A Beginner's Guide to the Documentation of LaTeX and other resources, with supplementary notes from a fellow student

C. G. P. J.

November 12, 2022

Dedication

For Nicole, that we may work together.

Contents

Chapter 1

Introduction

I decided to compile a guide to the documentation of LaTeX. Such a guide is needed, because there are many good works that can help a person to become erudite in LaTeX, but they are not always easy to find; and in every place where a good work is not found, a less satisfactory explanation stands.

This guide is accompanied by (and is a part of) the a repository, which is available on github. The name of the repository is "A Beginner Guide To LaTeX Resources".

`https://github.com/chrisgpj/A-Beginner-Guide-To-LaTeX-Resources`

The kernel of this guide is Chapters three, four, and five. These chapters direct you to other resources, and from there you will be able to find your way. The reading time for those two chapters is less than half an hour.

A note on the text

Throughout this guide I have used "LaTeX" in many places, rather than "LaTeX". While LaTeX is the official rendering, "LaTeX" is official in plain-text environments, and I prefer this in prose.

Chapter 2

The Structure of This Booklet

This booklet is *not* an introduction to LaTeX, and it is *not* to expound the virtues of LaTeX; such things have already been written. This booklet instead attempts to help other people by directing them to documentations and tips that I wish I had on-hand much earlier in my journey. Therefore, when I provide advice and examples, I will tend to refer you to other resources, rather than re-inventing the wheel myself.

If you are very early-on in your LaTeX journey, then you may wish to refer to chapters three, four, and five.

An index may also be added to this booklet. Using the index, you can find where I talk about something like installing LaTeX, or just getting a simple document up with a nice font. These are reasonable things to want to know! In fact, it is famously the wisdom of LaTeX masters that people spend too much time concerned with LaTeX and not with writing. But I do encourage you to go on from such beginning stages to find cohesive documents or resources that can help you understand LaTeX more fully. If you do this, you will be better-off in the short, medium, and long terms.

2.1 Part One

The first part of this booklet is chapters three, four, and five.

In chapter three, I tell you about some of the documentation for LaTeX. In this guide, 'documentation' is anologous with official guides and handbooks, or to guides written by people who can be considered authorities in the realm of LaTeX

In chapter four, I introduce you to some resources that will be valuable for becoming a rounded thinker in the realm of markup languages and typesetting. These are mostly things that I was introduced to through the digital text course at Signum University.

In chapter five, I share my experiences on you about online resources, official and unofficial. The chapter contains some of my commentary about this matter. In the realm of LaTeX there is a blurriness between authoritative documentation and unstructured help. Therefore, the commentary in this chapter is of universal interest and can be read as a supplement to chapter three.

2.2 Part Two

The second part of this booklet is chapters six, seven, and eight.

In chapter five, I provide you with some examples of LaTeX at work. These examples are real-world examples from early on in my LaTeX journey (actually, I am indefatigably in this part of the journey, still). The examples here are very bad, if you judge them by quality of work, but they have the virtue of showing how LaTeX can be useful even when you do it badly. They also show many mistakes to avoid.

LaTeX not only becomes more useful as you progress, but also becomes exponentially easier to write with, until it is easier than using the more common 'what you see is what you get' word processors. Chapter six speaks to improving your workflow with LaTeX. In this chapter, I tell you how I go about using LaTeX now.

In chapter seven, I share some features and capabilities of LaTeX that I have found useful for doing simple document preparation. I also take a troubleshooting approach, trying to spot things that can seem a bit jarring when learning LaTeX. This chapter contains some commentary on the culture of

LaTeX.

2.3 Part Three

Part three of this booklet contains chapters nine and ten.

Chapter nine is my attempt at a glossary of terms of concepts. This chapter is quite dense, because it tries to clarify pivotal concepts, including those that I have found to challenging in the past.

Chapter ten is my attempt at an index of topics in this guide.

Chapter 3

LaTeX Documentation

Without good documentation, there is no such thing as a good system or program, or whatever the *thing* is. It is my view that good documentation cannot save a bad project, but bad documentation can hamstring a good one.

If you read this chapter, you will arrive at the end with the knowledge of where to go to learn about LaTeX, where to lookup reference for commands and markup, and where to go for tutorials on your very first documents.

However, it is necessary to speak briefly on LaTeX first, or else confusion will abound later-on. The following synopsis is the bare minimum.

3.1 The Structure of LaTeX

The place to begin is with /TeX.

TeX was made by Donald Knuth to improve the typesetting of academic articles and books, which had been declining rapidly with the introduction of digital technologies.

An axiom of TeX is that is it a *typesetting* program. The manual typesetters of old had to forge a path from handwritten manuscripts to formatted print. They had to make decisions about what fonts to use, what types of words to make italic, how to format their chapter-breaks and so on. The manuscript would be 'marked-up' to communicate specific instances of typesetting action; an underlined word usually meant 'make this italic', for example. However, even at this stage there was still a lot of work to be done in figuring out exactly how to execute those instructions — when to break to a new line, when to hyphenate a word, and such. Typesetters were constantly calculating within the overall instructions that they had developed for the text[1]

The way TeX works is sometimes likened to the process of typesetting. We write the overall instructions and markup the text using the TeX language. The TeX program then takes those instructions and markup, and transforms these into machine-readable instructions for the TeX typesetting engine, which runs calculations to optimise how the document is rendered, and produces an output, usually in the form of a PDF (portable document format) file.

You will notice that there is a TeX markup/programming language, a TeX program, and a TeX typesetting engine. These things were all made as part of a synchronous workflow, but there is also good reason to consider these elements of the workflow separately.

Outside of typographic professionals and programming professionals, most people use LaTeX, not TeX.

LaTeX is a set of macros written for TeX by Leslie Lamport in the early 80's (TeX had its origins in the 70's!). In LaTeX, a few simple lines of code sets you up with a document that would be much more complicated and challenging in TeX. You actually are using TeX in the background, and the typesetting engine is still a TeX engine; you're just using a few LaTeX commands to represent many more TeX commands. If little customisation is introduced to a LaTeX document, it will naturally resemble one of the base templates (that is: the default macros), but it is quite easy to customise the document from this baseline, and in many cases you will even interweave a few commands in base TeX language. Because LaTeX can streamline many parts of the process, we can focus effort where it belongs: on the text itself.

[1] I believe that, to this day, we have not surpassed the technical excellence of the typesetters of the 19th century.

People use LaTeX today because of:

- The typography is excellent
- The input file is plain text, which comes with a number of advantages. A few are:

- You can use markup and commands to indicate formatting, meaning that the content (that is: your writing) is actually separate from the presentation.
- Following from the above, the file is often more stable
- Being able to 'say what you mean', rather than going through a 'what you see is what you get' system, creates a lot clarity and precision.

- It is relatively easy to produce things like tables of contents, bibliographies, and cross-references. LaTeX is considered ideal for thesis-writing because of this.

Official Documentation, or Just Offal?

Knuth wrote a manual for TeX, just as he has written many books on computer programming. Some people have noted that this is not exactly for a general audience, because it teaches programmers how to typeset, rather than teaching the TeX programming language. However, we need not worry about these things, because we are using LaTeX.

Lamport wrote a manual, "LaTeX: A Document Preparation System". Even though it is from 1985, that book is, for the most part, viable. However, very few people seem to use it. It is barely ever referenced on forums, and I suspect that it has simply not been made accessible enough for it to remain at the top of the recommended resource list. The community has stepped in with other resources, some of which are written by people who have made excellent packages for LaTeX and have been deeply involved in its maintenance. In any case, I feel that the absence of 'official documentation' explains much about the travails of learning LaTeX.

3.2 A couple of Favourite Documentations

So, what has the community produced and made available for free? From what I can see, probably everything you need.

Memoir

One of my favourite contributions is the memoir class documentation. In my own mind, this is the official documentation for LaTeX.

Now, the document class is the emphvery first thing you tell LaTeX. It corresponds to the TeX macros that you are working with as your baseline. Memoir is the most popular and well-liked class for writing books and theses. It is the class I use, because it has great specifications for page sizes and margins, and these are things I need to consider when binding. The documentation for memoir is so good that is well-regarded as a general documentation for LaTeX by the broader community. However, some things in it are specific to the class.

The memoir documentation is in the repository, in the "Documentations" folder.

I like it because it is comprehensive, well-explained, sensitive to broader concepts and theories while still being detailed, does not make too many assumptions about your circumstances, and has excellent cross-referencing to other resources.

I advise you to look at the memoir documentation, but not necessarily to use it as your first starting point. I know that you are a wonderful, intelligent, trailblazing scientist, but one of the below recommendations will still serve you well as the first step; memoir is something to come back to in a bit.

A Not So Short Introduction

The "Not So Short Introduction to LaTeX" is a well-established reference guide and introduction to LaTeX. I like it because it acts as a reference book, but then does not skim over things too fast. It genuinely introduces you to LaTeX by carefully thinking about the sorts of things that new La-

TeX users would find useful. Consequently, it is indeed not very short — but that is why such things have a table of contents.

I recommend this guide as one of your possible starting points. It will provide you with a good introduction, good reference material, and it will help you to write concise code from the outset.

Overleaf

Continuing our journey to a good beginning-point: Overleaf. Now, Overleaf is not really a documentation source. It is a website that offers a free, online LaTeX editor, and has some pretty good tutorial/reference-guides available as well. I think that you will get on quite well starting with the overleaf editor if you do not have something like VS Code set up; without such a setup, LaTeX can get a bit overwhelming.

But my purpose here is to examine the documentation side of things. Overleaf has documentation, the first part of which is a tutorial ambitiously named "learn LaTeX in 30 minutes". This is a great place to start if you want to get going. As you move forwards, you should transition to also using things like the 'Not So Short Introduction', and the memoir class documentation. These documentations are essentially whole books at your disposal. They are more comprehensive than overleaf, and allow some more space to get around the theory in a structured way.

I like overleaf because it is aware of the symbiotic role it plays in the LaTeX community by supplying these beginning points.

`https://www.overleaf.com/learn/latex/Learn_LaTeX_in_30_minutes`

3.3 Summary So Far

- TEX can mean a typesetting language, program, or engine; or all at once (and by reading elsewhere you will discover much more nuance in the components of the TEX).
- LATEX is a streamlined way of working with TEX

- A well-made online platform, with documentation, like overleaf is a good place for you to start, but you should have more cogent resources on hand as well.
- The 'Not So Short Introduction' is excellent.
- The memoir class documentation is excellent, and together with the not so short introduction, will provide you with most of the resources needed for understanding LaTeX at a beginner level.
- LaTeX is an interesting case of a program where the closest thing to the official documentation is not widely used. This has led to a nebulous certainty about where one should go, and that is why I have written this guide. Some documentations provided in this chapter is suitable for beginners, yet are impressively comprehensive, written at a high level by people who have actually contributed to LaTeX.

3.4 Other Documentations

Chapter 4

Deep Roots

I have noticed that abstract concepts such as 'descriptive markup' can be learnt from just about any perspective, and stand you in good stead. Concepts and ideas are inherently good at being adapted to new terrain. If they struggle to do so, it is more likely that one's understanding is procedural or technical in nature. It may be good to learn things from a different discipline, or in a purely academic environment, so that in order for the concept to become applied, one has to draw on a deeper understanding.

Because of my views on epistemology, outlined above, I have included this chapter: it is a group of resources that allow you to contemplate foundational concepts, but away from the shadow of LaTeX.

4.1 Design

Font Design and the Design of Communication

For this topic, see the 2007 documentary *Helvetica*. It is available to watch at *watchdocumentaries.com/helvetica*. This documentary is also a valuable introduction to the different approaches to design.

A Further Comment on Design

In the LaTeX community, it is often emphasised that text is meant to be read, not to hang in art galleries — or some similar sentiment. That approach is a fairly functional one. Certainly, books do need to be read, and this places a weight on the functional traits of navigability and legibility. But after reading some of the guidance in the LaTeX community, one might think that serving such traits with a professional aptitude is all that LaTeX can do, and perhaps even all that is worth doing. Counterpoints to that view are *everywhere*. My favourite example is a book by Max Porter called *Grief Is The Thing With Feathers*. The text blurs the idea of paragraphs, and uses the positioning of text to as a method of visual communication. These are things that the *not so short introduction* would find peculiar. Yet LaTeX would be excellent for specifying indents and positioning. I have used it for such purposes myself. LaTeX is a freely available software that allows a high level of control over the typesetting process. The input is plain text as the input. Not only does this allow the process to be explicit, rather than being done by feel alone, with the ensuing risk of haphazard work — it also lends stability to the input file and removes dependency for understanding it from any particular piece of graphic-design software. While LaTeX is a typesetting program, and will be strongest when working in a traditional typesetting context, its possibilities should not be overlooked.

Typography

Following from *Helvetica*, you may find that the importance of typographic design has been illuminated, but you will not be any wiser on how to go about the nitty-gritty. *Butterick's Practical Typography* is one resource to help. It is available at: . I find that it pushes some opinions, just as the LaTeX documentations tend to do — but at least you have some varied sources now.

One point of difference I noticed is the approach to the spacing after full stops. The LaTeX community seems to be fine with the approach of there being a double space after a full-stop, while Butterick goes in for single. I like single, but am more or less agnostic about this matter. If you *really* want to see the metaphorical ant-nest swarm, tell the LaTeX users what many designers say: that justified text looks awful, and we should embrace ragged text as a standard. The ensuing conflict would probably bring up the requirements of double-columned journal articles, or densely-packed books, in opposition with a notion of ideal design. It is ironic that most LaTeX-produced documents are being distributed digitally and are somewhat free from these space-limited environments. Being an enlightened soul, I use both types of paragraph formatting, depending on the work.

This is a good time to mention that the LaTeX community is one that works with text a lot. They are scientists and writers and coders, and they have to say a lot of things with text — even more so if you include mathematical equations (try putting *those* in an audio-book!) Donald Knuth wrote LaTeX to reverse the declining of quality of typesetting that was affecting academic articles and books in the 70's. The LaTeX community is still — for the most part — the target audience envisaged by Knuth, a legacy preserved by the fact that these people have been writing most of the packages for LaTeX. But that is just the thing: they are scientists and writers and coders, but not always designers, nor typographers, by trade — even though they have usually achieved impressive skill in these areas.

As mentioned, there were some typography issues around the 70's. Typography wasn't in decline, as such, but the introduction of new digital tools was creating some very poor results, and it had to be fixed (apparently). Knuth was not the only person to tackle this problem. The program *Scribe* also came out in the early 80's. (note: Interestingly, between scribe, Latex, and generalised markup language, the idea of descriptive markup absolutely exploded in the late 80's). We seem to have gotten past the proliferation of terrible typography, now, but LaTeX remains an excellent-quality typesetting program, with a number of virtues besides.

Structured hierarchy of objects

The idea that text has a logical structure is implicit in LaTeX. Such ideas can be challenged, and should be when they are presented with too much rigidity. But it is a rather interesting thing to say, I think. Who said that text had a logical structure? I write illogical things all the time. The observation that text has a structure is a well-evidenced, though.

The first text-book on accounting was written in 1494 by Luca Pacioli, a mathematician who helped to develop perspective painting. It has many short chapters. The 'text-book' is itself a part of a larger series on math and geometry. Each chapter speaks to a topic, and none of chapters carry on to another part of the series; they are nested within the books. In turn, paragraphs are nested within chapters. This is all sensible enough. But I bring up Pacioli because he wrote thirty-seven chapters in his book, many shorter than a single A4 page in a modern equivalent. He evidently thought that structure and partitioning was important. I would say that 'in the old days, people couldn't get away with sketchy structure'; however some ancient poetry transcriptions are pretty sketchy, so saying that would be misguided.

Anyway, text tends to structure, and this is articulated in a functional way for the modern world by an article called "What is Text, Really", which is in the repository, in the 'articles' folder. The article outlines the usefulness of the 'structured hierarchy of objects' model of text, which is usually what people mean when they talk about 'logical structure'.

People say that LaTeX encourages structure. It does, but you can also get away with bad practices if you want, and in many cases, the structure is only implied, rather than explicated. In something like XML, the structure is often (but not necessarily) more explicit.

4.2 Descriptive Markup

Descriptive markup is another foundational concept, but I do not have a source on hand that goes into a deep rigorous discussion on the idea, so that you can mull it over. One possibility is to look for early articles on SGML (standardised generalised markup language), which was the most comprehensive application of the idea.

Recall the idea that a printer's markup of a text might use something like an underline to indicate italic text. This is a surprisingly nuanced idea. Underlining is a common way of indicating emphasis in handwritten texts. Emphasis is usually communicated in printed texts with italics, so it makes sense that an underline means 'make this italic'. However, the underline could also just mean this word is emphasised — the decision on exactly how to render the emphasis could come later. This discussion shows how markup can be used in different ways. The italicisation of a text is a procedure, and the markup to indicating italics can be called 'procedural markup'. The style of markup that simply says 'this is emphasised' is more descriptive, and can be called 'descriptive markup'.

Markup is a broad concept, and the different kinds can be hard to tell apart, at times. However, it's worth noting that they all differ from the 'What you see is what you get' (WYSIWYG) approach, which is used in things like microsoft word. If you italicise a bunch of words in a WYSIWYG processor, all of that information can be lost by something as simple as transferring it to a computer with a different processor or font. However, if the italics are indicated using a markup language, written in plain text, you will not run that risk. The file is much more stable and the formatting is explicit. This approach allows more precise control and is sometimes called 'what you mean is what you get'. An even more powerful approach is to use a markup where different *kinds* of italic words are communicated. For instance, you might want to know that some words are italicised because they are scientific species names, while others are italicised for the purposes of emphasis. This line of thought will lead you towards descriptive markup. If you develop a markup that can be used to describe anything at all, that would be a generalised markup.

LaTeX can be quite descriptive, but in reality it tends to have a procedural flavour. After all, its purpose is to perform the typesetting activity. Markdown is an interesting case where it is sort of procedural, yet in the context of simple document writing, it could be seen as descriptive for a very narrow range of concepts.

Descriptive markup leads you quite quickly to making clear where there are paragraphs, rather than just paragraph breaks, and sections, rather than just section breaks. Doing this adds a sort of comprehensive clarity to a text. At this point, I have arrived back at the idea of a structured hierarchy of objects. Descriptive markup often specifies the structure of the text.

Ironically, descriptive markup has provided one of the challenges to the OHCO model. If I begin to describe a text, I might not know how the sections are organised at first; it might not even have sections at all. Encountering a blank line-break, I may indicate this feature without knowing it is a section-break. What I am doing is describing the features of the manuscript, rather than the conceptual structure of the text, so that I can better understand or examine it. I am looking at the text in a different, but entirely valid way. By mixing different approaches to description, I could run into trouble. For instance, I could be describing physical lines of a manuscript, and also sentences — such an activity would be reasonable for poetry. But lines on a page and sentences are structures that do not nest with each other! The OHCO model no longer holds because I mixed approaches too much. Some of the issues with the OHCO model are discussed in an article, TK. As this article points out, the TEI (text encoding initiative) schema of XML — which is probably the most comprehensive text markup schema anywhere — implies structure but does not explicate it. I consider this ambuguity to be a testament of how theory and application are always holding each other to account.

Chapter 5

Online Resources

Note: Most things are available online now, but what I mean by ’online’ is that the thing is *online in character.* A good example of online in character is the W3schools tutorials, which are quite interactive; another is the vast swathe of less structured help one can find on forums. In contrast, a PDF like the memoir documentation can be found on the web and viewed digitally, but it is really just a piece of structured text, similar to what is often seen in print.

I have drawn a somewhat arbitrary line and put stable texts in the documentation pile, and more interactive or changeable websites in the online help pile.

5.1 A Few Words On Online Help

I’m of the view that LaTeX has been around long enough for things to get messy. Some features of LaTeX are no longer supported, by which I mean there are packages which are not being updated and are starting to not work with other packages. Furthermore, people are using LaTeX in different ways: different machines and operating systems, different approaches to off-line and on-line work, different approaches to fonts, and so on. This is all well and good until a ’solution’ makes an assumption about what you are doing. Good solutions are those which are always true, or have their realm of truth well-framed, even if that means they are a little less user-friendly. If you look at the popular forum *Stack Exchange*, the best answers are those which are both explicit and robust, not necessarily those which are short; and they are rarely the sort of answers that begin with “Just use [...]”, for those answers are often making assumption that instantly breaks down - often this is the very reason the problem was posted. The whole mode of online help struggles with always having to assume a certain amount about someone’s knowledge or experience.

The above reasons contributed to my judgement that websites and documentations have to be considered as different types of resources. A good documentation is worth its (figurative) weight in gold; even if you don’t read documentations front-to-back, you still want them on-hand. Websites are often good but a generic ’got to this forum’ means you have to sift through a lot of bullshit. Furthermore, many websites cited as being helpful assume that you have read official documentations - despite not providing clear references or pathways to finding said documentations.

So, my approach is to curate online help and record useful things here - see the below section titled ’Online Help Links’.

For documentations, I have taken the approach of discussing these above, as well as downloading the pdf’s and providing them in the folders of this repository. But it is also important a pathway to where these were found - see the below section titled ’Links to Documentations’.

Finally, I have tried to outline what resources are the most ’official’ LaTeX help in the online realm - see the section titled ’Official Help’.

5.2 Online Help

The LaTeX Project

This is also a base for LaTex. It is the official website of LaTeX in that it is the website of the project responsible for maintaining the entity of LaTeX.

$https://www.latex-project.org/$

However, the project funds are actually administered by the Tex Users Group.

Tex User Group

https://www.tug.org/

This more informal looking website is — I understand — the most official base for LaTeX and TeX that there is online.

The Tex Users Group is the administrator of the LaTeX Project.

The Comprehensive Tex Archive Network

https://www.ctan.org

Example of things you would go to this site for are fonts catalogues, documentation downloads, and information and mirrors for packages. It functions as a kind of repository for the TeX Users Group.

e.g. https://www.ctan.org/pkg/texbytopic

A side note on Documentation state

> [IMPORTANT: The below statements are nullified. Leslie Lamport wrote a manual but it is strangely very rarely referenced - possibly it is not written in an accessble way, or the book itself is not very accessible?!]

It will be noticed by navigating this page that LaTeX is not obviously documented. I am surprised at this, but it does appear to be the case (although I am now reading through an auxiliary documentation that *must* refer to the core documentation, to check). The lack of documentation explains much of the disarray of the community help.

I am not the only person to raise this. A question was posted on Stack Exchange and did not yield an answer:

https://tex.stackexchange.com/questions/47958/where-can-i-find-the-standard-latex-reference-manual

But this thread highlights the ambiguity in a more positive way:

https://tex.stackexchange.com/questions/66/which-manuals-are-on-your-tex-reference-shelf.

I am pleased to say that the original poster mentioned the documentation for the memoir package. Upon discovering the memoir documentation, I also found the quality so high that I adopted it as the official reference in my own mind.

LaTeX2E Unofficial Reference Guide Website

Not yet examined.

https://mirror.aarnet.edu.au/pub/CTAN/info/latex2e-help-texinfo/latex2e.html

Stack Exchange

https://tex.stackexchange.com/

When I say ’forums’, Stack Exchange is usually the one that I mean. It is exceptionally useful, but more so when you take a pocket-full of salt on your journey, and have good foundational resources on-hand. Without already being some way along, you will not be able to detect when answers are talking to a different problem than the one you have, nor will you be able to understand the full implication of the code that is posted. The best situation is where you already have a good understanding, and just need a slight shift in perspective to help you see the missing piece in your approach.

Stack Exchange can also be a source of code-snippets, particular in base TeX language, that you do not necessarily need to understand in order to use.

I provide a number of resources that I have found on Stack Exchange in chapter TK.

Overleaf

5.3 Links to Documentation

5.4 Official Help

Chapter 6

LaTeX at Work

6.1 LaTeX Applied in a small non-technical workplace

LaTeX is sometimes said to be a solution for writing theses and large documents. It is, but I am able to show with these two examples that it can also prove useful in non-technical workplaces for small documents.

Pocket-journal Copy

In 2021, I made a pocket-journal for Saddler & Co. I had also been a writer with the same business, so I collaborated with the creative director to produce website copy, an email launch, and a letter that introduced the collaboration. The letter and was sent out with the journal to each customer. A traditional word processor was used to create the copy, which was then exported as a pdf so that there was a stable version. The launch occurred just before Christmas, and we sold forty of the first batch of fifty within 48 hours.

I moved interstate soon after, but when I was settled I was contracted to produce another batch of journals. In the intervening time, I realised I wanted to make some changes to the letter, which I would print on my own special paper-stock and send to Saddler & Co with the journals. One problem was that my version of the word processor didn't have the initial font. I was working with an automatic equivalent for some time before I realised that some of the formatting had also changed. Somehow, the spacing had been changed on my version of the document. My first printout looked bad because the font was too small, but when I picked a better option, the spacing issues made themselves known, and I found that the letter spilled over two pages rather than the intended one page.

To solve the formatting issues with the document, I took the text and put it into Overleaf. I chose a standard LATEX font, which improved the appearance and meant that there would never be a case of accidental incompatibility. I knew that the font was standard because you need to use a specific kind of code to chose non-standard ones; also, the name of the font was written right there in the code, making it very explicit what I was using. Then, I explicated the line-spacing patterns and page-margins. Overleaf was good because it was online and shareable, yet with the plain-text input there were no issues with the online version of a word processor altering the fonts and such.

I gained all of this benefit in less than an hour, despite only just starting with LATEX and employing several bad practices. Furthermore, the LATEX document looked better, which is just a matter of the standard fonts and kerning being good. I used the overleaf documentation to help chose complimentary fonts. These problems could have been solved in a normal word processor, but after a few unpleasant false-steps, I felt that LATEX would prove a more reliable option with a bit of initial time-investment. These days, I would find the formatting quicker to do in LaTeX in the first instance.

The LaTeX code and the pdf for the letter are in the repository, in the folder called 'Customer Letter', inside the folder called 'LaTeX Examples'

Nice Looking Inventory Sheets

My primary role at Saddler & Co was QC (Quality Check) officer[1]. This role determines what items enter the inventory. I began using some datasheets that recorded the flow of items through the business with more precision. The tables were dense and accompanied by other kinds of information

[1]It used to mean Quality Control, but we decided that quality is something to be made, not controlled.

on the same page. To make sure the tables were easy to navigate and write on, I used LaTeX. This helped reduce paper-use by having dense tables that were still easy to navigate. The tables were also more visually pleasing, and without this I am not sure that they would have been able to reduced inventory errors in the way they did.

I gained all of this benefit because TeX was designed as a typesetting system for academic work, whereas microsoft word is more of an on-screen word processor with functions such as tables tacked-on over time. The many measurements involved in formatting tables means that LaTeX, with it's very explicit measurements written in the code, is often a good choice if you need to reduce the space used while retaining legibility. It did take me a while to do some of the data-sheets though, and I used many bad practices.

I have provided the code and pdf output for the simplest LaTeX tables I made. It is in the folder called 'Inventory Table', inside the folder called 'LaTeX Examples'.

Chapter 7

LaTeX Workflow

LaTeX can seem sort of daunting, but it is really quite smooth once you get going. It is a bit similar to using R. If you are not familiar with the idea of using code, it will feel difficult. But once you become familiar with the virtues of replicable analyses[1], have gotten used to having well-organised work repositories, and have found an IDE (integrated development environment) that you like (such as R studio), it is hard to imagine doing it another way.

Below, I briefly look over

- Installing latex.
- Some features that will be important for your workflow later on (unfortunately I do not really understand many of these)
- How I write documents and use Latex now. This forms a sort of example of one way of doing things (although it will not be optimised.)

7.1 Installing LaTex, and Some Core features to note

Distributions of LaTeX can be found on CTAN. Recall from page 23 that CTAN is associated with the Tex User Group, which administers the LaTex

[1]Also gained in LaTeX, as I have tried to show

Project. So, you can really go to any of these places, but I suggest going to the LateX project, and downloading whatever suits your operating system.

URL

Now, usually you will end up with a TexLive, plus some other tools that help you use LaTex on your operating system.

TexLive is important because it is is the recent up-to-date TeX stuff. Want to use packages? Want to use any font newer than the early 80's? Want to use different languages? All of this stuff required updates to LaTeX, and that is why TexLive is important.

One of the major things you get from TexLive is the different Tex programs/engines. TeX predates the compatibility of fonts and character encoding in the digital realm. It is wise to use one of the variations of the TeX program that was made so that TeX could make use of modern fonts and encodings. This situation is described at length in the 'XeTeX companion'. XeTeX is one of the engines that allows modern fonts and languages. It is the one I always use. TeXLive comes with a lot of fonts, so you will be pretty set up with options straight away. I describe how I got about selecting fonts a bit later.

The distribution should also come with a program where you can enter plain text and tell the engine to produce an output document from that input. On Mac, the program is TeXShop. This isn't a very integrated environment, but it has a key function, which is that you can select the engine you are using. This is important because some key parts of your code will change depending on the engine.

Starting off with something like TeXShop, you will be bombarded with auxiliary files. These are outlined in the Not so Short Introduction. I will say the the .log file is particularly handy. These files can be handled by organising your digital space a bit, and by setting up an IDE. This is important when you get start inserting images and such, anyway.

7.2 How I use LaTeX Now

How I Write Text on a Computer

LaTeX is now easier for me to use than any other typesetting option, and I have not been using it for long. Until recently, it only came into my life for specific purposes, but now I am happy to use it for regular work.

There are plenty of reasons for using LaTeX, but there is still a case for just writing simple documents with a word processor. I still do exactly that, much of the time. I sometimes like to use text-edit and make rich-text files, because the program is minimal and allows me to produce black text, in my preferred font, against a white back-ground. I like not having lots of code in the text, and the WYSIWYG mode acheives that.

However, I always tend to switch over to LaTeX now. When a work becomes more complete, the LaTeX pdf will look even better will be good way to reflect on it. Furthermore, there is less and less code in my text. LaTeX is pretty minimal for basic text, and the preamble is increasingly short and concise. It takes so little time to render something into LaTeX that I am starting to do it sooner rather than later, and it means I can easily slot in a table of contents by doing something as simple as this at the top of the document:

```
\tableofcontents
```

Definitely, as soon as I step away from simple text and into formatting, I must use LaTeX. The frustration of writing poetry and sorting out the indents for this bit of whole text, and indents for just the first line elsewhere, and indents of this kind for the dot-points... It will drive one mad. It is much better to do things like this:

```
\begin{verse}
    No words were laid on stream or stone,
    when Durin woke and walked alone.
\end{verse}
```

At the end of the day, I find that I can simplify things, either working in hyper-minimal environments and getting the benefit of those, or working in LaTeX and getting the benefit of that.

The Small Things That Make It Work

The ease of writing in LaTeX is certainly catalysed by the resources available to me. Obtaining some equivalency of what I am doing — or looking for something even better — is recommended.

I work with a program called VS code. It is a free IDE for coding work, and it has *tons* of extensions. Here are some of the key features I gain:

- I can organise my files/repository in the same program that I do the actual work in;
- I can install excellent TeX extensions that have full functionality, like the selection XeTeX as my engine.
- I can view pdf's in the same program. I can even organise it so that I have two different tabs: one with the TeX file (which is the plain-text input file), and the other with the preview of the pdf. If I edit the input, I can update the preview without even having to close it, or change the part that I am viewing.
- I can install extensions that highlight issues with the TeX code, allowing my to maintain it easily on-the-go.
- I can link the project to Github, simply pushing it to the online repository whenever I wish, and working in the normal manner otherwise.
- I can fold the sections of my document using comments in my code (these don't affect the output in any way), so that even a very large document can be collapsed down to only a couple of lines showing the main parts.
- The TeX extensions also have some built-in functionality that lets me navigate the document structure easily.

- I can use the regex[2] feature of VS codes search-and-replace functionality to help highlight words, or convert markdown documents to LaTeX.
- I can use VS code for coding activities that might be related to the document being written.

Any good IDE will give you some similar benefits. It does not need to be VS code, and it is perfectly sensible to use different tools for different things. My overall finding is that setting up your workflow will make things exponentially easier compared to my own initial experiences with LaTeX, saving you time and reducing stress[3].

Pandora's Box

I find the digital realm frustratingly arbitrary. It seems like the sort of place that is filled with acronyms, many of which people don't even have to know the origin of in order to be skilled. I can make some progress, but for the most-part I struggle with not investing my time in proper learning, and feeling that people are always assuming some unreasonable, overly-arbitrary array of initial knowledge. In many ways, this guide is an attempt to impose order and a situation that did not seem to make sense. Now, as you know, learning the basics of how things are operating is incredibly liberating, and without embracing that, we probably wouldn't even realise the usefulness of something like R.

In the realm of LaTeX, a thing that is likely to drag you into the lower levels of your operating system is fonts. In order to specify fonts in LaTeX, you need the name of the font as saved on your computer.

```
\documentclass{memoir}
\usepackage{fontspec}

\defaultfontfeatures{Scale=MatchLowercase}
\setmainfont[ItalicFont={QTSchoolCentury-Italic}]{QTSchoolCentury}
```

[2]'Regex' means 'regular expression', a syntax for creating conditional statements

[3]A very similar thing can be said for having a good piece of documentation on hand rather than a bad one, or none at all.

In the above code, QTSchoolCentury is the name of a font in my system (I do not need to enter to entension). There is not a standardised name for fonts, as such, unless you are working with standard TeX-fonts.

TeXLive comes with a lot of other fonts, and you can also download you own — but how do you tell LaTeX what font you want if the program is not already familiar with the it? It turns out that even if you know the file-name for the font, LaTeX could still be looking for it in the wrong place.

I found out that when using LaTeX (XeTeX, to be precise), the program was referencing the system folder for fonts, while the TeXLive distribution had installed the fonts within the usr/local folder, which is not usually visible, because it contains sensitive files that should not be edited by naive users. I now keep the fonts that I want to use in the system's font folder, because that seems like a sensible place to reference fonts from. The other solution is to alter where LaTeX is looking for the fonts, which takes a bit of code.

TeX is good in the sense that you do not need to go very deep as far as programming skill goes. Writing the actual TeX documents elegantly is a far more important skill than programming aptitude. At the same time, you will only benefit from carefully isolating and solving a technical problem. Solving problems should be done with access to good-quality documentation. A day in the library saves a month in the lab, as the saying goes (although sometimes the lab-practice might be just what you need).

All that said, it's good that XeTeX exists, or else you have to get through all of this:

https://tex.stackexchange.com/questions/202789/where-to-find-the-family-code-for-a-custom-font

7.3 Good Workflow in LaTeX Code

The Structure of LaTeX documents (that is the plain text input file, which is usually saved as a .tex file) is as follows:

```
    Preamble
    Marked-up Text
```

Remember that the mark-up in LaTeX is presentation-focused, and it is separate to the content:

```
\emph{Hello, World.} = Hello, World. + [emphasise this text]
```

= *Hello, World.*

The marked-up text can include all sorts of formatting commands as well, but if you consider these as formatting instructions, they are an extension of presentational mark-up.

My first LaTeX documents were just a mess. Now, things are somewhat better. The reason for this is that I am more familiar with writing the actual LaTeX code. When I say 'LaTeX code here', I mean writing the preamble, and marking-up the text. Both of these involve using the LaTeX coding language, as can be seen in the command syntax being used to do things like load packages in the preamble, and to emphasise text in the document.

```
    \usepackage[]{}

    [...]

    \emph{}
```

Generally speaking, you want to reduce noise in your text. If a rule can be specified in the preamble rather than the text, that is usually the better option. However, you will then want to make sure that your preamble is elegant, tidy, and powerful.

The rest of this section is a sample of clarifications and tips that I have found useful for writing better documents.

The very first lines of LaTeX code in a document

When using overleaf, you will see that each document has the first bit of code filled-in.

```
\documentclass{article}
\usepackage[utf8]{inputenc}
```

This first line of the preamble are foundational because **the document class is the set of base TeX macros that you are working with**. The article class is one of the originals. I usually use memoir. Whenever I use one of memoir's commands, I am actually invoking a set of TeX code that the authors wrote.

The input encoding is specified, because even if it is usually assumed, mutually intelligible encoding is the thing that allows you to tell the engine what to typeset. UTF-8 is designed to facilitate working with all the world's languages, and that is what we want when it comes to typesetting. Not sorting out one's encoding is fine, until one tries to quote something from a foreign alphabet, and then it all falls apart. Overleaf probably has multi-language use in mind when they decided to include the encoding — and it is not bad practice, anyway. Closely related to the input encoding is the font encoding, which takes care out the output. This is explained in a post here:

https://tex.stackexchange.com/questions/44694/fontenc-vs-inputenc

Some of these encodings are starting to look pretty old. When UTF-8 came along, things gradually changed. That is why when using XeTeX, the input encoding and font encoding are usually not specified, and one uses the fontspec package instead.

Packages and stuff

Packages provide more TeX macros, massively extending the functionality of LaTeX. These seem to be a part of the TeXLive distribution, and they usually work without clashes, despite working on overlapping elements of the typeset. It is certainly important to be careful with the order in which packages are loaded, and which ones you use, but I think the laTeX is quite well managed. LaTeX also seems to have fewer specialised packages like those seen in R.

Some useful packages are:

- tabularx
- graphicx
- ragged2e
- geometry

The geometry package is a common example of a clash. It specifies things like page size and margins. It is often used with the article class. However, the memoir class has these specifications as part of its standard options. Without referring to the documentation for memoir, and adjusting method accordingly, one's once-reliable method for customising page-size could fail.

The ragged2e package is interesting because it is not usually a clash, as such, but can lead to confusion about commands. In the normal classes, the command \raggedright will make the right side of the text ragged. However, radded2e introduced the command \RaggedRight. The ragged2e option is less ragged than the standard, because it is willing to hyphenate words.

The ragged2e package, to my mind, brings up the matter of clean code. If the package is loaded, but the commands are in lower-case, then it only causes confusion: did the author of the document actually want to use ragged2e? By removing the package, they may be able to clarify that they are fine with the base functionality. Alternatively, querying this matter might alert them to the fact that they meant to use a different command in some places.

Good Code Habits

This thread has some good thoughts on the topic:

I certainly agree with the problems with large preambles, and the habit of ending paragraphs with \\, which I used to do a lot.

The Not so Short Intro and memoir documentation will help you become a concise TeX coder, both through direct instruction, and indirect influence. I think I spent too much time cobbling it together with Overleaf and forums, which does not lead to such a cohesive view of LaTeX — but such things can happen when one works in isolation.

Stack Exchange is very useful — probably essential — these days. However, it can still be hit-and-miss, as I have outlined in chapter five. One part of the LaTeX culture I don't like is the reductionist motif of saying 'what you're doing is against the spirit of LaTeX', without providing a viable solution to the problem. The 'spirit of LaTeX' that people are referring to is that you specify typesetting rules, but then focus on the content (that means you don't perpetually fiddle around with the formatting). Since being a good LaTeX coder is independent of both typographic skill and of one's excellence as a writer, I find this approach stupid and patronising. The broader community thinks so, too: the answers that win-out on stack exchange often express the 'spirit of LaTeX' as a disclaimer, before providing a truly considerate answer.

At the end of the day, you are trying to write a good document, and 'good' is subjective. Just because some code seems awkward or unusual, that does not mean you are doing a bad job. In fact, your clunky code could be the first step in a process that ultimately produces excellent work.

Bits and pieces

Absolutely stop a page-break

See the code given in the top answer in this thread:

https://tex.stackexchange.com/questions/94699/absolutely-definitely-preventing-page-break

Although the top answer is simple and only provides code, it was appropriate for the context.

TeX is designed to calculate the typeset and decide where to put the breaks. Therefore, specifying exactly where breaks and hyphens go is pushing against the grain. There are many solutions to this challenge. Line breaks are discussed later.

Strange spacing on a page

LaTeX alters the spaces between words so that each line is the same width in justified text. Similarly, it considers vertical spacing of pages elements. It doesn't like one page to be nearly empty, and the next very full. In some documents, it will try to space out elements to fill up the space. Sometimes this looks awful. The human eye usually doesn't have an issue with a page ending short, so long at the elements are in the normal arrangement until that point.

I have encountered this problem, and all I had to do was use \raggedbottom.

Formatting in a particular area

Some commands take arguments, allowing you to — in effect — markup the text so that it becomes the argument for a command:

```
\emph{Here is a text.}
```

Other commands change the rules in use for the document and affect whatever comes after.

```
   This text is in the main font.

   \EBGaramond This text is in EB Garamond.

   I defined this font in the preamble.

   \normalfont This text is in the main font, again.
```

Note that there is a single space after the command. This is not rendered, because it is to separate the command from the content. Using empty curly-braces can end a command without using a space. Multiple spaces are collapsed into one in LaTeX, so I sometime fine the curly braces less confusing

An environment can also be specified.

```
\begin{center}
    Welcome to the center.
\end{center}
```