Where you should be

By the end of Semester 1, most projects should at least be in this position:

- Have a clear idea of what the scope of the project is
- Have a clear, concrete plan to finalise development of the first prototype (for whatever that means for the particular project)
- Have a significant amount of design work or other preliminary work completed
- Have some code written that forms a substantial part of an initial prototype

Where you should be

In the project crunch period, i.e., the two weeks full time before the break for the festive period, you should ideally:

- Have a complete, working prototype that can be tested in some fashion
- Have a clear idea of how you will improve and adapt this in the next semester
- Have a clear plan for the evaluation

Status report

The **status report** is due Friday 16th December @ 5PM.

It is a small amount of work. It should take a couple of hours to produce (at most). The report must cover:

- **Project Description**: A clear description of the project, i.e. what it is that has to be done
- **Progress: R**eport what you have achieved as of time of writing
- **Plan of work: A** detailed plan of work, with deliverables, for the remainder of the project
- **Problems and risks: A**ny difficulties that you foresee or have experienced, and mitigations.

Status report

- There is a template and an exemplar on Moodle, which will give you a clear idea of what to expect.
- You must submit a single PDF via Moodle and **email it to your supervisor**.

L4 Project Evaluation: Did I do good?

Evaluating projects

Projects will typically investigate problems via a scientific process:

- Develop evaluation questions
- Build software ("apparatus") and a process ("protocol") to implement ways of gathering evidence
- Run *controlled* experiments to gather evidence
- Analyse the evidence to summarise it
- Discuss the findings and limitations of the investigation.

Not just for research projects

Software engineering projects will typically look at some aspects of:

- How well were the requirements met?
- How satisfied are the clients?
  - e.g. via interviews, walkthroughs
- Testing
  - do the tests pass?
  - do the tests cover the codebase?
  - do they cover the requirements?
- Performance analysis
  - does the software run sufficiently quickly/storage compactly/etc.
  - if user facing, usability metrics

What is a good evaluation?

- **Asks a relevant question**
  - Project: *"Building a static analyser to improve LLVM code generation"*
  - Question: *"How does static analysis affect run-times when computing register renaming strategies?"*
- **Collects evidence in a well thought out way, directed to the question**
  - *"A series of simulations were run with standard test bench programmes under three different renaming strategies."*
- **Analyses this data correctly and scientifically**
  - for example, visualisations, statistical tests, qualitative analyses
  - *"Figure 8 shows a Box plot of run time for 1000 runs of the simulator under each strategy."*

What is a good evaluation?

- **Discusses conclusions that can be drawn, in a fair and unbiased way.**
  - **The goal is not to show that your project is amazing**
  - Don't set up usability studies against straw man alternatives, for example :)
  - The goal is to show you know how to find out what you did, correctly and honestly.
  - *"None of the renaming strategies were superior to XYZ, but it is clear that random renaming is highly biased and can break down under many conditions"*

Being the best is not the best

- Be fair and honest in your appraisal of your own work
- Don't blow it out of proportion
- Be professional and provide a fair, objective assessment
  - Give credit where it is due
  - Explain limitations and caveats
- Your marker wants to see that you understand how you have done in context
  - Not that it is great, or better than existing approaches, or whatever.

Being the best is not the best

- **Avoid** evaluations which are biased towards what you have done.
- **Definitely** do not adjust or select results to make your work look better! (This is academic misconduct)
- **If you do an experiment you must report it!**
  - Even if the results do not support what you are trying to show; or else you are committing academic misconduct.

Bad results

It is not a negative result if you find your project does not have strong results

- e.g. is worse than a competitor, or has no apparent effect.

The key thing is to show that you know ***how*** to determine the quality of the product, and that you did so carefully and diligently.

- If results are negative, say so, and analyse why they are like this.
- If the results are inconclusive, discuss how you might have investigated this differently.

Scientific approach

You will not be able to investigate every part of a project.

- In more scientifically-oriented projects, there will be many variables you could explore, but you will only be able to explore a subset.
- In a more engineering-focused projects, there will be many tests and measures of success you could have investigated.

Scientific approach

You will have to:

- Choose a **sensible, representative** subset of options
- Discuss why that selection makes sense
- Investigate the options carefully and precisely

Scientific approach

Most of all:

- **BE SYSTEMATIC**
- If you are trying to identify an effect don't change multiple things at once
- Make sure you capture all the data you need. Log carefully and thoroughly.
- Where possible, automate the processing and analysis of data. DO NOT DO IT "*BY HAND*".
  - Make analysis an automated, replicable procedure which transforms raw data to graphs, statistics, etc. ready to insert into the dissertation.

Analysing well

You will have to analyse your results. This can take many forms:

- Qualitative analysis (e.g. coding) to group results from interviews or free-text questionnaires
- Statistical testing, to show quantitative differences between groups of results are not the result of sampling error but show true differences
- Visualisation of data, to show trends and patterns, like line plots, Box plots
- Summary tables of statistics (e.g. tables of means and standard deviations), to summarise repeated measurements

Your analysis needs to sensibly summarise what you have evaluated in a concise form.

Discussion

You must then present a discussion which puts the analysis into context. This might cover questions like:

- What do the graphs you've plotted show; how are they to be interpreted?
- For example, what does it *mean* that the average compilation time doubled?
- Just how much better is your project than competitor X?
- What else could you have done?

Visualisations

Use appropriate visualisations to display data. You are judged on your ability to:

- Select appropriate visualisations styles (e.g. bar charts versus scatterplots)
- Refine and collate data suitable for visualising
- Select appropriate *relevant* visualisations to include
- Present visualisations accurately (e.g. all details present)
- and aesthetically correctly.
    - e.g. not massive/tiny/drawn in MS Paint.

Visualisations

- Visualisations may play a lesser or greater role in your project depending on its nature.
- Most projects will have at least one or two graphs.
- Typically, favour graphs over tables of numbers.
  - It is good to have some tables too though!

Planning visualisations

A useful way of structuring an evaluation is to draw sketches of the visualisations you *want* to present, before doing **any** evaluation.

That is, draw out the pages that you will have and sketch in some outlines of the graphs/figures that will go there.

Then set about designing the process to actually capture the evidence and process it to get those graphs.

Representing uncertainty

If you visualise summaries of data (e.g. averages) **include representation of uncertainty**.

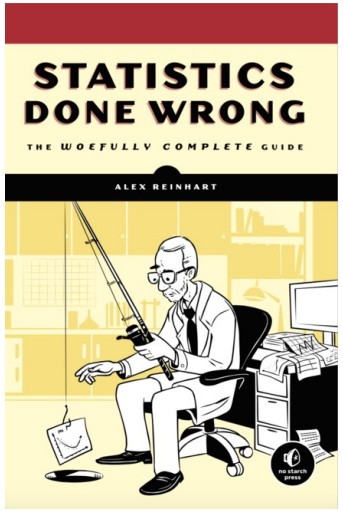
- This might be error bars, or a Box plot, or some more specialised visual device.
- **Never** present average results without at least some context about the variability of the data which generates those summaries.

There is further guidance in the visualisation guide available in the project template.

Statistical tests

If you are using statistical testing, it is easy to get things seriously wrong unless you understand what you are doing.

The book "Statistics Done Wrong" by Reinhart is a very helpful guide in not making errors in analysis.



Statistical tests

Make **sure**:

- You understand what you are testing: what are the dependent and independent variables?
- You understand very clearly what conclusions you can draw from the testing procedure (e.g. from null hypothesis rejection testing).
- What kinds of tests are appropriate for the data you have
- That you have correctly accounted for effects of multiple testing
- That you haven't done anything invalid like choose a testing procedure *after* looking at the data.

Reporting ethical data

IF YOU INVOLVE DATA FROM ANY HUMAN SUBJECTS AT ALL, YOU NEED ETHICAL APPROVAL.

- Make sure you state in your evaluation that you have sought and obtained necessary approval, if any data from human subjects appears.
- Simple cases: **Ethics checklist**, sign and include
- Harder cases (e.g. nonstandard hardware): **School Ethics coordinator**
- Even harder cases: **College Ethics Board** -- detailed paperwork and will take at least 4 weeks.
- Harder still: Forget about it (e.g. studies with NHS patients)

Preparing the dissertation

Basic rules: LaTeX, page limits, plagiarism

- You submit a single report, submitted as PDF.
- The dissertation page limit will be ~35 - 40 pages. Final guidance will be made available early next semester.
- You may include appendices beyond the limit, but no guarantee they will be read.
- LaTeX template is provided. You must **use it**.
- **You must not plagiarise anything. Be extremely careful!**

Basic rules: English, citations

- You will be assessed on the clarity and quality of your writing.
- You will be assessed on the clarity and quality of your writing, as well as the content.
- You are expected to cite appropriate literature and have a formal bibliography
  - There is no required size; maybe it's one reference; maybe it's one hundred
  - It must be appropriate for the project

Standard structure

Adjust as needed, but most projects will follow this pattern.

- **Introduction**: why should I care and what are you doing and what are you actually doing?
- **Background**: what did other people do, and how is it relevant to what you want to do?
- **Analysis/Requirements:** What is the problem that you want to solve, and how did you arrive at that?
- **Design:** How is this problem to be approached, without reference to specific implementation details.
- **Implementation:** What did you do to implement this idea, and what technical achievements did you make?
- **Evaluation:** How good is your solution?
- **Conclusion:** Summarise the whole project for a lazy reader who didn't read the rest (e.g. a prize-awarding committee).

Guidance

- **Introduction:** *Motivate* first, then state the general problem clearly.
- **Background:** Don't give a laundry list of references or other software. Tie everything you say to your problem.
- Present an argument. **Don't write a tutorial**; provide background and cite references for further information.
- **Analysis:** Make it clear how you derived the constrained form of your problem via a clear and logical process.
- **Design:** Design should cover the abstract design in such a way that someone else might be able to do what you did, but with a different language or library or tool.
- **Implementation:** You can't talk about everything. Cover the high level first, then cover important, relevant or impressive details.
- **Evaluation:** Ask specific questions that address the general problem and answer them with evidence. Be fair and be scientific.
- **Conclusion:** Summarise briefly and fairly. Indicate what future work could be done, but remember:
  - **you won't get credit for things you haven't done**.

Assessment

- You will be assessed on your technical writing.
- We want to see:
  - Clear, technical writing
  - Evidence of abstract thought and organisation of ideas
  - Appropriate standard of English
  - Use of visual aids (diagrams, plots, tables)
  - Well laid-out, visually pleasing document.
  - Fair and honest representations

Who is the reader?

This is the key question for any writing. Your reader:

- is a trained computer scientist: *don't explain basics*.
- has limited time: *keep on topic*.
- has no idea why any would want to do this: *motivate clearly*
- might not know *anything* about your project in particular: *explain your project*.
- but might know precise details and check them: *be precise and strive for accuracy.*
- doesn't know or care about you: *personal discussions are irrelevant*.

Remember, you will be marked by your supervisor and one or more members of staff.

Bear in mind, you might also have your project read by a prize-awarding committee or possibly a future employer.

Writing well

Oh No!! I need to write a dissertation

- Set aside a portion of a day and study how to write.
- **This isn't wasted time. It's an easy, pleasant and effective way to get started**
- Read some of the recommended resources, particularly the article by Simon Peyton-Jones and the essay by Kurt Vonnegut *before* you write anything.
- Set out a plan. Write "inside-out":
  - Easy things first: the technical details are things you already know.
  - Hard things later: stating the general problem and making a logical argument required reflection.

Some good books on writing

There are many style guides on good English writing. You don't need to read these, but they will improve how you write.

- "The Elements of Style" *Strunk and White* A bit outdated, and American, but a classic.
- "A Sense of Style" *Pinker* Excellent, though long
- "Style: The Basics of Clarity and Grace" *Williams*
- "Politics and the English Language" *Orwell*
- "How to Write with Style" *Vonnegut*
- "How to write a great research paper" video by Simon Peyton-Jones (recommended watching, even though you aren't writing a research paper)

Orwell's rules:

- Never use a metaphor, simile, or other figure of speech which you are used to seeing in print.
- Never use a long word where a short one will do.
- If it is possible to cut a word out, always cut it out.
- Never use the passive where you can use the active.
- Never use a foreign phrase, a scientific word, or a jargon word if you can think of an everyday English equivalent.
- *Don't use foreign, scientific or jargon terms if plain English works.*
- Break any of these rules sooner than say anything outright barbarous.

Clear, lucid writing

Cut to the chase:

- Favour brevity.
- Avoid convoluted, languorous, frivolous, florid, poetic or extraneous language which drags on endlessly (perhaps with parenthetical asides strewn throughout which elaborate without stating anything new) while failing to drive forward towards the fundamental key point, with a grand display of your wit, eloquence and sophistication.

Technical, not personal writing

Write the abstract problem, and how it was solved, and how you showed it was solved.

- Don't brain dump. You must structure what you say;
  - a collection of facts without organisation is without value.
- Use precise, technical language and stay on topic.
  - The reader of the report does not care about you but about the work that you did.
- **Do not write a blog**.
  - Your report should not be (purely) chronological. Your feelings, how you progressed each week, etc. are not relevant. It's not "one damn thing after another"! You must present a coherent, logical argument.

Who cares?

- This project was important for me personally because I was able to challenge myself in learning and implementing many new concepts, including graph data structures, distance measuring algorithms and machine learning concepts such as K- means clustering. Although algorithms and data structures is a module that I had undertaken, we did not cover graph abstract data types. This was a completely new concept to me, and one which I enjoyed learning about, especially in providing me with the knowledge to implement a flexible way of modelling real life objects. [...] From a personal point of view, the problem for me was overcoming the many challenges outlined above. Learning about advanced data structures such as the 2 graph and its many algorithms was a challenge, as was learning about machine learning and distance measurement.

Who cares?

- I chose to use the Java library because I was more familiar with Java as an environment.
- The project progressed well until my spleen went funny and this seriously affected my progress.
- It was difficult to implement the efficient algorithm with my limited understanding of algebra.
- I decided to use square designs for my buttons because I love Art Deco styling.

These are exaggerated examples, but if you wrote anything like the above in a dissertation it is likely to be scored out with the comment "who cares?" next to it.

This writing is inappropriate in a technical document.

Avoid clichés

Bad

- In a nutshell, at the end of the day the solution space, after all is said and done, is a can of worms.

Good

- We have no idea what we are doing.

Avoid excessive jargon and irrelevant detail

Bad

- Data was encoded for the protocol layer to be transported via TCP/IP using a polymorphic serialization library with first- class support for Unicode UTF-8 string sequences.

Good

- Data was sent using protobuf , which supports UTF-8.

Avoid excessive formality or archaic writing

Bad

- "I interrogated the client as to what the revised requirements ought to be forthwith."
- "Upon completing such a testing protocol, it became apparent that the antediluvian type system was quite wholly unsound."

Good

- "The client was asked for feedback on the requirements."
- "After testing, the type system was found to be unsound."

Don't pad writing with junk.

- A majority of – most
- Come to a conclusion – conclude
- for the purpose of – for
- has been proved to be – is
- not withstanding the fact – although
- give consideration to – consider
- in the neighbourhood of – near
- they are present in greater abundance -- there are more of them

More here: http://www.justinmclachlan.com/archive/writing/100-deadwood-phrases-to-cut-from-your-writing/

Weasel words

Delete all vague or irrelevant words, particularly superfluous adjectives.

Be precise, don't hedge and don't presuppose the reader's judgment. Never use these words unless you are using a particular technical sense ("extremely large cardinals", "brilliant reflection"):

- quite, often, fairly, actually
- maybe, partly, mainly, most,
- surprisingly, remarkably, exceedingly, obviously,
- unbelievably, incredibly, extremely
- very, vast, huge, tiny, massive, great, brilliant

Weasel words

Bad

- We quite often found a fairly surprising number of huge weasels under the sink eating exceedingly good cakes.

Good

- On Tuesday and Wednesday, we found two one kilo weasels under the sink sharing a cherry bakewell.

See here: http://matt.might.net/articles/shell-scripts-for-passive-voice-weasel-words-duplicates/

Good visual style

Your dissertation should not look sloppy. Typography is important. It doesn't have to be *beautiful* but messy, bitty documents will definitely count against you.

The template has been optimised to make your dissertation look good.

- Figures used appropriately, sized appropriately, and placed appropriately and referenced in text
- Annotations etc. clear and tidy
- Cross-references should be used and correct (no [?] in the document or "see previous section" -- **refer explicitly using LaTeX cross-references**)
- Equations formatted correctly (if you have any), referenced correctly.
- Text broken into paragraphs and sections logically and neatly.

Literate, grammatical prose

- Splell thigs correctl. Even recieve. Spell check. You will be ruthlessly penalised for failing to do this.
- Write grammatically, in full sentences.
- You don't need to show off your vocabulary. Keep it tight.
- If you aren't a native English speaker, there is help that LEADS can provide for writing to support your writing skills.
  - https://www.gla.ac.uk/myglasgow/leads/students/writingstudyadvice/scienceengineering/

Tense and voice

- Use a consistent tense and voice.
- From the reader's perspective, your project happened in the past, and continues to exist. This usually means using the **past perfect** tense.
  - "The research showed that.."
  - "We developed…"
- You may need to use other tenses at specific points (e.g. in the introduction), but make sure that tense is consistent.
- You can write in the active voice *"We found that…"* or passively (often less clear) *"It was found that…"*, but do so consistently and appropriately.
  - https://writing.wisc.edu/handbook/style/ccs_activevoice/
- The norm is to write in the plural first persion ("we") even though there is only one of you (I hope!).

Evidence of abstract thinking

Evidence of abstract thinking

- You must state the general problem, without specifics, and then specialise it.
- You need to break down the problem from its abstract form into implementable parts, *showing a clear and logical train of thought*
- **Note** the abstract problem, and the train of thought, are not usually completely clear *until you have finished the project!*
  - This is partly why a chronological report is not appropriate.
  - You need to reflect on what you have done, pull out the key problem and form the argument.

Evidence of abstract thinking

Bad (introduction)

- "This project is a Java implementation of a DPSK modulation scheme with LDPC error correction (from the jldpc library) to transfer iOS XML files at 44Khz to Windows machines."

Good

- "This project develops a method to transceive data through free-air sound using standard speakers and microphones."

Don't mix abstraction levels

This is one of the most common serious errors in undergraduate dissertations.

- Keep the specifics of your ideas out of the way of the larger concepts. Don't describe the trees until the reader understands the forest.
- Think about the dissertation as a hierarchy of abstraction levels. Don't mix concepts from one level with another.
- Frame your writing. You should set out what you are going to write about in writing; and it should be clear how each step follows from the next.

Don't mix abstraction levels

Bad First paragraph of the **Implementation**:

- The pin array previously created was hard-coded for 198 pins and did not allow any easy addition or subtraction of pins. To make this more versatile, the first thing to do would be to remove this hard-coded pin array. An object will be created which will be responsible for spawning the pins. This object's behaviour will be laid out in a C# class which will be attached to the game object as a script. The script should provide an open slot for the prefab pin to be clicked and dragged in using Unity's inspector.

Don't mix abstraction levels

Good

- The effect of rendering tactile sensations with a pin array under the finger depends on the number of pins available. There are many array-style tactile devices with different pin configurations. This project involved adapting the tactile rendering pipeline to support arbitrary pin configurations in simulation.

[much later]

- The implementation platform was Unity. Each pin was modelled as a separate object whose properties were exposed to the developer via the Unity inspector.

Don't mix levels of abstraction (II)

- Start with abstract thoughts and general structures, and refine to introduce details.
- Do not suddenly switch between abstract concepts and specific details. This is a hallmark of brain dumps and sloppy thinking.

Don't mix levels of abstraction (II)

Bad

- The system must store student records securely. We used curve25519 implemented via the NaCL C library with a key size of 256 bits to ensure that public key distribution was compliant with the ISO regulations. This had an average total signing time of 0.5ms per compressed message, which we hope to reduce to 100us by rewriting the modular multiplication in ARMv8 assembly. The security architecture also included an append-only logging module.

Good

- The system must store student records securely. There are two parts to this: cryptographic signing to ensure authenticity of each update and irrevocable logging to allow for auditing. To support signing, we used an elliptic curve based method, using curve25519 as provided by NaCL [cite]...

Don't contextualise too far

The late 1970s and the 80s saw computers make it into the homes of the general population due to the advent of the 'mini-computer'. Before this, computers were most commonly found as large mainframes in companies which required whole rooms just to house them (Interaction Design Foundation, n.d.). The computers pre-late-1970s were only used by computer specialists – experts, who were trained in their use (Carroll, 2013).

As such, the computers and their software were engineered for the use of computer experts who knew all about them. This created an issue for the spread of the 'mini-computer': how could users with no experience nor training be able to use a specialized machine and its software which was built for computer specialists? This is what saw the creation of the field of Human Computer Interaction (Interaction Design Foundation, n.d.).

- **This had no meaningful bearing on the project itself, which is about tactile interfaces in VR.** Its just wasted space.
- Remember to focus on ***your*** problem, for a reader who is a trained computer scientist.

Creating an argument

An argument

At each stage of the process, you are narrowing doing the problem to approach something that can be implemented (e.g. by writing software). Some of this is by:

- ***Argument***, using logical reasoning:
  - *"We need to be able to implement this in real time. This requires a 20ms response for my application. We need to process 200 batches of data per response. We therefore need to target 0.1ms processing time for each batch."*
- ***Reference***, using background sources:
  - *"McInnes et al. 2017 suggests that a topological approach may be more effective; this is backed up by the results in..."*
- ***Experiment***, using results you found.
  - *"The first interface was very frustrating for users who had to click the back button nine times to edit each field."* or *"During requirements capture, the client specified that they must be able to see all of the zebrafish on one image*", or *"Simulations with the first generator showed that it produced unacceptable autocorrelations modulo small primes*".

Finishing the argument: evaluation

At the end of this you will have:

- defined a problem;
- narrowed a problem to something implementable;
- implemented it.

You must then show that you have addressed the problem!

This is the purpose of the evaluation. The evaluation must show that you have solved the problem, or at least how well you have approached solving it.

This is done by asking questions and seeking evidence to answer them.

Finishing the argument: evaluation

The way to approach this is to ask a general question, which you likely cannot answer directly, and then find some specialisations that you *can* find evidence for.

- **Problem:** a system to allow biologists to work out how fast zebrafish breed.
- **General question:** Can biologists work out how fast zebrafish breed with my system?
- **Specific question:** Can biologists reliably count the number of zebrafish displayed onscreen with the highlighting algorithm I have developed?
- **Evidence:** 10 biologists were asked to count the number of zebrafish displayed on screen, from a standard set of zebrafish mating videos. The true count and the biologists' estimated counts with and without highlighting are shown in Figure 1.
- **Conclusion**: the highlighting I developed reduced count standard deviation by a factor of 3.

Logical structure

- Your dissertation has to reveal this logical structure.
- Use typography and writing to indicate the argument.
- That means sectioning in logical places to provide the skeleton of the argument, and writing in such a way that the ideas presented flow together, with an obvious and smooth transition from one block of prose to the next.

Subdivide sections appropriately

- Use sections/subsections/subsubsections judiciously
- A good rule:
  - You should be able to understand the report just by reading the section headings, just at a coarser level of detail.
- You want to split the text up logically, but avoid dicing everything into tiny chunks.
- Sectioning lets you cross-reference precisely, which you should use where appropriate.
- Sectioning gives a table of contents as a guide to the reader.

Make sure there is a flow of argument

Each part of your report should logically follow from the previous. There are two basic ways this can happen:

- You explicitly write a bridging sentence. *"As discussed in Section 2.1.1, the error correction is critical. This section discusses…"*
- It is obvious from the overall structure of the report, following an outline you have established earlier. For example, you state that you will present an experimental design, followed by the analysis of results in the introduction to an evaluation, and then you do.
- Minimise context changes. Try and group relevant discussions of one concept in one place.

Consistent, technically accurate writing

Define abbreviations, and symbols

Use and capitalise them consistently

Bad

- By CSR, we can conclude that $g_a, f_1 \in G$. Of course csr is a weak form of structure. … Strong sequential relations are part of…
- [What is CSR?! or g_a?]

Good

- By crisp sequential relations (CSR) we can conclude that $g_a, f_1 \in G,$ where g_a is the primary order, f_1 is the secondary order, and G is the set of all sequential orders. Of course, CSR is a weak form of structure. … Crisp sequential relations are part of…

Numerical issues

Basic scientific practice: *Don't use excessive significant figures*

- We found that 95.652% of users preferred the accelerated mouse.
- The system ran 1.23102x times faster

Use as many as you have reasonable evidence for. If you tested on 23 users, round to nearest 1% is plenty.

- 95% of user preferred the accelerated mouse.
- The system ran 1.2x times faster

Averaging

If you are reporting averages (means), consider reporting standard deviations as well, so that the variability of your results are clear, as well as the number of elements averaged over.

- The system ran one frame in 212.1ms (std. dev. 18.2ms, 200 trials)

A literature review

Depending on your project, your background section may be more of a formal literature review, where you discuss relevant academic papers to your topic.

Other projects might be less formal, and primarily discuss existing software.

Whatever you do, you must show three things:

- you understand what other people have done; (knowledge)
- you understand the limitations of what others have done, and how viewpoints differ (critical thinking)
- you can tie this to your project (logical thinking)

Guidelines

No laundry lists. Absolutely do not simply list a bunch of references. *The game is not to catch them all*. Cite relevant work, and *discuss it*. If there is nothing to discuss, then it isn't relevant.

- DO NOT DO THIS. "There have been studies of VR sickness [XYZ 2010, ABC 2011, DEF 2019]."

Think critically State what other people have written, and comment on it if appropriate.

- "XYZ 2012 suggests that VR headsets cause motion-sickness if used for more than an hour. However, they tested with pre-Oculus hardware and may be influenced by low frame-rates of that era's technology".

Guidelines

Compare and contrast fairly If there are multiple viewpoints, discuss them fairly.

- "XYZ found evidence of VR sickness, but ABC suggests that this does not occur if framerates are better than 90Hz. However, ABC's argument is based on an experiment with simple geometric shapes, and may not induce the realistic optical flow that XYZ tested with."

Tie to the project Everything you write in the background must be part of an argument to do with your general problem. You must make it crystal clear how background you discuss relates to what you are doing. **[Problem: How to use VR effectively in museums]**

- "VR sickness is a major problem for ad hoc uses like museums, as there will be a large population of transient inexperienced users passing through. It is essential to characterise how often VR sickness occurs, and what can be done to prevent it. In the literature, the prevalence and causes of VR sickness has been extensively studied..."

What should you cite?

- There is no minimum or maximum number of references. It is wholly project dependent. Typically we might see five to fifty references, more usually around a dozen or so.

There are three basic uses of references:

- **to specifically attribute** You must cite anything you quote or take imagery from, or otherwise directly use ideas from, or you are committing plagiarism.
- **to discuss ideas from** Cite texts that contribute to your arguments. You need to show you are aware of, and can comment critically on work done by others.
- **to provide background** Cite texts that providing supporting material. For example, don't explain to the reader how a recurrent neural network works in the form of a tutorial. This isn't your job. Instead, briefly describe it and refer to a citation for further details.

Citations and bibliography

- Cite references adjacent to where you discuss them. They must go adjacent to literal quotes.
- In the study of owls, there is one primary axiom: "The owls are not what they seem." (Lynch, 1993)
- Use the Harvard style for referencing, which will give (author, year) style citations. This is the default in the template.
- If you quote text, you should not edit it unless you have to remove words for brevity. If you do this, include an ellipsis or replaced text in square brackets to indicate your edits.
  - "This was a vision […] I was on the veranda of a vast estate."

Citations and bibliography

It's fine to cite web pages. Make sure you include the "last accessed" information

- Smith, John. “Obama inaugurated as President.” CNN.com. http://www.cnn.com/POLITICS/01/21/obama_inaugurated/index.html (accessed February 1, 2009). [from: http://www.bibme.org/citation-guide/chicago/website/]

If you cite as part of a sentence, don't include parentheses (use \citet{} to make the citation)

- Bad: The problems with bad citing style were discussed in (Williamson 2018).
- Good: The problems with bad citing style were discussed in Williamson 2018.

How to write the abstract

Write the abstract last

- Abstract is formulaic:
  - Motivate; set aims; describe; explain results

- "XYZ is bad. This project investigated ABC to determine if it was better. ABC used XXX and YYY to implement ZZZ. It was found that ABC was better than XYZ, though it caused rabies in half of subjects."

Typography

Format equations properly

If you need to use equations, format them correctly. There are many editors online that can help you write good LaTeX.

Bad

- `$$f(\xi = 1/2(\int f(x e^{2\pi i x } \xi) [Fourier transform]$$`

Good

- `$$\hat{f}(\xi) = \frac{1}{2}\left[ \int_{-\infty}^{\infty} f(x) e^{2\pi i x \xi} \right],\ \text{[Fourier transform]}. $$`
- Real fractions, brackets sized correctly, details like integration limits included and hat over `$\hat{f}$`, text formatted correctly in equation, correctly punctuated, superscripts contain all symbols.

Format equations properly

Tools

- DeTexIfy lets you look up LaTeX symbols by drawing them.
- There are many online editors; I have no specific recommendation, but these both work:
  - https://www.latex4technics.com/
  - http://rogercortesi.com/eqn

Format equations properly

Use appropriate mathematical symbols

- Don't invent your own mad symbology!
  - p=299792458 m/s where p is the speed of light
- See *Mathematical Notation - A Guide for Scientists and Engineers* by Scheinerman if it isn't clear to you what you should use.

Appropriate use of supporting material

Clear, useful tables, figures and diagrams

- When information could be displayed better without words, do so.
- **Visualisation is important.** Most reports will have at least some visual material, and these are an important part of the assessment criteria.
- Basic rules: Visualise if you can. Avoid repetition of similar visuals where possible.

Clear, useful tables, figures and diagrams

Key types of visualisation:

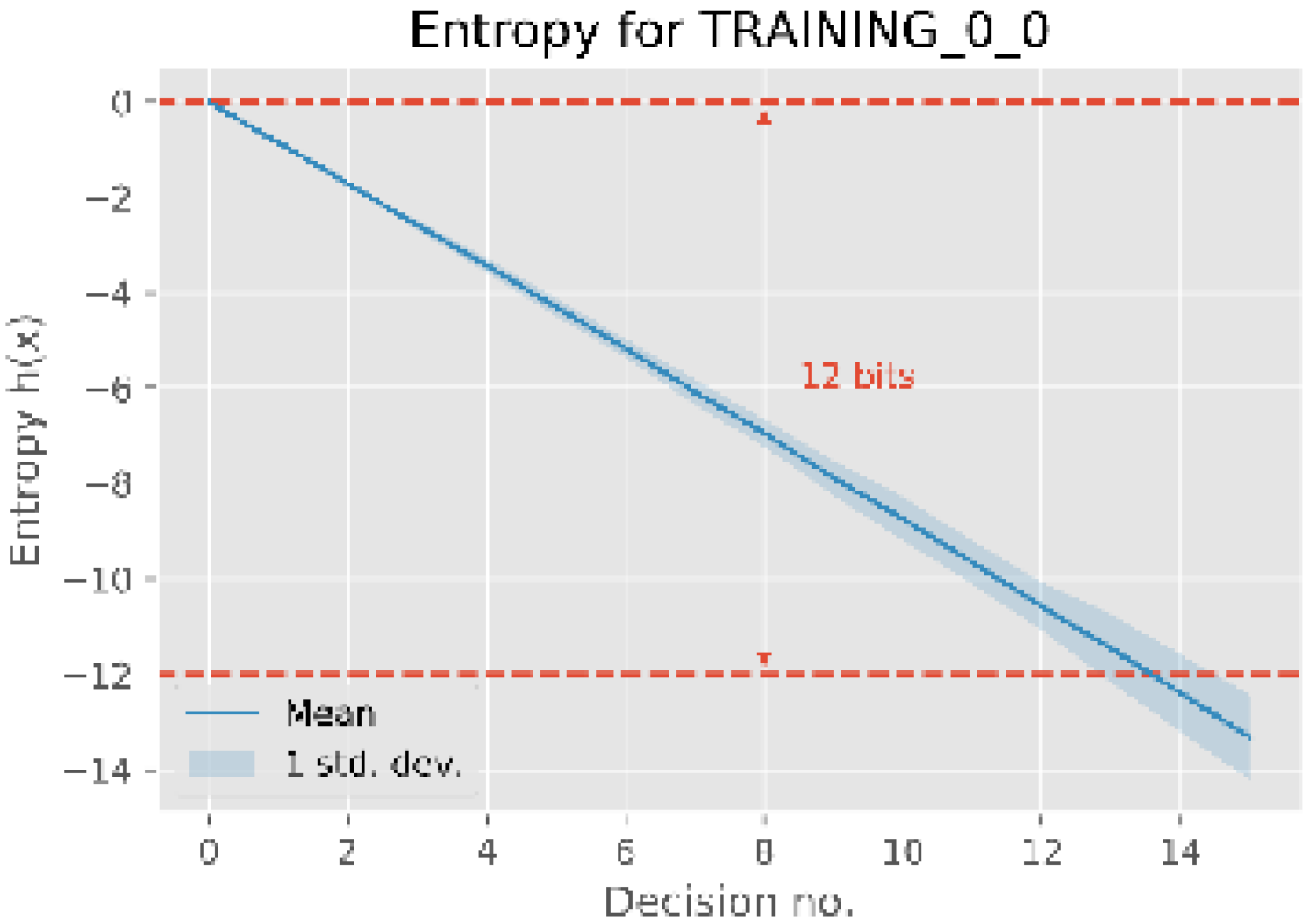
- **Table** organise results which have regular structure; e.g. analysis results
- **Diagram** explain structure or process visually; e.g. system architecture.
- **Image** photograph or screen capture representing snapshot of the world.
- **Graph** represent data on the page; e.g. performance curve

Avoid using bitmap images if possible

- Don't use .png / .jpg files if you can use vector formats
- .pdf is most convenient for LaTeX inclusion
- Edit vector files using a vector editor
- Keep files in vector formats (.svg for example)
- Keep and edit files in vector formats (.svg for example)

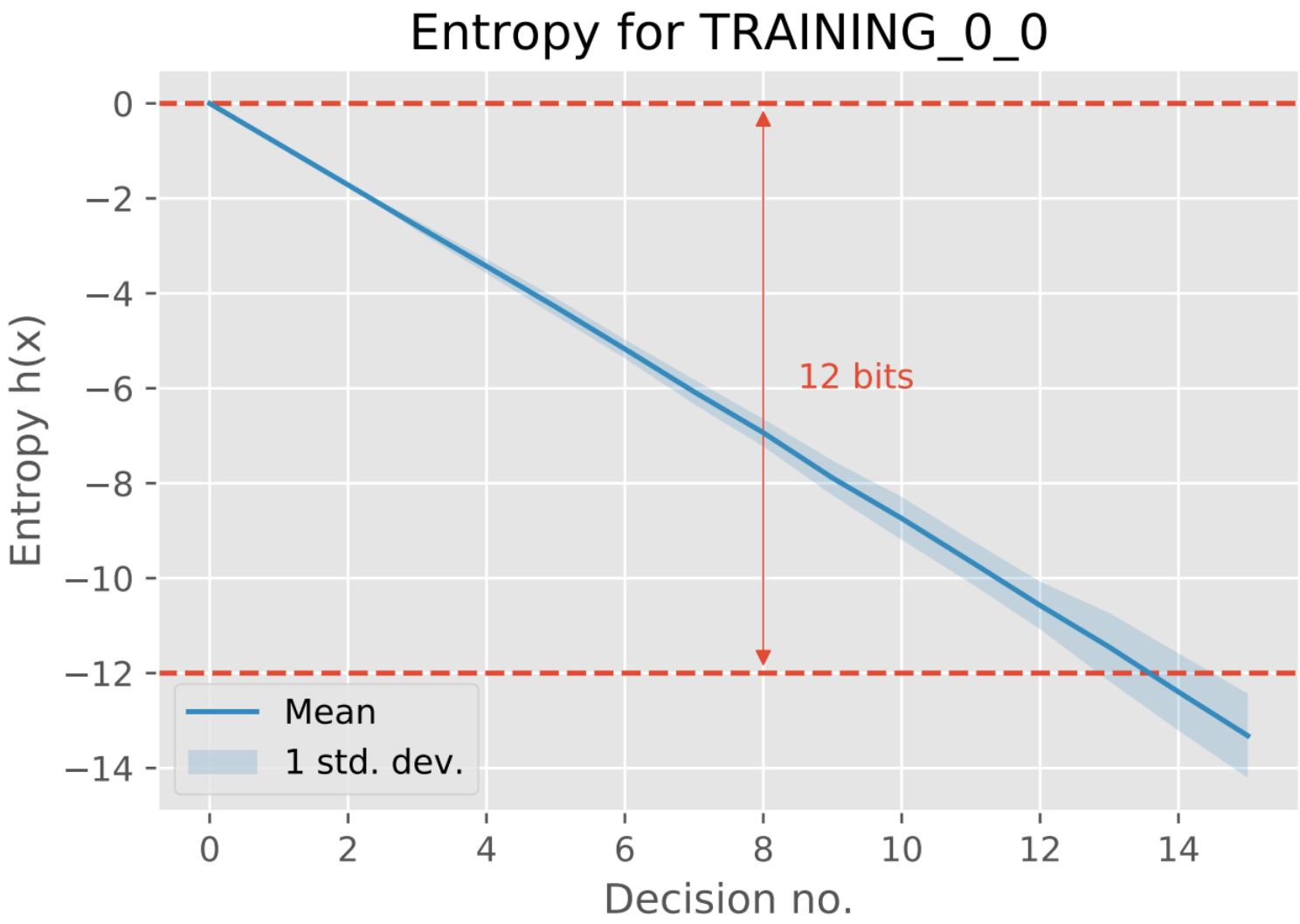
Avoid using bitmap images if possible

- Bad



Avoid using bitmap images if possible

- Good



Caption correctly and clearly

Figure captions should state:

- *what is being shown*,
- *what the reader should see*,
- *why the reader should care*,

with enough context that it isn't necessary to read the body text.

Caption correctly and clearly

- **Bad**

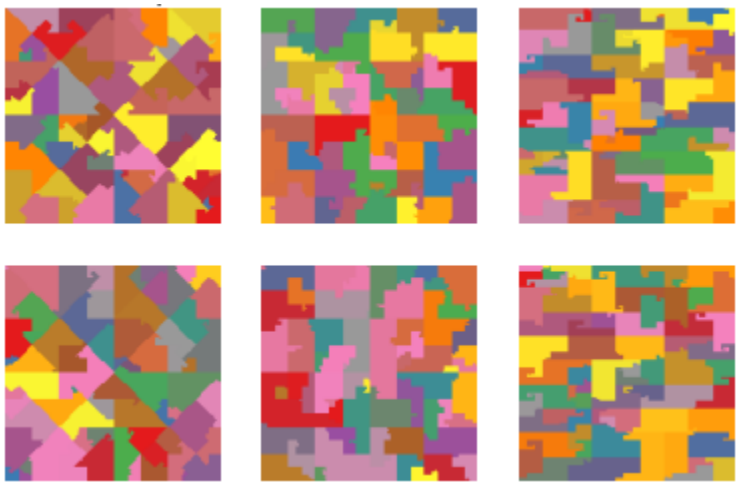


Figure 1. Different space filling curves.

Caption correctly and clearly

- Good

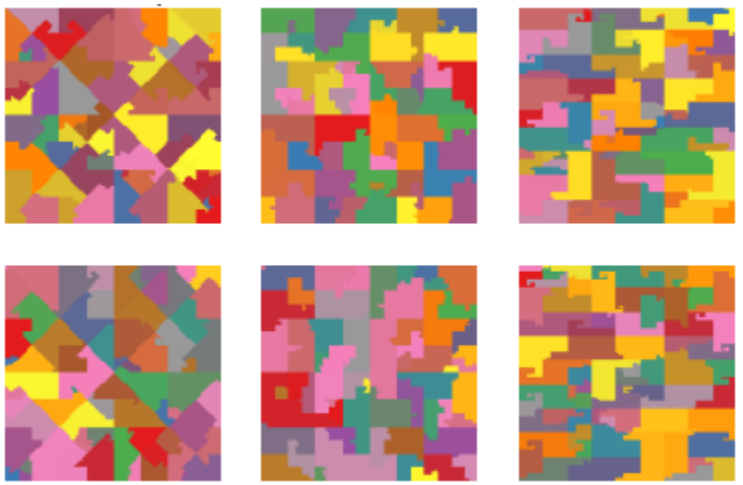


Figure 1. The properties of three different 2D space filling curves. Left to right: Peano-Sierpinski curve, Hilbert curve, Balanced GP curve [Haverkort 2009]. Each curve is shown by randomly partitioning a sequence of points into a set of contiguous regions, and colouring each partition differently.

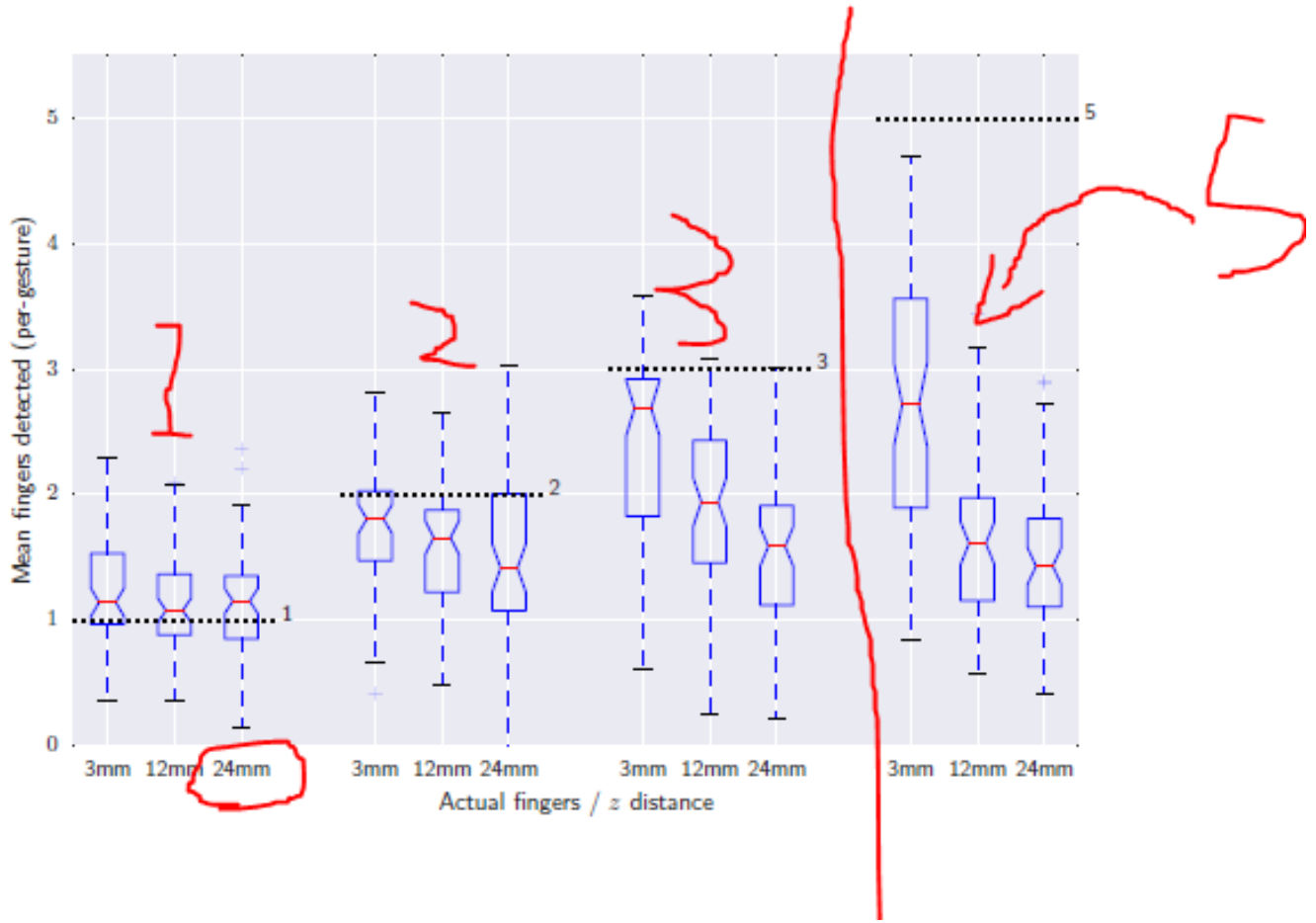
"It is clear from Figure 1 that the Peano-Sierpinski generates more diamond-shaped structures, the Hilbert curve is rectangular but rather irregular, while the balanced GP maintains a more even distribution at the cost of occasional "tail curlicues".

Size appropriately

- Your figures must be readable on-screen at approximately A4-like size.
- This means axis labels, ticks, etc. need to be readable!
- Also: do not use gigantic text or huge symbols in a figure!
- *Do not use massive figures where the on-figure text is radically larger than the body text.*
- *Do not use microscopic figures that are unreadable.*

Annotate clearly

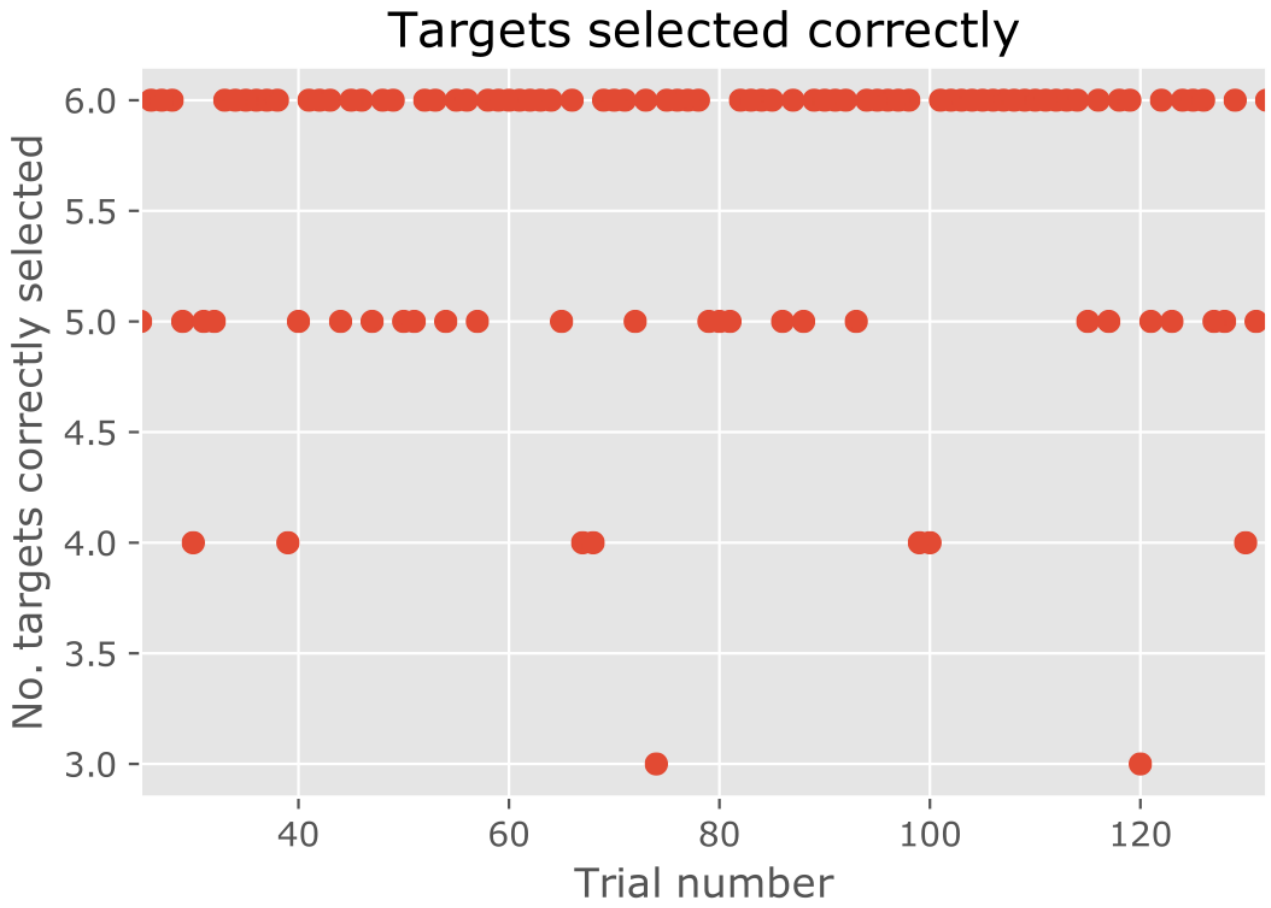
- Do not scrawl on figures. If you must edit a figure by hand, make sure the result looks correct
  - For example, similar font, similar size, clear arrows indicating what you are annotating, etc.



Fairness and honesty

Visualise honestly

- Do not edit visualisations in a way that might distort conclusions. *This is academic dishonesty.*

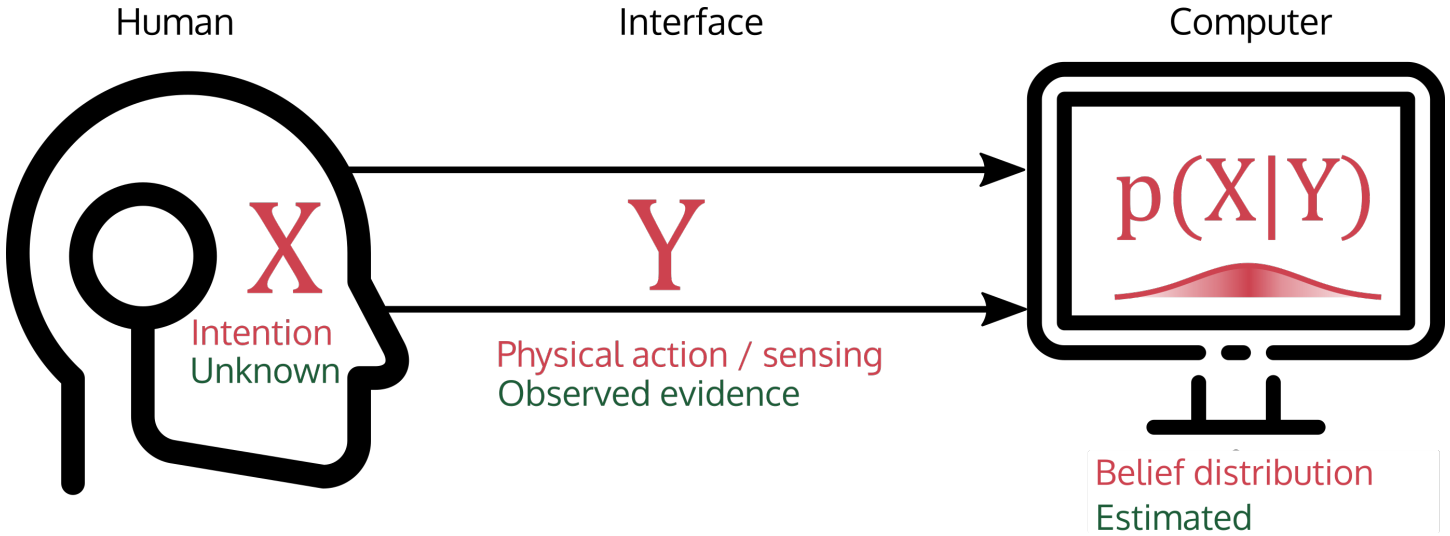


Visualise honestly

- *What happened to those bad results at the bottom?*
- Minimise editing you have to do. If you have to hand edit figures, how will you automate the generation of visualisations?
- If you have to use mock data (e.g. a mock-up graphs), state this very clearly and explain what you are showing. Be very careful that no-one could mistake mock data for real data.

Correct referencing of images

- Reference image sources directly in the caption



- **Bad**
  - *Figure 1: The interaction problem as a problem of conditional probabilistic inference.*
- **Good**
  - *Figure 1: The interaction problem as a problem of conditional probabilistic inference. Adapted from Williamson 2018, p.6.*

Back up with evidence

If you make a statement, and it's not common knowledge, you must back it up with evidence. There are three common forms for evidence in dissertations:

- Make an argument.
  - "By Jensen's inequality, we find that E[log(x)] \leq log(E[X])"
- Cite existing work.
  - "No modern relational database actually implements relational algebra [Soandso 2012]"
- Provide your own results.
  - "Reaction times are slowed by drinking alcohol, as shown in Figure 9."

DO NOT PLAGIARISE

Make 100% sure you do not plagiarise *anything* in your project.

- If you use text from another source, quote it, and put a citation *next to it*
- If you use ideas from another source, reference them with a citation
- If you use ideas drawn directly from another source, reference them with a citation.
- If you use a figure from another source, cite in *in the caption*
- If you redraw or adapt a figure, cite this clearly *in the caption*.

DO NOT PLAGIARISE

If you use source code or data from another source:

- Cite it in the dissertation
- Also cite it appropriately in the source code
- Also cite it appropriately in the source code, where it is used, and state the license.

If you hire someone else to write your project, expect to be expelled. Degrees awarded may be revoked if obtained by fraud.

Plagiarism penalties

The penalty for plagiarism, even the most minor lapses, is **severe**. These cases are dealt with by *Senate*, not by the School.

Academic staff are bound to report all cases of plagiarism to Senate.

- The minimum you could expect is a drop in bands (typically two) for minor infractions (e.g. clearly accidental);
- More severe cases (wilful, deceptive or extensive) could involve:
  - Awarding H with no possibility of resit for project (no Honours degree)
  - Expulsion (no degree at all)

There are appeals, but you have virtually no chance of appealing a plagiarism verdict where there is evidence.

Copyright

- You must comply with (software) licenses that are involved
  - Do **not** distribute software, as source or binary, if you don't have permission to do so.

Copyright

- Use of images from other sources in dissertation is typically permitted *if*
  - The image is directly relevant to the academic work (e.g. you are discussing a diagram)
  - There isn't a reasonable alternative to using the image (e.g. you couldn't just take another photo or plot another graph)
  - The dissertation is not being published (e.g. you are not making it into a book)
  - You clearly cite the source.
- If you use Creative Commons imagery, adhere to the acknowledgements requirements.

Evaluation

Structure

- What is the problem? *Restate the general problem*
  - E.g., "How does my server compare to Apache?".
- What questions could we ask? *State specific, concrete questions that can be answered with evidence*
  - *E.g.*, "What is average uptime of a server configured with my web server?"
- What did we do? *State **precisely** how you went about the evaluation*.
  - You must describe the methods, apparatus and procedure fully enough **that someone else could reproduce your experiment and get the same data**.

Structure

- What data did we get? *Present summary of results*,
  - E.g., number of trials, total data captured, number of participants.
- Analyse data. *Present graphs, statistics or tables which draw out evidence that answers your questions from the data.*
  - E.g., "Figure 1(a) shows a histogram of uptimes for my server and Figure 1(b) shows a histogram of uptimes for Apache. It is clear that my server has many fewer reboot failures"
- Draw conclusions. *Answer the questions you asked precisely, referring to your analysis*.
  - E.g., "My server is superior in uptime when load is <50% but is highly unstable as load increases."

Linking to the assessment criteria

Assessment criteria

- **Analysis** (15%) Clarity of thought; precise formulation of problem; understanding of context. Has the student analysed the problem, and devised a suitable approach for solving it? Has the student surveyed relevant literature
  - and existing software products?
  - Has he/she captured the requirements?
  - Have clear research questions been developed?

Assessment criteria

- **Software Product** (40%) Software design, implementation, and documentation where appropriate Is the software well- designed, functional, reliable, robust, efficient, usable, maintainable, and well-documented? Has it been demonstrated? Is the product represented adequately in the dissertation?

- **Research Product** (40%) Quality of the research, innovation, rigour in the way it is conducted, Has the research been conducted well (i.e. scientifically sound)? Does it show evidence of original thinking? Are there significant errors? Would the research be worth of publication after revision?

Assessment criteria

- **Evaluation** (10%) Testing and user evaluation where appropriate; suggestions for future work Has the software been thoroughly tested, and subjected to appropriate user evaluation? Does the student have good suggestions for further work?

- **Dissertation** (20%) Completeness and coherence, organisation, literacy, bibliography Is the dissertation complete, well- organised, and literate? Does it clearly explain the problem, and how the software was designed, implemented, tested, and evaluated? Does it contain a bibliography and proper citations? Is it well illustrated with appropriate technical imagery?

Dealing with the page limit

You will write more than fits in the page limit. Some strategies:

- Move details to the appendices (e.g. additional figures, tables). Keep the key things in the main body.
- Trim down wording. You can always be more concise.
- Size figures appropriately, without making them unreadable.
- Cut down sections that don't talk about what *you* did; for example trim background and future work first.
- Don't repeat lots of figures with similar data. Try and find a way to combine them or reduce them (for example, layer figures, or move less interesting ones to the appendix)
- Don't include big chunks of source code.

What goes in the appendices?

You don't *need* any appendices. It's fine not to have them. Common things to include are:

- Copies of ethics approvals (required if obtained)
- Copies of questionnaires etc. used to gather data from subjects.
- Extensive data or graphs that are too bulky to fit in the main body of the report.
- User manuals, or instructions to set up and run your project.
- Extra design materials, like wireframes or results from a requirements capture exercise.
- **Don't include your source code in the appendices**. It will be submitted separately.

Submitting

- Submit your dissertation and code on Moodle.
- Make sure you format your submission as follows:
  - [student_id].pdf with the dissertation.
  - [student_id].zip with the entire source code for your project.

Penalties

- If you do not follow the submission guidelines carefully (wrong template, wrong file structure) you will receive a **two band penalty**.
- If you submit late, you will be penalised two bands per day, starting the moment after the submission deadline.
- The project is 40 credits. Think about what this will do to your degree classification!