1995-01-01,13502.5827420794  
United States,USA,1995-01-01,28690.8757013347  
United Kingdom,GBR,1996-01-01,21946.108059434  
"Korea, Rep.",KOR,1996-01-01,14694.0962445008  
United States,USA,1996-01-01,29967.7127181749  
United Kingdom,GBR,1997-01-01,23069.3890783783  
"Korea, Rep.",KOR,1997-01-01,15721.7056576921  
United States,USA,1997-01-01,31459.1389804773  
United Kingdom,GBR,1998-01-01,23688.3819751261  
"Korea, Rep.",KOR,1998-01-01,14974.6595747118  
United States,USA,1998-01-01,32853.6769523009  
United Kingdom,GBR,1999-01-01,24493.5013657365  
"Korea, Rep.",KOR,1999-01-01,16807.1294764195  
United States,USA,1999-01-01,34515.3902272076  
United Kingdom,GBR,2000-01-01,26531.5844868141  
"Korea, Rep.",KOR,2000-01-01,18538.8355325592  
United States,USA,2000-01-01,36329.9560727102  
United Kingdom,GBR,2001-01-01,27815.2232222773
```


Delivering data 3.

Receiving data direct from an API

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",

  "title": {
    "text": "British GDP Growth",
    "subtitle": ["QoQ4 % Growth", "Source: ONS via ECO-API"]
  },

  "width": 400,
  "height": 300,

  "data": {"url": "https://api.economicsobservatory.com/gbr/grow?vega"},

  "mark": "line",

  "encoding": {
    "x": {"field": "date", "type": "temporal", "title": null},
    "y": {"field": "value", "type": "quantitative", "title": null}
  }
}
```

Delivering data 3.

Receiving data direct from an API

```
{
  "$schema": "https://vega.github.io/schema/vega-lite/v5.json",

  "title": {
    "text": "British GDP Growth",
    "subtitle": ["QoQ4 % Growth", "Source: ONS via ECO-API"]
  },

  "width": 400,
  "height": 300,

  "data": {"url": "https://api.economicsobservatory.com/gbr/grow?vega"},

  "mark": "line",

  "encoding": {
    "x": {"field": "date", "type": "temporal", "title": null},
    "y": {"field": "value", "type": "quantitative", "title": null}
  }
}
```

The data comes from an external source.

In this case the Economics Observatory API.

[There are thousands of APIs you can use, we will meet more later in the course]

Delivering data 3.

Receiving data direct from an API

```
▼ [
  ▼ {
    "date": "1956-02",
    "value": 2.6
  },
  ▼ {
    "date": "1956-05",
    "value": 2.5
  },
  ▼ {
    "date": "1956-08",
    "value": 0.3
  },
  ▼ {
    "date": "1956-11",
    "value": 1.5
  },
  ▼ {
    "date": "1957-02",
    "value": 2.2
  },
  -
]
```

APIs (in this context) are just like urls.

You can open the API link in your browser to see what is being delivered to your machine.

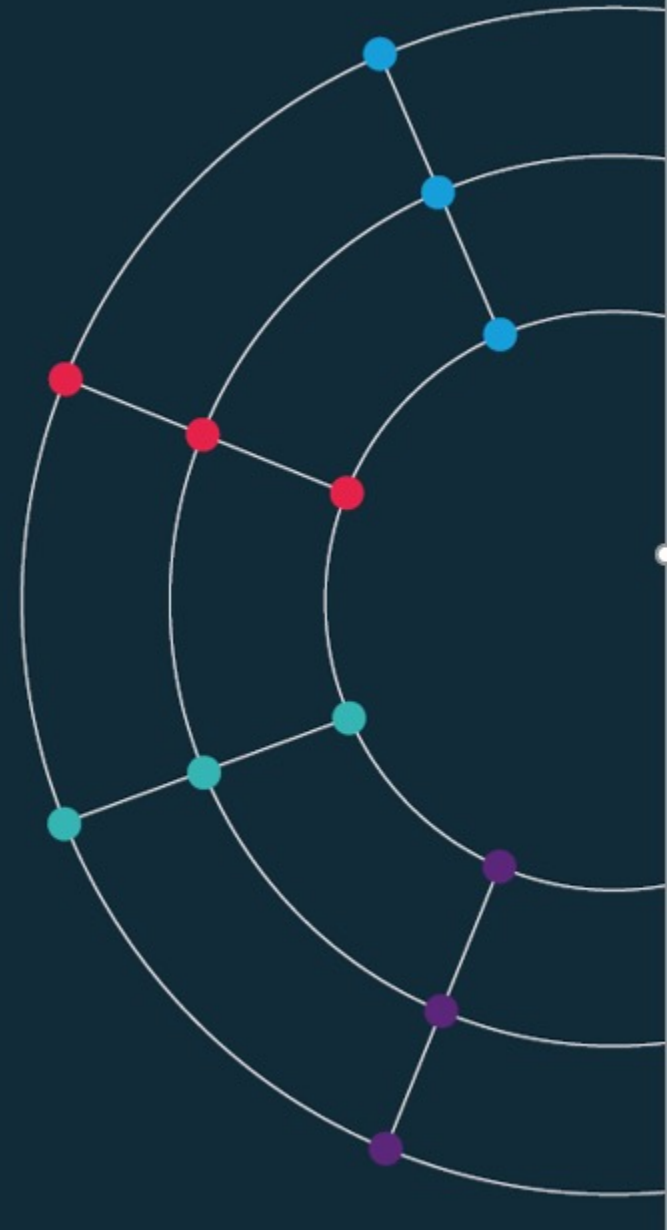
Here are the underlying numbers from the current example

Note the 'key-value pairs'

1.3 Code along 1.

Playing with Chart specs

<https://economicsobservatory.com/modern-data-visualisation>



Code-along.

First steps towards JSON visualisations

In this first practical session, we will explore the three data delivery types:

1. Embedded into the design of the chart: “s1_chart1.json”
2. Via CSV file on GitHub: “s1_chart2.json”
3. Via an API: “s1_chart3.json”

The files can be found here:

<http://economicsobservatory.com/modern-data-visualisation>

Code-along tool: <https://vega.github.io/editor>

Tool – Vega-Lite editor



Code-along: our tasks

Edit three files to make your first JSON visualisations

File 1. Inline data. ([s1_chart1.json](#))

- Paste the JSON into Vega Lite Editor (<https://vega.github.io/editor>). Now change:
 - The colour of the bars.
 - Swap some of the data to countries that interest you
 - The size of the chart

File 2. GitHub data. ([s1_chart2.json](#)). Changes:

- The colour variable
- The title text
- [*The variable being plotted*]

File 3. API data. ([s1_chart3.json](#)). Changes:

- The API data being delivered.
- The title text
- The line colour

<https://economicsobservatory.com/modern-data-visualisation>

Code-along: our tasks

Edit three files to make your first JSON visualisations

File 1. Inline data. ([s1_chart1.json](#))

- Paste the JSON into Vega Lite Editor (<https://vega.github.io/editor>). Now change:
 - The colour of the bars.
 - Swap some of the data to countries that interest you
 - The size of the chart

File 2. GitHub data. ([s1_chart2.json](#)). Changes:

- The colour variable
- The title text
- [*The variable being plotted*]

File 3. API data. ([s1_chart3.json](#)). Changes:

- The API data being delivered.
- The title text
- The line colour

When happy with your chart:

1. Copy and paste the JSON into your text editor (Visual Studio Code).
2. Save it locally on your machine.
3. Add it to your GitHub pages repository.

<https://economicsobservatory.com/modern-data-visualisation>

