Executable Research Papers in Bioinformatics: A Case Study

Bernhard Finke

Master of Science
Artificial Intelligence
School of Informatics
University of Edinburgh
2022

Abstract

Executable Research Papers are a valuable, though underused, resource to improve the replicability of computational research. In this paper we walk the reader through the development of such an executable paper, highlighting common issues. We present feedback received from researchers, which indicates that there is a real interest in their further growth. To further push executable papers into the mainstream, journals need to take concrete steps, including streamlining their development and providing real incentives for researchers.

Research Ethics Approval

This project obtained approval from the Informatics Research Ethics committee.
Ethics application number: 439060
Date when approval was obtained: 2022-07-22
The participants' information sheet and a consent form are included in the appendix.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Bernhard Finke*)

Acknowledgements

Thank you to Prof. Douglas Armstrong for providing invaluable guidance throughout the process of developing the Master's Project and dissertation.

Thank you to Dr. Oksana Sorokina for your support and for being so helpful whenever I faced technical issues.

Thank you to those researchers who took time out of their busy days to provide me with feedback.

Thank you to Henry and David for proof-reading this dissertation.

Table of Contents

Chapter 1

Introduction

Much has been made of the replication crisis in fields like psychology, with one research paper attempting to replicate 100 studies finding that only 36% of replications returned significant results [5]. The common perception that psychological studies are not reproducible has since led to much rethinking in the field and journals are now more likely to publish replication studies than they were previously [15]. In computational research, where issues like differences across study populations are not a key factor in reproducibility, it could be argued that replication is still just as important of a concern. One study, published in 2022, retrieved 2109 packages of code in the programming language R from a Harvard repository and found that 56% of code packages could not be executed without issue, even when ”code cleaning was applied” [25].

Executable Research Papers (ERPs) may be one of the most valuable tools to address this computational replication crisis. First proposed in 2011 [8], and then largely forgotten about until as recently as 2020, when journals like eLife started to roll out ERP publishing tools [26], ERPs can enhance reproducibility in two meaningful ways:

- By enouraging researchers to provide the code used for their analysis in an accessible manner, written for the purpose of being used by others, rather than being executed by the original researcher once in the specific software environment they were using at the time.
- By allowing ERP users to interact with the parameters chosen for analysis and alter methods or assumptions to see how this would change the final results.

To assess what issues currently exist with ERPs and what can be done to encourage wider publication of ERPs, we developed an ERP of our own in the field of bioinformat-

ics. The ERP is an interactive implementation of the Nature article, 'A unified resource and configurable model of the synapse proteome and its role in disease' ('A unified resource'), developed in collaboration with two of the authors of the original paper, Dr. Oksana Sorokina and Prof. Douglas Armstrong [22]. The paper collected and analysed over 8000 proteins previously detected in brain synapse studies and our ERP allows users to interact with the analysis in a simple and user-friendly way to provide a more in-depth understanding of the synaptic dataset.

This dissertation gives an overview of the development process, beginning, in chapter 2, with some background on the analysis in 'A unified resource' as well as on the concept of executable papers. We continue into chapter 3 by addressing the requirements for the app and proceed into the details of how the app was implemented. We also provide some examples of downstream analysis made possible by the app and its features. To assess the value of the app in as objective a manner as possible we requested feedback from synaptic researchers, which we summarise in chapter 4. Finally, we conclude with recommendations for the future of ERPs.

Chapter 2

Background

2.1 A unified resource and configurable model of the synapse proteome and its role in disease

In this section we give a brief overview of the contributions and analyses in the paper 'A unified resource'.

The primary contribution and purpose of the paper was the integration of 58 different published datasets from 2000 to 2020 on the synaptic proteome, or the set of proteins found in brain synapses, into a single SQLite dataset containing 8,087 proteins found in studies on humans, rats, and mice [22]. The database was then annotated with supplementary information: firstly, proteins were combined with the papers they were found in, including the sub-cellular localisation the paper looked at, either 'postsynaptic', 'presynaptic' or 'synaptosome', the brain region, of which 16 were included in the

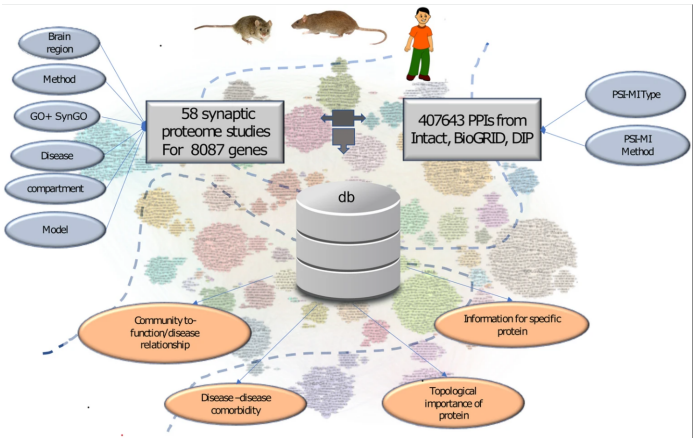

Figure 2.1: Fig. 2 in original paper: Structure of SQLite database [22]

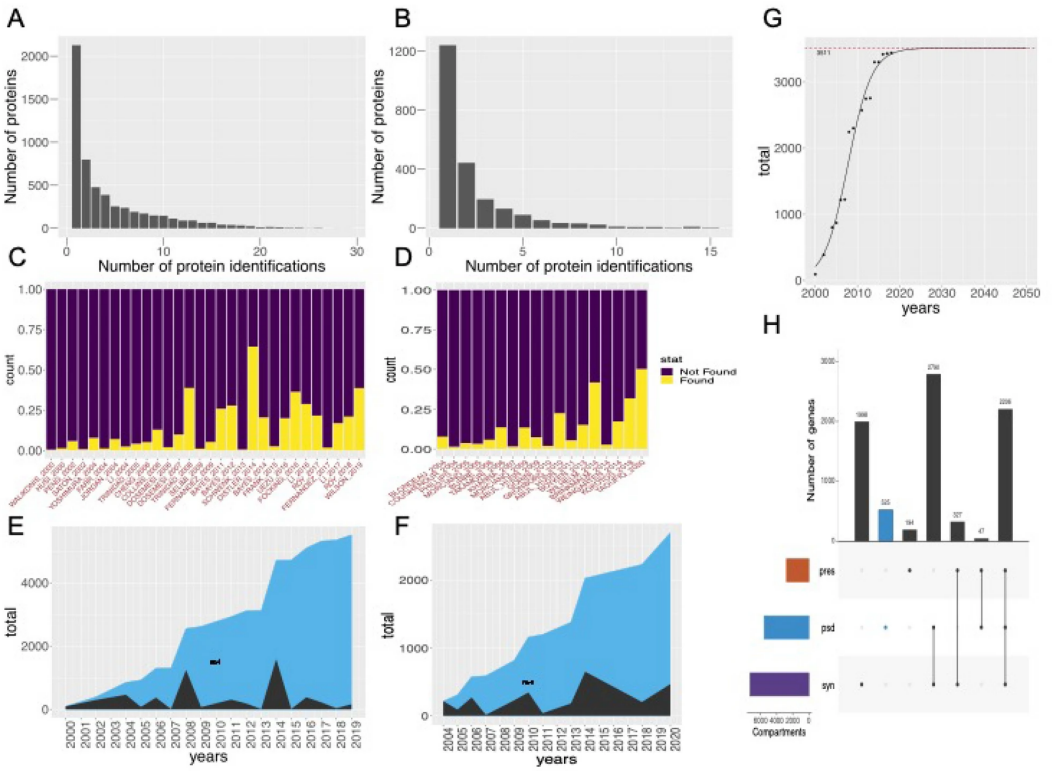

Figure 2.2: Fig. 1 in original paper: Visualisation of synaptosomal proteins over time and by paper [22]

database, the method used to sequence the proteins, 'shotgun' or 'target' and the mammal species studied [20]. Additionally, proteins were annotated for commonly associated diseases and GO biological processes. Separately, pairs of proteins were combined with their known protein-protein interactions (PPIs). The database includes proteins from 28 postsynaptic (PSP) studies, covering 5560 proteins, 18 presynaptic studies which contributed 2772 proteins, and 11 synaptosomal studies covering 7198 proteins. Figure 2.1 gives a visual overview of the structure of the database.

Beyond the database, the paper also includes some analysis on the proteins identified in the synaptic proteome, where figure 2.2, or figure 1 in the original paper, covers the distribution of proteins over time and by paper, and figure 2.3, figure 3 in the original paper, presents some examples of more complex analysis made possible by the provided database [22]. Figures 1A and 1B cover the amount of times proteins were identified in papers, split up by postsynaptic and presynaptic proteins respectively. Figures 1C and 1D visualise what percentage of known proteins were identified in each study, again broken up by sub-cellular localisation. Figures 1E and 1F, again split up into postsynaptic and presynaptic proteins, show how new proteins have been accumulated

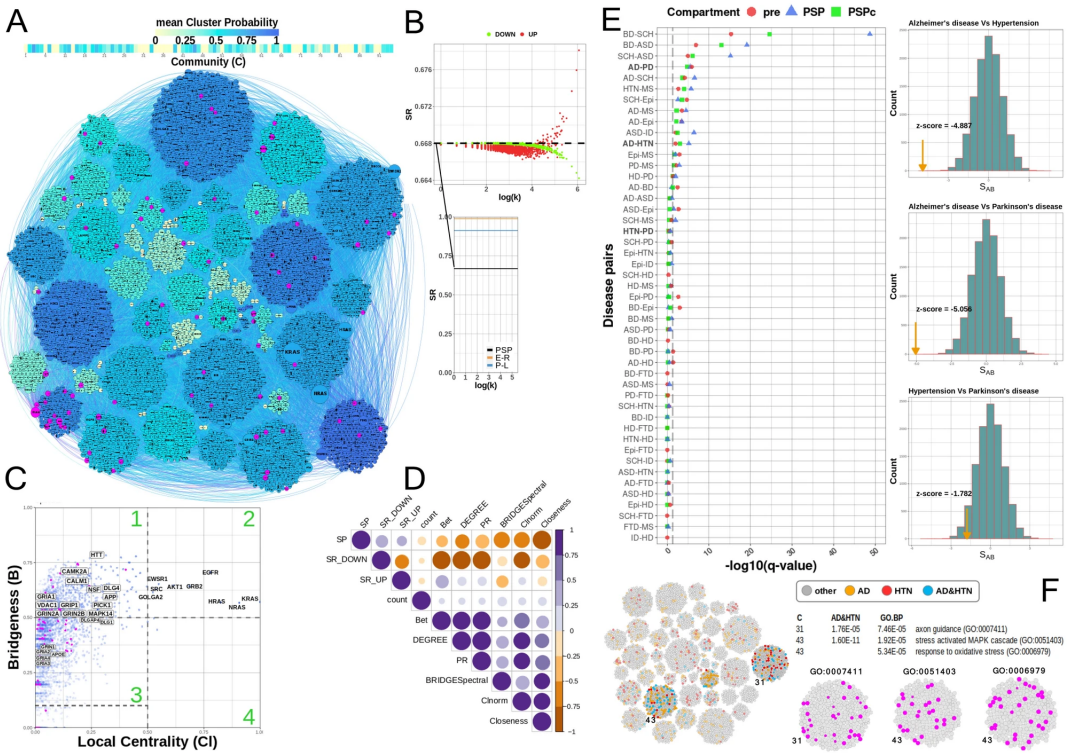

Figure 2.3: Fig. 3 in original paper: Investigation of network structure of proteins and their interactions with disease [22]

over the years since 2000, with new proteins in a year coloured black, and the total number of proteins coloured blue. Figure 1G also shows how postsynaptic proteins have accumulated over the years, but this figure only includes so-called 'consensus' proteins, or those proteins that have been identified in two or more studies. Figure 1G additionally includes a sigmoid curve, which is used to determine the likely maximum number of postsynaptic consensus (PSPc) proteins which will be identified in total. Using the asymptote of the sigmoid curve, the authors believe that 3,499 proteins will be identified as PSPc proteins, meaning more than 98% of total proteins have already been identified. Finally, figure 1H is essentially a Venn diagram, showing unique and overlapping proteins in each of the three sub-cellular localisations.

Figure 3 concerns itself with the network structure of the PPIs and their associations with disease [22]. All plots in this figure visualise only PSP proteins, as these were of highest relevance to the paper. We skip over figures 3A and 3B, as these were not included in the executable paper, see section 3.1.1. Figure 3C shows us the 'bridgeness' of PSP proteins compared to their local centrality (Cl). The authors use a specific clustering method called 'spectral clustering' to determine community membership, or to which cluster each protein belongs. Using the calculated communities, we can

estimate bridgeness, a measure of to what degree a given vertex in a network, in this case the proteins, belong to multiple different communities [18]. Bridgeness is defined as the normalised distance between its "membership vector" $\mathbf{u_i} = [u_{1i}, u_{2i}, ..., u_{ci}]$, measuring the probability of a vertex v belonging to each community, and the "reference vector" $[\frac{1}{c}, \frac{1}{c}, ..., \frac{1}{c}]$, where c is the number of communities identified by a given clustering algorithm:

$$b_i = 1 - \sqrt{\frac{c}{c-1} \sum_{j=1}^{c} (u_{ji} - \frac{1}{c})^2} \tag{2.1}$$

Local centrality, also often referred to interchangeably as semi-local centrality, is another method for estimating the "topological importance" of vertices in a network, using the degree of a vertex and the number of nearest neighbours [21]. The local centrality Cl(v) of vertex v is given by:

$$Q(u) = \sum_{w \in \Gamma(u)} N(w) \tag{2.2}$$

$$Cl(v) = \sum_{u \in \Gamma(u)} Q(u), \tag{2.3}$$

where N(w) is the count of nearest and next nearest neighbours for vertex w, and $\Gamma(u)$ contains the set of nearest neighbours of vertex u. Combining the two measures then allows for the categorization of proteins, by their 'global and 'local' influence, with proteins in region 1 considered true bridging proteins [22]. Figure 3D shows the reader how different centrality measures, including for example degree centrality, which measures the "total amount of direct links" with other vertices, and closeness centrality, measuring the sum of distances from one vertex to other vertices, correlate with each other [28]. A correlation close to 1 implies that two centrality measures identify a large number of the same proteins as having similar centrality [22]. Figures 3E visualizes disease-disease relationships, with points to the right of the x-axis indicating a higher likelihood of disease overlap. Disease overlap can be quantified using the following equation:

$$S_{AB} \equiv \langle d_{AB} \rangle - \frac{\langle d_{AA} \rangle + \langle d_{BB} \rangle}{2}, \tag{2.4}$$

where $\langle d_{XX} \rangle$ represents the "mean shortest network distance between proteins associated with disease" X [21]. $\langle d_{AB} \rangle$ then quantifies the "mean shortest distance between diseases" A and B. Large positive S_{AB} values imply a high separation between

diseases, while large negative values imply a significant overlap. The same process was then repeated using a randomised model, over 10,000 iterations, ”drawing the same number of [Gene Disease Associations (GDAs)] for each disease at random” [21]. The original and randomised models are then compared to obtain a z-score:

$$z - score_{AB} = \frac{S_{AB} - \langle S_{AB}^{rand} \rangle}{\sigma(S_{AB}^{rand})} \tag{2.5}$$

Lastly, p-values are calculated for the significance of disease pair associations, and these p-values are further propagated to obtain the final q-values in the plot [22], where a q-value is a method designed to reduce the false positive rate in situations where multiple tests are performed simultaneously [23]. The plot shows a dashed line at $q = 0.05$ to highlight the point at which disease overlaps become statistically significant. Plot 3F shows us the community structure of the PSP network, similarly to plot 3A [22]. In this plot, however, disease-associated proteins are highlighted, for the diseases Alzheimer's Disease (AD) and Huntington's Disease (HTN). Using a Fisher Exact Test, it was determined that communities 31 and 43 were enriched in both AD- and HTN-associated proteins.

2.2 Executable Research Papers

The concept of executable research papers (ERPs), or executable research articles, was first championed by the 'Executable Paper Grand Challenge Workshop', taking place in 2011 [8]. The authors motivate the need for ERPs by highlighting how ”the crucial elements in data intensive research”, namely the data sets and code used to perform analysis, are often completely separate from the final research paper. Utilized data and code is then either 'available for request' or found on ”personal or institutional websites”, usually accompanied by issues with consistency and documentation. We can therefore define an ERP as a document containing the data, code, and analysis contained in a classical research paper, in an interactive and user-friendly format. An article on reproducible research published as far back as 2001, gives what they refer to as Claerbout's Principle: ”An article about computational science in a scientific publication is not the scholarship itself, it is merely advertising of the scholarship. The actual scholarship is the complete software development environment and the complete set of instructions which generated the figures” [6].

If computational researchers have been aware of these issues for decades, why then

have computational researchers and research infrastructures been so slow to integrate ERPs into the publishing workflow? In bioinformatics, for example, the first mainstream attempt at allowing the publication of ERPs only occurred as recently as 2020, when the biomedical science-focused journal eLife rolled out their ERP functionality [26]. Nüst et al. give some reasons why researchers may choose not to commit to reproducible research, including: ”a lack of incentives in terms of citation and promotion”, as well as ”the effort required to clean data and code” [19]. To be clear, the concept of the ERP can be situtated in the midst of a more general, burgeoning ’Open Science’ movement in all fields of study, but particularly in computationally-focused fields, aiming to improve reproducibility of research. Though ’Open Science’ is still an ill-defined term, there are a number of commonly-associated concepts, which we list here [27]:

- Open Code: also often referred to as open-source, is a principle that underlines the importance of providing source code required for a specific piece of research, made available to the public, under a ”licence that grants users rights to run the program for any purpose, (...) to modify it, and to redistribute it (...)” [7]. The concept traces its origins back to the early 1970s, when the so-called ”Hacker Ethic” established practices of ”openness, collaboration and knowledge-sharing”, and was further bolstered by the development of the Unix operating system, which was the first operating system to provide all of its source code under a ”permissive license”. Claimed benefits of open code practices include an increased trust by readers and the general public in the accuracy of the research, improvements in reproducibility and a greater influence on other researchers [2].

- Open Data: this is a very similar idea to open code, in which scientific data is made available for re-use ”without price or permission barriers” [17]. The concept is more novel than open code, but provides similar advantages to open code, including better reproducibility and greater trust in research. In his article in Nature Precedings, Murray-Rust highlights the mismatch between researchers, who generally view published data as belonging to the public, and journals, who consider published data to be their copyrighted property.

- Open Access (OA): this term refers to the online publication of articles in scholarly journals provided without restrictions on access [10]. Laakso et. al highlight how open access publication has been made possible by the Internet, as online publication removes many of the costs associated with traditional publishing including printing and shipping costs of physical journals. They distinguish

between Gold OA, in which the document is ”made available by the publisher to whom the document has been submitted”, and Green OA, where a work is archived by the author(s) of the article on their own website or their institution’s website. To accentuate the growing importance of OA, they show that the publication of OA research has ”grown at a much faster rate than” the total volume of published research articles.

Having established the growing importance of concepts in Open Science, we return to the subject of ERPs. In a blog post by eLife relaying the experiences of researchers who have created ERPs using their service, ERP authors highlight their positive experiences in developing executable versions of their research [24]. All authors surveyed responded that they would consider publishing another ERP, and would recommend publishing ERPs to other researchers. However one author underscores that there should be more incentives to creating ERPs built in to the publishing industry, while others mention that ERP publishing tools need to be streamlined further, to allow for more easily accessible publication. Another interesting aspect of the responses is that many of the surveyed authors indicated that they had already had well-written and reproducible code, which made the publication of an ERP relatively straightforward. Consequently, a push to make ERPs more established could lead to more generally high-quality code practices among computational researchers. While a survey of researchers who have successfully completed ERPs may not fully represent the opinions of all scientists, especially those who decided not to complete an ERP as it was intractable for them, the responses nonetheless indicate that ERPs have the potential to travel further into the mainstream in computation-focused research.

In a 2020 comment in Nature, Lasser provides an overview of some issues associated with her experience completing an ERP [12]. The first such issue is hosting, both of the code, images and data going into the analysis, as well as the interactive paper itself. She underlines that finding a suitable online platform to host the paper itself means that the final users do not have to install any software themselves, significantly improving on ease of use. Another key issue is striking a balance between reproducibility and the ”narrative flow” of the document. Crucially, similarly to the researchers in the last paragraph, Lasser reflects positively on the whole experience, with one of the most important benefits of an ERP being that it is possible to present and contrast multiple different methods of analysis, since page limits imposed by publishers are no longer a concern in executable papers.

To conclude this section, we provide an overview of biomedical journals and re-

search infrastructures allowing publication of ERPs. According to an older guide on the 'Software Sustainability Institute', Nature had been considering it, but the link provided no longer works, and no updates have been provided by Nature [4]. As already mentioned, eLife provides relatively extensive ERP publishing capabilities and their collection currently contains 12 published papers [3]. Their approach to publishing ERPs is very reasonable, with the full text of the article interspersed with figures and their associated code, and hosted on eLife directly, which can be executed live by the reader, simply by pressing a button [14]. A drawback to their approach is that it is not apparent which variables or parameters it is possible to change in the figures, nor is there extensive documentation to allow the reader to easily understand what is going in the figures. Overall, their publication style focuses more on reproducibility than on comparing methods of analysis, which Lasser highlighted as a potential advantage [12]. EBRAINS, a "digital research infrastructure" dealing with brain-related research and allowing the publication of 'live papers', currently list 24 published papers [1]. Their infrastructure focuses more on publishing specific brain-related data, including "morphologies" and "electrophysiological data", but also allows for sharing custom datasets and files, either hosted externally or on their collaboratory. As a result, published models and executable papers are not standardized on EBRAINS.

Chapter 3

Design

3.1 Requirements

In this section we provide a high-level overview of the requirements for the paper, determined through extensive communication with my supervisor and the lead author of 'A unified resource'.

3.1.1 Content

In terms of the actual content of the ERP, it was decided that adding interactive versions of the figures would be the primary added value in the executable paper. Re-doing the analysis that went in to creating the database was not deemed necessary, as this was relatively straightforward, and full walkthroughs of the usage of the database are already available through Edinburgh DataShare [20]. Instead, the central idea was to replicate the figures and to add different views of the data set and different methods to compare how this affects the final visualisation. In the original paper, for example, the first set of figures includes only plots of postsynaptic and presynaptic proteins, due to constraints on the number of plots imposed by the publisher, where in the executable version it is possible to show genes from synaptosomal studies as well. An example from figure 3 would be that different clustering methods can be compared, rather than just showing clusters determined by the spectral clustering algorithm.

Out of the 14 plots from figures 1 and 3, it was decided that only two would be left out of the executable paper: 3A, showing the "community structure" of the PSP network, with clusters coloured according to their mean cluster probability, was not included as it was deemed infeasible to include in a live paper [22]. Figure 3B, visualising

”graph entropy plots (...) comparing the structure of the PSP network (...) against (...) randomised models of similar size”, was also not implemented as the authors I was in contact with considered it sufficient to have demonstrated this analysis once, and deemed that there was no additional value in presenting the plot for different sub-cellular localisations. Still, given more time, it might have been possible to determine a way to include plot 3A. Including all of the original plots would lead to a more complete executable paper, as well as make it more clear to users how these plots were generated.

Aside from the interactive figures, it was decided that all other content in the original paper should be included in the executable version. This includes all of the text from the paper, hyperlinks to supplementary materials, as well as the original non-executable figures and their captions. We believe that this approach provides a potential template for future ERPs, especially as the ERP essentially supplants the original paper, adding additional features, without excluding any of the original content. A clear drawback to this suggestion is that publishing all computationally-focused research papers as executable papers would result in additional work for the authors, without providing further incentives.

3.1.2 Style

Another important concern was the software used to develop the paper. Since the figures in the paper require a lot of data manipulation and augmentation, the two obvious candidates for the programming language chosen were Python and R. As the original code to create the figures was written in R, it was considered more convenient to use R for the executable paper as well. This meant that some code could be re-used and that the plots resemble those in the original paper more closely. A completely different approach would have been using a dashboard framework, such as Microsoft’s Power BI. While these kinds of products provide straightforward methods to develop interactive visualisations, which are often very visually appealing, they are also primarily designed for business analytics and are unlikely to provide some of the more complex functionalities we require, such as large network visualisation.

Having chosen a programming language, the next task was to choose a suitable framework for developing the paper. The first choice here was between using a static document using a notebook like RMarkdown, and a dynamic, interactive framework, like RShiny. Even though using a static document could make hosting easier, especially if only the final pdf file is included, we chose to use RShiny. This is because RShiny

includes in-built functions for selecting parameters and we believe being able to select parameters using a menu greatly improves ease of use over having to directly interact with the code. Another alternative is Jupyter Notebook, which provides support for both Python and R. While such a Notebook, containing text interspersed with code blocks which can be executed live by the user, would be a viable option, we still prefer the ease of use provided by RShiny's selectors.

The final choice of framework came down to the format of the RShiny file, either an app or an RMarkdown style document. There are only marginal differences between the two solutions. Using an app was chosen, because, based on preliminary research, it seems like it is slightly easier to host an RShiny app as well as to apply formatting. Using a Shiny RMarkdown document however allows for inserting code chunks directly into the document. While this is a feature commonly associated with ERPs, we determined that it was not absolutely necessary, especially as the hosted app can be accompanied by the source code wherever it is hosted online. Additionally, adding code chunks could interfere with the narrative flow of the document. Still, if it was determined in the evaluation that researchers strongly preferred including code chunks, it would be relatively straightforward to convert the app into a Shiny RMarkdown document.

Finally, another important aspect of the style of the paper is the actual formatting of the paper. The supervisor of this project highlighted that the formatting of publications in Nature and similar journals can improve readability. RShiny allows for straightforward formatting, as it has CSS functionalities built in. The main formatting changes which were implemented were: increasing the left and right margins, so the text and figures do not take up all of the horizontal space on the page, putting the non-executable versions of the figures in boxes so they are more obvious to the user, and increasing the line spacing of the text.

3.1.3 Hosting

The last issue discussed with the authors of the original paper was how and where to host the ERP. Since Nature, which published the original paper, does not provide ERP functionality, this was not an option. Additionally, publishing an ERP with eLife is only possible if the original paper was published through eLife. The final option was EBRAINS, which the authors indicated a preference for. As the original article deals with brain-related research, this also fit thematically. So far though, the process of getting the app onto EBRAINS has proved a little difficult. The idea was originally

to host the app directly through the EBRAINS collaboratory, however after weeks of communication with one of the members of the support team at EBRAINS, another member of the support team informed me, rather frustratingly, that the running of Shiny apps through their collaboratory is not currently supported, and EBRAINS had no plans to implement this feature. The alternative provided by EBRAINS was to host the app directly through their containerization software 'OpenShift'. Due to the need to get a working version of the app out to researchers in time, so that they can provide feedback before the dissertation deadline, this part of the process has been put on the back burner and will be re-examined once final tweaks to the app have been made in response to researchers' feedback.

As a result, the app is currently being hosted online using the ShinyApps service, and can at the time of writing be found at https://bfinke.shinyapps.io/bioerp/. While ShinyApps provides a very simple method to host the app, simply requiring making an account with the service, installing the library 'rsconnect' and running the command $deployApp()$ in the RStudio development environment, it has distinct drawbacks which mean it would not be ideal for hosting the app indefinitely. The first is that ShinyApps only allows for 25 hours of 'active' time per month with a free account, meaning there could be a point when a researcher wants to access the app, but is not able to. The second major drawback is their memory limit, set at 1 gigabyte with a free account. Since some of the files used in the app are very large, it is possible for the app to run into this memory limit and crash for a user. Nevertheless, working within this requirement has led to some optimizations made, which significantly improve the app's efficiency and thus have improved the overall quality of the app. While it would be preferrable to already have the app deployed onto EBRAINS, for the purposes of testing and evaluation, ShinyApps is a suitable temporary solution.

Aside from the hosting of the app, hosting the source code and data files is also important. Similarly to the SQLite dataset, these files could be hosted on Edinburgh DataShare. A link to this page on DataShare could then be provided along with a link to the final OpenShift implementation on an EBRAINS Live Paper webpage, so that users can both interact with the app and inspect the analysis behind it simply and from a single page.

3.2 Details of Implementation

In this section, we provide a more in-depth specification of the implementation process of the app.

3.2.1 Overall App Implementation

A Shiny app consists of two core functions: *shinyUI*, providing the user interface of an app, and *shinyServer*, which deals with the computation required for the app. An app can be written in two separate files, one for the UI and one for the server, or it can be written in a single file, called app.R, with the line $shinyApp(ui = ui, server = server)$ invoked at the end of the file. The second option was chosen for this project, as it was felt this would make the source code easier to comprehend and look over.

The UI function is relatively straightforward and contains the following:

1. CSS styling to implement the required formatting choices.

2. The full text of the original article, with additional text to explain the newly added elements in the executable paper, including hyperlinks to other parts of the paper and external supplementary materials.

3. Selectors for choosing parameters included in each of the figures, see figure 3.1 for an example. Selectors for plots in figure 1 are labelled to indicate which parameters correspond to the choice of parameters in the original paper. This was not the case for selectors in figure 3 plots, as there are some significant differences between the parameters in these plots in the ERP and original paper.

4. The placement of the static and interactive figures as well as tables and the added dataset download button, see section 3.2.4, defined in the server function.

The order of the elements in the ERP were preserved exactly as in the original paper, with the executable figures inserted directly below the static figures and their descriptions.

The server function contains the code used to create each of the interactive plots and tables, as well as to render the static figures saved as png files. It also includes the code used to download the specific views of the data which go in to creating the plots. Outside of both functions, we define the functions used to generate the data going in to each of the plots. They are named *GenerateFigN*, where N is the number and letter

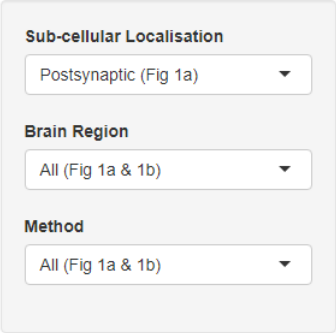

Figure 3.1: Example of a selector in the ERP (selector for figure 1a and 1b)

associated with a given plot, for example 1h. These functions are called by both the plotting functions and the download handler, and take as input the specific parameters chosen by the user. Other elements found in this section, outside of the UI and Server functions, include:

1. Various helper functions, such as switch statements mapping disease ids to their common abbreviations.
2. Two generic selector functions, defined here to reduce code duplication.
3. An empty plot, containing the text 'No proteins in current selection', which is used in all plots in figure 1 when a user selects a set of parameters containing no known proteins.
4. Code to load in and select required data from the SQLite database and pre-defined cluster graphs, used in various plots in figure 3.
5. The loading of required libraries for running the code. It was aimed to use as few external libraries as possible, so that users can more easily work with the source code and so that no compatibility issues occur with other versions of R, than the one used in development. In total, only 11 libraries are required for the ERP, including *shiny* itself, and three standard Tidyverse packages: $dplyr$ and $tidyr$, used for manipulating data, and $ggplot2$ required for plotting most of the figures in the ERP. The other 7 packages are used for more specific purposes and we will enumerate them in the next sections.

3.2.2 Figure 1

To generate the plots in figure 1, data from the SQLite database is required. To load in the SQLite database, we use the R package *RSQLite*. Almost all of the data required for figure 1 can be found in the SQL view 'FullGeneFullPaperFullRegion'. Specifically, we select: 'Localisation', 'BrainRegion', 'MethodID', 'HumanEntrez', 'Paper', and 'Year' from this view. As the view contains the ids not the names of sequencing methods, we also select 'ID' and 'Name' from the Method table. We then merge both tables on 'MethodID' and remove 'MethodID' to replace the method ids with their names. Finally, we disconnect from the SQLite database.

Our first plot allows the user to visualise both plots 1A and 1B, as we can now simply switch between sub-cellular localisations rather than having to present the plots separately. We can also visualise proteins from synaptosomal studies, which were not included in the original paper, as well as visualising all of the 58 studies on the same plot. Similarly, plots 1C and 1D, as well as 1E and 1F, were also combined into a single interactive plot. The other parameters which can be chosen in this plot are the brain region of a study which identified proteins and the sequencing method used in a given paper. Retrospectively, we could have also added an additional parameter to choose in which mammals proteins were identified, with the likely options being human, mouse, rat, and all. Still, since the database only contains 58 studies, with only 5 looking at humans, adding this parameter may give the false impression that a number of proteins are not present in a given species. This issue also occurs with the brain region parameter, since a number of given brain regions only saw 1 or 2 studies. If in the future an updated, more extensive database was developed, it would be easier to see the value of both of these parameters. Looking at the evaluation results, see section 4.2, however, no-one, including the original authors of the paper, bemoaned the omission of this parameter, so it does not seem to be an important parameter to researchers. To generate this plot, we first call the function *GetNumPapersSubset*, which filters for a selected localisation, brain region, and method, and then annotates each protein with the number of papers it has been found in. The data generator function then selects each unique protein, using the column 'HumanEntrez' and creates a table of the number of proteins identified a certain number of times. In the server function, this table is then input into a bar chart using *ggplot*2.

The second plot, combining plots 1C and 1D, includes the same parameters to select, with the addition of an evidence level, corresponding to the minimum number of papers

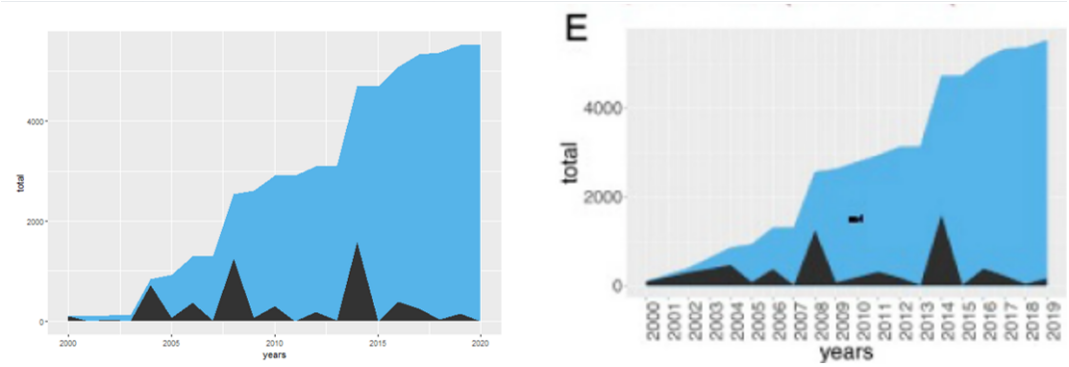

Figure 3.2: Side-by-side of plot 1E from ERP (left) and original paper (right)

a protein has to have been identified in to be included in the plot. This parameter was not present for plots 1A and 1B, as it would merely remove the first n bars, which would not add to the analysis present in those plots. The function to filter for a given evidence level, *SubsetNumPapers*, first calls the function *GetNumPapersSubset* and then filters out those proteins under the selected minimum. We then count and save the total number of proteins included in the subset, before grouping by the paper and year in which proteins were identified, counting the number of proteins in each paper and dividing by the total to obtain a percentage. To replicate the plot exactly as it is present in the original paper, we add a column called 'stat', and assign the value 'Found' to all percentages. We then duplicate the data and mutate the percentage in the duplicated data to be $1 - percentage$ to obtain the 'stat' 'NotFound'. We then input these data into a bar chart, recolour the bars to match the yellow and purple colour scheme found in the original paper and order the bars by the year column.

The third plot, 1E and 1F, contains exactly the same parameters as plot 1C/1D. We again call *SubsetNumPapers*, then arrange by year and remove all entries which are not the first occurrence of a specific protein. We then count the number of proteins first identified in each year, and fill the list of years with 0s for years where no studies identified new proteins. If we inspect figure 3.2, it seems like this last step was neglected in the original paper. Looking at the years 2001 and 2003 specifically, we can see clearly from the plot on the left that no studies identified new proteins in these years. The plot on the right however seems to imply that there were a non-zero number of proteins identified in these years, as the plot connects the values in 2000 and 2004 with a straight line. Finally, we calculate the cumulative sum of new proteins to get the data coloured in blue. These data are then visualised in a $ggplot2$ area plot. The authors of the paper indicated that, given a chance to redo the analysis, they would have changed this plot

from showing years on the x-axis to individual papers. While this seems like a useful improvement, the aim of the ERP is to replicate the figures as closely as possible to their presentation in the original paper, so we stuck with showing years on the x-axis.

For plot 1G, we can use the same data generator function as in plots 1E/1F, as the data points in the blue area plot correspond exactly to the points in 1G, except that we set the default evidence level to 'included in at least 2 papers', to only visualise consensus proteins. As with plots 1C/1D and 1E/1F, this level can be set at 1 by the user, which would result in a fit over all proteins, not just consensus proteins. In this plot we also add an additional parameter option for the type of fit. In an RMarkdown document used to generate the plot for the original paper, shared with me by the lead author, three different fits are attempted. These fits are: linear, the logistic curve found in the original paper, and a similar, more generic, sigmoid curve. In the documentation of the file, it is highlighted that this sigmoid curve is negative in the year 2000 and since it is not possible for negative numbers of proteins to have been identified, this fit was not included in the ERP. We did however include the linear fit, using the $ggplot2$ function *geom_smooth* with the method set to 'lm'. We then calculate the R-squared error for the fit, plot the data points and the fit, and overlay the error on the plot. The inclusion of the linear model was a justifiable decision, as evidenced by high R-squared values for fits over all proteins in each sub-cellular localisation, which are consistently above 0.8 (where 1.0 represents a perfect fit). When the sigmoid curve is chosen, we again use *geom_smooth* to calculate the fit, this time using the method 'nls', or non-linear least squares, and pass the *SSLogis* function as our formula, which gives the logistic curve. From this we extract the asymptote, which we round to a whole number. Again, we plot the datapoints and our fit, extend the x-axis to the year 2050, as in the original paper, and overlay the plot with text indicating the total predicted consensus proteins, the asymptote of the logistic curve. If the parameters chosen result in only one datapoint, more specifically that the specific subset of proteins chosen were only identified in studies in a single year, a logistic fit cannot be achieved, and this is indicated to the user in a similar fashion to the empty plot mentioned in section 3.2.1.

Finally, we look at plot 1H. The specific plot used to create the Venn diagram-like figure is an 'UpSet' plot, found in the R library, $UpSetR$. As the plot compares numbers of proteins between sub-cellular localisations, it is not included as a selectable parameter for this plot. We do, however, still include the other three standard parameters, namely brain region, sequencing method, and evidence level. To generate the data for this plot, we call *SubsetNumPapers* as usual, except that we require the localisation to be set to

'All'. We then filter by localisation to get the total sets of proteins in each localisation. Using a combination of the functions $intersect$ and $setdiff$, we then find those sets of proteins included in all localisations, those only found in a specific localisation, and those overlapping over specific localisations. Finally, we count the number of proteins in each set and input this information into the UpSet plot, setting the colours for each sub-cellular compartment to those found in the original static plot. If only one of the sets contains non-zero numbers of proteins, we again indicate this to the user, as an UpSet plot requires at least two sets to be able to run properly.

3.2.3 Figure 3

Since the analysis in figure 3 is quite complex, and much of it was completed using the graph visualisation platform 'Gephi', some of the authors of 'A unified resource' are developing a custom R package called *AnNet*, or 'Analysis of Networks' [16]. While it was originally planned simply to install the package, as the package is currently under development and requires installation of a number of non-standard BioConductor and GitHub packages, some of which are not compatible with older versions of R, having to install the packages onto a hosting site or on a user's own computer may cause issues with reproducibility. Even on my computer, certain difficulties presented themselves: at the time of beginning to program figure 3, the AnNet package required R version 4.0 or higher to run, which presented an issue as these versions of R are buggy in respect to running Shiny apps. It was therefore decided to simply copy those functions which are necessary for replicating figure 3 into an additional R file, currently named 'AnNetFuncs.R'. The contents of this file are then loaded in to the app using the *source* command.

Figure 3C is the most computationally intensive of the plots to calculate from scratch, with some individual steps in the process taking tens of minutes. As a result, it was decided that some of the data should be saved rather than re-calculated live each time the user changes parameters. While this can definitely harm the reproducibility aspect of the ERP, as a user reading the source code can no longer fully comprehend how the data have been generated, we would still consider this approach preferable to requiring a user to wait half an hour to change a single parameter in plot 3C. Nevertheless, if a user would like to replicate the process for generating the saved data this is still possible, as the developers of the *AnNet* package provide a vignette, containing all the steps required to generate the data we saved. Specifically, we save the network graph of each

of the three sub-cellular localisations as GML, or Graph Modeling Language, files, containing information on proteins' community membership using different clustering algorithms, centrality measure values, as well as their disease associations. These graph files, which are used by all plots in figure 3, are loaded into R using the package *igraph*, which is also used to perform further analysis on the graphs. We additionally save the consensus matrices (conmat), which are required to calculate bridgeness. As these are very large files, containing entries equal to the number of proteins in a network squared, these are saved as R data (.RDS) files. While the files were originally all loaded at runtime, we decided to only load the specific conmats as they were required and remove from memory previously loaded conmats, as, if all conmats are loaded the app used over 2 gigabytes of memory, well above the 1 gigabyte provided by a free ShinyApps account. The data can then be generated simply by accessing the required network graph, loading the required conmat, and calling the *AnNet* function *getBridgeness* with the graph, conmat and specific clustering algorithm as input.

Figure 3C allows a user to specify four parameters for the plot:

1. Sub-cellular localisation, again these include 'Postsynaptic', 'Presynaptic' and 'Synaptosome'.

2. Different clustering methods, we provide options to choose from Louvain, InfoMAP, Fast-Greedy, and Walktrap. The original paper used spectral clustering, which we do not provide in the ERP, nevertheless, the default option 'Louvain', results in similar bridgeness values to spectral clustering, see figure 3.3.

3. The centrality measure, including 'local centrality' as in the original plot.

4. Which proteins to highlight on the graph. In the static plot from the original paper, the highlighted proteins are hand-curated, while in the ERP we instead allow a user to select a threshold of the minimum number of papers a protein has to have been identified in to be highlighted in the graph.

As calculating bridgeness still takes a noticeable amount of time, usually around 15-30 seconds, and since, when a Shiny app is loaded, all reactive elements need to finish processing before the app is fully loaded, we have added a button to allow the user to specify when to execute this figure, which heavily cuts down on initial loading time. We also move the function loading the graph, conmat, and bridgeness data into a separate reactive function, so that changing centrality measure or which genes to

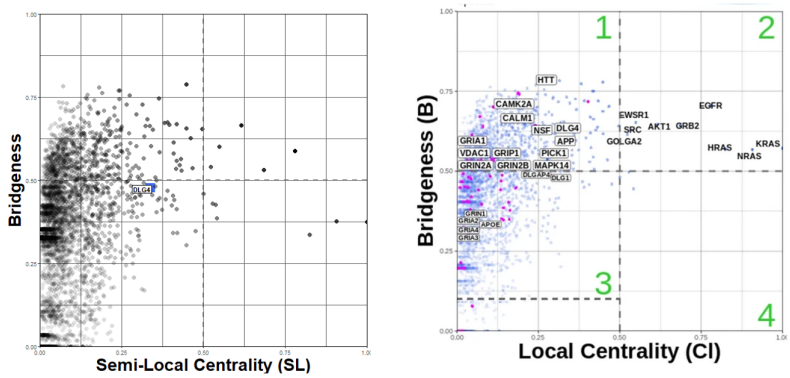

Figure 3.3: Side-by-side of plot 3C from ERP, using Louvain clustering (left), and original paper, using spectral clustering (right)

highlight does not require re-calculation of bridgeness. To generate the actual plot, we first determine which proteins should be highlighted using our previously defined *GetNumPapersSubset* and then plot centrality values saved in the network graph against calculated bridgeness values. The code to generate this plot was also found in the AnNet vignette and minimally changed to allow the plot to react to changing parameters [16]. The plot uses a specific kind of label called *geom_label_repel*, contained in the package *ggrepel*, which minimizes visual overlaps with other labels and other elements in the plot.

In contrast, plot 3D is a lot simpler to compute, and the only changeable parameter is sub-cellular localisation. To generate the correlation matrix, we first populate a matrix with lists of each centrality measure for each protein and then calculate their correlation, using the standard R function *cor*. To generate the plot, we use the *corrplot* function, found in the homonymous package *corrplot*. As slightly different centrality measures were included in the network graphs, the correlation matrix plot also contains different centrality measures, specifically both contain local, degree, betweenness, PageRank, and closeness centrality, while the static plot also contains shortest path, SR up, SR down, count, and CNorm centrality, and the ERP plot contains mean shortest path, and single-destination shortest path centrality.

Generating plot 3E again requires some computation time, so we also save the disease pairs tables as RDS files. Again, this step can be found in the *AnNet* vignette, and if a user already has the graphs saved, they are only required to run one *AnNet* command, *runPermDisease*. To calculate the final q-values in the plot, *runPermDisease* requires the qvalue function from the package *WGCNA*, which we again copied into the 'AnNetFuncs' file from the official repository [11]. For this plot, we provide a number

of changeable parameters:

1. Sub-cellular localisation is again a selectable parameter, though in this plot the user can select more than one localisation at a time, resulting in up to three different coloured sets of points on the plot. We think this parameter adds value to the ERP as including more localisations allows the user to compare between localisations, while removing localisations can improve the readability of an often dense plot. While the original paper showed presynaptic, postsynaptic, and PSPc proteins, ours replaces PSPc proteins with synaptosomal ones.

2. We also allow a user to select which diseases should be included, again allowing multiple diseases at the same time.

3. There is also an option to change the upper x-limit of the plot, so users can 'zoom in' to the left-hand side of the plot, where more q-values overlap.

4. It is also possible to change the q-value significance threshold from its default value of 0.05 to another value, such as 0.01.

5. Finally, users can sort disease-disease pairs by their q-values in descending order, using either the median, mean, max, or min. The authors of 'A unified resource' indicated that it is not possible to determine exactly how the pairs are ordered in the static plot, so this seemed like a simple, but valuable addition to the ERP.

To generate the data for 3E, we first load those disease pair tables, which are included in the user's selection. We then filter out disease-disease pairs which compare the same disease to itself, as their q-values are all equal to 1. Next, we map disease ids to their associated abbreviations. We then filter out those diseases which the user did not select, and calculate the negative base-10 logarithm of the q-values, which are shown in the final plot. The plot is then created using a $ggplot2$ scatterplot, in which we reorder the y-axis by the selected q-value measure, set the upper x-limit to the user-defined value and add a vertical dashed line at the selected significance threshold. The plot is quite tall when all diseases are included, but includes a lot of blank space if only two or three diseases are shown. As a result, we use a Shiny reactive UI function to make the height of the plot a function of the number of included diseases.

As plot 3F was originally created in GEPHI, our implementation looks quite different to the one in the original paper, see figure 3.4. Additionally, to test for which clusters are over-represented for proteins associated with a specific disease, we add an extra table,

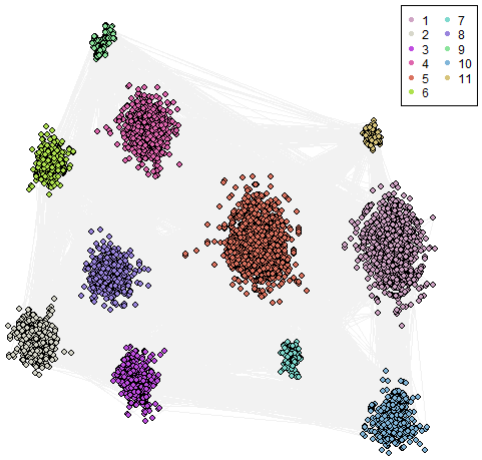

Figure 3.4: Plot 3F, for postsynaptic proteins and Louvain clustering, with clusters highlighted by colour

containing the p-values for each cluster being enriched for a given disease. The plot and table have two possible states: by default, no disease is selected, and the proteins in the plot are coloured by cluster membership, while the table reads 'No disease selected'. If the user then selects a disease, the proteins are coloured blue if they are associated with the disease, or red if they are not. The table then shows a list of clusters, the number of disease-associated proteins and the total number of proteins, as well as the p-value. To highlight rows containing p-values below 0.05 and 0.01, we colour those rows using the library *DT* (DataTable). Shiny datatables provide some additional interactivity in that the user can sort by a given column, for example by ascending p-value, or search for a specific string or number.

To generate the data for plot 3F, we simply make a data frame out of the proteins and their membership for a specific clustering algorithm, adding a Boolean column if a disease is selected. To access the names, memberships and disease annotations of each protein, we use *igraph*. To plot the data, we define a colour palette, using the function *distinctColorPalette* from the R library *randomcoloR*. We also define the spatial layout of the protein clusters using the *AnNet* function *layoutByCluster*, making sure to only call this when the localisation and clustering method change, so that changing between diseases does not lead to a new layout and it is easier for the user to compare between different diseases. We then define the colour palette, using our pre-defined distinct colour palette if no disease is selected, or red and blue if a disease is selected. Finally, we use an *igraph* plot to show the clustered proteins, placing a legend

in the top right of the plot.

For the table 3F, we use the previously defined membership data frame, and calculate a hypergeometric test using the following formula, supplied by the lead author of the paper:

$$p.val = min(1, cumsum(dhyper(0:(a-1), b, (c-b), d))), \tag{3.1}$$

where a is the number of disease-associated proteins in a given cluster, and d is the total number of proteins in the cluster, b is the number of disease-associated proteins in the whole graph network, and c is the total number of proteins in the network. Finally, the data is input into the table, colouring rows by p-value.

3.2.4 Additional Features

Aside from the interactive plots, two additional features were added, which it is believed add to the overall value of the ERP.

The first is an implementation of Table 1 in the original paper, which contains a list of the 58 studies used to fill the SQLite database. It has 5 columns, specifying the study name and its associated citation, the number of genes the study identified, the sub-cellular compartment (localisation) the study looked at, the brain region(s), and the species studied. Rows in the table are colour-coded by the sub-cellular localisation studied. To replicate the table in the R app, we simply copy the columns into a data frame, and convert it into a *DT* datatable, and then use *DT* functionality to colour the rows as in the original table. This approach was preferred to simply copying the png files of the table from the original paper, since, as already mentioned, a data table in RShiny allows the user to sort the table by a specific row, and to filter the entries by a given string, such as the species studied. How much added value this table actually adds to the ERP will be determined in section 4.2.

The second added feature is a dataset download button, located below the abstract and above the introduction of the paper. It was at one point also planned to add a button to download the plots themselves as png files, however this feature was later removed for multiple reasons. One was that it can be difficult to download plots not created through *ggplot*2, for example the UpSet plot, and it would lower the consistency of the app if some plots can be downloaded while others cannot. It was also pointed out by my supervisor that users can already download the plots, simply by right-clicking and saving the image. Alternatively, a user could download the source code and save the

image themselves if, for some reason, they required the plot in full resolution.

The dataset downloader, in contrast, does have the potential to provide value to the user, as it can help to facilitate further downstream analysis. Using a reactive switch statement, the downloader accesses the data generator functions with the current user-selected parameters, so a user could easily download for example the accumulation of new proteins in figure 1E/1F separately for the methods 'target' and 'shotgun' with just a few clicks. We provide a list the datasets that are downloaded for each figure:

1. For plot 1A/1B: the number of studies proteins have been identified in, and how many proteins were identified with that frequency.

2. For plot 1C/1D: a list of papers, with the year the study was completed in, and the percentage of known proteins identified / not identified in that study.

3. For plot 1E/1F & plot 1G: a list of years, and the number of new proteins identified in the year, along with the total number of proteins identified up to that year.

4. For plot 1H: a list of localisations, including overlaps of localisations, and the number of proteins in each category.

5. For plot 3C: for this plot we provide three different options, including: 1) the network graph as an *igraph* object, 2) the consensus matrix, which is provided as an RDS file due to its size, and 3) a list of protein names and entrez ids, and their associated bridgeness values.

6. For plot 3D: the correlation matrix between different centrality measures.

7. For plot 3E: a list of disease pairs, for each selected localisation, their q-value, and their negative log-10 q-value.

8. For plot 3F: a list of proteins, their cluster number, and, if a disease is currently selected, whether each protein is associated with that disease (either TRUE or FALSE).

9. For table 3F: a list of clusters, the number of disease-associated proteins and total number of proteins in the cluster, as well as the p-value indicating how likely it is that each cluster is enriched for a given disease.

RShiny provides a very useful *downloadHander* function, through which we are able to specify the filename and content that will be downloaded. For most of the plots,

we simply concatenate together 'data', the plot number, and '.csv' to obtain the filename, where for the network graph, we replace '.csv' with '.gml' and with '.RDS' for the consensus matrix. For the content function, we call *write.csv* with the output from the reactive switch statement, for all datasets, except the graph and consensus matrix, for which we call *write_graph* and *saveRDS* respectively. In the next section, we provide some examples of downstream analysis that can be accomplished using the downloaded datasets, to further motivate the inclusion of the download button.

3.3 Examples of Further Analysis

3.3.1 Relationship Between Number of Identified Proteins and Potential False Positives

For our first example of downstream analysis made possible by the data download function, we download the data corresponding to Plot 1E, with standard parameters: Localisation = Postsynaptic, Brain Region = All, Method = All, and Evidence Level = 1, as well as the same data for PSPc proteins, Evidence Level = 2. For this analysis we treat proteins that were only identified once as false positives. We chose the postsynaptic localisation as it has the largest number of associated studies. We are interested in two related questions: whether there is a statistically quantifiable association between false positive protein hits and the number of proteins identified in a year, as well as with progressing years.

Listing 3.1: Code to plot figure 3.5 and calculate correlations

```
library(dplyr)
library(ggplot2)

psp <- read.csv("data1e,1f.csv")
pspc <- read.csv("data1e,1f,cons.csv") %>% rename("TotalGenes_c" = "TotalGenes")
post_identifications <- merge(psp, pspc, by="Year") %>% mutate(Diff = (TotalGenes -
    TotalGenes_c)/TotalGenes)

cor(post_identifications$Year, post_identifications$Diff)
cor(post_identifications$TotalGenes, post_identifications$Diff)

ggplot(post_identifications, aes(x=Year, y=Diff)) + geom_point() + ylab("Fraction_of_
    potential_false_positives") +
  geom_smooth(method="lm") + ggtitle("Fraction_of_potential_false_positives_by_year")
      + theme(plot.title = element_text(hjust = 0.5))
```

```
ggplot(post_identifications, aes(x=Diff, y=TotalGenes)) + geom_point() + xlab("
    Fraction of potential false positives") +
  ylab("Total number of proteins identified in a year") + geom_smooth(method="lm") +
  ggtitle("Total proteins identified versus fraction of potential false positives in
      a year") + theme(plot.title = element_text(hjust = 0.5))
```

As we can see in the code in listing 3.1, performing this analysis is possible in only a few lines of R code, using the downloaded datasets. We begin by loading standard libraries, *dplyr* and *ggplot2*. We then read in both of the datasets, renaming the TotalGenes column in the PSPc dataset to indicate they are values for consensus proteins. We can then merge the two datasets, and calculate the fraction of potential false positives (Diff). At this point we are already ready to calculate the correlation between years and the number of potential false positives, a value of 0.946, and between the number of total genes identified and potential false positives, equal to 0.971. Both values indicate that there is some merit to our hypothesis that more false positives were identified in later studies, as well as in studies that identified a greater number of total proteins. We can also see similar relationships by visualising the data as scatterplots, see figure 3.5. At this point, we might recognize that the relationship between years and fraction of false positives is down to the fact that more total proteins were identified in later years, but the purpose of this section is merely to highlight potential further analyses, not to perform rigorous statistical analysis.

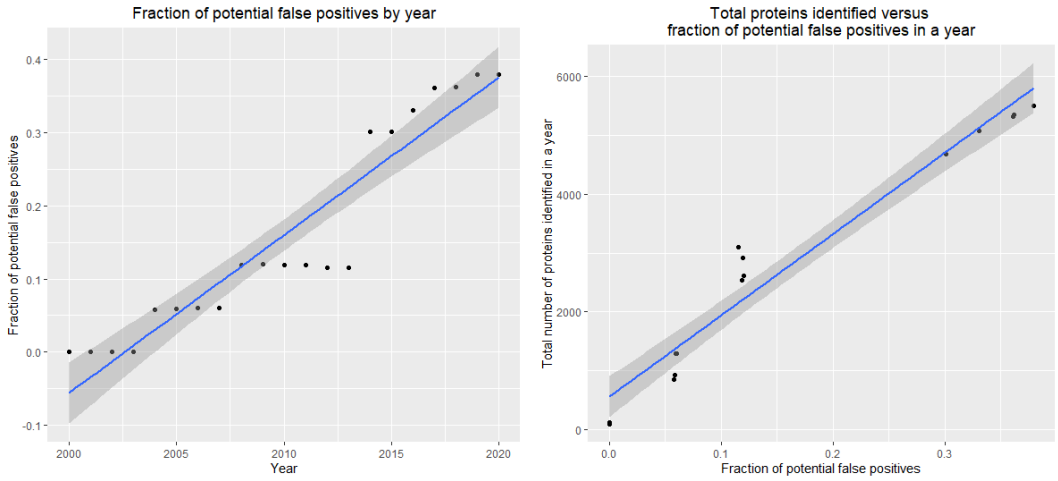

Figure 3.5: Example of plots from downstream analysis using data downloaded from plot 1E/1F

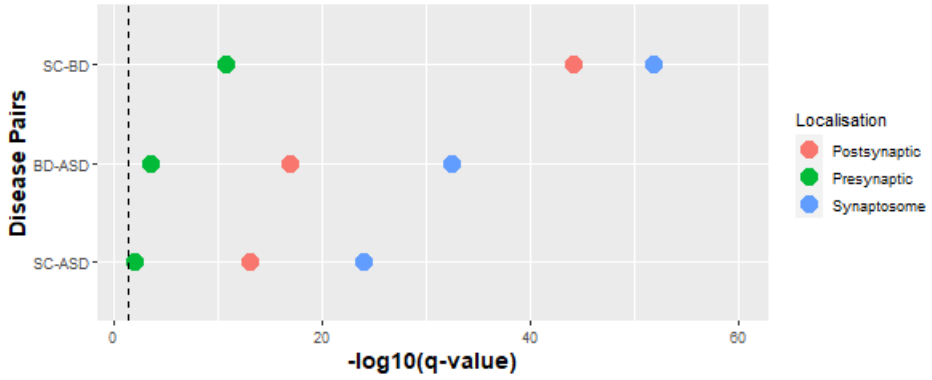

Figure 3.6: Most significant disease pairs from plot 3E

3.3.2 Determining Proteins Associated with Multiple Diseases

As figure 3 visualises more complex analysis than figure 1, it also has the potential to allow for more interesting downstream analysis. One potential analysis could be looking at how consistent correlations between centrality measures in plot 3D are across different localisations. Instead, we are interested in finding a list of proteins, which are highly associated with multiple diseases. If we look at figure 3.6, we can see that schizophrenia, autism spectrum disorder, and bipolar disorder are very significantly associated with similar proteins, especially for proteins identified in synaptosomal studies. In the app we can then take a look at the overrepresentation analysis in table 3F, for synaptosomal proteins, using Louvain clustering, which we choose arbitrarily, and see that cluster 1 is significantly overrepresented for all three chosen diseases. To figure out which proteins in cluster 1 are associated with all three diseases, we download the data from plot 3F, for all three diseases.

As can be seen in listing 3.2, this analysis can also be performed in very few steps. First, we load the *dplyr* library. We then import our downloaded datasets to R, renaming the disease column to the specific disease in each data frame. At this point all we need to do is merge the three data frames into one, and filter by membership and disease columns. Finally, we print out the protein ids. This list of ids can be converted into protein names. Our final list contains 13 proteins: CTNNB1, NCAM1, NTRK2, GRIN2B, CADM1, GRIN2A, GAD1, NOS1AP, ITIH4, SH3PXD2A, GRIP1, STT3A, RAB39A. A bioinformatician could then use this list to, for example, determine what these proteins have in common.

Listing 3.2: Code to determine proteins associated with ASD, SC & BD in synaptosomal Lovuain cluster 1

```
library(dplyr)

asd <- read.csv("data3f_asd.csv") %>% rename("asd" = "disease")
bd <- read.csv("data3f_bd.csv") %>% rename("bd" = "disease")
sc <- read.csv("data3f_sc.csv") %>% rename("sc" = "disease")

diseases <- merge(merge(asd, bd), sc) %>% filter(membership == "1" & asd & bd & sc)
print(diseases$names)
```

Having given two potential analyses that can be carried out, we believe we have sufficiently demonstrated the value of the download button, as well as combining information from multiple plots together can better help our understanding of the synaptic dataset.

Chapter 4

Evaluation

4.1 Setup

To determine how well-executed and useful the ERP actually is to researchers, a number of researchers with an interest in the synaptic proteome, including the authors of 'A unified resource', were contacted and asked for feedback. Researchers were instructed to access the app on its current shinyapps.io domain, complete some suggested interactions with the app, to familiarize themselves with the contents of the ERP, and given a list of questions to answer. In our feedback request we attempted to maximize the number of potential responses by keeping the questionnaire as concise as possible, as well as clearly stating the purpose of the evaluation.

We provide here the list of suggested instructions for interacting with the app:

- Fig 1a/b: Change Sub-cellular Localisation from *postsynaptic* to *presynaptic* or *synaptosome*
- Fig 1c/d: Increase Evidence from *one* to *two* or more papers
- Fig 1e/f: Change Sub-cellular Localisation from *Postsynaptic* to *All* and Brain Region from *All* to *Forebrain*
- Fig 1g: Change Evidence to different levels
- Fig 1h: Change Method from *All* to *Shotgun* or *Target*
- Fig 3c: Click button to generate Figure 3c (takes 15-20s)
- Fig 3c: Change Clustering Method to a method other than *Louvain* clustering (and click to regenerate)

- Fig 3d: Switch between Sub-cellular Localisations
- Fig 3e: Remove *AD*, *ASD*, and *MS* from Diseases list and change q-value threshold to *0.01*
- Fig 3f: Inspect a selected Localisation and clustering method for over-representation of *Huntington's disease*

We also provide the list of questions:

1. Did you find that interacting with the figures was straightforward, or were there any issues you came across?
2. Do the executable figures add to your understanding of the data presented, especially as compared to the original paper?
3. Is the layout of the paper visually appealing, or are there elements which hinder your comprehension of the material?
4. Does the interactive table (Table 1) add value to the paper, as compared to a static printed table?
5. Choose the most interesting figure to you and download the dataset for that figure using the download button at the top of the page. Can you think of a way to perform additional analyses on the data you downloaded? If so, please give an example.
6. Is there anything you would add to the paper to improve its understandability or capacity for delivering information?
7. Based on your previous responses, would you consider developing an executable research paper in the future?
8. Do you think an executable research paper, such as this, can improve reproducibility of research?
9. Please give an overall rating out of 10 for the quality of the paper (where 10 is the best)

When writing the questions, we were interested primarily in finding out how useable and extensive our app is specifically, but also wanted to discover potential improvements

to the app, as well as what researchers' opinions on ERPs were more generally. The feedback questions were ordered in such a fashion as to move thematically from general questions about the quality of the ERP, then to more specific questions, and finally to questions about the concept of ERPs as a whole. The more important questions for the evaluation, those on the actual quality of the product, were asked towards the beginning of the questionnaire, as suggested by research on questionnaire design [9]. Asking more general and conceptual questions at the end also means that the respondent is likely to have though more deeply about ERPs by the time they have arrived at the final questions. In designing the feedback, we also followed other well-documented guidelines for questionnaires, including avoiding asking multi-barreled questions, or questions which ask multiple questions at the same time, as well as avoiding negative language, or confusing wordings such as double negatives [9]. We also wanted to avoid respondents defaulting to more positive responses, to avoid acquiesence bias, by providing both positive and negative options in the question formulation, such as in question 1. Overall, we were happy with the responses we received, with researchers often providing in-depth answers to questions that could be perceived as yes/no questions, such as question 8.

4.2 Results

Overall, 6 researchers responded to our requests for feedback, with 4 being authors of 'A unified resource'. Their responses were generally relatively extensive and very positive, usually indicating that the ERP added to their understanding of the synaptic proteome, and that they would consider developing an ERP themselves in the future. We provide here a summary of responses:

1. All respondents indicated that interacting with the figures in the app was straightforward. Two researchers did highlight minor issues: the first highlighted that changing clustering methods to Walktrap and Fast-Greedy caused memory issues, particularly in plot 3F, and led the shinyapps server to disconnect. It is yet to be seen if these issues would occur on EBRAINS, as they may allow for higher memory usage on their page. The second indicated that they were confused by the download button, mistakenly assuming its use was required to execute subsequent plots. We will return to the exact placement of the download button later, which was highlighted by other researchers in later responses. The second respondent also indicated that they were confused by the inclusion of the original

static figures, but also mentioned that they had not read the surrounding text.

2. Respondents generally indicated that the executable plots did add to their understanding of the data presented. Three researchers specifically called attention to the value added by being able to change parameters and ”explore the data more dynamically”, with one of these researchers highlighting that statistical thresholds and other methods can often be ”chosen arbitrarily” in a paper, so it is interesting to see to what extent varying these choices can lead to different results. One of the researchers indicated that they did not see much value in the executable versions of the first figure, but that figure 3 was more useful. Obviously, researchers will differ in what in what they consider useful for their research, but since the third figure is more complex it makes sense that this one would provide more value.

3. Five of the surveyed researchers explicitly agreed that the layout was logical and did not cause issues to their understanding. Nevertheless, one respondent suggested that they would personally prefer to see the layout of the interactive plots correspond exactly to the layout of the static plots or alternatively to provide a tab to switch between figures, rather than having them laid out sequentially. While this has the potential to look more visually consistent, forcing both the original and interactive plots to sit side-by-side could lead to readability issues, especially since it has already been established that the ERP should not take up all of the horizontal space on a page. Another respondent indicated that they would like to see the space taken up by plots to be more consistent, with one plot (3C) currently bleeding out of the right margin. Plot 3C was originally enlarged to ensure readability, but its size should have been reduced slightly when margins were added. Reducing the width of the plot is an easy fix and will definitely be implemented in the final version of the ERP.

4. Again five out of six researchers did deem the addition of the interactive version of Table 1 to improve the paper. Suggestions for improving the table included being able to filter and then export selected rows, as well as being able to select a specific paper to find out which proteins were included in that paper.

5. For the question about performing additional analysis using the download button, one researcher did not provide a response, while another suggested they were not experienced enough in the field to provide a suggestion of further analysis, but that answering this question helped them to understand the purpose of the

download button, and that it is a feature that should be more widely included in publications. Another respondent indicated that plot 3F was the most interesting to them, but they were not sure what additional analysis could be performed. The three other respondents provided examples of further analysis: the first was to download datasets for all three localisations and then combine them into one plot; the second appreciated being able to download the graph file in plot 3C and suggested this could then be imported into a network graph service, such as Cytoscape or iGraph, to perform further analysis; the third downloaded bridging proteins from plot 3C and would complete ”additional enrichment analysis on the identified proteins”.

6. Four of six respondents suggested improvements to plot 3F would be good to see: the main improvement requested was to add more interactivity to plot 3F, in the form of being able to zoom in on vertices, being able to hover over specific vertices and receive information about which protein they represented, which cluster they belonged to, and whether they were associated with a chosen disease. Two respondents also gave potential improvements to the download functionality, one researcher suggested moving the download capabilities directly into the selectors. This also seems like a sensible way to make it clearer to a user what exactly the download button does. The same researcher also indicated they would like the download button to download the specific subset of data currently viewed in the plot. This is already how the download button works, but moving the buttons directly next to the plots could potentially make this clearer to the user. The other researcher suggested, in table 3F, they would like to be able to download a list of proteins by their cluster membership. Again, this feature is already enabled, by downloading the data in plot 3F, but should be more well labelled so that it is not as easy for the user to miss this capability. Another researcher, who had previously indicated their issues with Walktrap and Fast-Greedy clustering, suggested replacing these clustering options with a different kind of clustering, called Leiden clustering. A different respondent suggested adding more datasets so that brain regions are covered more accurately, but added that this was an issue better adressed by the database provided by the original paper. Finally, one researcher indicated they had no ideas yet on how to improve the ERP.

7. Five out of six respondents would consider developing an ERP of their own

in the future, while one would not consider it. Of those that indicated they would consider creating an ERP in the future, one suggested that ERPs should be more commonly used, while another wanted to see journals provide a user-friendly method to publish ERPs. This researcher also asked specifically for recommendations on how to get started developing an ERP, implying that there is a real interest in furthering the cause of ERPs among computational researchers, though this interest may be hampered by barriers to entry, which journals should aim to address.

8. When asked about whether an ERP in our chosen format could improve reproducibility in computational research, five believed ERPs could do so, with two respondents highlighting the value of being able to easily access data through the download button. The sixth researcher suggested the improvement in reproducibility was only marginal, and that they consider adding code blocks to be more useful.

9. Four researchers gave the ERP a score of 8 out of 10, one gave a score of 9, suggesting that there were still minor issues that could be addressed, while another gave it a 12, though this was based on representation, as the respondent did not consider their background sufficient to judge the content of the ERP. Normalising the 12 to a 10, we end up with a final median score of 8, and a mean of 8.5, suggesting the ERP is above average in quality.

4.3 Final Steps

Having summarised the feedback from researchers, we conclude by giving an overview of the steps required to finalise the ERP development process:

1. Reduce width of plot 3C, to ensure consistency with the width of the whole app.

2. Replace download button at the start of the document with individual download buttons accompanying each of the plots. Also label these, so users are aware of what datasets it is possible to download.

3. Add a feature to be able to download, or otherwise access the proteins identified in each of the studies in table 1.

4. Add Leiden clustering as an option, potentially removing Fast-Greedy and Walktrap if similar memory issues occur when hosted on EBRAINS.

5. Add further interactivity to plot 3F. The R package *plotly* includes all of the features requested, including being able to zoom in on the plot and hover over and label vertices.

6. Convert the RShiny app to a Shiny RMarkdown document, to allow inclusion of code blocks in the app. An additional benefit to this change will be that it will make it easier to highlight to the user how to regenerate from scratch the data files we have chosen to save, to reduce on loading times, improving the completeness of the ERP.

7. Once all of these changes have been implemented, the source code and additional data files should be uploaded to Edinburgh DataShare to allow researchers to easily access them.

8. Work out the last issues with hosting the Shiny ERP through OpenShift on EBRAINS.

Chapter 5

Conclusion

In this dissertation, we have walked the reader through the design of a specific ERP in bioinformatics. From this experience we can conclude that RShiny is a valid approach to developing an ERP, with Shiny RMarkdown documents probably being preferable to Shiny apps, due to the fact that they allow for inclusion of code blocks in the rendered paper. Additionally, the RStudio team has recently announced that they are developing Shiny for Python, increasing the viability of Shiny for ERPs beyond R programmers [13]. Being able to interact with parameters to easily visualise how this affects the results of analysis was generally considered a worthwhile addition to the research. The inclusion of the download button was a feature that was highlighted by multiple respondents as beneficial to replication and furthering of research. Reflecting on my own experience developing the app, I would definitely say that the process has caused me to think more deeply about how to improve replicability, such as by consciously minimizing the number of dependencies required to run a program, as well as more general issues in publications, such as effective data visualisation. Overall, users indicated that the ERP contributed to their further understanding of the analysis in 'A unified resource', with one respondent even indicating that interacting with the app was "fun".

We have also been able to provide further evidence that researchers are largely interested in developing ERPs and that they consider ERPs to be beneficial to the replication of research. Issues which may be holding researchers back include a lack of incentives for doing so, as well as that there is a barrier to entry for those who have never attempted to create an ERP before. These are both issues which journals need to address in the future, along with making hosting ERPs more straightforward, as highlighted by our experience with EBRAINS. To refer back to Claerbout's principle, journals could improve incentives for publishing ERPs easily by placing ERPs on the same level as

more classical articles, for example by counting an ERP as an independent publication. Solving the related issues of hosting and streamlining the development of ERPs is a little more complicated, but one approach could be making Shiny documents easier to host on journal pages, as well as providing guides on getting started. Nevertheless, this is an exciting time for ERPs, as the concept is slowly starting to take off among researchers and journals, and we would strongly recommend any interested researchers to give the development of their own ERP a try.

Bibliography

[1] Interactive resource sheets for computational studies in neuroscience. EBRAINS Live Papers.

[2] Open code. *PLOS*, Aug 2022.

[3] Fred Atherden. Executable research articles. eLife, Jun 2021.

[4] Steven R. Brandt. Where can i publish executable papers and notebooks. Software Sustainability Institute.

[5] Open Science Collaboration. Estimating the reproducibility of psychological science. *Science*, 349(6251):aac4716, 2015.

[6] Jan de Leeuw. Reproducible research. the bottom line. *UCLA: Department of Statistics, UCLA*, Mar 2001.

[7] Laura Fortunato and Galassi Mark. The case for free and open source software in research and scholarship. *Philosophical Transactions of the Royal Society*, 379(2197), Mar 2021.

[8] Ann Gabriel and Rebecca Capone. Executable paper grand challenge workshop. *Procedia Computer Science*, 4:577–578, 2011. Proceedings of the International Conference on Computational Science, ICCS 2011.

[9] Hunter Gehlbach and Anthony R. Artino. The survey checklist (manifesto). *Academic Medicine*, 93(3):360–366, 2018.

[10] Mikael Laakso, Patrik Welling, Helena Bukvova, Linus Nyman, Bo-Christer Björk, and Turid Hedlund. The development of open access journal publishing from 1993 to 2009. *PloS one*, 6(6):e20961, 2011.

[11] Peter Langfelder and Steve Horvath. Wgcna: An r package for weighted correlation network analysis. *BMC Bioinformatics*, 9(559), Dec 2008.

[12] Jana Lasser. Creating an executable paper is a journey through open science. *Communications Physics*, 3(143), Aug 2020.

[13] Sharon Machlis. Rstudio unveils shiny for python. *InfoWorld*, Jul 2022.

[14] Emil Bargmann Madsen, Mathias Wullum Nielsen, Josefine Bjørnholm, Reshma Jagsi, and Jens Peter Andersen. Meta-research: Author-level data confirm the widening gender gap in publishing rates during covid-19. *eLife*, 11:e76559, Mar 2022.

[15] Gary Marcus. The crisis in social psychology that isn't. The New Yorker, May 2013.

[16] Colin Mclean, Anatoly Sorokin, Oksana Sorokina, and J. Douglas Armstrong. *AnNet: Analysis of Networks*, 2022. R package version 0.0.0.9002.

[17] Peter Murray-Rust. Open data in science. *Nature Precedings*, Jan 2008.

[18] Tamás Nepusz, Andrea Petróczi, László Négyessy, and Fülöp Bazsó. Fuzzy communities and the concept of bridgeness in complex networks. *Physical review. E, Statistical, nonlinear, and soft matter physics*, 77:016107, Feb 2008.

[19] Daniel Nüst, Markus Konkol, Edzer J. Pebesma, Christian Kray, Marc Schutzeichel, Holger Przibytzin, and Jörg Lorenz. Opening the publication process with executable research compendia. *D Lib Mag.*, 23, 2017.

[20] Oksana Sorokina. Synaptic proteome sqlite database, 2000-2020 [dataset]. *University of Edinburgh. School of Informatics. Systems Neuroscience group*, 2021.

[21] Oksana Sorokina, Colin Mclean, Mike D. R. Croning, Katharina F. Heil, Emilia Wysocka, Xin He, David Sterratt, Seth G. N. Grant, T. Ian Simpson, and J. Douglas Armstrong. A configurable model of the synaptic proteome reveals the molecular mechanisms of disease co-morbidity. Published, October 2020.

[22] Oksana Sorokina, Colin Mclean, Mike D. R. Croning, Katharina F. Heil, Emilia Wysocka, Xin He, David Sterratt, Seth G. N. Grant, T. Ian Simpson, and J. Douglas Armstrong. A unified resource and configurable model of the synapse proteome and its role in disease. *Scientific Reports*, 11(9967), 2021.

[23] John D. Storey and Robert Tibshirani. Statistical significance for genomewide studies. *Proceedings of the National Academy of Sciences*, 100(16):9440–9445, 2003.

[24] Simon Traugust, Matt Nolan, James Watson, Karl Huang, Matteo Mancini, and Sophie de Buyl. elife authors relay their experiences with executable research articles. eLife, Jun 2021.

[25] Ana Trisovic, Matthew K. Lau, Thomas Pasquier, and Mercè Crosas. A large-scale study on research code quality and execution. *Scientific Data*, 9, Feb 2022.

[26] Emmy Tsang and Giuliano Maciocci. Welcome to a new era of reproducible publishing. eLife, Aug 2020.

[27] Ruben Vicente-Saez and Clara Martinez-Fuentes. Open science now: A systematic literature review for an integrated definition. *Journal of Business Research*, 88:428–436, 2018.

[28] Junlong Zhang and Yu Luo. Degree centrality, betweenness centrality, and closeness centrality in social network. In *Proceedings of the 2017 2nd International Conference on Modelling, Simulation and Applied Mathematics (MSAM2017)*, pages 300–303. Atlantis Press, Mar 2017.

Appendix A

Participants' information sheet

Participant Information Sheet

Project title:	Executable Research Papers in Bioinformatics
Principal investigator:	Dr Douglas Armstrong
Researcher collecting data:	Bernhard Finke
Funder (if applicable):	None

This study was certified according to the Informatics Research Ethics Process, RT number 439060. Please take time to read the following information carefully. You should keep this page for your records.

Who are the researchers?

Bernhard Finke – MSc student at University of Edinburgh

Dr Douglas Armstrong – Professor in Systems Neurobiology at University of Edinburgh and Supervisor of MSc student

What is the purpose of the study?

To gauge the potential value to researchers of the executable research paper based on the figures in 'A unified resource and configurable model of the synapse proteome and its role in disease'.

Why have I been asked to take part?

As you have been identified as a researcher with an interest in the synaptic proteome, we believe you could provide valuable feedback on the executable paper.

Do I have to take part?

No – participation in this study is entirely up to you. You can withdraw from the study at any time, up until the 10th of August, 2022 without giving a reason. After this point, personal data will be deleted and anonymised data will be combined such that it is impossible to remove individual information from the analysis. Your rights will not be affected. If you wish to withdraw, contact the PI. We will keep copies of your original consent, and of your withdrawal request.

What will happen if I decide to take part?

All that would be required to participate in the study, is to try out the executable paper and provide as much or as little feedback as you deem appropriate.

Are there any risks associated with taking part?

There are no significant risks associated with participation.

What will happen to the results of this study?

The results of this study may be summarised in published articles, reports and presentations. Quotes or key findings will be anonymized: We will remove any information that could, in our assessment, allow anyone to identify you. With your consent, information can also be used for future research. Your data may be archived for a maximum of 1 month. All potentially identifiable data will be deleted within this timeframe if it has not already been deleted as part of anonymization.

Data protection and confidentiality.

Your data will be processed in accordance with Data Protection Law. All information collected about you will be kept strictly confidential. Your data will be referred to by a unique participant number rather than by name. Your data will only be viewed by the researcher/research team (Bernhard Finke and Dr Douglas Armstrong).

All electronic data will be stored on a password-protected encrypted computer, on the School of Informatics' secure file servers, or on the University's secure encrypted cloud storage services (DataShare, ownCloud, or Sharepoint) and all paper records will be stored in a locked filing cabinet in the PI's office. Your consent information will be kept separately from your responses in order to minimise risk.

What are my data protection rights?

The University of Edinburgh is a Data Controller for the information you provide. You have the right to access information held about you. Your right of access can be exercised in accordance Data Protection Law. You also have other rights including rights of correction, erasure and objection. For more details, including the right to lodge a complaint with the Information Commissioner's Office, please visit www.ico.org.uk. Questions, comments and requests about your personal data can also be sent to the University Data Protection Officer at dpo@ed.ac.uk.

THE UNIVERSITY of EDINBURGH informatics

Who can I contact?

If you have any further questions about the study, please contact the lead researcher, Bernhard Finke (e-mail: s2255150@ed.ac.uk).

If you wish to make a complaint about the study, please contact inf-ethics@inf.ed.ac.uk. When you contact us, please provide the study title and detail the nature of your complaint.

Updated information.

If the research project changes in any way, an updated Participant Information Sheet will be made available on http://web.inf.ed.ac.uk/infweb/research/study-updates.

Alternative formats.

To request this document in an alternative format, such as large print or on coloured paper, please contact Bernhard Finke (e-mail: s2255150@ed.ac.uk).

General information.

For general information about how we use your data, go to: edin.ac/privacy-research

THE UNIVERSITY of EDINBURGH
informatics

Appendix B

Participants' consent form

Participant number:_____

Participant Consent Form

Project title:	Executable Research Papers in Bioinformatics
Principal investigator (PI):	Dr Douglas Armstrong
Researcher:	Bernhard Finke
PI contact details:	douglas.armstrong@ed.ac.uk

By participating in the study you agree that:

- I have read and understood the Participant Information Sheet for the above study, that I have had the opportunity to ask questions, and that any questions I had were answered to my satisfaction.
- My participation is voluntary, and that I can withdraw at any time without giving a reason. Withdrawing will not affect any of my rights.
- I consent to my anonymised data being used in academic publications and presentations.
- I understand that my anonymised data will be stored for the duration outlined in the Participant Information Sheet.

By signing the document below, you agree to take part in the study:

Name of person giving consent	Date	Signature
Name of person taking consent Bernhard Finke	Date 27/07/2022	Signature