chapter{Code-Based Interviews}

This chapter will discuss the results of interviews discussing the opinions of programming specialists in relation to the codes. As previously discussed, two languages were used to create the plots; R and Python. Initially, the JavaScript D3 library was also considered, but discounted due to time constraints.

This part of the study will comprise analysis of a series of 20 minute interviews with 6 programming specialists, with the intention of learning their opinions regarding general opinions on the languages discussed, the plotting libraries used in these languages, and on the specific code used to create the visuals for the survey. The participants are additionally asked for their opinions on the control and logarithmically scaled bar plots from section 1 of the survey, asking them to compare between languages how 'publication ready' each output seems

The six interviewees, Dr. Vincent Knight, Dr. Geraint Palmer, Dr. Andreas Artemiou, Dr. Nikoleta E. Glynatsi, Henry Wilde and Professor Owen Jones, are currently, or have been, researchers and academics with the Cardiff University School of Mathematics.

Note that Dr. Knight is the supervisor for this project and so has a significant familiarity with the project background and design of the study, and so his comments will be considered more as additional points of interest as opposed to being included in a thorough, unbiased analysis. 'Unbiased' here refers to being unbiased towards the project as a whole.

About semi-structured interviews

Background on semi-structured interviews

A semi-structured interview is a form of qualitative data collection involving verbally asking a series of respondents a set of open-ended questions, opening a discussion around the given topic.

The term 'semi-structured' refers to the way in which the interviewer for this kind of interview will prepare a set of questions, but not rigidly stick to them as they may do a structured interview or survey. As explained by @longhurst, this allows the discussion to be opened up in a conversational format, allowing the interviewees to *"explore issues they feel are important"*.

This discussion of "issues" may refer more to interviews assessing, for example, an important social issue or opinions of a workplace. The opening up of the discussion in terms of a more technical-based interview can allow avenues to be traveled down that the interviewer may not otherwise have considered, leading to a wider scope of investigation in addition to the deeper focus on the questions in hand.

In this way, a semi-structured interview allows the interviewer to have some control over the direction of the interview and the topics of discussion while at the same time being guided by the interviewee's responses, allowing the researcher to gain all the information necessary about the given topic in addition to extra thoughts from the respondent.

As with the survey, there are positives and negatives to using this form of data collection.

A large positive of using a semi-structured interview is outlined above; the ability to gain in depth information regarding the topic at hand in addition to widening the scope of the investigation. This face-to-face communication, however, sacrifices the anonymity provided by written surveys. The negative aspects of a lack of anonymity have been explored in a previous chapter, but this is outweighed by the beneficial qualitative information gathered using this method.

An additional difference between this format and a survey is the analysis of results. The quantitative survey, as previously seen, provides empirical quantitative information and so can be analysed numerically using a variety of statistical methods. A survey on the other hand is much harder to quantify empirically and analysis instead focuses on textual analysis of the interviews, much akin to a literature review. As stated by @leech, "Unstructured interviews are best used as a source of insight, not for hypothesis testing".

Prior to administering an interview, the format of the interview and questions must be carefully thought out, the same as with a survey. For example, it could be very easy to unintentionally ask leading questions and guide the discussion in a biased direction.

Additionally, questions must be phrased in a way such that they are targeted towards the wanted information whilst also allowing for open discussion.

When administering the interview, as described by @leech, it is important to keep the interviewee at ease and build a good rapport, and the interviewer should ensure that the respondent can see that they are listening to, interested in and understand what they are saying.

It is stated by @Leech that *"The interviewer should seem professional and generally knowledgeable, but less knowledgeable than the respondent on the particular topic of the interview"*, and this is to make sure the interviewees feel comfortable and 'unthreatened', while also having a mutual respect for the interviewer.

Purpose of a semi-structured interview for this study

A semi-structured interview format was selected for analysis of the programming of visualisation as the programming specialists being interviewed are known to have more experience in programming than the researcher. Thus, the interview participants may have additional viewpoints and knowledge of connected topics within the subject area that have not been considered in the given questions.

This topic is also much more subjective and lends itself much more to long-form open-ended questions than the survey topic.

Interview Design and Implementation

Prior to the interview, questions were formulated in discussion with the project supervisor, Dr. Vincent Knight, who also partook as a respondent to certain questions in a 'test' interview, for which the transcript also contains discussion around the creation of the interview. In addition, Dr. Knight was shown a live version of the plotting codes via screen sharing as opposed to screenshots prior to interview.

The questions were formulated as below, with the intention of wording and tone of voice being the same or at least very similar for every respondent;

- Do you have any particular bias towards either language; R or Python?
- Do you have any initial comments on the codes? For example in terms of readability, or similarities and differences between the two.
- Do you feel that either code could be changed in any way?
- Based on these codes and your own knowledge of programming, how well suited you think each language is to visualisation?
- for a beginner with equal experience in both languages, which library do you feel would be easier to learn, ggplot2 or matplotlib?
- Which package or library do you feel gives the most publication ready output?
- How much freedom do you feel each language or package allows for customisation of plots?
- Do you have other comments that you've thought of during this interview?

These questions are set with the purpose of gaining the respondents' subjective opinions on these topics as well as insights from their own programming expertise.

As discussed, the questions were worded with the intention of not being leading, with the tone of voice also a factor in trying to keep the discussion neutral. In addition, respondents were reminded throughout, as with the survey, that the interview was not a technical test, and rather a conversation regarding their subjective opinions, following comments from Dr. Knight in the test interview stating that *"any conversation with an* *academic, they can sometimes feel like they're tested. So it's always trying to say like, yeah, this is subjective; I'm just interested in your thoughts."*. his can allow for more open answers, as there is less emphasis on stating a 'correct' answer and the respondent may feel more at ease to say whatever comes into their head.

The interviewees were initially contacted in a group email sent out by the project supervisor, after which individual, 1-to-1 Zoom meetings were arranged. In each individual meeting arrangement, the respondent was sent screenshots of some of the visualisation code, consisting of two folders, one for the R code and one for the Python, and were asked to have a brief look over these prior to the interview. The purpose of this was to build respondent familiarity with the codes and give them time to formulate any initial comments or opinions.

The folders consisted of the functions used to create each of the sets of bar plots and the line plots. It was decided to omit comments or docstrings to allow for a purely subjective judgment on the appearance of code itself as opposed to its functionality. The lack of documentation also allows the screenshots to be as readable as possible, since adding the function documentation would take up a significant amount of screen and thus reduce the quality of the screenshots, as mentioned by the project supervisor, Dr. Vincent Knight, in the initial 'test' interview; "maybe, they don't call the strings in R, but a leading overall bit of documentation at the start would be nice. But I understand as well that that could take up quite a fair bit of the screen".

In order to put the codes into a standard format, opinionated code formatters were applied to each, with 'black' [@black] being applied to the Python code and 'styler' [@styler] to the R. Screenshots showing the implementation of these are seen in appendices 6 and 7.

Analysis

The format of the interviews were recorded zoom meetings, for which consent was given by each respondent prior to starting the recordings. After the interviews were completed and recordings obtained, the online transcription software Trint was used to transcribe the interviews for analysis.

These transcriptions can be found in appendix 8, and a question-by-question summary of responses can be found in appendix 9.

Main Findings

Language Biases

The first question regarding language bias is asked to gauge whether any opinions when comparing languages or packages and libraries are influenced by the preferred language. It was found that no respondent was unbiased, with four holding a bias towards Python and the other two towards R, although respondents tended to indicate some degree of familiarity in both.

with Dr. Palmer, Dr. Glynatsi, Dr. Knight and Wilde showing a Python bias, but highlighting some R familiarity;

- "I use Python mostly, but I'm fairly new at R." - Dr. Palmer
- "I am familiar with both languages, but I am very biased towards Python because it's the language I use most." - Dr. Glynatsi
- "So I'm a lot more familiar with Python than I am with R or a lot more fluent in Python than I am with R". - Dr. Knight
- "I mainly use Python, but very recently I've started using R for a project I'm working on." - Wilde

There are varying degrees of R experience highlighted here, but a very strong bias towards Python.

Dr. Artemiou and Professor Jones both stated they have an R bias, but with Dr. Artemiou implying some Python familiarity, saying he is *"much more familiar with R than Python"*; the *"much more"* in this phrase implying a degree of Python familiarity, but notions towards R being heavily preferred.

The question asked to Prof. Jones was potentially leading, with an addition to the question of *"I'm kind of gathering that you probably like R."*. In hindsight, this would not have been asked in this way as it invites bias, although this bias was confirmed; "Yes! I'm an R person. I wrote a book on R, so. . . ", without reference to Python and rather a discussion on the degree to which he is acquainted with ggplot and the tidyverse.

In fact, this respondent went on to mention Hadley Wickham's book on the Tidyverse and graphics, which was then explored and discussed in this report.

It was found that all interviewees' reasoning behind the language biases or preferences were based on the language they use the most, and thus which they are more comfortable in, as opposed to any strong opinions against the less favoured language. It should be noted that programmers do tend to hold certain conscious or subconscious loyalties towards the language they favour, which will be considered when looking at opinions regarding each language, alongside the interviewer's own biases and loyalties towards R.

An insight from Prof. Jones depicts that *"once you've learnt how to do graphics you are happy with, you tend to stick to the system you've learnt because learning a new system takes so long"*, which is reflected here, and also relates to the earlier discussion on choosing visualisation tools; the best tool may in fact be the one that the user is already familiar with. Expanding on this, he then mentions learning the language Julia, which he says uses an import of matplotlib, but also allows R to give Julia output, and so this could be a useful tool for visualisation for both R and Python programmers. It is stated as being designed for numerical work and *"much faster than both R and Python, but still not too hard to use."*.

In future research, Julia could potentially be discussed in a similar way to R and Python have been in this paper.

Comments on the plotting libraries

This section will focus on any general and comments and opinions on the plotting libraries *ggplot2* and matplotlib.

Firstly, thinking about how the libraries function within their respective languages, Prof. Jones mentions how *"ggplot has essentially its own programming language."*, and this correlates with a discussion from Dr. Palmer on the programming styles of the two libraries, whereby he notes that *"ggplot is very object oriented. whereas the rest of R is not"*, which relates to Prof. Jones' comment on ggplot having essentially its own language; it programs using R but doesn't necessarily operate in an R-like fashion.

Dr. Palmer also notes how the pyplot interface of matplotlib functions opposite to this, in the way that it is, as he quotes *"un-pythonic, it's very different to how you write the rest of Python"*, and in fact may appear more similar to R in its more programmatic nature.

This could imply that, for a predominant R user, the pyplot interface could be a good resource for plotting in Python, and similar for a Python user and ggplot. In fact, the reason that the pyplot interface was used for this project was due to it appearing easier than standard matplotlib to the author, and in hindsight this may be due to the author's own bias and familiarity towards R and lack of Python experience.

It is also mentioned by Dr. Palmer that the standard matplotlib without the pyplot interface writes very similar to the rest of python. This could imply then that the standard matplotlib may be a more familiar tool for a Python user.

For example, one Python programmer, Henry Wilde, mentions that he tends to use the non-pyplot method; "the the fig.ax matplotlib thing is something I would do", and then goes on to explain that this is because setting scales, labels and ticks can be performed in a single call with that method as opposed to multiple calls as with pyplot. He does, however, remark that pyplot is *"meant to be more readable"*, and that this could be a reason for using it. This could be down to the Python preference, but also could be unrelated to the language bias and instead just a general preference.

On the pyplot interface, Dr. Knight notes that this interface is *"meant to be relatively straightforward"* since it is designed to be similar to MATLAB's plotter. Therefore this interface may just in general be a fairly simple way to pick up the skill of creating visualisations.

Knight also mentions how, unlike the rest of Python, which is *"unapologetically like, no, no, no, there's no confusion. There's one way to do something"*, pyplot provides a second way to create visualisations, and then discusses how this can be useful as a *"gateway drug"*, although then states how base matplotlib is *"almost harder"* to him when he uses it as a predominant pyplot user.

When asked which would be the best for a beginner to pick up, Knight states how ggplot could be recommended for a beginner, *"not because it's easiest, but because by learning it, you'll not only learn syntax for visualisation, but you'll learn the graphics of visualisation as well"* and *"equips you better than anything else"*. It can be inferred from this that, while extra mental exertion may be required to learn ggplot with the grammar and syntax that goes along with it, it will set a good base for going on to learn other tools as well as general visualisation concepts.

Prof. Jones agrees that ggplot would be the preferred for a beginner to learn and, at least for basic plotting, the easiest, stating that it has *"a nice, logical system for for doing things"*. Jones also mentions, although, that ggplot may be harder when learning to *"do something a bit different"* from the *"standard set of plots that it caters to"* than matplotlib, concluding then that for a beginner producing only simple plots, ggplot could be preferable.

Additionally to this, Dr. Artemiou, after giving a disclaimer that he is *"biased towards R"*, agrees that ggplot appears easier to learn and that this library makes it *"very simple to change the parameters and do whatever you want to"*.

When discussing the easiest library to pick up, Dr. Palmer states his position as *"Probably matplotlib"*, with the reasoning behind this response being that some of the syntax is similar to the rest of Python, and this is the assuming he is referring to base matplotlib. He then mentions how ggplot is *"not like anything else you've seen in R, so it might be really hard to get."*

Dr. Glynatsi agrees that matplotlib would be easier for a beginner in visualisation to learn, also noting that Python in general may be slightly easier when learning programming. An interesting note is also that she mentions *"as a person whose English is not [her] first language, [she finds] Python to be more readable"*, which could be an interesting topic for further investigation; whether native language has an impact on ease of learning different programming languages.

Additionally, Dr. Glynatsi notes the differences in the packages in terms of seeing which terms are part of the relevant library, mentioning that, apart from the opening ggplot function in the *ggplot2* plotting structure, *"there's no other indication that it's part of ggplot plot"*, but with the matplotlib one can *"see that is part of the plt library"* since *"Everything's after that plotting instance"*. With regard to the ggplot, Dr. Glynatsi also points out the *"weird plusses"*, for which the purpose of is not immediately clear, and could be a confusing factor when learning to use the tool.

Correlating with this, Dr. Palmer makes the argument that, for ggplot, *"when you're not used to it, it feels a little unintuitive"*, however also brings up the fact that *"you realise that you can do like some really cool things with like one variable or something and it's really neat"*. An example he uses is the functionality of the `aes()` argument; *"you can just tell it what your axes are, and then soon as you've told it what the axes are just throw our data at it"*, which is indeed a very nice feature of this library.

When discussing a beginner learning visualisation, Wilde mentions having learned *ggplot2* in the two weeks prior to the interview, and states, disagreeing with other interviewees, that he *"would say that's easier"* since *"it is very clear"* and *"you just have lots of pieces that can all work together"*. With regard to matplotlib, he says that *"you have to have an understanding of what you're actually trying to do, like from a programmatic, programmatical point of view"*, such as understanding dimensions of arrays, whereas this is not as much of a concern with ggplot.

When discussing the suitability of each language to visualisation, Dr. Artemiou makes the comment that *"using the ggplot package at least, you know, if you use something that is specifically designed for visualisation, so it should be suited for visualisation"*, which is a very good point; a library specifically designed to be good at visualisation should be well suited to do this task. The question regarding this was perhaps poorly worded in this respect, however some interesting insights were still gained.

Wilde, when considering this topic, expresses that he *[thinks] matplotlib is great* due to the *huge amount of control* a user has over every single element, and *"it's very easy to access it, if you are familiar with how to access objects in Python"*. This adds to the argument that, if a user is already familiar with Python, then matplotlib could well be the best tool for them to use for visualisation, and Wilde quotes it as *"a very, very sensible tool to have"*.

Wilde also notes, however, that *"It's kind of difficult to do complicated things very quickly in matplotlib"*, contrasting this with ggplot, whereby *"You can have very, very complicated data in whatever form, and you can plot it in a relatively complex way very, very quickly"*. This ability to deal with complex data in this manner is, again, a definite advantage of ggplot, and is helped by the specialty that R holds in complex analysis of big data.

Prof. Jones articulates that visualisation is *"a strength of R absolutely and the ggplot in particular"*, and that, considering matplotlib as a MATLAB import, *"Matlab certainly has good graphics as well"*.

Freedom of Customisation

Another factor to consider when looking at programming tools is the amount of freedom that the user has for customisation, in terms of creating bespoke plots for their needs.

In general, both are regarded among interviewees as allowing a large amount of freedom, with neither being hailed as particularly better than the other, with any arising comparisons discussing more the ease of customisation as opposed to the overall ability of the library.

Prof. Jones states that, for a less experienced user, he believes matplotlib provides more freedom as does Dr. Palmer, with the reasoning that, with matplotlib, *"you know how to access the ticks or whatever"*, but with ggplot *"you've got to have this new object on and then you've got to go look something up"*, mirrored perhaps by Dr. Artemiou's statement of *"the most, let's say, challenging thing is probably, knowing how to do it"* and Dr. Glynatsi's of *"given that you know, you have some knowledge, I think it's very easy for both languages"*.

Overall, it appears opinions on the amount of freedom in customisation of plot depends heavily on the users own programming knowledge and skills as opposed to any particular features of the packages.

Opinions on whether the specific visualisation code could be changed

This section will focus on the interviewees' opinions on the screenshots of code used to make the specific visualisations for the survey. The purpose of asking this was both to get feedback on the code and to understand how different people with different programming backgrounds would code visuals such as these.

First note that there were many comments on lack of documentation and variable explanations, however when these came up it was explained to interviewees that this was intentional, and the purpose behind this.

Dr. Artemiou mentioned that the code *"busy"* and perhaps slightly to complex, stating that *"Sometimes you can achieve whatever you want with a much simpler code"*, and so this is perhaps an area of improvement to focus on, on part of the author. He does however then go on to say that *"that is no wrong or right, I'm happy either way"*. The reasoning behind the perhaps over-complicated code could either be the author's programming, or the language, or both. Dr. Artemiou did not specify the language however, just mentioned this as an overarching comment, so it is more likely due to the author struggling with efficiency. Overall however, although stating that *"Yes, it can be changed"*, this interviewee appeared to be overall fairly indifferent.

Similarly, Prof. Jones appeared to not have too much of an opinion on changing the code at all, simply stating that *"this is a sort of code I would expect to be sitting behind the scenes with some other sort of interface for me to use"*. This is an interesting comment, and could potentially be down to the fact that the codes for producing the plots are written as functions, which would lend it to providing an interface into which the user may input arguments. Indeed, on further consideration, this code could quite easily potentially be put into an interactive shiny app, whereby the user can enter a number and get a plot showing a number of obstacles for the ninja warrior data. A further comment from Prof. Jones states that, when the code is of this format, *"In which case, as long as it works, I don't really care too much, I suppose"*.

Interestingly, it was the two R programmers that appeared more indifferent towards the codes, with the Python-predominant programmers having more of an opinion. Whether this is a result of experience in using each of these languages or just a general personal preference, however, is unclear.

Dr. Palmer makes the comment that the code *"defined local variables as the same thing as a global variable"*, and gives the variable named `viridis` as an example; the global variable was overwritten in the function by a local variable. This did not impact the functionality of the code, however will be considered when writing programs in future.

Dr. Glynatsi also brings up an issue of overwriting, stating *"you are overwriting the plot each time, so that's an error, in my opinion"*. This is expanded on as *"the plots are writing themselves. So you're not returning each plot"*, referring to the fact that each function creates multiple plots, and saves them individually within the function rather than returning a single output. Perhaps it would have been preferable to create codes such that the output is a single plot, with the plot type defined as an argument.

An additional note from Dr. Glynatsi states that *"it's very obvious what is the colour, it's very obvious that you're changing the y labels and the x labels and things like that"*, which the priorly discussed layered grammatical format of both languages lends to, since each layer adds a new one of these attributes. A perhaps opposing comment from Wilde is that, for example, rather than the separate calls to create the x and y labels, *"you can just do one labs call, and just have all of them together"*

There were a couple of comments on the use of functions, with Dr. Palmer asking *"Did you write those with Vince or did you write them yourself?"*, stating the reasoning behind this being that *"Vince really likes putting them in functions, so it's just interesting that [the interviewer] used functions"*. Indeed, Dr. Knight confirms this, saying *"I like that you've put your plotting code within- um in a functional way so within functions"*.

Dr. Palmer's own views oppose this, where he states; *"I like functions for most things, plotting is one of the ones that I just don't see why we do it functions"*. These two strongly opposing views show that the use of functions can be a very subjective matter, and it is up to the programmer whether they feel functions are appropriate to use.

Opinions on the visualisations themselves

After answering questions related to the codes,