Data Science - Portfolio tasks

Your portfolio makes up 20% of your grade. It is made up of 10 coding challenges (CC) each of which produce visualisations on your website which we will grade.

Each of the ten tasks will be graded out of 2%, with full marks for complete, 1% given for semi-complete or well-attempted, and 0% for missing or badly incomplete.

Your portfolio must appear on its own page, for example, mine would be: www.rdeconomist.github.io/portfolio

Clearly label each challenge on your portfolio, using a heading.

We **strongly advise** you to complete each task in the week it is set.

Portfolio – list of coding challenges

CC1. Hosting. Display charts in your own site.

Set up your own Github account and live page using GitHub pages.

Make sure you understand the difference between your repo and your site.

For example, my repo and web sites are:

https://github.com/RDeconomist/RDeconomist.github.io

https://rdeconomist.github.io

Add your first index.html file to your repo

Check that your live site is working.

Now add **two charts** using the vegaEmbed function.

Tips:

Chart options can be found here:

https://www.richarddavies.io/library

For a given chart spec, you can visualise it (check what it looks like) using the Vega-Lite editor:

https://vega.github.io/editor

Stock take: Your site should now have two charts that you found and embedded in it. [Remove any examples that were given to you in class]

CC2. Building. Create your own visualisations

Set up an account on the Economics Observatory Data Hub.

Build two separate charts using the “create” tool.

When you are happy with your charts, take the json code, and paste it into Visual Studio Code.

Save these files into your GitHub repo.

Now embed these two charts in your page using the VegaEmbed function we learned previously.

Stock take: By this stage you should have a live web site, with four embedded charts.

CC3. Debate. Use a visualisation in policy commentary

Produce two charts that support or refute (or are related to in some way) to a topic of policy debate.

Your task:

- Set out (not more than 25 words) a policy topic. [Hint: use an html <p> tag to insert a paragraph].
- Make two charts that support, or refute, it or a related to an argument on this topic. This could be two that support, two that refute, or a mixture.
- Comment (not more than 25 words) on what you find. Your commentary should be placed near the chart in a way that makes it intuitive to your site's users (i.e., above, below, or to the side).

<u>Stock take</u>: By this stage your web site, should have 6 embedded charts.

CC4. Replication. Re-create, then improve, someone else's chart.

Find a chart that a policy organisation, a journalist, think tank, television channel or company has used. Your challenge this week is to replicate it, and then improve on it. Refer back to our discussion of the do's and don'ts of visualisation for guidance.

Your task:

- Crate an <u>image file</u> of their chart. In Windows this can be done using the "snipping tool" for example. Display this file on your page. [It could be a .jpg or .png file, for example].
- Now replicate their chart in Vega-Lite. You are free to trace (i.e. make up, manually) the data to do this. Your task is not to get the data perfect, but rather to align your axes, titles and colours to look like theirs. So, if it is the Financial Times, for example, you would need to set a pink background.
- Now improve their chart. Change the titles, axes, colours and other elements of the chart specification to make it better. Explain briefly (not more than 25 words) what you have done.

Alternative. If you cannot find a suitable chart, then use Tufte's 'worst graphic' example of a bad (i.e. very low) data-to-ink ratio. Display an image of this (see Annex), then try to (roughly) re-create it in Vega; and, finally, improve it in Vega. The result of this, as above, should be one image, and two new charts.

<u>Stock take</u>: *By this stage your live web site should have 8 embedded charts, and one embedded image file.*

CC5. Accessing data: Scraper and API.

By this stage we have discussed the simple ways of scraping data from websites. We have also discussed using APIs so that your site has live link to a data source.

API task (1 point)

- Add a chart to your site that uses a live link to an API. This could be any type of chart, using any API.
- Below your chart, add a functional description of the API. That is, describe the base url, and then the elements that are needed to call data. Include the full final url that you used in your chart. See the FRED example in the lecture notes (Week 4) as an example.

Scraper task (1 point)

Using a Google Colab python notebook, scrape a website. 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) on what you did – the data source, challenges etc.

CC6. Loops. Build a dashboard.

[1st point]

Use a loop to batch download six different series as JSON files.

Save these to your GitHub account and use these (as “raw” files) to supply the data to six (or more) charts on a theme of your choice.

Notes:

- We have discussed various APIs in class. You could use one of these.
- We have discussed using loops to batch download from APIs, using FRED. Apply these skills to your API of choice.
- Charts should be time series.
- You can cover any topic, or topics of your choice.
- Given that you are making 6 very similar charts, they should be smaller than the normal chart size you are using on your page.
- Display them as a grid, i.e. 3x2 or 2x3 depending on the style of your page.
- Ensure you add a link to the Google Collab python notebook where you conducted your data analysis. **Make sure you make it publicly accessible**.

[2nd point]

As above, but also use a loop in your Javascript in order to embed the six charts.

That is, rather that writing out embed six times, you should be writing it once, inside a loop that iterates six times.

CC7. Maps. Base maps and choropleths

Produce **two maps** and embed them in your portfolio page.

Country coverage:

- One map should be of Scotland
- The other should be of Wales.

Types of map:

- One map should be a coordinates map.
- The other map should be a choropleth.

What data to map?

- This is up to you. It can be anything that you are interested in.

If in doubt, refer to the lecture and seminar, where extensive guidance on mapping these countries was given.

https://github.com/RDeconomist/RDeconomist.github.io/tree/main/charts/maps

CC8. Big Data. Extracting a story from millions of prices.

Produce two charts using either of the two UK prices datasets provided in class. Both charts can be from the same database, or you can produce one from each. Explain, in no more than 50 words, what you have done.

Notes:

Two live research datasets are discussed at length in the lecture and seminar.

- **Long Run Prices Database (LRPD)**. This data was first set out in Davies (2021). It is monthly data, starting in 1988. It is for both supermarkets and smaller stores. It has a regional marker.
- **AutoCPI data**. This data was first set out in Davies and McElvoy (2024). It is just for the largest supermarkets. This is daily data, starting in 2023.

Accessing the data

- **LRPD**. You can read about the LRPD database, and see related papers, here: https://richarddavies.io/prices. To access the data, the CSV files can be found HERE.
- **AutoCPI**. Access to the AutoCPI data is given via the Colab notebooks that are discussed in the lecture and seminar.

Tips

You will need to simplify (AKA reduce, collapse, group) the data before you plot it. Code showing how to do this is taught in the lecture.

On common pitfall is to attempt to plot a scatter of all the data for a single product. This can result in thousands of observations for some products, which may make the chart very slow to load (and your website very slow). We advise you reduce down the data further.

CC9. Two interactive charts.

Interactive charts

Produce **two charts** that include interactivity.

The simplest interactivity we discussed—tooltip, and simple colour selection—do not count.

It could be a slider, drop-down box, clickable legend. Or any other form of interactivity that allows the user to better understand your data.

Examples of interactivity are discussed at length in our lecture

Examples can be found on my chart library.

www.richarddavies.io/library

Further examples are provided throughout the course via Colab notebooks. And there are many on the Vega-Lite site.

Note/tip: These two charts can be related to (and used in) your project.

CC10. Advanced Analysis and Machine Learning

Advanced analytics chart [one point]

Produce a chart that uses more advanced analytics that standard line, bar or scatter charts. This could include any of:

Bubble, histogram of distributions, de-trended (including univariate regression), shock analysis, Diff-in-Diff, heat maps. Examples, including code, are discussed in the lecture.

Machine learning [one point]

Conduct an applied data analysis using any of the machine learning techniques taught in the Week 10 lecture and seminar.

One option is Supervised learning (Regression or Classification); another is the Unsupervised learning (Clustering or Dimensionality reduction) method.

Make sure that you:

- Transform the input data in the standardised X matrix form.
- Transform the target data in the standardised y vector form for the Supervised learning task.
- Use sklearn inside python to apply the machine algorithms. You may use any algorithm found in the library, not just the ones discussed in class.
- Submit a link to your Google Colab workbook. Make sure your workbook has read access, and there is a clear link to it.
- Set out (not more than 25 words) a hypothesis you will examine.
- Conduct the data analysis in python. Make at least one chart. This can be done in python (matplotlib), or you may choose to export the data and visualise it in Vega-Lite.
- Comment (not more than 25 words) on what you find.

Note/tip: This work can be related to (and used in) your project.