

Hacking Religion: TRS & Data Science in Action

Jeremy H. Kidwell

2023-09-29

Table of contents

Preface	5
1 Introduction: Hacking Religion	6
1.1 Who this book is for	6
1.2 Why this book?	6
1.3 The hacker way	6
1.4 Why programmatic data science?	6
1.5 Learning to code: my way	7
1.6 Getting set up	8
2 The 2021 UK Census	10
2.1 Your first project: the UK Census	10
2.2 Examining data:	11
2.3 Parsing and Exploring your data	13
2.4 Making your first data visualisation: the humble bar chart	13
2.4.1 Base R	14
2.4.2 GGPLOT	14
2.5 Is your chart accurate? Telling the truth in data science	21
2.6 Making our script reproducible	23
2.7 Multifactor Visualisation	23
References	33
3 Survey Data: Spotlight Project	34
4 Loading in some data	36
5 How can you ask about religion?	38
6 Q56 follow-ups	43

7 Religious Affiliation c - Muslim Denomination Sub-question	44
8 Q57	45
9 Religiosity	46
10 Q58	47
11 Faceted plot working with 3x3 grid	48
12 Q59	49
13 Faceted plot working with 3x3 grid	50
14 Comparing with attitudes surrounding climate change	51
15 Q6	52
16 Subsetting	53
16.1 Q57 subsetting based on Religiosity _____	53
_____	
16.2 Subsetting based on Spirituality _____	53
_____	
16.2.1 Nature relatedness _____	53
_____	
17 Calculate overall mean nature-relatedness score based on six questions:	54
18 Create low/med/high bins based on Mean and +1/-1 Standard Deviation	55
18.0.1 Spirituality scale _____	55
_____	
19 Calculate overall mean spirituality score based on six questions:	56
20 Create low/med/high bins based on Mean and +1/-1 Standard Deviation	57
References	58
21 Mapping churches: geospatial data science	59

22 Administrative shapes - the UK	62
23 Load in Ordnance Survey OpenMap Points Data	65
References	69
24 Data scraping, corpus analysis and wordclouds	70
References	71
25 Summary	72
References	73

Preface

This is a Quarto book.

To learn more about Quarto books visit <https://quarto.org/docs/books>.

1 Introduction: Hacking Religion

1.1 Who this book is for

1.2 Why this book?

1.3 The hacker way

1. Tell the truth
2. Do not deceive using beauty
3. Work transparently: research as open code using open data
4. Draw others in: produce reproducible research
5. Learn by doing

1.4 Why programmatic data science?

This isn't just a book about data analysis, I'm proposing an approach which might be thought of as research-as-code, where you write out instructions to execute the various steps of work. The upside of this is that other researchers can learn from your work, correct and build on it as part of the commons. It takes a bit more time to learn and set things up, but the upside is that you'll gain access to a set of tools and a research philosophy which is much more powerful.

1.5 Learning to code: my way

This guide is a little different from other textbooks targetting learning to code. I remember when I was first starting out, I went through a fair few guides, and they all tended to spend about 200 pages on various theoretical bits, how you form an integer, or data structures, subroutines, or whatever, such that it was weeks before I got to actually *do* anything. I know some people, may prefer this approach, but I dramatically prefer a problem-focussed approach to learning. Give me something that is broken, or a problem to solve, which engages the things I want to figure out and the motivation for learning just comes much more naturally. And we know from research in cognitive science that these kinds of problem-focussed approaches can tend to facilitate faster learning and better retention, so it's not just my personal preference, but also justified! It will be helpful for you to be aware of this approach when you get into the book as it explains some of the editorial choices I've made and the way I've structured things. Each chapter focusses on a *problem* which is particularly salient for the use of data science to conduct research into religion. That problem will be my focal point, guiding choices of specific aspects of programming to introduce to you as we work our way around that data set and some of the crucial questions that arise in terms of how we handle it. If you find this approach unsatisfying, luckily there are a number of really terrific guides which lay things out slowly and methodically and I will explicitly signpost some of these along the way so that you can do a “deep dive” when you feel like it. Otherwise, I'll take an accelerated approach to this introduction to data science in R. I expect that you will identify adjacent resources and perhaps even come up with your own creative approaches along the way, which incidentally is how real data science tends to work in practice.

There are a range of terrific textbooks out there which cover all these elements in greater depth and more slowly. In particular, I'd recommend that many readers will want to check out Hadley Wickham's “R For Data Science” book. I'll include marginal notes in this guide pointing to sections of that book, and a few others which unpack the basic mechanics of R in more detail.

1.6 Getting set up

Every single tool, programming language and data set we refer to in this book is free and open source. These tools have been produced by professionals and volunteers who are passionate about data science and research and want to share it with the world, and in order to do this (and following the “hacker way”) they’ve made these tools freely available. This also means that you aren’t restricted to a specific proprietary, expensive, or unavailable piece of software to do this work. I’ll make a few opinionated recommendations here based on my own preferences and experience, but it’s really up to your own style and approach. In fact, given that this is an open source textbook, you can even propose additions to this chapter explaining other tools you’ve found that you want to share with others.

There are, right now, primarily two languages that statisticians and data scientists use for this kind of programmatic data science: python and R. Each language has its merits and I won’t rehash the debates between various factions. For this book, we’ll be using the R language. This is, in part, because the R user community and libraries tend to scale a bit better for the work that I’m commanding in this book. However, it’s entirely possible that one could use python for all these exercises, and perhaps in the future we’ll have volume two of this book outlining python approaches to the same operations.

Bearing this in mind, the first step you’ll need to take is to download and install R. You can find instructions and install packages for a wide range of hardware on the The Comprehensive R Archive Network (or “CRAN”): <https://cran.rstudio.com>. Once you’ve installed R, you’ve got some choices to make about the kind of programming environment you’d like to use. You can just use a plain text editor like `textedit` to write your code and then execute your programs using the R software you’ve just installed. However, most users, myself included, tend to use an integrated development environment (or “IDE”). This is usually another software package with a guided user interface and some visual elements that make it faster to write and test your code. Some IDE packages, will have built-in reference tools so you can

look up options for libraries you use in your code, they will allow you to visualise the results of your code execution, and perhaps most important of all, will enable you to execute your programs line by line so you can spot errors more quickly (we call this “debugging”). The two most popular IDE platforms for R coding at the time of writing this textbook are RStudio and Visual Studio. You should download and try out both and stick with your favourite, as the differences are largely aesthetic. I use a combination of RStudio and an enhanced plain text editor Sublime Text for my coding.

Once you have R and your pick of an IDE, you are ready to go! Proceed to the next chapter and we’ll dive right in and get started!

2 The 2021 UK Census

2.1 Your first project: the UK Census

Let's start by importing some data into R. Because R is what is called an object-oriented programming language, we'll always take our information and give it a home inside a named object. There are many different kinds of objects, which you can specify, but usually R will assign a type that seems to fit best.

In the example below, we're going to read in data from a comma separated value file ("csv") which has rows of information on separate lines in a text file with each column separated by a comma. This is one of the standard plain text file formats. R has a function you can use to import this efficiently called "read.csv". Each line of code in R usually starts with the object, and then follows with instructions on what we're going to put inside it, where that comes from, and how to format it:

```
setwd("/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion")
library(here)  |> suppressPackageStartupMessages()
library(tidyverse)  |> suppressPackageStartupMessages()
here::i_am("chapter_1.qmd")
```

here() starts at /Users/kidwellj/gits/hacking_religion_textbook/hacking_religion

```
# Set up local workspace:
if (dir.exists("data") == FALSE) {
  dir.create("data")
}
if (dir.exists("figures") == FALSE) {
  dir.create("figures")
}
```

If you'd like to explore this all in a bit more depth, you can find a very helpful summary in R for Data Science, chapter 8, "[data import](#)".

```

if (dir.exists("derivedData") == FALSE) {
  dir.create("derivedData")
}

uk_census_2021_religion <- read.csv(here("example_data", "census2021-ts030-rgn.csv"))

```

2.2 Examining data:

What's in the table? You can take a quick look at either the top of the data frame, or the bottom using one of the following commands:

```
head(uk_census_2021_religion)
```

	geography	total	no_religion	christian	buddhist	hindu	jewish
1	North East	2647012	1058122	1343948	7026	10924	4389
2	North West	7417397	2419624	3895779	23028	49749	33285
3	Yorkshire and The Humber	5480774	2161185	2461519	15803	29243	9355
4	East Midlands	4880054	1950354	2214151	14521	120345	4313
5	West Midlands	5950756	1955003	2770559	18804	88116	4394
6	East	6335072	2544509	2955071	26814	86631	42012
	muslim	sikh	other	no_response			
1	72102	7206	9950	133345			
2	563105	11862	28103	392862			
3	442533	24034	23618	313484			
4	210766	53950	24813	286841			
5	569963	172398	31805	339714			
6	234744	24284	36380	384627			

This is actually a fairly ugly table, so I'll use an R tool called kable to give you prettier tables in the future, like this:

```
knitr::kable(head(uk_census_2021_religion))
```

	geography	total	no_religion	christian	landhist	histjewish	muslim	sikh	other	no_response
North	North	26470	1025812	234394	7826	10924	3897210	7206	9950133345	
East	East									
North	North	74173	2471962	28957	230284974	33285	6310	1862	2810392862	
West	West									
Yorkshire	Yorkshire	54807	216118	2461519	8032924	3355	4425	22103	2361813484	
and The	and The									
Humber	Humber									
East	East	48800	15950352	2141514	52112034	31321076	63950	24812	86841	
Mid-	Mid-									
lands	lands									
West	West	59507	1955002	7705518	80488116	39456996	372398	80539714		
Mid-	Mid-									
lands	lands									
East	East	63350	2724450295502	681486634	201234724284	638084627				

You can see how I've nested the previous command inside the `kable` command. For reference, in some cases when you're working with really complex scripts with many different libraries and functions, they may end up with functions that have the same name. You can specify the library where the function is meant to come from by preceding it with `::` as we've done `knitr::` above. The same kind of output can be gotten using `tail`:

```
knitr::kable(tail(uk_census_2021_religion))
```

	geography	total	no_religion	christian	landhist	histjewish	muslim	sikh	other	no_response
5	West	59507	1955002	7705518	80488116	39456996	372398	80539714		
	Mid-									
	lands									
6	East	63350	2724450295502	681486634	201234724284	638084627				
7	London	87997	2380403577687	425453034546	318754454875015662					
8	South	92780	158309431335943315474	8682090674348409866279						
	East									
9	South	57012	261336263582257927746	3878015274653688467732						
	West									
10	Wales	31074	1944639835477	80751224204466947404815926	95041					

2.3 Parsing and Exploring your data

The first thing you’re going to want to do is to take a smaller subset of a large data set, either by filtering out certain columns or rows. Now let’s say we want to just work with the data from the West Midlands, and we’d like to omit some of the columns. We can choose a specific range of columns using `select`, like this:

You can use the `filter` command to do this. To give an example, `filter` can pick a single row in the following way:

```
uk_census_2021_religion_wmids <- uk_census_2021_religion %>% filter(geography=="West Midland")
```

Now we’ll use `select` in a different way to narrow our data to specific columns that are needed (no totals!).

In keeping with my goal to demonstrate data science through examples, we’re going to move on to producing some snappy looking charts for this data.

Some readers will want to pause here and check out Hadley Wickham’s “R For Data Science” book, in the section, “[Data visualisation](#)” to get a fuller explanation of how to explore your data.

2.4 Making your first data visulation: the humble bar chart

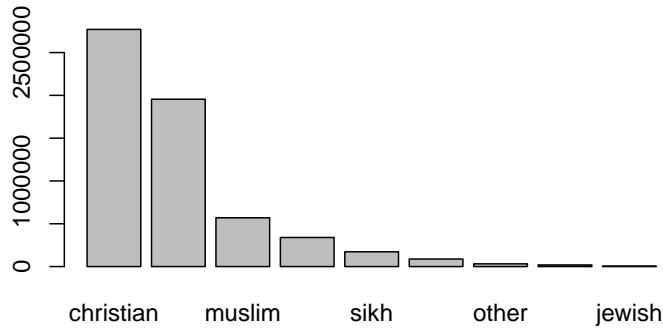
We’ve got a nice lean set of data, so now it’s time to visualise this. We’ll start by making a pie chart:

```
uk_census_2021_religion_wmids <- uk_census_2021_religion_wmids %>% select(no_religion:no_respondents)
uk_census_2021_religion_wmids <- gather(uk_census_2021_religion_wmids)
```

There are two basic ways to do visualisations in R. You can work with basic functions in R, often called “base R” or you can work with an alternative library called `ggplot`:

2.4.1 Base R

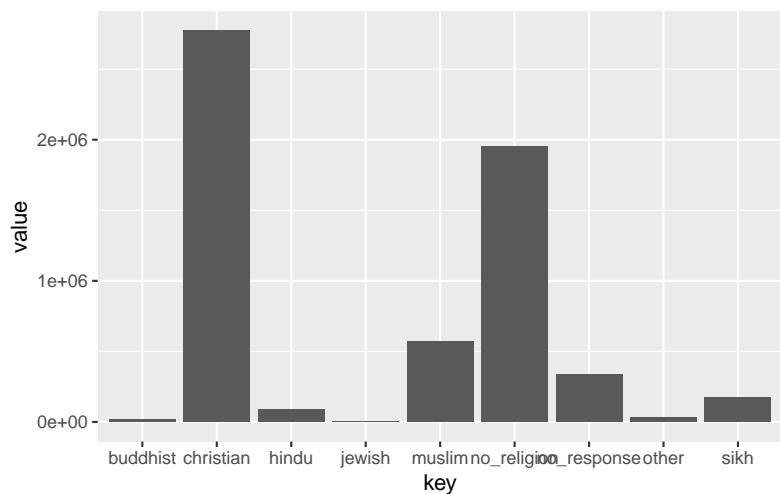
```
df <- uk_census_2021_religion_wmids[order(uk_census_2021_religion_wmids$value, decreasing = T)  
barplot(height=df$value, names=df$key)
```



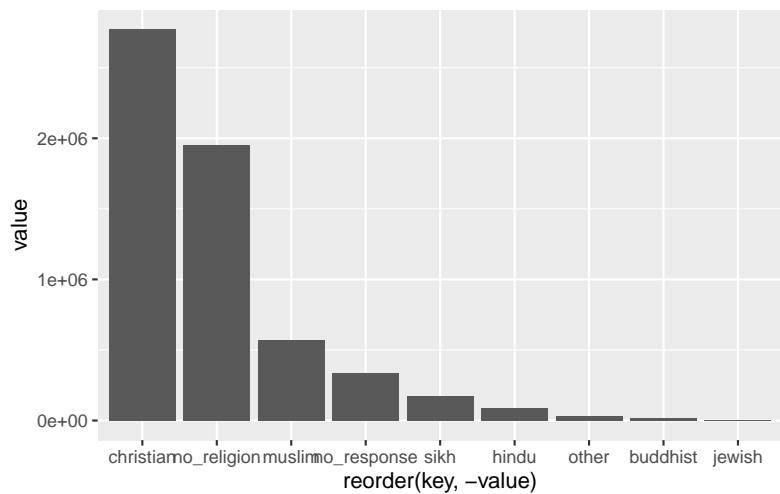
2.4.2 GGPlot

```
ggplot(uk_census_2021_religion_wmids, aes(x = key, y = value)) + ①  
  geom_bar(stat = "identity")
```

- ② We'll re-order the column by size.



```
ggplot(uk_census_2021_religion_wmids, aes(x= reorder(key,-value),value)) + geom_bar(stat = "identity")
```



Let's assume we're working with a data set that doesn't include a "totals" column and that we might want to get sums for each column. This is pretty easy to do in R:

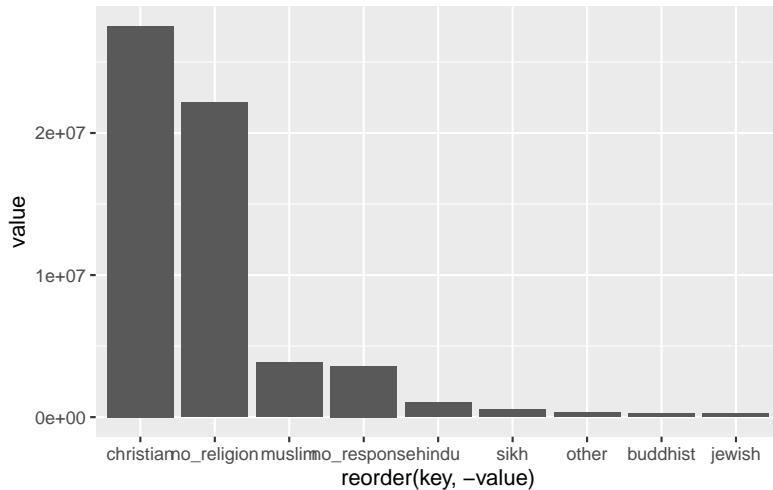
```
uk_census_2021_religion_totals <- uk_census_2021_religion %>% select(no_religion:no_response)
uk_census_2021_religion_totals <- uk_census_2021_religion_totals %>%
```

```

    summarise(across(everything(), ~ sum(., na.rm = TRUE))) ②
uk_census_2021_religion_totals <- gather(uk_census_2021_religion_totals) ③
ggplot(uk_census_2021_religion_totals, aes(x= reorder(key,-value),value)) + geom_bar(stat = "identity")

```

- ① First, remove the column with region names and the totals for the regions as we want just integer data.
- ② Second calculate the totals. In this example we use the tidyverse library `dplyr()`, but you can also do this using base R with `colsums()` like this:
`uk_census_2021_religion_totals <- colSums(uk_census_2021_religion_totals, na.rm = TRUE)`. The downside with base R is that you'll also need to convert the result into a dataframe for `ggplot` like this: `uk_census_2021_religion_totals <- as.data.frame(uk_census_2021_religion_totals)`
- ③ In order to visualise this data using `ggplot`, we need to shift this data from wide to long format. This is a quick job using `gather()`
- ④ Now plot it out and have a look!



You might have noticed that these two dataframes give us somewhat different results. But with data science, it's much more interesting to compare these two side-by-side in a visualisation. We can join these two dataframes and plot the bars side by side using `bind()` - which can be done by columns with `cbind()` and rows using `rbind()`:

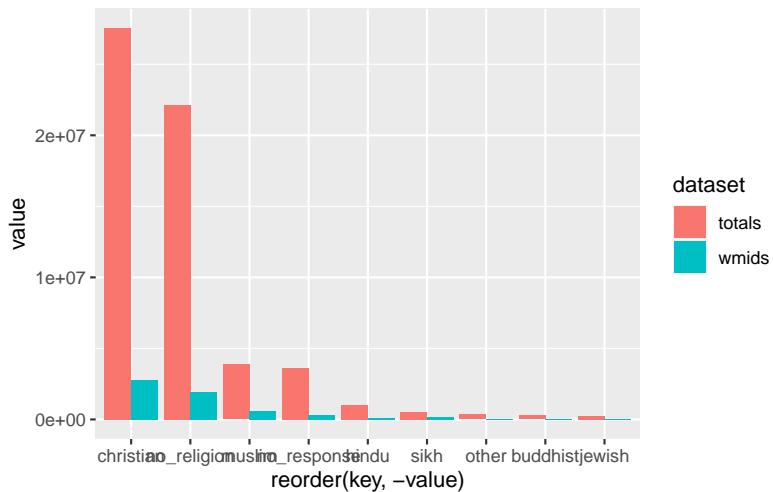
```
uk_census_2021_religion_merged <- rbind(uk_census_2021_religion_totals, uk_census_2021_relig
```

Do you notice there's going to be a problem here? How can we tell one set from the other? We need to add in something identifiable first! This isn't too hard to do as we can simply create a new column for each with identifiable information before we bind them:

```
uk_census_2021_religion_totals$dataset <- c("totals")
uk_census_2021_religion_wmids$dataset <- c("wmids")
uk_census_2021_religion_merged <- rbind(uk_census_2021_religion_totals, uk_census_2021_relig
```

Now we're ready to plot out our data as a grouped barplot:

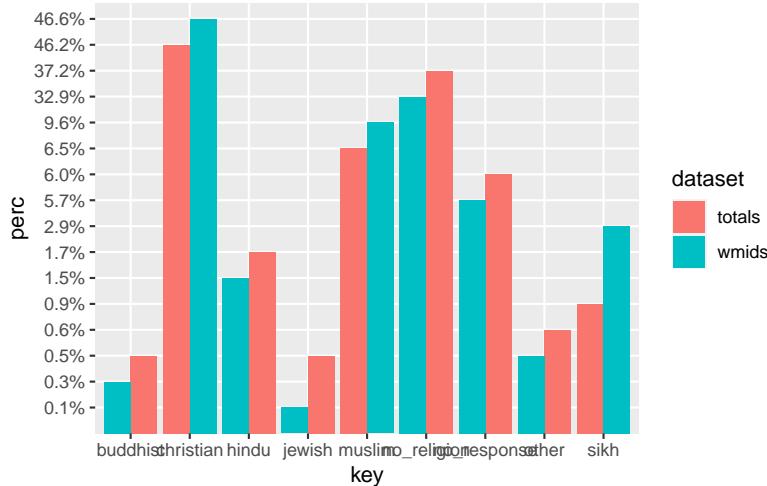
```
ggplot(uk_census_2021_religion_merged, aes(fill=dataset, x= reorder(key,-value), value)) + g
```



If you're looking closely, you will notice that I've added two elements to our previous ggplot. I've asked ggplot to fill in the columns with reference to the `dataset` column we've just created. Then I've also asked ggplot to alter the `position="dodge"` which places bars side by side rather than stacked on top of one another. You can give it a try without this instruction to see how this works. We will use stacked bars in a later chapter, so remember this feature.

If you inspect our chart, you can see that we're getting closer, but it's not really that helpful to compare the totals. What we need to do is get percentages that can be compared side by side. This is easy to do using another `dplyr` feature `mutate`:

```
uk_census_2021_religion_totals <- uk_census_2021_religion_totals %>%
  dplyr::mutate(perc = scales::percent(value / sum(value)), .by = key, trim = FALSE) ③
uk_census_2021_religion_wmids <- uk_census_2021_religion_wmids %>%
  dplyr::mutate(perc = scales::percent(value / sum(value)), .by = key, trim = FALSE) ④
uk_census_2021_religion_merged <- rbind(uk_census_2021_religion_totals, uk_census_2021_religion_wmids)
ggplot(uk_census_2021_religion_merged, aes(fill=dataset, key=key, perc)) + geom_bar(position=
```



Now you can see a very rough comparison, which sets bars from the W Midlands data and overall data side by side for each category. The same principles that we've used here can be applied to draw in more data. You could, for example, compare census data from different years, e.g. 2001 2011 and 2021. Our use of `dplyr::mutate` above can be repeated to add an infinite number of further series' which can be plotted in bar groups.

We'll draw this data into comparison with later sets in the next chapter. But the one glaring issue which remains for our chart is that it's lacking in really any aesthetic refinements. This is where `ggplot` really shines as a tool as you can add all sorts of things.

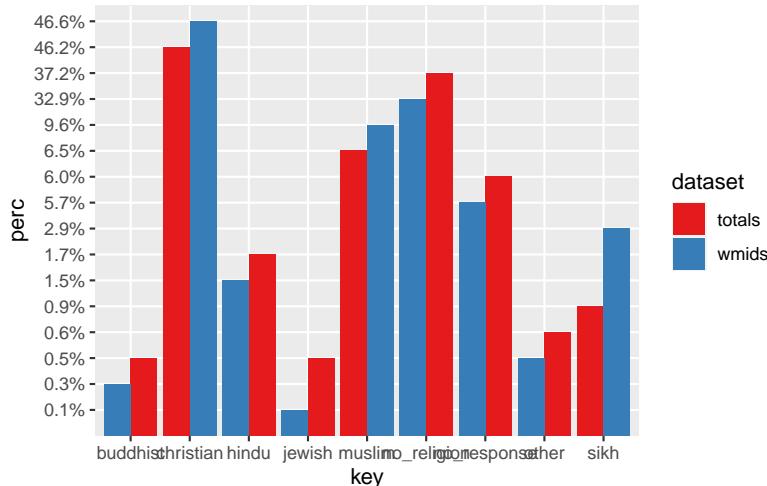
It's worth noting that an alternative

```
approach is to leave the numbers
as they are, and simply label them, like
dplyr::mutate(perc = scales::percent(value / sum(value)), .by = key, trim = FALSE) ③
dplyr::mutate(perc = scales::percent(value / sum(value)), .by = key, trim = FALSE) ④
rbind(uk_census_2021_religion_totals, uk_census_2021_religion_wmids)
geom_bar(position=
```

downside of this approach is that it won't transfer to tables if you make them.

These are basically just added to our `ggplot` code. So, for example, let's say we want to improve the colours used for our bars. You can specify the formatting for the fill on the `scale` using `scale_fill_brewer`. This uses a particular tool (and a personal favourite of mine) called `colorbrewer`. Part of my appreciation of this tool is that you can pick colours which are not just visually pleasing, and produce useful contrast / complementary schemes, but you can also work proactively to accommodate colourblindness. Working with colour schemes which can be divergent in a visually obvious way will be even more important when we work on geospatial data and maps in a later chapter.

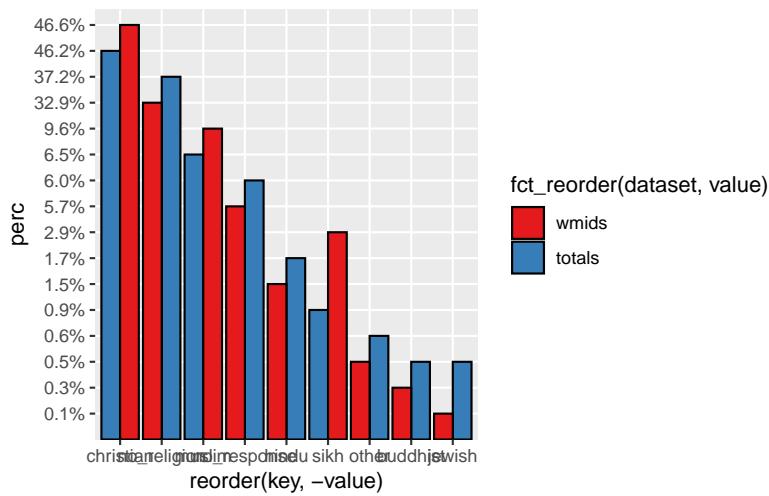
```
ggplot(uk_census_2021_religion_merged, aes(fill=dataset, x=key, y=perc)) + geom_bar(position
```



We might also want to add a border to our bars to make them more visually striking (notice the addition of `color` to the `geom_bar` below. I've also added `reorder()` to the `x` value to sort descending from the largest to smallest.

```
uk_census_2021_religion_merged$dataset <- factor(uk_census_2021_religion_merged$dataset, levels=c("wmids", "totals"))
ggplot(uk_census_2021_religion_merged, aes(fill=fct_reorder(dataset, value), x=reorder(key, -
```

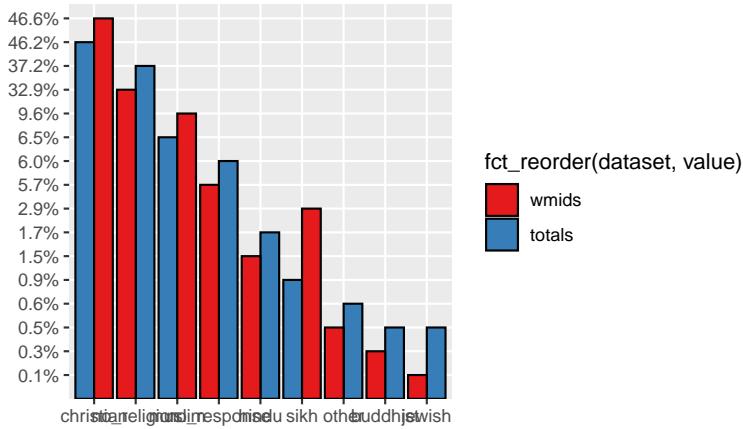
You can find more information about reordering ggplots on the [R](#)



We can fine tune a few other visual features here as well, like adding a title with `ggtitle` and a them with some prettier fonts with `theme_ipsum()` (which requires the `hrbrthemes()` library). We can also remove the x and y axis labels (not the data labels, which are rather important).

```
ggplot(uk_census_2021_religion_merged, aes(fill=fct_reorder(dataset, value), x=reorder(key, -value)))
```

Religious Affiliation in the UK: 2021



2.5 Is your chart accurate? Telling the truth in data science

There is some technical work yet to be done fine-tuning the visualisation of our chart here. But I'd like to pause for a moment and consider an ethical question. Is the title of this chart truthful and accurate? On one hand, it is a straight-forward reference to the nature of the question asked on the 2021 census survey instrument. However, as you will see in the next chapter, large data sets from the same year which asked a fairly similar question yield different results. Part of this could be attributed to the amount of non-response to this specific question which, in the 2021 census is between 5-6% across many demographics. It's possible (though perhaps unlikely) that all those non-responses were Sikh respondents who felt uncomfortable identifying themselves on such a survey. If even half of the non-responses were of this nature, this would dramatically shift the results especially in comparison to other minority groups. So there is some work for us to do here in representing non-response as a category on the census.

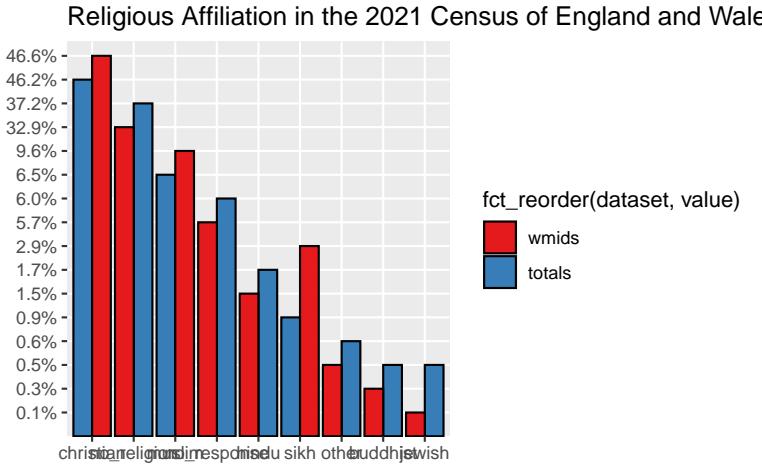
It's equally possible that someone might feel uncertain when answering, but nonetheless land on a particular decision marking "Christian" when they wondered if they should instead tick "no religion. Some surveys attempt to capture uncertainty in this way, asking respondents to mark how confident they are about their answers, but the census hasn't capture this so we simply don't know. If a large portion of respondents in the "Christian" category were hovering between this and another response, again, they might shift their answers when responding on a different day, perhaps having just had a conversation with a friend which shifted their thinking. Even the inertia of survey design can have an effect on this, so responding to other questions in a particular way, thinking about ethnic identity, for example, can prime a person to think about their religious identity in a different or more focussed way, altering their response to the question. For this reason, some survey instruments randomise the order of questions. This hasn't been done on the census (which would have been quite hard work given that most of the instruments were printed hard copies!),

so again, we can't really be sure if those answers given are stable.

Finally, researchers have also found that when people are asked to mark their religious affiliation, sometimes they can prefer to mark more than one answer. A person might consider themselves to be "Muslim" but also "Spiritual but not religious" preferring the combination of those identities. It is also the case that respondents can identify with more unexpected hybrid religious identities, such as "Christian" and "Hindu". The census only allows respondents to tick a single box for the religion category. It is worth noting that, in contrast, the responses for ethnicity allow for combinations. Given that this is the case, it's impossible to know which way a person went at the fork in the road as they were forced to choose just one half of this kind of hybrid identity. Finally, it is interesting to wonder exactly what it means for a person when they tick a box like this. Is it because they attend synagogue on a weekly basis? Some persons would consider weekly attendance at worship a prerequisite for membership in a group, but others would not. Indeed we can infer from surveys and research which aims to track rates of participation in weekly worship that many people who tick boxes for particular religious identities on the census have never attended a worship service at all.

What does this mean for our results? Are they completely unreliable and invalid? I don't think this is the case or that taking a clear-eyed look at the force and stability of our underlying data should be cause for despair. Instead, the most appropriate response is humility. Someone has made a statement which is recorded in the census, of this we can be sure. They felt it to be an accurate response on some level based on the information they had at the time. And with regard to the census, it is a massive, almost completely population level, sample so there is additional validity there. The easiest way to represent all this reality in the form of speaking truthfully about our data is to acknowledge that however valid it may seem, it is nonetheless a snapshot. For this reason, I would always advise that the best title for a chart is one which specifies the data set. We should also probably do something different with those non-responses:

```
ggplot(uk_census_2021_religion_merged, aes(fill=fct_reorder(dataset, value), x=reorder(key, -
```



Change orientation of X axis labels + theme(axis.text.x = element_text(angle = 90, vjust = 0.5, hjust=1))

Relabel fields Simplify y-axis labels Add percentage text to bars
(or maybe save for next chapter?)

2.6 Making our script reproducible

Let's take a moment to review our hacker code. I've just spent some time addressing how we can be truthful in our data science work. We haven't done much yet to talk about reproducibility.

2.7 Multifactor Visualisation

One element of R data analysis that can get really interesting is working with multiple variables. Above we've looked at the breakdown of religious affiliation across the whole of England and Wales (Scotland operates an independent census), and by placing this data alongside a specific region, we've already made

a basic entry into working with multiple variables but this can get much more interesting. Adding an additional quantitative variable (also known as bivariate data) into the mix, however can also generate a lot more information and we have to think about visualising it in different ways which can still communicate with visual clarity in spite of the additional visual noise which is inevitable with enhanced complexity. Let's have a look at the way that religion in England and Wales breaks down by ethnicity.

💡 What is Nomis?

For the UK, census data is made available for programmatic research like this via an organisation called NOMIS. Luckily for us, there is an R library you can use to access nomis directly which greatly simplifies the process of pulling data down from the platform. It's worth noting that if you're not in the UK, there are similar options for other countries. Nearly every R textbook I've ever seen works with USA census data, so you'll find plenty of documentation available on the tools you can use for US Census data. Similarly for the EU, Canada, Australia etc.

If you want to draw some data from the nomis platform yourself in R, have a look at the [companion cookbook repository](#).

```
# Get table of Census 2011 religion data from nomis
# Note: for reproducible code used to generate the dataset used in the book, see the cookbook
z <- readRDS(file = (here("example_data", "z.rds")))

# Filter down to simplified dataset with England / Wales and percentages without totals
uk_census_2011_religion <- filter(z, GEOGRAPHY_NAME=="England and Wales" & RURAL_URBAN_NAME=="Urban")
# Drop unnecessary columns
uk_census_2011_religion <- select(uk_census_2011_religion, C_RELPUK11_NAME, OBS_VALUE)
# Plot results
plot1 <- ggplot(uk_census_2011_religion, aes(x = C_RELPUK11_NAME, y = OBS_VALUE)) + geom_bar()
# ggsave(filename = "plot.png", plot = plot1)

# grab daata from nomis for 2001 census religion / ethnicity
```

```

z0 <- readRDS(file = (here("example_data", "z0.rds")))

# select relevant columns
uk_census_2001_religion_ethnicity <- select(z0, GEOGRAPHY_NAME, C_RELPUK11_NAME, C_ETHHUK11_NAME)
# Filter down to simplified dataset with England / Wales and percentages without totals
uk_census_2001_religion_ethnicity <- filter(uk_census_2001_religion_ethnicity, GEOGRAPHY_NAME %in% c("England", "Wales"))
# Simplify data to only include general totals and omit subcategories
uk_census_2001_religion_ethnicity <- uk_census_2001_religion_ethnicity %>% filter(grepl('Total', C_RELPUK11_NAME))

# grab data from nomis for 2011 census religion / ethnicity table
z1 <- readRDS(file = (here("example_data", "z1.rds")))

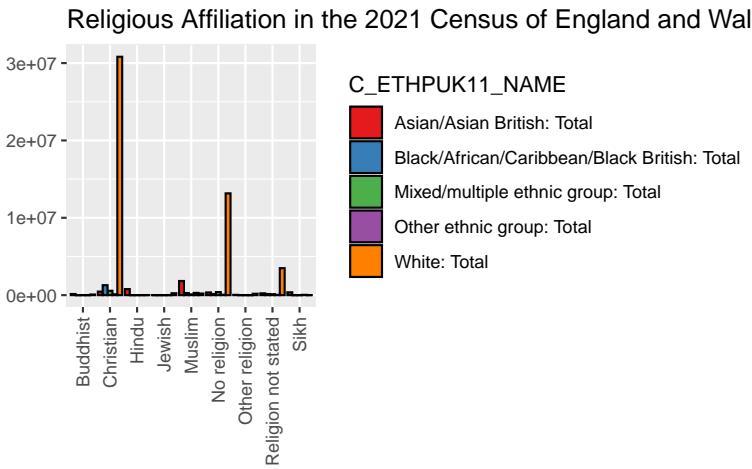
# select relevant columns
uk_census_2011_religion_ethnicity <- select(z1, GEOGRAPHY_NAME, C_RELPUK11_NAME, C_ETHPUK11_NAME)
# Filter down to simplified dataset with England / Wales and percentages without totals
uk_census_2011_religion_ethnicity <- filter(uk_census_2011_religion_ethnicity, GEOGRAPHY_NAME %in% c("England", "Wales"))
# Simplify data to only include general totals and omit subcategories
uk_census_2011_religion_ethnicity <- uk_census_2011_religion_ethnicity %>% filter(grepl('Total', C_RELPUK11_NAME))

# grab data from nomis for 2021 census religion / ethnicity table
z2 <- readRDS(file = (here("example_data", "z2.rds")))

# select relevant columns
uk_census_2021_religion_ethnicity <- select(z2, GEOGRAPHY_NAME, C2021_RELIGION_10_NAME, C2021_ETHN_8_NAME)
# Filter down to simplified dataset with England / Wales and percentages without totals
uk_census_2021_religion_ethnicity <- filter(uk_census_2021_religion_ethnicity, GEOGRAPHY_NAME %in% c("England", "Wales"))
# 2021 census includes white sub-groups so we need to omit those so we just have totals:
uk_census_2021_religion_ethnicity <- filter(uk_census_2021_religion_ethnicity, C2021_ETH_8_NAME == "Total")

ggplot(uk_census_2011_religion_ethnicity, aes(fill=C_ETHPUK11_NAME, x=C_RELPUK11_NAME, y=OBS))

```

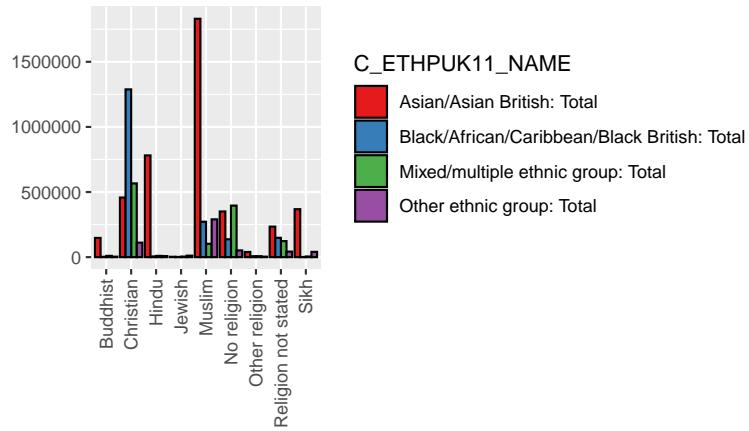


The trouble with using grouped bars here, as you can see, is that there are quite sharp disparities which make it hard to compare in meaningful ways. We could use logarithmic rather than linear scaling as an option, but this is hard for many general public audiences to appreciate without guidance. One alternative quick fix is to extract data from “white” respondents which can then be placed in a separate chart with a different scale.

```
# Filter down to simplified dataset with England / Wales and percentages without totals
uk_census_2011_religion_ethnicity_white <- filter(uk_census_2011_religion_ethnicity, C_ETHPUK11_NAME == "White: Total")
uk_census_2011_religion_ethnicity_nonwhite <- filter(uk_census_2011_religion_ethnicity, C_ETHPUK11_NAME != "White: Total")

ggplot(uk_census_2011_religion_ethnicity_nonwhite, aes(fill=C_ETHPUK11_NAME, x=C_RELPUK11_NAME))
```

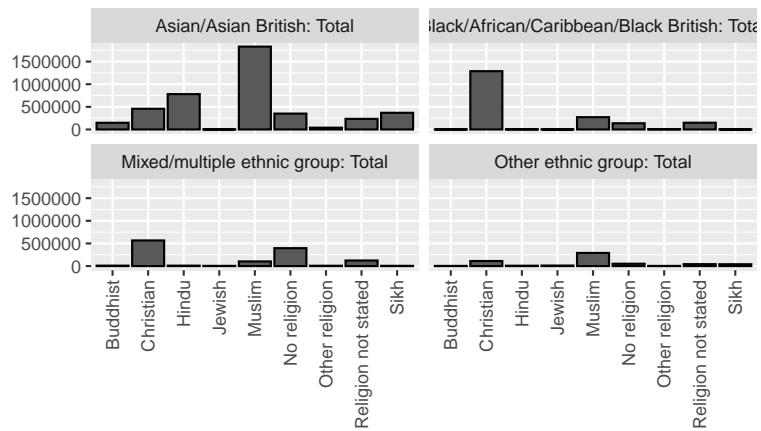
Religious Affiliation in the 2021 Census of England and Wales



This still doesn't quite render with as much visual clarity and communication as I'd like. For a better look, we can use a technique in R called "faceting" to create a series of small charts which can be viewed alongside one another.

```
ggplot(uk_census_2011_religion_ethnicity_nonwhite, aes(x=C_RELPUK11_NAME, y=OBS_VALUE)) + geom_bar()
```

Religious Affiliation in the 2011 Census of England and Wales



For our finale chart, I'd like to take the faceted chart we've just done, and add in totals for the previous two census years (2001

and 2011) so we can see how trends are changing in terms of religious affiliation within ethnic self-identification categories. We'll draw on some techniques we're already developed above using `rbind()` to connect up each of these charts (after we've added a column identifying each chart by the census year). We will also need to use one new technique to change the wording of ethnic categories as this isn't consistent from one census to the next and `ggplot` will struggle to chart things if the terms being used are exactly the same. We'll use `mutate()` again to accomplish this with some slightly different code.

```
# First add column to each dataframe so we don't lose track of the census it comes from:
uk_census_2001_religion_ethnicity$dataset <- c("2001")
uk_census_2011_religion_ethnicity$dataset <- c("2011")
uk_census_2021_religion_ethnicity$dataset <- c("2021")

# Let's tidy the names of each column:

names(uk_census_2001_religion_ethnicity) <- c("Geography", "Religion", "Ethnicity", "Value",
names(uk_census_2011_religion_ethnicity) <- c("Geography", "Religion", "Ethnicity", "Value",
names(uk_census_2021_religion_ethnicity) <- c("Geography", "Religion", "Ethnicity", "Value",

# Next we need to change the terms using mutate()
uk_census_2001_religion_ethnicity <- uk_census_2001_religion_ethnicity %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^White: Total$", replacement = "White")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^Mixed: Total$", replacement = "Mixed")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^Asian: Total$", replacement = "Asian")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^Black or Black British: Total$", replacement = "Black")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^Chinese or Other ethnic group: Total$", replacement = "Other"))

uk_census_2011_religion_ethnicity <- uk_census_2011_religion_ethnicity %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^White: Total$", replacement = "White")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
    pattern = "^Mixed/multiple ethnic group: Total$", replacement = "Mixed")) %>%
```

```

mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Asian/Asian British: Total$", replacement = "Asian")) %>%
mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Black/African/Caribbean/Black British: Total$", replacement = "Black"))
mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Other ethnic group: Total$", replacement = "Other"))

uk_census_2021_religion_ethnicity <- uk_census_2021_religion_ethnicity %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3White: Total$", replacement = "White")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Mixed or Multiple ethnic groups$", replacement = "Mixed")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Asian, Asian British or Asian Welsh$", replacement = "Asian")) %>%
  mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Black, Black British, Black Welsh, Caribbean or African$", replacement = "Black"))
  mutate(Ethnicity = str_replace_all(Ethnicity,
                                   pattern = "^\u00c2\u00a3Other ethnic group$", replacement = "Other"))

# Now let's merge the tables:

uk_census_merged_religion_ethnicity <- rbind(uk_census_2021_religion_ethnicity, uk_census_2011_religion_ethnicity)

uk_census_merged_religion_ethnicity <- rbind(uk_census_merged_religion_ethnicity, uk_census_2011_religion_ethnicity)

# As above, we'll split out non-white and white:

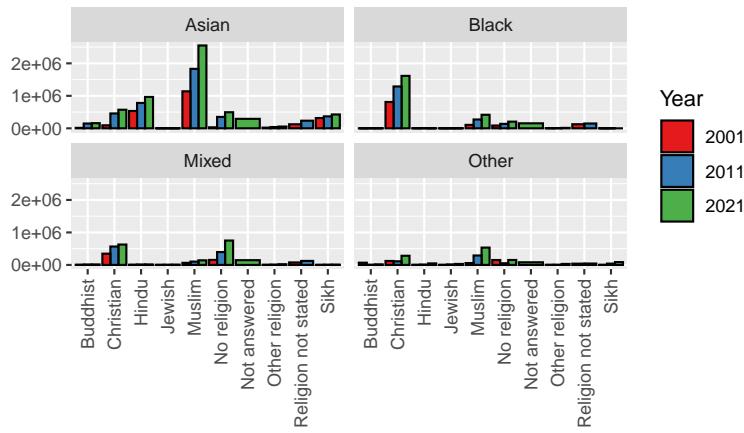
uk_census_merged_religion_ethnicity_nonwhite <- filter(uk_census_merged_religion_ethnicity,
                                                       Ethnicity != "White" & Ethnicity != "Black")

# Time to plot!

ggplot(uk_census_merged_religion_ethnicity_nonwhite, aes(fill=Year, x=Religion, y=Value)) +

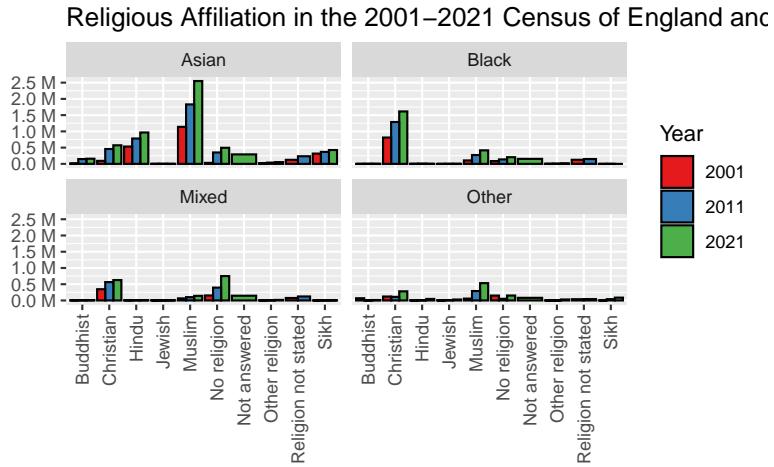
```

Religious Affiliation in the 2001–2021 Census of England and Wales by Ethnic Group



There are a few formatting issues which remain. Our y-axis number labels are in scientific format which isn't really very easy to read. You can use the very powerful and flexible `scales()` library to bring in some more readable formatting of numbers in a variety of places in R including in ggplot visualizations.

```
library(scales) |> suppressPackageStartupMessages()
ggplot(uk_census_merged_religion_ethnicity_nonwhite, aes(fill=Year, x=Religion, y=Value)) +
```

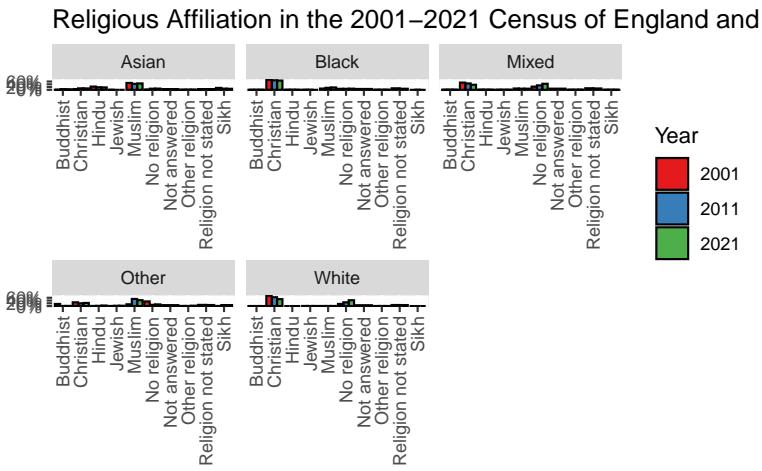


```
# https://ggplot2-book.org/scales-position#sec-position-continuous-breaks
```

This chart shows an increase in almost every category, though it's a bit hard to read in some cases. However, this information is based on the increase in raw numbers. It's possible that numbers may be going up, but in some cases the percentage share for a particular category has actually gone down. Let's transform and visualise our data as percentages to see what kind of trends we can actually isolate:

```
uk_census_merged_religion_ethnicity <- uk_census_merged_religion_ethnicity %>%
  group_by(Ethnicity, Year) %>%
  dplyr::mutate(Percent = Value/sum(Value))

ggplot(uk_census_merged_religion_ethnicity, aes(fill=Year, x=Religion, y=Percent)) + geom_bar
```



Now you can see why this shift is important - the visualisation tells a completely different story in some cases across the two different charts. In the first, working off raw numbers we see a net increase in Christianity across all categories. But if we take into account the fact that the overall share of population is growing for each of these groups, their actual composition is changing in a different direction. The proportion of each group is declining across the three census periods (albeit with

an exception for the “Other” category from 2011 to 2021).

To highlight a few features of this final plot, I’ve used a specific feature within `facet_wrap scales = "free_x"` to let each of the individual facets adjust the total range on the x-axis. Since we’re looking at trends here and not absolute values, having correspondence across scales isn’t important and this makes for something a bit more visually tidy. I’ve also shifted the code for `scale_y_continuous` to render values as percentages (rather than millions).

In case you want to print this plot out and hang it on your wall, you can use the `ggsave` tool to render the chart as an image file:

```
plot1 <- ggplot(uk_census_merged_religion_ethnicity, aes(fill=Year, x=Religion, y=Percent))

ggsave("chart.png", plot=plot1, width = 8, height = 10, units=c("in"))
```

References

3 Survey Data: Spotlight Project

In the last chapter we explored some high level data about religion in the UK. This was a census sample, which usually refers to an attempt to get as comprehensive a sample as possible. But this is actually fairly unusual in practice. Depending on how complex a subject is, and how representative we want our data to be, it's much more common to use selective sampling, that is survey responses at $n=100$ or $n=1000$ at a maximum. The advantage of a census sample is that you can explore how a wide range of other factors - particularly demographics - intersect with your question. And this can be really valuable in the study of religion, particularly as you will see as we go along that responses to some questions are more strongly correlated to things like economic status or educational attainment than they are to religious affiliation. It can be hard to tell if this is the case unless you have enough of a sample to break down into a number of different kinds of subsets. But census samples are complex and expensive to gather, so they're quite rare in practice.

For this chapter, I'm going to walk you through a data set that a colleague (Charles Ogunbode) and I collected in 2021. Another problem with smaller, more selective samples is that researchers can often undersample minoritised ethnic groups. This is particularly the case with climate change research. Until the time we conducted this research, there had not been a single study investigating the specific experiences of people of colour in relation to climate change in the UK. Past researchers had been content to work with large samples, and assumed that if they had done 1000 surveys and 50 of these were completed by people of colour, they could "tick" the box. But 5% is actually well below levels of representation in the UK generally, and even more sharply the case for specific communities. And

if we bear in mind that non-white respondents are (of course!) a highly heterogenous group, we're even more behind in terms of collecting data that can improve our knowledge. Up until recently researchers just haven't been paying close enough attention to catch the significant neglect of the empirical field that this represents.

While I've framed my comments above in terms of climate change research, it is also the case that, especially in diverse societies like the USA, Canada, the UK etc., paying attention to non-majority groups and people and communities of colour automatically draws in a strongly religious sample. This is highlighted in one recent study done in the UK, the "[Black British Voices Report](#)" in which the researchers observed that "84% of respondents described themselves as religious and/or spiritual". My comments above in terms of controlling for other factors remains important here - these same researchers also note that "despite their significant importance to the lives of Black Britons, only 7% of survey respondents reported that their religion was more defining of their identity than their race".

We've decided to open up access to our data and I'm highlighting it in this book because it's a unique opportunity to explore a dataset that emphasises diversity from the start, and by extension, provides some really interesting ways to use data science techniques to explore religion in the UK.

4 Loading in some data

```
# R Setup -----  
setwd("/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion")  
library(here)  
  
here() starts at /Users/kidwellj/gits/hacking_religion_textbook  
  
library(tidyverse)  
  
-- Attaching core tidyverse packages ----- tidyverse 2.0.0 --  
v dplyr     1.1.3     v readr     2.1.4  
v forcats   1.0.0     v stringr   1.5.0  
v ggplot2   3.4.3     v tibble    3.2.1  
v lubridate  1.9.3     v tidyrr    1.3.0  
v purrr     1.0.2  
  
-- Conflicts ----- tidyverse_conflicts() --  
x dplyr::filter() masks stats::filter()  
x dplyr::lag()   masks stats::lag()  
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become  
  
library(haven) # used for importing SPSS .sav files  
here::i_am("chapter_2.qmd")  
  
here() starts at /Users/kidwellj/gits/hacking_religion_textbook/hacking_religion  
  
climate_experience_data <- read_sav(here("example_data", "climate_experience_data.sav"))
```

The first thing to note here is that we've drawn in a different type of data file, this time from an `.sav` file, usually produced by the statistics software package SPSS. This uses a different R Library (I use `haven` for this). The upside is that in some cases where you have survey data with both a code and a value like "1" is equivalent to "very much agree" this will preserve both in the R dataframe that is created. Now that you've loaded in data, you have a new R dataframe called "climate_experience_data" with a lot of columns with just under 1000 survey responses.

5 How can you ask about religion?

One of the challenges we faced when running this study is how to gather responsible data from surveys regarding religious identity. We'll dive into this in depth as we do analysis and look at some of the agreements and conflicts in terms of respondent attribution. Just to set the stage, we used the following kinds of question to ask about religion and spirituality:

1. Question 56 asks respondents simply, "What is your religion?" and then provides a range of possible answers. We included follow-up questions regarding denomination for respondents who indicated they were "Christian" or "Muslim". For respondents who ticked "Christian" we asked, "What is your denomination?" and for respondents who ticked "Muslim" we asked "Which of the following would you identify with?" and then left a range of possible options which could be ticked such as "Sunni," "Shia," "Sufi" etc.

This is one way of measuring religion, that is, to ask a person if they consider themselves formally affiliated with a particular group. This kind of question has some (serious) limitations, but we'll get to that in a moment.

We also asked respondents (Q57): "Regardless of whether you belong to a particular religion, how religious would you say you are?" and then provided a slider from 0 (not religious at all) to 10 (very religious).

We included some classic indicators about how often respondents go to worship (Q58): Apart from weddings, funerals and other special occasions, how often do you attend religious services? and (Q59): "Q59 Apart from when you are at religious services, how often do you pray?"

- More than once a week (1)
- Once a week (2)
- At least once a month (3)
- Only on special holy days (4)
- Never (5)

Each of these measures a particular kind of dimension, and it is interesting to note that sometimes there are stronger correlations between how often a person attends worship services (weekly versus once a year) and a particular view, than there is between their affiliation (if they are Christian or Pagan). We'll do some exploratory work shortly to see how this is the case in our sample. We also included a series of questions about spirituality in Q52 and used a nature relatedness scale Q51.

You'll find that many surveys will only use one of these forms of question and ignore the rest. I think this is a really bad idea as religious belonging, identity, and spirituality are far too complex to work off a single form of response. We can also test out how these different attributions relate to other demographic features, like interest in politics, economic attainment, etc.

So who's religious?

As I've already hinted in the previous chapter, measuring religiosity is complicated. I suspect some readers may be wondering something like, "what's the right question to ask?" here. Do we get the most accurate representation by asking people to self-report their religious affiliation? Or is it more accurate to ask individuals to report on how religious they are? Is it, perhaps, better to assume that the indirect query about practice, e.g. how frequently one attends services at a place of worship may be the most reliable proxy?

Highlight challenges of various approaches pointing to literature.

Let's dive into the data and see how this all works out. We'll start with the question 56 data, around religious affiliation:

```
religious_affiliation <- as_tibble(as_factor(climate_experience_data$Q56))      ①
names(religious_affiliation) <- c("response")  ②
religious_affiliation <- filter(religious_affiliation, !is.na(response)) ③
```

There are few things we need to do here to get the data into initial proper shape. This might be called “cleaning” the data:

1. Because we imported this data from an SPSS .sav file format using the R `haven()` library, we need to start by adapting the data into a format that our visualization engine `ggplot` can handle (a dataframe).
2. Next we'll rename the columns so these names are a bit more useful.
3. We need to omit non-responses so these don't mess with the counting (these are NA in R)

If we pause at this point to view the data, you'll see it's basically just a long list of survey responses. What we need is a count of each unique response (or `factor`). This will take a few more steps:

```
religious_affiliation_sums <- religious_affiliation %>%
  dplyr::count(response, sort = TRUE) %>%      ①
  dplyr::mutate(response =forcats::fct_rev(forcats::fct_inorder(response)))    ②
religious_affiliation_sums <- religious_affiliation_sums %>%
  dplyr::mutate(perc = scales::percent(n / sum(n), accuracy = .1, trim = FALSE)) ③
```

- ① First we generate new a dataframe with sums per category and
- ② ...sort in descending order
- ③ Then we add new column with percentages based on the sums you've just generated

That should give us a tidy table of results, which you can see if you view the contents of our new `religious_affiliation_sums` dataframe:

```
head(religious_affiliation_sums)

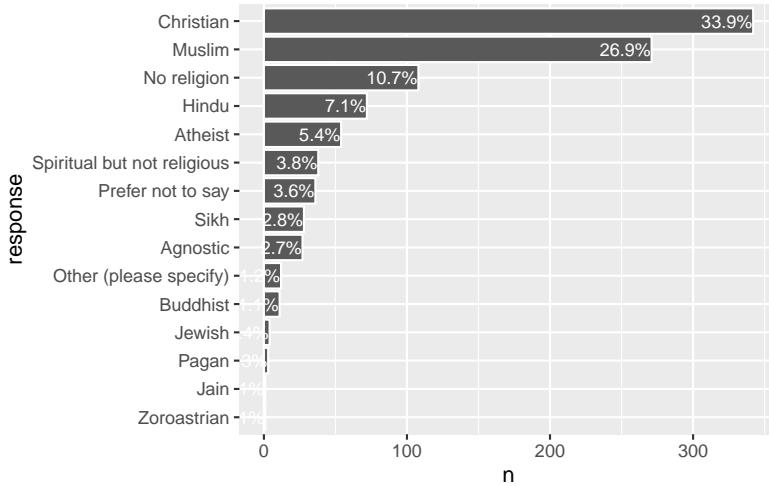
# A tibble: 6 x 3
```

```

response           n  perc
<fct>            <int> <chr>
1 Christian        342 "33.9%"
2 Muslim           271 "26.9%"
3 No religion     108 "10.7%"
4 Hindu            72  " 7.1%"
5 Atheist          54  " 5.4%"
6 Spiritual but not religious 38  " 3.8%"

# make plot
ggplot(religious_affiliation_sums, aes(x = n, y = response)) +
  geom_col(colour = "white") +
  ## add percentage labels
  geom_text(aes(label = perc),
            ## make labels left-aligned and white
            hjust = 1, nudge_x = -.5, colour = "white", size=3)

```



I've added one feature to our chart that wasn't in the bar charts in chapter 1, text labels with the actual value on each bar.

You may be thinking about the plots we've just finished in chapter 1 and wondering how they compare. Let's use the same facet approach that we've just used to render this data in a subsetted way.

```

# First we need to add in data on ethnic self-identification from our respondents:
df <- select(climate_experience_data, Q56, Q0)
religious_affiliation_ethnicity <- as_tibble(as_factor(df))
names(religious_affiliation_ethnicity) <- c("Religion", "Ethnicity")

religious_affiliation_ethnicity_sums <- religious_affiliation_ethnicity %>%
  group_by(Ethnicity) %>%
  dplyr::count(Religion, sort = TRUE) %>%      ①
  dplyr::mutate(Religion =forcats::fct_rev(forcats::fct_inorder(Religion)))

plot1 <- ggplot(religious_affiliation_ethnicity_sums, aes(x = n, y = Religion)) +
  geom_col(colour = "white") + facet_wrap(~Ethnicity, scales="free_x")

ggsave("chart.png", plot=plot1, width = 8, height = 10, units=c("in"))

```

Use mutate to put “prefer not to say” at the bottom # Info here:

<https://r4ds.had.co.nz/factors.html#modifying-factor-levels>

6 Q56 follow-ups

```
caption <- "Christian Denomination" # TODO: copy
plot above for Q56 to add two additional plots using
climate_experience_data_namedQ56bandclimate_experience_data_namedQ56c
# Religious Affiliation b - Christian Denomination Subquestion
christian_denomination <- qualtrics_process_single_multiple_choice(climate_experience_data_namedQ56b)ch
- chart_single_result_flextable(climate_experience_data_namedQ56b,
desc(Count)) christian_denomination_table save_as_docx(christian_denomination_table,
path = "./figures/q56_religious_affiliation_xn_denomination.docx")

christian_denomination_hi <- filter(climate_experience_data_named,
Q56 == "Christian", Q57_bin == "high") christian_denomination_hi
<- qualtrics_process_single_multiple_choice(christian_denomination_hi$Q56b)
christian_denomination_hi
```

7 Religious Affiliation c - Muslim Denomination Subquestion

```
caption <- "Islamic Identity" # Should the label be different
than income since the data examined is the Affiliation? #
TODO: adjust plot to factor using numbered responses on
this question (perhaps also above) religious_affiliationc <-
qualtrics_process_single_multiple_choice(climate_experience_data_namedQ56c)religious_affiliationc$plot <-
plot_horizontal_bar(religious_affiliationc)religious_affiliationc$plot <-
religious_affiliationc$plot + labs(caption = caption, x =
"", y = "")religious_affiliationc$plotggssave("figures/q56c_religious_affiliation.png", width =
20, height = 10, units = "cm")religious_affiliationc$table <-
chart_single_resultflextable(climate_experience_data_namedQ56c,
Count)religious_affiliationc_table save_as_docx(religious_affiliationc_table,
path = "./figures/q56_religious_affiliation_islam.docx")
```

8 Q57

9 Religiosity

```
caption <- "Respondent Religiosity" religiosity <- qualtrics_process_single_multiple_choice(as.character(climate  
-plot_horizbar(religiosity)religiosity_pplot <- religiosity_pplot+  
labs(caption = caption, x = "", y = "")religiosity_plotggssave("figures/q57_religiosity_plot.png", width =  
20, height = 10, units = "cm")religiosity_table <- chart_single_resultflextable(climate_experience_data_namedQ57_1  
desc(Variable)) religiosity_table save_as_docx(religious_affiliationc_table,  
path = "./figures/q57_religiousity.docx")
```

10 Q58

```
caption <- "Respondent Attendance of Religious Services" religious_service_attend <- qualtrics_process_single_multiple_choice(climate_experience_data_namedQ58)religious_service_attend <- plot_horizontal_bar(religious_service_attend)religious_service_attend$lot <- religious_service_attend$lot + labs(title = caption, x = "", y = "")religious_service_attend$lot$gg$save("figures/q58_religious_service_attend.png", width = 20, height = 10, units = "cm")religious_service_attend$table <- chart_single_result_flextable(climate_experience_data_namedQ58, Count) religious_service_attend_table$save_as_docx(religious_service_attend_table, path = "./figures/q58_religious_service_attend.docx")
```

11 Faceted plot working with 3x3 grid

```
df <- select(climate_experience_data, Q52_bin, Q53_bin,
Q57_bin, Q58) names(df) <- c("Q52_bin", "Q53_bin",
"Q57_bin", "response") facet_names <- c(Q52_bin = "Spirituality",
Q53_bin = "Politics L/R", Q57_bin = "Religiosity",
low="low", medium="medium", high="high") facet_labeller
<- function(variable,value){return(facet_names[value])}
dfresponse <- factor(dfresponse, ordered = TRUE, levels =
c("1", "2", "3", "4", "5")) dfresponse <- fct_recode(dfresponse,
"More than once a week" = "1", "Once a week" = "2", "At
least once a month" = "3", "Only on special holy days" =
"4", "Never" = "5") df %>% # we need to get the data
including facet info in long format, so we use pivot_longer()
pivot_longer(!response, names_to = "bin_name", values_to
= "b") %>% # add counts for plot below count(response,
bin_name, b) %>% group_by(bin_name,b) %>% mutate(perc=paste0(round(n*100/sum(n),1),"%")) %>% # run ggplot ggplot(aes(x = n, y = "", fill = response))
+ geom_col(position=position_fill(), aes(fill=response)) +
geom_text(aes(label = perc), position = position_fill(vjust=.5),
size=2) + scale_fill_brewer(palette ="Dark2", type = "qual")
+ scale_x_continuous(labels = scales::percent_format()) +
facet_grid(vars(b), vars(bin_name), labeller=as_labeller(facet_names))
+ labs(caption = caption, x = "", y ="") + guides(fill =
guide_legend(title = NULL)) ggsave("figures/q58_faceted.png",
width = 30, height = 10, units = "cm")
```

12 Q59

```
caption <- "Respondent Prayer Outside of Religious Services"  
prayer <- qualtrics_process_single_multiple_choice(climate_experience_data_namedQ59)  
prayer$lot <- plot_horizontal_bar(prayer)$lot  
prayer$lot <- -prayer$lot + labs(caption = caption, x = "", y = "")  
prayer$lot$gsave("figures/q59_prayer.png", width = 20, height = 10, units = "cm")  
prayer_table <- chart_single_result_flextable(climate_experience_data_namedQ59, Count)  
prayer_table $save_as_docx(prayer_table, path = "./figures/q59_prayer.docx")
```

13 Faceted plot working with 3x3 grid

```
df <- select(climate_experience_data, Q52_bin, Q53_bin,
Q57_bin, Q59) names(df) <- c("Q52_bin", "Q53_bin",
"Q57_bin", "response") facet_names <- c(Q52_bin = "Spirituality",
Q53_bin = "Politics L/R", Q57_bin = "Religiosity",
low="low", medium="medium", high="high") facet_labeller
<- function(variable,value){return(facet_names[value])}
dfresponse <- factor(dfresponse, ordered = TRUE, levels =
c("1", "2", "3", "4", "5")) dfresponse <- fct_recode(dfresponse,
"More than once a week" = "1", "Once a week" = "2", "At
least once a month" = "3", "Only on special holy days" =
"4", "Never" = "5") df %>% # we need to get the data
including facet info in long format, so we use pivot_longer()
pivot_longer(!response, names_to = "bin_name", values_to
= "b") %>% # add counts for plot below count(response,
bin_name, b) %>% group_by(bin_name,b) %>% mutate(perc=paste0(round(n*100/sum(n),1),"%")) %>% # run ggplot ggplot(aes(x = n, y = "", fill = response))
+ geom_col(position=position_fill(), aes(fill=response)) + geom_text(aes(label = perc), position = position_fill(vjust=.5),
size=2) + scale_fill_brewer(palette ="Dark2", type = "qual")
+ scale_x_continuous(labels = scales::percent_format()) +
facet_grid(vars(b), vars(bin_name), labeller=as_labeller(facet_names))
+ labs(caption = caption, x = "", y ="") + guides(fill =
guide_legend(title = NULL)) ggsave("figures/q59_faceted.png",
width = 30, height = 10, units = "cm")
```

14 Comparing with attitudes surrounding climate change

15 Q6

```
q6_data <- qualtrics_process_single_multiple_choice_unsorted_streamlined(climate_experience_data$Q6)
title <- "Do you think the climate is changing?"

level_order <- c("Don't know", "Definitely not changing", "Probably not changing", "Probably changing", "Definitely changing") ## code if a specific palette is needed for matching fill = wheel(ochre, num = as.integer(count(q6_data[1]))) # make plot q6_data_plot <- ggplot(q6_data, aes(x = n, y = response, fill = fill)) +
  geom_col(colour = "white") + ## add percentage labels
  geom_text(aes(label = perc), ## make labels left-aligned and white hjust = 1, colour = "black", size=4) + # use nudge_x = 30, to shift position ## reduce spacing between labels and bars
  scale_fill_identity(guide = "none") + ## get rid of all elements except y axis labels + adjust plot margin
  theme_ipsum_rc() + theme(plot.margin = margin(rep(15, 4))) +
  easy_center_title() + # with thanks for helpful info on doing wrap here: https://stackoverflow.com/questions/21878974/long-axis-labels-via-labeller-label-wrap-in-ggplot2
  scale_y_discrete(labels = wrap_format(30), limits = level_order) + theme(plot.title = element_text(size =18, hjust = 0.5), axis.text.y = element_text(size =16)) + labs(title = title, x = "", y = "")

q6_data_plot

ggsave("figures/q6.png", width = 18, height = 12, units = "cm")
```

16 Subsetting

16.1 Q57 subsetting based on Religiosity

```
climate_experience_data <- climate_experience_data %>%
  mutate( Q57_bin = case_when( Q57_1 > mean(Q57_1) +
    sd(Q57_1) ~ "high", Q57_1 < mean(Q57_1) - sd(Q57_1)
    ~ "low", TRUE ~ "medium" ) %>% factor(levels = c("low",
  "medium", "high")) )
```

16.2 Subsetting based on Spirituality

16.2.1 Nature relatedness

17 Calculate overall mean nature-relatedness score based on six questions:

```
climate_experience_data$Q51_score <- rowMeans(select(climate_experience_data,  
Q51_remote_vacation:Q51_heritage))
```

18 Create low/med/high bins based on Mean and +1/-1 Standard Deviation

```
climate_experience_data <- climate_experience_data  
%>% mutate( Q51_bin = case_when( Q51_score >  
mean(Q51_score) + sd(Q51_score) ~ "high", Q51_score  
< mean(Q51_score) - sd(Q51_score) ~ "low", TRUE ~  
"medium" ) %>% factor(levels = c("low", "medium", "high"))  
)
```

18.0.1 Spirituality scale _____

19 Calculate overall mean spirituality score based on six questions:

```
climate_experience_data$Q52_score <- rowMeans(select(climate_experience_data,  
Q52a_1:Q52f_1))
```

20 Create low/med/high bins based on Mean and +1/-1 Standard Deviation

```
climate_experience_data <- climate_experience_data  
%>% mutate( Q52_bin = case_when( Q52_score >  
mean(Q52_score) + sd(Q52_score) ~ "high", Q52_score  
< mean(Q52_score) - sd(Q52_score) ~ "low", TRUE ~  
"medium" ) %>% factor(levels = c("low", "medium", "high"))  
)
```

💡 What is Religion?

Content tbd

💡 Hybrid Religious Identity

Content tbd

💡 What is Secularisation?

Content tbd

References

21 Mapping churches: geospatial data science

Until recently, most data science books didn't have a section on geospatial data. It was considered a specialist form of research best left to GIS technicians who tended to use proprietary tools like ArcGIS. This has changed significantly in the past five years, but you'll still be hard pressed to find an introduction to the subject which strays very far from a few simple data sets (mostly of the USA) and relatively uncomplicated geospatial operations. I actually first began learning R, back in 2013, right when open source geospatial research tools were beginning to be developed with quite a lot more energy and geospatial data is my personal favourite data science playground, so in this book we're going to go much deeper than is usual. There are also good reasons to take things a few steps further in the particular forms of data and inquiry that religion takes us into.

Recommend <https://r-spatial.org/book/>

Geospatial data is, in the most basic form, working with maps. This means that most of your data can be a quite simple dataframe, e.g. just a list of names or categories associated with a set of X and Y coordinates. Once you have a set of items, however, things get interesting very quickly, as you can layer data sets on top of one another. We're going to begin this chapter by developing a geolocated data set of churches in the UK. This information is readily and freely available online thanks to the UK Ordnance Survey, a quasi-governmental agency which maintains the various (now digital) maps of Britain. Lucky for us, the Ordnance Survey has an open data product that anyone can use!

Before we begin, there are some key things we should note about geospatial data. Geospatial data tends to fall into one

of two kinds: points and polygons. Points can be any kind of feature: a house, a church, a pub, someone's favourite ancient oak tree, or some kind of sacred relic. Polygons tend to be associated with wider areas, and as such can be used to describe large features, e.g. an Ocean, a local authority, or a mountain, or also demographic features, like a census Output Area with associated census summaries. Points are very simple data representations, an X and Y coordinate. Polygons are much more complex, often containing dozens or even thousands of points.

The most complex aspect of point data relates to the ways that coordinates are encoded, as they will always need to be associated with a coordinate reference system (CRS) which describes how they are situated with respect to the planet earth. The most common CRS is the WGS, though for our data sets we'll also come into contact with the BGS, a specifically British coordinate reference system. There are dozens of CRS, usually mapping onto a specific geographical region. Bearing in mind the way that you need to use a CRS to understand how coordinates can be associated with specific parts of the earth, you can see how this is a bit like survey data, where you also need a "codebook" to understand what the specific response values map onto, e.g. a "1" means "strongly agree" and so on. We also saw, in a previous chapter, how some forms of data have the codebook already baked into the factor data, simplifying the process of interpreting the data. In a similar way, some types of geospatial data sets can also come with CRS "baked in" while we'll need to define CRS for other types. Here are some of the most common types of geospatial data files:

CSV: "comma separated values" a plain text file containing various coordinates
Shapefile: a legacy file format, often still in use, but being replaced by others for a variety of good reasons. For more on this see [<http://switchfromshapefile.org/>] Geopackage: one of the more recent ways of packaging up geospatial data. Geopackages can contain a wide variety of different data and are easily portable. GeoJSON: a file format commonly used in other forms of coding, the "JSON" (an acronym for JavaScript Object Notation) is meant to be easily interchangeable across various platforms. GeoJSON is an augmented version of JSON data with coordinates added in.

Now that you have a sense of some of the basic aspects of geospatial data, let's dive in and do a bit of learning in action.

22 Administrative shapes - the UK

A good starting point is to acquire some administrative data. This is a way of referring to political boundaries, whether country borders or those of a local authority or some other administrative unit. For our purposes, we're going to import several different types of administrative boundary which will be used at different points in our visualisations below. It's worth noting that the data we use here was prepared to support the 2011 census, and make use of the shapefile format.

```
library(sf) |> suppressPackageStartupMessages()
library(here) |> suppressPackageStartupMessages()
library(tidyverse) |> suppressPackageStartupMessages()
setwd("/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion")
here::i_am("chapter_3.qmd")

here() starts at /Users/kidwellj/gits/hacking_religion_textbook/hacking_religion

# Download administrative boundaries for whole UK at country level
if (file.exists(here("data", "infuse_uk_2011_clipped.shp")) == FALSE) {
  download.file("https://borders.ukdataservice.ac.uk/ukborders/easy_download/prebuilt/shape/in
unzip("data/infuse_uk_2011_clipped.zip", exdir = "data")
}
uk_countries <- st_read(here("data", "infuse_uk_2011_clipped.shp"))

Reading layer `infuse_uk_2011_clipped' from data source
`/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/data/infuse_uk_2011_clipped'
using driver `ESRI Shapefile'
Simple feature collection with 1 feature and 3 fields
Geometry type: MULTIPOLYGON
```

```
Dimension: XY
Bounding box: xmin: -69.1254 ymin: 5337.9 xmax: 655604.7 ymax: 1220302
Projected CRS: OSGB36 / British National Grid
```

```
# Download administrative boundaries for whole UK at regions level
if (file.exists(here("data", "infuse_rgn_2011_clipped.shp")) == FALSE) {
  download.file("https://borders.ukdataservice.ac.uk/ukborders/easy_download/prebuilt/shape/in
  unzip("data/infuse_rgn_2011_clipped.zip", exdir = "data")
}
uk_rgn <- st_read(here("data", "infuse_rgn_2011_clipped.shp"))
```

```
Reading layer `infuse_rgn_2011_clipped' from data source
`/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/data/infuse_rgn_2011_clipped.shp'
using driver `ESRI Shapefile'
Simple feature collection with 9 features and 2 fields
Geometry type: MULTIPOLYGON
Dimension: XY
Bounding box: xmin: 82672 ymin: 5337.9 xmax: 655604.7 ymax: 657534.1
Projected CRS: OSGB36 / British National Grid
```

```
# Download administrative boundaries for whole UK at local authority level
if (file.exists(here("data", "infuse_dist_lyr_2011_clipped.shp")) == FALSE) {
  download.file("https://borders.ukdataservice.ac.uk/ukborders/easy_download/prebuilt/shape/in
  unzip("data/infuse_dist_lyr_2011_clipped.zip", exdir = "data")
}
local_authorities <- st_read(here("data", "infuse_dist_lyr_2011_clipped.shp"))
```

```
Reading layer `infuse_dist_lyr_2011_clipped' from data source
`/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/data/infuse_dist_lyr_2011_clipped.shp'
using driver `ESRI Shapefile'
Simple feature collection with 404 features and 3 fields
Geometry type: MULTIPOLYGON
Dimension: XY
Bounding box: xmin: -69.1254 ymin: 5337.9 xmax: 655604.7 ymax: 1220302
Projected CRS: OSGB36 / British National Grid
```

```

# Download building outlines for whole UK
if (file.exists(here("data", "infuse_dist_lyr_2011_simplified_100m_buildings_simplified.gpkg"))
  download.file("https://zenodo.org/record/6395804/files/infuse_dist_lyr_2011_simplified_100m_buildings_simplified.gpkg")
local_authorities_buildings_clip <- st_read(here("data", "infuse_dist_lyr_2011_simplified_100m_buildings_simplified.gpkg"))

Reading layer `infuse_dist_lyr_2011_simplified_100m_buildings_overlay_simplified` from data source
  using driver `GPKG'
Simple feature collection with 403 features and 0 fields
Geometry type: MULTIPOLYGON
Dimension:      XY
Bounding box:  xmin: -69.1254 ymin: 5524.797 xmax: 655986.4 ymax: 1219597
Projected CRS: OSGB36 / British National Grid

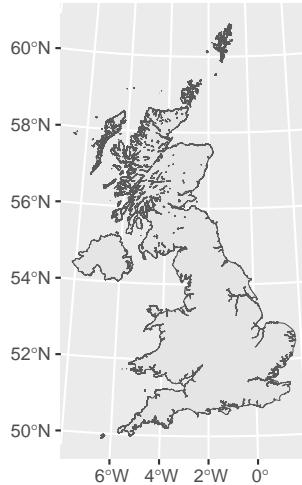
```

Before we move on, let's plot a simple map and have a look at one of our administrative layers. We can use ggplot with a new type of shape `geom_sf()` to plot the contents of a geospatial data file with polygons which is loaded as a `simplefeature` in R.

```

ggplot(uk_countries) +
  geom_sf()

```

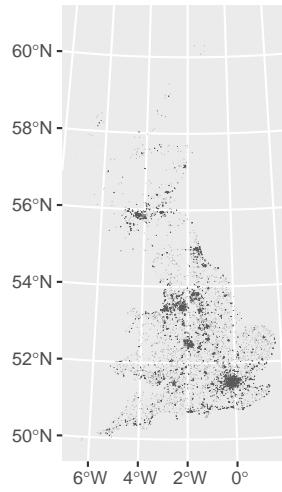


23 Load in Ordnance Survey OpenMap Points Data

```
# Note: for more advanced reproducible scripts which demonstrate how these data surces have  
# obtained, see the companion cookbook here: https://github.com/kidwellj/hacking\_religion\_co  
  
os_openmap_pow <- st_read(here("data", "os_openmap_pow.gpkg"))
```

```
Reading layer `os_openmap_pow' from data source  
`/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/data/os_openmap_pow.gpkg'  
using driver `GPKG'  
Simple feature collection with 48759 features and 5 fields  
Geometry type: POLYGON  
Dimension: XY  
Bounding box: xmin: 64594.12 ymin: 8287.54 xmax: 655238.1 ymax: 1214662  
Projected CRS: OSGB36 / British National Grid
```

```
ggplot(os_openmap_pow) +  
  geom_sf()
```



It's worth noting that the way that you load geospatial data in R has changed quite dramatically since 2020 with the introduction of the `sf` package in R. Much of the documentation you will come across "out there" will make reference to a set of functions which are now deprecated.

Let's use that data we've just loaded to make our first map:

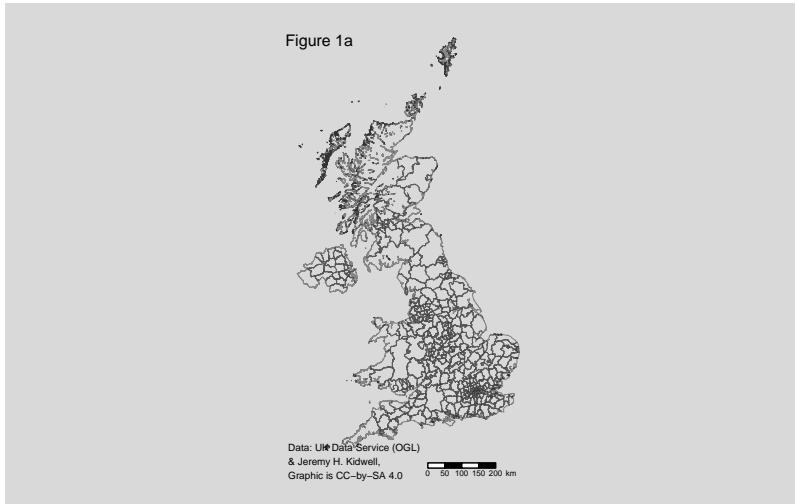
```
# Generate choropleth map of respondent locations
# using temporary palette here so that 0s are white
library(tmap) |> suppressPackageStartupMessages()
# palette <- c(white, "#a8ddb5", "#43a2ca")
map1 <- tm_shape(local_authorities) +
  # tm_fill(col = "surveys_count", , palette = palette, title = "Concentration of survey resp
  tm_borders(alpha=.5, lwd=0.1) +
  # for intermediate polygon geometries
  # tm_shape(local_authorities) +
  # tm_borders(lwd=0.6) +
  # for dots from original dataset
  # tm_dots("red", size = .05, alpha = .4) +
  tm_scale_bar(position = c("right", "bottom")) +
  tm_style("gray") +
  tm_credits("Data: UK Data Service (OGL)\n& Jeremy H. Kidwell,\nGraphic is CC-by-SA 4.0",
             size = 0.4,
             position = c("left", "bottom"),
             bg_color = "#f0f0f0")
```

```

        just = c("left", "bottom"),
        align = "left") +
tm_layout(asp = NA,
          frame = FALSE,
          title = "Figure 1a",
          title.size = .7,
          legend.title.size = .7,
          inner.margins = c(0.1, 0.1, 0.05, 0.05)
      )

map1

```



```

# save image
tmap_save(map1, here("figures", "map.png"), width=1920, height=1080, asp=0)

```

Map saved to /Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/figures/map.png

Resolution: 1920 by 1080 pixels

Size: 6.4 by 3.6 inches (300 dpi)

```

# subsetting ordnance survey openmap data for measuring clusters and proximity

os_openmap_important_buildings <- st_read(here("data", "os_openmap_important_buildings.gpkg"))

Reading layer `important_buildings' from data source
`/Users/kidwellj/gits/hacking_religion_textbook/hacking_religion/data/os_openmap_important_b
using driver `GPKG'
Simple feature collection with 229800 features and 5 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 64594.12 ymin: 8125.44 xmax: 655500.5 ymax: 1214662
Projected CRS: OSGB36 / British National Grid

# add pubs, check_cashing, pawnbrokers, SSSI
## subsets

# OSM data

# Note: for more advanced reproducible scripts which demonstrate how these data surces have
# obtained, see the companion cookbook here: https://github.com/kidwellj/hacking_religion_c

# osm_uk_points <- st_read(system.file(here("data", "pow_osm.gpkg", package = "spData")))
# vector_filepath = system.file("data/osm-gb-2018Aug29_pow_osm.pbf", package = "sf")
# osm_uk_points = st_read(vector_filepath)

Guides to geographies: https://rconsortium.github.io/censusguide/
https://ocsi.uk/2019/03/18/lsoas-leps-and-lookups-a-beginners-guide-to-statistical-geographies/

Extract places of worship from Ordnance survey open data set
Calculate proximity to pubs

```

References

24 Data scraping, corpus analysis and wordclouds

References

25 Summary

An open textbook introducing data science to religious studies

References