R for Newspaper Data

Yann Ryan

2020-01-03

Contents

Chapter 1

Preface & Acknowledgements

I spent the last year or so working at the British Library as a Curator of Newspaper Data. It was an experimental position, conceived on the premise that the *data* generated by newspaper digitisation projects, in this case the *Heritage Made Digital* project, deserves its own curatorial role and set of practices.

It was an exciting time to work on such a project while 'big data' analysis using newspaper content, by historians, is no longer in its infancy, it is firmly in a soul-searching adolescence, and it's been exciting working on an evolving discipline, one which is developing its own set of practices, understanding where and how its bias works and the implications this might have, and generally moving from a naive optimism to being very academically rigorous and self-reflective.

The British Library was a wonderful, inspiring place to work, and over the year I spent a lot of time getting to grips with the extent and wonderful complications involved in getting access to and analysing the data that the Library has (and elsewhere). This book, then, is meant as something of a roadmap - we are not quite at the final stage of the journey but great steps forward have been taken. The only way I know how to get there is through a programming language, and specific examples.

This book is based on my experiences in this position, as well as some from my own PhD work. It's conceived as somewhere between a 'how-to' and an inspirational (or aspirational) piece of writing intending to survey and encourage the use of newspaper data. It does not cover everything, partially because of my own shortcomings and partially because newspaper data is a vast, incomprehensible and ever-expanding field. I am grateful to those who have taken the time to clear the dense thickets (or perhaps sweeping for mines is a more suitable metaphor, given the risks of failure), including Ryan Cordell, Jo Guldi, Bob Nicholson and

Paul Fyfe.

I owe a huge debt of gratitude to the Library, and in particular the Lead Curator of News and Moving Image, Luke McKernan, for his help and support during my time, as well as the rest of the News Collections Team, Beth and Stephen, the Digital Scholarship team, and the wonderful team working on the *Living with Machines* project.

This is a book about how to find, download and use this newspaper data. It has been written, in part, by an employee who worked at the British Library, but it does not represent their views in any way.

Chapter 2

Introduction

2.1 What is newspaper data?

Difference between content and data? It's arbitrary, all of the digitised content could be considered data. Perhaps it takes on that name when we do certain acts on it? For example the images become data

Get a nice image for here

2.2 Why is it useful?

There is a *lot* of newspaper data available now for historical researchers. Across the globe, Heritage Organisations are digitising their collections. Most news digitisation projects do OCR and zoning, meaning that the digitised images are processed so that the text is machine readable, and then divided into articles. It's far from perfect - we'll show some examples in a later chapter - but it does generate a large amount of data: both the digitised images, and the underlying text and information about the structure. Once you get hold of this data, the rewards can be huge: looking just at English-language users in the last few years, researchers have used it to understand Victorian jokes, trace the movement of information and ideas, understand the effects of industrialisation, track the meetings of radical groups, and of course understand more about the newspapers themselves and the print culture that surrounds them.

While there has been a lot digitised, their is much, much more still to be done. The collection, in any country, is far from representative. But we must work with what we've got. The new histories of the press will be written by looking at text at scale, drawing broad conclusions, understanding genre, authorship and so forth through data analysis.

We're just at the beginning: in the last few years projects have been using neural

networks methods to improve the kinds of things we can do: the Living with machines project, for example, or several projects at the Royal Library in the Netherlands. The methods I describe here are simplistic, but they can still add to our understanding.

// # Data - adds to statistics, gives another point of view. Another way of thinking about historical fact?

// #

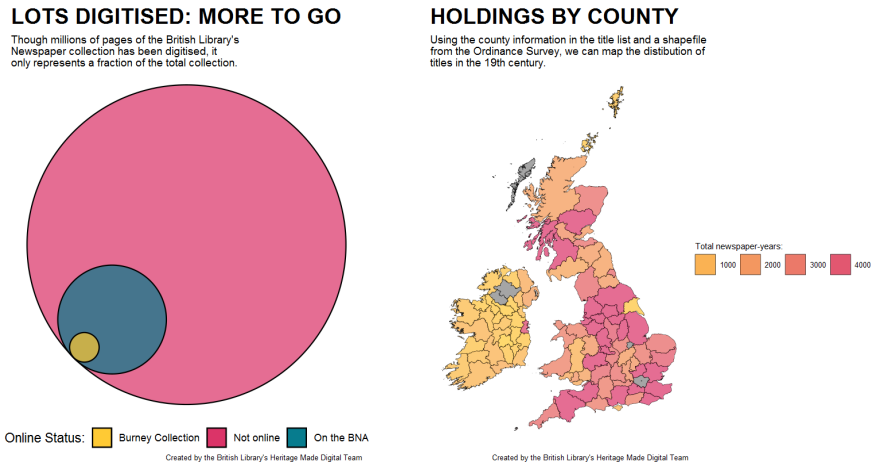


Figure 2.1: The portion of British Library Digitised Newspaper Content, and the physical collection, mapped

2.3 Short history of newspapers, newspaper digitisation?

2.3.1 Burney and EEBO?

2.3.2 JISC 1

2.3.3 JISC 2

2.3.4 BNA

2.4 Goals

Hopefully, by the end of this book, you will have:

- Know what newspaper data is available, in what format, across a variety of countries and languages.

- Understand something of the various XML formats which make up most newspaper data sources
- Have been introduced to a number of tools which are particularly useful for large-scale text mining of huge corpora: n-gram counters, topic modelling, text re-use. Including some specific to news, such as the R library *Newsflow*.
- Understand how the tools can be used to answer some basic historical questions (if not provide any answers)

Historians have used newspaper data to do x and y. Newspapers have long bee thought of as a proxy for public opinion, a historical source or 'first draft of history', used to study the movement and genesis of knowledge and information, help to understand the mix of private and public, how power can be advanced through news, and so forth. Some people might not even be interested in the news in its own right: how it was created, distributed, shaped, what the political pressures were, how it was read at different times, and how it came to shape those who read it.

What do you need in advance? Probably a mix of coding skills etc.

2.5 Format of the book - bookdown and github

Will try and explain as much as possible, but will take shortcuts Not a programming expert, so it may not be optimised, the best of way doing things. It's just the way I've found that works for me. Part of the reason for doing this is because writing in bookdown and GitHub is quite geeky fun. I will unashamedly do things because I *can*, on occasion.

Bookdown allows for code and figures directly in the text. I'll make a very simple dataframe and plot it.

First load a couple of packages:

```
library(ggplot2)
library(dplyr)
```

Make a dataframe:

```
a <- c(10,20,30,40)
b <- c('book', 'pen', 'textbook', 'pencil_case')
c <- c(TRUE,FALSE,TRUE,FALSE)
d <- c(2.5, 8, 10, 7)
df <- data.frame(a,b,c,d)
```

Plot it:

```
df %>% ggplot() + geom_point(aes(x = a, y = d, color = b), size = 5)
```

There are bits of Python throughout, where I've only managed to work something out using that language. This is done using `Reticulate`, which allows for Python

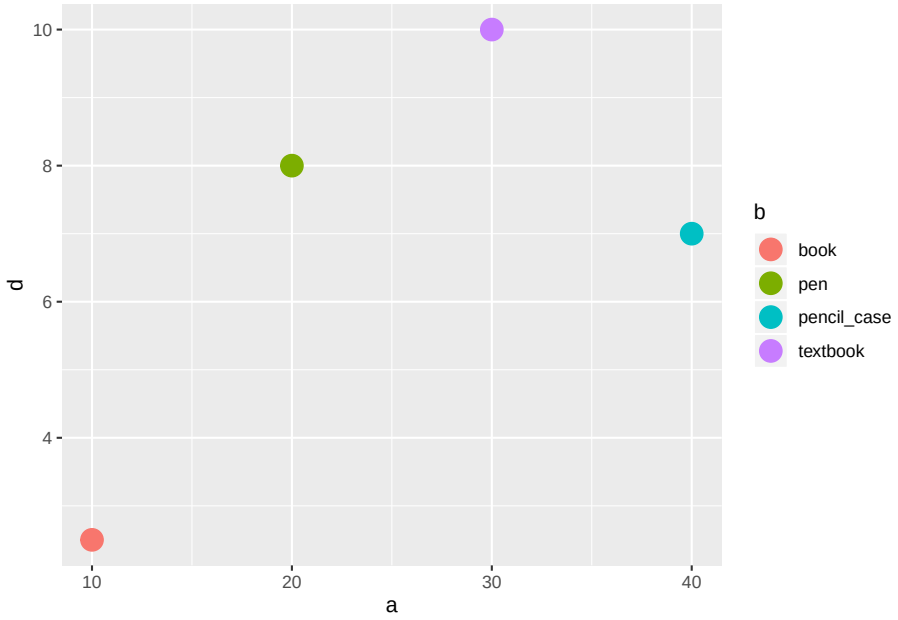


Figure 2.2: plotting example

within R markdown. I hope I'll be able to revise at some stage and do everything through one language.

How did I write the book? In R-studio, bookdown and bibdesk There are a million things I could and would like to explain detail.

There are a couple of things you'll need in order to build a copy of the book yourself. This may not be important to you. One thing that needs to be done is entering a bing API key for the relevant chapter. The book won't complete without this.

2.6 Why R?

Used to be idiosyncratic, is becoming very widely used by data scientist, digital humanities, social scientists. A lot of this is because of developers like Hadley Wickham and R studio - the tidyverse, but also data.table.

2.6.1 Who uses R?

2.6.2 The Tidyverse

Historians using the language: Sharon Howard, work on criminal tattoos. Bruno Rodrigues

Writing a book with such a specific goal is a bit of a weird proposition

2.7 Sources

2.7.1 Country by country

2.8 Methods

2.8.1 Network analysis of seventeenth century

2.8.2 Mapping

2.8.3 Geocoding

2.9 Text mining

2.10 Format of the book

The rest of this book is divided into three sections: * Data, which goes through the availability of various news datasets, for the moment with a heavy emphasis on the UK * Methods, which talks through some of the techniques in a broad sense * Examples, which uses the techniques above to actually do some stuff.

Chapter 3

How to use this book

3.1 What is this?

This might look like a static book but in fact it's a living, breathing document. It's actually written using a programming language. The language is called R. It's an open source language that was originally used for very specifically statistics-heavy work, but it's become much more general in the last few years. Part of this is in thanks to R-Studio, which is an IDE for R. This book is written in R-studio using a language called R markdown. This is a way of writing and formatting text like a word processor, except you use code to specify what you want and the program decides how it should look and renders it, rather than moving around the elements you see on front of you. All the plots you see are not images, but rather real bits of code, which run every time the book is rendered, and produce the relevant plots or tables. It means that all the code you type *has* to work, otherwise the book can't render. That's a pretty sweet way of making sure everything is correct!

You can just read this book as is, but you can also play around with it, change the code and make your own copy, simply by downloading R-Studio and a bunch of packages to go along with it. You'll also see that with a little bit more work, you can clone a copy and make changes, which can then be *suggested* as changes for this version, or just kept in your copy.

3.2 Make changes directly to the markdown files on Github.

You don't actually need to download and use R-Studio to make changes to the book, but it does of course limit you to simply changing the text. You can simply go to the Github page, click edit, and change whatever you want. It will

create a copy for, known as a 'branch', and you can keep working away on it, or you can submit a pull request. If this is approved, it will then get added to the master branch, and be added to the 'live' version of the book.

3.3 Download a copy

This book lives on a public repository on GitHub. That means that it can be downloaded and potentially changed by anyone. You can download a copy of all the files that make up this book, which includes code, figures, pictures, a bibliography file and some other bits.

It's pretty easy to download, you don't even need a github account. Click download. Extract the zip.

3.4 Download R and R-Studio

The first thing you'll need is R and R-studio. They are separate things, which might seem confusing or really obvious, depending on your exposure to programming languages. R is the main software. You could actually just download this and run it with a command prompt and a text editor, if you wanted to. R-Studio is an interface to the language, which makes it much easier to use, and adds lots of extra features.

This bit is pretty simple. You download the latest version of R, install it, and then download R-studio and install that.

R-Studio has four main windows, each with a number of tabs. The top left is where you write code, and view the dataframes you've created or imported. The top right contains your environment, which is a list of all your dataframes, variables and functions and other bits (and also some extra tabs once we've installed some more packages). The bottom right is a list of files in your project's directory.

Now you can open it in R-Studio.

3.5 Licence terms

It's released under a CC-0 licence. This means you can use it for whatever purpose you like, including commercial. More details about creative commons licences are here:

3.6 Clone a copy to GitHub

There's another way to get the book into github, one which allows you to make and suggest changes much more efficiently, though it takes some preparation to

get up and running first. You need to install git to your computer, sign in, and then start a new project, this time clicking 'Version Control', then click on the 'git' option, and then copy and paste the URL of the repository into the first box. When this works, you'll automatically download all the files from github. There's a lot to github, but basically, if you 'pull', you get the latest version from the repository, if you 'commit' then 'push', you push to the repository. You 'push' to your own branch: it's only by submitting a 'pull request' that your version will get considered and merged with the master.

If you want to contribute to the book, you'll probably be at a stage where you can do this.

3.7 Make your own version

If you want, you can just make a 'branch' and never have the intention of having it merge. You could change the name, rip out bits, redo all the code and so forth.

Part I

Part I: Sources and data

Chapter 4

Newspaper data - international

Most countries have digitised all or part of their newspaper collection. Digitised newspapers are a form of 'soft power' or an item of cultural pride. These newspapers, unfortunately, exist in national silos, using different formats, with different levels of access. There's not much to do about that except try and understand the different ways which content has been digitised. Several projects are currently doing pan-national things, including Impresso, and Europeana, but the majority, by far, is not serachable across nations.

In addition, the level and method of access varies widely. Some countries (Australia's *Trove* and the U.S's *Chronicling America* project) provide API access. Others allow for bulk downloads of the XML. Others have free access to all out-of-copyright material, and then allow bulk downloads on request, or through collaboration (such as the KB Library in the Netherlands). Other countries (notably the UK) partner with commercial companies to provide access, including free access on the Library's site.

The data, too, comes in many different forms. Most digitised newspapers are processed with OCR software to create machine readable text. This processing usually results in data in XML format, specifying some information about the content. This will definitely include metadata about the newspaper, information on the software used and the owner of the data, and might include information on the the way the page is laid out, and information on how each article has been segmented. Each software will have a different way of processing the data, and a different output. This will also change as software gets updated. For example, the XML output of the JISC titles includes a different XML file for each article. On the other hand, the newer digitised content from FMP contains an XML file for each page, and a METS file which specifies how the articles can be recreated. This will be outlined in some more detail in a further chapter, but

it's best to think of it as an ongoing, flexible format. We'll explain how to work with some of them but each will be different.

Sources

Lancaster Newsbooks Burney Collection EEBO EEBO TCP (pamphlets) BNA HMD newspapers JISC through GALE

4.1 UK

4.1.1 British newspaper archive

The British Newspaper Archive is a partnership between FindMyPast and the British Library, digitising newspaper pages from the BL's collections. The process is ongoing, digitising from microfilm and originals. Pages are segmented and OCRd, and can be full text searched. It is a commercial site and a fee is charged to access content away from BL reading rooms. Material from HMD newspaper digitisation project will most likely be added to the site.

Added 6 million pages last year

https://www.britishnewspaperarchive.co.uk/

28 million pages

450,000,000 (total physical collection)

Access rights
Free within the BL reading rooms, paid subscription off-site

Features
Download individual pages; OCR; segmentation; search; browse

Example projects Content analysis of 150 years of British Periodicals; Victorian Meme Machine; Protest and the politics of space and place; Frederick Douglas in Britain; Scissors & Paste; Oceanic Exchanges

4.2 Outside UK:

4.2.1 Chronicling America

Library of Congress U.S

Chronicling America is a Library of Congress initiative providing access to historic newspapers. Grants were given to participants to digitise approximately 100,000 newspaper pages each, which should represent "history, geographic coverage and events of note". Mostly from microfilm for speed and cost. PDFs are available for download, full text search. CA has an API which allows for the creation of third party applications and sites using the bulk data. Good

search functions but difficult to browse by date/region. More information here: https://oceanicexchanges.org/2018-10-17-data-reports-ca/

Between 1 - 2 million over the past 5 years, 3.3 million pages expected this year.

https://chroniclingamerica.loc.gov/

https://chroniclingamerica.loc.gov/about/api/

14 million pages

(probably impossible to find total as it is a collection of individual State projects)

JPEG2000 and PDF for download

Free - no key needed for API

OCR; segmentation; full text download; search; API

Data Visualization: Journalism's Journey West; An Epidemiology of Information: Data Mining the 1918 Influenza Epidemic; Bookworm; American Lynching; America's Public Bible; Historical Agricultural News; Chronicling Hoosier; US News Map

4.2.2 Delpher

Koninkklijke Bibliotheek Netherlands

Delpher is the Netherland's national media archive database. It contains 11 million pages of newpapers. It has certain datasets available for bulk download. The front page interface is clean and allows for very quick browsing by date. There is also an associated site labs.kb.nl which hosts datasets and projects. The bulk download has both XML and plain text files, but not images. The site contains 1.5 million newspapers or 11 million pages, representing 15% of all newspapers published. Some external news sources are searchable through Delpher, with the link going through to the external site.

130 titles to be digitised between 2018 - 2021. About 50,000 newspapers (issues) added in 2018.

https://www.delpher.nl/nl/kranten

Full dataset here: https://www.delpher.nl/nl/platform/pages/helpitems?nid=513&scrollitem=true; derived datasets here: http://lab.kb.nl/datasets?f%5B0%5D=field_product_type%3A1

11 million pages (15% of total) 1.4 million newspapers from national library plus 2.9 million external newspapers.

73,333,333 pages

Funding from Metamorfoze - Netherlands national programme for the preservation of paper heritage. https://www.metamorfoze.nl/english

No images in bulk download. Individual pages can be downloaded as text, JPEG, PDF.

Free access, bulk download until 1876

OCR; segmentation; browse by date; search; bulk download of everything up to 1876; cut clippings and download text or image

Siamese; Frame Generator; Dictionary Viewer; Narralyzer; Genre classifier

4.2.3 Eluxemburgensia

Bibliothéque Nationale de Luxembourg Luxembourg

Luxemburg's open data sets, currently the only one released is a historical newspaper dataset. Contains a series of 'packages', made to look like different versions of a web product or software, with very clear uses for each. XML/ALTO files. Link to a tool which extracts data from the XML, available on GitHub. Segmented and OCRd.

15 - 20k pages per month at peak - carried out by external provider

http://www.eluxemburgensia.lu/R/RN=422843522&local_base=SERIALS.

http://data.bnl.lu/data/historical-newspapers/

650,000 pages, 12% of total

5.4 million

Government funded - through National Library/Ministry of Culture

XML/ALTO and PDF, full text and TIFF images

Free access to bulk downloads, free access to PDF downloads.

OCR; search; segmentation; full text; XML download; bulk download

Brodigues blog uses

4.2.4 Europeana

Europeana Consortium (funded by the European Union) Europe

Newspapers; images; art; manuscripts; maps; music; others

Europeana is an aggregator for many of Europe's national newspaper collections, containing 4,129,989 newspapers (issues, presumably, rather than pages). Some are image or full text and other records are metadata only. 876,724 have had OCR processing. Newpapers can be browsed by date, country and title. Can be filtered by rights, which is useful. Images can be downloaded but users are directed back to the original source for more features. The plan is to tag up to 10 million pages with metadata and named entities.

Difficult to tell - about 1,000 additional records tagged ‘newspaper’ were added in 2018.

Currently available through http://www.theeuropeanlibrary.org/tel4/newspapers also https://www.europeana.eu/portal/en/collections/newspapers as part of the wider Europeana site.

4129989 newspapers

Funding: European Union

Multiple sources with different restrictions

OCR; search; segmentation; bulk data download; metadata

4.2.5 Finnish Newspaper Database

National Library of Finland Finland

Newspapers; journals; maps; ephemera

Finland’s national newspaper digitisation, run by the National Library of Finland. 7 million pages, 4 million of which are free to use and out of copyright. Some extensive tools for users: can download full text, PDF or XML of all pages. Have been OCRd and everything is also available on Europeana. Can be browsed by year and title. There’s also bulk data download available, multiple packages.

Digitising at 1 million pages per year according to https://oceanicexchanges.org/2018-02-20-data-reports-finland/

https://digi.kansalliskirjasto.fi/etusivu

https://digi.kansalliskirjasto.fi/opendata/submit

6.4 million newspaper pages: all material up to 1929 plus more.

6.4 million pages PDF download

Free & open until 1929, restrictions thereafter (approx 4 million pages free access). Bulk downloads are available, N-Gram list and other data packages.

OCR; search; segmentation; full text; XML download for individual pages; bulk data download

“Finnish text reuse (http://comhis.fi/clusters); Local Letters to Newspapers; Geography of the nineteenth-century “lingonberry rush” "

4.2.6 Gallica

Bibliothéque Nationale de France France

Newspapers; books; manuscripts; images; music sheets; sounds; objects

France's national newspaper digitisation site, part of their larger digital library of BnF. Intended to have 3.5 million pages, over 30 titles, free of charge. Contains useful search functions, containing snippets of the newspaper image where the term is found. Full text and PDF downloads are available. There's also some interprative essays on the site, and the newspapers can be browsed by theme. Searches for certain themes or topics result in a 'Gallica advises' section above the search results with (presumably curated) information. Can be browsed by region, both region published and region covered.

https://gallica.bnf.fr/html/und/presse-et-revues/les-principaux-quotidiens

2.3 million images/pages PDF, JPEG Free and open for early material OCR; search; browse by date; browse by region;

4.2.7 Trove

National Library of Australia Australia

Newspapers; Govt. Gazettes; Journals & datasets; books; pictures & photos; sound, music & video; diaries & letters; websites; people

Trove is an online database of resources relating to Australia, including newspapers and other material. It is a centralised repository for material from a large number of suppliers, including 221,289,583 newspaper articles. Trove has a very effective and useful API and is used by lots of external researchers as well as for commerical use. The metadata about resources is processed for searching via the website and API, but Trove actually hosts the newspaper content themselves. They also have user tools such as making corrections, tagging etc with a substantial volunteer community making changes. More information here: https://oceanicexchanges.org/2018-05-29-data-reports-trove/

400,000 pages in the past financial year.

https://trove.nla.gov.au/newspaper/

http://help.nla.gov.au/trove/building-with-trove/api

24 million pages

3,800 titles pre-1955. 1,500 of these have been digitised.

Funded by National Library but also State and local libraries sometimes pay to have their material digitised/uploaded PDF download

Free, open API. Material is mostly pre-1955 and therefore can be digitised without copyright agreement. Some post-1955 material with agreement from the publishers.

OCR; segmentation; search;browse by date; API; metadata

Drifter; Culture collage; Eyes on the Past; The Front Page; Trove Zone Explorer

4.2.8 Papers Past

National Library of New Zealand New Zealand

Newspapers; magazines & journals; Letters & Diaries; Parliamentary papers

Papers Past is New Zealand's digitised document resource. It has four sections: Newspapers, Magazines & Journals, Letters & Diaries and Parliamentary Papers. The Newspaper collection is a selection of digitised NZ and Pacific newspapers, including early te reo Maori ones. Content is searchable, and it is possible to browse by date, title or region. Each title has a mini-history on the site. No indication of how much material has been digitised as a percentage of total. Particularly high-quality scans and accurate OCR/segmentation.

350-400k pages per year

https://paperspast.natlib.govt.nz/newspapers

Supplied on case-by-case basis

40% available through digitalnz.org API

5.4 million pages

Free open access. Images and text can be printed. Assumption that anything over 100 years is out of copyright.

OCR; segmentation; search;browse by date

4.2.9 Hemeroteca Nacional Digital de Mexico

(1735–1969) This is the national newspaper collection of Mexico. Titles from 1722 - 2010, with restrictions on newer content. Has full-text search but doesn't seem to have access to full text download. Restrictions on PDF downloads. Some basic visualisations and information on the most consulted titles.

http://www.hndm.unam.mx/index.php/es/

4.2.10 Belgica Press

This is Belgium's national newspaper digitisation project. 4 million pages have been digitised, from 1831 to 1950, for a select number of titles (L'Echo De La Presse, Gazet Van Brussel, Het Handelsblad, L'Indépendance Belge (printed in Britain), Le Messager De Gand, De Nieuwe Gids, De Nieuwe Standaard and Het Nieuws Van Den Dag). Up to 1919 is freely available online. Seems to have full text search but not download, and no easy way to download full images.

https://www.belgicapress.be/

4.2 million pages, 1.2 million available externaly (rest within library)

Restrictions on PDF download and full text versions, material after 1919.

OCR; search; browse by date

4.2.11 Welsh Newspaper Archive

To fill in - have a large collection

4.2.12 National Library of Scotland

4.2.13 Italian newspapers

Doesn't seem to be any centralised repository, but various archives are available through the websites of individual newspapers.

4.2.14 Hemeroteca Digital (Spain)

4.2.15 ANNO - Austrian Newspapers Online

Anno is Austria's national digital newspaper collection. It contains 500 titles, over 10 million pages, from the 16th century to the 1940s. They are mostly OCRd and can be full-text searched. The site can be browsed by title or date, through a calendar.

4.2.16 BC Historical Newspapers

The BC Historical Newspapers project features digitized versions of historical papers from around the province. The titles, which range from the Abbotsford Post to the Ymir Mirror, date from 1859 to 1995. Has some pretty advanced browse and search features, and can download PDFs, full text and metadata files. https://open.library.ubc.ca/collections/bcnewspapers

OCR; full text download; metadata download; browse by date, title

4.2.17 Media Stream

Media Stream is Denmark's web access for the Royal Library's newspaper, radio and television commercials. 35 million pages of newspapers have been digitised, access to anything after January 1919 is available only in the library itself. The archive has full-text search, but seemingly no way to access the full text of individual articles. Pre 1919 material can be downloaded as PDFs. There's also a link to a tutorial on how to search.

http://www2.statsbiblioteket.dk/mediestream/avis

35000000 pages PDF, free to 1919 OCR; browse by title

4.2.18 Stare hrvatske novine

Croatia's national digitised newspapers. Select papers starting from 1789. Limited functionality.

4.2.19 ZEFYS

Access to holdings in the Staatsbibliothek Berlin and other German newspapers and German newspapers abroad. 276,015 copies of 193 newspapers. Mixed functionality, some are OCRd, others just images. Has a 'year cloud' for browsing by year, which also illustrates quickly which years have most quantity of news. Seems to run just from 1890 to 1939.

http://zefys.staatsbibliothek-berlin.de/index.php?id=start

4.2.20 Timarit.is

Newspaper archive for Iceland, Greenland and Faroe Islands. Currently contains 1179 titles, 64683 articles, 5619584 pages (presumably the articles are low because it's only counting ones that have been segmented successfully). Full text search is available, as is full text and PDF downloads. Can be browsed by titles. Some early titles possibly manuscript.

http://timarit.is/listing_init.jsp?lang=en

5619584 pages PDF FREE OCR (limited); full text search; browse by title

4.2.21 Google news archive

Google news archive originally started as a standalone product but now is incorporated into Google News. Mixture of titles available for searching, but has not been updated since 2011. Significantly has a full archive of the NYT. OCRd, but the text is not browsable or downloadable. Can browse by title or search. Perhaps more functionality on various Google news apps

https://news.google.com/newspapers

4.2.22 Baltic Digital Library

Collection of digital objects, including newspapers. Some interesting features including structure (essentially allowing for browsing through titles/issues.

4.2.23 NewspaperSG

Singapore's national newspaper collection. Newspapers from 1831 onwards, plus a catelogue of the National Library's microfilm holdings. Interesting 'search terms visualiser', essentially an N-Gram timeline, with up to five words at a time. Gives actual numbers rather than relative frequency, which isn't very helpful if the material is not evenly spread out over the time. Also 'this week in history' and 'most viewed articles' sections. Search brings up OCR text first, rather than images. Cannot download full files

4.2.24 Svenska dagstidningar

Partial Swedish Newspaper archive, containing over 500 titles, from 1645 to the present day. Material up until 1903 is freely available, anything after that only in the reading rooms of various Swedish libraries.

https://tidningar.kb.se/

500 titles

4.2.25 Handwritten Newspapers Project

Archive of handwritten news from around the world. First issue is from 1700. Browsable by date. Some images but mostly links or catelogue records

https://handwrittennews.com/

4.2.26 The Medici Archive Project

The Medici archive project consists of two parts: Building Interaction Archives (BIA) allows for the querying of manuscript documents, including handwritten news (avvisi). Some have full text transcriptions or English synopses. Documents can be compared via split screen, and there is some ability to browse or search named entities. The Medici Interactive Archive (MIA) is the second part, which intends to make the whole Medici archive available to researchers, including the ability to edit transcriptions, for example.
http://www.medici.org/

Chapter 5

UK Newspaper Data

5.1 Intro to British Library Newspapers

> Systematic collection of newspapers at the British Museum (the precursor of the British Library) did not really begin until 1822. At that time publishers were obliged to supply copies of their newspapers to the Stamp Office so that they could be taxed. In 1822 it was agreed that these copies would be passed to the British Museum after a period of three years. From 1869 onwards newspapers were included in the legal deposit legislation and were then deposited directly at the British Museum. This systematic application of legal deposit requirements means that many thousands of complete runs of newspapers have accumulated. The majority of newspapers collected are those published since 1800.[1]

What does this all mean for the data? First of all, it means that only a fraction of what was published has been preserved or collected, and only a fraction of that which has been collected has been digitised.

It's all very well being able to access the content, but for the purposes of the kind of things we'd like to do, access to the data is needed.

Lancaster Newsbook Corpus

5.2 JISC Newspaper digitisation projects

Maybe a separate chapter?

[1] Ed King, 'Digitisation of British Newspapers 1800-1900', 2007 <https://www.gale.com/intl/essays/ed-king-digitisation-of-british-newspapers-1800-1900> [accessed 2007].

Most of the projects in the UK which have used newspaper data have been using the *British Library's 19th Century Newspapers* collection. This is an interesting collection of content and worth outlining in some detail. Knowing the sources, geographical makeup and motivation behind the titles in the collection can be really helpful in thinking about what is and is not in the sources.

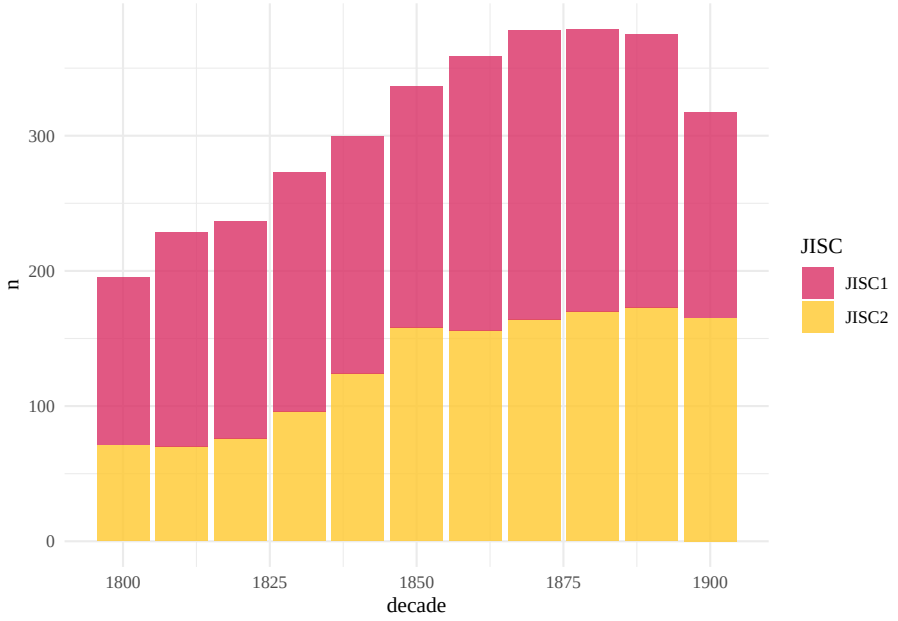


Figure 5.1: Very approximate chart of JISC titles, assuming that we had complete runs for all. Counted by year rather than number of pages digitised.

The JISC newspaper digitisation program began in 2004, when The British Library received two million pounds from the Joint Information Systems Committee (JISC) to complete a newspaper digitisation project. A plan was made to digitise up to two million pages, across 49 titles.[2] A second phase of the project digitised a further 22 titles.[3]

The titles cover England, Scotland, Wales and Ireland, and it should be noted that the latter is underrepresented although it was obviously an integral part of the United Kingdom at the time of the publication of these newspapers - something that's often overlooked in projects using the JISC data. They cover about 40 cities 5.2, and are spread across 24 counties within Great Britain 5.3,

[2] King.

[3] Jane Shaw, 'Selection of Newspapers', *British Library Newspapers*, 2007 <https://www.gale.com/intl/essays/jane-shaw-selection-of-newspapers>; Jane Shaw, '10 Billion Words: The British Library British Newspapers 1800-1900 Project: Some Guidelines for Large-Scale Newspaper Digitisation', 2005 <https://archive.ifla.org/IV/ifla71/papers/154e-Shaw.pdf> for a good brief overview to the selection process for JISC 1.

plus Dublin and Belfast.

> The forty-eight titles chosen represent a very large cross-section of 19th century press and publishing history. Three principles guided the work of the selection panel: firstly, that newspapers from all over the UK would be represented in the database; in practice, this meant selecting a significant regional or city title, from a large number of potential candidate titles. Secondly, the whole of the nineteenth century would be covered; and thirdly, that, once a newspaper title was selected, all of the issues available at the British Library would be digitised. To maximise content, only the last timed edition was digitised. No variant editions were included. Thirdly, once a newspaper was selected, all of its run of issue would be digitised.[4]

Jane Shaw wrote, in 2007:

> The academic panel made their selection using the following eligibility criteria:
>
> To ensure that complete runs of newspapers are scanned To have the most complete date range, 1800-1900, covered by the titles selected To have the greatest UK-wide coverage as possible To include the specialist area of Chartism (many of which are short runs) To consider the coverage of the title: e.g., the London area; a large urban area (e.g., Birmingham); a larger regional/rural area To consider the numbers printed - a large circulation The paper was successful in its time via its sales To consider the different editions for dailies and weeklies and their importance for article inclusion or exclusion To consider special content, e.g., the newspaper espoused a certain political viewpoint (radical/conservative) The paper was influential via its editorials.[5]

What's really clear, is that the selection was driven by assumed historical need by the Library's users, plus some practicalities around copyright, microfilm and

The result was a heavily curated collection, albeit with decent academic rigour and good intentions and like all collections created in this way, it is subject, quite rightly, to a lot of scrutiny by historians.[6]

[list the titles]

This is all covered in lots of detail elsewhere, including some really interesting critiques of the access and so forth.[7] But the overall makeup of it is clear, and

[4] Ed King, 'British Library Digitisation: Access and Copyright', 2008.

[5] Shaw.

[6] Paul Fyfe, 'An Archaeology of Victorian Newspapers', *Victorian Periodicals Review*, 49.4 (2016), 546–77 <https://doi.org/10.1353/vpr.2016.0039> for example.

[7] Thomas Smits, 'Making the News National: Using Digitized Newspapers to Study the Distribution of the Queen's Speech by W. H. Smith & Son, 1846–1858', *Victorian Periodi-*

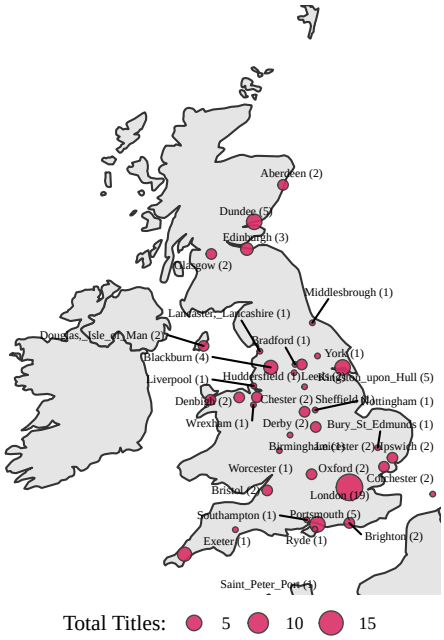


Figure 5.2: JISC 1 & 2 titles, by city. Irish titles not plotted

this was a very specifically curated collection, though it was also influenced by contingency, in that it used microfilm (sometimes new microfilm). But overall, one might say that the collection has specific historical relevant, and was in ways representative. It does, though, only represent a tiny fraction of the newspaper collection, and by being relevant and restricted to 'important' titles, it does of course miss other voices. For example, much of the Library's collection consists of short runs, and much of it has not been microfilmed, which means it won't have been selected for digitisation. This means that 2019 digitisation selection policies are indirectly *greatly* influenced by microfilm selection policies of the 70s, 80s, and 90s. Subsequent digitisation projects are trying to rectify these motivations, but again, it's good to keep in mind the

Currently researchers access this either through Gale, or through the British Library as an external researcher. Many researchers have requested access to the collection through Gale, which they will apparently do in exchange for a fee for the costs of the hard drives and presumably some labour time. The specifics of the XML used, and some code for extracting the data, are available in the following chapter.

---

cals Review, 49.4 (2016), 598–625 <https://doi.org/10.1353/vpr.2016.0041>; James Mussell, 'Elemental Forms: Elemental Forms: The Newspaper as Popular Genre in the Nineteenth Century', *Media History*, 20.1 (2014), 4–20 <https://doi.org/10.1080/13688804.2014.880264> both include some discussion and critique of the British Library Newspaper Collection.

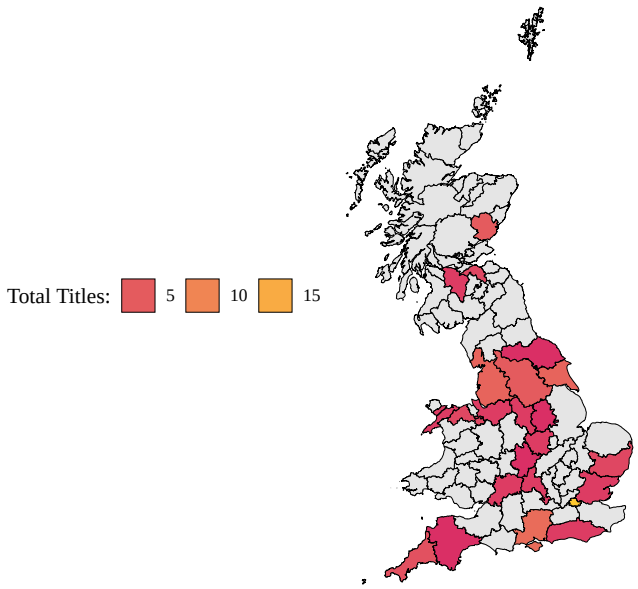


Figure 5.3: JISC 1 & 2 Digitisation Programmes

Some researchers have also got access to the collection through

5.3 BRITISH NEWSPAPER ARCHIVE

Most of the British Library's digitised newspaper collection is available on the British Newspaper Archive (BNA). The BNA is a commercial product run by a family history company called FindMyPast. FindMyPast is responsible for digitising large amounts of the Library's newspapers, mostly through microfilm. As such, they have a very different focus to the JISC digitisation projects. The BNA is constantly growing, and it already dwarfs the JISC projects by number of pages: the BNA currently hosts about 33 million pages, against the 3 million or so of the two JISC projects 5.4

There are several important implications for this. First, most data-driven historical work carried out on newspapers has used the JISC data, rather than the BNA collection, because of relative ease-of-access. It's an important point, as there may be an assumption that this kind of work is generally In addition, as it is a contantly evolving dataset, reproducibility is difficult. There are some exceptions: the Bristol N-Gram and named entity datasets used the FMP data, processing the top phrases and words from about 16 million pages. The collection has doubled in size since then: it's likely that were it to be run again the results would be different.

This is not only because of volume but also because of the change in focus and digitisation policy. Newspapers are selected for various reasons, but an underlying principle of coverage seems to be important: newspapers are obviously not selected at random, which necessarily results in a changing, evolving idea of bias.

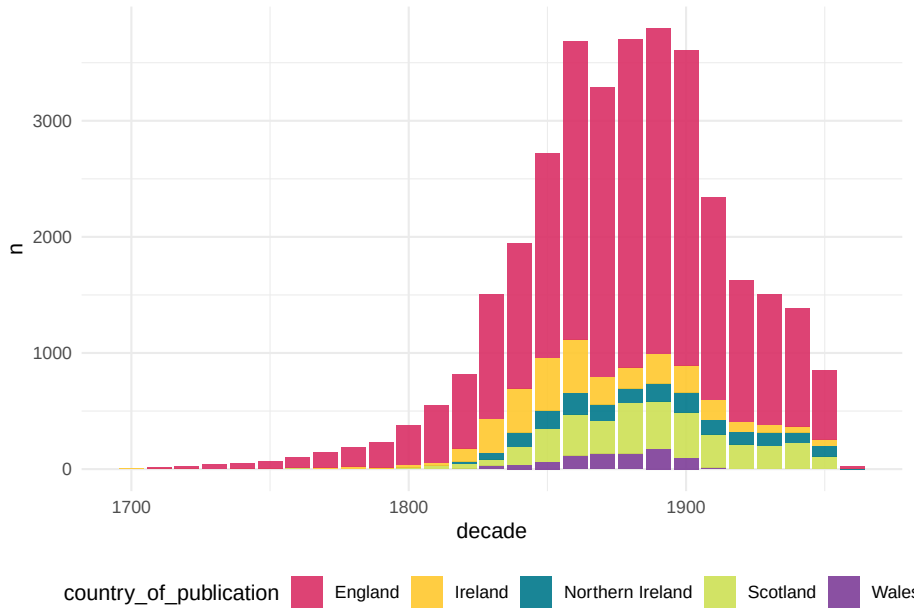


Figure 5.4: Newspaper-years on the British Newspaper Archive. Includes JISC content above.

5.4 GALE

5.4.1 Getting access to GALE data - experience of others

5.5 HMD data- on repository

5.5.1 Project goals

The Heritage Made Digital Project is a project within the British Library to digitise 19th century newspapers. It has a specific curatorial focus. It picked titles which are completely out of copyright, which means that they all *finished* publication before 1879. It also aimed to preserve: because of this, it chose titles which were not on microfilm, and were also in poor or unfit condition. Newspaper

[8]data from https://www.britishnewspaperarchive.co.uk/titles/

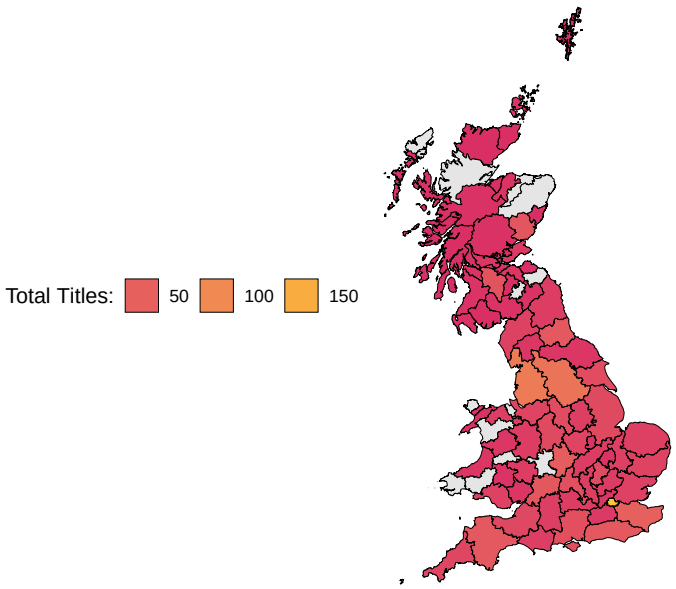


Figure 5.5: Titles on the British Newspaper Archive.[8]

volumes in unfit condition cannot be called up by readers: this meant that if a volume is not microfilmed and is in this state, it can't be read by *anyone*.

The other curatorial goal was to focus on 'national' titles. In practice this meant choosing titles printed in London but without a specific London focus. This might seem like a strange selection policy considering the general view that stuff outside London is neglected, but it sort of makes sense: JISC focused on regional titles, then local, and all the 'big' nationals like the Times or the Manchester Guardian have been digitised but sit behind paywalls within those organisations. This means that a bunch of historically important titles may have fallen through the cracks, and this projects is digitising some of those.[9]

The important thing to note is that the is, like JISC, a selection made by historians for preservation and curatorial purposes. Again, it's worth thinking about how this affects the bias.

The good news is that as these newspapers are deemed out of copyright, the data can be made freely downloadable. Currently the first batch of newspapers,

[9]The term national is debatable, but it's been used to try and distinguish from titles which clearly had a focus on one region. Even this is difficult: regionals would have often had a national focus, and were in any case reprinting many national stories. But their audience would have been primarily in a limited geographical area, unlike a bunch of London-based titles, which were printed and sent out across the country, first by train, then the stories themselves by telegraph.

in METS/ALTO format, are available on the British LIbrary's Open Repository. They have a CC-0 licence, which means they can be used for any purpose whatsoever. Much of the data examples in the following chapters will use this data.

[list HMD titles] [chart of type of runs, what's been digitised so far] ### What's available ### XML flavour

5.6 Luxembourg data

Chapter 6

Data formats

6.1 XML - different flavours

6.2 Full-text transcriptions - like the Lancaster Corpus

6.3 CQPWeb newspaper corpora

// # Don't forget about this one

Chapter 7

OCR and its problems

7.1 What is OCR?

7.2 What is it like in BL newspapers?

In a word, terrible. Well, it varies a lot, and the truth is, nobody *really* knows what it's like, because that would involve having large sets of very accurate, manually transcribed newspapers, to compare to the OCR text. Subjectively, we can probably make a few generalisations.

It gets better as the software gets better, but not particularly quickly, because much of the quality is dependant on things to do with the physical form.

Digitising from print is much better than from microfilm. But print can still be bad.

Standard text is much better than non-standard. For example, different fonts, sizes, and so forth.

Advertisements seem to have particularly bad OCR - they are generally not in regular blocks of text, which the OCR software finds difficult, and they often used non-standard characters or fonts to stand out.

The time dimension is not clear: type probably got better, but it also got smaller, more columns.

Problems with the physical page have a huge effect: rips, tears, foxing, dark patches and so forth. Many errors are not because of the microfilm, digital image or software, and may not be fixable.

What does this all mean? Well, it introduces bias, and probably in non-random ways, but in ways that have implications for our work. If things are digitised from a mix of print and microfilm, for example, we might get very different

results for the print portion, which might easily be mis-attributed to a historical finding. Perhaps there were twice as many mentions of cheese in the 1850s than in the 1890s? It's probably best to rule out that this is not just because later newspapers had a difficult font, or they were digitised from microfilm instead of print.[1]

7.3 What information do we have on OCR?

Can you add anything on different between JISC and BNA 'confidence scores'? Do a comparison?

7.3.1 Error rate

The error rate reported by the software is a confidence score. The overall confidence score is an average of all these scores. It's not testing against anything but itself: it could have a high confidence score but be completely wrong - perhaps the computer has a misguided sense of its own accuracy? Or it could be crippled with self-doubt, and report very low scores whilst in fact getting everything correct.

7.3.2 Ground truth - testing against transcription

7.4 Impact on analysis

It depends. Broad analysis still seems to work - keyword searches, for example, come up with broadly expected results. It might be more important in finer work, for example Natural Language Processing (NLP). NLP relies on

Why You (A Humanist) Should Care About Optical Character Recognition

[1] Mark John Hill and Simon Hengchen, 'Quantifying the Impact of Dirty Ocr on Historical Text Analysis: Eighteenth Century Collections Online as a Case Study', *Digital Scholarship in the Humanities : DSH*, 2019 <https://doi.org/10.1093/llc/fqz024>, @Cordell_2017, @Piotrowski_2012, @cordell-ocr.

Part II

Part I: Basic data methods

Chapter 8

Methods

8.1 Mapping seventeenth century newsbooks

8.1.1 Mapping with R

8.1.1.1 What can R do

8.1.1.2 Limitations

8.1.2 Steps

8.1.2.1 Making a background map

8.1.2.2 Geocoding

8.1.2.3 Drawing points

8.1.2.4 Further customisation

8.2 Network analysis of seventeenth century newsbooks

8.2.1 Background

8.2.2 Network analysis

8.2.3 Steps

8.3 Network analysis using NLP and Lancaster newsbooks

8.3.1 Background

8.3.2 Sources used

8.3.3 Steps

8.3.4 Choropleth maps - nineteenth century

Chapter 9

R and the Tidyverse

The motivation behind this book was to provide a way to access and analyse newspaper data using a programming language that I am familiar with. The reason is simple: you write what you know, and I know R best. With its interface, R-Studio, I think it has the easiest transition from someone used to spreadsheet programs, and you'll realise that most of what you do is filter, sort, count and select columns in a data format called a dataframe.

A dataframe is basically a spreadsheet - it contains rows with observations, and columns with variables. Each row is generally a *thing*, for want of a better word. A thing that wants to be counted, either by summarising it as a more general thing, or turning it into something else and then counting it, or removing some of the things first and then counting the leftovers. For us, the thing might be a record of a newspaper title, or a newspaper article (and its text), or it might be a single word.

You can do a lot more cool things with a thing in a dataframe. A thing might be a single polygon, in a huge dataframe of polygons or lines, all of which add up to a map, which we can then count, sort, filter and eventually draw.

While I want this book to stand alone, it's not meant to function as a tutorial, but rather I will try to explain just enough to do the analysis seen in this book.

9.1 R

R has been around for a while. It's a good programming language for data analysis and plotting, and it is relatively easy to create nice-looking plots.

9.1.1 Base R commands

I don't use them very much, but R does have a bunch of very well-developed commands for doing the sorting, filtering and counting mentioned above. If you want to learn base R, I recommend the following:

It is worth understanding the main types of data that you'll come across, in your environment window. First, you'll have dataframes. These are the spreadsheet-like objects which you'll use in most analyses. They have rows and columns.

Next are variables. A variable is assigned to a name, and then used for various purposes.

You'll often here of something called a vector. A vector is like a python list. It's sort of like a single column in a dataframe (spreadsheet), a 2D dataframe. It can have different types: for example, a character vector looks like this `c("apples", "bananas", "oranges")`

A dataframe is just a bunch of vectors side by side.

```
library(tibble)
```

```
fruit = c("apples", "bananas", "oranges", "apples")
colour = c("green", "yellow", "orange", "red")
amount = c(2,5,10,8)
```

```
fruit_data = data.frame(fruit,colour,amount, stringsAsFactors = FALSE)
```

```
fruit_data
```

```
##     fruit colour amount
## 1  apples  green      2
## 2 bananas yellow      5
## 3 oranges orange     10
## 4  apples    red      8
```

Notice above that the third column, the amount, has under it instead of . That's because R is treating it as a number, rather than a character. This means you can add them up and do all sorts of other mathy type things to them.

Anyway, that's a dataframe.

9.1.2 R studio

R studio is an IDE. It's unusual in that it's very self-contained and ubiqutous. One of the attractions for R when I began, was that it seemed so much less confusing than Python, which has many different ways of accessing it. While you can use R in many ways too, R-Studio seems much more of a 'catch-all' that appeals to everyone. It's available here: [link]

It has four main windows: you'll type your code in the top left, your environments

are on the right (it lists your dataframes, variables and so forth. The bottom right contains the files in your project folder, as well as some other tabs showing plots and help.The bottom left shows the console - that's where everything you do is sent and where it is acted on by R.

The help screen is particularly useful. Typing `?` followed by a function brings up a screen explaining the function on the bottom right panel. So typing: `?data.frame()` into the R console will display a screen containing information on a function and its various options.

into

9.2 Tidyverse

Most of the work in this book is done using a set of packages developed for R called the 'tidyverse'. These enhance and improve a large range of R functions, with much nice syntax - and they're faster too. It's really a bunch of individual packages for sorting, filtering and plotting data frames. They can be divided into a number of diferent categories.

They all work in the same way. The first argument is the thing you want to operate on. This is nearly always a data frame. After come other arguments, which are often specific columns, or certain variables you want to do something with.

9.2.1 Select, pull

`select()` allows you to select columns. You can use names or numbers to pick the columns, and you can use a `-` sign to select everything *but* a given column.

Using the fruit data frame we created above: We can select just the fruit and colour columns:

```
select(fruit_data, fruit, colour)
```

```
##     fruit colour
## 1  apples  green
## 2 bananas yellow
## 3 oranges orange
## 4  apples    red
```

Select everything but the colour column:

```
select(fruit_data, -colour)
```

```
##     fruit amount
## 1  apples      2
## 2 bananas      5
## 3 oranges     10
```

```
## 4  apples      8
```

Select the first two columns:

```
select(fruit_data, 1:2)
```

```
##     fruit colour
## 1  apples  green
## 2 bananas yellow
## 3 oranges orange
## 4  apples    red
```

9.2.2 group_by, tally, summarise

The next group of functions group things together and count them. Sounds boring but you would be amazed by how much of data science just seems to be doing those two things in various combinations.

`group_by()` puts rows with the same variable in a column of your dataframe into a group. Once they're in a group, you can count them or summarise them by another variable.

First you need to create a new dataframe with the grouped fruit.

```
grouped_fruit = group_by(fruit_data, fruit)
```

Next we use `tally()`. This counts all the instances of each fruit group.

```
tally(grouped_fruit)
```

```
## # A tibble: 3 x 2
##   fruit       n
##   <chr>   <int>
## 1 apples      2
## 2 bananas     1
## 3 oranges     1
```

See? Now the apples are grouped together rather than being two separate rows, and there's a new column called n, which contains the result of the count.

If we specify that we want to count by something else, we can add that in as a 'weight', by adding `wt =` as an argument in the function.

```
tally(grouped_fruit, wt = amount)
```

```
## # A tibble: 3 x 2
##   fruit       n
##   <chr>   <dbl>
## 1 apples     10
## 2 bananas     5
## 3 oranges    10
```

That counts the amounts of each fruit, ignoring the colour.

9.2.3 filter

Another quite obviously useful function. This filters the dataframe based on a condition which you set within the function.

Just red fruit:

```
filter(fruit_data, colour == 'red')
```

```
##    fruit colour amount
## 1 apples    red      8
```

Just fruit with at least 5 pieces:

```
filter(fruit_data, amount >=5)
```

```
##     fruit colour amount
## 1 bananas yellow      5
## 2 oranges orange     10
## 3  apples    red      8
```

9.2.4 sort, arrange, top_n

Another useful set of functions, often you want to sort things. The function `arrange()` does this very nicely. You specify the data frame, and the variable you would like to sort by.

```
arrange(fruit_data, amount)
```

```
##     fruit colour amount
## 1  apples  green      2
## 2 bananas yellow      5
## 3  apples    red      8
## 4 oranges orange     10
```

Sorting is ascending by default, but you can specify descending using `desc()`:

```
arrange(fruit_data, desc(amount))
```

```
##     fruit colour amount
## 1 oranges orange     10
## 2  apples    red      8
## 3 bananas yellow      5
## 4  apples  green      2
```

If you 'sort`arrange()` by a list of characters, you'll get alphabetical order:

```
arrange(fruit_data, fruit)
```

```
##     fruit colour amount
## 1  apples  green      2
## 2  apples    red      8
## 3 bananas yellow      5
## 4 oranges orange     10
```

You can sort by multiple things:

```
arrange(fruit_data, fruit, desc(amount))
```

```
##     fruit colour amount
## 1  apples    red      8
## 2  apples  green      2
## 3 bananas yellow      5
## 4 oranges orange     10
```

Notice that now red apples are first.

9.2.5 Piping

Another great feature of the tidyverse is that you can 'pipe' commands through a bunch of functions. This means that you can do one operate, and pass the result to another operation. The previous dataframe is passed as the first argument of the next function by using the pipe `%>%` command. It works like this:

```
fruit_data %>%
  filter(colour != 'yellow') %>%
  group_by(fruit) %>%
  tally(amount) %>%
  arrange(desc(n))
```

```
## # A tibble: 2 x 2
##   fruit       n
##   <chr>   <dbl>
## 1 apples     10
## 2 oranges    10
```

That code block, written in prose: "take fruit data, remove any yellow colour fruit, count the fruits by type and amount, and arrange in descending order of the total"

9.2.6 Plotting

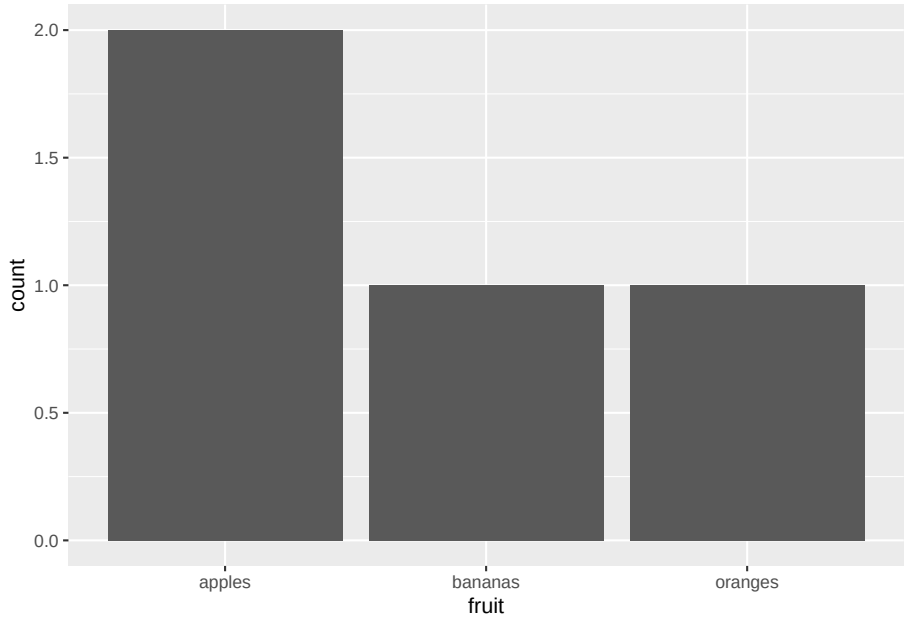
The tidyverse includes a pretty amazing plotting library called ggplot2. This is used by piping your dataframe to a function called `ggplot()`. It's too much to go into now, but the basic idea is that you add your data, then you can add plot elements which are called geoms. Some common ones are `geom_line()`, `geom_bar` and `geom_point`. To the geom function you add aesthetics, which is basically telling the function which bits of your data should be responsible for

which parts of the visualisation. These are added using `aes()`. I'll explain a bit more about some in other parts of this book.

As an example:

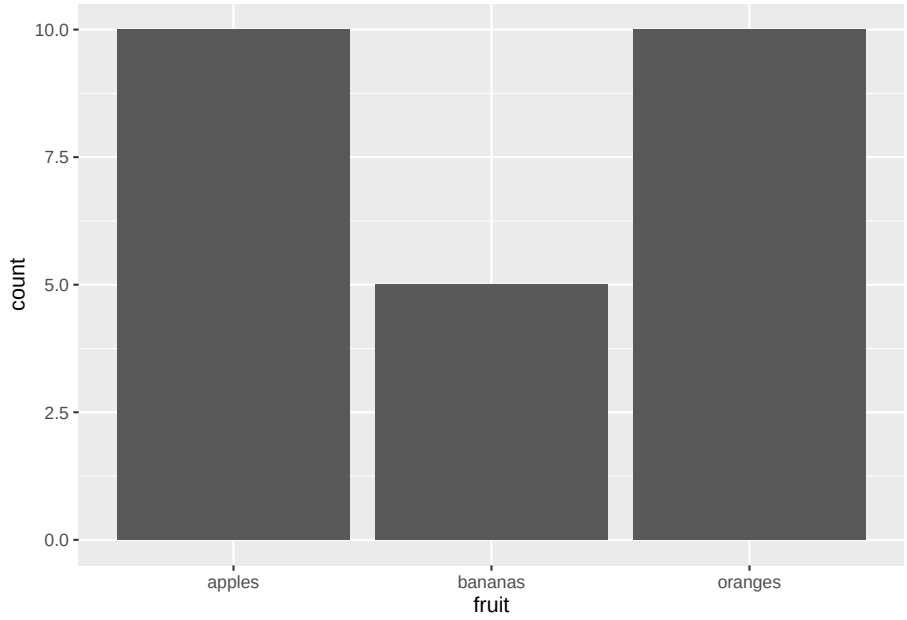
Bar chart of different types of fruit (one each of bananas and oranges, two types of apple)

```
fruit_data %>% ggplot() + geom_bar(aes(x = fruit))
```



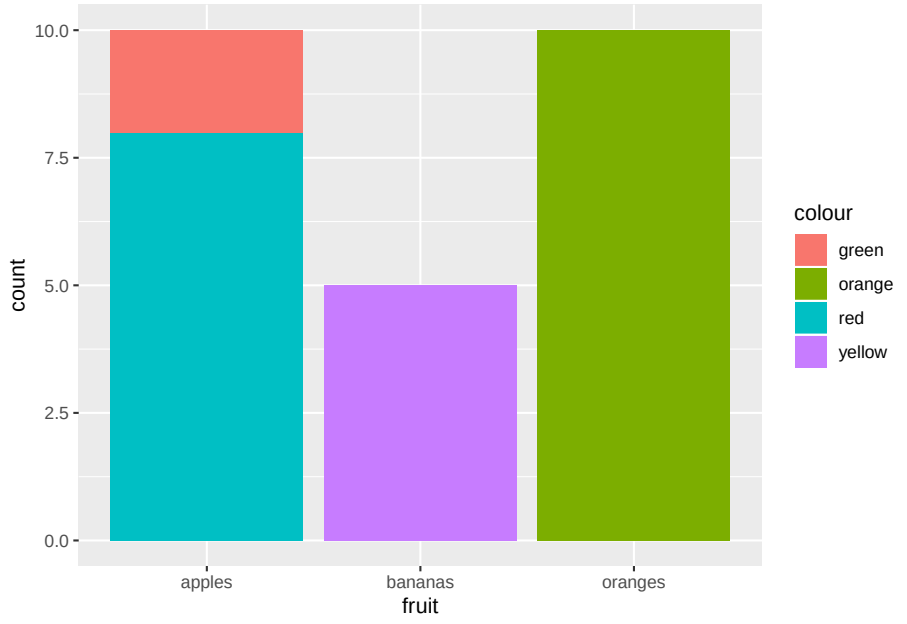
Counting the total amount of fruit:

```
fruit_data %>% ggplot() + geom_bar(aes(x = fruit, weight = amount))
```



Charting amounts and fruit colours:

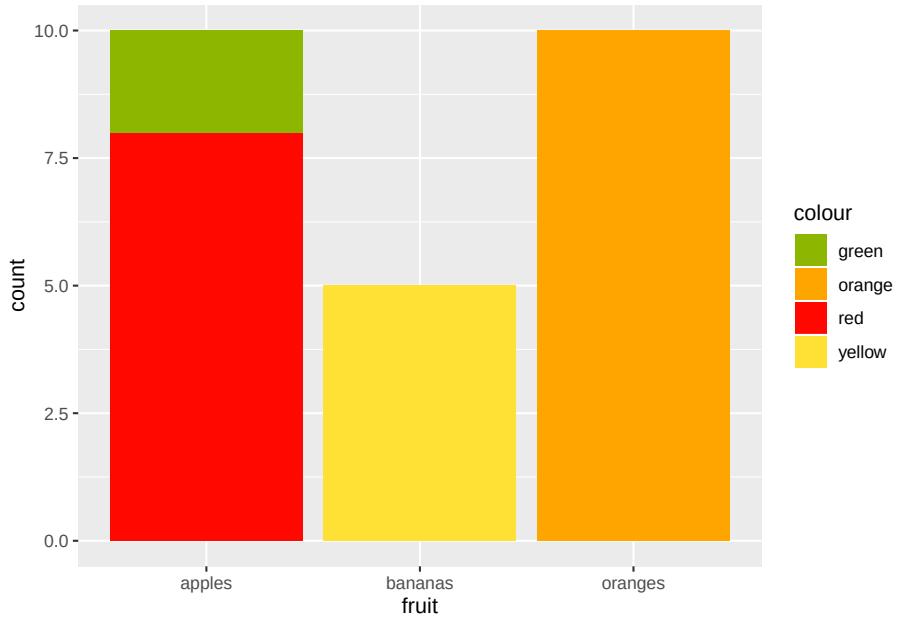
```
fruit_data %>% ggplot() + geom_bar(aes(x = fruit, weight = amount, fill = colour))
```



And just because it annoys me having random colours, we can map them to the

actual colours:

```
fruit_data %>%
  ggplot() +
  geom_bar(aes(x = fruit, weight = amount, fill = colour)) +
  scale_fill_manual(values = c("orange" = "orange",
                               "green" = "#8db600",
                               "red" = "#ff0800",
                               "yellow" = "#ffe135"))
```



9.2.7 Doing this with newspaper data

Who cares about fruit? Nobody, that's who. We want newspaper data! Let's load a dataset of metadata for all the titles held by the library, and do some counting and sorting.

Download from here: British Library Research Repository

You would need to extract into your project folder first, if you're following along at home:

`read_csv` reads the csv from file.

```
title_list = read_csv('data/BritishAndIrishNewspapersTitleList_20191118.csv')
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
```

```
##   title_id = col_double(),
##   nid = col_double(),
##   nlp = col_double(),
##   first_date_held = col_double(),
##   publication_date_one = col_double(),
##   publication_date_two = col_double()
## )

## See spec(...) for full column specifications.
```

Select some particularly relevant columns:

```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication)
```

```
## # A tibble: 24,927 x 4
##    publication_title            first_date_held last_date_held country_of_publi~
##    <chr>                                  <dbl> <chr>          <chr>
##  1 Corante, or, Newes from Ita~            1621 1621           The Netherlands
##  2 Corante, or, Newes from Ita~            1621 1621           The Netherlands
##  3 Corante, or, Newes from Ita~            1621 1621           The Netherlands
##  4 Corante, or, Newes from Ita~            1621 1621           England
##  5 Courant Newes out of Italy,~            1621 1621           The Netherlands
##  6 "A Relation of the late Occ~            1622 1622           England
##  7 "A Relation of the late Occ~            1622 1622           England
##  8 "A Relation of the late Occ~            1622 1622           England
##  9 "A Relation of the late Occ~            1622 1622           England
## 10 "A Relation of the late Occ~            1622 1622           England
## # ... with 24,917 more rows
```

Arrange in order of the latest date of publication, and then by the first date of publication:

```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication) %>%
  arrange(desc(last_date_held), first_date_held)
```

```
## # A tibble: 24,927 x 4
##    publication_title            first_date_held last_date_held country_of_publi~
##    <chr>                                  <dbl> <chr>          <chr>
##  1 Shrewsbury chronicle                    1773 Continuing     England
##  2 London times|The Times|Time~            1788 Continuing     England
```

```
##  3 Observer (London)|Observer ~            1791 Continuing     England
##  4 Limerick chronicle                      1800 Continuing     Ireland
##  5 Hampshire chronicle|The Ham~            1816 Continuing     England
##  6 The Inverness Courier, and ~            1817 Continuing     Scotland
##  7 Sunday times (London)|Sunda~            1822 Continuing     England
##  8 The Impartial Reporter, etc             1825 Continuing     Northern Ireland
##  9 Impartial reporter and farm~            1825 Continuing     Northern Ireland
## 10 Aberdeen observer                       1829 Continuing     Scotland
## # ... with 24,917 more rows
```

Group and count by country of publication:

```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication) %>%
  arrange(desc(last_date_held)) %>%
  group_by(country_of_publication) %>%
  tally()
```

```
## # A tibble: 40 x 2
##    country_of_publication              n
##    <chr>                           <int>
##  1 Bermuda Islands                    24
##  2 Cayman Islands                      1
##  3 England                         20465
##  4 England|Hong Kong                   1
##  5 England|India                       2
##  6 England|Iran                        2
##  7 England|Ireland                    10
##  8 England|Ireland|Northern Ireland   10
##  9 England|Jamaica                     7
## 10 England|Malta                       2
## # ... with 30 more rows
```

Arrange again, this time in descending order of number of titles for each country:

```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication) %>%
  arrange(desc(last_date_held)) %>%
  group_by(country_of_publication) %>%
  tally() %>%
  arrange(desc(n))
```

```
## # A tibble: 40 x 2
##    country_of_publication                n
##    <chr>                             <int>
##  1 England                           20465
##  2 Scotland                           1778
##  3 Ireland                            1050
##  4 Wales                              1019
##  5 Northern Ireland                    415
##  6 England|Wales                        58
##  7 Bermuda Islands                      24
##  8 England|Scotland                     13
##  9 England|Ireland                      10
## 10 England|Ireland|Northern Ireland     10
## # ... with 30 more rows
```

Filter only those with more than 100 titles:

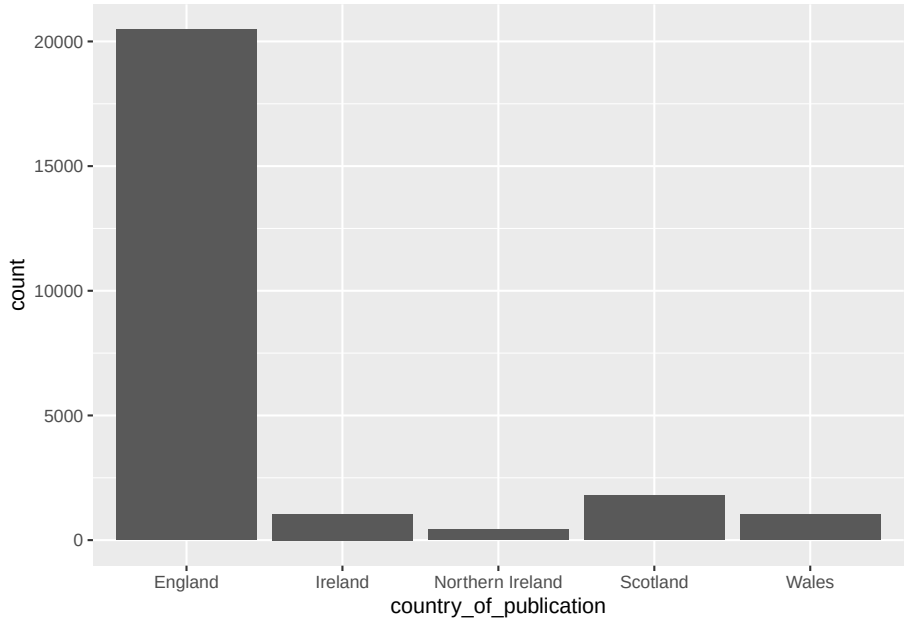
```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication) %>%
  arrange(desc(last_date_held)) %>%
  group_by(country_of_publication) %>%
  tally() %>%
  arrange(desc(n)) %>%
  filter(n>=100)
```

```
## # A tibble: 5 x 2
##   country_of_publication     n
##   <chr>                  <int>
## 1 England                20465
## 2 Scotland                1778
## 3 Ireland                 1050
## 4 Wales                   1019
## 5 Northern Ireland         415
```

Make a simple bar chart:

```
title_list %>%
  select(publication_title,
         first_date_held,
         last_date_held,
         country_of_publication) %>%
  arrange(desc(last_date_held)) %>%
  group_by(country_of_publication) %>%
  tally() %>%
  arrange(desc(n)) %>%
```

```
  filter(n>=100) %>%
  ggplot() +
  geom_bar(aes(x = country_of_publication, weight = n))
```



So that's a very quick introduction to R. There's loads of places to learn more.

R-studio cheat sheets

The Pirate's Guide to R, a good beginners guide to base R

R for data science, which teaches the tidyverse in detail

Learn how to make a book like this using Bookdown

Chapter 10

Regular expressions

To be written by Bruno

Chapter 11

Charting over time with a curated dataset

Chapter 12

Network analysis with curated seventeenth-century dataset

Chapter 13

Geocode and map newspaper titles

R is also really good for creating maps, both for visualisations and for spatial analysis.

13.1 Points

Another thing we can do is create a map of all the titles. For this we'll need three things

- A background map of the UK and Ireland
- A count of the total titles for each city
- A list of coordinates for all the cities. This last one is a little trickier than the other two, as I'll explain. Without some manually editing we might end up with some slightly dodgy results. But let's just see where we end up.

13.1.1 Drawing a background map. ‘

Maps is a library for R which, unsurprisingly, contains some map data, including some high resolution maps of the world. To draw the map, first we install the library.

```
install.packages('maps')
```

Next we load two libraries to use: maps and ggplot2. We'll use a function from ggplot2 called `map_data` to turn some data from the maps package into a dataframe, and then we'll use some more plotting functions from ggplot2 to draw the map.

```
library(ggplot2)
library(maps)
```

First we assign a name to the dataframe we'll create with `map_data`

```
worldmap = map_data('world')
```

Take a look at the dataframe we've created:

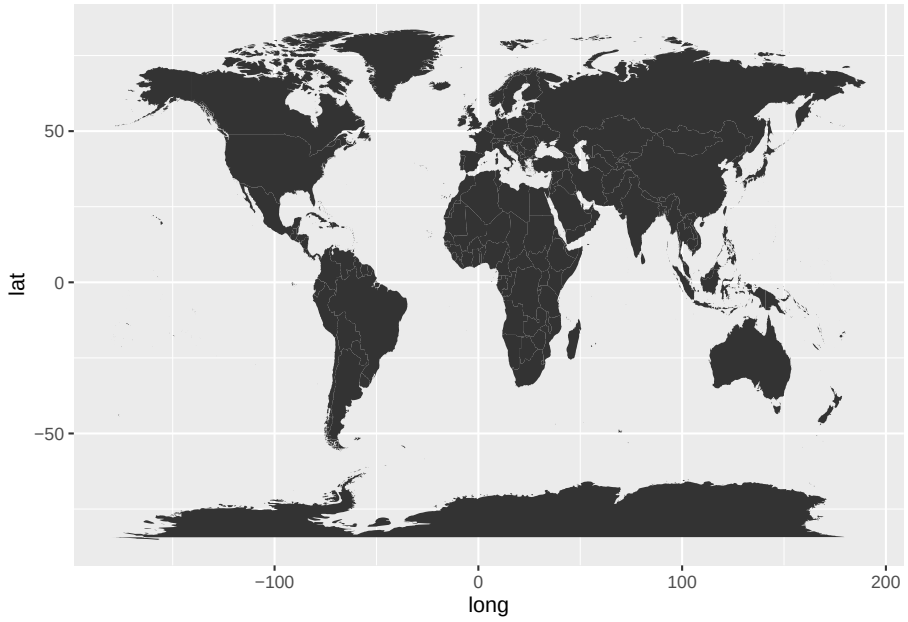
```
head(worldmap, 20)
```

```
##         long      lat group order      region subregion
## 1  -69.89912 12.45200     1     1       Aruba      <NA>
## 2  -69.89571 12.42300     1     2       Aruba      <NA>
## 3  -69.94219 12.43853     1     3       Aruba      <NA>
## 4  -70.00415 12.50049     1     4       Aruba      <NA>
## 5  -70.06612 12.54697     1     5       Aruba      <NA>
## 6  -70.05088 12.59707     1     6       Aruba      <NA>
## 7  -70.03511 12.61411     1     7       Aruba      <NA>
## 8  -69.97314 12.56763     1     8       Aruba      <NA>
## 9  -69.91181 12.48047     1     9       Aruba      <NA>
## 10 -69.89912 12.45200     1    10       Aruba      <NA>
## 12  74.89131 37.23164     2    12 Afghanistan      <NA>
## 13  74.84023 37.22505     2    13 Afghanistan      <NA>
## 14  74.76738 37.24917     2    14 Afghanistan      <NA>
## 15  74.73896 37.28564     2    15 Afghanistan      <NA>
## 16  74.72666 37.29072     2    16 Afghanistan      <NA>
## 17  74.66895 37.26670     2    17 Afghanistan      <NA>
## 18  74.55899 37.23662     2    18 Afghanistan      <NA>
## 19  74.37217 37.15771     2    19 Afghanistan      <NA>
## 20  74.37617 37.13735     2    20 Afghanistan      <NA>
## 21  74.49796 37.05722     2    21 Afghanistan      <NA>
```

It's a big table with about 100,000 rows. Each row has a latitude and longitude, and a group. Each region and sub-region in the dataframe has its own group number. We'll use a function `geom_polygon` which tells ggplot to draw a polygon (a bunch of connected lines) for each group, and display it.

With the `aes()`, x tells ggplot2 the longitude of each point, y the latitude, and group makes sure the polygons are grouped together correctly.

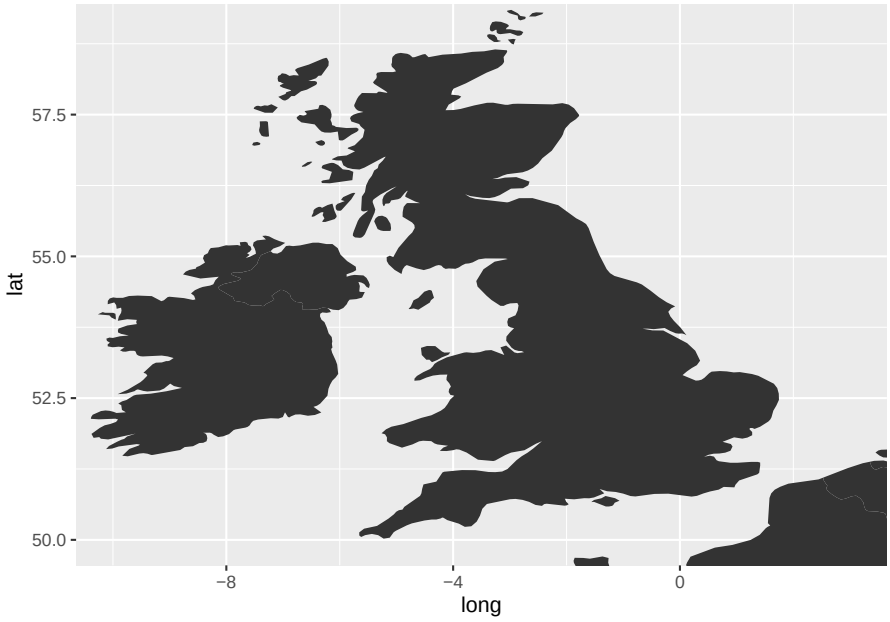
```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group))
```



Right, it needs a *bit* of tweaking. First, we only want to plot points in the UK and Ireland. There's obviously way too much map for this, so the first thing we should do is restrict it to a rectangle which includes those two countries.

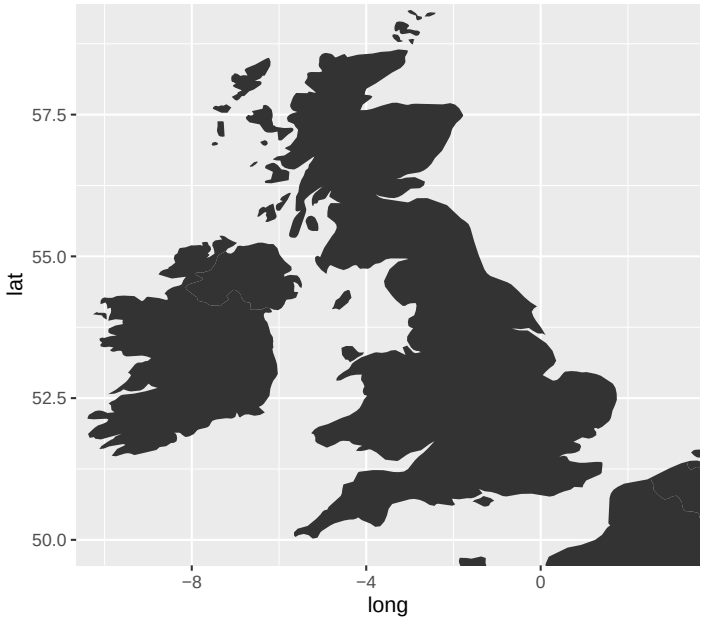
We can do that with `coord_fixed()`. `coord_fixed()` is used to fix the aspect ratio of a coordinate system, but can be used to specify a bounding box by using two of its arguments: `xlim=` and `ylim=`. These each take a vector (a series of numbers) with two numbers. A vector is created using `c()`. Each number specifies the limits for that axis. So `xlim = c(0,10)` means *restrict the x-axis to 0 and 10*. The axes correspond to the lines of longitude (x) and latitude (y). We'll restrict the x-axis to c(-10, 4) and the y-axis to c(54, 60) which should just about cover the UK and Ireland.

```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group)) +
  coord_fixed(xlim = c(-10,3), ylim = c(50, 59))
```



We can also change the ratio of the coordinates using `coord_fixed()`. The default is 1, but by specifying a different one with the argument `ratio =`, that can be changed. Using `ratio = 1.3` results in a less squashed-looking map.

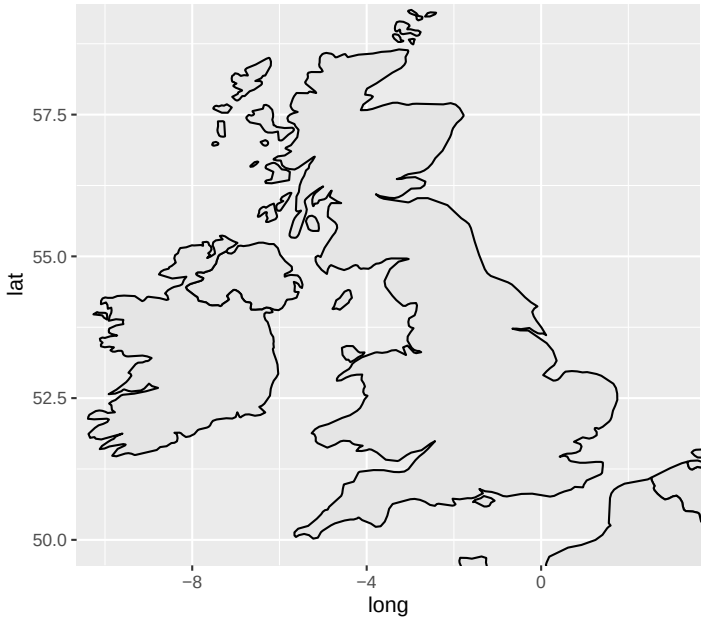
```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group)) +
  coord_fixed(ratio = 1.3, xlim = c(-10,3), ylim = c(50, 59))
```



A couple more things, which I'll run through quickly.

We can specify fill and line colors usings `fill =` and `color =` inside `geom_polygon()` but *outside* `aes()`.

```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group), fill = 'gray90',
  coord_fixed(ratio = 1.3, xlim = c(-10,3), ylim = c(50, 59))
```



We probably don't need the grids or panels in the background. We can get rid of these with `+ theme_void()`, but bear in mind we might need to add some theme elements back, manually.

```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group), fill =
  coord_fixed(ratio = 1.3, xlim = c(-10,3), ylim = c(50, 59)) +
  theme_void()
```

13.1.2 Get a count of the total titles for each city

This next bit uses some of the functions demonstrated in the introduction to R and the tidyverse, namely `group_by()` and `tally()`.

First load the rest of the tidyverse packages.

```
library(tidyverse)
```

Next, load the title list, which can be dowloaded here:

```
title_list = read_csv('data/BritishAndIrishNewspapersTitleList_20191118.csv')
```

We can quite easily make a new data frame, which will just include each location and the total number of instances in the dataset.

```
location_counts = title_list %>%
  group_by(country_of_publication, general_area_of_coverage, coverage_city) %>%
  tally()
```

Arranging these in descending order of their count shows how many of each we have:

```
location_counts %>%
  arrange(desc(n))
```

```
## # A tibble: 2,189 x 4
## # Groups:   country_of_publication, general_area_of_coverage [531]
```

```
##    country_of_publication general_area_of_coverage  coverage_city     n
##    <chr>                  <chr>                     <chr>         <int>
##  1 England                London                    London         5781
##  2 Ireland                Dublin (Ireland : County) Dublin          415
##  3 Scotland               Strathclyde               Glasgow         309
##  4 England                Greater Manchester        Manchester      265
##  5 England                West Midlands             Birmingham      260
##  6 England                Merseyside                Liverpool       220
##  7 England                Avon                      Bristol         175
##  8 Scotland               Lothian                   Edinburgh       162
##  9 England                South Yorkshire           Sheffield       133
## 10 England                Nottinghamshire           Nottingham      127
## # ... with 2,179 more rows
```

13.1.3 Get a list of points.

Well, we've cheated and made this one already, with lots of manual work.

We'll load a dataset containing geocorrected points

```
geocorrected = read_csv('data/geocorrected.csv')
```

This needs a little bit of pre-processing. First use a library called `snakecase` and a function called `to_snakecase()` to standardise the column names.

```
library(snakecase)
colnames(geocorrected)  = to_snake_case(colnames(geocorrected))
```

Then make the coordinate columns numeric using `mutate()`

```
geocorrected = geocorrected %>%
  mutate(wikilat = as.numeric(wikilat)) %>%
  mutate(wikilon = as.numeric(wikilon))
```

```
## Warning: NAs introduced by coercion
```

```
## Warning: NAs introduced by coercion
```

Last, change the NA values to the same format as the ones in the title list.

```
geocorrected = geocorrected %>%
  mutate(country_of_publication = replace(country_of_publication, country_of_publicati
  mutate(coverage_city = replace(coverage_city, coverage_city == 'na', NA))
```

Using `left_join()` we will merge these dataframes, joining up each set of location information to its coordinates and standardised name.

```
lc_with_geo = location_counts %>% left_join(geocorrected, by = c('coverage_city' ,'gene
```

Right, now we're going to use `group_by()` and `tally()` again, this time on the the wikititle, wikilat and wikilon columns. This is because the wikititle is a

standardised title, which means it will group together cities properly, rather than giving a different row for slightly different combinations of the three geographic information columns.

```
lc_with_geo_counts = lc_with_geo %>% group_by(wikititle, wikilat, wikilon) %>% tally(n)
```

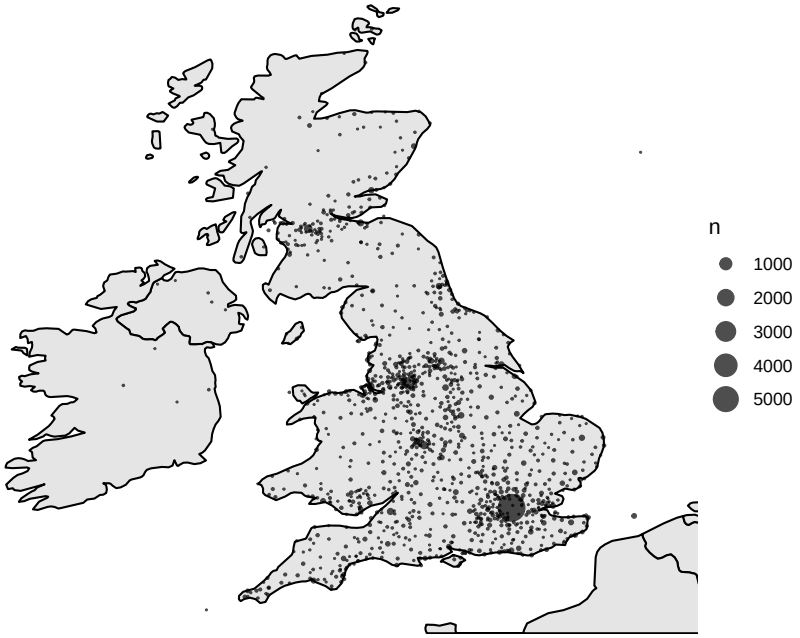
OK, lc_with_geo_counts is what we want to plot. This contains the city title, coordinates and counts for all the relevant places in our dataset. But first we again need the map we created earlier.

```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group), fill = 'gray90',
  coord_fixed(ratio = 1.3, xlim = c(-10,3), ylim = c(50, 59)) +
  theme_void()
```

Now we will plot the cities using `geom_point()` We'll specify the `lc_with_geo_counts` as the argument to `data =` within `geom_point()`. The x axis position of each point is the longitude, and the y axis the latitude. We'll also use the argument `size = n` within the `aes()`, to tell ggplot2 to size the points by the column n, which contains the counts for each of our locations, and the argument `alpha = .7` outside the `aes()`, to make the points more transparent and slightly easier to read overlapping ones.

One last thing we'll add is `+scale_size_area()`. This sizes the points using their radius rather than diameter, which is a more correct way of representing numbers using circles!

```
ggplot() + geom_polygon(data = worldmap, aes(x = long, y = lat, group = group), fill =
  coord_fixed(ratio = 1.3, xlim = c(-10,3), ylim = c(50, 59)) +
  theme_void() + geom_point(data = lc_with_geo_counts, aes(x = as.numeric(wikilon), y =
  scale_size_area()
```



13.2 Choropleth map

13.2.1 Which libraries are needed

```
library(sf)
library(rgdal)
library(broom)
library(rgeos)
```

13.2.2 Download a shapefile

From here

13.2.3 Have a look at the shapefile

```
england_shp = readOGR(dsn = "data/english_ceremonial_counties", layer = "English Ceremo
```

```
## OGR data source with driver: ESRI Shapefile
```

```
## Source: "/Users/yannryan/Desktop/r-projects/r-for-news-data/data/english_ceremonial_counties",
## with 47 features
## It has 2 fields
```

The next thing to do is use a library called broom and a function called `tidy()`. This turns the shapefile into a dataframe which can be read and plotted by ggplot2.

First change the projection:

```
pc <- spTransform( england_shp, CRS( "+init=EPSG:4326" ) )
```

Then use `gSimplify()` to simplify it very slightly:

```
pc <- gSimplify(pc, tol = 0.00001)
```

```
pc_tidy = broom::tidy(pc, region = 'NAME')
```

Add back in the data, which gSimplify doesn't keep:

```
pc_df <- england_shp@data
pc <- sp::SpatialPolygonsDataFrame(pc, pc_df)
```

Finally, do the tidy again. It should work this time.

```
pc_tidy = broom::tidy(pc, region = 'NAME')
```

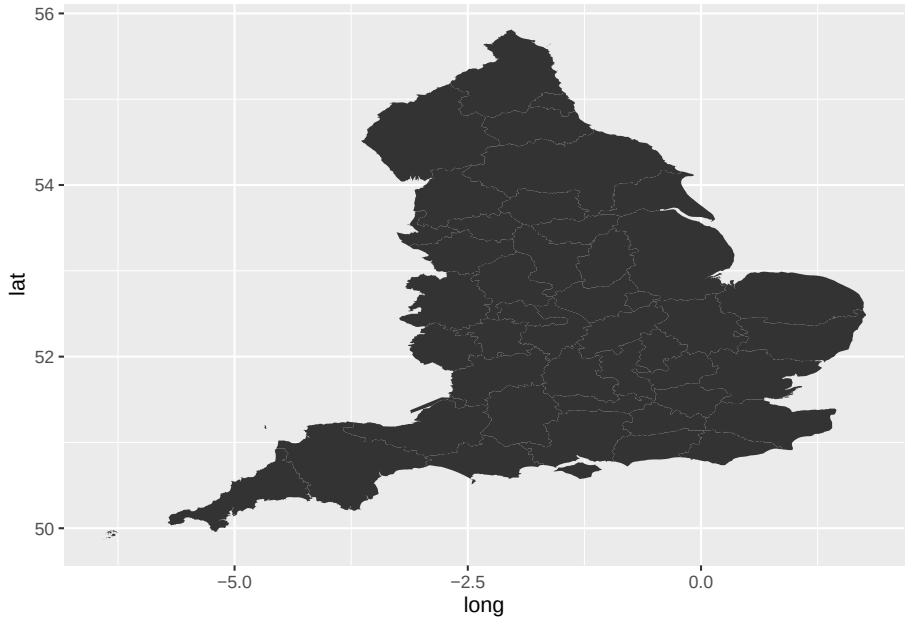
https://gis.stackexchange.com/questions/243569/simplify-polygons-of-sf-object

```
county_totals = title_list %>%
  group_by(general_area_of_coverage) %>%
  tally()
```

```
pc_tidy = broom::tidy(pc, region = 'NAME')
```

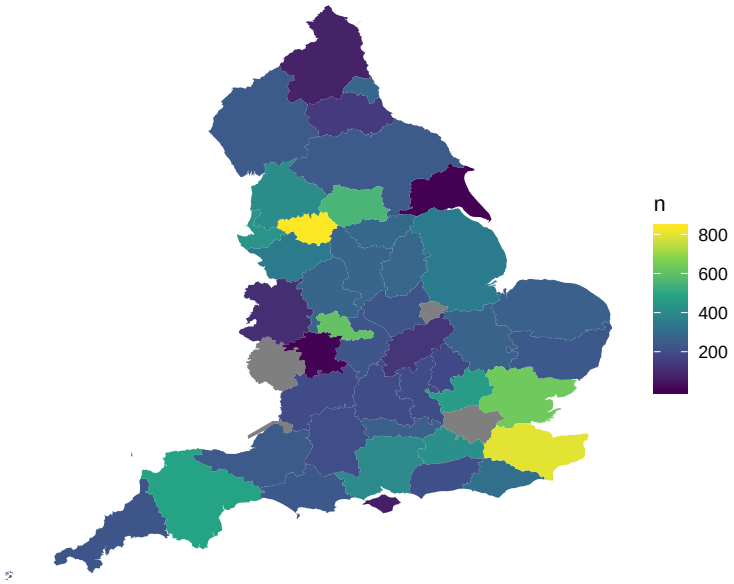
Draw the map:

```
ggplot() +
  geom_polygon(data = pc_tidy, aes(x = long, y = lat, group = group))
```



Add in the counts using `left_join()`

```
ggplot() +
  geom_polygon(data = pc_tidy %>%
                 left_join(county_totals,
                           by = c('id' = 'general_area_of_coverage')),
               aes(x = long, y = lat,
                   fill = n, group = group)) +
  theme_void() +
  coord_fixed(1.3) + scale_fill_viridis_c()
```




```
anti_join(title_list %>%
            filter(country_of_publication == 'England') %>%
            group_by(general_area_of_coverage) %>%
            tally(),pc_tidy %>%
            group_by(id) %>%
            tally() %>%
            select(id), by = c('general_area_of_coverage' = 'id')) %>%
  arrange(desc(n))
```

```
## # A tibble: 274 x 2
##    general_area_of_coverage     n
##    <chr>                    <int>
##  1 London                    5866
##  2 <NA>                       379
##  3 Humberside                 322
##  4 Avon                       321
##  5 Hereford & Worcester       225
##  6 Cleveland                  122
##  7 Jersey                      33
##  8 Isle of Man                 31
##  9 Guernsey                    29
## 10 Great Britain               7
## # ... with 264 more rows
```

Table 13.1: A table of the first 10 rows of the mtcars data.

	mpg	cyl	disp	hp	drat	wt	qsec	vs
Mazda RX4	21.0	6	160.0	110	3.90	2.620	16.46	0
Mazda RX4 Wag	21.0	6	160.0	110	3.90	2.875	17.02	0
Datsun 710	22.8	4	108.0	93	3.85	2.320	18.61	1
Hornet 4 Drive	21.4	6	258.0	110	3.08	3.215	19.44	1
Hornet Sportabout	18.7	8	360.0	175	3.15	3.440	17.02	0
Valiant	18.1	6	225.0	105	2.76	3.460	20.22	1
Duster 360	14.3	8	360.0	245	3.21	3.570	15.84	0
Merc 240D	24.4	4	146.7	62	3.69	3.190	20.00	1
Merc 230	22.8	4	140.8	95	3.92	3.150	22.90	1
Merc 280	19.2	6	167.6	123	3.92	3.440	18.30	1

```
anti_join(pc_tidy %>% group_by(id) %>% tally() %>% select(id),title_list %>% filter(cou
```

```
## # A tibble: 4 x 1
##   id
##   <chr>
## 1 County of Bristol
## 2 Greater London
## 3 Herefordshire
## 4 Rutland
```

```
knitr::kable(
  head(mtcars[, 1:8], 10), booktabs = TRUE,
  caption = 'A table of the first 10 rows of the mtcars data.'
)
```

13.2.4 Count the general area of coverage field

13.2.5 Change some of the names to fit the shapefile

13.2.6

Part III

Part III: Examples

Chapter 14

Text mining

14.1 What were the most common words used in newspaper titles in the nineteenth century?_

Figure 14.1: title

14.1.1 Titles don't just help you identity a newspaper, but they might tell you a little bit about the time in which they were established. With a bibliographic list of *all* our UK and Irish titles, we can count the most frequent words and track them over time and place, using text mining and data analysis.

14.1.2 What is this document?

This is a markdown file, made from a Jupyter notebook. A jupyter notebook is usually an interactive document which can contain code, images and text, and a markdown file is a static version of that. Each bit of code runs in a cell, and the output is displayed directly below.

The code I've used is R, which is a language particularly good for data analysis, but another language, Python, is probably used in Jupyter more frequently. If you're going to work in R, I would recommend downloading R-Studio to do all your work, which could then be copied-and-pasted over the Jupyter notebook if needed, like I've done here.

There are tonnes of places to get started working with R, Python, Jupyter notebooks and so forth, and we would recommend looking here in the first instance:

https://programminghistorian.org/

https://software-carpentry.org/

First we need to load some libraries which we'll use. A library is just a bunch of functions* grouped together, usually with a particular overall purpose or theme.

'tidyverse' is actually a number of libraries with hundreds of useful functions to make lots of data analysis easier. It includes a very powerful plotting library called 'ggplot2'.

It's usually the first thing I load, before even deciding what I'm going to do with my data.

'readxl' is a library which.. reads excel files..

Lots of this code uses something called piping. This is a function in one of our tidyverse libraries which allows you to do something to your data, and then pass it along to *another* function using this symbol: %>%

It allows you to string lots of changes to your data together in one block of code, so you might filter it, then pass the filtered data to another function which summarises it, and pass it on to another function which plots it as a graph.

* You might say a function is a pre-made block of code which does something to some data. It has a name and often one or more *arguments*. The first argument

is often a space for you to specify the thing you want to do the function on, and subsequent arguments might be additional parameters.

```
library(tidyverse)
library(readxl)
```

The first thing we do is load the whole title list as a variable called 'working_list', specifying the sheet of the excel file we'd like to use. We'll dive a little deeper into the structure and how we might filter in another notebook.

```
working_list <- read_csv(
"BritishAndIrishNewspapersTitleList_20191118.csv",
local = locale(encoding = "latin1"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   title_id = col_double(),
##   nid = col_double(),
##   nlp = col_double(),
##   first_date_held = col_double(),
##   publication_date_one = col_double(),
##   publication_date_two = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

Let's just look at the nineteenth century - we'll use the filter() function from dplyr, which is one of the libraries in the tidyverse. We then use %>% to pipe the filtered data to a function called head(), which displays a set number of rows of your data frama - useful for taking a peek at the structure.

```
working_list %>%
  filter(last_date_held>1799) %>%
  filter(first_date_held<1900) %>%
head(2)
```

```
## # A tibble: 2 x 24
##   title_id   nid   nlp publication_tit~ edition preceding_titles
##      <dbl> <dbl> <dbl> <chr>            <chr>   <chr>
## 1 13774146    NA    NA London gazette   <NA>    Continues: Oxfo~
## 2 13943676 32666   537 The Newcastle C~ <NA>    <NA>
## # ... with 18 more variables: succeeding_titles <chr>,
## #   place_of_publication <chr>, country_of_publication <chr>,
## #   general_area_of_coverage <chr>, coverage_city <chr>,
## #   first_geographical_subject_heading <chr>,
## #   subsequent_geographical_subject_headings <chr>, first_date_held <dbl>,
## #   last_date_held <chr>, publication_date_one <dbl>,
## #   publication_date_two <dbl>, current_publication_frequency <chr>,
```

```
## #   publisher <chr>, holdings_more_information <chr>,
## #   free_text_information_about_dates_of_publication <chr>,
## #   online_status <chr>, link_to_british_newspaper_archive <chr>,
## #   explore_link <chr>
```

We have some duplicated newspaper titles in this dataframe. We can get rid of these for this analysis, though there are reasons we left them in which I won't go into now. We can delete any duplicated titles using a function called distinct() on NID id field.

```
working_list %>%
  filter(last_date_held>1799) %>%
  filter(first_date_held<1900) %>%
  distinct(NID, .keep_all = TRUE) %>%
head(2)
```

```
## Warning: Trying to compute distinct() for variables not found in the data:
## - `NID`
## This is an error, but only a warning is raised for compatibility reasons.
## The operation will return the input unchanged.
```

```
## # A tibble: 2 x 24
##   title_id   nid   nlp publication_tit~ edition preceding_titles
##      <dbl> <dbl> <dbl> <chr>            <chr>   <chr>
## 1 13774146    NA    NA London gazette   <NA>    Continues: Oxfo~
## 2 13943676 32666   537 The Newcastle C~ <NA>    <NA>
## # ... with 18 more variables: succeeding_titles <chr>,
## #   place_of_publication <chr>, country_of_publication <chr>,
## #   general_area_of_coverage <chr>, coverage_city <chr>,
## #   first_geographical_subject_heading <chr>,
## #   subsequent_geographical_subject_headings <chr>, first_date_held <dbl>,
## #   last_date_held <chr>, publication_date_one <dbl>,
## #   publication_date_two <dbl>, current_publication_frequency <chr>,
## #   publisher <chr>, holdings_more_information <chr>,
## #   free_text_information_about_dates_of_publication <chr>,
## #   online_status <chr>, link_to_british_newspaper_archive <chr>,
## #   explore_link <chr>
```

Select only the information we need. Let's create a new dataframe, run the filter and distinct functions, and select some date information along with the titles, Afterwards we might want to choose geographic information instead, so we'll keep our original data frame:

```
titles_dates = working_list %>%
 filter(last_date_held>1799) %>%
  filter(first_date_held<1900) %>%
  distinct(NID, .keep_all = TRUE) %>%
  select(publication_title, first_date_held)
```

```
## Warning: Trying to compute distinct() for variables not found in the data:
## - `NID`
## This is an error, but only a warning is raised for compatibility reasons.
## The operation will return the input unchanged.
```

To count the title keywords, we can tokenise our data. This splits everything into individual words. For this we need to load a library called 'tidytext', which contains lots of functions for text mining.

```
library(tidytext)
```

```
tokenised_titles_dates = titles_dates %>%
  unnest_tokens(word, publication_title)
```

Now we'll get rid of stopwords - the very frequently-used words like 'the' or 'an' and so forth. It's not always appropriate to remove stopwords, and in fact sometimes they are the most interesting, but I think here it will make things easier to manage.

```
data(stop_words) # this loads a dataset of stopwords
```

```
tokenised_titles_dates = tokenised_titles_dates %>%
  anti_join(stop_words) # this does an 'anti-join' which removes any word in one list which also
```

```
## Joining, by = "word"
```

Let's do some simple analysis first. We can count the most common words overall:

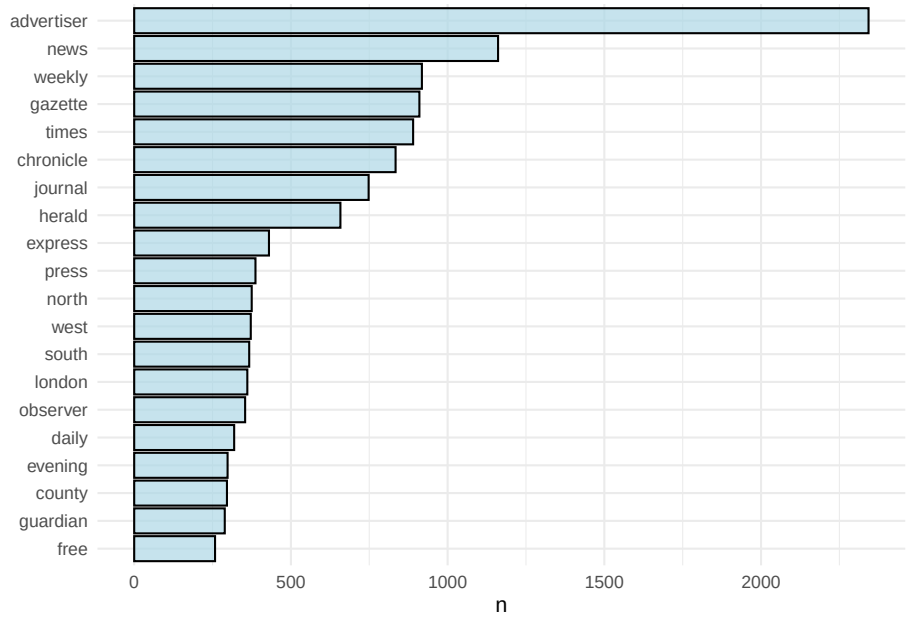
```
tokenised_titles_dates %>%
  count(word, sort = TRUE) %>% head(20)
```

```
## # A tibble: 20 x 2
##    word           n
##    <chr>      <int>
##  1 advertiser  2343
##  2 news        1161
##  3 weekly       918
##  4 gazette      910
##  5 times        890
##  6 chronicle    834
##  7 journal      748
##  8 herald       658
##  9 express      430
## 10 press        387
## 11 north        375
## 12 west         372
## 13 south        367
## 14 london       361
```

```
## 15 observer     354
## 16 daily        319
## 17 evening      298
## 18 county       296
## 19 guardian     289
## 20 free         258
```

We can turn it into a bar chart.

```
tokenised_titles_dates %>%
  count(word, sort = TRUE) %>%
  mutate(word = reorder(word, n)) %>%
 head(20) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



Advertiser is the most popular word in a title! Lots of other words are at the top which might be expected, like 'news', 'daily', 'evening' and so forth.

It might be more interesting to look at the changes in the top words over time.

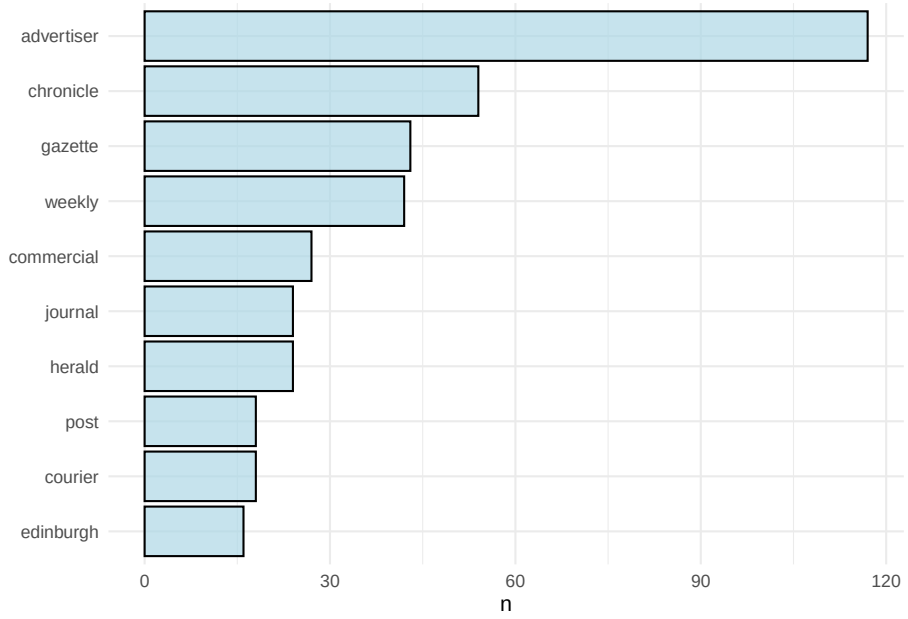
This adds a new column with the date 'floored' to the previous 20. When we group and count again, everything between 1800 and 1819 will become 1800, everything between 1820 and 1839 will become 1820 and so forth.

```
tokenised_titles_dates = tokenised_titles_dates %>%
  mutate(timespan = first_date_held - first_date_held %% 20)
```

Now look at the top ten title words for each of these twenty-year periods:

1800 - 1819:

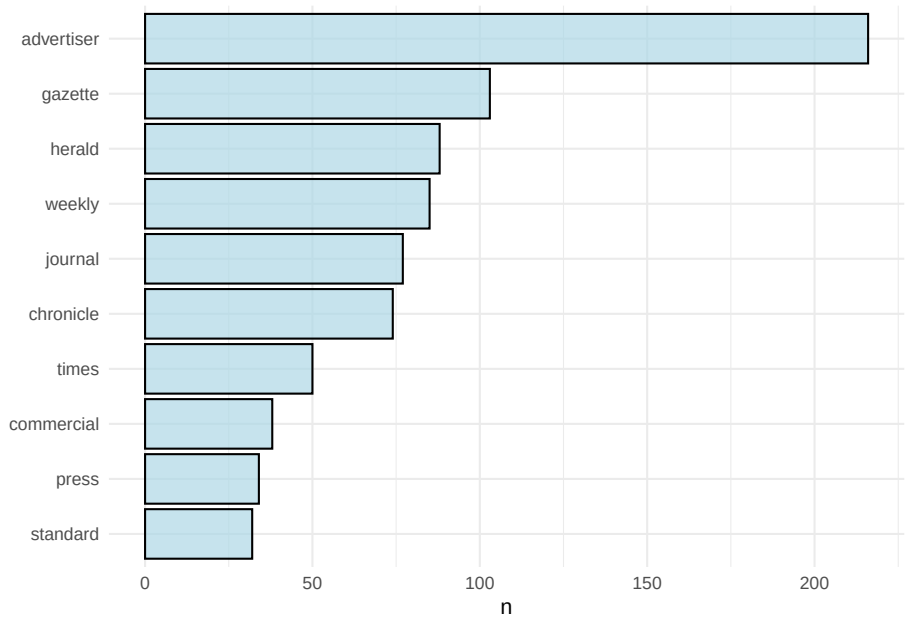
```
tokenised_titles_dates %>%
  filter(timespan == '1800') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  head(10) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



1820 - 1839:

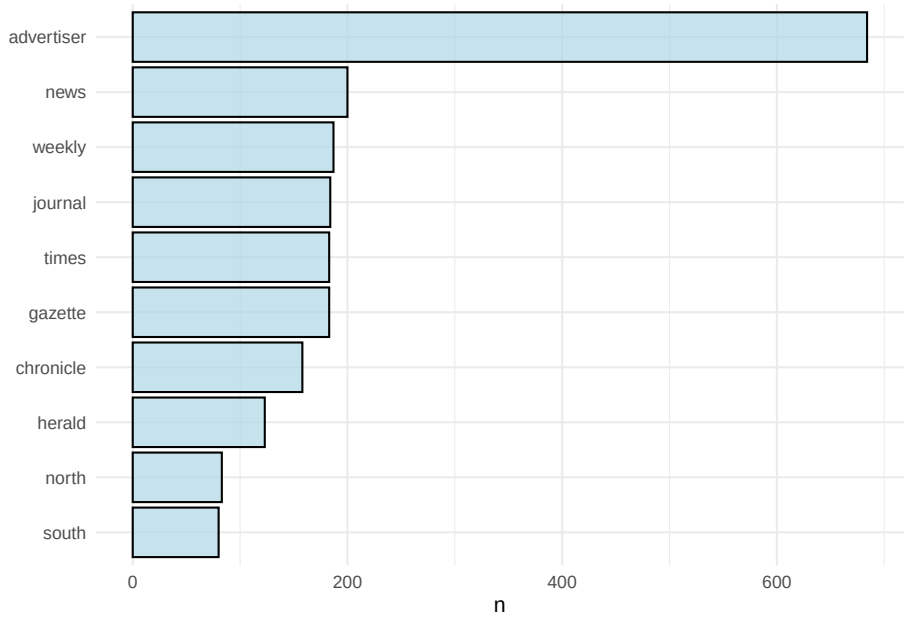
```
tokenised_titles_dates %>%
  filter(timespan == '1820') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
```

```
  head(10) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



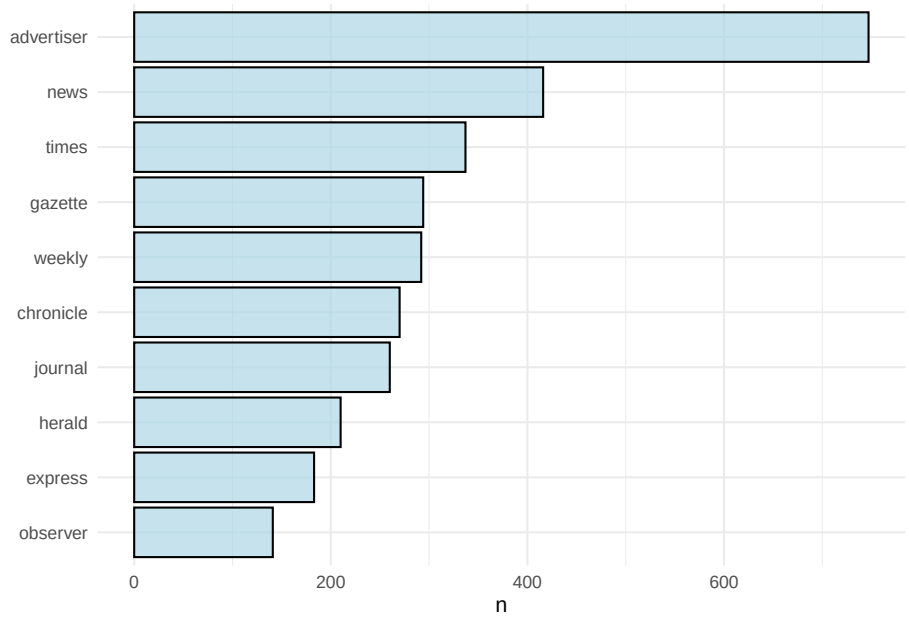
1840 - 1859 (it's interesting how 'British' has fallen out of the top ten, and 'north' and 'south' have been bumped up. Even something this simple can confirming interesting things about the growth of the regional press!

```
tokenised_titles_dates %>%
  filter(timespan == '1840') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  head(10) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



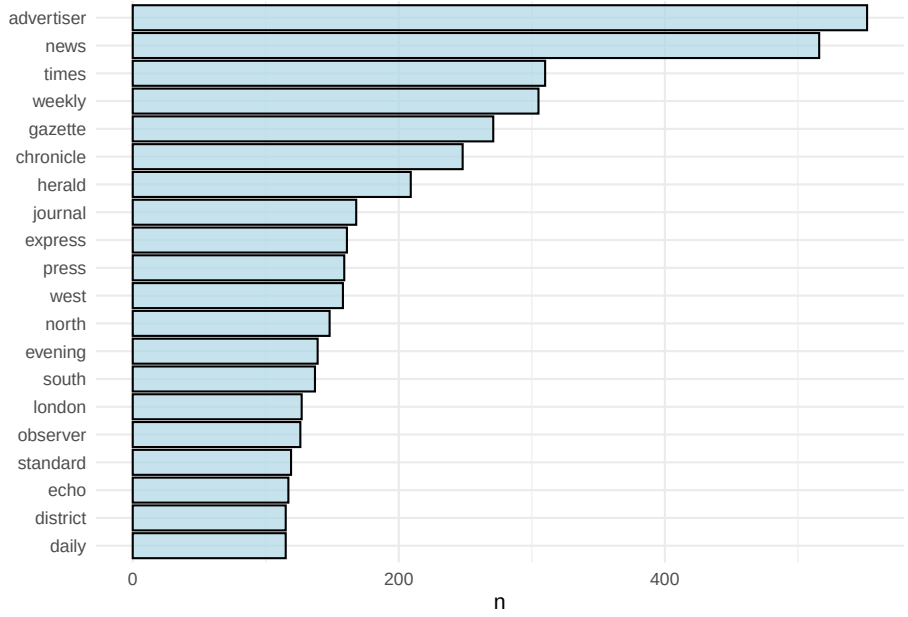
1860 - 1879:

```
tokenised_titles_dates %>%
  filter(timespan == '1860') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  head(10) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



1880 - 1899

```
tokenised_titles_dates %>%
  filter(timespan == '1880') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  filter(n > 100) %>%
  mutate(word = reorder(word, n)) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



There might be better ways of counting. How about a line chart which tracks a certain number of keywords over time?

We know that the number of titles per year increases a lot over the century, so we'll need to make a relative rather than absolute values. This next bit adds a count of *all* the words per decade, and puts it beside each word. Then we can divide one by the other and get a fraction. I'll show the first few lines of each dataframe to show what I mean.

```
title_words = tokenised_titles_dates %>%
  mutate(decade = first_date_held - first_date_held %% 10) %>%
  count(decade, word, sort = TRUE)
head(title_words, 5)
```

```
## # A tibble: 5 x 3
##   decade word           n
##    <dbl> <chr>      <int>
## 1   1850 advertiser   506
## 2   1860 advertiser   425
## 3   1870 advertiser   322
## 4   1880 advertiser   285
## 5   1890 news         276
```

```
total_words <- title_words %>%
  group_by(decade) %>%
  summarize(total = sum(n))
```

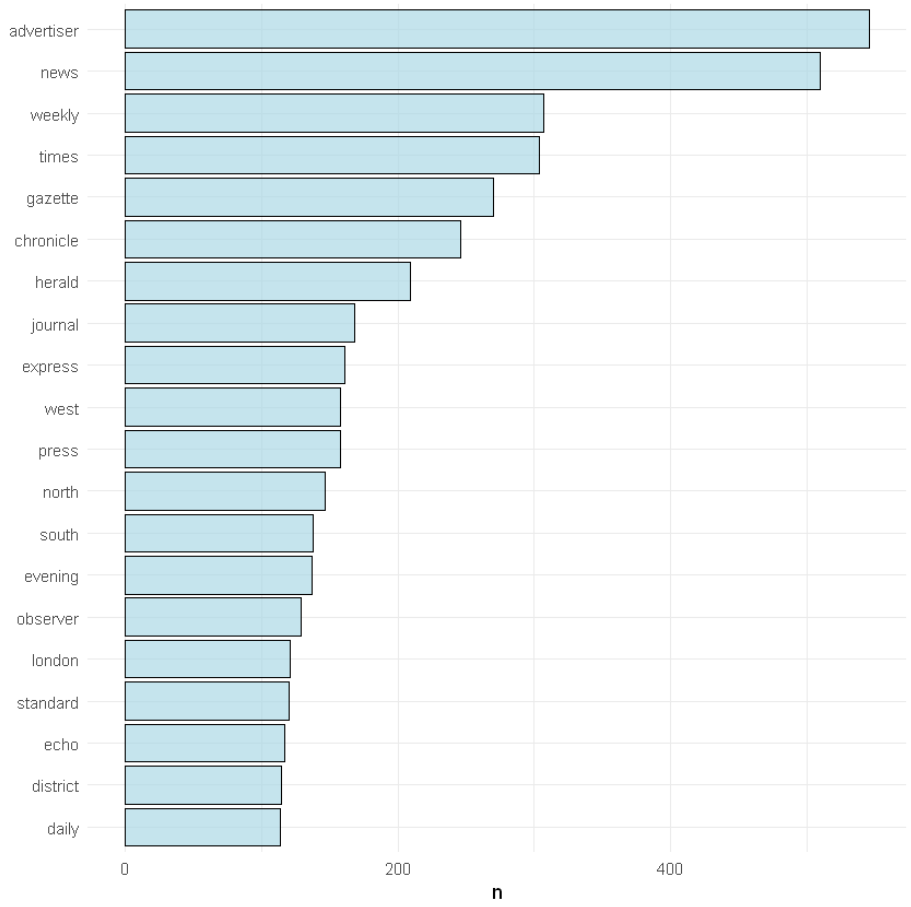


Figure 14.2: png

```
head(total_words, 5)
```

```
## # A tibble: 5 x 2
##   decade total
##    <dbl> <int>
## 1   1660     2
## 2   1710     2
## 3   1720    26
## 4   1730    17
## 5   1740    21
```

```
title_words <- left_join(title_words, total_words)
```

```
## Joining, by = "decade"
```

```
head(title_words, 5)
```

```
## # A tibble: 5 x 4
##   decade word           n total
##    <dbl> <chr>      <int> <int>
## 1   1850 advertiser   506  5416
## 2   1860 advertiser   425  5835
## 3   1870 advertiser   322  6581
## 4   1880 advertiser   285  6396
## 5   1890 news         276  6304
```

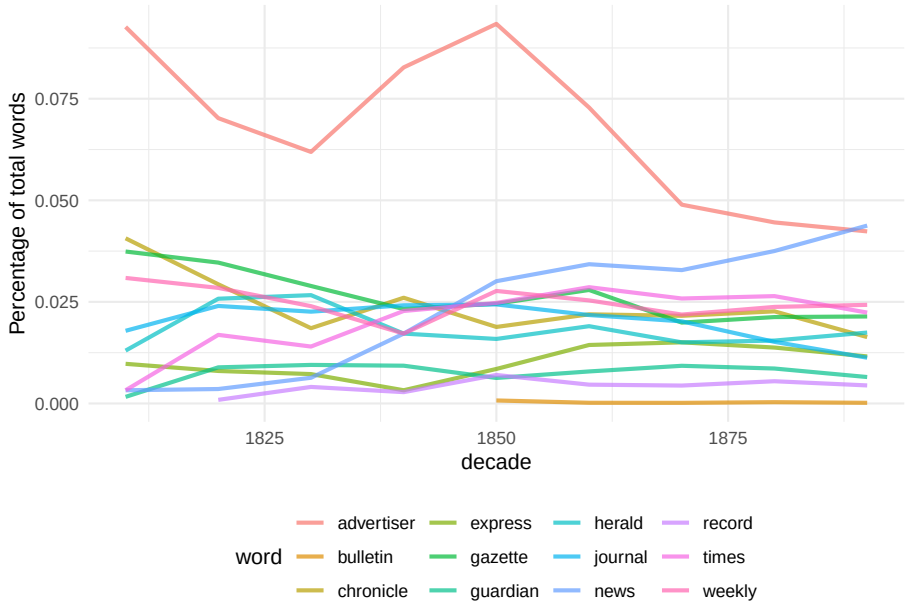
We'll pick a list of terms to plot, because otherwise it'll be unreadable.

```
top_terms = c('advertiser',
              'news',
              'weekly',
              'times',
              'gazette',
              'chronicle',
              'herald',
              'journal',
              'express',
              'bulletin',
              'record',
              'guardian',
              'express')
```

Now we draw a plot, using n/total as the y-axis variable to plot the fraction of the total:

```
title_words %>%
  filter(word %in% top_terms) %>%
  filter(decade >1800) %>%
```

```
  ggplot(aes(x = decade, y = n/total, color = word)) +
  geom_line(size = 1, alpha = .7,stat = 'identity') +
  theme_minimal() +
  ylab(label = "Percentage of total words") + theme(legend.position = 'bottom')
```



Well, that's quite interesting. Advertiser declines in the second half of the century, and is overtaken by 'news' right at the end.

14.1.3 How about differences by country?

Go back to the working list and make a version with country information:

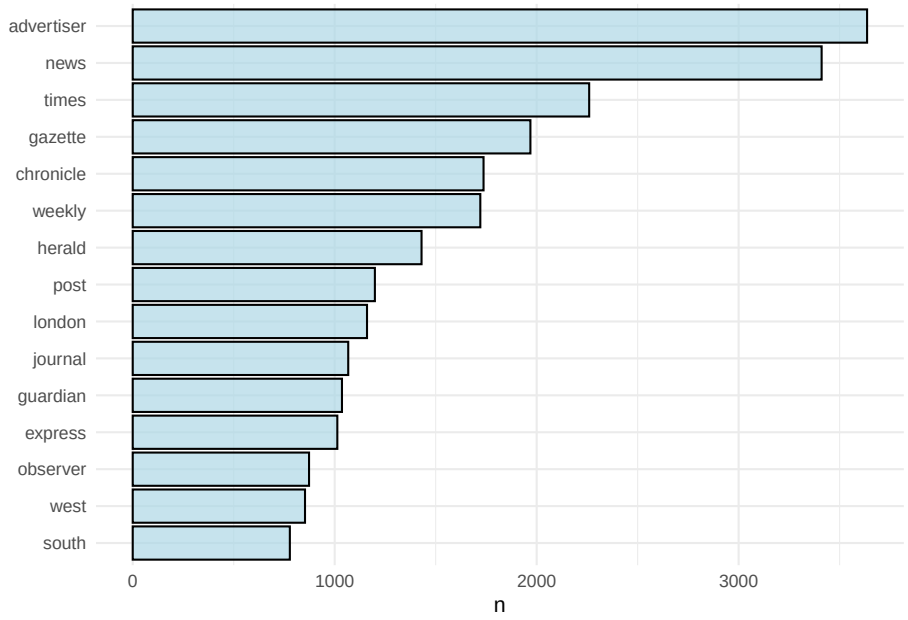
```
titles_countries = working_list %>%
  select(publication_title, country_of_publication)

tokenised_titles_countries = titles_countries %>%
  unnest_tokens(word, publication_title)

tokenised_titles_countries = tokenised_titles_countries %>%
  anti_join(stop_words)
```

```
## Joining, by = "word"
```

Let's make an overall chart for each country:

England:

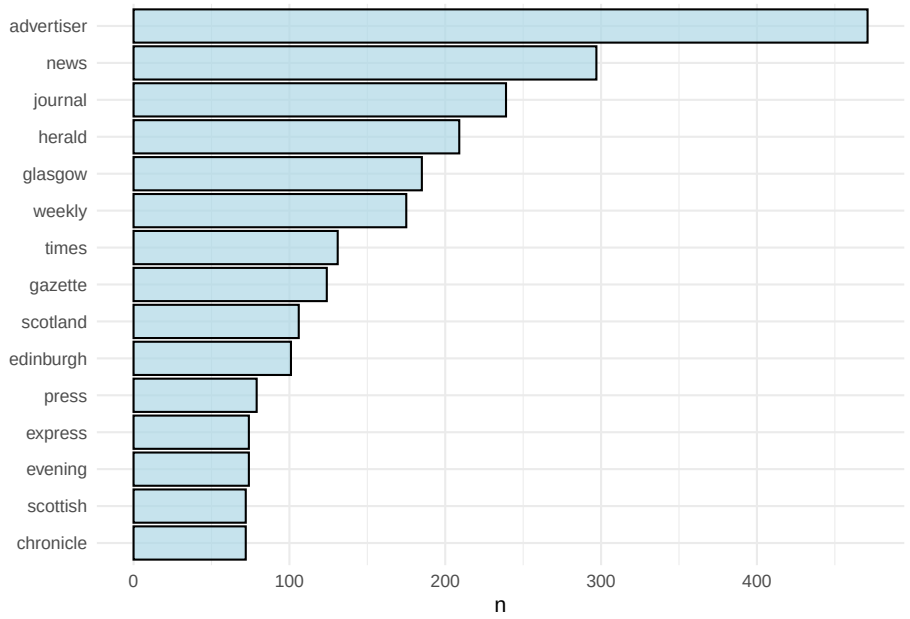
```
tokenised_titles_countries %>%
  filter(country_of_publication == 'England') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  head(15) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



Scotland:

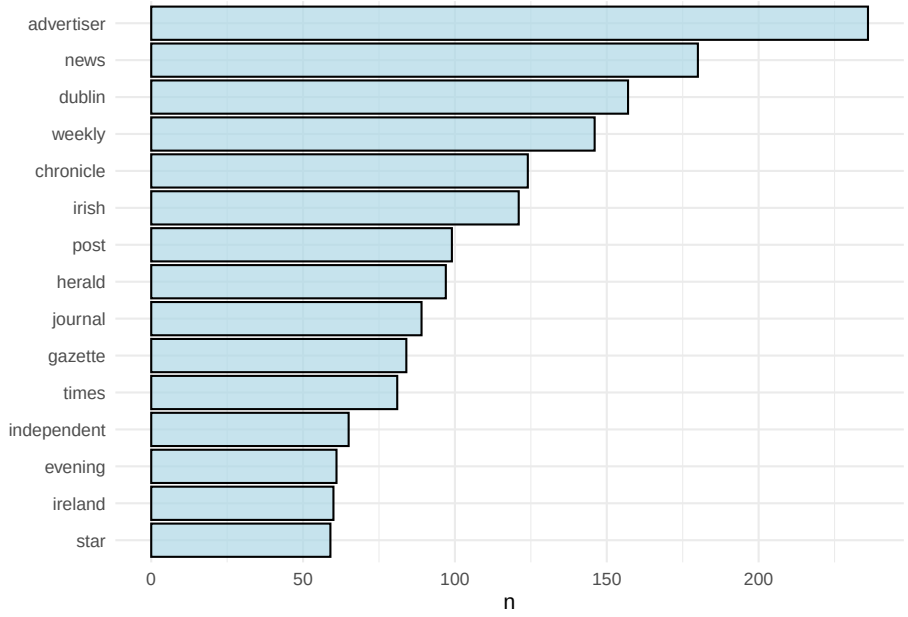
```
tokenised_titles_countries %>%
  filter(country_of_publication == 'Scotland') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  filter(n > 40) %>%
  mutate(word = reorder(word, n)) %>%
  head(15) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
```

```
  coord_flip()
```



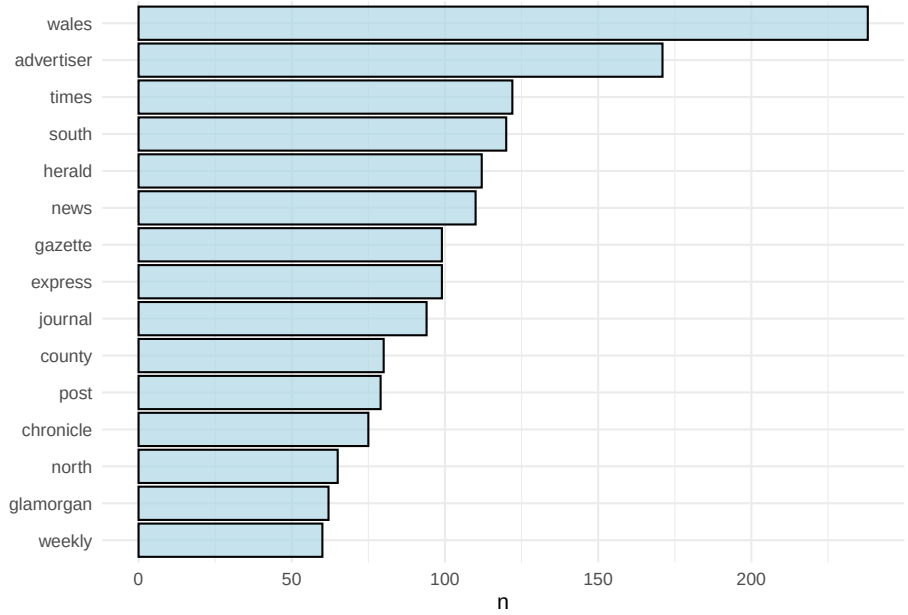
Ireland:

```
tokenised_titles_countries %>%
  filter(country_of_publication %in% c('Ireland', 'Northern Ireland')) %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
  head(15) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



Wales:

```
tokenised_titles_countries %>%
  filter(country_of_publication == 'Wales') %>%
  count(word, sort = TRUE) %>%
  ungroup() %>%
  mutate(word = reorder(word, n)) %>%
 head(15) %>%
  ggplot(aes(word, n)) +
  geom_col(fill = 'lightblue', color = 'black', alpha = .7) +
  xlab(NULL) +
  theme_minimal() +
  coord_flip()
```



How about looking for the most unique terms for each country? That might tell us something interesting.

First we'll select just the key countries:

```
countryList = c('England', 'Ireland', 'Wales', 'Scotland', 'Northern Ireland')
```

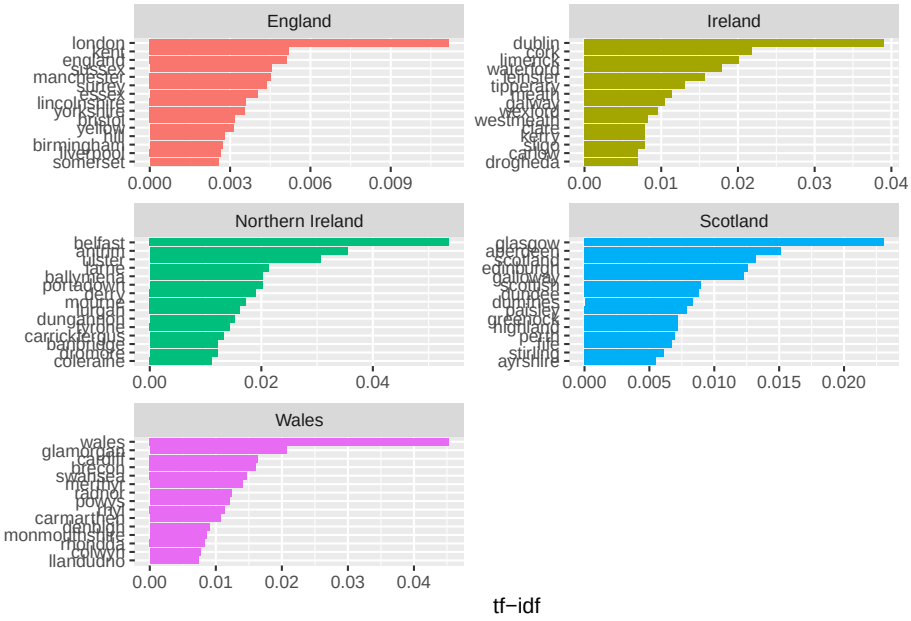
Next we'll use a function which gives the 'tf-idf' score for each word. This measures the frequency of the word in comparison to its frequency in all other countries, giving us words that are more unique to titles from that country.

```
total_by_country = tokenised_titles_countries %>%
  filter(country_of_publication %in% countryList) %>%
  count(country_of_publication, word, sort = TRUE)


total_by_country <- total_by_country %>%
  bind_tf_idf(word, country_of_publication, n)
```

```
total_by_country %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(country_of_publication) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = country_of_publication)) +
  geom_col(show.legend = FALSE) +
```

```
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~country_of_publication, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```



Unsurprisingly, this mostly gives us placenames, as used in the titles, which are obviously only going to be used in one country. There's a couple of interesting things: 'Wales' and 'Scotland' have a high score, but not 'Ireland'. Why would Ireland not be used in a newspaper title in the same way as Welsh or Scottish newspapers?

We need a way to try and filter out geographic places as they're drowning out other potentially interesting terms. We can make a list of places from our original title list which would be a good start.

```
all_places = read_csv(
"BritishAndIrishNewspapersTitleList_20191118.csv",
local = locale(encoding = "latin1"))
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   title_id = col_double(),
##   nid = col_double(),
##   nlp = col_double(),
```

```
##   first_date_held = col_double(),
##   publication_date_one = col_double(),
##   publication_date_two = col_double()
## )
```

```
## See spec(...) for full column specifications.
```

```
list_of_places = c(all_places$first_geographical_subject_heading,
                           all_places$subsequent_geographical_subject_headings,
                           all_places$general_area_of_coverage,
                           all_places$coverage_city,
                           all_places$place_of_publication,
                           all_places$country_of_publication)

list_of_places = as.data.frame(list_of_places) %>%
  group_by(list_of_places) %>% count() %>% select(list_of_places)
```

```
## Warning: Factor `list_of_places` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

```
## Warning: Factor `list_of_places` contains implicit NA, consider using
## `forcats::fct_explicit_na`
```

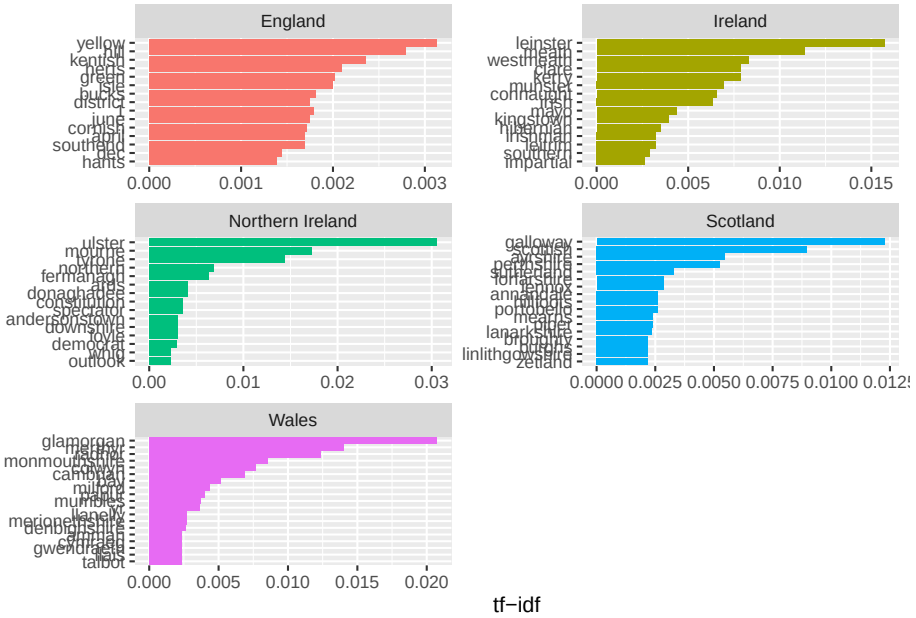
```
list_of_places = tolower(list_of_places$list_of_places)
```

It's a bit crude but it's given us a list of 5,000 or so places which we can use to filter our word list.

Plotting the filtered list:

```
total_by_country %>%
  filter(!word %in% list_of_places) %>%
  arrange(desc(tf_idf)) %>%
  mutate(word = factor(word, levels = rev(unique(word)))) %>%
  group_by(country_of_publication) %>%
  top_n(15) %>%
  ungroup() %>%
  ggplot(aes(word, tf_idf, fill = country_of_publication)) +
  geom_col(show.legend = FALSE) +
  labs(x = NULL, y = "tf-idf") +
  facet_wrap(~country_of_publication, ncol = 2, scales = "free") +
  coord_flip()
```

```
## Selecting by tf_idf
```



There are still lots of geographic terms, but there are some other words 'unique' to each country. Ireland and Northern Ireland high-scoring words are particularly interesting (of course we're looking at 19th century titles so the division is meaningless, but it still represents some kind of regionality). Only Irish titles tend to have ideological terms like 'nationalist', 'democrat', 'loyalty', 'impartial' and so forth. Newspapers with 'illustrated' or 'Sunday' in the title are unique to England, and, intriguingly, 'visitors'.

Chapter 15

Extract text from HMD titles

15.1 Build functions

15.2 Extract text

Chapter 16

Term Frequencies

Chapter 17

Text reuse

Chapter 18

Sentiment analysis

Chapter 19

Existing work

Make a list of tutorials, research projects etc.

Things/people to reference:

Laurel Brake, Jim Mussell, other HMD advisory board people, Joad Raymond (don't forget about News networks)

Bob Nicholson, Melodee Beals, Ryan Cordell

Bloggers like Bruno Rodrigues

The BL Labs stuff on github

Other code repositories would be good.

Chapter 20

Further reading

Cordell, Ryan, "'Q I-Jtb the Raven": Taking Dirty Ocr Seriously', *Book History*, 20.1 (2017), 188–225 <https://doi.org/10.1353/bh.2017.0006>

Fyfe, Paul, 'An Archaeology of Victorian Newspapers', *Victorian Periodicals Review*, 49.4 (2016), 546–77 <https://doi.org/10.1353/vpr.2016.0039>

Hill, Mark John, and Simon Hengchen, 'Quantifying the Impact of Dirty Ocr on Historical Text Analysis: Eighteenth Century Collections Online as a Case Study', *Digital Scholarship in the Humanities : DSH*, 2019 <https://doi.org/10.1093/llc/fqz024>

King, Ed, 'British Library Digitisation: Access and Copyright', 2008

———, 'Digitisation of British Newspapers 1800-1900', 2007 <https://www.gale.com/intl/essays/ed-king-digitisation-of-british-newspapers-1800-1900> [accessed 2007]

Mussell, James, 'Elemental Forms: Elemental Forms: The Newspaper as Popular Genre in the Nineteenth Century', *Media History*, 20.1 (2014), 4–20 <https://doi.org/10.1080/13688804.2014.880264>

Piotrowski, Michael, 'Natural Language Processing for Historical Texts', *Synthesis Lectures on Human Language Technologies*, 5.2 (2012), 1–157 <https://doi.org/10.2200/s00436ed1v01y201207hlt017>

Shaw, Jane, '10 Billion Words: The British Library British Newspapers 1800-1900 Project: Some Guidelines for Large-Scale Newspaper Digitisation', 2005 <https://archive.ifla.org/IV/ifla71/papers/154e-Shaw.pdf>

———, 'Selection of Newspapers', *British Library Newspapers*, 2007 <https://www.gale.com/intl/essays/jane-shaw-selection-of-newspapers>

Smith, David, and Ryan Cordell, 'A Research Agenda for Historical and Multilingual Optical Character Recognition', *Northeaster University*, 2018

Smits, Thomas, 'Making the News National: Using Digitized Newspapers to Study the Distribution of the Queen's Speech by W. H. Smith & Son, 1846–1858', *Victorian Periodicals Review*, 49.4 (2016), 598–625 <https://doi.org/10.1353/vpr.2016.0041>