Advanced Epidemiological Analysis

Andreas M. Neophytou and G. Brooke Anderson

2021-10-25

Contents

Chapter 1

Overview

This is a the coursebook for the Colorado State University course ERHS 732, Advanced Epidemiological Analysis. This course provides the opportunity to implement theoretical expertise through designing and conducting advanced epidemiologic research analyses and to gain in-depth experience analyzing datasets from the environmental epidemiology literature.

This book is in development over the Fall 2021 semester.

1.1 License

Click on the **Next** button (or navigate using the links in the table of contents) to continue.

Chapter 2

Course information

This is a the coursebook for the Colorado State University course ERHS 732, Advanced Epidemiological Analysis. This course provides the opportunity to implement theoretical expertise through designing and conducting advanced epidemiologic research analyses and to gain in-depth experience analyzing datasets from the environmental epidemiology literature. This course will complement the student's training in advanced epidemiological methods, leveraging regression approaches and statistical programming, providing the opportunity to implement their theoretical expertise through designing and conducting advanced epidemiologic research analyses. Although basic theoretical frameworks behind analysis and statistical modeling approaches will be introduced, this course will not go into depth into statistical and epidemiologic theory and students are expected to be familiar with general epidemiologic concepts such as confounding, selection bias etc. During the course, students will gain in-depth experience analyzing two datasets from the environmental epidemiology literature—(1) time series data with daily measures of weather, air pollution, and cardiorespiratory outcomes in London, England and (2) a dataset with measures from the Framingham Heart Study. Additional datasets and studies will be discussed and explored as a supplement.

This class will utilize a variety of instructional formats, including short lectures, readings, topic specific examples from the substantive literature, discussion and directed group work on in-course coding exercises putting lecture and discussion content into practice. A variety of teaching modalities will be used, including group discussions, student directed discussions, and in-class group exercises. It is expected that before coming to class, students will read the required papers for the week, as well as any associated code included in the papers' supplemental materials. Students should come to class prepared to do statistical programming (i.e., bring a laptop with statistical software, download any datasets needed for the week etc). Participation is based on in-class coding exercises based on each week's topic. If a student misses a class, they will be expected to complete

the in-course exercise outside of class to receive credit for participation in that exercise. Students will be required to do mid-term and final projects which will be presented in class and submitted as a written write-up describing the project.

2.1 Course learning objectives

The learning objectives for this proposed course complement core epidemiology and statistics courses required by the program and provide the opportunity for students to implement theoretical skills and knowledge gained in those courses in a more applied setting.

Upon successful completion of this course students will be able to:

1. List several possible statistical approaches to answering an epidemiological research questions. (*Knowledge*)
2. Choose among analytical approaches learned in previous courses to identify one that is reasonable for an epidemiological research question. (*Application*)
3. Design a plan for cleaning and analyzing data to answer an epidemiological research question, drawing on techniques learned in previous and concurrent courses. (*Synthesis*)
4. Justify the methods and code used to answer an epidemiological research question. (*Evaluation*)
5. Explain the advantages and limitations of a chosen methodological approach for evaluating epidemiological data. (*Evaluation*)
6. Apply advanced epidemiological methods to analyze example data, using a regression modeling framework. (*Application*)
7. Apply statistical programming techniques learned in previous courses to prepare epidemiological data for statistical analysis and to conduct the analysis. (*Application*)
8. Interpret the output from statistical analyses of data for an epidemiological research question. (*Evaluation*)
9. Defend conclusions from their analysis. (*Comprehension*)
10. Write a report describing the methods, results, and conclusions from an epidemiological analysis. (*Application*)
11. Construct a reproducible document with embedded code to clean and analyze data to answer an epidemiological research question. (*Application*)

2.2 Meeting time and place

The class will meet on Mondays, 2:00–3:40 PM on the Colorado State University campus in MRB 312.

2.3 Class Structure and Expectations

- **Homework/preparation:** It is expected that *before* coming to class, students will read the required papers for the week, as well as the online book sections assigned for the week. Reading assignments will be announced the week before each class session. Students should come to class prepared to prepared to do statistical programming (i.e., bring in a laptop with statistical software, download any datasets needed for the week).
- **In-class schedule:**
  - Topic overview: Each class will start with a brief overview of the week's topic. This will focus on the material covered in that week's assigned reading in the online book and papers.
  - Discussion of analysis and coding points: Students and faculty will be divided into small groups to discuss the assigned reading and think more deeply about the content. This is a time to bring up questions and relate the chapter concepts to other datasets and/or analysis methods you are familiar with.
  - Group work: In small groups, students will work on designing an epidemiological analysis for the week's topic and developing code to implement that analysis. This will follow the prompts given in the assigned reading from the online book for the week.
  - Wrap-up: We will reconvene as one group at the end to discuss topics that came up in small group work and to outline expectations for students before the next meeting.

2.4 Course grading

Assessment Components	Percentage of Grade
Midterm written report	30
Midterm presentation	15
Final written report	30
Final presentation	15
Participation in in-course exercises	10

- **Midterm report (written and presentation):** Students will work in groups to prepare an oral presentation and accompanying written report presenting an epidemiologic analysis using a time series dataset similar to the London dataset used for the first half of the course. The group may pick a research question based on the topics covered in the first half of the course. The presentation should be 15 minutes and should be structured like a conference presentation (Introduction, Methods, Results, and Discussion). The written report should be approximately six pages (single spaced) and should cover the same topics. It should include at least two (up to four) well-designed figures and / or tables. The written report should be created following reproducible research principles and using a

bibliography referencing system (e.g., BibTex if the student uses RMarkdown to write the report). The report should be written to the standard expected for a peer-reviewed publication in terms of clarity, grammar, spelling, and referencing. These Midterm reports will be due (and presented) the eighth week of class (seventh class session, since there will be no class for Labor Day).

- **Final report (written and presentation):** Each student will prepare an oral presentation and accompanying written report presenting an epidemiologic analysis using either a dataset similar to the Framingham dataset used for the second half of the course or their own dataset from their research. The student may pick a research question based on the topics covered in the second half of the course. The presentation should be 10 minutes and should be structured like a conference presentation (Introduction, Methods, Results, and Discussion). The written report should be approximately six pages (single spaced) and should cover the same topics. It should include at least two well-designed figures and / or tables. The written report should be created following reproducible research principles and using a bibliography referencing system (e.g., BibTex if the student uses RMarkdown to write the report). The report should be written to the standard expected for a peer-reviewed publication in terms of clarity, grammar, spelling, and referencing. The final presentations will be given during the assigned time period for finals for our course.
- **Participation:** Attendance is an essential part of participating in the class. We understand things come up, however it is expected you attend every class and come prepared. Further, it is expected that you will actively participate in discussions and group work during the class period.

2.5 Course Schedule

Class	Date	Study type	Topic	Book sections
1	August 30	Time series	Time series / case-crossover study designs	3.1–3.4
2	September 3	Time series	Time series / case-crossover study designs	3.5
3	September 13	Time series	Generalized linear models	4.1–4.2
4	September 20	Time series	Generalized linear models	4.3
5	September 27	Time series	Natural experiments	5.1–5.3
6	October 4	Time series	Risk assessment	TBD
7	October 11	Time series	Group midterm reports	None
8	October 18	Cohort	Longitudinal cohort study designs	TBD
9	October 25	Cohort	Longitudinal cohort study designs	TBD
10	November 1	Cohort	Inverse probability weighting, Propensity scores	TBD
11	November 8	Cohort	Mixed models	TBD
12	November 15	Cohort	Instrumental variables	TBD
13	November 29	Cohort	Counterfactuals / Causal inference	TBD
14	December 6	Cohort	Finals preperation	None
15	December 13	Cohort	Finals presentation	None

2.6 Textbooks and Course Materials

Readings for this course will focus on peer-reviewed literature that will be posted for the students in the class, as well as assigned reading from this online book.

Additional general references that will be useful to students throughout the semester include:

- Garrett Grolemund and Hadley Wickham, *R for Data Science*, O'Reilly, 2017. (Available for free online at https://r4ds.had.co.nz/ and in print through most large book sellers.)
- Miguel A. Hernán and James M. Robins, *Causal Inference: What If*, Boca Raton: Chapman & Hall/CRC, 2020. (Available for free online at https://cdn1.sph.harvard.edu/wp-content/uploads/sites/1268/2021/01/ciwhatif_hernanrobins_31jan21.pdf with a print version anticipated in 2021.)
- Francesca Dominici and Roger D. Peng, *Statistical Methods for Environmental Epidemiology with R*, Springer, 2008. (Available online through the CSU library or in print through Springer.)

2.7 Prerequisites and Preparation

This course assumes experience in epidemiology and some experience programming in statistical programming (e.g., R, SAS) and statistics. Students should have taken (1) Epidemiologic Methods (ERHS 532/PBHL 570), (2) Advanced Epidemiology (ERHS 640), (3) R Programming for Research (ERHS 535) or SAS and Epidemiologic Data Management (ERHS 534), and (4) Design and Data Analysis for Researchers I (STAT 511), or equivalent courses or experience, prior to taking this course. While previous SAS experience is acceptable for the course, example code will be in R, so you will find it helpful to review the basics of R prior to class.

If you would like to prepare for this course, the best way is by reviewing R programming, epidemiology, and regression modeling. In each chapter of the book, we provide the required and supplemental reading for the week. If you need a review in any of these topics, we recommend starting with the papers and book chapters listed as general reviews of these topics in the chapters' supplemental reading sections.

2.8 Academic Honesty

Lack of knowledge of the academic honesty policy is not a reasonable explanation for a violation. For more on Colorado State University's policies on Academic Integrity / Misconduct.

Chapter 3

Time series / case-crossover studies

We'll start by exploring common characteristics in time series data for environmental epidemiology. In the first have of the class, we're focusing on a very specific type of study—one that leverages large-scale vital statistics data, collected at a regular time scale (e.g., daily), combined with large-scale measurements of a climate-related exposure, with the goal of estimating the typical relationship between the level of the exposure and risk of a health outcome. For example, we may have daily measurements of particulate matter pollution for a city, measured daily at a set of Environmental Protection Agency (EPA) monitors. We want to investigate how risk of cardiovascular mortality changes in the city from day to day in association with these pollution levels. If we have daily counts of the number of cardiovascular deaths in the city, we can create a statistical model that fits the exposure-response association between particulate matter concentration and daily risk of cardiovascular mortality. These statistical models—and the type of data used to fit them—will be the focus of the first part of this course.

3.1 Readings

The required readings for this chapter are:

- Bhaskaran et al. (2013) Provides an overview of time series regression in environmental epidemiology.
- Vicedo-Cabrera et al. (2019) Provides a tutorial of all the steps for a projecting of health impacts of temperature extremes under climate change. One of the steps is to fit the exposure-response association using present-day data (the section on "Estimation of Exposure-Response Associations" in the paper). In this chapter, we will go into details on that step, and

that section of the paper is the only required reading for this chapter. Later in the class, we'll look at other steps covered in this paper. Supplemental material for this paper is available to download by clicking http://links.lww.com/EDE/B504. You will need the data in this supplement for the exercises for class.

The following are supplemental readings (i.e., not required, but may be of interest) associated with the material in this chapter:

- Armstrong et al. (2012) Commentary that provides context on how epidemiological research on temperature and health can help inform climate change policy.
- Dominici and Peng (2008c) Overview of study designs for studying climate-related exposures (air pollution in this case) and human health. Chapter in a book that is available online through the CSU library.
- Armstrong (2006) Covers similar material as Bhaskaran et al. (2013), but with more focus on the statistical modeling framework
- Gasparrini and Armstrong (2010) Describes some of the advances made to time series study designs and statistical analysis, specifically in the context of temperature
- Basu et al. (2005) Compares time series and case-crossover study designs in the context of exploring temperature and health. Includes a nice illustration of different referent periods, including time-stratified.
- Armstrong et al. (2014) This paper describes different data structures for case-crossover data, as well as how conditional Poisson regression can be used in some cases to fit a statistical model to these data. Supplemental material for this paper is available at https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/1471-2288-14-122#Sec13.
- Imai et al. (2015) Typically, the time series study design covered in this chapter is used to study non-communicable health outcomes. This paper discusses opportunities and limitations in applying a similar framework for infectious disease.
- Dominici and Peng (2008b) Heavier on statistics. Describes some of the statistical challenges of working with time series data for air pollution epidemiology. Chapter in a book that is available online through the CSU library.
- Lu and Zeger (2007) Heavier on statistics. This paper shows how, under conditions often common for environmental epidemiology studies, case-crossover and time series methods are equivalent.
- Gasparrini (2014) Heavier on statistics. This provides the statistical framework for the distributed lag model for environmental epidemiology time series studies.
- Dunn and Smyth (2018) Introduction to statistical models, moving into regression models and generalized linear models. Chapter in a book that is available online through the CSU library.
- James et al. (2013) General overview of linear regression, with an R coding "lab" at the end to provide coding examples. Covers model fit, continuous,

binary, and categorical covariates, and interaction terms. Chapter in a book that is available online through the CSU library.

3.2 Time series and case-crossover study designs

In the first half of this course, we'll take a deep look at how researchers can study how environmental exposures and health risk are linked using *time series studies*. Let's start by exploring the study design for this type of study, as well as a closely linked study design, that of *case-crossover studies*.

It's important to clarify the vocabulary we're using here. We'll use the terms *time series study* and *case-crossover study* to refer specifically to a type of study common for studying air pollution and other climate-related exposures. However, both terms have broader definitions, particularly in fields outside environmental epidemiology. For example, a *time series study* more generally refers to a study where data is available for the same unit (e.g., a city) for multiple time points, typically at regularly-spaced times (e.g., daily). A variety of statistical methods have been developed to apply to gain insight from this type of data, some of which are currently rarely used in the specific fields of air pollution and climate epidemiology that we'll explore here. For example, there are methods to address autocorrelation over time in measurements—that is, that measurements taken at closer time points are likely somewhat correlated—that we won't cover here and that you won't see applied often in environmental epidemiology studies, but that might be the focus of a "Time Series" course in a statistics or economics department.

In air pollution and climate epidemiology, time series studies typically begin with study data collected for an aggregated area (e.g., city, county, ZIP code) and with a daily resolution. These data are usually secondary data, originally collected by the government or other organizations through vital statistics or other medical records (for the health data) and networks of monitors for the exposure data. In the next section of this chapter, we'll explore common characteristics of these data. These data are used in a time series study to investigate how changes in the daily level of the exposure is associated with risk of a health outcome, focusing on the short-term period. For example, a study might investigate how risk of respiratory hospitalization in a city changes in relationship with the concentration of particulate matter during the week or two following exposure. The study period for these studies is often very long (often a decade or longer), and while single-community time series studies can be conducted, many time series studies for environmental epidemiology now include a large set of communities of national or international scope.

The study design essentially compares a community with itself at different time points—asking if health risk tends to be higher on days when exposure is higher. By comparing the community to itself, the design removes many challenges that would come up when comparing one community to another (e.g., is respiratory

hospitalization risk higher in city A than city B because particulate matter concentrations are typically higher in city A?). Communities differ in demographics and other factors that influence health risk, and it can be hard to properly control for these when exploring the role of environmental exposures. By comparison, demographics tend to change slowly over time (at least, compared to a daily scale) within a community.

One limitation, however, is that the study design is often best-suited to study acute effects, but more limited in studying chronic health effects. This is tied to the design and traditional ways of statistically modeling the resulting data. Since a community is compared with itself, the design removes challenges in comparing across communities, but it introduces new ones in comparing across time. Both environmental exposures and rates of health outcomes can have strong patterns over time, both across the year (e.g., mortality rates tend to follow a strong seasonal pattern, with higher rates in winter) and across longer periods (e.g., over the decade or longer of a study period). These patterns must be addressed through the statistical model fit to the time series data, and they make it hard to disentangle chronic effects of the exposure from unrelated temporal patterns in the exposure and outcome, and so most time series studies will focus on the short-term (or acute) association between exposure and outcome, typically looking at a period of at most about a month following exposure.

The term *case-crossover study* is a bit more specific than *time series study*, although there has been a strong movement in environmental epidemiology towards applying a specific version of the design, and so in this field the term often now implies this more specific version of the design. Broadly, a case-crossover study is one in which the conditions at the time of a health outcome are compared to conditions at other times that should otherwise (i.e., outside of the exposure of interest) be comparable. A case-crossover study could, for example, investigate the association between weather and car accidents by taking a set of car accidents and investigating how weather during the car accident compared to weather in the same location the week before.

One choice in a case-crossover study design is how to select the control time periods. Early studies tended to use a simple method for this—for example, taking the day before, or a day the week before, or some similar period somewhat close to the day of the outcome. As researchers applied the study design to large sets of data (e.g., all deaths in a community over multiple years), they noticed that some choices could create bias in estimates. As a result, most environmental epidemiology case-crossover studies now use a *time-stratified* approach to selecting control days. This selects a set of control days that typically include days both before and after the day of the health outcome, and are a defined set of days within a "stratum" that should be comparable in terms of temporal trends. For daily-resolved data, this stratum typically will include all the days within a month, year, and day of week. For example, one stratum of comparable days might be all the Mondays in January of 2010. These stratums are created throughout the study period, and then days are only compared to

other days within their stratum (although, fortunately, there are ways you can apply a single statistical model to fit all the data for this approach rather than having to fit code stratum-by-stratum over many years).

When this is applied to data at an aggregated level (e.g., city, county, or ZIP code), it is in spirit very similar to a time series study design, in that you are comparing a community to itself at different time points. The main difference is that a time series study uses statistical modeling to control from potential confounding from temporal patterns, while a case-crossover study of this type instead controls for this potential confounding by only comparing days that should be "comparable" in terms of temporal trends, for example, comparing a day only to other days in the same month, year, and day of week. You will often hear that case-crossover studies therefore address potential confounding for temporal patterns "by design" rather than "statistically" (as in time series studies). However, in practice (and as we'll explore in this class), in environmental epidemiology, case-crossover studies often are applied to aggregated community-level data, rather than individual-level data, with exposure assumed to be the same for everyone in the community on a given day. Under these assumptions, time series and case-crossover studies have been determined to be essentially equivalent (and, in fact, can use the same study data), only with slightly different terms used to control for temporal patterns in the statistical model fit to the data. Several interesting papers have been written to explore differences and similarities in these two study designs as applied in environmental epidemiology (Basu et al., 2005; Armstrong et al., 2014; Lu and Zeger, 2007).

These types of study designs in practice use similar datasets. In earlier presentations of the case-crossover design, these data would be set up a bit differently for statistical modeling. More recent work, however, has clarified how they can be modeled similarly to when using a time series study design, allowing the data to be set up in a similar way (Armstrong et al., 2014).

Several excellent commentaries or reviews are available that provide more details on these two study designs and how they have been used specifically investigate the relationship between climate-related exposures and health (Bhaskaran et al., 2013; Armstrong, 2006; Gasparrini and Armstrong, 2010). Further, these designs are just two tools in a wider collection of study designs that can be used to explore the health effects of climate-related exposures. Dominici and Peng (2008c) provides a nice overview of this broader set of designs.

3.3 Time series data

Let's explore the type of dataset that can be used for these time series–style studies in environmental epidemiology. In the examples in this chapter, we'll be using data that comes as part of the Supplemental Material in one of this chapter's required readings, (Vicedo-Cabrera et al., 2019). Follow the link for the supplement for this article and then look for the file "lndn_obs.csv". This

is the file we'll use as the example data in this chapter.

These data are saved in a csv format (that is, a plain text file, with commas used as the delimiter), and so they can be read into R using the `read_csv` function from the `readr` package (part of the tidyverse). For example, you can use the following code to read in these data, assuming you have saved them in a "data" subdirectory of your current working directory:

```
library(tidyverse) # Loads all the tidyverse packages, including readr
obs <- read_csv("data/lndn_obs.csv")
obs
```

```
## # A tibble: 8,279 x 14
##    date        year month   day   doy dow     all all_0_64 all_65_74 all_75_84
##    <date>     <dbl> <dbl> <dbl> <dbl> <chr> <dbl>    <dbl>     <dbl>     <dbl>
##  1 1990-01-01  1990     1     1     1 Mon     220       38        38        82
##  2 1990-01-02  1990     1     2     2 Tue     257       50        67        87
##  3 1990-01-03  1990     1     3     3 Wed     245       39        59        86
##  4 1990-01-04  1990     1     4     4 Thu     226       41        45        77
##  5 1990-01-05  1990     1     5     5 Fri     236       45        54        85
##  6 1990-01-06  1990     1     6     6 Sat     235       48        48        84
##  7 1990-01-07  1990     1     7     7 Sun     231       38        49        96
##  8 1990-01-08  1990     1     8     8 Mon     235       46        57        76
##  9 1990-01-09  1990     1     9     9 Tue     250       48        54        96
## 10 1990-01-10  1990     1    10    10 Wed     214       44        46        62
## # ... with 8,269 more rows, and 4 more variables: all_85plus <dbl>,
## #   tmean <dbl>, tmin <dbl>, tmax <dbl>
```

This example dataset shows many characteristics that are common for datasets for time series studies in environmental epidemiology. Time series data are essentially a sequence of data points repeatedly taken over a certain time interval (e.g., day, week, month etc). General characteristics of time series data for environmental epidemiology studies are:

- Observations are given at an aggregated level. For example, instead of individual observations for each person in London, the `obs` data give counts of deaths throughout London. The level of aggregation is often determined by geopolitical boundaries, for example, counties or ZIP codes in the US.
- Observations are given at regularly spaced time steps over a period. In the `obs` dataset, the time interval is day. Typically, values will be provided continuously over that time period, with observations for each time interval. Occasionally, however, the time series data may only be available for particular seasons (e.g., only warm season dates for an ozone study), or there may be some missing data on either the exposure or health outcome over the course of the study period.
- Observations are available at the same time step (e.g., daily) for (1) the health outcome, (2) the environmental exposure of interest, and (3) potential time-varying confounders. In the `obs` dataset, the health outcome

is mortality (from all causes; sometimes, the health outcome will focus on a specific cause of mortality or other health outcomes such as hospitalizations or emergency room visits). Counts are given for everyone in the city for each day (`all` column), as well as for specific age categories (`all_0_64` for all deaths among those up to 64 years old, and so on). The exposure of interest in the `obs` dataset is temperature, and three metrics of this are included (`tmean`, `tmin`, and `tmax`). Day of the week is one time-varying factor that could be a confounder, or at least help explain variation in the outcome (mortality). This is included through the `dow` variable in the `obs` data. Sometimes, you will also see a marker for holidays included as a potential time-varying confounder, or other exposure variables (temperature is a potential confounder, for example, when investigating the relationship between air pollution and mortality risk).

- Multiple metrics of an exposure and / or multiple health outcome counts may be included for each time step. In the `obs` example, three metrics of temperature are included (minimum daily temperature, maximum daily temperature, and mean daily temperature). Several counts of mortality are included, providing information for specific age categories in the population. The different metrics of exposure will typically be fit in separate models, either as a sensitivity analysis or to explore how exposure measurement affects epidemiological results. If different health outcome counts are available, these can be modeled in separate statistical models to determine an exposure-response function for each outcome.

3.4 Exploratory data analysis

When working with time series data, it is helpful to start with some exploratory data analysis. This type of time series data will often be secondary data—it is data that was previously collected, as you are re-using it. Exploratory data analysis is particularly important with secondary data like this. For primary data that you collected yourself, following protocols that you designed yourself, you will often be very familiar with the structure of the data and any quirks in it by the time you are ready to fit a statistical model. With secondary data, however, you will typically start with much less familiarity about the data, how it was collected, and any potential issues with it, like missing data and outliers.

Exploratory data analysis can help you become familiar with your data. You can use summaries and plots to explore the parameters of the data, and also to identify trends and patterns that may be useful in designing an appropriate statistical model. For example, you can explore how values of the health outcome are distributed, which can help you determine what type of regression model would be appropriate, and to see if there are potential confounders that have regular relationships with both the health outcome and the exposure of interest. You can see how many observations have missing data for the outcome, the exposure, or confounders of interest, and you can see if there are any measurements

that look unusual. This can help in identifying quirks in how the data were recorded—for example, in some cases ground-based weather monitors use -99 or -999 to represent missing values, definitely something you want to catch and clean-up in your data (replacing with R's `NA` for missing values) before fitting a statistical model!

The following applied exercise will take you through some of the questions you might want to answer through this type of exploratory analysis. In general, the `tidyverse` suite of R packages has loads of tools for exploring and visualizing data in R. The `lubridate` package from the `tidyverse`, for example, is an excellent tool for working with date-time data in R, and time series data will typically have at least one column with the timestamp of the observation (e.g., the date for daily data). You may find it worthwhile to explore this package some more. There is a helpful chapter in Wickham and Grolemund (2016), https://r4ds.had.co.nz/dates-and-times.html, as well as a cheatsheet at https://evoldyn.gitlab.io/evomics-2018/ref-sheets/R_lubridate.pdf. For visualizations, if you are still learning techniques in R, two books you may find useful are Healy (2018) (available online at https://socviz.co/) and Chang (2018) (available online at http://www.cookbook-r.com/Graphs/).

Applied: Exploring time series data

Read the example time series data into R and explore it to answer the following questions:

1. What is the study period for the example `obs` dataset? (i.e., what dates / years are covered by the time series data?)
2. Are there any missing dates (i.e., dates with nothing recorded) within this time period? Are there any recorded dates where health outcome measurements are missing? Any where exposure measurements are missing?
3. Are there seasonal trends in the exposure? In the outcome?
4. Are there long-term trends in the exposure? In the outcome?
5. Is the outcome associated with day of week? Is the exposure associated with day of week?

Based on your exploratory analysis in this section, talk about the potential for confounding when these data are analyzed to estimate the association between daily temperature and city-wide mortality. Is confounding by seasonal trends a concern? How about confounding by long-term trends in exposure and mortality? How about confounding by day of week?

Applied exercise: Example code

1. **What is the study period for the example `obs` dataset? (i.e., what dates / years are covered by the time series data?)**

In the `obs` dataset, the date of each observation is included in a column called `date`. The data type of this column is "Date"—you can check this by using the `class` function from base R:

```
class(obs$date)
```

```
## [1] "Date"
```

Since this column has a "Date" data type, you can run some mathematical function calls on it. For example, you can use the `min` function from base R to get the earliest date in the dataset and the `max` function to get the latest.

```
min(obs$date)
```

```
## [1] "1990-01-01"
```

```
max(obs$date)
```

```
## [1] "2012-08-31"
```

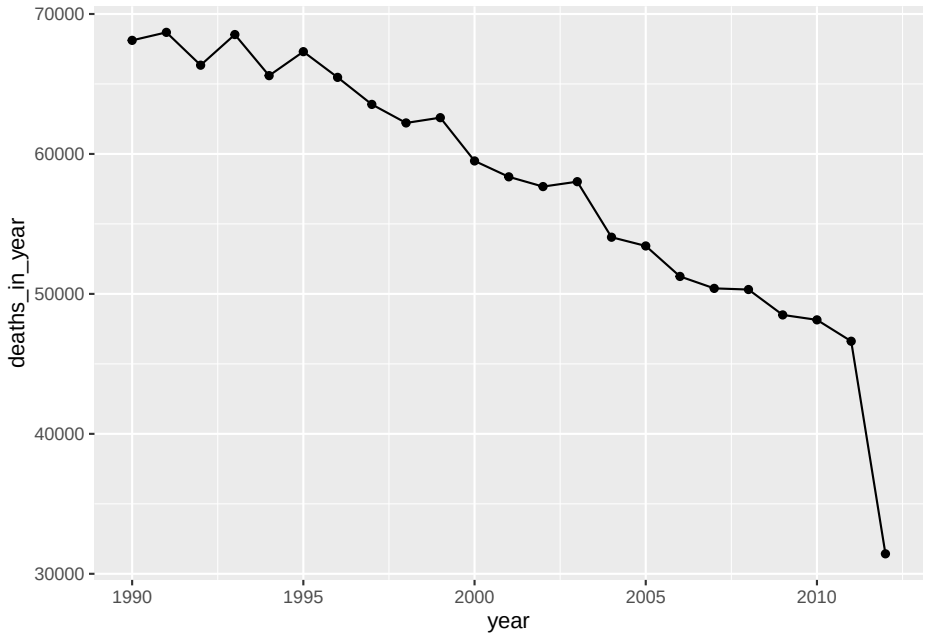
You can also run the `range` function to get both the earliest and latest dates with a single call:

```
range(obs$date)
```

```
## [1] "1990-01-01" "2012-08-31"
```

This provides the range of the study period for these data. One interesting point is that it's not a round set of years—instead, the data ends during the summer of the last study year. This doesn't present a big problem, but is certainly something to keep in mind if you're trying to calculate yearly averages of any values for the dataset. If you're getting the average of something that varies by season (e.g., temperature), it could be slightly weighted by the months that are included versus excluded in the partial final year of the dataset. Similarly, if you group by year and then count totals by year, the number will be smaller for the last year, since only part of the year's included. For example, if you wanted to count the total deaths in each year of the study period, it will look like they go down a lot the last year, when really it's only because only about half of the last year is included in the study period:

```
obs %>%
  group_by(year) %>%
  summarize(deaths_in_year = sum(all)) %>%
  ggplot(aes(x = year, y = deaths_in_year)) +
  geom_line() +
  geom_point()
```



2. **Are there any missing dates within this time period? Are there any recorded dates where health outcome measurements are missing? Any where exposure measurements are missing?**

There are a few things you should check to answer this question. First (and easiest), you can check to see if there are any `NA` values within any of the observations in the dataset. This helps answer the second and third parts of the question. The `summary` function will provide a summary of the values in each column of the dataset, including the count of missing values (`NA`s) if there are any:

```
summary(obs)
```

```
##       date                 year          month             day        
##  Min.   :1990-01-01   Min.   :1990   Min.   : 1.000   Min.   : 1.00  
##  1st Qu.:1995-09-01   1st Qu.:1995   1st Qu.: 3.000   1st Qu.: 8.00  
##  Median :2001-05-02   Median :2001   Median : 6.000   Median :16.00  
##  Mean   :2001-05-02   Mean   :2001   Mean   : 6.464   Mean   :15.73  
##  3rd Qu.:2006-12-31   3rd Qu.:2006   3rd Qu.: 9.000   3rd Qu.:23.00  
##  Max.   :2012-08-31   Max.   :2012   Max.   :12.000   Max.   :31.00  
##       doy            dow                 all           all_0_64     
##  Min.   :  1.0   Length:8279        Min.   : 81.0   Min.   : 9.0  
##  1st Qu.: 90.5   Class :character   1st Qu.:138.0   1st Qu.:27.0  
##  Median :180.0   Mode  :character   Median :157.0   Median :32.0  
##  Mean   :181.3                      Mean   :160.2   Mean   :32.4  
##  3rd Qu.:272.0                      3rd Qu.:178.0   3rd Qu.:37.0
```

```
##  Max.   :366.0                     Max.   :363.0   Max.   :64.0
##    all_65_74       all_75_84        all_85plus         tmean
##  Min.   : 6.00   Min.   : 17.00   Min.   : 17.00   Min.   :-5.503
##  1st Qu.:23.00   1st Qu.: 41.00   1st Qu.: 39.00   1st Qu.: 7.470
##  Median :29.00   Median : 49.00   Median : 45.00   Median :11.465
##  Mean   :30.45   Mean   : 50.65   Mean   : 46.68   Mean   :11.614
##  3rd Qu.:37.00   3rd Qu.: 58.00   3rd Qu.: 53.00   3rd Qu.:15.931
##  Max.   :70.00   Max.   :138.00   Max.   :128.00   Max.   :29.143
##       tmin             tmax
##  Min.   :-8.940   Min.   :-3.785
##  1st Qu.: 3.674   1st Qu.:10.300
##  Median : 7.638   Median :14.782
##  Mean   : 7.468   Mean   :15.058
##  3rd Qu.:11.438   3rd Qu.:19.830
##  Max.   :20.438   Max.   :37.087
```

Based on this analysis, all observations are complete for all dates included in the dataset. There are no listings for `NA`s for any of the columns, and this indicates no missing values in the dates for which there's a row in the data.

However, this does not guarantee that every date between the start date and end date of the study period are included in the recorded data. Sometimes, some dates might not get recorded at all in the dataset, and the `summary` function won't help you determine when this is the case. One common example in environmental epidemiology is with ozone pollution data. These are sometimes only measured in the warm season, and so may be shared in a dataset with all dates outside of the warm season excluded.

There are a few alternative explorations you can do to check this. Perhaps the easiest is to check the number of days between the start and end date of the study period, and then see if the number of observations in the dataset is the same:

```
# Calculate number of days in study period
obs %>%             # Using piping (%>%) throughout to keep code clear
  pull(date) %>%    # Extract the `date` column as a vector
  range() %>%       # Take the range of dates (earliest and latest)
  diff()            # Calculate time difference from start to finish of study
```

```
## Time difference of 8278 days
```

```
# Get number of observations in dataset---should be 1 more than time difference
obs %>%
  nrow()
```

```
## [1] 8279
```

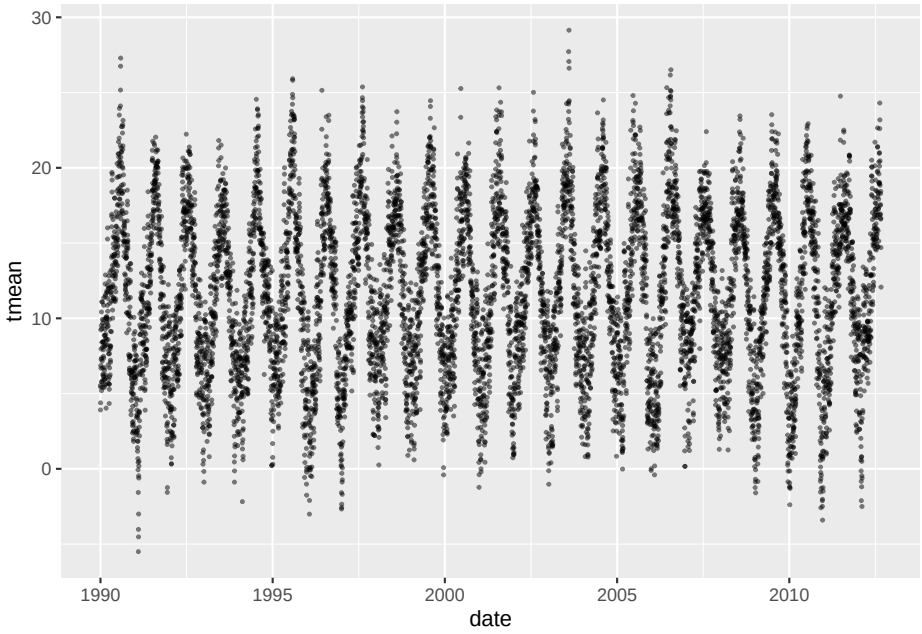
This indicates that there is an observation for every date over the study period, since the number of observations should be one more than the time difference.

In the next question, we'll be plotting observations by time, and typically this will also help you see if there are large chunks of missing dates in the data.

3. **Are there seasonal trends in the exposure? In the outcome?**

You can use a simple plot to visualize patterns over time in both the exposure and the outcome. For example, the following code plots a dot for each daily temperature observation over the study period. The points are set to a smaller size (`size = 0.5`) and plotted with some transparency (`alpha = 0.5`) since there are so many observations.

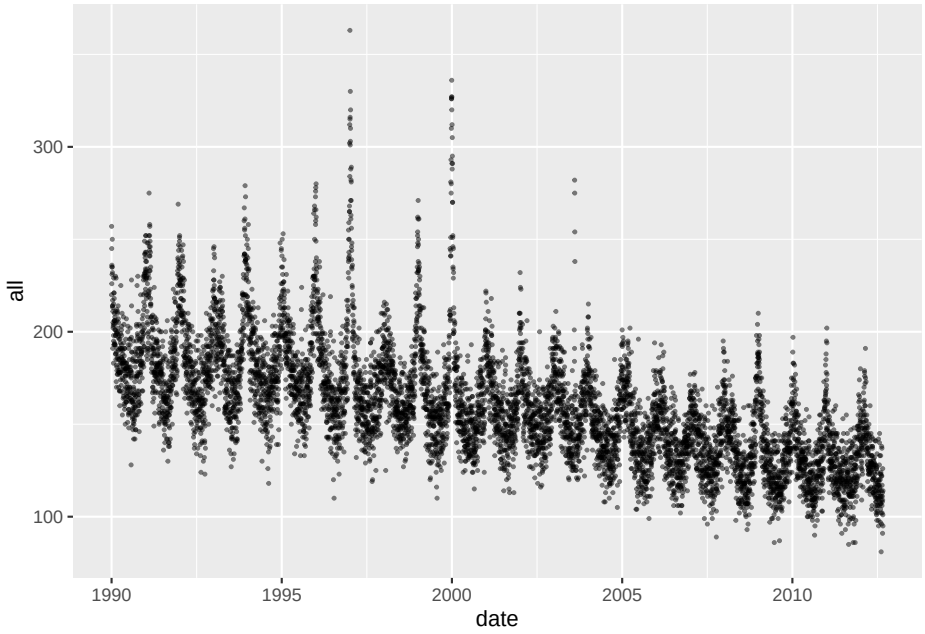
```
ggplot(obs, aes(x = date, y = tmean)) +
  geom_point(alpha = 0.5, size = 0.5)
```



There is (unsurprisingly) clear evidence here of a strong seasonal trend in mean temperature, with values typically lowest in the winter and highest in the summer.

You can plot the outcome variable in the same way:

```
ggplot(obs, aes(x = date, y = all)) +
  geom_point(alpha = 0.5, size = 0.5)
```

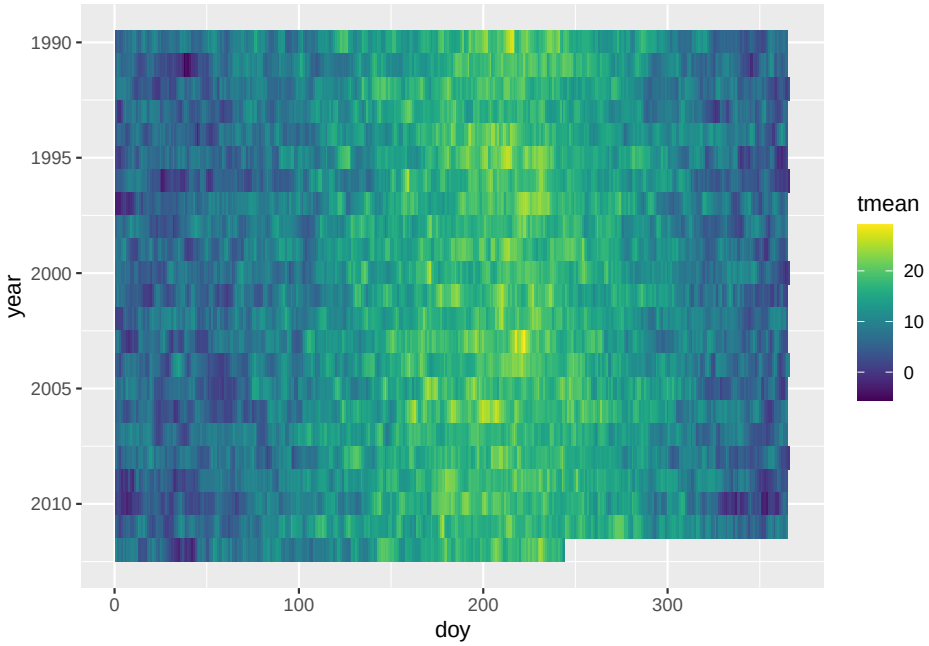


Again, there are seasonal trends, although in this case they are inversed. Mortality tends to be highest in the winter and lowest in the summer. Further, the seasonal pattern is not equally strong in all years—some years it has a much higher winter peak, probably in conjunction with severe influenza seasons.

Another way to look for seasonal trends is with a heatmap-style visualization, with day of year along the x-axis and year along the y-axis. This allows you to see patterns that repeat around the same time of the year each year (and also unusual deviations from normal seasonal patterns).

For example, here's a plot showing temperature in each year, where the observations are aligned on the x-axis by time in year. We're using the `doy`—which stands for "day of year" (i.e., Jan 1 = 1; Jan 2 = 2; … Dec 31 = 365 as long as it's not a leap year) as the measure of time in the year. We've reversed the y-axis so that the earliest years in the study period start at the top of the visual, then later study years come later—this is a personal style, and it would be no problem to leave the y-axis as-is. We've used the `viridis` color scale for the fill, since that has a number of features that make it preferable to the default R color scale, including that it is perceptible for most types of color blindness and be printed out in grayscale and still be correctly interpreted.

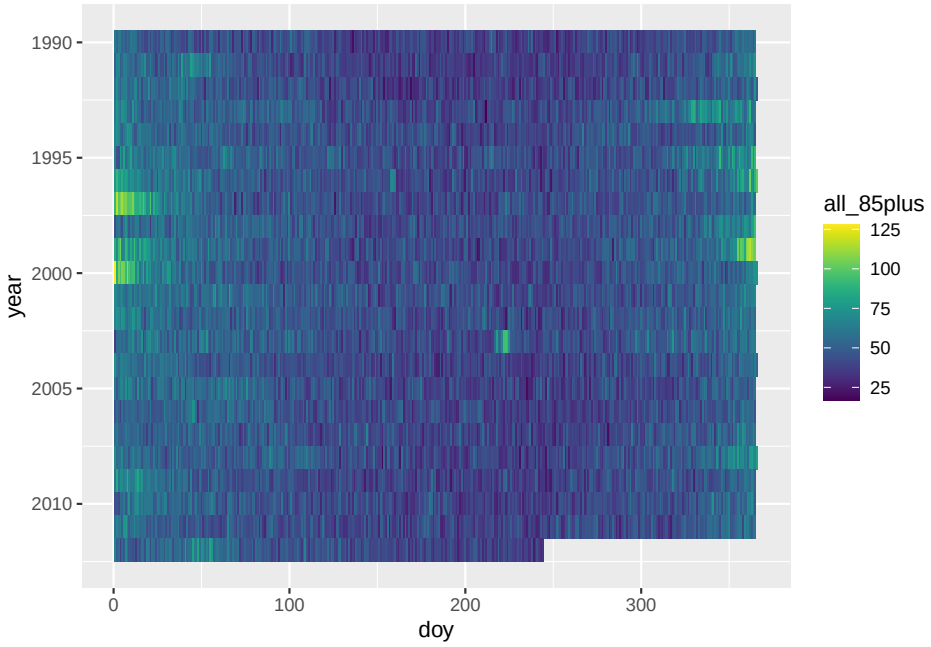
```
library(viridis)
ggplot(obs, aes(x = doy, y = year, fill = tmean)) +
  geom_tile() +
  scale_y_reverse() +
  scale_fill_viridis()
```



From this visualization, you can see that temperatures tend to be higher in the summer months and lower in the winter months. "Spells" of extreme heat or cold are visible—where extreme temperatures tend to persist over a period, rather than randomly fluctuating within a season. You can also see unusual events, like the extreme heat wave in the summer of 2003, indicated with the brightest yellow in the plot.

We created the same style of plot for the health outcome. In this case, we focused on mortality among the oldest age group, as temperature sensitivity tends to increase with age, so this might be where the strongest patterns are evident.

```
ggplot(obs, aes(x = doy, y = year, fill = all_85plus)) +
  geom_tile() +
  scale_y_reverse() +
  scale_fill_viridis()
```

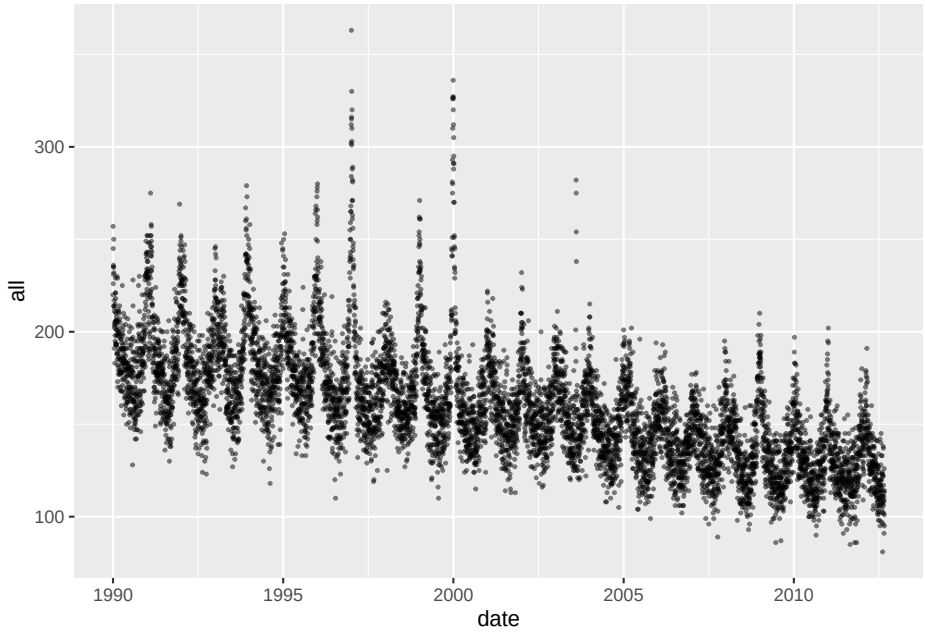


For mortality, there tends to be an increase in the winter compared to the summer. Some winters have stretches with particularly high mortality—these are likely a result of seasons with strong influenza outbreaks. You can also see on this plot the impact of the 2003 heat wave on mortality among this oldest age group—an unusual spot of light green in the summer.

4. **Are there long-term trends in the exposure? In the outcome?**

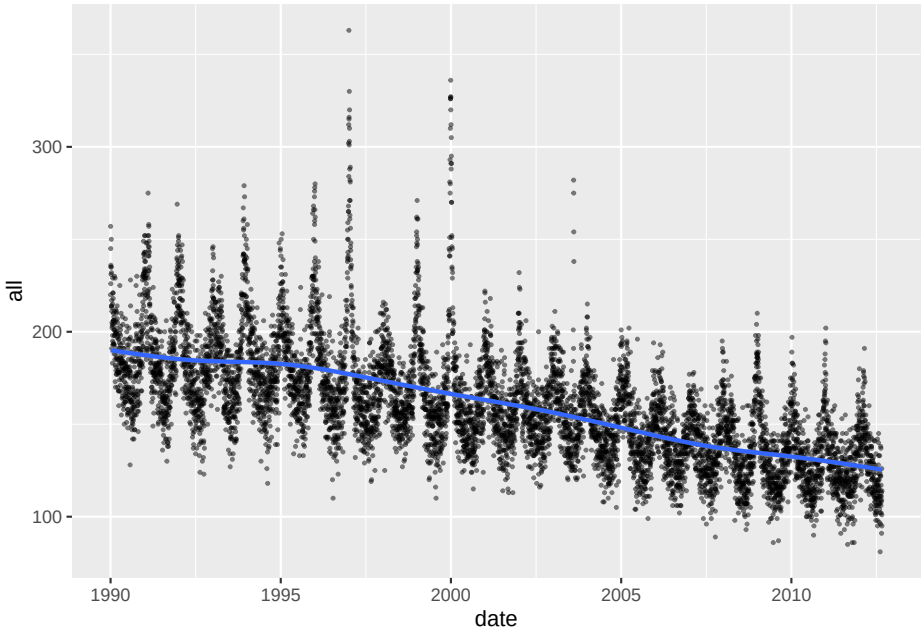
Some of the plots we created in the last section help in exploring this question. For example, the following plot shows a clear pattern of decreasing daily mortality counts, on average, over the course of the study period:

```
ggplot(obs, aes(x = date, y = all)) +
  geom_point(alpha = 0.5, size = 0.5)
```



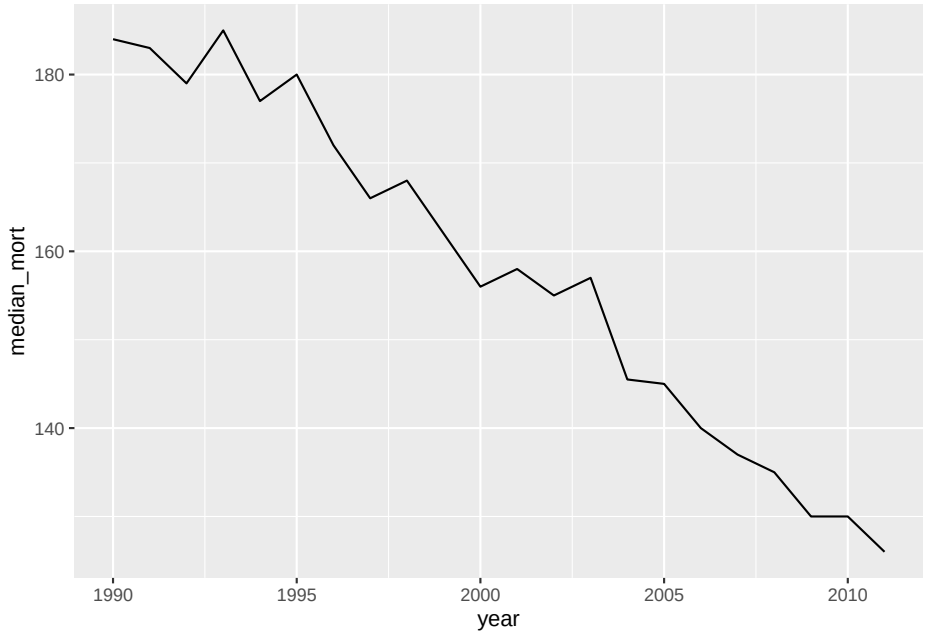
It can be helpful to add a smooth line to help detect these longer-term patterns, which you can do with `geom_smooth`:

```
ggplot(obs, aes(x = date, y = all)) +
  geom_point(alpha = 0.5, size = 0.5) +
  geom_smooth()
```



You could also take the median mortality count across each year in the study period, although you should take out any years without a full year's worth of data before you do this, since there are seasonal trends in the outcome:

```
obs %>%
  group_by(year) %>%
  filter(year != 2012) %>% # Take out the last year
  summarize(median_mort = median(all)) %>%
  ggplot(aes(x = year, y = median_mort)) +
  geom_line()
```



Again, we see a clear pattern of decreasing mortality rates in this city over time. This means we need to think carefully about long-term time patterns as a potential confounder. It will be particularly important to think about this if the exposure also has a strong pattern over time. For example, air pollution regulations have meant that, in many cities, there may be long-term decreases in pollution concentrations over a study period.

5. **Is the outcome associated with day of week? Is the exposure associated with day of week?**

The data already has day of week as a column in the data (`dow`). However, this is in a character data type, so it doesn't have the order of weekdays encoded (e.g., Monday comes before Tuesday). This makes it hard to look for patterns related to things like weekend / weekday.

```
class(obs$dow)
```

```
## [1] "character"
```

We could convert this to a factor and encode the weekday order when we do it, but it's even easier to just recreate the column from the `date` column. We used the `wday` function from the `lubridate` package to do this—it extracts weekday as a factor, with the order of weekdays encoded (using a special "ordered" factor type):

```
library(lubridate)
obs <- obs %>%
  mutate(dow = wday(date, label = TRUE))
```

```
class(obs$dow)
```

```
## [1] "ordered" "factor"
```

```
levels(obs$dow)
```

```
## [1] "Sun" "Mon" "Tue" "Wed" "Thu" "Fri" "Sat"
```

We looked at the mean, median, and 25th and 75th quantiles of the mortality counts by day of week:

```
obs %>%
  group_by(dow) %>%
  summarize(mean(all),
            median(all),
            quantile(all, 0.25),
            quantile(all, 0.75))
```

```
## # A tibble: 7 x 5
##   dow   `mean(all)` `median(all)` `quantile(all, 0.25)` `quantile(all, 0.75)`
## * <ord>       <dbl>         <dbl>                 <dbl>                 <dbl>
## 1 Sun          156.           154                   136                   173
## 2 Mon          161.           159                   138                   179
## 3 Tue          161.           158                   139                   179
## 4 Wed          160.           157                   138.                  179
## 5 Thu          161.           158                   139                   179
## 6 Fri          162.           159                   141                   179
## 7 Sat          159.           156                   137                   178
```

Mortality tends to be a bit higher on weekdays than weekends, but it's not a dramatic difference.

We did the same check for temperature:

```
obs %>%
  group_by(dow) %>%
  summarize(mean(tmean),
            median(tmean),
            quantile(tmean, 0.25),
            quantile(tmean, 0.75))
```

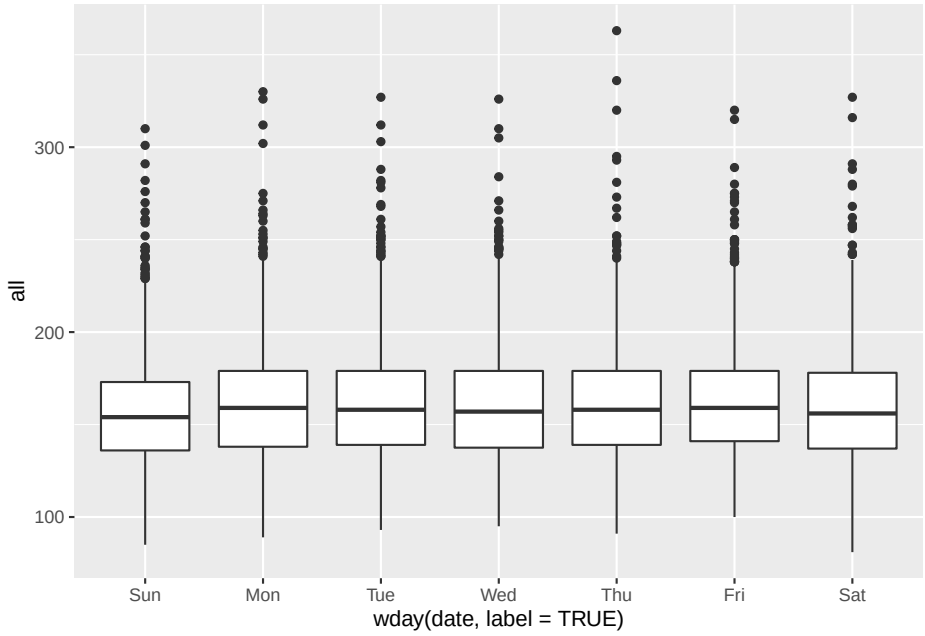
```
## # A tibble: 7 x 5
##   dow   `mean(tmean)` `median(tmean)` `quantile(tmean, 0.2~ `quantile(tmean, 0.~
## * <ord>         <dbl>           <dbl>                 <dbl>                 <dbl>
## 1 Sun            11.6            11.3                  7.48                  15.9
## 2 Mon            11.6            11.4                  7.33                  15.8
## 3 Tue            11.5            11.4                  7.48                  15.9
## 4 Wed            11.7            11.7                  7.64                  16.0
## 5 Thu            11.6            11.5                  7.57                  16.0
```

```
## 6 Fri          11.6          11.6          7.41          15.8
## 7 Sat          11.6          11.5          7.53          15.9
```

In this case, there does not seem to be much of a pattern by weekday.

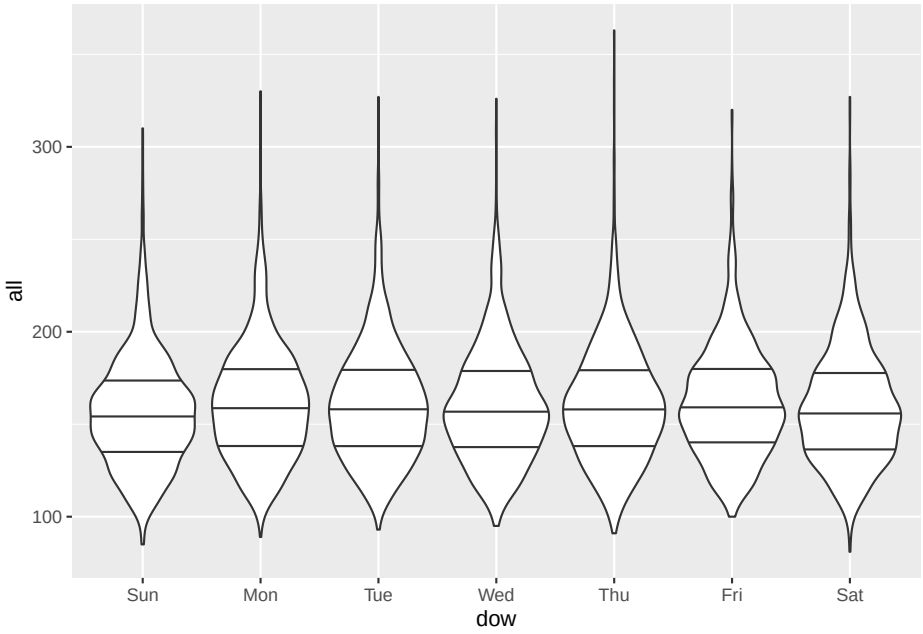
You can also visualize the association using boxplots:

```
ggplot(obs, aes(x = wday(date, label = TRUE), y = all)) +
  geom_boxplot()
```



You can also try violin plots—these show the full distribution better than boxplots, which only show quantiles.

```
ggplot(obs, aes(x = dow, y = all)) +
  geom_violin(draw_quantiles = c(0.25, 0.5, 0.75))
```



All these reinforce that there are some small differences in weekend versus weekday patterns for mortality. There isn't much pattern by weekday with temperature, so in this case weekday is unlikely to be a confounder (the same is not true with air pollution, which often varies based on commuting patterns and so can have stronger weekend/weekday differences). However, since it does help some in explaining variation in the health outcome, it might be worth including in our models anyway, to help reduce random noise.

Exploratory data analysis is an excellent tool for exploring your data before you begin fitting a statistical model, and you should get in the habit of using it regularly in your research. Dominici and Peng (2008a) provides another walkthrough of exploring this type of data, including some more advanced tools for exploring autocorrelation and time patterns.

3.5 Statistical modeling for a time series study

Now that we've explored the data typical of a time series study in climate epidemiology, we'll look at how we can fit a statistical model to those data to gain insight into the relationship between the exposure and acute health effects. Very broadly, we'll be using a statistical model to answer the question: How does the relative risk of a health outcome change as the level of the exposure changes, after controlling for potential confounders?

In the rest of this chapter and the next chapter, we'll move step-by-step to build up to the statistical models that are now typically used in these studies. Along the way, we'll discuss key components and choices in this modeling process. The

statistical modeling is based heavily on regression modeling, and specifically generalized linear regression. To help you get the most of this section, you may find it helpful to review regression modeling and generalized linear models. Some resources for that include Dunn and Smyth (2018) and James et al. (2013) .

One of the readings for this week, Vicedo-Cabrera et al. (2019), includes a section on fitting exposure-response functions to describe the association between daily mean temperature and mortality risk. This article includes example code in its supplemental material, with code for fitting the model to these time series data in the file named "01EstimationERassociation.r". Please download that file and take a look at the code.

The model in the code may at first seem complex, but it is made up of a number of fairly straightforward pieces (although some may initially seem complex):

- The model framework is a *generalized linear model (GLM)*
- This GLM is fit assuming an *error distribution* and a *link function* appropriate for count data
- The GLM is fit assuming an *error distribution* that is also appropriate for data that may be *overdispersed*
- The model includes control for day of the week by including a *categorical variable*
- The model includes control for long-term and seasonal trends by including a *spline* (in this case, a *natural cubic spline*) for the day in the study
- The model fits a flexible, non-linear association between temperature and mortality risk, also using a spline
- The model fits a flexible non-linear association between temperature on a series of preceeding days and current day and mortality risk on the current day using a *distributed lag approach*
- The model jointly describes both of the two previous non-linear associations by fitting these two elements through one construct in the GLM, a *cross-basis term*

In this section and the next chapter, we will work through the elements, building up the code to get to the full model that is fit in Vicedo-Cabrera et al. (2019).

Fitting a GLM to time series data

The generalized linear model (GLM) framework unites a number of types of regression models you may have previously worked with. One basic regression model that can be fit within this framework is a linear regression model. However, the framework also allows you to also fit, among others, logistic regression models (useful when the outcome variable can only take one of two values, e.g., success / failure or alive / dead) and Poisson regression models (useful when the outcome variable is a count or rate). This generalized framework brings some unity to these different types of regression models. From a practical standpoint, it has allowed software developers to easily provide a common interface to fit these types of models. In R, the common function call to fit GLMs is `glm`.

Within the GLM framework, the elements that separate different regression models include the link function and the error distribution. The error distribution encodes the assumption you are enforcing about how the errors after fitting the model are distributed. If the outcome data are normally distributed (a.k.a., follow a Gaussian distribution), after accounting for variance explained in the outcome by any of the model covariates, then a linear regression model may be appropriate. For count data—like numbers of deaths a day—this is unlikely, unless the average daily mortality count is very high (count data tend to come closer to a normal distribution the further their average gets from 0). For binary data—like whether each person in a study population died on a given day or not—normally distributed errors are also unlikely. Instead, in these two cases, it is typically more appropriate to fit GLMs with Poisson and binomial "families", respectively, where the family designation includes an appropriate specification for the variance when fitting the model based on these outcome types.

The other element that distinguishes different types of regression within the GLM framework is the link function. The link function applies a transformation on the combination of independent variables in the regression equation when fitting the model. With normally distributed data, an *identity link* is often appropriate—with this link, the combination of independent variables remain unchanged (i.e., keep their initial "identity"). With count data, a *log link* is often more appropriate, while with binomial data, a *logit link* is often used.

Finally, data will often not perfectly adhere to assumptions. For example, the Poisson family of GLMs assumes that variance follows a Poisson distribution (The probability mass function for Poisson distribution $X \sim \mathsf{Poisson}(\mu)$ is denoted by $f(k;\mu) = Pr[X = k] = \frac{\mu^k e^{-\mu}}{k!}$, where k is the number of occurences, and μ is equal to the expected number of cases). With this distribution, the variance is equal to the mean ($\mu = E(X) = Var(X)$). With real-life data, this assumption is often not valid, and in many cases the variance in real life count data is larger than the mean. This can be accounted for when fitting a GLM by setting an error distribution that does not require the variance to equal the mean—instead, both a mean value and something like a variance are estimated from the data, assuming an overdispersion parameter ϕ so that $Var(X) = \phi E(X)$. In environmental epidemiology, time series are often fit to allow for this overdispersion. This is because if the data are overdispersed but the model does not account for this, the standard errors on the estimates of the model parameters may be artificially small. If the data are not overdispersed ($\phi = 1$), the model will identify this when being fit to the data, so it is typically better to prefer to allow for overdispersion in the model (if the size of the data were small, you may want to be parsimonious and avoid unneeded complexity in the model, but this is typically not the case with time series data).

In the next section, you will work through the steps of developing a GLM to fit the example dataset `obs`. For now, you will only fit a linear association between mean daily temperature and mortality risk, eventually including control for day

of week. In later work, especially the next chapter, we will build up other components of the model, including control for the potential confounders of long-term and seasonal patterns, as well as advancing the model to fit non-linear associations, distributed by time, through splines, a distributed lag approach, and a cross-basis term.

Applied: Fitting a GLM to time series data

In R, the function call used to fit GLMs is `glm`. Most of you have likely covered GLMs, and ideally this function call, in previous courses. If you are unfamiliar with its basic use, you will want to refresh yourself on this topic—you can use some of the resources noted earlier in this section and in the chapter's "Supplemental Readings" to do so.

1. Fit a GLM to estimate the association between mean daily temperature (as the independent variable) and daily mortality count (as the dependent variable), first fitting a linear regression. (Since the mortality data are counts, we will want to shift to a different type of regression within the GLM framework, but this step allows you to develop a simple `glm` call, and to remember where to include the data and the independent and dependent variables within this function call.)
2. Change your function call to fit a regression model in the Poisson family.
3. Change your function call to allow for overdispersion in the outcome data (daily mortality count). How does the estimated coefficient for temperature change between the model fit for #2 and this model? Check both the central estimate and its estimated standard error.
4. Change your function call to include control for day of week.

Applied exercise: Example code

1. **Fit a GLM to estimate the association between mean daily temperature (as the independent variable) and daily mortality count (as the dependent variable), first fitting a linear regression.**

This is the model you are fitting:

$Y_t = \beta_0 + \beta_1 X1_t + \epsilon$

where Y_t is the mortality count on day t, $X1_t$ is the mean temperature for day t and ϵ is the error term. Since this is a linear model we are assuming a Gaussian error distribution $\epsilon \sim \mathsf{N}(0, \sigma^2)$, where σ^2 is the variance not explained by the covariates (here just temperature).

To do this, you will use the `glm` call. If you would like to save model fit results to use later, you assign the output a name as an R object (`mod_linear_reg` in the example code). If your study data are in a dataframe, you can specify these data in the `glm` call with the `data` parameter. Once you do this, you can use column names directly in the model formula. In the model formula, the dependent variable is specified first (`all`, the column for daily mortality counts for all ages, in this example), followed by a tilde (`~`), followed by all independent

variables (only `tmean` in this example). If multiple independent variables are included, they are joined using `+`. We'll see an example when we start adding control for confounders later.

```
mod_linear_reg <- glm(all ~ tmean, data = obs)
```

Once you have fit a model and assigned it to an R object, you can explore it and use resulting values. First, the `print` method for a regression model gives some summary information. This method is automatically called if you enter the model object's name at the console:

```
mod_linear_reg
```

```
##
## Call:  glm(formula = all ~ tmean, data = obs)
##
## Coefficients:
## (Intercept)        tmean
##     187.647       -2.366
##
## Degrees of Freedom: 8278 Total (i.e. Null);  8277 Residual
## Null Deviance:       8161000
## Residual Deviance: 6766000   AIC: 79020
```

More information is printed if you run the `summary` method on the model object:

```
summary(mod_linear_reg)
```

```
##
## Call:
## glm(formula = all ~ tmean, data = obs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -77.301  -20.365   -1.605   17.502  169.280
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 187.64658    0.73557  255.10   <2e-16 ***
## tmean        -2.36555    0.05726  -41.31   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 817.4629)
##
##     Null deviance: 8161196  on 8278  degrees of freedom
## Residual deviance: 6766140  on 8277  degrees of freedom
## AIC: 79019
```

```
##
## Number of Fisher Scoring iterations: 2
```

Make sure you are familiar with the information provided from the model object, as well as how to interpret values like the coefficient estimates and their standard errors and p-values. These basic elements should have been covered in previous coursework (even if a different programming language was used to fit the model), and so we will not be covering them in great depth here, but instead focusing on some of the more advanced elements of how regression models are commonly fit to data from time series and case-crossover study designs in environmental epidemiology. For a refresher on the basics of fitting statistical models in R, you may want to check out Chapters 22 through 24 of Wickham and Grolemund (2016), a book that is available online, as well as Dunn and Smyth (2018) and James et al. (2013) .

Finally, there are some newer tools for extracting information from model fit objects. The `broom` package extracts different elements from these objects and returns them in a "tidy" data format, which makes it much easier to use the output further in analysis with functions from the "tidyverse" suite of R packages. These tools are very popular and powerful, and so the `broom` tools can be very useful in working with output from regression modeling in R.

The `broom` package includes three main functions for extracting data from regression model objects. First, the `glance` function returns overall data about the model fit, including the AIC and BIC:

```
library(broom)
glance(mod_linear_reg)
```

```
## # A tibble: 1 x 8
##   null.deviance df.null  logLik    AIC    BIC deviance df.residual  nobs
##           <dbl>   <int>   <dbl>  <dbl>  <dbl>    <dbl>       <int> <int>
## 1      8161196.    8278 -39507. 79019. 79041. 6766140.        8277  8279
```

The `tidy` function returns data at the level of the model coefficients, including the estimate for each model parameter, its standard error, test statistic, and p-value.

```
tidy(mod_linear_reg)
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic p.value
##   <chr>          <dbl>     <dbl>     <dbl>   <dbl>
## 1 (Intercept)   188.      0.736      255.        0
## 2 tmean          -2.37    0.0573     -41.3       0
```

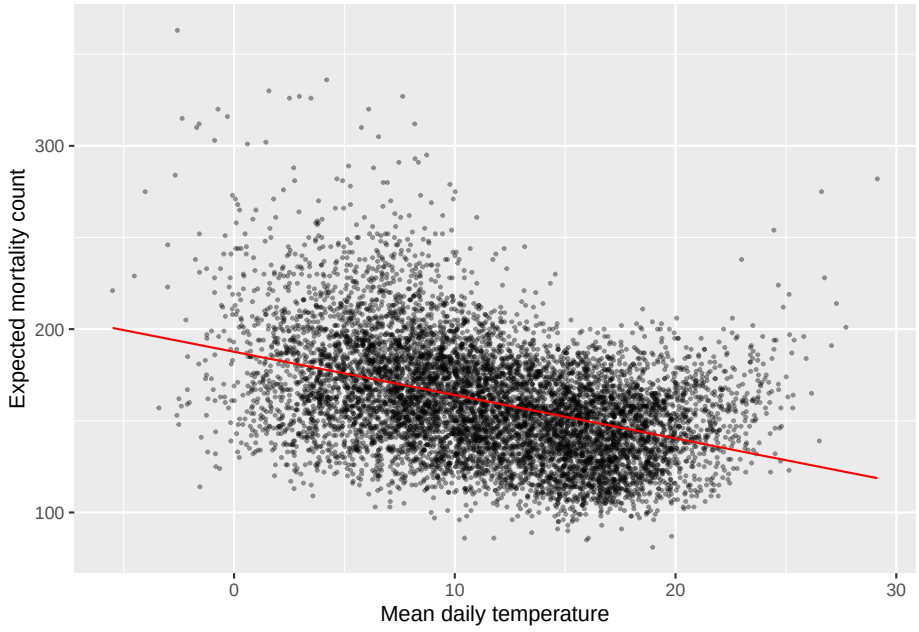
Finally, the `augment` function returns data at the level of the original observations, including the fitted value for each observation, the residual between the fitted and true value, and some measures of influence on the model fit.

```
augment(mod_linear_reg)
```

```
## # A tibble: 8,279 x 8
##      all tmean .fitted .resid .std.resid     .hat .sigma  .cooksd
##    <dbl> <dbl>   <dbl>  <dbl>      <dbl>    <dbl>  <dbl>    <dbl>
##  1   220  3.91    178.   41.6       1.46 0.000359   28.6 0.000380
##  2   257  5.55    175.   82.5       2.89 0.000268   28.6 0.00112
##  3   245  4.39    177.   67.7       2.37 0.000330   28.6 0.000928
##  4   226  5.43    175.   51.2       1.79 0.000274   28.6 0.000440
##  5   236  6.87    171.   64.6       2.26 0.000211   28.6 0.000539
##  6   235  9.23    166.   69.2       2.42 0.000144   28.6 0.000420
##  7   231  6.69    172.   59.2       2.07 0.000218   28.6 0.000467
##  8   235  7.96    169.   66.2       2.31 0.000174   28.6 0.000467
##  9   250  7.27    170.   79.5       2.78 0.000197   28.6 0.000761
## 10   214  9.51    165.   48.9       1.71 0.000139   28.6 0.000202
## # ... with 8,269 more rows
```

One way you can use `augment` is to graph the fitted values for each observation after fitting the model:

```
mod_linear_reg %>%
  augment() %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = .fitted), color = "red") +
  labs(x = "Mean daily temperature", y = "Expected mortality count")
```



For more on the `broom` package, including some excellent examples of how it can be used to streamline complex regression analyses, see Robinson (2014). There is also a nice example of how it can be used in one of the chapters of Wickham and Grolemund (2016), available online at https://r4ds.had.co.nz/many-models.html.

2. **Change your function call to fit a regression model in the Poisson family.**

A linear regression is often not appropriate when fitting a model where the outcome variable provides counts, as with the example data, since such data often don't follow a normal distribution. A Poisson regression is typically preferred.

For a count distribution were $Y \sim$ Poisson( ) we typically fit a model such as

$g(Y) = \beta_0 + \beta_1 X1$, where $g()$ represents the link function, in this case a log function so that $log(Y) = \beta_0 + \beta_1 X1$. We can also express this as $Y = exp(\beta_0 + \beta_1 X1)$.

In the `glm` call, you can specify this with the `family` parameter, for which "poisson" is one choice.

```
mod_pois_reg <- glm(all ~ tmean, data = obs, family = "poisson")
```

One thing to keep in mind with this change is that the model now uses a non-identity link between the combination of independent variable(s) and the dependent variable. You will need to keep this in mind when you interpret the estimates of the regression coefficients. While the coefficient estimate for `tmean` from the linear regression could be interpreted as the expected increase in mortality counts for a one-unit (i.e., one degree Celsius) increase in temperature, now the estimated coefficient should be interpreted as the expected increase in the natural log-transform of mortality count for a one-unit increase in temperature.

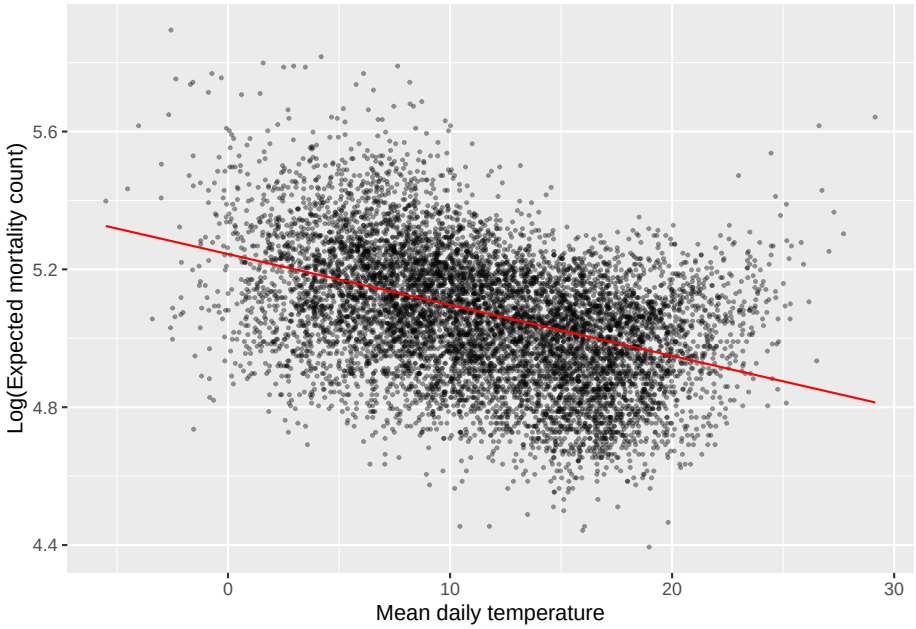
```
summary(mod_pois_reg)
```

```
##
## Call:
## glm(formula = all ~ tmean, family = "poisson", data = obs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.5945  -1.6365  -0.1167   1.3652  12.2221
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.2445409  0.0019704 2661.67   <2e-16 ***
## tmean       -0.0147728  0.0001583  -93.29   <2e-16 ***
## ---
```

```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for poisson family taken to be 1)
##
##     Null deviance: 49297  on 8278  degrees of freedom
## Residual deviance: 40587  on 8277  degrees of freedom
## AIC: 97690
##
## Number of Fisher Scoring iterations: 4
```

You can see this even more clearly if you take a look at the association between temperature for each observation and the expected mortality count fit by the model. First, if you look at the fitted values without transforming, they will still be in a state where mortality count is log-transformed. You can see by looking at the range of the y-scale that these values are for the log of expected mortality, rather than expected mortality (compare, for example, to the similar plot shown from the first model, which was linear), and that the fitted association for that *transformation*, not for untransformed mortality counts, is linear:

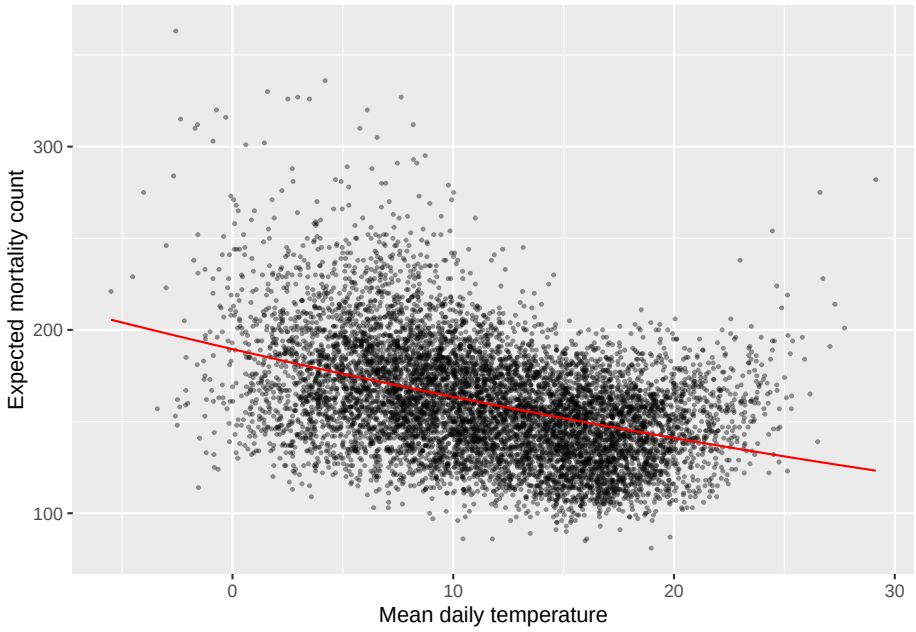
```
mod_pois_reg %>%
  augment() %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = log(all)), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = .fitted), color = "red") +
  labs(x = "Mean daily temperature", y = "Log(Expected mortality count)")
```



You can use exponentiation to transform the fitted values back to just be the

expected mortality count based on the model fit. Once you make this transformation, you can see how the link in the Poisson family specification enforced a curved relationship between mean daily temperature and the untransformed expected mortality count.

```
mod_pois_reg %>%
  augment() %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Mean daily temperature", y = "Expected mortality count")
```



For this model, we can interpret the coefficient for the temperature covariate as the expected log relative risk in the health outcome associated with a one-unit increase in temperature. We can exponentiate this value to get an estimate of the relative risk:

```
# Extract the temperature coefficient from the model
tmean_coef <- mod_pois_reg %>%
  tidy() %>%
  filter(term == "tmean") %>% # Extract the row with the tmean estimates
  pull(estimate) # Extract just the point estimate

# Estimate of the log relative risk for a one-unit increase in temperature
tmean_coef
```

```
## [1] -0.0147728
```

```
# Estimate of the relative risk for a one-unit increase in temperature
exp(tmean_coef)
```

```
## [1] 0.9853358
```

If you want to estimate the confidence interval for this estimate, you should calculate that *before* exponentiating.

3. **Change your function call to allow for overdispersion in the outcome data (daily mortality count). How does the estimated coefficient for temperature change between the model fit for #2 and this model? Check both the central estimate and its estimated standard error.**

In the R `glm` call, there is a family that is similar to Poisson (including using a log link), but that allows for overdispersion. You can specify it with the "quasipoisson" choice for the `family` parameter in the `glm` call:

```
mod_ovdisp_reg <- glm(all ~ tmean, data = obs, family = "quasipoisson")
```

When you use this family, there will be some new information in the summary for the model object. It will now include a dispersion parameter (ϕ). If this is close to 1, then the data were close to the assumed variance for a Poisson distribution (i.e., there was little evidence of overdispersion). In the example, the overdispersion is around 5, suggesting the data are overdispersed (this might come down some when we start including independent variables that explain some of the variation in the outcome variable, like long-term and seasonal trends).

```
summary(mod_ovdisp_reg)
```

```
##
## Call:
## glm(formula = all ~ tmean, family = "quasipoisson", data = obs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.5945  -1.6365  -0.1167   1.3652  12.2221
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  5.2445409  0.0044087  1189.6   <2e-16 ***
## tmean       -0.0147728  0.0003543   -41.7   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 5.006304)
##
##     Null deviance: 49297  on 8278  degrees of freedom
```

```
## Residual deviance: 40587  on 8277  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

If you compare the estimates of the temperature coefficient from the Poisson regression with those when you allow for overdispersion, you'll see something interesting:

```
tidy(mod_pois_reg) %>%
  filter(term == "tmean")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 tmean  -0.0148  0.000158     -93.3       0
```

```
tidy(mod_ovdisp_reg) %>%
  filter(term == "tmean")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 tmean  -0.0148  0.000354     -41.7       0
```

The central estimate (`estimate` column) is very similar. However, the estimated standard error is larger when the model allows for overdispersion. This indicates that the Poisson model was too simple, and that its inherent assumption that data were not overdispersed was problematic. If you naively used a Poisson regression in this case, then you would estimate a confidence interval on the temperature coefficient that would be too narrow. This could cause you to conclude that the estimate was statistically significant when you should not have (although in this case, the estimate is statistically significant under both models).

4. **Change your function call to include control for day of week.**

Day of week is included in the data as a categorical variable, using a data type in R called a factor. You are now essentially fitting this model:

$log(Y) = \beta_0 + \beta_1 X1 + \gamma^{'} X2,$

where $X2$ is a categorical variable for day of the week and $\gamma^{'}$ represents a vector of parameters associated with each category.

It is pretty straightforward to include factors as independent variables in calls to `glm`: you just add the column name to the list of other independent variables with a `+`. In this case, we need to do one more step: earlier, we added order to `dow`, so it would "remember" the order of the week days (Monday before Tuesday, etc.). However, we need to strip off this order before we include the

factor in the `glm` call. One way to do this is with the `factor` call, specifying `ordered = FALSE`. Here is the full call to fit this model:

```
mod_ctrl_dow <- glm(all ~ tmean + factor(dow, ordered = FALSE),
                    data = obs, family = "quasipoisson")
```

When you look at the summary for the model object, you can see that the model has fit a separate model parameter for six of the seven weekdays. The one weekday that isn't fit (Sunday in this case) serves as a baseline —these estimates specify how the log of the expected mortality count is expected to differ on, for example, Monday versus Sunday (by about 0.03), if the temperature is the same for the two days.

```
summary(mod_ctrl_dow)
```

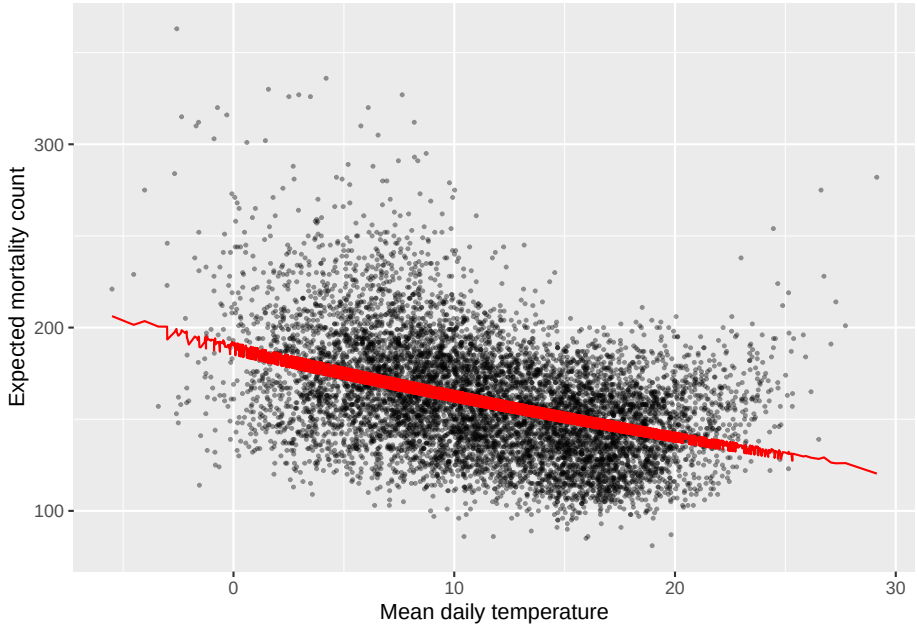
```
##
## Call:
## glm(formula = all ~ tmean + factor(dow, ordered = FALSE), family = "quasipoisson",
##     data = obs)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.3211  -1.6476  -0.1313   1.3549  12.5286
##
## Coefficients:
##                                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)                     5.2208502  0.0065277 799.804  < 2e-16 ***
## tmean                          -0.0147723  0.0003538 -41.750  < 2e-16 ***
## factor(dow, ordered = FALSE)Mon  0.0299282  0.0072910   4.105 4.08e-05 ***
## factor(dow, ordered = FALSE)Tue  0.0292575  0.0072920   4.012 6.07e-05 ***
## factor(dow, ordered = FALSE)Wed  0.0255224  0.0073020   3.495 0.000476 ***
## factor(dow, ordered = FALSE)Thu  0.0269580  0.0072985   3.694 0.000222 ***
## factor(dow, ordered = FALSE)Fri  0.0355431  0.0072834   4.880 1.08e-06 ***
## factor(dow, ordered = FALSE)Sat  0.0181489  0.0073158   2.481 0.013129 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for quasipoisson family taken to be 4.992004)
##
##     Null deviance: 49297  on 8278  degrees of freedom
## Residual deviance: 40434  on 8271  degrees of freedom
## AIC: NA
##
## Number of Fisher Scoring iterations: 4
```

You can also see from this summary that the coefficients for the day of the week are all statistically significant. Even though we didn't see a big difference in

mortality counts by day of week in our exploratory analysis, this suggests that it does help explain some variance in mortality observations and will likely be worth including in the final model.

The model now includes day of week when fitting an expected mortality count for each observation. As a result, if you plot fitted values of expected mortality versus mean daily temperature, you'll see some "hoppiness" in the fitted line:

```
mod_ctrl_dow %>%
  augment() %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Mean daily temperature", y = "Expected mortality count")
```

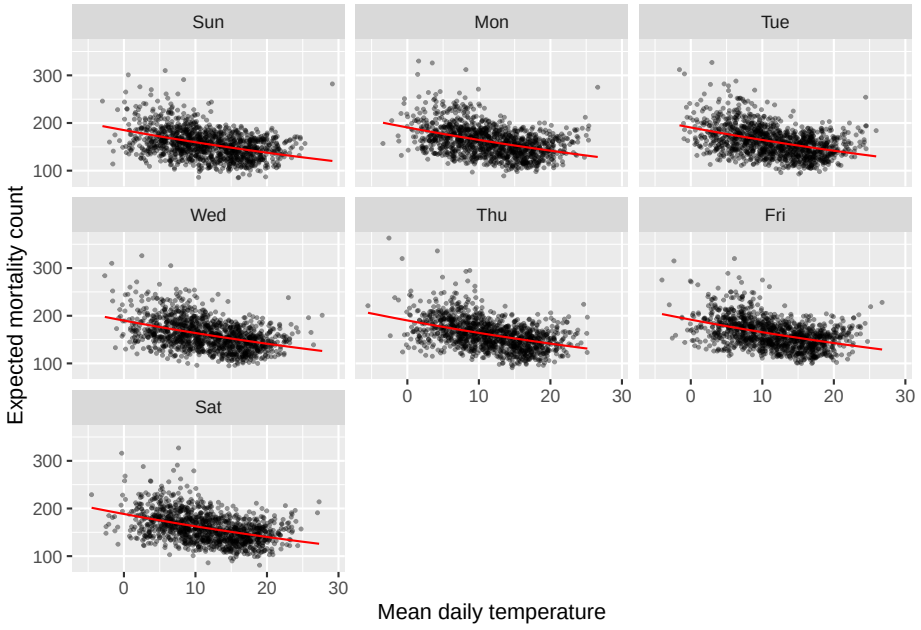


This is because each fitted value is also incorporating the expected influence of day of week on the mortality count, and that varies across the observations (i.e., you could have two days with the same temperature, but different expected mortality from the model, because they occur on different days).

If you plot the model fits separately for each day of the week, you'll see that the line is smooth across all observations from the same day of the week:

```
mod_ctrl_dow %>%
  augment() %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
```

```
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Mean daily temperature", y = "Expected mortality count") +
  facet_wrap(~ obs$dow)
```



Wrapping up

At this point, the coefficient estimates suggests that risk of mortality tends to decrease as temperature increases. Do you think this is reasonable? What else might be important to build into the model based on your analysis up to this point?

Chapter 4

Generalized linear models

4.1 Readings

The readings for this chapter are:

- Bhaskaran et al. (2013) Provides an overview of time series regression in environmental epidemiology.
- Vicedo-Cabrera et al. (2019) Provides a tutorial of all the steps for a projecting of health impacts of temperature extremes under climate change. One of the steps is to fit the exposure-response association using present-day data (the section on "Estimation of Exposure-Response Associations" in the paper). In this chapter, we will go into details on that step, and that section of the paper is the only required reading for this chapter. Later in the class, we'll look at other steps covered in this paper. Supplemental material for this paper is available to download by clicking http://links.lww.com/EDE/B504. You will need the data in this supplement for the exercises for class.
- Armstrong et al. (2014) This paper describes different data structures for case-crossover data, as well as how conditional Poisson regression can be used in some cases to fit a statistical model to these data. Supplemental material for this paper is available at https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/1471-2288-14-122#Sec13.

The following are supplemental readings (i.e., not required, but may be of interest) associated with the material in this chapter:

- Armstrong (2006) Covers similar material as Bhaskaran et al. (2013), but with more focus on the statistical modeling framework (highly recommended!)
- Gasparrini and Armstrong (2010) Describes some of the advances made to time series study designs and statistical analysis, specifically in the context

of temperature

- Basu et al. (2005) Compares time series and case-crossover study designs in the context of exploring temperature and health. Includes a nice illustration of different referent periods, including time-stratified.
- Imai et al. (2015) Typically, the time series study design covered in this chapter is used to study non-communicable health outcomes. This paper discusses opportunities and limitations in applying a similar framework for infectious disease.
- Lu and Zeger (2007) Heavier on statistics. This paper shows how, under conditions often common for environmental epidemiology studies, case-crossover and time series methods are equivalent.
- Gasparrini (2014) Heavier on statistics. This provides the statistical framework for the distributed lag model for environmental epidemiology time series studies.
- Dunn and Smyth (2018) Introduction to statistical models, moving into regression models and generalized linear models. Chapter in a book that is available online through the CSU library.
- James et al. (2013) General overview of linear regression, with an R coding "lab" at the end to provide coding examples. Covers model fit, continuous, binary, and categorical covariates, and interaction terms. Chapter in a book that is available online through the CSU library.

4.2 Splines in GLMs

We saw from the last model, with a linear term for mean daily temperature, that the suggested effect on mortality is a decrease in daily mortality counts with increasing temperature. However, as you've probably guessed that's likely not entirely accurate. A linear term for the effect of exposure restricts us to an effect that can be fitted with a straight line (either a null effect or a monotonically increasing or decreasing effect with increasing exposure).

This clearly is problematic in some cases. One example is when exploring the association between temperature and health risk. Based on human physiology, we would expect many health risks to be elevated at temperature extremes, whether those are extreme cold or extreme heat. A linear term would be inadequate to describe this kind of U-shaped association. Other effects might have a threshold—for example, heat stroke might have a very low risk at most temperatures, only increasing with temperature above a certain threshold.

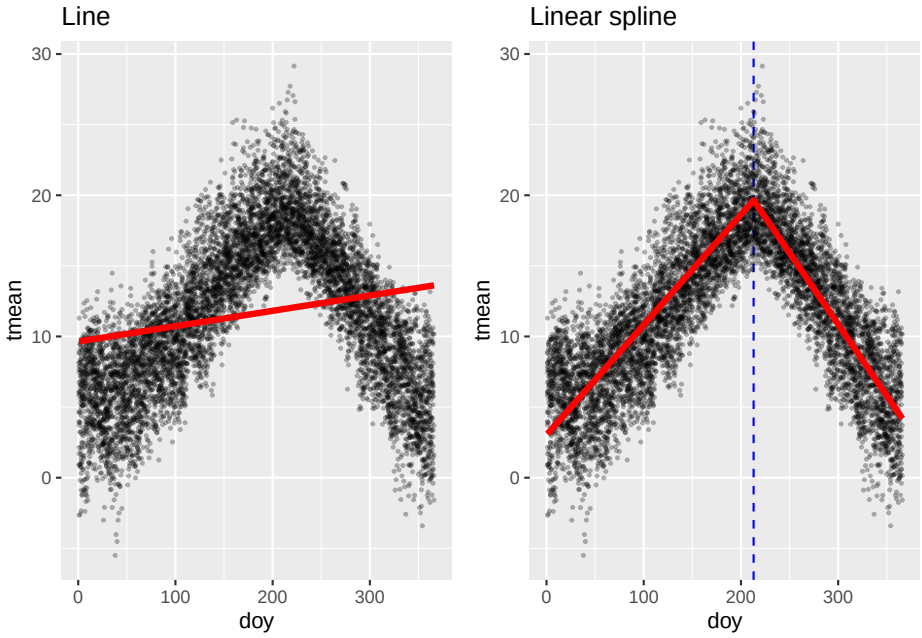
We can capture non-linear patterns in effects, by using different functions of X. Examples are $\sqrt{X}$, X^2, or more complex smoothing functions, such as polynomials or splines. Polynomials might at first make a lot of sense, especially since you've likely come across polynomial terms in mathematics classes since grade school. However, it turns out that they have some undesirable properties. A key one is that they can have extreme behavior, particularly when using a high-order polynomial, and particularly outside the range of data that are available

to fit the model.

An alternative that is generally preferred for environmental epidemiology studies is the regression spline. The word "spline" originally comes from drafting and engineering (ship building, in particular), where it described a flexible piece of wood or metal that you could use to draw a curved line—it created a curve that was flexible enough—but just flexible enough—to fit a space (see Wikipedia's very interesting article on flat splines for more).

Splines follow a similar idea in mathematics, making them helpful tools when a line won't fit your data well. In general, a spline fits together a few simpler functions to create something with a curve or non-linearity. Each simpler function operates within an interval of the data, and then they join together at "knots" along the range of the data. Regression splines are therefore simple parametric smoothing function, which fit separate polynomial in each interval of the range of the predictor; these can be linear, quadratic, and cubic.

The simplest example is a linear spline (also called a piecewise linear function). This type of spline creates non-linearity by having a breakpoint at the knot, allowing the slope of the line to be different in the intervals of data on either side of the knot. The following plot shows an example. Say you want to explore how mean temperature varies by the day in the year (Jan 1 = 1, Jan 2 = 2, and so on) in the London example dataset from the last chapter. Temperature tends to increase with day of year for a while, but then it changes around the start of August, after that decreasing with day of year. This patterns means that a line will give a bad fit for how temperature changes with day of year, since it smooths right through that change. On the other hand, you can get a very reasonable fit using a linear spline with a knot around August 1 (day 213 in the year). These two examples are shown in the following plot, with the linear function fit on the left and the linear spline on the right:



If you were to write out these regression models in mathematical notation, the linear one is very simple:

$$Y_t = \alpha + \beta X_t$$

where Y_t is the temperature on day t, α is the model intercept, X_t is the day of the year on day t, and β is the estimated coefficient for X_t.

The notation for the model with the linear spline is a bit more complex:

$$Y_t = \alpha + \beta_1 X_t + \beta_2 (X_t - k)_+$$

Here, Y_t is again the temperature on day t, X_t is again the day of the year on day t, and α is again the intercept. The term k is the "knot"—the value of X where we're letting the slope change. In this example, we're using $k = 213$. The term $(X_t - k)_+$ has a special meaning—it takes the value 0 if X_t is in the interval to the left of the knot, while if X_t is in the interval to the right of the knot, it takes the value of X_t minus the knot value:

$$(X_t - k)_+ = \begin{cases} 0, & \text{if } X_t < k \\ X_t - k, & \text{if } X_t \geq k \end{cases}$$

In this model, the coefficient β_1 estimates the slope of the line to the left of the knot, while β_2 estimates how that slope will change to the right of the knot.

Fortunately, we usually won't have to get this complex in the model notation, especially when we use more complex splines (where the notation would get even more complex). Instead, we'll often write out the regression equation in a simpler way, just indicating that we're using a function of the covariate, rather than the covariate directly:

$$Y_t = \alpha + f(X_t| \ )$$

where we can note that $f(X_t)$ is a function of day of the year (X_t), fit in this case using a linear spline, and with a set of estimated coefficients for that function (see Armstrong (2006) for an example of using this model notation).

While a linear spline is the simplest conceptually (and mathematically), it often isn't very satisfying, because it fits a function with a sharp breakpoint, which often isn't realistic. For example, the linear spline fit above suggests that the relationship between day of the year and temperature changes abruptly and dramatically on August 1 of the year. In reality, we know that this change in the relationship between day of year and temperature is probably a lot smoother.

To fit smoother shapes, we can move to higher level splines. Cubic splines ("cubic" because they include terms of the covariate up to the third power) are very popular. An example of a cubic spline function is $X + X^2 + X^3 + I((X > X_0) * (X - X_0)^3)$. This particular function is a cubic spline with four degrees of freedom ($df = 4$) and one knot (X_0). A special type of cubic spline called a natural cubic spline is particularly popular. Unlike a polynomial function, a natural cubic spline "behaves" better outside the range of the data used to fit the model—they are constrained to continue on a linear trajectory once they pass beyond the range of the data.

Regression splines can be fit in a GLM via the package `splines`. Two commonly used examples of regression splines are b-splines and natural cubic splines. Vicedo-Cabrera et al. (2019) uses natural cubic splines, which can be fit with the `ns` (for "natural spline") function from the `splines` package.

While splines are great for fitting non-linear relationships, they do create some challenges in interpreting the results. When you fit a linear relationship for a covariate, you will get a single estimate that helps describe the fitted relationship between that covariate and the outcome variable. However, when you use a non-linear function, you'll end up with a mix of coefficients associated with that function. Sometimes, you will use splines to control for a potential confounder (as we will in the exercises for this first part of the chapter). In this case, you don't need to worry about interpreting the estimated coefficients—you're just trying to control for the variable, rather than inferring anything about how it's associated with the outcome. In later parts of this chapter, we'll talk about how to interpret these coefficients if you're using a spline for the exposure that you're interested in, when we talk more broadly about basis functions.

Applied: Including a spline in a GLM

For this exercise, you will continue to build up the model that you began in the examples in the previous chapter. The example uses the data provided with one of this chapter's readings, Vicedo-Cabrera et al. (2019).

1. Start by fitting a somewhat simple model—how are daily mortality counts associated with (a) a linear and (b) a non-linear function of time? Is a linear term appropriate to describe this association? What types of patterns are captured by a non-linear function that are missed by a linear function?
2. In the last chapter, the final version of the model used a GLM with an overdispersed Poisson distribution, including control for day of week. Start from this model and add control for long-term and seasonal trends over the study period.
3. Refine your model to fit for a non-linear, rather than linear, function of temperature in the model. Does a non-linear term seem to be more appropriate than a linear term?

Applied exercise: Example code

1. **Start by fitting a somewhat simple model—how are daily mortality counts associated with (a) a linear and (b) a non-linear function of time?**

It is helpful to start by loading the R packages you are likely to need, as well as the example dataset. You may also need to re-load the example data and perform the steps taken to clean it in the last chapter:

```
# Load some packages that will likely be useful
library(tidyverse)
library(viridis)
library(lubridate)
library(broom)

# Load and clean the data
obs <- read_csv("data/lndn_obs.csv") %>%
  mutate(dow = wday(date, label = TRUE))
```

For this first question, the aim is to model the association between time and daily mortality counts within the example data. This approach is often used to explore and, if needed, adjust for temporal factors in the data.

There are a number of factors that can act over time to create patterns in both environmental exposures and health outcomes. For example, there may be changes in air pollution exposures over the years of a study because of changes in regulations or growth or decline of factories and automobile traffic in an area. Changes in health care and in population demographics can cause patterns in health outcomes over the study period. At a shorter, seasonal term, there are also factors that could influence both exposures and outcomes, including seasonal changes in climate, seasonal changes in emissions, and seasonal patterns

in health outcomes.

It can be difficult to pinpoint and measure these temporal factors, and so instead a common practice is to include model control based on the time in the study. This can be measured, for example, as the day since the start of the study period.

You can easily add a column for day in study for a dataset that includes date. R saves dates in a special format, which we're using the in `obs` dataset:

```
class(obs$date)
```

```
## [1] "Date"
```

However, this is just a fancy overlay on a value that's ultimately saved as a number. Like most Unix programs, the date is saved as the number of days since the Unix "epoch", January 1, 1970. You can take advantage of this convention—if you use `as.numeric` around a date in R, it will give you a number that gets one unit higher for every new date. Here's the example for the first date in our example data:

```
obs$date[1]
```

```
## [1] "1990-01-01"
```

```
as.numeric(obs$date[1])
```

```
## [1] 7305
```

And here's the example for the next date:

```
obs$date[2]
```

```
## [1] "1990-01-02"
```

```
as.numeric(obs$date[2])
```

```
## [1] 7306
```

You can use this convention to add a column that gives days since the first study date. While you could also use the `1:n()` call to get a number for each row that goes from 1 to the number of rows, that approach would not catch any "skips" in dates in the data (e.g., missing dates if only warm-season data are included). The use of the dates is more robust:

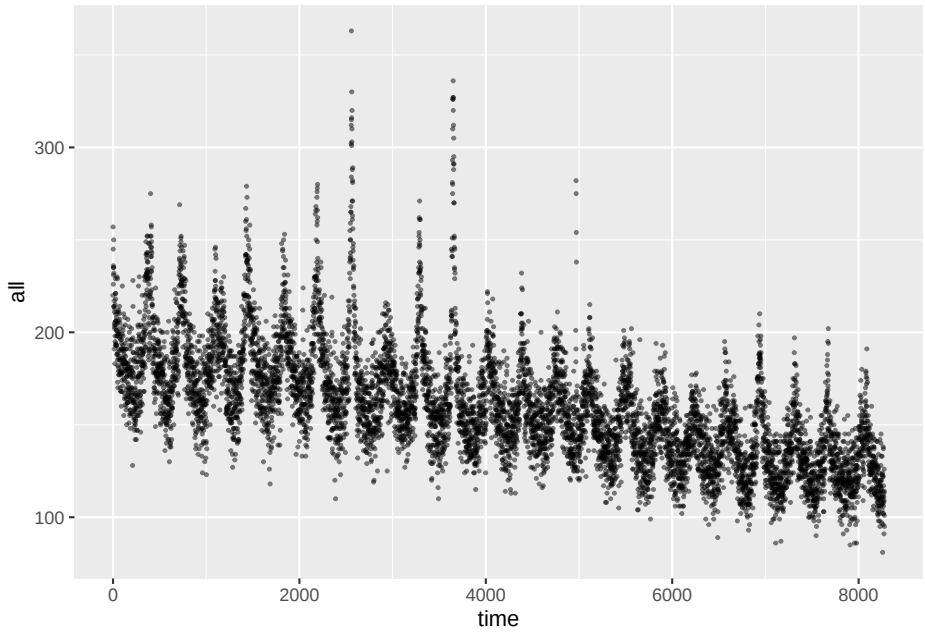
```
obs <- obs %>%
  mutate(time = as.numeric(date) - first(as.numeric(date)))

obs %>%
  select(date, time)
```

```
## # A tibble: 8,279 x 2
##    date        time
```

```
##    <date>     <dbl>
##  1 1990-01-01     0
##  2 1990-01-02     1
##  3 1990-01-03     2
##  4 1990-01-04     3
##  5 1990-01-05     4
##  6 1990-01-06     5
##  7 1990-01-07     6
##  8 1990-01-08     7
##  9 1990-01-09     8
## 10 1990-01-10     9
## # ... with 8,269 more rows
```

As a next step, it is always useful to use exploratory data analysis to look at the patterns that might exist for an association, before you start designing and fitting the regression model.

```
ggplot(obs,
       aes(x = time, y = all)) +
  geom_point(size = 0.5, alpha = 0.5)
```



There are clear patterns between time and daily mortality counts in these data. First, there is a clear long-term pattern, with mortality rates declining on average over time. Second, there are clear seasonal patterns, with higher mortality generally in the winter and lower rates in the summer.

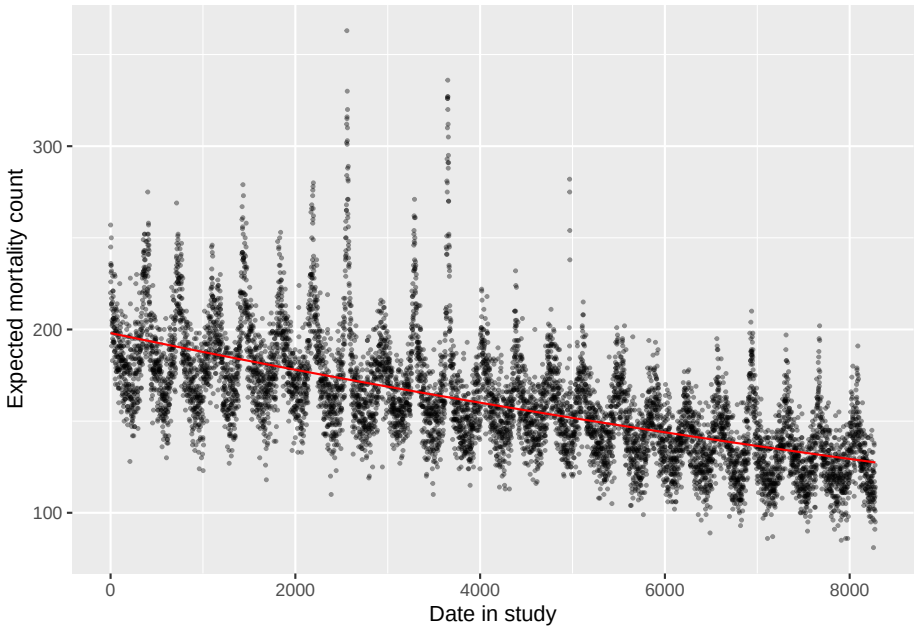
To model this, we can start with fitting a linear term. In the last chapter, we

determined that the mortality outcome data can be fit using a GLM with a Poisson family, allowing for overdispersion as it is common in real-life count data like these. To include time as a linear term, we can just include that column name to the right of the ~ in the model formula:

```
mod_time <- glm(all ~ time,
                data = obs, family = "quasipoisson")
```

You can use the `augment` function from the `broom` package to pull out the fitted estimate for each of the original observations and plot that, along with the observed data, to get an idea of what this model has captured:

```
mod_time %>%
  augment() %>%
  ggplot(aes(x = time)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Date in study", y = "Expected mortality count")
```



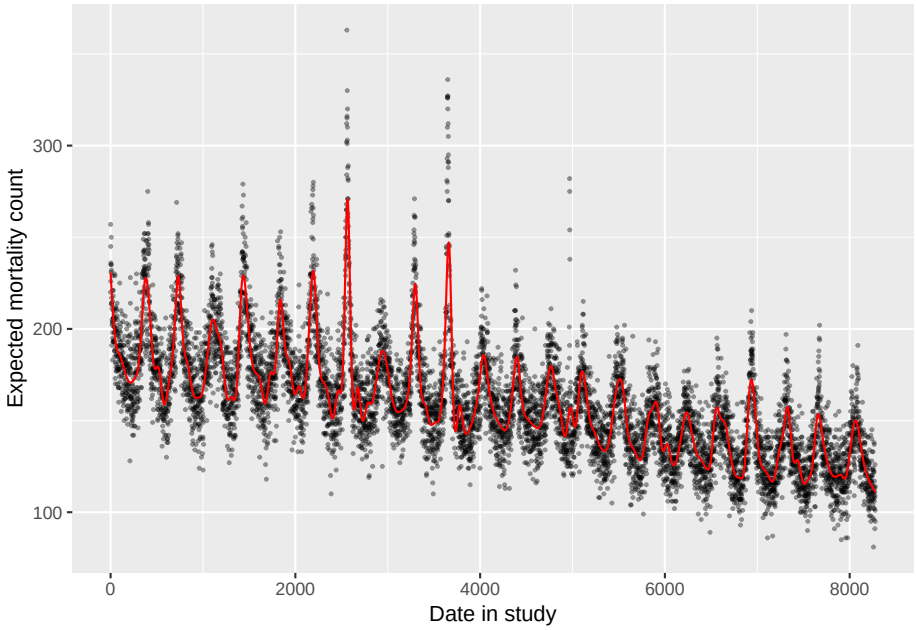
This linear trend captures the long-term trend in mortality rates fairly well in this case. This won't always be the case, as there may be some health outcomes—or some study populations—where the long-term pattern over the study period might be less linear than in this example. Further, the linear term is completely unsuccessful in capturing the shorter-term trends in mortality rate. These oscillate, and so would be impossible to capture over multiple years with a linear trend.

Instead, it's helpful to use a non-linear term for time in the model. We can use a natural cubic spline for this, using the `ns` function from the `splines` package. You will need to clarify how flexible the spline function should be, and this can be specified through the degrees of freedom for the spline. A spline with more degrees of freedom will be "wigglier" over a given data range compared to a spline with fewer degrees of freedom. Let's start by using 158 degrees of freedom, which translates to about 7 degrees of freedom per year:

```
library(splines)
mod_time_nonlin <- glm(all ~ ns(time, df = 158),
                       data = obs, family = "quasipoisson")
```

You can visualize the model results in a similar way to how we visualized the last model. However, there is one extra step. The `augment` function only carries through columns in the original data (`obs`) that were directly used in fitting the model. Now that we're using a transformation of the `time` column, by wrapping it in `ns`, the `time` column is no longer included in the `augment` output. However, we can easily add it back in using `mutate`, pulling it from the original `obs` dataset, and then proceed as before.

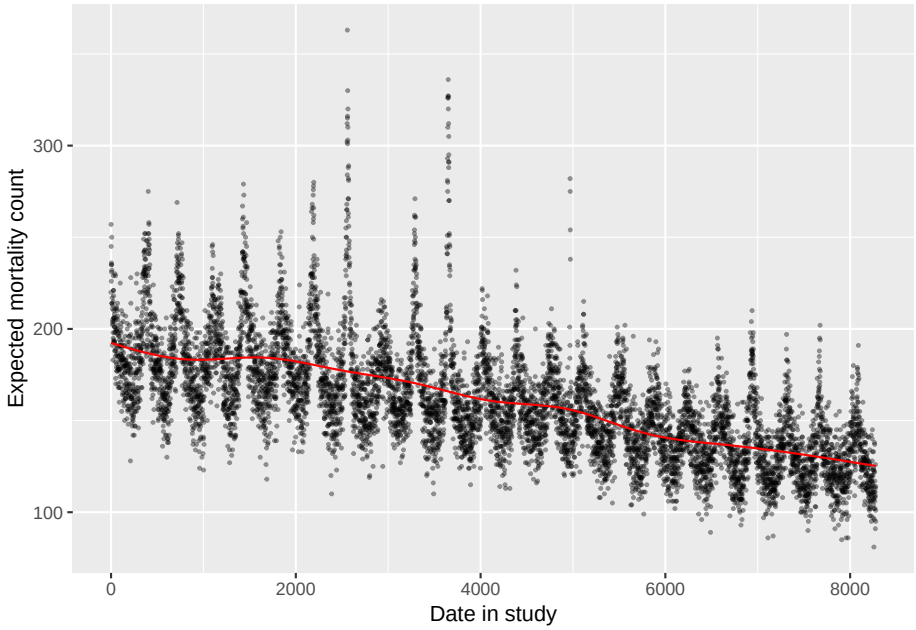
```
mod_time_nonlin %>%
  augment() %>%
  mutate(time = obs$time) %>%
  ggplot(aes(x = time)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Date in study", y = "Expected mortality count")
```



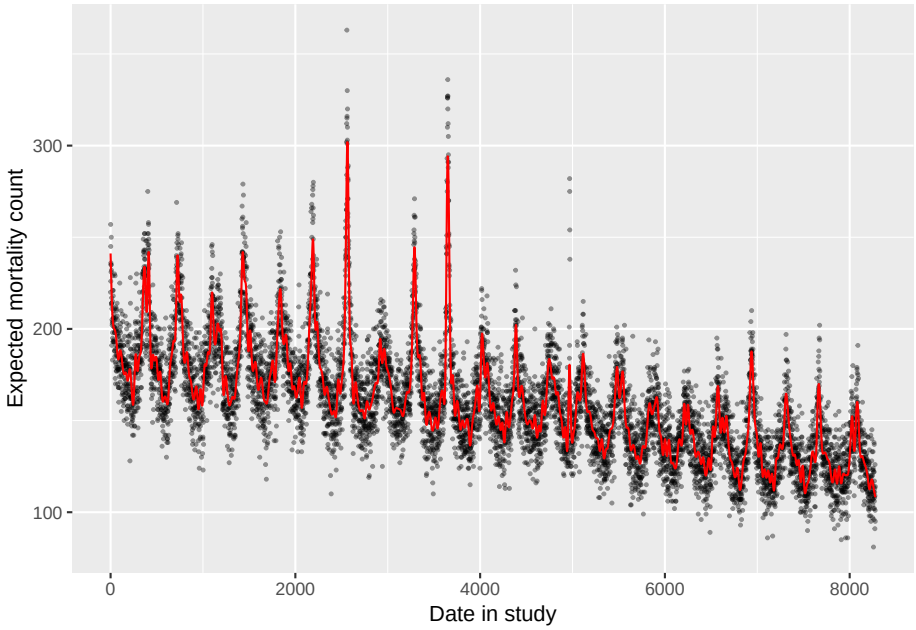
The non-linear term for time has allowed enough flexibility that the model now captures both long-term and seasonal trends in the data.

You might wonder how many degrees of freedom you should use for this time spline. In practice, researchers often using about 6–8 degrees of freedom per year of the study, in the case of year-round data. You can explore how changing the degrees of freedom changes the way the model fits to the observed data. As you use more degrees of freedom, the line will capture very short-term effects, and may start to interfere with the shorter-term associations between environmental exposures and health risk that you are trying to capture. Even in the example model we just fit, for example, it looks like the control for time may be capturing some patterns that were likely caused by heatwaves (the rare summer peaks, including one from the 1995 heatwave). Conversely, if too few degrees of freedom are used, the model will shift to look much more like the linear model, with inadequate control for seasonal patterns.

```
# A model with many less d.f. for the time spline
mod_time_nonlin_lowdf <- glm(all ~ ns(time, df = 10),
                             data = obs, family = "quasipoisson")
mod_time_nonlin_lowdf %>%
  augment() %>%
  mutate(time = obs$time) %>%
  ggplot(aes(x = time)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Date in study", y = "Expected mortality count")
```




```
# A model with many more d.f. for the time spline
# (Takes a little while to run)
mod_time_nonlin_highdf <- glm(all ~ ns(time, df = 400),
                              data = obs, family = "quasipoisson")
mod_time_nonlin_highdf %>%
  augment() %>%
  mutate(time = obs$time) %>%
  ggplot(aes(x = time)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_line(aes(y = exp(.fitted)), color = "red") +
  labs(x = "Date in study", y = "Expected mortality count")
```



In all cases, when you fit a non-linear function of an explanatory variable, it will make the model summary results look much more complicated, e.g.:

```
mod_time_nonlin_lowdf %>%
  tidy()
```

```
## # A tibble: 11 x 5
##    term                 estimate std.error statistic   p.value
##    <chr>                   <dbl>     <dbl>     <dbl>     <dbl>
##  1 (Intercept)           5.26     0.00948    555.   0.
##  2 ns(time, df = 10)1   -0.0260   0.0119      -2.18 2.93e-  2
##  3 ns(time, df = 10)2   -0.0860   0.0155      -5.56 2.85e-  8
##  4 ns(time, df = 10)3   -0.114    0.0139      -8.15 4.01e- 16
##  5 ns(time, df = 10)4   -0.196    0.0151     -13.0  4.47e- 38
##  6 ns(time, df = 10)5   -0.187    0.0148     -12.6  2.80e- 36
##  7 ns(time, df = 10)6   -0.315    0.0154     -20.5  5.62e- 91
##  8 ns(time, df = 10)7   -0.337    0.0154     -21.9  1.95e-103
##  9 ns(time, df = 10)8   -0.358    0.0135     -26.5  1.56e-148
## 10 ns(time, df = 10)9   -0.467    0.0244     -19.2  4.49e- 80
## 11 ns(time, df = 10)10  -0.392    0.0126     -31.2  8.01e-202
```

You can see that there are multiple model coefficients for the variable fit using a spline function, the same as the number of degrees of freedom. These model coefficients are very hard to interpret on their own. When we are using the spline to *control* for a factor that might serve as a confounder of the association of interest, we typically won't need to try to interpret these model coefficients—

instead, we are interested in accounting for how this factor explains variability in the outcome, without needing to quantify the association as a key result. However, there are also cases where we want to use a spline to fit the association with the exposure that we are interested in. In this case, we will want to be able to interpret model coefficients from the spline. Later in this chapter, we will introduce the `dlnm` package, which includes functions to both fit and interpret natural cubic splines within GLMs for environmental epidemiology.

2. **Start from the last model created in the last chapter and add control for long-term and seasonal trends over the study period.**

The last model fit in the last chapter was the following, which fits for the association between a linear term of temperature and mortality risk, with control for day of week:

```
mod_ctrl_dow <- glm(all ~ tmean + factor(dow, ordered = FALSE),
                    data = obs, family = "quasipoisson")
```

To add control for long-term and seasonal trends, you can take the natural cubic spline function of temperature that you just fit and include it among the explanatory / independent variables from the model in the last chapter. If you want to control for only long-term trends, a linear term of the `time` column could work, as we discovered in the first part of this chapter's exercise. However, seasonal trends could certainly confound the association of interest. Mortality rates have a clear seasonal pattern, and temperature does as well, and these patterns create the potential for confounding when we look at how temperature and mortality risk are associated, beyond any seasonally-driven pathways.

```
mod_ctrl_dow_time <- glm(all ~ tmean + factor(dow, ordered = FALSE) +
                           ns(time, df = 158),
                         data = obs, family = "quasipoisson")
```

You can see the influence of this seasonal confounding if you look at the model results. When we look at the results from the model that did not control for long-term and seasonal trends, we get an estimate that mortality rates tend to be lower on days with higher temperature, with a negative term for `tmean`:

```
mod_ctrl_dow %>%
  tidy() %>%
  filter(term == "tmean")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic p.value
##   <chr>    <dbl>     <dbl>     <dbl>   <dbl>
## 1 tmean  -0.0148  0.000354     -41.7       0
```

Conversely, when we include control for long-term and seasonal trends, the estimated association between mortality rates and temperature is reversed, estimating increased mortality rates on days with higher temperature, *controlling*

for long-term and seasonal trends:

```
mod_ctrl_dow_time %>%
  tidy() %>%
  filter(term == "tmean")
```

```
## # A tibble: 1 x 5
##   term  estimate std.error statistic  p.value
##   <chr>    <dbl>     <dbl>     <dbl>    <dbl>
## 1 tmean  0.00370  0.000395      9.36 1.02e-20
```

3. **Refine your model to fit for a non-linear, rather than linear, function of temperature in the model.**

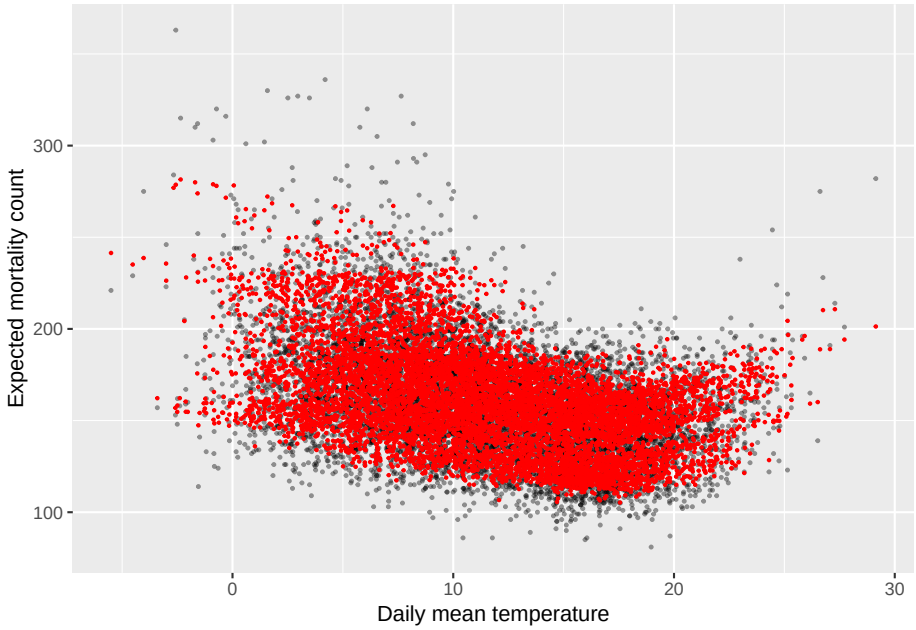
You can use a spline in the same way to fit a non-linear function for the exposure of interest in the model (temperature). We'll start there. However, as mentioned earlier, it's a bit tricky to interpret the coefficients from the fit model—you no longer generate a single coefficient for the exposure of interest, but instead several related to the spline. Therefore, once we show how to fit using `ns` directly, we'll show how you can do the same thing using specialized functions in the `dlnm` package. This package includes a lot of nice functions for not only fitting an association using a non-linear term, but also for interpreting the results after the model is fit.

First, here is code that can be used to fit the model using `ns` directly, similarly to the approach we used to control for temporal patterns with a flexible function:

```
mod_ctrl_nl_temp <- glm(all ~ ns(tmean, 4) + factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

We can plot the predicted values from this fitted model (red points in the plot below) compared to the observed data (black dots) using our usual method of using `augment` to extract the predicted values:

```
mod_ctrl_nl_temp %>%
  augment() %>%
  mutate(tmean = obs$tmean) %>%
  ggplot(aes(x = tmean)) +
  geom_point(aes(y = all), alpha = 0.4, size = 0.5) +
  geom_point(aes(y = exp(.fitted)), color = "red",  size = 0.4) +
  labs(x = "Daily mean temperature", y = "Expected mortality count")
```



However, these predictions are very variable at any given temperature. This reflects how other independent variables, like long-term and seasonal trends, explain variability in mortality. This makes sense, but it makes it a bit hard to investigate the role of temperature specifically. Let's look, then, at some other options for viewing the results that will help us focus on the association of the exposure we care about (temperature) with the outcome.

The next part has a lot of steps, so it might at first seem confusing. However, fortunately we won't always have to do all the steps ourselves—there is a nice R package that will help us. We'll look at the process first, though, and then the easier way to use a package to help with some of the steps. Also, this process gives us a first look at how the idea of basis functions work. We'll later expand on these to look at non-linear relationships with the exposure at different lag times, using cross-basis functions, so this forms a starting point for moving into the idea of a cross-basis.

First, we need to think about how temperature gets included in the regression model—specifically, as a function of *basis variables* rather than as a single variable. In other words, to include temperature as a nonlinear function, we're going to create a structure in the regression equation that includes the variable of temperature in several different terms, in different transformations in each term. I simple example of this is a second-degree polynomial. If we wanted to include a second-order polynomial function of temperature in the regression, then we'd use the following basis:

$$\beta_1 T + \beta_2 T^2$$

Notice that here we're using the same independent variable (T, which stands for daily temperature), but we're including both untransformed T and also T squared. This function of T might go into the regression equation as something like:

$$E(Y) = \alpha + \beta_1 T + \beta_2 T^2 + ns(time) + {}'\mathbf{D}$$

where α is the intercept, $ns(time)$ is a natural cubic spline that controls for time (long-term and seasonal trends) and $\mathbf{D}$ is day of week, with as the set of coefficients associated with day of week. (Here, I've included the outcome, Y, untransformed, as you would for a linear regression, but of course the same idea works with Poisson regression, when you'd instead have $log(E(Y))$ on the left of the equation.)

When you fit a regression model in a program like R, it sets up the observed data into something called a *model matrix*, where it has the values for each observation for each of the independent variables you want to include, plus a column of "1"s for the intercept, if you model structure includes one. The regression model will fit a coefficient for each of these columns in the model matrix. In a simple case, this model matrix will just repeat each of your independent variables. For example, the model matrix for the model `mod_overdisp_reg`, which we fit in the last chapter and which included only an intercept and a linear term for `tmean`, looks like this (the `model.matrix` function will give you the model matrix of any `glm` object that you get by running the `glm` function in R):

```
mod_ovdisp_reg %>%
  model.matrix() %>%
  head()
```

```
##   (Intercept)    tmean
## 1           1 3.913589
## 2           1 5.547919
## 3           1 4.385564
## 4           1 5.431046
## 5           1 6.867855
## 6           1 9.232628
```

We already start using the idea of basis variables when we include categorical variables, like day of week. Here is the model matrix for the `mod_ctrl_dow` model that we fit in the last chapter:

```
mod_ctrl_dow %>%
  model.matrix() %>%
  head()
```

```
##   (Intercept)    tmean factor(dow, ordered = FALSE)Mon
## 1           1 3.913589                               1
## 2           1 5.547919                               0
```

```
## 3           1 4.385564                                0
## 4           1 5.431046                                0
## 5           1 6.867855                                0
## 6           1 9.232628                                0
##   factor(dow, ordered = FALSE)Tue factor(dow, ordered = FALSE)Wed
## 1                               0                               0
## 2                               1                               0
## 3                               0                               1
## 4                               0                               0
## 5                               0                               0
## 6                               0                               0
##   factor(dow, ordered = FALSE)Thu factor(dow, ordered = FALSE)Fri
## 1                               0                               0
## 2                               0                               0
## 3                               0                               0
## 4                               1                               0
## 5                               0                               1
## 6                               0                               0
##   factor(dow, ordered = FALSE)Sat
## 1                               0
## 2                               0
## 3                               0
## 4                               0
## 5                               0
## 6                               1
```

You can see that the regression call broke the day-of-week variable up into a set of indicator variables, which equal either 1 or 0. There will be one less of these than the number of categories for the variable; in otherwords, if the categorical variable took two values (Weekend / Weekday), then this would be covered by a single column; since there are seven days in the week, the day-of-week variable breaks into six (7 - 1) columns. The first level of the categories (Sunday in this case) serves as a baseline and doesn't get a value. The others levels (Monday, Tuesday, etc.) each get their own indicator variable—so, their own column in the model matrix—which equals "1" on that day (i.e., "1" for the Monday column if the date of the observation is a Monday) and "0" on all other days.

Now let's go back and look at the example of including temperature in the model as a nonlinear function, starting with the simple example of using a second-degree polynomial. What would the basis for that look like? We can find out by fitting the model and then looking at the model matrix (the `I()` function lets us specify a transformation of a column from the data when we set up the regression equation structure in `glm`):

```
mod_polynomial_temp <- glm(all ~ tmean + I(tmean ^ 2),
                           data = obs, family = "quasipoisson")
mod_polynomial_temp %>%
```

```
  model.matrix() %>%
  head()
```

```
##   (Intercept)    tmean I(tmean^2)
## 1           1 3.913589   15.31618
## 2           1 5.547919   30.77941
## 3           1 4.385564   19.23317
## 4           1 5.431046   29.49627
## 5           1 6.867855   47.16743
## 6           1 9.232628   85.24142
```

You can see that this has created two columns based on the temperature variable, one with it untransformed and one with it squared. The regression will estimate coefficients for each of these basis variables of temperature, and then that allows you to fit a model with a function of temperature, rather than solely temperature. Here are the coefficients from this model:

```
mod_polynomial_temp %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term         estimate std.error statistic   p.value
##   <chr>           <dbl>     <dbl>     <dbl>     <dbl>
## 1 (Intercept)  5.31     0.00675       787.  0.
## 2 tmean       -0.0298   0.00126       -23.6 2.27e-119
## 3 I(tmean^2)   0.000667 0.0000538      12.4 5.70e- 35
```

Therefore, it's fit the following model to describe the association between temperature and mortality:

$$log(E(Y)) = 5.3092 - 0.0298T + 0.0007T^2$$

If you want to estimate the relative risk of mortality at 15 degrees Celsius versus 10 degrees Celsius with this model (which is confounded by long-term and seasonal trends, so has some issues we'll want to fix), you can take the following process. Take the two equations for the expected mortality when T is 15 and 10, respectively:

$$log(E(Y|T = 15)) = 5.3092 - 0.0298(15) + 0.0007(15^2) log(E(Y|T = 10)) = 5.3092 - 0.0298(10) + 0.0007(10^2)$$

If you subtract the second from the first, you get:

$$log(E(Y|T = 15)) - log(E(Y|T = 10)) = (5.3092 - 5.3092) - 0.0298(15 - 10) + 0.0007(15^2 - 10^2)$$

which simplifies to (if the left part isn't clear, review the rules for how you can manipulate logarithms):

$$log(\frac{E(Y|T=15)}{E(Y|T=10)}) = -0.0298(5) + 0.0007(125) = -0.0615$$

Exponentiate both sides, to get to the relative risk at 15 degrees versus 10 degrees (in other words, the ratio of expected mortality at 15 degrees to that at 10 degrees):

$$\frac{E(Y|T=15)}{E(Y|T=10)} = e^{-0.0615} = 0.94$$

You can see that the basis variables are great in helping us explore how an independent variable might be related to the outcome in a non-linear way, but there's a bit of a cost in terms of us needing to take some extra steps to interpret the results from that model. Also, note that there's not a single coefficient that we can extract from the model as a summary of the relationship. Instead, we needed to pick a reference temperature (10 degrees in this example) and compare to that to get an estimated relative risk. We'll see the same pattern as we move to using natural cubic splines to create the basis variables for temperature.

Now let's move to a spline. The function the spline runs to transform the temperature variable into the different basis variables is more complex than for the polynomial example, but the result is similar: you get several columns to fit in the model for a variable, compared to the single column you would include if you were only fitting a linear term. If you take a look at the output of running the `ns` function on temperature in the data, you can see that it creates several new columns (one for each degree of freedom), which will be the basis variables in the regression:

```
ns(obs$tmean, df = 4) %>%
  head()
```

```
##              1           2         3          4
## [1,] 0.1769619 -0.20432696 0.4777110 -0.2733841
## [2,] 0.2860253 -0.19686120 0.4602563 -0.2633951
## [3,] 0.2049283 -0.20410506 0.4771922 -0.2730872
## [4,] 0.2770456 -0.19804188 0.4630167 -0.2649748
## [5,] 0.4012496 -0.17595974 0.4113892 -0.2354295
## [6,] 0.6581858 -0.09844302 0.2476396 -0.1417190
```

The output from `ns` also has some metadata, included in the attributes of the object, that have some other information about the spline function, including where the knots were placed and where the boundary knots (which are at the outer ranges of the data) are placed. You can the `str` function to explore both the data and metadata stored in this object:

```
ns(obs$tmean, df = 4) %>%
  str()
```

```
##  'ns' num [1:8279, 1:4] 0.177 0.286 0.205 0.277 0.401 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : NULL
##   ..$ : chr [1:4] "1" "2" "3" "4"
##  - attr(*, "degree")= int 3
##  - attr(*, "knots")= Named num [1:3] 7.47 11.47 15.93
##   ..- attr(*, "names")= chr [1:3] "25%" "50%" "75%"
##  - attr(*, "Boundary.knots")= num [1:2] -5.5 29.1
##  - attr(*, "intercept")= logi FALSE
```

This is saying, for example, that the internal knots for this spline were put at the 25th, 50th, and 75th quantiles of the temperature data, where were 7.47 degrees, 11.47 degrees, and 15.93 degrees. (The defaults for `ns` is to place knots evenly at percentiles, based on the number of degrees of freedom you specify. You can change this by placing the knots "by hand", using the `knots` argument in the `ns` function.)

This spline object can be used not just for its basis values, but also to "predict" new basis values for a new set of temperature values. For example, you could figure out what the basis values from this spline would be for every degree of temperature between -6 and 29 using the following code:

```
temp_spline <- ns(obs$tmean, df = 4)
temp_spline_preds <- predict(temp_spline, newx = -5:30)
temp_spline_preds %>%
  head()
```

```
##                 1           2          3           4
## [1,] 2.689545e-05 -0.01606406 0.03755734 -0.02149328
## [2,] 7.189726e-04 -0.04768237 0.11148013 -0.06379775
## [3,] 3.321933e-03 -0.07825364 0.18295494 -0.10470130
## [4,] 9.107577e-03 -0.10708098 0.25035250 -0.14327152
## [5,] 1.934771e-02 -0.13346753 0.31204355 -0.17857602
## [6,] 3.531412e-02 -0.15671641 0.36639881 -0.20968240
```

This is handy, because it will help us visualize the relationship we fit in a regression model—we can start with these basis values for each degree in our temperature range, and then use the regression coefficients for each basis variable to estimate relative risk at that temperature compared to a reference temperature, exactly as we did in the equations for the polynomial function of temperature earlier, comparing 15 degrees C to the reference of 10 degrees.

All we need now are the regression coefficients for each of these temperature basis variables. We can extract those from the model we fit earlier, where we included `ns(tmean, 4)` as one of the model terms. Here, I'm using `tidy` to get the model coefficients and then, because there are *lots* of them (from fitting a spline for time with lots of degrees of freedom), I'm using the `str_detect` function from the `stringr` package to pick out just those with "tmean" in the

`term` column:

```
mod_ctrl_nl_temp %>%
  tidy() %>%
  filter(str_detect(term, "tmean"))
```

```
## # A tibble: 4 x 5
##   term          estimate std.error statistic  p.value
##   <chr>            <dbl>     <dbl>     <dbl>    <dbl>
## 1 ns(tmean, 4)1  -0.0700    0.0135    -5.18  2.30e- 7
## 2 ns(tmean, 4)2  -0.0660    0.0126    -5.24  1.61e- 7
## 3 ns(tmean, 4)3   0.0110    0.0313     0.350 7.26e- 1
## 4 ns(tmean, 4)4   0.347     0.0177    19.6   2.02e-83
```

You can add on `pull` to pull out just the `estimate` column as a vector, which will be helpful when we want to multiple these coefficients by the basis values for each degree of temperature across our temperature range:

```
temp_spline_ests <- mod_ctrl_nl_temp %>%
  tidy() %>%
  filter(str_detect(term, "tmean")) %>%
  pull(estimate)
temp_spline_ests
```

```
## [1] -0.06995454 -0.06596939  0.01095574  0.34659715
```

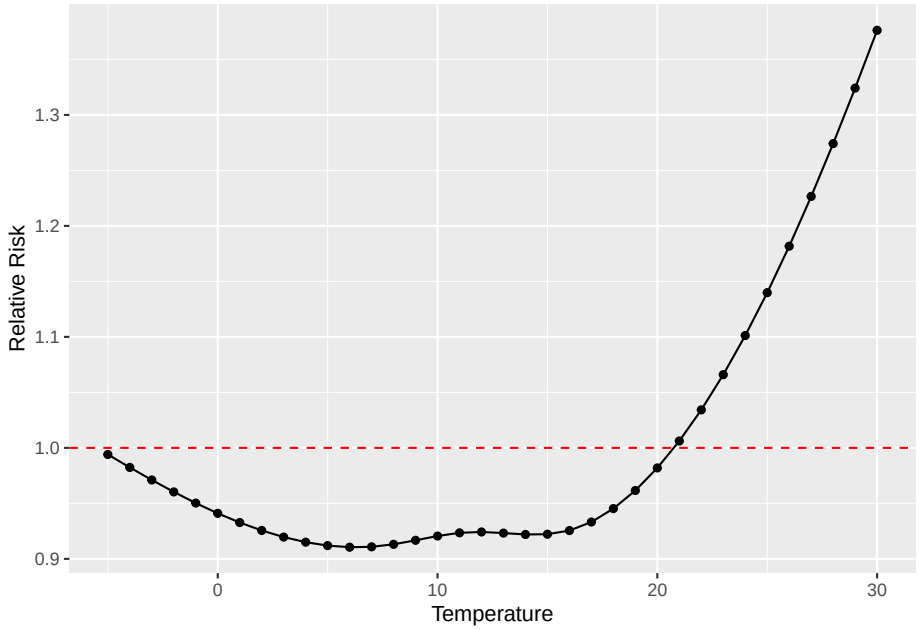
Now, let's put this together to see how relative risk of mortality changes as you move across the temperature range! First, we can set up a dataframe that has a column with each unit of temperature across our range—these are the temperatures where we want to estimate relative risk—and then the estimated relative risk at that temperature. By default, we'll be comparing to the lowest temperature in the original data, but we'll talk in a minute about how to adjust to a different reference temperature. You can use matrix multiplication (`%*%`) as a shorthand way to multiple each column of the spline basis variables from `temp_spline_preds` by its estimated coefficient from the regression model, saved in `temp_spline_ests`, and then add all of those values together (this is the same idea as what we did in the equations earlier, for the polynomial basis). Then, to get from log relative risk to relative risk, we'll exponentiate that with `exp`. You can see the first rows of the results below:

```
pred_temp_function <- tibble(
  temp = -5:30,
  temp_func = temp_spline_preds %*% temp_spline_ests,
  rr = exp(temp_func)
)

pred_temp_function %>%
  head()
```

```
## # A tibble: 6 x 3
##    temp temp_func[,1] rr[,1]
##   <int>         <dbl>  <dbl>
## 1    -5      -0.00598  0.994
## 2    -4      -0.0178   0.982
## 3    -3      -0.0294   0.971
## 4    -2      -0.0405   0.960
## 5    -1      -0.0510   0.950
## 6     0      -0.0608   0.941
```

This is now very easy to plot, with a reference line added at a relative risk of 1.0:

```
ggplot(pred_temp_function, aes(x = temp, y = rr)) +
  geom_point() +
  geom_line() +
  labs(x = "Temperature",
       y = "Relative Risk") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)
```



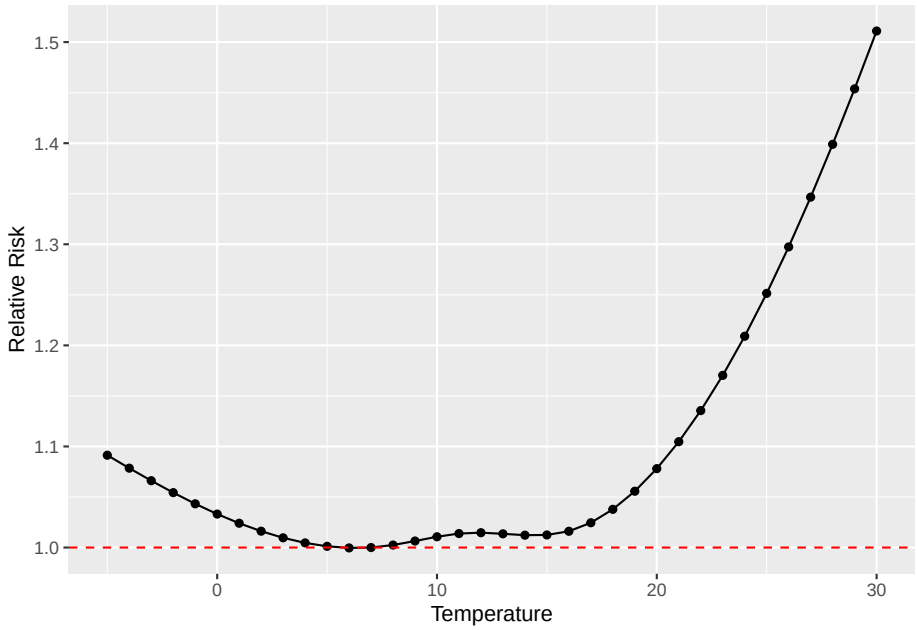
We might want to shift this, so we're comparing the temperature at which mortality risk is lowest (sometimes called the *minimum mortality temperature*). This tends to be at milder temperatures, in the middle of our range, rather than at the minimum temperature in the range. To start, let's see what temperature aligns with the lowest relative risk of mortality:

```
pred_temp_function %>%
  filter(rr == min(rr))
```

```
## # A tibble: 1 x 3
##    temp temp_func[,1] rr[,1]
##   <int>         <dbl>  <dbl>
## 1     6       -0.0938  0.911
```

We can therefore realign so that the relative risk equals 1.0 when the temperature is 7 degrees C, and all other relative risks are relative to a reference temperature of 7 degrees C:

```
pred_temp_function <- pred_temp_function %>%
  mutate(temp_func_reset = temp_func - temp_func[temp == 7],
         rr_reset = exp(temp_func_reset)
)

ggplot(pred_temp_function, aes(x = temp, y = rr_reset)) +
  geom_point() +
  geom_line() +
  labs(x = "Temperature",
       y = "Relative Risk") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)
```



This is a fairly cumbersome process, as you've seen with this example, and it would be a pain if we had to do it every time we wanted to visualize and

explore results from fitting regressions that include basis variables from splines. Fortunately, there's a nice R package called `dlnm` (for "distributed lag nonlinear models") that will help take care of a lot of these "under the hood" steps for us. Even better, this package will let us move on to more complex crossbasis functions, where we fit non-linear functions in two dimensions, and help us visualize and interpret results from those models.

To start, make sure you have the `dlnm` package installed on your computer, and then load it in your R session. This package has a function called `crossbasis` that lets you build a crossbasis function to use in a regression model. We'll talk more about these types of functions later in this chapter; for right now, we'll be a bit simpler and just use it to create our spline function of temperature.

In the `crossbasis` function, you start by putting in the vector of the variable that you want to expand into basis variables. In our case, this is temperature, which we have saved as the `tmean` column in the `obs` dataset. You use the `argvar` to give some information about the basis you want to use for that variable. This will both include the type of basis function (`"ns"` for a natural cubic spline), and then also arguments you want to pass to that basis function, like degrees of freedom (`df`) or knot locations (`knots`) if you're fitting a spline. The other arguments allow you to specify a function of lagged time; we won't use that yet, so you can just include `lag = 0` and `arglag = list(fun = "integer")` to create a crossbasis where we only consider same-day effects in a simple way. If you look at the output, you'll see it's very similar to the basis created by the `ns` call earlier (in fact, the values in each column should be identical):

```
library(dlnm)
temp_basis <- crossbasis(obs$tmean, lag = 0,
                         argvar = list(fun = "ns", df = 4),
                         arglag = list(fun = "integer"))
temp_basis %>%
  head()
```

```
##           v1.l1       v2.l1     v3.l1      v4.l1
## [1,] 0.1769619 -0.20432696 0.4777110 -0.2733841
## [2,] 0.2860253 -0.19686120 0.4602563 -0.2633951
## [3,] 0.2049283 -0.20410506 0.4771922 -0.2730872
## [4,] 0.2770456 -0.19804188 0.4630167 -0.2649748
## [5,] 0.4012496 -0.17595974 0.4113892 -0.2354295
## [6,] 0.6581858 -0.09844302 0.2476396 -0.1417190
```

To estimate the regression coefficients, we'll put this whole crossbasis in as one of our terms in the `glm` regression equation:

```
dlnm_mod_1 <- glm(all ~ temp_basis + factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

As when we fit the spline earlier, you can see that this gives us a set of coefficients,

one for each column in the matrix of crossbasis variables:

```
dlnm_mod_1 %>%
  tidy() %>%
  filter(str_detect(term, "temp_basis"))
```

```
## # A tibble: 4 x 5
##   term            estimate std.error statistic  p.value
##   <chr>              <dbl>     <dbl>     <dbl>    <dbl>
## 1 temp_basisv1.l1  -0.0700    0.0135    -5.18  2.30e- 7
## 2 temp_basisv2.l1  -0.0660    0.0126    -5.24  1.61e- 7
## 3 temp_basisv3.l1   0.0110    0.0313     0.350 7.26e- 1
## 4 temp_basisv4.l1   0.347     0.0177    19.6   2.02e-83
```

However, there's an advantage with using `crossbasis`, even though it's looked pretty similar to using `ns` up to now. That's that there are some special functions that let us predict and visualize the model results without having to do all the work we did before.

For example, there's a function called `crosspred` that will give us a number of values, including the estimated relative risk compared to a reference value. To use this, we need to input the object name of our crossbasis object (`temp_basis`) and the name of our regression model object (`dlnm_mod_1`). We also want to tell it which temperature we want to use as our reference (`cen = 7` to compare everything to 7 degrees C) and what interval we want for predictions (`by = 1` will give us an estimate for every degree temperature along our range). The output is a list with a lot of elements, which you can get a view of with `str`:

```
crosspred(basis = temp_basis, model = dlnm_mod_1, cen = 7, by = 1) %>%
  str()
```

```
## List of 19
##  $ predvar     : num [1:35] -5 -4 -3 -2 -1 0 1 2 3 4 ...
##  $ cen         : num 7
##  $ lag         : num [1:2] 0 0
##  $ bylag       : num 1
##  $ coefficients: Named num [1:4] -0.07 -0.066 0.011 0.347
##   ..- attr(*, "names")= chr [1:4] "temp_basisv1.l1" "temp_basisv2.l1" "temp_basisv3
##  $ vcov        : num [1:4, 1:4] 1.83e-04 1.12e-04 3.85e-04 6.58e-05 1.12e-04 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:4] "temp_basisv1.l1" "temp_basisv2.l1" "temp_basisv3.l1" "temp_bas
##   .. ..$ : chr [1:4] "temp_basisv1.l1" "temp_basisv2.l1" "temp_basisv3.l1" "temp_bas
##  $ matfit      : num [1:35, 1] 0.0874 0.0756 0.064 0.0529 0.0424 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:35] "-5" "-4" "-3" "-2" ...
##   .. ..$ : chr [1, 1] "lag0"
##  $ matse       : num [1:35, 1] 0.01435 0.01254 0.01077 0.00907 0.00746 ...
##   ..- attr(*, "dimnames")=List of 2
```

```
##   .. ..$ : chr [1:35] "-5" "-4" "-3" "-2" ...
##   .. ..$ : chr [1, 1] "lag0"
##  $ allfit      : Named num [1:35] 0.0874 0.0756 0.064 0.0529 0.0424 ...
##   ..- attr(*, "names")= chr [1:35] "-5" "-4" "-3" "-2" ...
##  $ allse       : Named num [1:35] 0.01435 0.01254 0.01077 0.00907 0.00746 ...
##   ..- attr(*, "names")= chr [1:35] "-5" "-4" "-3" "-2" ...
##  $ matRRfit    : num [1:35, 1] 1.09 1.08 1.07 1.05 1.04 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:35] "-5" "-4" "-3" "-2" ...
##   .. ..$ : chr [1, 1] "lag0"
##  $ matRRlow    : num [1:35, 1] 1.06 1.05 1.04 1.04 1.03 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:35] "-5" "-4" "-3" "-2" ...
##   .. ..$ : chr [1, 1] "lag0"
##  $ matRRhigh   : num [1:35, 1] 1.12 1.11 1.09 1.07 1.06 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:35] "-5" "-4" "-3" "-2" ...
##   .. ..$ : chr [1, 1] "lag0"
##  $ allRRfit    : Named num [1:35] 1.09 1.08 1.07 1.05 1.04 ...
##   ..- attr(*, "names")= chr [1:35] "-5" "-4" "-3" "-2" ...
##  $ allRRlow    : Named num [1:35] 1.06 1.05 1.04 1.04 1.03 ...
##   ..- attr(*, "names")= chr [1:35] "-5" "-4" "-3" "-2" ...
##  $ allRRhigh   : Named num [1:35] 1.12 1.11 1.09 1.07 1.06 ...
##   ..- attr(*, "names")= chr [1:35] "-5" "-4" "-3" "-2" ...
##  $ ci.level    : num 0.95
##  $ model.class : chr [1:2] "glm" "lm"
##  $ model.link  : chr "log"
##  - attr(*, "class")= chr "crosspred"
```

The one we'll look at right now is `allRRfit`. You can extract it with (here I'm showing how you can do it with piping and the `pluck` function to pull out an element of a list, but you could do other approaches, too):

```
est_rr <- dlnm_mod_1 %>%
  crosspred(basis = temp_basis, model = ., cen = 7, by = 1) %>%
  pluck("allRRfit")
est_rr
```

```
##        -5        -4        -3        -2        -1         0         1         2
## 1.0913393 1.0785207 1.0661255 1.0543222 1.0432720 1.0331298 1.0240458 1.0161668
##         3         4         5         6         7         8         9        10
## 1.0096382 1.0046057 1.0012180 0.9996288 1.0000000 1.0024680 1.0064490 1.0106777
##        11        12        13        14        15        16        17        18
## 1.0138437 1.0147024 1.0135774 1.0122284 1.0124622 1.0160921 1.0245314 1.0378080
##        19        20        21        22        23        24        25        26
## 1.0557076 1.0780540 1.1046963 1.1354974 1.1703226 1.2090286 1.2514526 1.2974015
##        27        28        29
```

```
## 1.3466411 1.3988856 1.4537866
```

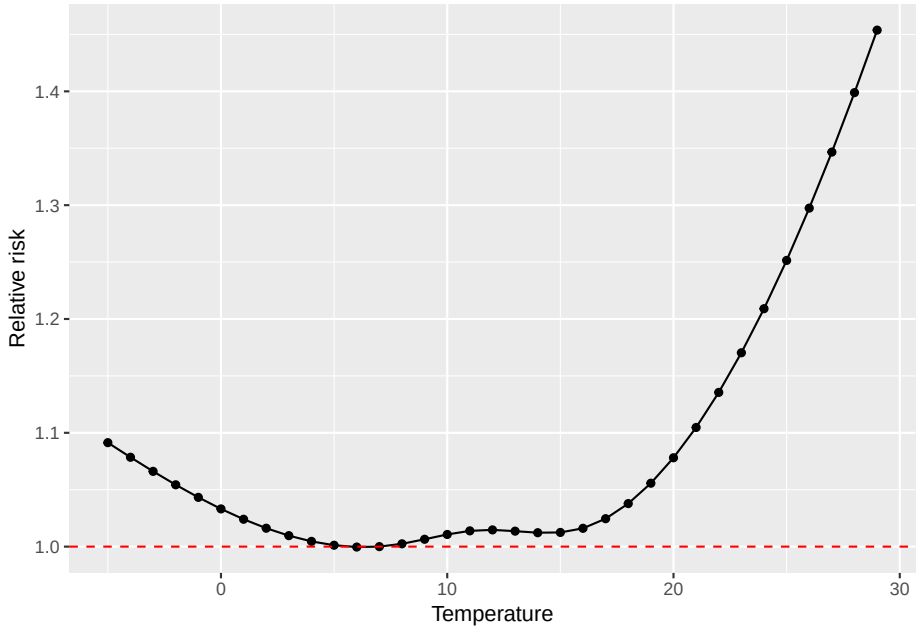
To make it easier to work with, let's put this in a dataframe. Note that the temperatures are included as the names in the vector, so we can extract those with `names`:

```
dlnm_temp_function <- tibble(tmean = as.numeric(names(est_rr)),
                             rr = est_rr)

dlnm_temp_function %>%
  head()
```

```
## # A tibble: 6 x 2
##   tmean    rr
##   <dbl> <dbl>
## 1    -5  1.09
## 2    -4  1.08
## 3    -3  1.07
## 4    -2  1.05
## 5    -1  1.04
## 6     0  1.03
```

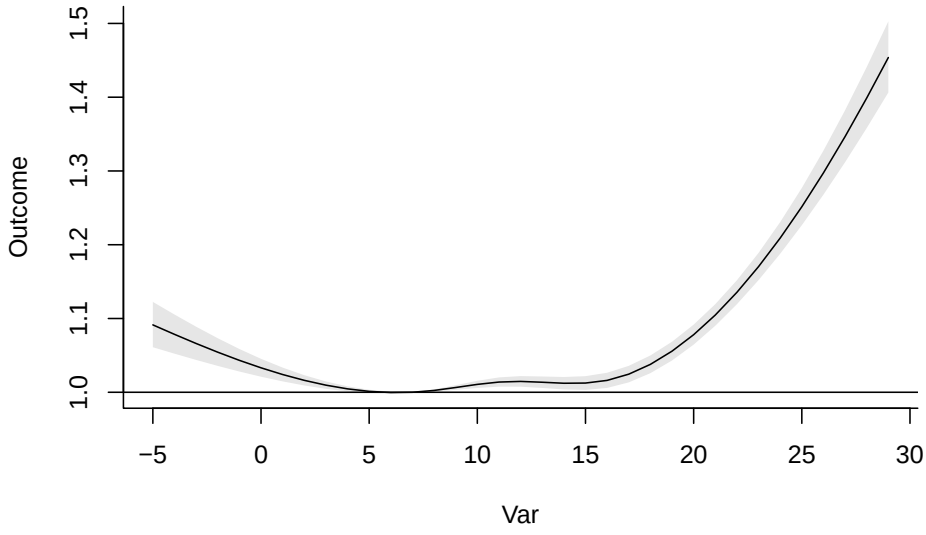
This has gotten us (much more quickly!) to estimates of the relative risk of mortality at each temperature compared to a reference of 7 degrees C. We can plot this the same way we did earlier, and you'll notice that this plot is identical to the one we created based on the regression with `ns` earlier:

```
ggplot(dlnm_temp_function, aes(x = tmean, y = rr)) +
  geom_point() +
  geom_line() +
  labs(x = "Temperature", y = "Relative risk") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)
```



The `dlnm` package also includes some functions specifically for plotting. One downside is that they're in base R, rather than based on `ggplot2`, which makes them a bit harder to customize. However, they're a great way to get a first look at your results:

```
dlnm_mod_1 %>%
  crosspred(basis = temp_basis, model = ., cen = 7, by = 1) %>%
  plot()
```



Don't worry if all of this about basis functions is a lot! If this is the first time you've seen this, it will likely take a few passes to get comfortable with it—and while the idea of basis variables is fairly straightforward, it's certainly not straightforward to figure out how to interpret the results of the model that you fit. It is worth the effort, though, as this is a very powerful tool when working with regression modeling in environmental epidemiology.

4.3 Distributed lags and cross-basis functions in GLMs

Next, let's explore how we can add into our model something that lets us look at delayed effects and the potential for mortality displacement. So far, we have only considered how temperature affects mortality risk on the day of exposure—in other words, if it is very hot today, does risk of mortality go up substantially today? For many exposures, however, risk can persist after the day of exposure.

In some cases, this may be caused by a delay in detection of the outcome in the data we have. For example, extreme heat today could precipitate a respiratory or cardiac event that the person may not seek treatment for or result in an outcome like mortality until tomorrow, and so in the data the outcome would be recorded at a one-day delay from the exposure.

In other cases, the path from the exposure to the outcome may take several days. For example, cold temperatures are often found to be associated with respiratory outcomes a week or more after exposures (something you can look for in the example data!), and one potential pathway is that cold temperatures might promote the spread of respiratory infectious disease, like influenza and colds, both through making people stay inside in more crowded, less ventilated conditions and also through effects on humidity and other conditions that might influence the behavior of respiratory droplets, which are key players in spreading some respiratory diseases. Since there is an incubation time up to several days for these infectious diseases, we might expect a lagged association between temperature and respiratory outcomes under this scenario.

There is another lagged pattern that can be important to check for, as well—one that indicates mortality displacement. For many ambient environmental exposures, there are clear indications that associations with health outcomes are strongest among older adults or other populations with a disproportionately high percent of people who are frail before exposure. One question that comes up, then, is if, when an exposure is associated with excess deaths, these are deaths that are displaced in time by only a few days or weeks, rather than deaths that represent a larger time of life lost. This question—of whether deaths observed to be associated with an environmental exposure were displaced by only a small amount of time—can be explored by investigating whether an observed increase in mortality across a community during an exposure is then offset by a lower-than-expected rate of mortality in the following days and weeks.This

phenomenon is sometimes also referred to as 'harvesting' or a 'harvesting effect'.

All of these questions can be explored through a type of model called a *distributed lag model*. Conceptually, these models investigate how an exposure today is associated with risk of a health outcome not only today, but also in the following days. In time-series studies where we typically want to estimate potential short-term and/or acute effects, and because we are concerned about potential confounding by seasonal trends—and have incorporated control for these trends in our models—we often restrict these models to only look up to about a month following exposure. However, this can help in identifying delayed effects of an exposure days to weeks following the initial exposure, and can also help in determining the role of mortality displacement in accounting for some or all of the excess deaths that are associated with an exposure. While some environmental exposures of interest have mostly immediate effects (e.g., hot temperature), others, including many air pollutants, tend to have effects that are stretched over a longer period of one or more days from exposure. As a result, distributed lag models are used widely in environmental epidemiology studies of time series data, to ensure that important associations aren't missed by limiting the model to focus on risk that are detectable on the same day as the exposure. They can also help us identify influential windows of exposure as well as help eliminate potential confounding by exposures occurring at proximal times to the window of interest. For example, today's exposure may appear to have an effect on a health outcome even if in reality yesterday's exposure is the influential one and we don't control for it, because today's and yesterday's exposure to temperature or air pollution are likely to be very similar.

To fit a distributed lag model, you can start with a fairly straightforward extension of the regression models we've been fitting, especially if you are investigating an exposure that can be plausibly fit with a linear term (rather than a more complex non-linear basis function) in the regression model. Say you are starting, for example, from the following regression model:

$$log(E(Y)) = \alpha + \beta X + \ ' \mathbf{W} + ns(time)$$

In this model, you are using a Poisson regression to investigate how risk of the health outcome (Y) changes for every unit increase in the exposure (X), assuming a linear relationship between the exposure and the log of the health outcome. The model can include control for confounders like day of week ($\mathbf{W}$) and long-term and seasonal time trends ($ns(time)$). This model should look familiar from some of the models we've fit earlier to the example temperature data.

To expand this model to explore associations of temperature at different lags, the simplest approach is to add a term for each lag—in other words, instead of only having a term for temperature on the same day as the outcome measurement, we'll also include a term for temperature the previous day, and the day before

that, and so on. For example, here's what the model would look like if we included terms up to lag 2:

$$log(E(Y)) = \alpha + \beta_0 X_0 + \beta_1 X_1 + \beta_2 X_2 + \ ' \mathbf{W} + ns(time)$$

where X_0 is temperature on the same day as the measured outcome Y (with an associated coefficient β_0), X_1 is the temperature the day before (with an associated coefficient β_1), and X_2 is the temperature two days before (with an associated coefficient β_2). The interpretation of each of these coefficients is similar—$exp(\beta_2)$, for example, gives our estimate of the change in the relative risk of the outcome Y for every one-unit increase in the exposure two days prior.

It can become cumbersome to write out the separate terms for each lag day, so you'll often see equations for distributed lag models written using a shorter approach with the $\sum$ symbol. The $\sum$ symbol expresses several terms that are added together and so is well-suited for shortening the expression of a distributed lag model. For example, the previous model equation could be rewritten as:

$$log(E(Y)) = \alpha + \sum_{l=0}^{2} \beta_l X_l + \ ' \mathbf{W} + ns(time)$$

This version of a distributed lag model is nice because it is fairly simple—we're taking the idea of fitting a linear term for an exposure, and then expanding it to include the exposure on earlier days by adding the same type of term for exposure measured at each lag. However, it does have some downsides. The main downside is that exposure tends to be very strongly correlated from one day to the next. Any time you put different terms in a regression model that are correlated, you create the risk of *collinearity*, and associated problems with fitting the model. The term *collinearity* refers to the idea that you have two or more columns in your model matrix that give identical (or, if you loosen the definition a bit, very similar) information. To fit coefficients for each term, the modeling algorithm is trying to identify weights to place on each column in the model matrix that result in a model that gives the best likelihood of the data you see. If two or more columns have (almost) identical information, then the algorithm struggles to determine how to distribute weight across these two columns. The implications are that you can end up with a lot of instability in model coefficients for the columns that are collinear, as well as very inflated estimates of the standard error. Long story short, it can cause problems in your regression model if you include independent variables that are very strongly correlated with each other, and so we usually try to avoid doing that.

There are alternative distributed lag models that can help avoid this instability that can arise from fitting a distributed lag by using a separate term for each lag. All of them share a common feature—they somehow fit a function of the lags instead of each separate lag, so that we end up adding fewer terms to the regression model (and, in turn, are somewhat constraining the pattern that the

distributed lag effects follow). At the most extreme, you can use a function that reduces all the lag terms to a single term, which you can do by averaging the exposure across all lags. In other words, you could take the same-day temperature, yesterday's temperature, and the temperature the day before that, and average those three temperatures to create the term you put in the model. The model equation in this case would look like:

$$log(E(Y)) = \alpha + \beta_{\overline{0-2}} X_{\overline{0-2}} + \ ' \mathbf{W} + ns(time)$$

where $\overline{0-2}$ represents an average over lags 0 to 2, and so $X_{\overline{0-2}}$ is the average of the exposure on lags 0, 1, and 2 compared to the day that Y is measured.

This approach avoids any issues with collinearity across exposure terms from different lags, because you're only including a single term for exposure in the model. However, there is a downside—this model will provide you with a single estimate for the lagged effects, without allowing you to explore patterns across lags (for example, is the association mostly on the day of exposure, or are there some delayed effects on following days?). Instead, the main assumption is that the effect is constant across all days in the window of interest.

Another approach is more complex, but it allows you to explore patterns across lags. Instead of reducing the lagged exposure to a single term in the model, you can reduce it to a few terms that represent the basis for a smooth function. Often, a polynomial or natural cubic spline function will be used for this. The details of setting up the model become more complex, but fortunately there are software packages (like the `dlnm` package in R) that help in managing this set-up, as we'll explore in the exercise.

Packages like `dlnm` can help in extending the complexity of the distributed lag model in other ways, too. So far, we have explored using a distributed lag model where we are happy to model the exposure using a linear term. As we've seen in earlier exercises, this isn't ideal for temperature. Instead, temperature has a non-linear association with mortality risk, with the lowest risk at mild temperatures and then increased risk at both lower (colder) and higher (hotter) temperatures.

We can fit a GLM that incorporates the exposure using a non-linear shape while, at the same time, including a non-linear function to describe how the association evolves across lagged time following the exposure. The approach is to extend the idea of using a basis function—instead of using a basis function in a single dimension, we'll use a cross of basis functions in two dimensions. One dimension is the level of exposure (e.g., cold to hot temperatures), and the other is the timing between the exposure and the outcome (e.g., same-day to lagged by many days). In terms of the mechanics of building a model matrix to use to fit the regression model, everything follows directly from the idea we explored earlier of using a function like a spline to create basis variables in that model matrix for a term with a non-linear association with the outcome variable. However, the mechanics do get quite cumbersome as so many terms are included, and so

again we will use the `dlnm` package to handle these mechanics as we start to incorporate these cross-basis functions within a GLM.

Applied: Including a cross-basis function for exposure-lag-response in a GLM

For this exercise, you will continue to build up the model that you began in the examples in the previous chapter. The example uses the data provided with one of this chapter's readings, Vicedo-Cabrera et al. (2019).

1. Start with the simple model describing how are daily mortality counts are associated with a linear function of temperature. Include control for day of week and long-term / seasonal trends. Add to this model a term that describes the lagged effects of temperature up to a week following exposure, using a separate model term for each lag. What patterns do you detect in the lagged effects of temperature?
2. Start with the model from the first question, but change it to use a smooth function of lag to detect lagged effects, rather than fitting a seperate term for each lag. Extend the model to look at lagged effects up to a month following exposure. What patterns do you see in these lagged effects?
3. Refine your model to fit for a non-linear, rather than linear, function of temperature, while also incorporating a function to describe lagged effects up to a month following exposure. Based on this model, what does the association between temperature and mortality look like at lag 0 days? How about at lag 21 days? What does the pattern of lagged effects look like when comparing the relative risk of mortality at a hot temperature (e.g., 28 degrees C) to a milder temperature (7 degrees C)? What about when comparing mortality at a cold temperature (e.g., -4 degrees C) to a milder temperature (7 degrees C)? If you use a constant term for the effect of exposure over a month as opposed to the smooth function you just used, how does the overall cumulative RR compare?
4. Assume that we are interested in the potential effect of a 5-day heatwave occurring on lags 0 to 4? You can assess this as the effect of hot temperature (e.g., 28 degrees C) during these lags, while the temperature remains warm but more mild (e.g., 20 degrees C) the rest of the month, against constant 20 degrees C for the whole month. What is the effect under the 'constant' model? How about the smooth function lag model?

Applied exercise: Example code

1. **Start with the simple model describing how are daily mortality counts are associated with a linear function of temperature. Include control for day of week and long-term / seasonal trends. Add to this model a term that describes the lagged effects of temperature up to a week following exposure, using a separate model term for each lag. What patterns do you detect in the lagged effects of temperature?**

We'll start by looking at distributed lag associations by fitting a model with separate terms for each of the lags we want to fit. Here, we want to fit up to a

week, so we'll want eight terms in total: ones for the same day (X_0), the previous day (X_1), two days before (X_2), three days before (X_3), four days before (X_4), five days before (X_5), six days before (X_6), and seven days before (X_7).

Right now, our data have temperature lined up with the date of the recorded deaths:

```
obs %>%
  select(date, all, tmean) %>%
  head()
```

```
## # A tibble: 6 x 3
##   date         all tmean
##   <date>     <dbl> <dbl>
## 1 1990-01-01   220  3.91
## 2 1990-01-02   257  5.55
## 3 1990-01-03   245  4.39
## 4 1990-01-04   226  5.43
## 5 1990-01-05   236  6.87
## 6 1990-01-06   235  9.23
```

To add lagged temperature to the model, then, we'll need to add some columns to our data, so that the row for an observation includes not only the temperature on that day, but also on each of the seven previous days. The `lag` function from the `tidyverse` suite of packages can be used to create these columns:

```
obs <- obs %>%
  mutate(tmean_1 = lag(tmean, n = 1),
         tmean_2 = lag(tmean, n = 2),
         tmean_3 = lag(tmean, n = 3),
         tmean_4 = lag(tmean, n = 4),
         tmean_5 = lag(tmean, n = 5),
         tmean_6 = lag(tmean, n = 6),
         tmean_7 = lag(tmean, n = 7))
```

You can see below that each of these have taken `tmean` and then offset it by moving it down by one or more rows (down one row for lag 1, two for lag 2, etc.). As a result, we now have columns that give us today's temperature, yesterday's, and so on, all lined up with the row with the observation for daily mortality:

```
obs %>%
  select(date, all, tmean, tmean_1:tmean_7) %>%
  head()
```

```
## # A tibble: 6 x 10
##   date         all tmean tmean_1 tmean_2 tmean_3 tmean_4 tmean_5 tmean_6 tmean_7
##   <date>     <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>
## 1 1990-01-01   220  3.91   NA      NA      NA      NA      NA          NA      NA
## 2 1990-01-02   257  5.55    3.91   NA      NA      NA      NA          NA      NA
```

```
## 3 1990-01-03   245  4.39    5.55    3.91   NA      NA      NA          NA      NA
## 4 1990-01-04   226  5.43    4.39    5.55    3.91   NA      NA          NA      NA
## 5 1990-01-05   236  6.87    5.43    4.39    5.55    3.91   NA          NA      NA
## 6 1990-01-06   235  9.23    6.87    5.43    4.39    5.55    3.91       NA      NA
```

You'll notice that this process has created some missing data right at the beginning of the dataset—we don't know what the temperature was the day before Jan. 1, 1990, for example, since the study data starts on Jan. 1, 1990. With most time series, we'll have plenty of days in the study period, so we'll usually just allow these early dates to drop when the model is fit.

Now we'll add all these terms to our regression model:

```
dist_lag_mod_1 <- glm(all ~ tmean + tmean_1 + tmean_2 + tmean_3 + tmean_4 +
                        tmean_5 + tmean_6 + tmean_7 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

If you look at the output from the model, you can see that a regression coefficient has been fit for each of these distributed lag terms:

```
dist_lag_mod_1 %>%
  tidy() %>%
  filter(str_detect(term, "tmean"))
```

```
## # A tibble: 8 x 5
##   term     estimate std.error statistic  p.value
##   <chr>       <dbl>     <dbl>     <dbl>    <dbl>
## 1 tmean    0.00749   0.000592    12.6   2.67e-36
## 2 tmean_1 -0.00255   0.000788    -3.23  1.25e- 3
## 3 tmean_2 -0.00152   0.000797    -1.91  5.68e- 2
## 4 tmean_3 -0.00243   0.000798    -3.05  2.31e- 3
## 5 tmean_4 -0.00106   0.000798    -1.32  1.86e- 1
## 6 tmean_5 -0.000515  0.000797    -0.645 5.19e- 1
## 7 tmean_6 -0.00107   0.000789    -1.36  1.73e- 1
## 8 tmean_7 -0.00209   0.000593    -3.52  4.30e- 4
```

We can pull out all those estimates, calculate the 95% confidence intervals (do this *before* you take the exponential to get to the relative risk estimate, not after!), and then exponentiate all the terms to get estimates of relative risk of mortality per unit increase in temperature.

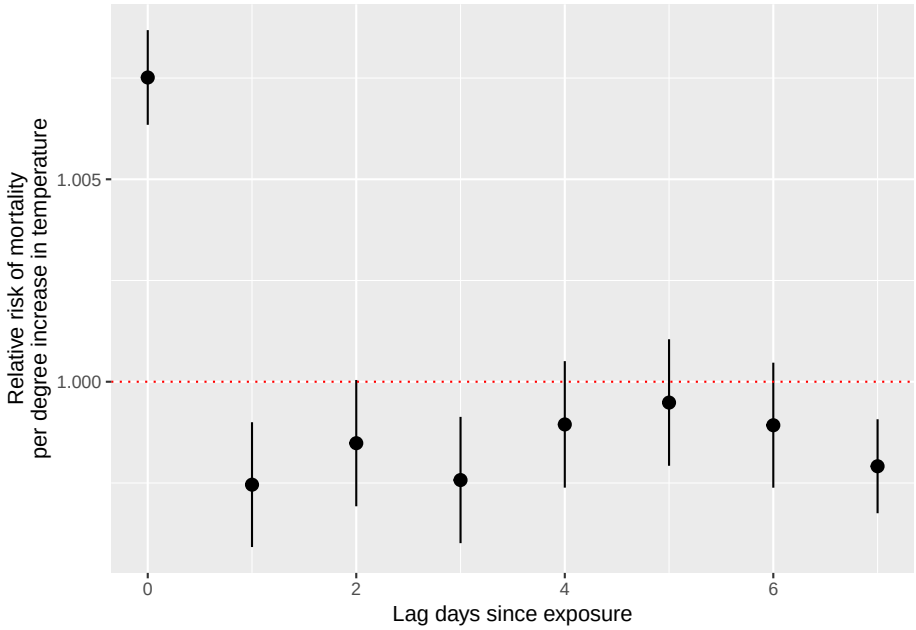
```
dist_lag_terms <- dist_lag_mod_1 %>%
  tidy() %>%
  filter(str_detect(term, "tmean")) %>%
  mutate(lag = 0:7,
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
```

```
         rr = exp(estimate),
         low_rr = exp(low_ci),
         high_rr = exp(high_ci)) %>%
  select(term, lag, rr, low_rr, high_rr)
dist_lag_terms
```

```
## # A tibble: 8 x 5
##   term      lag    rr low_rr high_rr
##   <chr>   <int> <dbl>  <dbl>   <dbl>
## 1 tmean       0 1.01   1.01    1.01
## 2 tmean_1     1 0.997  0.996   0.999
## 3 tmean_2     2 0.998  0.997   1.00
## 4 tmean_3     3 0.998  0.996   0.999
## 5 tmean_4     4 0.999  0.997   1.00
## 6 tmean_5     5 0.999  0.998   1.00
## 7 tmean_6     6 0.999  0.997   1.00
## 8 tmean_7     7 0.998  0.997   0.999
```

You can plot these estimates to explore how the association changes by lag (notice how `geom_pointrange` is very helpful here!):

```
dist_lag_terms %>%
  ggplot(aes(x = lag, y = rr)) +
  geom_point() +
  geom_pointrange(aes(ymin = low_rr, ymax = high_rr)) +
  labs(x = "Lag days since exposure",
       y = "Relative risk of mortality\nper degree increase in temperature") +
  geom_hline(yintercept = 1, color = "red", linetype = 3)
```



You can also, as an alternative, use the `dlnm` package to fit this model, using the `crossbasis` function to set up the distributed lag basis to include in the regression, and then the `crosspred` function to extract results after fitting the regression model and to plot results.

First, you'll again use `crossbasis` to create the basis, as you did earlier when using this package to fit a non-linear function of temperature. You will use the `lag` argument to say the maximum lag you want to include—in this case, seven days (`lag = 7`). You can use the `argvar` and `arglag` functions to specify the basis function for the exposure variable and the distributed lag component. For right now, we'll use a very basic linear function for the exposure (temperature), using `fun = "lin"` within the `argvar` list of arguments. To fit a separate coefficient for each lag, we can specify `fun = "integer"` in the `arglag` list of arguments (later we'll explore some other functions for the lag component).

```
library(dlnm)
dl_basis <- crossbasis(obs$tmean, lag = 7,
                       argvar = list(fun = "lin"),
                       arglag = list(fun = "integer"))
```

If you take a peak at this crossbasis, you'll notice that it's doing something similar to what we did when we added columns for lagged temperatures. The first column gives temperature on the same day, the second on the previous day (and so is offset one down from the first column), and so on. The crossbasis function deals with the missing early dates by setting *all* column values to be missing on those days; in practice, this difference in set up won't create any

difference in how the model is fit, as the `glm` function will by default exclude any observations with *any* columns of missing data.

```
dl_basis %>%
  head(n = 12)
```

```
##           v1.l1     v1.l2    v1.l3    v1.l4    v1.l5    v1.l6    v1.l7    v1.l8
##  [1,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [2,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [3,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [4,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [5,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [6,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [7,]        NA        NA       NA       NA       NA       NA       NA       NA
##  [8,]  7.958644  6.686216 9.232628 6.867855 5.431046 4.385564 5.547919 3.913589
##  [9,]  7.268950  7.958644 6.686216 9.232628 6.867855 5.431046 4.385564 5.547919
## [10,]  9.510644  7.268950 7.958644 6.686216 9.232628 6.867855 5.431046 4.385564
## [11,] 10.424808  9.510644 7.268950 7.958644 6.686216 9.232628 6.867855 5.431046
## [12,]  7.368595 10.424808 9.510644 7.268950 7.958644 6.686216 9.232628 6.867855
```

We can use this matrix of crossbasis variables now in our regression call:

```
dist_lag_mod_2 <- glm(all ~ dl_basis +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

Again, you can pull the regression coefficients directly out of the regression model object (and you can see how they're identical to those from our first distributed lag model):

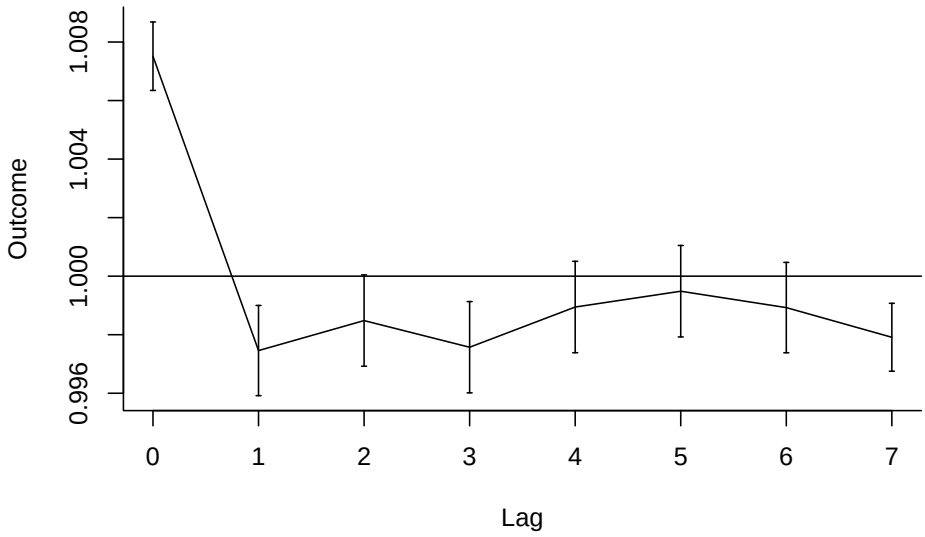
```
dist_lag_mod_2 %>%
  tidy() %>%
  filter(str_detect(term, "dl_basis"))
```

```
## # A tibble: 8 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 dl_basisv1.l1  0.00749   0.000592    12.6   2.67e-36
## 2 dl_basisv1.l2 -0.00255   0.000788    -3.23  1.25e- 3
## 3 dl_basisv1.l3 -0.00152   0.000797    -1.91  5.68e- 2
## 4 dl_basisv1.l4 -0.00243   0.000798    -3.05  2.31e- 3
## 5 dl_basisv1.l5 -0.00106   0.000798    -1.32  1.86e- 1
## 6 dl_basisv1.l6 -0.000515  0.000797    -0.645 5.19e- 1
## 7 dl_basisv1.l7 -0.00107   0.000789    -1.36  1.73e- 1
## 8 dl_basisv1.l8 -0.00209   0.000593    -3.52  4.30e- 4
```

You can proceed from here to plot estimates by lag using ggplot, but you can also use some of the special functions for plotting in `dlnm`. We can plot the

"slice" of comparing the relative risk at 1 degree C (`var = 1`) (this will give us the one-unit increase, because by default the crossbasis set-up with 0 degrees as the baseline, so this is one degree above that value). The `crosspred` function predicts based on two things: (1) a crossbasis and (2) a regression model object that uses that crossbasis. Once you create an object with `crosspred`, then the `plot` function has a special method for plotting that object (to see the helpfile for this plot method, type `?plot.crosspred` in your R console).

```
crosspred(dl_basis, dist_lag_mod_2) %>%
  plot(ptype = "slices", var = 1, ci = "bars")
```



However, even though mechanically we can fit this model, we should think about whether it's a good idea, or whether we should contrain the lag function some. If you look at the correlations between temperature at different lags, you'll see they are very strongly correlated:

```
obs %>%
  select(tmean, tmean_1:tmean_7) %>%
  cor(use = "complete.obs")
```

```
##             tmean   tmean_1   tmean_2   tmean_3   tmean_4   tmean_5   tmean_6
## tmean   1.0000000 0.9431416 0.8845767 0.8462589 0.8215974 0.8031790 0.7891306
## tmean_1 0.9431416 1.0000000 0.9431447 0.8846236 0.8463419 0.8216464 0.8032469
## tmean_2 0.8845767 0.9431447 1.0000000 0.9431403 0.8846198 0.8463094 0.8216348
## tmean_3 0.8462589 0.8846236 0.9431403 1.0000000 0.9431414 0.8846132 0.8463072
## tmean_4 0.8215974 0.8463419 0.8846198 0.9431414 1.0000000 0.9431482 0.8846132
## tmean_5 0.8031790 0.8216464 0.8463094 0.8846132 0.9431482 1.0000000 0.9431506
## tmean_6 0.7891306 0.8032469 0.8216348 0.8463072 0.8846132 0.9431506 1.0000000
## tmean_7 0.7781337 0.7892093 0.8032205 0.8216271 0.8463046 0.8846283 0.9431482
##           tmean_7
```

```
## tmean   0.7781337
## tmean_1 0.7892093
## tmean_2 0.8032205
## tmean_3 0.8216271
## tmean_4 0.8463046
## tmean_5 0.8846283
## tmean_6 0.9431482
## tmean_7 1.0000000
```

This means that we could have some issues with collinearity if we use this simpler distributed lag model, with a separate coefficient for each lag Next, we'll explore a model that constrains the lagged effects to follow a smooth function, to help avoid this potential instability in the modeling.

2. **Start with the model from the first question, but change it to use a smooth function of lag to detect lagged effects, rather than fitting a separate term for each lag. Extend the model to look at lagged effects up to a month following exposure. What patterns do you see in these lagged effects?**

As a reminder, we can stabilize our model of lagged effects a bit by fitting the lagged effects to be constrained to follow a smooth function, instead of fitting a separate term for each lag. The `dlnm` package makes this very easy to do.

When you use `crossbasis` to set up a crossbasis function, you can make a lot of choices to customize that crossbasis. The crossbasis combines a basis function for incorporating the exposure in the regression model (temperature in this case), as well as a basis function for incorporating lagged effects of that exposure. The `crossbasis` functions gives you several choices for each of these basis functions for the two dimensions of the crossbasis. Here, we'll look at using that flexibility for the lag dimension; in the next part of the exercise, we'll add some more complexity in the basis function for the exposure, as well.

To customize the basis function for lagged effects, you can use the `arglag` parameter in the `crossbasis` function. You send this parameter a list of other parameters, which get sent to a general `onebasis` function that builds that basis function (this will be convenient when we start working with the exposure basis, too—you'll see that you can use the same parameters in setting up each basis function).

To see the range of functions you can use for this crossbasis function, check the help function for `onebasis`. When we fit a separate coefficient for each lag, we used the "integer" function, which creates a set of basis variables with indicators for each separate lag (this is really similar to the idea of basis variables for a categorical variable, like day of the week). This is what we did in the first part of the exercise.

Now, we want to instead use a smooth function, with fewer degrees of freedom (i.e., fewer columns created in the matrix of basis values). Two popular options

for that function are the "poly" function, which will fit a polynomial, or the "ns" function, which will fit a natural cubic spline (just like we did earlier to create a non-linear function of temperature). We'll use "ns" in this example (`fun = "ns"` in the list of parameters for `arglag`).

For some of these functions, you'll need to specify additional parameters to clarify how they should be built. For example, if you're using a spline, you need to specify how many degrees of freedom it should have with `df`. We'll use 4 degrees of freedom, but you can experiment with different values for this and see how it affects the shape of the resulting function (i.e., the plot we create later). If you were fitting a polynomial function, you'd need to specify the degree of the polynomial; if you were fitting strata (i.e., constant values within sets of lags), you'd need to specify the breaks for dividing those strata; if you were using a threshold function, you'd need to specify the threshold, and so on. You can find more details on these options in the helpfile for `onebasis`; in general, the `onebasis` function is using other underlying functions like `ns` and `poly` to build these functions, so the choice of parameters will depend on the parameters for those underlying functions.

Here is the final call for building this new crossbasis (note that we're continuing to use a linear function of temperature, and at this point still only looking at lags up to a week; later in this question we'll expand to look at longer lags):

```
dl_basis_2 <- crossbasis(obs$tmean, lag = 7,
                         argvar = list(fun = "lin"),
                         arglag = list(fun = "ns", df = 4))
```

If you look at this crossbasis object, you'll see that it's created a matrix of basis variables, similar to the last part of the exercise. However, instead of having the same number of columns as the number of lags (8), we now have fewer columns. We have 4, reflecting the degrees of freedom we specified for the spline function of lag. Therefore, we've constrained the lag function a bit compared to the previous part of the exercise. Again, the values are all missing for the first seven days, since the data isn't able to capture the lagged temperature for these days.

```
dl_basis_2 %>%
  head(n = 12)
```

```
##          v1.l1    v1.l2    v1.l3      v1.l4
##  [1,]       NA       NA       NA         NA
##  [2,]       NA       NA       NA         NA
##  [3,]       NA       NA       NA         NA
##  [4,]       NA       NA       NA         NA
##  [5,]       NA       NA       NA         NA
##  [6,]       NA       NA       NA         NA
##  [7,]       NA       NA       NA         NA
##  [8,] 10.53142 5.938107 17.26767 -2.9117904
```

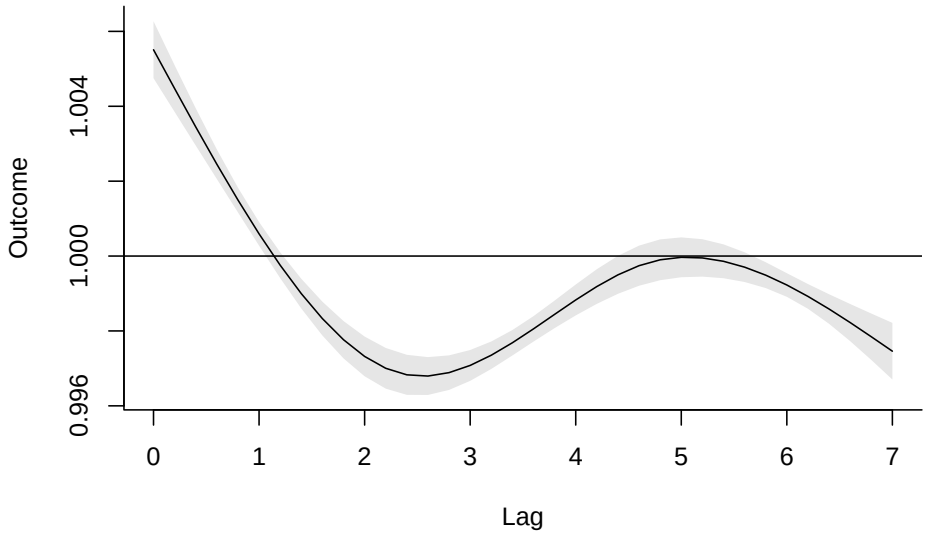
```
##  [9,] 11.46186 7.311815 17.99659 -2.0873124
## [10,] 10.91751 8.769682 19.69413 -3.2804334
## [11,] 10.94122 8.867758 22.33138 -2.7169313
## [12,] 13.00391 8.984807 22.20899 -0.3726342
```

Again, once we build the crossbasis function, we can put it in the regression equation and fit a regression model using `glm`:

```
dist_lag_mod_3 <- glm(all ~ dl_basis_2 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

Again, you can plot the results to see the pattern of associations by lag. The plot is just visualizing the results of predicting the model to certain values. By default, it will predict at each lag; this makes the plot a little "jumpy" if you're just looking at lags for a week or so, so to make it smoother, you might want to ask `crosspred` to predict to a finer resolution along the lag dimension (here, we're using `bylag = 0.2`).

```
crosspred(dl_basis_2, dist_lag_mod_3, at = 1, bylag = 0.2) %>%
  plot(ptype = "slices", var = 1)
```

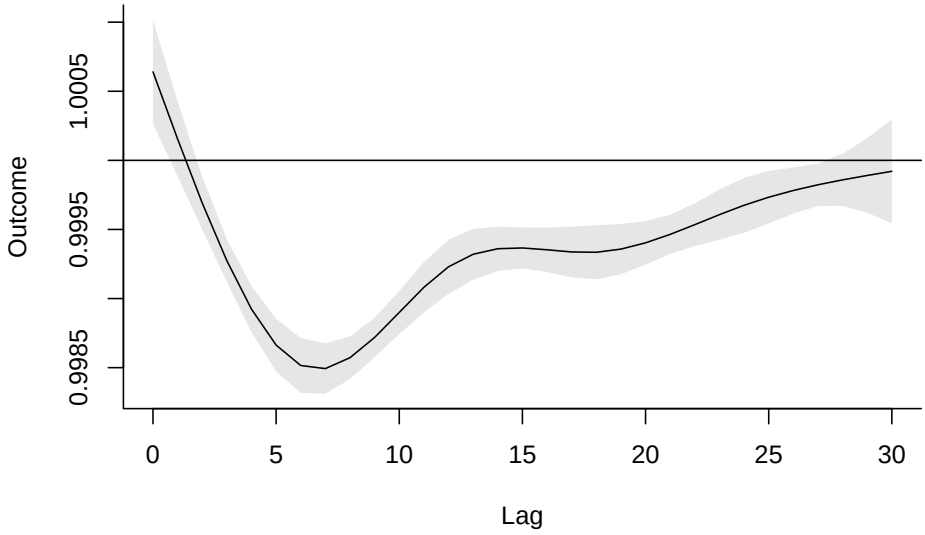


You can go back to the `crossbasis` call and try playing around with the degrees of freedom for the spline, to see how that affects the shape of this function. Be sure, though, that you don't use more degrees of freedom than there are lags (i.e., more than 8).

Now let's try to extend the model out to look at up to a month after exposure. To do this, all you need to do is change the `lag` argument in `crossbasis` to the longest lag you want to include. Since we're adding more lags, you'll likely

also want to increase the degrees of freedom you use for the basis function of lag; here, I'm using 6, but you can explore other values as well (again, don't use more than the number of lags, though). From there, the process of fitting the model and plotting the results is identical.

```
dl_basis_3 <- crossbasis(obs$tmean, lag = 30,
                         argvar = list(fun = "lin"),
                         arglag = list(fun = "ns", df = 6))
dist_lag_mod_4 <- glm(all ~ dl_basis_3 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
crosspred(dl_basis_3, dist_lag_mod_4, at = 1) %>%
  plot(ptype = "slices", var = 1)
```



3. **Refine your model to fit for a non-linear, rather than linear, function of temperature, while also incorporating a function to describe lagged effects up to a month following exposure. Based on this model, what does the association between temperature and mortality look like at lag 0 days? How about at lag 21 days? What does the pattern of lagged effects look like when comparing the relative risk of mortality at a hot temperature (e.g., 28 degrees C) to a milder temperature (7 degrees C)? What about when comparing mortality at a cold temperature (e.g., -4 degrees C) to a milder temperature (7 degrees C)? If you use a constant term for the effect of exposure over a month as opposed to the smooth function you just used, how does the overall cumulative RR compare?**

To extend the model to include temperature using a non-linear function, we'll

go back to the `crossbasis` call, but this time we'll make some changes to the exposure variable dimension. The basis function for the exposure is set using the `argvar` ("var" is for "variable") parameter. Again, this allows us to include a list of parameters that will be passed to `onebasis` to build a basis function and set up basis variables for this dimension of the crossbasis. We can use a spline for temperature by specifying `fun = "ns"`, as well as specify the degrees of freedom for that spline (here, I've used 4, but again, feel free to experiment):

```
dl_basis_4 <- crossbasis(obs$tmean, lag = 30,
                         argvar = list(fun = "ns", df = 4),
                         arglag = list(fun = "ns", df = 6))
```

Try looking at the crossbasis from this function (you'll need to look past the 30th row or so, since the earliest rows will all be missing since they're in that range where they don't have lag data available for the exposure). You can see that we now have a *lot* of columns: specifically, we have 4 (degrees of freedom for the temperature spline) times 6 (degrees of freedom for the lag spline) = 24 columns.

```
dl_basis_4 %>%
  as.data.frame() %>%
  slice(28:38)
```

```
##       v1.l1    v1.l2    v1.l3    v1.l4    v1.l5       v1.l6      v2.l1
## 1        NA       NA       NA       NA       NA          NA         NA
## 2        NA       NA       NA       NA       NA          NA         NA
## 3        NA       NA       NA       NA       NA          NA         NA
## 4  2.068939 3.299498 3.111270 1.738184 2.541810 -0.51512317 -0.5552531
## 5  1.955177 3.314862 3.125558 1.820183 2.690445 -0.40693744 -0.5967078
## 6  1.886111 3.299149 3.148516 1.893007 2.766242 -0.32385143 -0.6273517
## 7  1.882405 3.248469 3.180797 1.927784 2.804239 -0.08305771 -0.6419354
## 8  1.852973 3.165440 3.220817 1.900001 2.905860  0.08447527 -0.6588751
## 9  1.757370 3.058913 3.263914 1.795344 3.175517  0.03686863 -0.7106452
## 10 1.760596 2.944202 3.299498 1.779203 3.259667 -0.16712630 -0.7084665
## 11 1.797646 2.838520 3.314862 1.698607 3.497240 -0.17786583 -0.7118631
##         v2.l2      v2.l3      v2.l4      v2.l5         v2.l6    v3.l1    v3.l2
## 1          NA         NA         NA         NA            NA       NA       NA
## 2          NA         NA         NA         NA            NA       NA       NA
## 3          NA         NA         NA         NA            NA       NA       NA
## 4  -0.7011824 -0.7707934 -0.4847858 -0.9916438 -0.0956588789 1.405249 1.838701
## 5  -0.6979016 -0.7603742 -0.4547662 -0.9667940 -0.0791413415 1.478133 1.830775
## 6  -0.7047091 -0.7492423 -0.4287099 -0.9503322 -0.0554076103 1.526744 1.841501
## 7  -0.7234011 -0.7371123 -0.4081381 -0.9450033 -0.0058689802 1.541114 1.873929
## 8  -0.7537142 -0.7241504 -0.4035141 -0.9205842  0.0358717107 1.565836 1.927170
## 9  -0.7930344 -0.7111578 -0.4381402 -0.7997450 -0.0007806944 1.643309 1.996194
## 10 -0.8364025 -0.7011824 -0.4373333 -0.7657041 -0.0621066847 1.644968 2.071685
## 11 -0.8780716 -0.6979016 -0.4735973 -0.6384769 -0.0958995727 1.631672 2.143093
```

```
##       v3.13    v3.14    v3.15       v3.16      v4.11     v4.12     v4.13
## 1        NA       NA       NA          NA         NA        NA        NA
## 2        NA       NA       NA          NA         NA        NA        NA
## 3        NA       NA       NA          NA         NA        NA        NA
## 4  1.955381 1.180876 2.353392  0.20605400 -0.8041945 -1.052250 -1.119024
## 5  1.942554 1.126503 2.294679  0.17194514 -0.8459049 -1.047714 -1.111683
## 6  1.926528 1.081359 2.257968  0.12135036 -0.8737239 -1.053852 -1.102512
## 7  1.906672 1.048474 2.249542  0.01133949 -0.8819473 -1.072411 -1.091149
## 8  1.883354 1.051489 2.198457 -0.07875894 -0.8960954 -1.102879 -1.077804
## 9  1.858659 1.116576 2.004017 -0.03652356 -0.9404317 -1.142380 -1.063672
## 10 1.838701 1.121782 1.943746  0.09297047 -0.9413808 -1.185582 -1.052250
## 11 1.830775 1.177720 1.763872  0.11448195 -0.9337721 -1.226447 -1.047714
##         v4.14     v4.15        v4.16
## 1          NA        NA           NA
## 2          NA        NA           NA
## 3          NA        NA           NA
## 4  -0.6757908 -1.346797 -0.117920408
## 5  -0.6446743 -1.313197 -0.098400617
## 6  -0.6188389 -1.292188 -0.069446282
## 7  -0.6000200 -1.287366 -0.006489356
## 8  -0.6017453 -1.258131  0.045072099
## 9  -0.6389933 -1.146857  0.020901671
## 10 -0.6419722 -1.112366 -0.053205061
## 11 -0.6739847 -1.009427 -0.065515636
```

The `dlnm` package has done the work of setting up this crossbasis function, as well as creating this matrix of basis variables from our original simple column of temperature measurements. This is getting to be a much more complex function than the simple linear/unconstrained lag function that we started with. However, the underlying principles and mechanics are the same, they're just scaling up to allow some more complex "shapes" in our exposure-response association.

We can add this new crossbasis function to the regression model, and then predict from the crossbasis and regression model to get a view of how temperature is (non-linearly) associated with mortality risk at different lags, comparing to a baseline temperature of 7 degrees C:

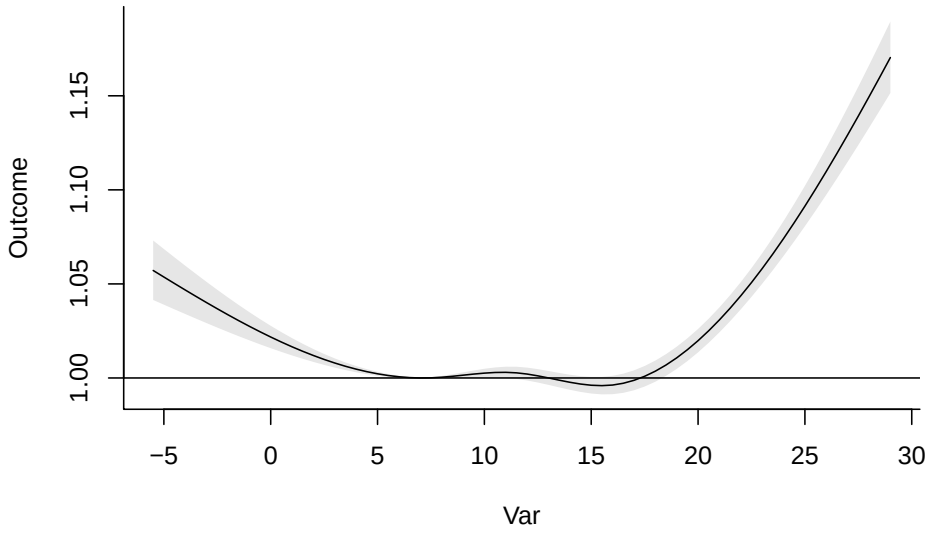
```
dist_lag_mod_5 <- glm(all ~ dl_basis_4 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")

cp_dl <- crosspred(dl_basis_4, dist_lag_mod_5, cen = 7, bylag = 1)
```

The object created by `crosspred` includes a *lot* of output (you can check out the whole thing by running `cp_dl` to print it out). We can extract elements from this object (for example, if we want to get effect estimates or estimates

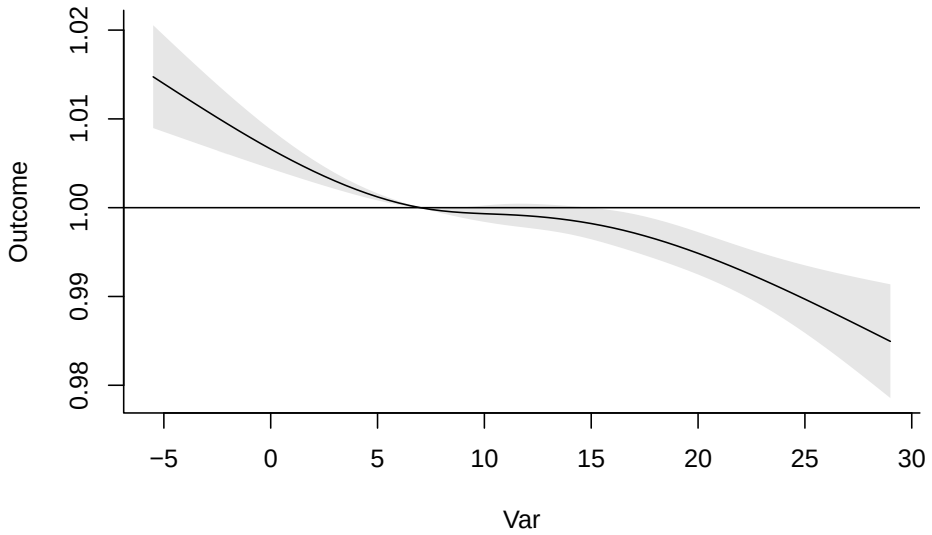
of confidence intervals). We can also create a variety of plots from this object. The following plots all use the `plot` method for `crosspred` objects, so you can access the helpfile with `?plot.crosspred`.

First, we can look at "slices," where we focus on just one lag value and look at the non-linear relationship between temperature and mortality risk at that lag only. For example, here are plots showing the relationship between temperature and mortality risk on the same day of exposure (lag 0) and 21 days after exposure (lag 21):

```
plot(cp_dl, ptype = "slices", lag = 0)
```



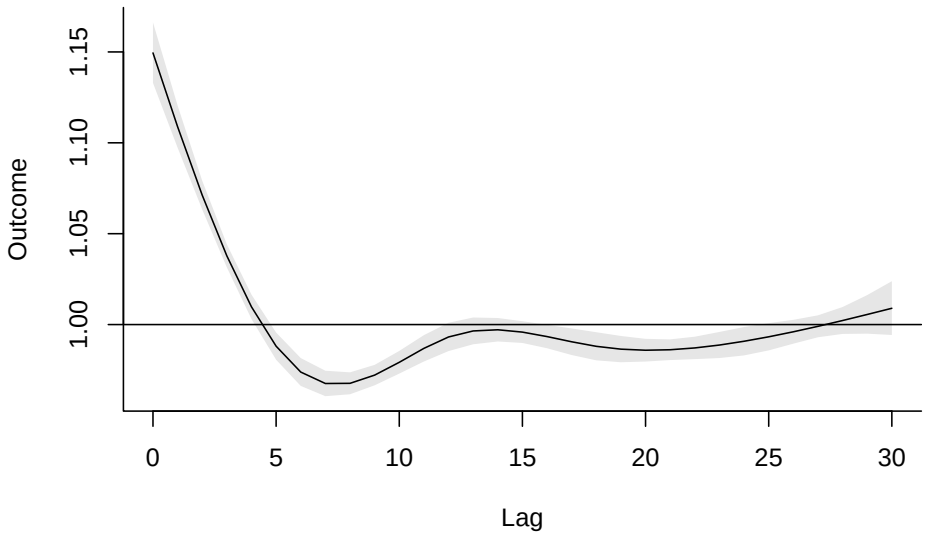

```
plot(cp_dl, ptype = "slices", lag = 21)
```



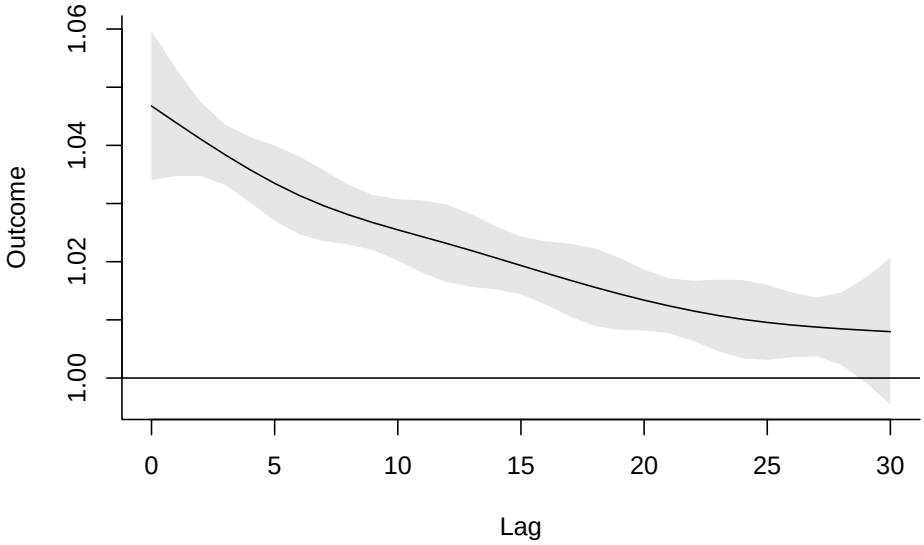
You can see that heat has a stronger relationship with mortality risk than cold at lag 0, while at a longer lag, there continues to be a positive association between cold temperature and mortality risk, but now the relationship with heat has shifted to be negative.

You can do the same idea of looking at a "slice" for a specific temperature, in this case seeing how effects vary across lags when comparing that temperature to the baseline temperature (7 degrees C, which we set when we ran `crosspred`). For example, here are "slices" for 28 degrees C and -4 degrees C:

```
plot(cp_dl, ptype = "slices", var = 28)
```



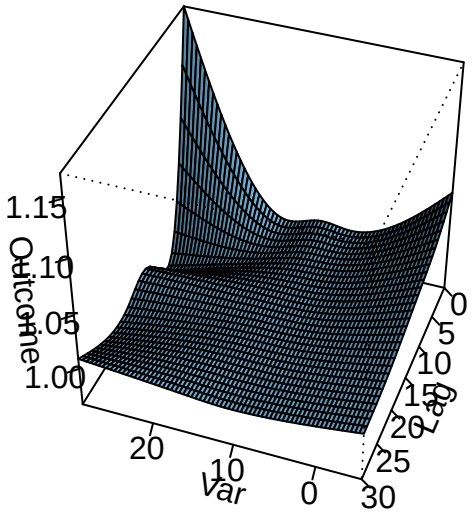

```
plot(cp_dl, ptype = "slices", var = -4)
```



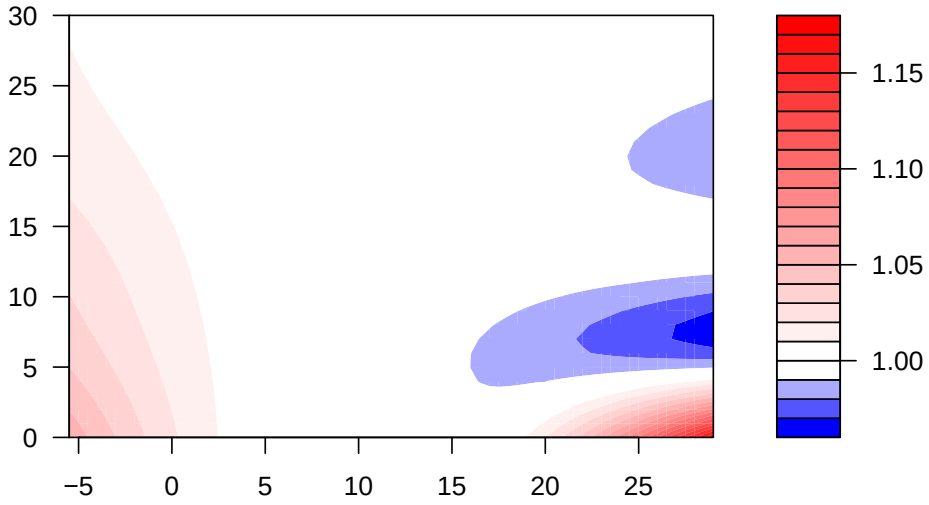
These plots again reinforce that heat's association with mortality risk is more immediate, while cold has an association that plays out over a much longer lag following exposure.

There are also some plots you can make from the `crosspred` object that help show the full, three-dimensional association between the exposure, lag time, and risk of the outcome. For example, the default plot created from the `crosspred` object is a three-dimensional plot (the `theta`, `phi`, and `lphi` parameters shift the angle we view it from), while the "contour" plot type shows us a contour plot of these results (in other words, showing the exposure and lag variables in two dimensions of a two-dimensional plot then showing relative risk using color for each combination). You can think of the 'sliced' plots from above as two-dimensional slices of the three-dimensional graph if we slice right at the selected `lag` or `var` value. Here is the code to create those plots:

```
plot(cp_dl, theta = 200, phi = 40, lphi = 30)
```




```
plot(cp_dl, ptype = "contour")
```



To see how a 'constant' lag-response model compares to the smooth function from this model we have to go back to the `crossbasis` and use the appropriate argument in the `arglag` parameter:

```
dl_basis_5 <- crossbasis(obs$tmean, lag = 30,
                         argvar = list(fun = "ns", df = 4),
                         arglag = list(fun = "strata", df = 1))
```

The function `strata` in `arglag` essentially breaks up the lag window into strata equaling the degrees of freedom specified, and the lag-response is assumed to be constant for each day within each stratum. Here we specified `df=1` so the lag-response is assumed to be constant for the entire lag window. In other words,

the exposure in each day in the lag window is constrained to have the same effect. If we look at the `crossbasis` from this function you'll see that we have far fewer columns. Remember that the total number of columns is determined by the total number of degrees of freedom, which is the product of the df's in each of `argvar` and `arglag` so in this case 4.

```
dl_basis_5 %>%
  as.data.frame() %>%
  slice(28:38)
```

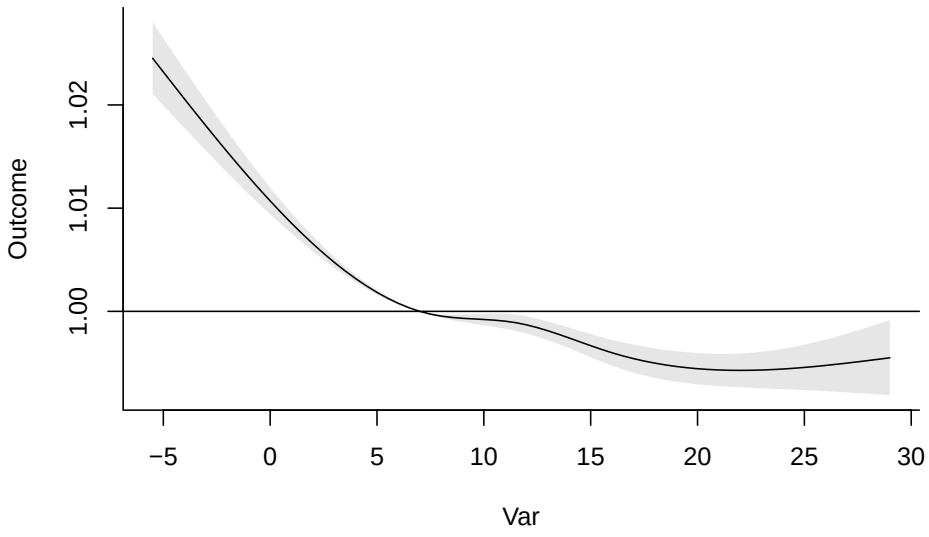
```
##       v1.l1     v2.l1     v3.l1     v4.l1
## 1        NA        NA        NA        NA
## 2        NA        NA        NA        NA
## 3        NA        NA        NA        NA
## 4  14.69900 -4.438015 10.919532 -6.249020
## 5  15.03355 -4.382485 10.789984 -6.174883
## 6  15.21075 -4.347038 10.707109 -6.127455
## 7  15.33913 -4.332385 10.672850 -6.107850
## 8  15.51220 -4.298983 10.594759 -6.063160
## 9  15.89771 -4.127076 10.317738 -5.904626
## 10 15.96057 -4.094014 10.267995 -5.876159
## 11 16.37863 -3.880472  9.958228 -5.698886
```

We can now run the same model as above replacing the `crossbasis` with the one we just created and repeat the prediction to see how the temperature effect comparing to a baseline temperature of 7 degrees C changes when we constrain the lag-response to be constant.

```
dist_lag_mod_6 <- glm(all ~ dl_basis_5 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")

cp_dlconstant <- crosspred(dl_basis_5, dist_lag_mod_6, cen = 7, bylag = 1)
```

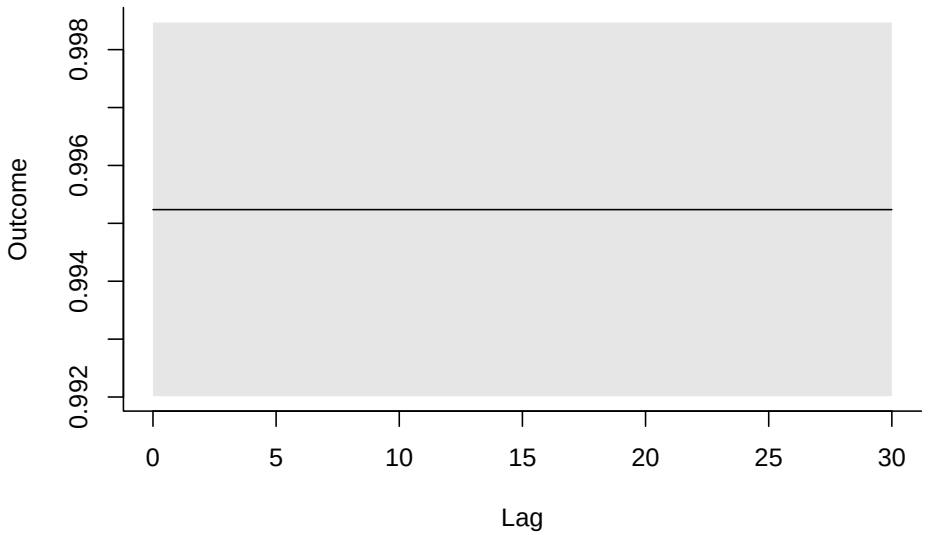
If we look at the same plots and sliced plots as above we can see that this is a much simple model.

```
plot(cp_dlconstant, ptype = "slices", lag = 0)
plot(cp_dlconstant, ptype = "slices", lag = 21)
```

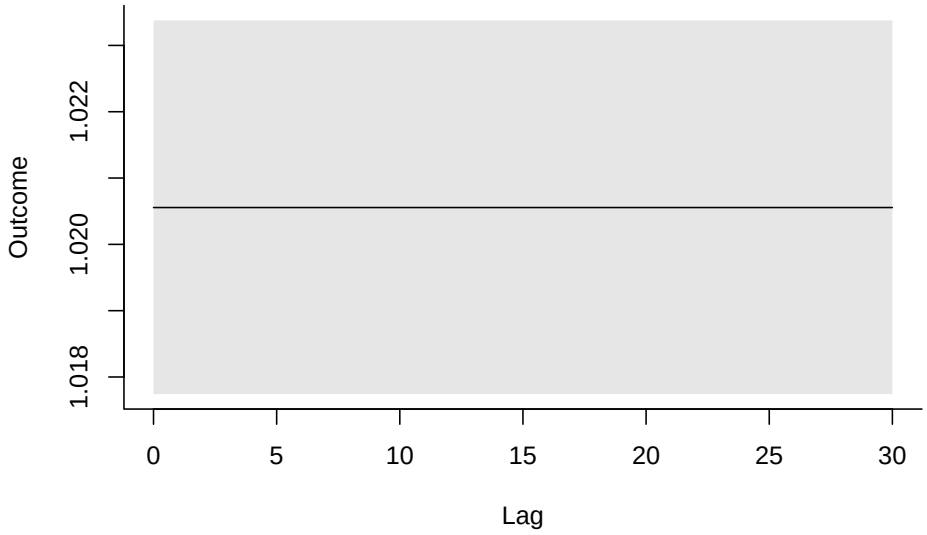


The two plots are identical (because we assume that the exposure-response is the same at all lags) with colder temperatures having a negative effect, while the warmer temperatures appear to be protective.

```
plot(cp_dlconstant, ptype = "slices", var = 28)
```



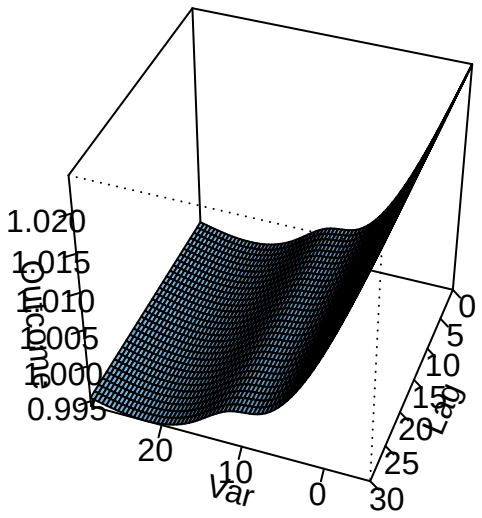

```
plot(cp_dlconstant, ptype = "slices", var = -4)
```



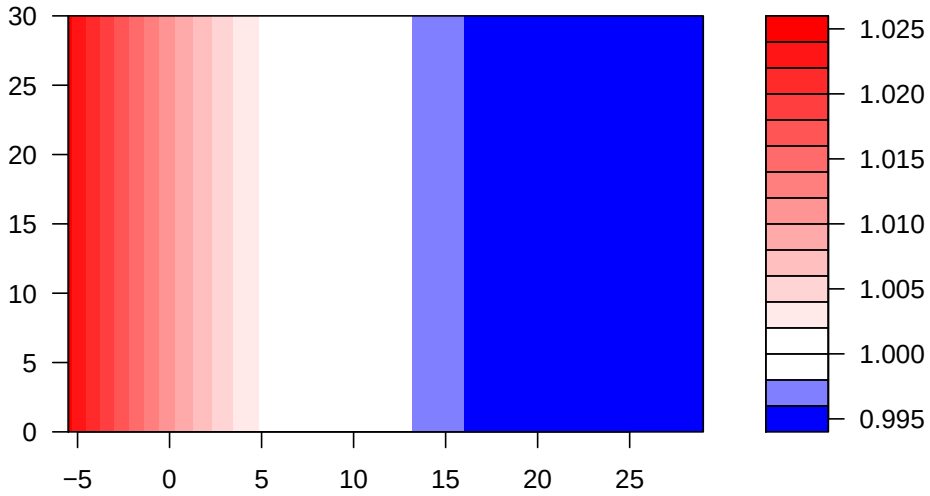
Here we see the constant lag-response more clearly, but at different effects depending on the set temperature. The hot temperature appears to be mildly protective (RR < 1.0), while the cold temperature is harmful (RR > 1.0).

The reduced complexity is evident in the 3-d plot (which looks "flat" on one of the dimensions) and the contour plot (which has large rectangular areas) as well.

```
plot(cp_dlconstant, theta = 200, phi = 40, lphi = 30)
```



```
plot(cp_dlconstant, ptype = "contour")
```



One other thing that we can do with the model output is we can estimate the cumulative risk across lags, for example for a month of hot temperature (28 degrees C) compared to cooler temperature (7 degrees C). We do this by essentially accumulating the RRs from each lag-day as follows. The crosspred object has estimates of the coefficients for each lag day (in the `matfit` element), and you could add these up yourself, or you can extract a difference element (`allfit`) that has already added these up for you to get a cumulative effect estimate. If you'd like to calculate confidence intervals, you can get an estimate of the standard error for this cumulative estimate with the `allse` element of the crosspred object. We can get all these for a temperature of 28 degrees C; since we set the crosspred object to be centered at 7 degrees C, this will allow us to compare 28 to 7 degrees C:

```
### parameter for each lag-day
cp_dlconstant$matfit[cp_dlconstant$predvar==28]
```

```
##  [1] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
##  [6] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
## [11] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
## [16] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
## [21] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
## [26] -0.004774825 -0.004774825 -0.004774825 -0.004774825 -0.004774825
## [31] -0.004774825
```

```
### combined model parameter for the cumulative RR (this is the sum for all of the abo
cp_dlconstant$allfit[cp_dlconstant$predvar==28]
```

```
##         28
## -0.1480196
```

```
### corresponding standard error for the cumulative RR parameter
cp_dlconstant$allse[cp_dlconstant$predvar==28]
```

```
##          28
## 0.05128501
```

If we exponentiate the model parameter we will get the cumulative RR estimate and likewise using the model parameter and standard error we can derive 95% Confidence Intervals (CIs) for the cumulative RR. For the central estimate:

$$RR = exp(\beta) = exp(-0.148) = 0.86$$

For the confidence interval:

$$exp(\beta \pm 1.96 * se) = exp(-0.148 \pm 1.96 * 0.05) = (0.78, 0.95)$$

The `crosspred` object, it turns out, stores this information as well (in the `allRRfit`, `allRRlow`, and `allRRhigh` elements), so we can extract these elements directly without doing the math:

```
cp_dlconstant$allRRfit[cp_dlconstant$predvar==28]
```

```
##        28
## 0.8624142
```

```
cp_dlconstant$allRRlow[cp_dlconstant$predvar==28]
```

```
##        28
## 0.7799415
```

```
cp_dlconstant$allRRhigh[cp_dlconstant$predvar==28]
```

```
##        28
## 0.9536078
```

Does this look accurate? Do we expect a 14% decrease in mortality comparing a very hot month to a cool month?

Now let's calculate the cumulative RR for temperature at 28 degrees for the smooth lag-response function model from before (i.e., pull those same elements from the model we fit before with a smoother lag-response function):

```
cp_dl$allRRfit[cp_dl$predvar==28]
```

```
##       28
## 1.078548
```

```
cp_dl$allRRlow[cp_dl$predvar==28]
```

```
##        28
## 0.9768645
```

```
cp_dl$allRRhigh[cp_dl$predvar==28]
```

```
##       28
## 1.190816
```

Based on this model, we infer there's about a 6% cumulative increase associated with a temperature of 28 degrees C compared to 7 degrees.

Does this look more accurate? Which of the two lag-response functions is more likely to be true?

4. **Assume that we are interested in the potential effect of a 5-day heatwave occurring on lags 0 to 4? You can assess this as the effect of hot temperature (e.g., 28 degrees C) during these lags, while the temperature remains warm but more mild (e.g., 20 degrees C) the rest of the month, against constant 20 degrees C for the whole month. What is the effect under the 'constant' model? How about the smooth function lag model?**

Using the `crosspred` function we can compare any exposure history over a lag-window of interest to a reference value (or values varying over time in the same window). Realistically, a whole month of extremely high exposures is unlikely. Extreme phenomena (such as 28 degree mean daily temperature in London) are more likely to last a shorter period of time. Furthermore, the effect of an event like a heat wave (likely occurring in the summer) is probably better assessed with a referent value more akin to what the heat wave is replacing, i.e., a warm but more mild summer temperature rather than temperatures more likely to occur in other seasons. We saw above that the effect of hot temperatures tends to be more short term, so let's assess the potential effect of a 5-day heatwave assessed on the last day of the heatwave. The rest of the days in the month are at 20 degrees C and we will use the same value as the referent value for the whole month. Realistically there will be variability in the temperatures during these days as well, but for the purposes of this exercise let's keep it constant at 20 for simplicity. Let's assess this effect using the constant lag-response model.

```
hw_constant <- crosspred(dl_basis_5, dist_lag_mod_6,
                         at = exphist(c(rep(20, 26), rep(28, 5)), times = 31, lag = 30)
                         cen = 20, bylag = 1)
```

The `exphist` function allows us to designate distinct exposure values for the entire lag-window. The exposure values here appear in an oldest (lag30) to most recent (lag0) order. The `times` argument determines were the list of exposure begins in the lag window (here `time = 31` or lag30 from 0-30). The effect estimate and corresponding CI can be extracted as above.

```
hw_constant$allRRfit
```

```
##       31
## 1.004005
```

```
hw_constant$allRRlow
```

```
##        31
## 0.9891951
```

```
hw_constant$allRRhigh
```

```
##       31
## 1.019037
```

Now, lets repeat this with the smooth function lag-response model:

```
hw_smooth<-crosspred(dl_basis_4, dist_lag_mod_5,
                     at = exphist(c(rep(20, 26), rep(28, 5)), times = 31, lag = 30) ,
                     cen = 20, bylag = 1)

hw_smooth$allRRfit
```

```
##       31
## 1.402728
```

```
hw_smooth$allRRlow
```

```
##       31
## 1.356389
```

```
hw_smooth$allRRhigh
```

```
##       31
## 1.450651
```

We see that the constant lag-response model underestimates the RR and does not capture any potential effect of a heatwave, while the smooth term lag-response model indicates that there is a 40% increase in mortality for the last day of the heatwave. Note that this increase in mortality is not the only effect, as based on out model the heatwave started having an effect on the same day it started, i.e. we would expect all 5 heatwave days to have a temperature related increase in mortality compared to the days before the heat-wave. We can look at the increase in mortality for each day if we look at the individual effect estimates:

```
### individual lag-day parameters
hw_smooth$matfit
```

```
##         lag0       lag1       lag2       lag3       lag4 lag5 lag6 lag7 lag8
## 31 0.1196662 0.09213574 0.06563143 0.04117951 0.01980617    0    0    0    0
```

```
##    lag9 lag10 lag11 lag12 lag13 lag14 lag15 lag16 lag17 lag18 lag19 lag20 lag21
## 31    0     0     0     0     0     0     0     0     0     0     0     0     0
##    lag22 lag23 lag24 lag25 lag26 lag27 lag28 lag29 lag30
## 31     0     0     0     0     0     0     0     0     0
```

```
#### individual lag-day RRs
hw_smooth$matRRfit
```

```
##        lag0     lag1     lag2     lag3     lag4 lag5 lag6 lag7 lag8 lag9 lag10
## 31 1.127121 1.096514 1.067833 1.042039 1.020004    1    1    1    1    1     1
##    lag11 lag12 lag13 lag14 lag15 lag16 lag17 lag18 lag19 lag20 lag21 lag22
## 31     1     1     1     1     1     1     1     1     1     1     1     1
##    lag23 lag24 lag25 lag26 lag27 lag28 lag29 lag30
## 31     1     1     1     1     1     1     1     1
```

The first day of the heatwave will have an effect corresponding to a lag0 effect:

$$RR = exp(0.107) = 1.11$$

The second day of the heatwave will have an effect corresponding to a lag0 and lag1 effect:

$$RR = exp(0.107 + 0.085) = 1.21$$

(This quantity is also equal to the product of the lag-day specific RRs for lag0 and lag1:)

$$1.11 \times 1.09 = 1.21$$

And so on, until we reach our last day of the heatwave with RR=1.39. There is a bidirectional relationship for this RR, as you can interpret this as the overall effect of the heatwave on mortality on the last day of the heatwave, or as the overall effect of temperature on the first day of the heatwave on mortality over the next 5 days.

Chapter 5

Natural experiments

5.1 Readings

The readings for this chapter are:

- Bernal et al. (2017) (on interrupted time series), with a correction to an equation in the paper at https://academic.oup.com/ije/article/49/4/1414/5900884. Example data and R code for the paper are available to download through a Supplemental Appendix. (Also, note that there is a correction of one of the statistical model equations in this paper, given at the same web address.)
- Bor et al. (2014) A systematic review of regression discontinuity designs in epidemiological research
- Lopez Bernal et al. (2013) An example of an epidemiological study that uses an interrupted time series study design. Includes a nice discussion of strengths and limitations of the study design.

The following are supplemental readings (i.e., not required, but may be of interest) associated with the material in this chapter:

- Barone-Adesi et al. (2011) The scientific paper highlighted as an example in the tutorial from the required reading
- Sargent et al. (2004) Another paper providing an example of an interrupted time series design to study the health impacts of a smoking ban
- Casey et al. (2018) A paper that leverages difference-in-differences, also in the context of a natural experiment
- Mendola (2018) An Invited Commentary on the previous reading
- Venkataramani et al. (2016) A good overview of regression discontinuity designs in health-related research
- Turner et al. (2021) Provides reccomendations on making graphs to illustrate interrupted time series data

- Linden (2017) An interesting analysis of challenges to validity when an interrupted time series study is conducted without controls (i.e., with a single group)
- Bottomley et al. (2019) Introduces a more complex approach for interrupted time series, where a control is included to help control for trends with time
- Dimick and Ryan (2014) A short overview of the difference-in-differences design
- Howe et al. (2016) An example of a study that uses difference-in-differences, in this case to study the role of Pokemon GO on physical activity
- Maciejewski and Basu (2020) A short overview of the regression discontinuity design
- Haber et al. (ress) An overview of potential study designs for assessing COVID-19 policies, including interrupted time series and difference-in-differences

5.2 Natural experiments

If you want to test a scientific hypothesis—like whether a vaccine has a certain success rate in preventing infection—the ideal way to do it is to run a randomized controlled experiment. The "controlled" part of the experiment indicates that there will be some study subjects that get the treatment and some that instead will serve as a comparison by not being treated—that is, they will be controls. The randomized part of the experiment is that you will divide the study subjects into groups of treated and controls randomly, rather than through some method that could introduce confounding (like by assigning all study subjects in one city to treatment and all in other city to control).

The randomized controlled experiment is an extraordinary tool in science, as it allows a comparison of treated versus controls while in theory removing the risk of confounding by external factors. For an external factor to confound the estimate of an association, it must itself be associated with both the treatment (or exposure, in most cases for environmental epidemiology) and the outcome. By randomly distributing the study subjects into treatment or control, you (in theory) break any possibly association between external factors and the treatment. This ensures exchangeability between the treated and controls, and we can use one group as a proxy for the other. In other words the treated are a good indication of what would happen to the controls if they themselves were treated and the controls are a good inciation of what woud happn to the treated if they were assigned to be controls.

This is only "in theory" for a few reasons. First, unless you have huge numbers of study subjects, there is a reasonable chance that just by chance there will be some differences between treated and controlled groups in some external factors. For example, if you have ten study subjects, five female and five male, and you

randomly distribute them into two groups, it wouldn't be that unusual to have one group with eight females and two males and the other with two females and eight males, and so there would remain an association between biological sex and treatment among the randomized study subjects. This chance is why you will often see, in papers describing randomized controlled experiments (e.g., for clinical trials), tables that give distributions of several external factors for the treatment and control groups. As you have more and more study subjects, it will become less likely to have a large difference in external factors by treatment group. One other reason, however, that the distribution of treatment is only random in theory is that some people may not comply or otherwise carry through with their assigned treatment. This mechanism is a risk regardless of the size of the study population. This could result from a subject not complying with the assigned treatment, but could also result from something like a doctor taking a study subject off the assigned treatment for one reason or another. We will revisit the issue of non-compliance, intention-to-treat analysis in a later chapter (instrumental variables).

Randomized controlled experiments are therefore powerful (although not iron-clad). However, there are many cases in environmental epidemiology where we cannot use them. This is for both practical and ethical reasons. From a practical standpoint, for example, we may want to explore how tropical cyclones affect health risk. It is (currently) impossible to create a hurricane and "apply" it to a group of study subjects that you've assigned to be "exposed". From an ethical standpoint, many of the exposures we study in environmental epidemiology are harmful, and it can often be unethical to assign a study subject to an exposure that is known or thought to be harmful for the sake of science. (By contrast, in a clinical trial of a medication, there is a chance that the medication will have harmful side effects, but it's being tested because there is a chance it will improve the health of the study subjects.) There are some cases where randomized controlled experiments can be used in environmental epidemiology—for example, better designs of indoor cookstoves are hypothesized to improve health, and so could be tested in this way. However, for many research questions, environmental epidemiologists often rely on observational data to help answer the question, rather than data generated through a randomized controlled trial.

We have been using observational data in our examples up to this point, and we've explored how we can use a regression modeling framework to control for potential confounders. In other words, in these examples, since we were unable to use randomization of the treatment (i.e., exposure) to remove risk of confounding from external factors, we instead had to think very careful about which external factors might confound our estimate of the association and in what way, and then address those factors in the statistical modeling through the use of control variables and sensitivity analysis.

In this chapter, we'll look at another way that we can leverage observational data to test scientific hypotheses, and that's through natural experiments. To be clear, these two methods are not mutually exclusive; instead, you could build

studies where you incorporate both a natural experiment as well as control for factors that might still cause some residual confounding if not controlled for.

A natural experiment occurs when something happens that divides a group of potential study subjects into some who are treated (or exposed) and some who are not, and does so in a way where this division should be pretty close to random, at least in regards to most external factors that could be confounders. This division could be by space—for example, you could have an exposure that hits one geographic area but spares a neighboring area that is otherwise very comparable. This division could also be by time—you could have a policy that is implemented at a certain date, and so the study population immediately before that date was unexposed to this new policy while those after the date were exposed.

Typically, the "randomization" of treatment (or exposure) imposed by the natural experiment won't be perfect, and it usually won't even be as good as what could be done under a controlled randomized experiment. However, it is often good enough that it allows us to use observational data, which is much easier and cheaper to get at a large scale than data generated by a controlled experiment. This has the advantages of increased statistical power, and in many cases observational data that cover a large scale may offer improved external validity for the study results, since the data are typically collected in conditions that more closely reflect "real life" conditions.

In this chapter, we will focus on a study design called an *interrupted time series*. This study design can be applied in cases where there has been an abrupt change at a certain time point that would influence how treatment is assigned before and after that time point. For example, there might be a new law that prevents indoor smoking in restaurants. If that policy goes into effect on a specific date, then exposure to secondary smoke in the population would change abruptly to be lower (at least, on average across the population) after that date compared to before. As another example, if the workers at a factory that is a heavy polluter goes on strike (and so all equipment stops running) on a specific date, then the pollution exposure in that city might abruptly change from before the strike to during the strike, and then change again after the strike is over.

This study design belongs to a more general type of natural experiment study design called a "regression discontinuity" design. Broadly, a regression discontinuity design can be applied anytime there is a threshold at which the chance of treatment (or exposure) changes abruptly. Interrupted time series are limited to cases where this threshold comes at a specific time, and where you have collected time series data. However, there are plenty of set-ups that would create other regression discontinuities that could be exploited to answer a scientific hypothesis.

For example, some medications are prescribed based on a threshold. For example, a doctor might describe when to start drug treatment for HIV based on a certain threshold of white blood cells measured in the patient. In this case,

the white blood cell count is negatively associated with disease severity and so with many outcomes from the disease, so you might expect many outcomes to increase in risk as this count decreases. However, if the treatment is helpful in decreasing risk for those outcomes, then it may be that people with white blood cell counts that are just barely low enough to receive treatment have lower risk of those outcomes than people just barely too high in the counts to receive treatment. The people just barely above the threshold are likely pretty similar to those just barely below the threshold, and so a "discontinuity" in the risk of health outcomes that happens as you cross that threshold is likely caused by the treatment, which changes abruptly while other factors change minimally if at all. In the next section of this chapter, we will explore how you can use a GLM regression model in the context of interrupted time series (and by extension, for regression discontinuity study designs in general).

5.3 Interrupted time series

In this section, we'll take a closer look at interrupted time series study designs, focusing on how you can use statistical models to fit time series data and draw conclusions from these studies.

First, for these studies, we will think about exposure in a slightly different way. In these studies, we often will not directly have the exposure that we care about. For example, the data in Bernal et al. (2017) aims to improve knowledge of whether exposure to secondary smoke alters the acute risk of cardiovascular events. However, they do not have measures of secondary smoke exposure for all their study subjects. Instead, they know the date when a ban was introduced in the study location that banned indoor smoking in public places. We can assume that, on the date of this ban, people in this community had an abrupt reduction in their exposure to indoor secondary smoke.

As a reminder, interrupted time series designs are part of a more general group of designs, those for regression discontinuities. This general group of designs shares the characteristic that the exposure of interest is often not directly measured, but instead another factor is measured that, at a certain threshold, abruptly changes average exposure to the exposure we do care about. The changing factor (time in the case of an interrupted time series, white blood cell counts in the example given earlier on HIV treatment) is called the *forcing variable* in the study, and the point along this variable at which the exposure of interest abruptly changes is called the *threshold* or *threshold rule* ((Bor et al., 2014) has a great overview of regression discontinuity studies in epidemiology with examples of different possible forcing variables and threshold rules).

Applied: Fitting a statistical regression model for data from an interrupted time series study

We will use a new example dataset for this chapter—the dataset that comes with one of the required reading papers for this chapter, Bernal et al. (2017).

Make sure you download the file named "sicily.csv" that comes as a supplement to this paper.

1. Load the data into R and explore the dataset. Plot the time series of the data on the outcome of interest as it varies over the study period. Use a visual marker in the plot to show the intervention (the date the law was implemented).
2. Next, let's look at how the rates of the outcome changes right before and right after the ban. At this stage, we won't worry about any underlying time trends in the outcome rates. Instead, compare the outcome rates in two groups—the data points collected for the time points right before the ban, and then the data collected for time points right after the ban. Try a few windows for capturing "right before" and "right after" (e.g., 3 months, 6 months, 12 months). How does the precision of the estimate change as you change that window? How do you think the potential might change for residual confounding from long-term trends in the outcome rates irrespective of the ban?
3. Now see if you can fit a regression model that uses all the data, while using a regression term to allow for a long-term trend in the rates of the outcome. You may want to try out several of the model structures shown in Figure 2 of Bernal et al. (2017). Can you interpret the model coefficients that you get from fitting these models? What conclusions do you draw about whether rates of hospitalization for acute coronary events are affected by exposure to secondary smoke?
4. Now let's try to control for long-term and seasonal trends beyond the linear term for time. Fit a model using a harmonic term like in Bernal et al. (2017). Do the coefficients from the model for the smoking ban and linear time term change? In previous examples, we've been using flexible functions for this such as splines. Repeat the model from above using a spline term. How does this model compare with respect to the smoking ban coefficient? Why?

Applied exercise: Example code

1. **Load the data into R and explore the dataset. Plot the time series of the data on the outcome of interest as it varies over the study period. Use a visual marker in the plot to show the intervention (the date the law was implemented).**

These data are available in the paper's supplement. If you download the supplemental data file on the paper's website, it gives you a zipped file that you can unzip to create a directory of several files. This directory includes "data/sicily.csv". You'll need to move that to your working directory or a subdirectory of your working directory. In the code below, I have it in a subdirectory named "data" of my working directory. The data is in a plain text, comma-separated file (you can tell by the ".csv" file extension), so you can use `read_csv` from the `tidyverse` suite of packages to read it in.

```
# Load some packages that will likely be useful
library(tidyverse)
library(viridis)
library(lubridate)
library(broom)

# Load and clean the data
sicily <- read_csv("data/sicily.csv")
```

Once you read the data in, you should check it out:

```
sicily
```

```
## # A tibble: 59 x 7
##     year month  aces  time smokban     pop  stdpop
##    <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>
##  1  2002     1   728     1       0 364277. 379875.
##  2  2002     2   659     2       0 364277. 376496.
##  3  2002     3   791     3       0 364277. 377041.
##  4  2002     4   734     4       0 364277. 377116.
##  5  2002     5   757     5       0 364277. 377383.
##  6  2002     6   726     6       0 364277. 374113.
##  7  2002     7   760     7       0 364277. 379513.
##  8  2002     8   740     8       0 364277. 376296.
##  9  2002     9   720     9       0 364277. 374653.
## 10  2002    10   814    10       0 364277. 378486.
## # ... with 49 more rows
```

In this case, it looks like the time-step for observations is by month. In other words, we have one measurement per month. The `time` variable has already been added—this gives us the month in the study period. The outcome variable of interest is `aces`, which gives the number of hospital admissions for acute coronary events (ACEs) in each month.

The forcing variable in the study is, since it is an interrupted time series, `time`. We will be comparing risk of ACE hospitalizations before and after the date of the indoor smoking ban, so that date will form our threshold rule for the study. The creators of this dataset already added a variable with data on this threshold rule: the `smokban` variable is `0` before the ban and `1` after.

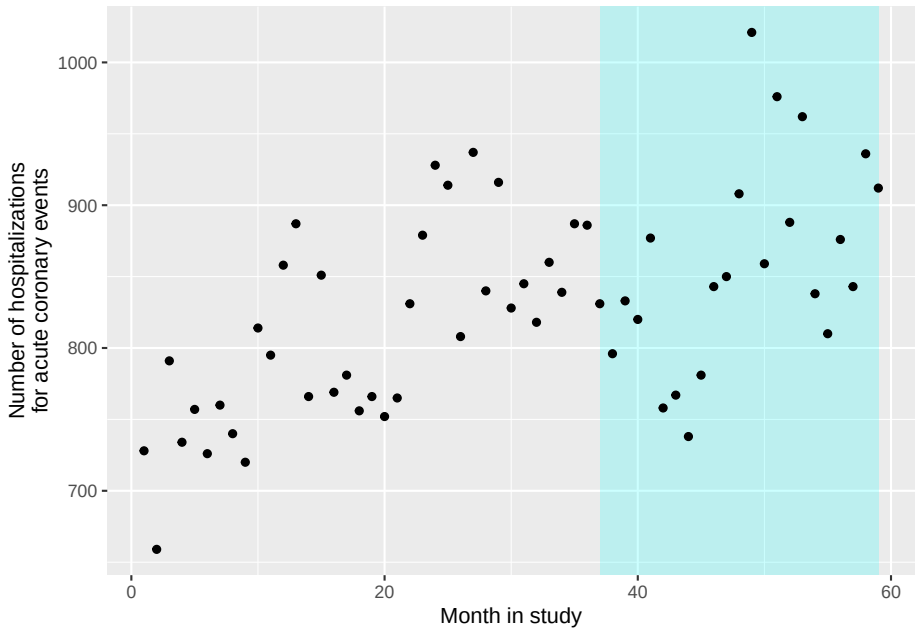
We can take a look using `smokban` to see when the ban was implemented. One way to do this is to group the data by the `smokban` variable and then use `slice` to see the first three rows within each group. Based on this, it looks like the ban started the 37th month of the study:

```
sicily %>%
  group_by(smokban) %>%
  slice(1:3)
```

```
## # A tibble: 6 x 7
## # Groups:   smokban [2]
##    year month  aces  time smokban     pop  stdpop
##   <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1  2002     1   728     1       0 364277. 379875.
## 2  2002     2   659     2       0 364277. 376496.
## 3  2002     3   791     3       0 364277. 377041.
## 4  2005     1   831    37       1 364421. 388153.
## 5  2005     2   796    38       1 364421. 388373.
## 6  2005     3   833    39       1 364421. 386470.
```

Let's use a plot to explore the relationship between the forcing value (time) and the outcome (ACE hospitalizations) before we look at the remaining variables in the data.

```
ggplot() +
  geom_polygon(aes(x = c(37, 59, 59, 37),
                   y = c(Inf, Inf, -Inf, -Inf)),
               fill = "cyan", alpha = 0.2) +
  geom_point(data = sicily, aes(x = time, y = aces)) +
  labs(x = "Month in study",
       y = "Number of hospitalizations\nfor acute coronary events")
```



As a note, to get this to work with `ggplot`, you need to be careful about where you specify the data. We've added the polygon using exact x and y values, rather than pulling them from a column in the dataframe (we knew from checking the data that the ban started for week 37 and then lasted the rest of the study, and

there are 59 weeks in the study). Since we are only using the `sicily` data when we add points, but not when we add the polygon, we need to specify those data only for the the `geom_point` layer, and not within `ggplot`. Also, the order of our layers matters a bit more here than it normally does. If we want the shaded polygon to go behind the points, we should add `geom_polygon` before we add `geom_point`, so the points get plotted on top.

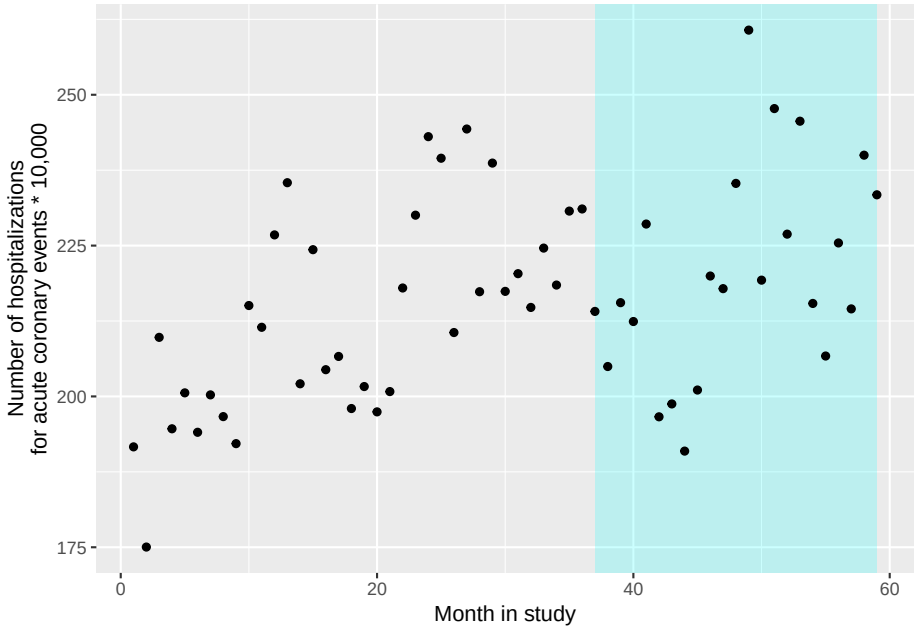
Two other variables are also included in the data, `pop` and `stdpop`. These give measures of the study's population size for every time point in the study, as well as that population as an age-standardized population (in person-years). In our examples in earlier studies, we didn't incorporate the size of the study population in any way. We used counts of deaths, fitting them with a Poisson regression. If the size of the total population only changes slowly (in comparison to the time-scale of the association we were looking at, which was from days up to about a month), and if the disease outcome is rare (that is, unlikely on any study day for any population member), then it is typically fine not to include information about the total population size on each study day (or month, if that is your time step, as in the example for this study).

However, in some cases you will have a population that changes more abruptly. For example, if you may want to count outcomes based on insurance claims. If the number of enrollees for that insurance change substantially on certain dates (e.g., the start of the month or start of the year, which can be points where many new people become eligible for the insurance), you can get abrupt changes in the size of your study population.

If you have the population size for each observation in the time series, then you can address this when you fit your statistical model. You can still fit the data with a Poisson regression model, but you will include something called an *offset*. You will include that in you regression model equation when you fit the regression with R, as we'll look at in the next part of the exercise. The *offset* also allows us to model rates rather than counts as we show later in the chapter.

For right now, we can use the information about population size to change our plot a bit, so that it shows the rate of hospitalizations for coronary events (times 10,000) in each month of the study, rather than the unadjusted count:

```
ggplot() +
  geom_polygon(aes(x = c(37, 59, 59, 37),
                   y = c(Inf, Inf, -Inf, -Inf)),
               fill = "cyan", alpha = 0.2) +
  geom_point(data = sicily, aes(x = time, y = aces / stdpop * (10 ^ 5))) +
  labs(x = "Month in study",
       y = "Number of hospitalizations\nfor acute coronary events * 10,000")
```



2. **Next, let's look at how the rates of the outcome changes right before and right after the ban. At this stage, we won't worry about any underlying time trends in the outcome rates. Instead, compare the outcome rates in two groups—the data points collected for the time points right before the ban, and then the data collected for time points right after the ban. Try a few windows for capturing "right before" and "right after" (e.g., 3 months, 6 months, 12 months). How does the precision of the estimate change as you change that window? How do you think the potential might change for residual confounding from long-term trends in the outcome rates irrespective of the ban?**

Let's start with the smallest window, 3 months. This means that we will pick out only the three data points before the ban (months 34, 35, and 36 of the study) and the three after (months 37, 38, and 39):

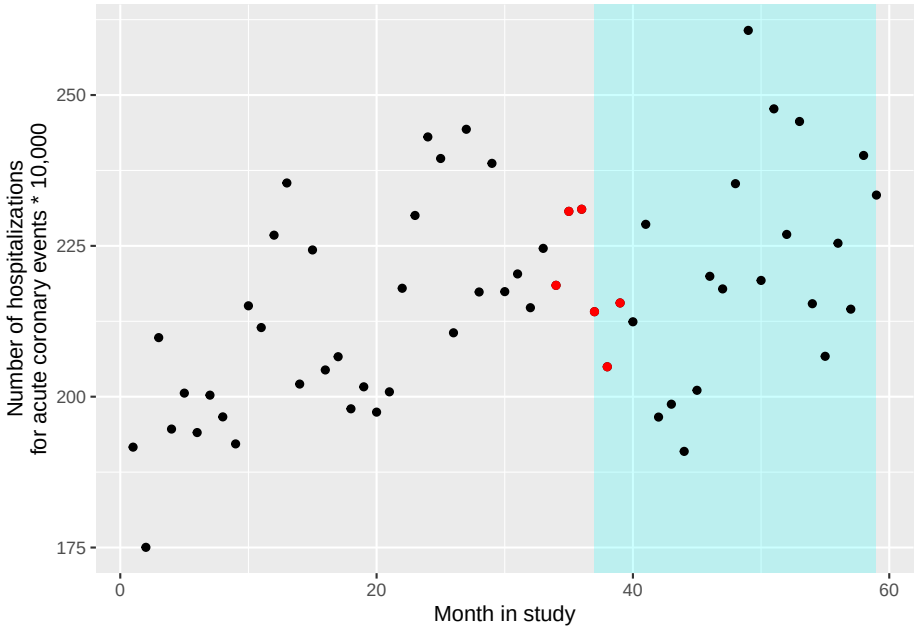
```
near_threshold <- sicily %>%
  filter(time %in% 34:39)
near_threshold
```

```
## # A tibble: 6 x 7
##    year month  aces  time smokban     pop  stdpop
##   <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>   <dbl>
## 1  2004    10   839    34       0 364700. 384052.
## 2  2004    11   887    35       0 364700. 384450.
## 3  2004    12   886    36       0 364700. 383428.
## 4  2005     1   831    37       1 364421. 388153.
```

```
## 5  2005     2   796    38       1 364421. 388373.
## 6  2005     3   833    39       1 364421. 386470.
```

To check that we've pulled the right months, we can add these in a different color to our plot from the last section:

```
ggplot() +
  geom_polygon(aes(x = c(37, 59, 59, 37),
                   y = c(Inf, Inf, -Inf, -Inf)),
               fill = "cyan", alpha = 0.2) +
  geom_point(data = sicily, aes(x = time, y = aces / stdpop * (10 ^ 5))) +
  geom_point(data = near_threshold,
             aes(x = time, y = aces / stdpop * (10 ^ 5)),
             color = "red") +
  labs(x = "Month in study",
       y = "Number of hospitalizations\nfor acute coronary events * 10,000")
```



Next, let's compare the rates of the outcome in these two groups: "untreated" (i.e., right before the date of the ban) and "treated" (i.e., right after the date of the ban). We can use a GLM to do this. It might seem somewhat like overkill—we could typically use a simpler statistical framework to determine the difference in two groups. However, using the GLM framework gets the job done even for a simple comparison, with the added benefit that it gives us room to expand as we think about adding control for some other factors. Further, it allows us to do some nice things like add an offset for the population size and allow for overdispersion.

To allow for overdispersion, we'll do the same thing we did in earlier chapters—we'll use the "quasipoisson" family when setting up the regression equation. This will have R estimate an additional parameter, one that estimates an overdispersion parameter for the data. For the population offset, we can incorporate that using the `offset` function. We'll take the log of the standardized population, so it will have the same transformation as the outcome variable in this structure (remember that, for Poisson GLMs, the link is a log). By wrapping it in `offset` when we include it in the model equation, we're telling R something special—it should include this variable in the structure of the equation when it fits the data, but it should not estimate a coefficient for it (as it will for everything else on the right side of the `~` in the nodel statement).

Here is the code you can use to fit this regression model:

```
near_thres_mod <- glm(aces ~ offset(log(stdpop)) + smokban,
                      data = near_threshold,
                      family = "quasipoisson")
```

The model equation for the model you are fitting with this code is:

$$log(E(Y_t)) = \beta_0 + \beta_1 X_t + log(StdPop_t)$$

(Note that there is no paratemer for the offset term!) We can rearrange this as:

$$log(E(Y_t)) - log(StdPop_t) = \beta_0 + \beta_1 X_t$$

And based on the properties of logarithmic functions the above is equivalent to this which is essenntially a model for rates

$$log(E(Y_t/StdPop_t)) = \beta_0 + \beta_1 X_t$$

$E(Y/Std.Pop.)$ is the expected rate of outcome Y on day t (the $StdPop_t$ in the denominator makes this a rate based on the standardize population size that day rather than just a count), β_0 is a model intercept, X_t is an indicator variable that is 1 after the ban and 0 before the ban, and β_1 is the estimate of the log rate ratio after the ban versus before the ban.

You can use the same technique as always to extract the estimates of the coefficients (again, notice that you don't get one for the offset term!):

```
library(broom)
near_thres_mod %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  -6.09      0.0168   -362.   3.49e-10
```

```
## 2 smokban      -0.0695    0.0241     -2.88 4.51e- 2
```

Based on these, there is a decrease in rates of ACE after the smoking ban, and we would reject the null hypothesis that the size of the change in rates post- versus pre-ban is 0 (because the p-value is lower than 0.05, although barely).

From these estimates, we can also get an estimate of the relative rate for the group right after the ban versus right before the ban, as well as the 95% confidence intervals for that estimate. Since we only have a few data points (6), we shouldn't use 1.96 in calculating the confidence intervals, but instead get the right value for our sample size using `qt`, which allows us to get quantiles of interest from the (student's) t-distribution. (The 1.96, is taken from the standard normal distribution (z-scores), but because of out small sample size n=6 the t-distribution will deviate considerably for that)) Since we have 6 data points and are estimating one coefficient plus the intercept, we have 6 - 2 degrees of freedom for this comparison:

```
near_thres_mod %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 4) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 4) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.933  0.872   0.998
```

(We can check the quantilies of the t-distriution for 4 df, if we want to see how far away from the 1.96 value we are)

```
qt(0.025, df = 4)
```

```
## [1] -2.776445
```

```
qt(0.975, df = 4)
```

```
## [1] 2.776445
```

We can see that ± 1.96 would have given us deceptively narrower CIs

Now we can try some larger windows. Here is the analysis for 6 month-windows on either side of the ban. The only things that change are the window for selecting data to include and the degrees of freedom used when calculating confidence intervals:

```
near_threshold2 <- sicily %>%
  filter(time %in% 31:42)
near_thres_mod2 <- glm(aces ~ offset(log(stdpop)) + smokban,
```

```
                      data = near_threshold2,
                      family = "quasipoisson")
near_thres_mod2 %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  -6.10      0.0167   -366.   5.74e-22
## 2 smokban      -0.0520    0.0239     -2.18 5.43e- 2
```

```
near_thres_mod2 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 10) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 10) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.949  0.900    1.00
```

Here is the analysis for 12 month-windows on either side of the ban. Again, the only things that change are the window for selecting data to include and the degrees of freedom used when calculating confidence intervals:

```
near_threshold3 <- sicily %>%
  filter(time %in% 25:48)
near_thres_mod3 <- glm(aces ~ offset(log(stdpop)) + smokban,
                      data = near_threshold3,
                      family = "quasipoisson")
near_thres_mod3 %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)  -6.09      0.0160   -382.   1.56e-43
## 2 smokban      -0.0656    0.0229     -2.86 9.04e- 3
```

```
near_thres_mod3 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 22) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 22) * std.error)) %>%
```

```
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.937  0.893   0.982
```

As we look at these different windows, we don't see much change in the estimated relative rate of the outcome after the ban versus before. However, our estimate of this relative rate is becoming more precise as it's estimated with more data points.

However, eventually we may run into problems with residual confounding. As we include data further and further from the threshold (the date of the ban), we have more and more problems with the assumption that the data before and after the threshold are comprable for any factors other than the status of the smoking ban. In particular, we're opening up more and more possibility for our estimate to be affected by residual confounding by long-term trends in the outcome rate over the study period as we include a wider window of days in that study period. In the next section, we'll look at how we can add control for a long-term trend to the model, so we can use all of the data while controlling for potential long-term trends, thus reducing the risk of residual confounding by them.

3. **Now see if you can fit a regression model that uses all the data, while using a regression term to allow for a long-term trend in the rates of the outcome. You may want to try out several of the model structures shown in Figure 2 of Bernal et al. (2017). Can you interpret the model coefficients that you get from fitting these models? What conclusions do you draw about whether rates of hospitalization for acute coronary events are affected by exposure to secondary smoke?**

Now we will fit a full regression model for an interrupted time series. We will add a term to fit a trend for steady trends along the forcing variable. For interrupted time series, that forcing variable is time, so we'll be adding a term to control for trends long-term in time. However, this general idea can be used with any type of regression discontinuity study design, some of which will have forcing variables other than time.

To add a trend for time—as long as we're willing to assume that the trend isn't changed when the ban takes effect, just that there is a "hop" in the trend line to account for an immediate change in rates—we can just add a term to our regression model. We'll now fit the following regression model:

$$log(E(Y_t/StdPop_t)) = \beta_0 + \beta_1 X_t + \beta_2 T$$

where all the terms are the same as in the last section, while now a linear term has been added for month in study (time; T) with an associated coefficient β_2. In Bernal et al. (2017), this corresponds to the model structure shown in panel (a) of Figure 2: There is an "interruption" at the time of the ban, and there is a long-term trend in the outcome rates regardless of this ban, but the ban only creates the immediate interruption, without changing the shape of the long-term trend.

Below is the code that will fit this model in R. We'll change the `data` argument to now include all available data points. The other change is that we're adding a linear term for time (`time`).

```
int_ts_mod1 <- glm(aces ~ offset(log(stdpop)) + smokban + time,
                   data = sicily,
                   family = "quasipoisson")
```

We can look at the results from fitting this model, using the same technique as in the last section:

```
int_ts_mod1 %>%
  tidy()
```

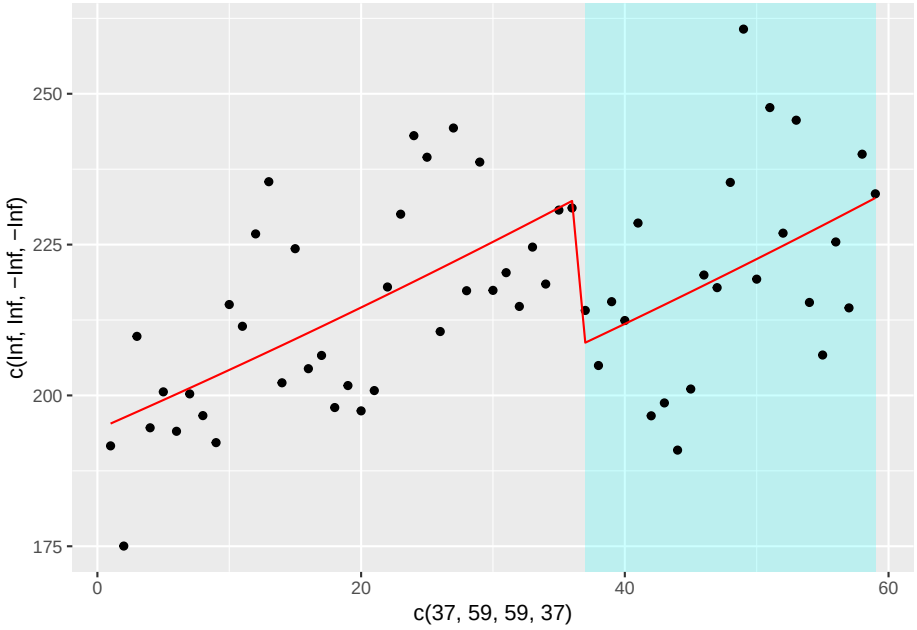
```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept) -6.24     0.0211     -296.   3.94e-91
## 2 smokban     -0.112    0.0325       -3.44 1.12e- 3
## 3 time         0.00494  0.000942      5.25 2.43e- 6
```

```
int_ts_mod1 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 56) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 56) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.894  0.838   0.955
```

These results show us a few things. First, there is clearly a long-term time trend—the p value on the "time" coefficient estimate is much lower than 0.05. Second, the estimated benefit from the smoking ban, in terms of reducing rates of ACE hospitalization, is a bit stronger when we control for long-term trends. We are now estimating about a 11% decrease in hospitalization rates in association with this indoor smoking ban. It's possible that our results in the last sections were somewhat confounded by long-term trends.

```
mod_data <- int_ts_mod1 %>%
  augment() %>%
  mutate(stdpop = sicily$stdpop,
         aces_adj = aces / stdpop * (10 ^ 5))
ggplot() +
  geom_polygon(aes(x = c(37, 59, 59, 37),
                   y = c(Inf, Inf, -Inf, -Inf)),
               fill = "cyan", alpha = 0.2) +
  geom_point(data = mod_data, aes(x = time,
                                  y = aces / stdpop * (10 ^ 5))) +
  geom_line(data = mod_data, aes(x = time,
                                 y = exp(.fitted) / stdpop * (10 ^ 5)),
            color = "red")
```



Next, we can try to fit the model stucture that is shown in panel (c) of Figure 2 in Bernal et al. (2017). This structure includes an abrupt interruption when the ban is introduced, as well as a change in the slope of the long-term trend following the ban.

The structure of this regression model will include an extra term, to allow the slope of the long-term trend to change after the ban. The model is:

$$log(E(Y_t/StdPop_t)) = \beta_0 + \beta_1 X_t + \beta_2 T + \beta_3 (T - T_0) X_t$$

where $(T - T_0)$ is a measure of time since the date of the ban and β_3 allows for a

change in the slope of the time trend line after the ban. (As a note, $(T - T_0)X_t$ could also be expressed as $(T - T_0)_+$ using the $(..)_+$ notation we introduced last week.)

To fit this model in R, you can run:

```
int_ts_mod2 <- glm(aces ~ offset(log(stdpop)) + smokban + time +
                     I(time - 36):smokban,
                    data = sicily,
                    family = "quasipoisson")
```

```
int_ts_mod2 %>%
  tidy()
```

```
## # A tibble: 4 x 5
##   term                   estimate std.error statistic  p.value
##   <chr>                     <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)          -6.24       0.0233   -268.    2.31e-87
## 2 smokban              -0.114      0.0356     -3.21  2.23e- 3
## 3 time                  0.00485    0.00107     4.52  3.28e- 5
## 4 smokban:I(time - 36)  0.000417   0.00231     0.181 8.57e- 1
```

```
int_ts_mod1 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 55) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 55) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.894  0.838   0.955
```

Based on these results, there is not a statistically significant change in the slope of the time trend after the ban compared to before (i.e., a p-value of 0.18 for testing against the null that this coefficient is 0). We get a very similar estimate of the abrupt change in ACE rates right when the ban was implemented as compared to the simpler model we fit without a term for change in slope of the long-term time trends.

4. **Now let's try to control for long-term and seasonal trends beyond the linear term for time. Fit a model using a harmonic term like in Bernal et al. (2017). Do the coefficients from the model for the smoking ban and linear time term change? In previous examples, we've been using flexible functions for this such as splines. Repeat the model from above using a spline term. How does this model compare with respect to the smoking ban**

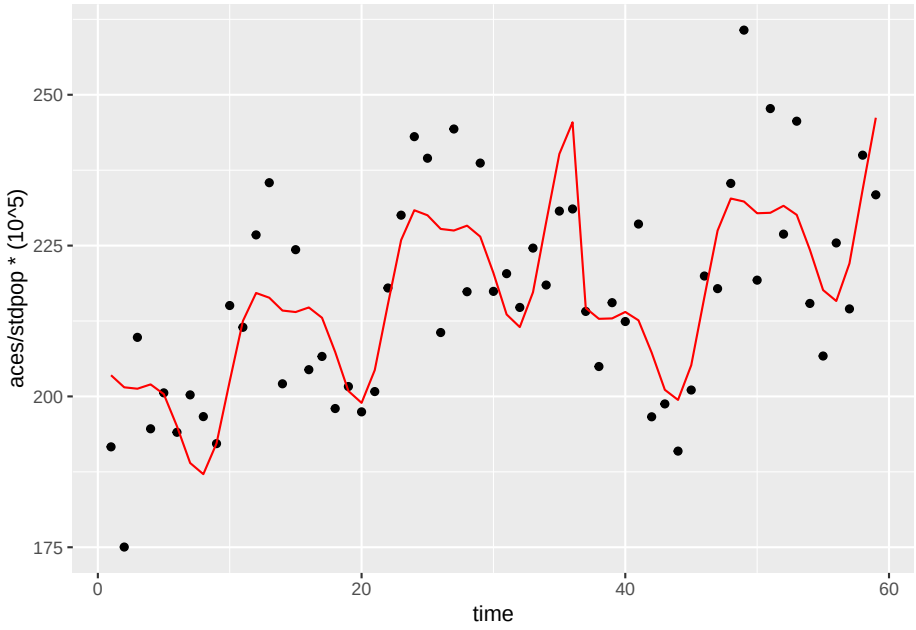
coefficient? Why?

We've been adjusting for long-term and seasonal trends using a flexible spline term so far. Let's try and incorporate this in the mode here. Here we will try a new function from the `tsModel` package called `harmonic`. The `harmonic` function will fit pairs of sine and cosine functions in a given period of time. Here we will define the function as `harmonic(month,2,12)` fitting two pairs of sine and cosine functions over a 12-month period. These combined sine and cosine functions create a sinusoidal patten with peaks and troughs that we've seen for seasonal treands previously.

```
library(tsModel)
int_ts_mod3 <- glm(aces ~ offset(log(stdpop)) + smokban + time +
                     I(time - 36):smokban + harmonic(month,2,12),
                      data = sicily,
                      family = "quasipoisson")
```

In order to visualize the effect of the harmonic function we can plot the predicted values from this model same as above:

```
mod_data2 <- int_ts_mod3 %>%
  augment() %>%
  mutate(stdpop = sicily$stdpop,
         aces_adj = aces / stdpop * (10 ^ 5))
ggplot() +
  geom_point(data = mod_data2, aes(x = time,
                                   y = aces / stdpop * (10 ^ 5))) +
  geom_line(data = mod_data2, aes(x = time,
                                  y = exp(.fitted) / stdpop * (10 ^ 5)),
            color = "red")
```



We see that the addition of the harmonic term now yields a more flexible overall function over time. Now let's check how the coefficients for `smokban` and `time` changed if at all

```
int_ts_mod3 %>%
  tidy()
```

```
## # A tibble: 8 x 5
##   term                    estimate std.error statistic  p.value
##   <chr>                      <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)              -6.25     0.0189    -331.    1.20e-86
## 2 smokban                  -0.132    0.0298      -4.43  5.07e- 5
## 3 time                      0.00510  0.000875     5.83  3.72e- 7
## 4 harmonic(month, 2, 12)1  0.0383   0.0102       3.77  4.27e- 4
## 5 harmonic(month, 2, 12)2 -0.0176   0.00970     -1.82  7.49e- 2
## 6 harmonic(month, 2, 12)3  0.0384   0.00971      3.95  2.42e- 4
## 7 harmonic(month, 2, 12)4  0.0150   0.00975      1.53  1.31e- 1
## 8 smokban:I(time - 36)     0.00148  0.00185      0.799 4.28e- 1
```

```
int_ts_mod3 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 51) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 51) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

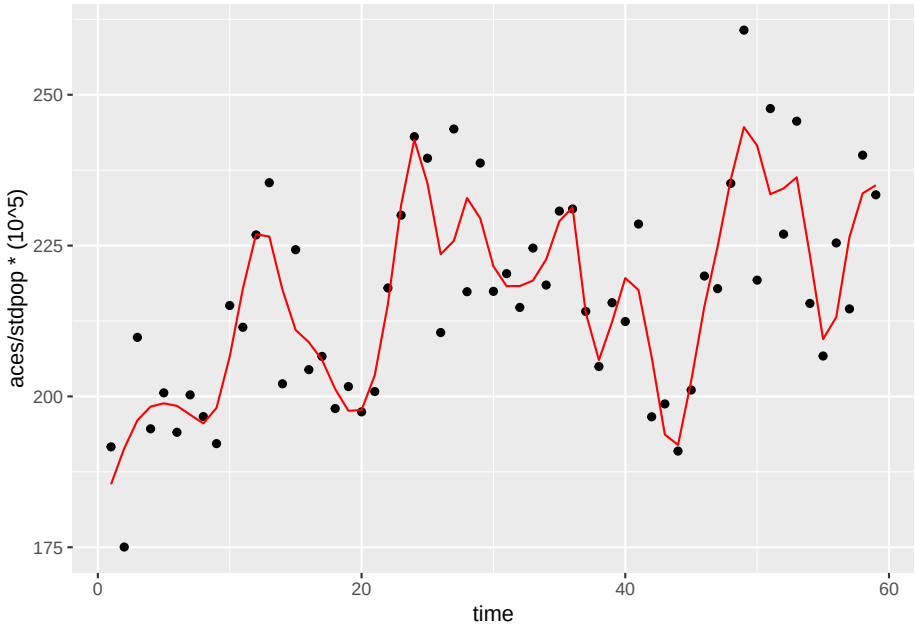
```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1 0.876  0.826   0.930
```

The change in the slope of the time trend after the ban is still not statistically significant, but we see that the RR for the smoking ban term is actually marginally lower, meaning a slighlty more protetive effect after adjusting for the seasonal trends with a harmonic term.

Now why did we use this type of function as opposed to a spline? Let's repeat the model from above with a natural spline term as we had been doing in the previous example, with 30 df's (~6 per year).

```
library(splines)
int_ts_mod4 <- glm(aces ~ offset(log(stdpop)) + smokban + time +
                     I(time - 36):smokban + ns(time, df=30),
                      data = sicily,
                      family = "quasipoisson")
```

```
mod_data3 <- int_ts_mod4 %>%
  augment() %>%
  mutate(stdpop = sicily$stdpop,
         aces_adj = aces / stdpop * (10 ^ 5))
ggplot() +
  geom_point(data = mod_data3, aes(x = time,
                                   y = aces / stdpop * (10 ^ 5))) +
  geom_line(data = mod_data3, aes(x = time,
                                  y = exp(.fitted) / stdpop * (10 ^ 5)),
            color = "red")
```



We see that the spline term has succesfully fitted a flexible function over time. Now let's check the smoking ban RR

```
int_ts_mod4 %>%
  tidy() %>%
  filter(term == "smokban") %>%
  mutate(rr = exp(estimate),
         low_rr = exp(estimate + qt(0.025, df = 25) * std.error),
         high_rr = exp(estimate + qt(0.975, df = 25) * std.error)) %>%
  select(rr, low_rr, high_rr)
```

```
## # A tibble: 1 x 3
##      rr low_rr high_rr
##   <dbl>  <dbl>   <dbl>
## 1  1.00  0.632    1.59
```

The RR for the smoking ban term is now null. A richly parameterized spline term with a lot of degrees of freedom is actually too felxible a function for this analysis. Because of all the flexibility we are affording it, it is absorbing the drop in ACE rates occurring immediately after the ban in the overall function for time. Although spline functions and their flexibility can be an invaluable tool in modeling complex exposure-respone relationships, in this case it actually masked the effect of interest.

Chapter 6

Estimating health impacts

6.1 Readings

The readings for this chapter are:

- Balbus et al. (2016) Overview on climate change and human health, including a section on quantifying health impacts. The first chapter in a report on climate change and human health by the US Global Change Research Program.
- Steenland and Armstrong (2006) (Just the section on attributable fraction) Overview on estimating burden of disease through measures like attributable risk and attributable number (Note: The equation for AF_{pop} in this paper in equation 3 aligns with the equations for attributable risk described in this chapter)
- Gasparrini and Leone (2014) Calculating attributable risk from distributed lag non-linear models. Example code is included for this paper.
- Vicedo-Cabrera et al. (2019) Provides a tutorial of all the steps for a projecting of health impacts of temperature extremes under climate change.

The following are supplemental readings (i.e., not required, but may be of interest) associated with the material in this chapter:

- Kinney et al. (2008) A review of challenges and approaches for projecting heat-related deaths under climate change
- Benichou (2006) Encyclopedia entry on attributable risk
- Northridge (1995) Brief summary of attributable risk with examples from epidemiology
- Greenland and Robins (1988) Definition and interpretation of attributable fractions

6.2 Attributable risk and attributable number

So far, we have focused on estimating the relative risk of exposure on a health outcome. In some cases—where we can assume a linear relationship between the exposure and the log counts of the outcome—we were able to get a single measure of the relative risk per unit change of the exposure. In this case, the single measure of relative risk applies to a one-unit change anywhere along the range of exposure observed for the population. One common example is the association between outdoor particulate matter concentration and mortality, which is typically modeled using a linear association, resulting in a single estimate of relative risk per unit increase in the exposure (as a note, this "unit" can be any constant change in the exposure—often it will be selected to reflect a realistically important change in the exposure, for example, an increase of $10\mu g/m^3$ for $PM_{2.5}$).

We have also looked at some more complex cases, where we've used a non-linear function of the exposure to model associated risk of the health outcome. In these cases, it becomes a bit more complicated to interpret the results. First, you need to identify a "baseline" of exposure to use as a comparison for measurements of relative risk. While for a linear association, you can measure a relative risk per unit change, regardless of where the change happens along the range of exposure, for a non-linear association, you must pick specific points to compare, and a unit change in exposure will vary across the range of exposure. For example, when looking at temperature and mortality in London, we compared risk at specific high and low temperatures to a baseline at a mild, "safe" temperature, one where mortality risk was observed to be at its lowest. In other words, we selected a baseline level of exposure to use as a reference and then compared relative risk at "slices" of the non-linear exposure-response function to get summaries of the observed relative risk. As we moved to cross-basis functions, we similarly took "slices" to look at risk at specific lags or specific temperatures.

These techniques point towards a general theme—when we are studying how an environmental exposure is associated with risk of a health outcome, we need to not only fit an appropriate statistical model to describe the association, but we also need to extract some summaries from that model that we can use to communicate what we've found. Not only is this important for communicating our findings, it's also critical for making sure our findings can be used to improve public health.

So far, we have focused on summarizing and communicating our results based on estimates of relative risk. These estimates are a common summary in published environmental epidemiology studies, and they're helpful in identifying how levels of an exposure translate to increased health risk. However, they fail to incorporate one facet that is very important when trying to translate findings to public health impacts—they do not incorporate how common or rare exposures of different levels are in real life. For example, say you fit a model that shows an exposure-response function for temperature and mortality that

is very flat everywhere except at very high temperatures. If those very high temperatures occur often (say, 20 days a year on average), then temperature would cause many more excess deaths in the community compared to if those temperatures are very rare (say, 1 day every five years on average). In other words, to interpret the public health impact of an exposure, you need to consider not only how health risk changes with the level of the exposure, but also how frequently different levels of exposure occur.

One epidemiological measure that incorporates these two facets is the measure of **attributable risk**. Attributable risk aims to measure the proportion of cases of disease in a population that is attributable to an exposure—in other words, what proportion of cases would you have avoided in a population if the exposure were eliminated (or, in cases where that doesn't make sense like temperature, held at its safest level)? A number of other terms are sometimes also used to describe this idea, including *population attributable risk*, *attributable fraction*. *Etiological* or *etiologic fraction* also appears in the literature as an equivalent or interchangeable term to *attributable fraction*, but as Greenland and Robins point out (Greenland and Robins, 1988) the interpretation of each is more nuanced, and the two are not necessarily the same quantity.

In the simple case of a binary exposure (in other words, everyone in the study population can either be exposed or unexposed, but there aren't different levels of exposure), the attributable risk (AR) can be expressed mathematically as:

$$AR = \frac{Pr(D) - Pr(D|\bar{E})}{Pr(D)}$$

where $Pr(D)$ is the probability of the disease in the population and $Pr(D|\bar{E})$ is the probability of the disease if the whole population were unexposed. This can be expressed in terms of the probability of exposure and the relative risk of exposure on the outcome (Benichou, 2006; Northridge, 1995):

$$AR = \frac{Pr(E) * (RR - 1)}{[Pr(E) * (RR - 1)] + 1}$$

where $Pr(E)$ is the probability of exposure in the population (again, this is for a binary exposure) and RR is the relative risk of the outcome associated with exposure.

Another similar measure is also helpful in estimating the impact of the exposure—the **attributable number**. The attributable number gives an estimate of the absolute number of cases that would have been avoided had the entire population been unexposed. It essentially takes the attributable risk and multiplies it by the prevalence of the outcome in the population, to move from a proportion of cases that are attributable to the exposure to the number of cases that are attributable (AN) (Benichou, 2006; Northridge, 1995):

$$AN = AR * n$$

where n is the total number of cases in the population.

This measure is often helpful in communicating total impacts to people who are not epidemiologists. For example, this measurement can be used to determine the number of excess deaths that were likely caused by a disaster, helping to conceptualize it's impact on a population. These estimates can also be extended to help calculate economic costs related to health impacts. For example, an estimate of excess hospitalizations related to an exposure can be multiplied by the average cost of hospitalization in that population for the outcome to generate a back-of-the-envelope estimate of the costs to an insurer related to that exposure.

It becomes a bit more complex to estimate the attributable risk when the exposure is continuous rather than binary, but the process is conceptually similar. First, you need to determine a baseline level of exposure. This will serve in creating a counterfactual—in the attributable risk, you will be comparing to an alternative scenario where the exposure is always at this level. There are a few ways you can pick this baseline. If the level of the exposure can really (and reasonably) take a value of 0, then you can use that for the comparison. For example, if you are studying smoking and health, it would be reasonable to set non-smoking as the baseline. For other exposures, there may always be some background level of the exposure—a level that couldn't be eliminated regardless of policy choices or other actions—and in that case it may make sense to set the baseline to this background level of exposure. Finally, there are some cases where the "safest" level comes within the range of exposure, rather than at a minimum or maximum value. Temperature is an example of this, where the lowest risk of many health outcomes occurs at a mild temperature in the middle of the temperature range. In this case, the baseline is often set at this "safest" level of exposure. For temperature and mortality, this point is called the *minimum mortality temperature*. Any of these choices can be fine, but since there are choices, it's important that you clearly describe your baseline when estimating attributable risk, and justify why it's a helpful baseline in interpreting health impacts of the exposure.

Second, you need to compare each observed exposure in the population to this baseline level. For each observation, you'll estimate the relative risk at that observed exposure compared to the baseline. You can then get an observation-level estimate of attributable fraction for each exposure of an exposure, which can be used to estimate a component of attributable number for each exposure, and sum these contributions to get an estimate for the population as a whole, incorporating the range of exposures among that population. Mathematically, the steps for this are to look at each observation in the population (i.e., each "exposure", which in a daily time series would be each date) and calculate the attributable fraction (AF_{x_i}) for that "exposure" (x_i) (adapted from Gasparrini

and Leone (2014)). You can calculate the attributable fraction for each observed exposure as:

$$AF_{x_i} = \frac{(RR_{x_i} - 1)}{RR_{x_i}}$$

We often estimate RR_{x_i} using regression, as the exponent of an estimated regression coefficient β_{x_i}. We can rearrange the equation to express it based on β_{x_i}, instead:

$$AF_{x_i} = \frac{exp(\beta_{x_i}) - 1}{exp(\beta_{x_i})} = \frac{exp(\beta_{x_i})}{exp(\beta_{x_i})} - \frac{1}{exp(\beta_{x_i})} = 1 - \frac{1}{exp(\beta_{x_i})} = 1 - exp(-\beta_{x_i})$$

If you have categories of exposure, you may have calculated β_{x_i} for each of those categories compared to a baseline. If you have fit a non-linear association, you can get the β_{x_i} for each of the many exposure levels (e.g., each temperature) by "slicing" the exposure-response function at that level in comparison to a baseline. In practice, we can calculate this for each day in a time series by "slicing" to get the estimate of β_{x_i} for that day's temperature.

By multiplying this by the total number of cases observed at exposure level (n_{x_i}), you can estimate the attributable number for that specific exposure (AN_{x_i}) (adapted from Gasparrini and Leone (2014)):

$$AN_{x_i} = n_{x_i} * AF_{x_i}$$

In practice, for a time series study, you could do this by taking your estimate of AF_{x_i} for each day in the study and then multiplying it by the number of cases (e.g., deaths) observed on that study day.

You can then sum all the observation-level attributable number estimates to get the total attributable number in the population (for a time series, this will be over the study period) (adapted from Gasparrini and Leone (2014)):

$$AN = \sum_{i=1}^{n} AN_{x_i}$$

From this attributable number, you can estimate an attributable fraction by dividing by the total number of cases in the population (adapted from Gasparrini and Leone (2014)):

$$AR = \frac{AN}{\sum_{i=1}^{n} n_i}$$

where n_i is the number of cases observed at exposure i.

Applied: Calculating attributable risk and attributable number from time series data

For this exercise, you will be using data and a model (`dlnm_mod_1`) that you fit in Chapter 4. To fit that model, you also created a crossbasis called `temp_basis`, and you'll need that object as well. The example code for this exercise copies the code to read in the data and create that model and crossbasis.

1. Start with the model you fit as `dlnm_mod_1` in Chapter 4, as well as the `obs` dataset you used to fit it. Using 7 degrees C as the baseline exposure, determine the number of deaths and the fraction of deaths attributable to heat in the first week of July in 2012 in London.
2. Extend this idea to calculate the attributable risk and attributable number of deaths related to temperature throughout the study period. Calculate these values separately for heat and cold.
3. Do the same calculation as in the previous part, but change the underlying model that you're using. In this case, use the `dist_lag_mod_5` you fit in Chapter 4. What differences are there in attributable risk and attributable number when you use a model that incorporates lagged effects (`dist_lag_mod_5`) compared to one that only fits the immediate association (`dlnm_mod_1`)?

Applied exercise: Example code

Here is the code to read in the data and create that model and crossbasis you will need in this exercise:

```
# Load some packages that will likely be useful
library(tidyverse)
library(viridis)
library(lubridate)
library(broom)

# Load and clean the data
obs <- read_csv("data/lndn_obs.csv") %>%
  mutate(dow = wday(date, label = TRUE)) %>%
  mutate(time = as.numeric(date) - first(as.numeric(date)))

library(dlnm)
library(splines)
temp_basis <- crossbasis(obs$tmean, lag = 0,
                         argvar = list(fun = "ns", df = 4),
                         arglag = list(fun = "integer"))
dlnm_mod_1 <- glm(all ~ temp_basis + factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

1. **Start with the model you fit as `dlnm_mod_1` in Chapter 4, as well as the `obs` dataset you used to fit it. Using 7 degrees C as the baseline exposure, determine the fraction of deaths attributable to heat in the first week of July in 2012 in London.**

Let's start by pulling out the observations for the first week in July of 2012 (note that you can use `interval` and `%within%` from lubridate to help pull observations with a date within a certain range):

```
library(lubridate)
july_week <- obs %>%
  filter(date %within% interval(ymd("2012-07-01"), ymd("2012-07-07"))) %>%
  select(date, all, tmean)
july_week
```

```
## # A tibble: 7 x 3
##   date         all tmean
##   <date>     <dbl> <dbl>
## 1 2012-07-01   103  14.6
## 2 2012-07-02   124  15.2
## 3 2012-07-03   106  17.3
## 4 2012-07-04   110  19.2
## 5 2012-07-05   106  18.5
## 6 2012-07-06   102  17.0
## 7 2012-07-07   109  16.0
```

In this data, we've pulled out the variables we'll need in calculating attributable risk and attributable fraction—we have the daily value of the exposure, mean temperature (`tmean`), and the daily number of cases, the count of daily deaths (`all`).

First, let's calculate the attributable fraction. To do this, we need to use the model that we fit earlier to determine what the log relative risk is for each day's temperature. You can use the `crosspred` function to determine the log relative risk at any temperature compared to a baseline temperature that you specify. In the `crosspred` call, you'll use the `cen` parameter to say which temperature you want to use as your baseline and the `at` parameter to say which temperatures you want to predict to—in this case, we want to predict to the temperatures for the first week in July of 2012 (`july_week$tmean`).

The `crosspred` object includes a lot of output; the `matfit` part gives the central estimates of log relative risk at each temperature:

```
crosspred(basis = temp_basis, model = dlnm_mod_1,
          cen = 7, at = july_week$tmean)$matfit
```

```
##                       lag0
## 14.6077699661255 0.01199433
## 15.190845489502  0.01275503
```

```
## 16.0049057006836 0.01599277
## 17.0358219146729 0.02461931
## 17.2731075286865 0.02731069
## 18.4983215332031 0.04512788
## 19.1906547546387 0.05791965
```

With some code, you can extract this and join it with the `july_2012` dataframe:

```
july_week <- crosspred(basis = temp_basis, model = dlnm_mod_1,
          cen = 7, at = july_week$tmean) %>%
  pluck("matfit") %>%
  as.data.frame() %>%
  rownames_to_column(var = "tmean") %>%
  rename("beta_x" = lag0) %>%
  mutate(tmean = as.numeric(tmean)) %>%
  inner_join(july_week, ., by = "tmean")
july_week
```

```
## # A tibble: 7 x 4
##   date         all tmean beta_x
##   <date>     <dbl> <dbl>  <dbl>
## 1 2012-07-01   103  14.6 0.0120
## 2 2012-07-02   124  15.2 0.0128
## 3 2012-07-03   106  17.3 0.0273
## 4 2012-07-04   110  19.2 0.0579
## 5 2012-07-05   106  18.5 0.0451
## 6 2012-07-06   102  17.0 0.0246
## 7 2012-07-07   109  16.0 0.0160
```

From this, you can calculate attributable risk on each day as $AR = 1 - exp(-\beta_x)$ and then calculate attributable number as $AN = AR * n$:

```
july_week <- july_week %>%
  mutate(attr_frac = 1 - exp(-beta_x),
         attr_num = all * attr_frac)
july_week
```

```
## # A tibble: 7 x 6
##   date         all tmean beta_x attr_frac attr_num
##   <date>     <dbl> <dbl>  <dbl>     <dbl>    <dbl>
## 1 2012-07-01   103  14.6 0.0120    0.0119     1.23
## 2 2012-07-02   124  15.2 0.0128    0.0127     1.57
## 3 2012-07-03   106  17.3 0.0273    0.0269     2.86
## 4 2012-07-04   110  19.2 0.0579    0.0563     6.19
## 5 2012-07-05   106  18.5 0.0451    0.0441     4.68
## 6 2012-07-06   102  17.0 0.0246    0.0243     2.48
## 7 2012-07-07   109  16.0 0.0160    0.0159     1.73
```

Over this week, the fraction of deaths attributable to heat on any day (compared to a baseline of 7 degrees C) ranged from about 1.2% of all deaths to about 5.6% of all deaths. Across these days, between 1.2 and 6.2 deaths on a given day were attributable to heat.

We can get the total number of deaths attributable to heat, as well as the total attributable fraction over the time period, by summing up across the week for the attributable number and then dividing this by the total number of cases in the week to get the attributable fraction:

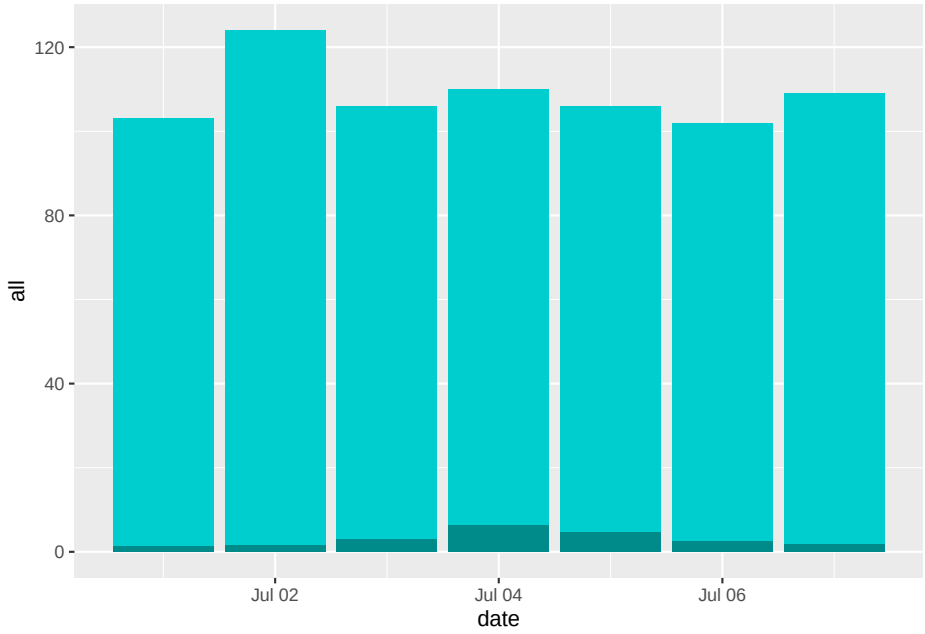
```
july_week %>%
  summarize(tot_attr_num = sum(attr_num),
            tot_attr_frac = tot_attr_num / sum(all))
```

```
## # A tibble: 1 x 2
##   tot_attr_num tot_attr_frac
##          <dbl>         <dbl>
## 1         20.7        0.0273
```

This tells us that there were about 21 deaths attributable to heat in London that week (i.e., deaths that would not have occurred if the temperature was instead 7 degrees C each day in the week), and that this makes up about 3% of all the deaths in London that week.

You can plot day-by-day estimates of the total deaths, highlighting the number that are attributable to temperature, during this period:

```
july_week %>%
  ggplot(aes(x = date)) +
  geom_col(aes(y = all), fill = "cyan3") +
  geom_col(aes(y = attr_num), fill = "cyan4")
```



With this example, you can see how you can calculate these estimates from scratch. However, this will become cumbersome as you move to larger and more complex examples. Fortunately, the authors of Gasparrini and Leone (2014) included an R script with code for a function to do this, both for these simpler examples and as you move to more complex examples. To download the script for this function, go here and save the file as a plain text file, with the extension ".R". (As a note, make sure you use this link for the file—an earlier version of the script was posted on the website of the article, but it's changed some since then, and you want the newer version.) Once you unzip this file, it should be an R script named "attrdl.R". Put this script somewhere convenient to your working directory and source the code in it using the `source` function. For example, if you saved the R script in a subdirectory "code" of the current working directory, you would run:

```
source("code/attrdl.R")
```

This will run all the code in the file, which creates a new function called `attrdl`. This function also has a helpfile, which you can find here.

The `attrdl` function has four required arguments: a vector with the exposure, a crossbasis function, a vector with the cases that correspond to each value in the exposure vector, and the value of the exposure that you want to use as a baseline. You can also include a fitted model. In our example, the exposure will be a vector of the temperature on each day of the first week of July 2012 in London, and the cases vector will be the number of deaths observed on each of those days. We will use 7 degrees C as our baseline. The crossbasis will be the crossbasis object we used to fit our model, and we'll include the fitted model.

We can run the function as:

```
attrdl(x = july_week$tmean, cases = july_week$all,
       basis = temp_basis, model = dlnm_mod_1, cen = 7)
```

```
## [1] 0.02727977
```

By default, it gives us the attributable risk for the week. You can see that this matches the value we got when we did this calculation from scratch, but it certainly saves us a lot of time. This function will also help us tackle tougher cases, like when we have model associations across a distributed lag.

You can also use this function to get the attributable number. You just need to add the argument `type = "an"`:

```
attrdl(x = july_week$tmean, cases = july_week$all,
       basis = temp_basis, model = dlnm_mod_1, cen = 7, type = "an")
```

```
## [1] 20.73262
```

If we want to get the values for each day of the week, we can include the argument `tot = FALSE`. This will give us an estimate for each date, rather than the estimate of the total over the period:

```
attrdl(x = july_week$tmean, cases = july_week$all,
       basis = temp_basis, model = dlnm_mod_1, cen = 7,
       type = "an", tot = FALSE)
```

```
## [1] 1.228037 1.571579 2.855759 6.190165 4.677225 2.480511 1.729346
```

Again, you can see that this agrees with the calculations we did from scratch.

2. **Extend this idea to calculate the attributable risk and attributable number of deaths related to temperature throughout the study period. Calculate these values separately for heat and cold.**

With the `attrdl` function, it's very straightforward to extend these calculations to look at the whole study period, rather than just one week. The only tricky part, conceptually, is that we want to generate separate estimates for heat and cold. Heat-attributable mortality will only occur when the temperature is higher than our baseline of 7 degrees Celsius, while cold-attributable mortality will only occur when the temperature is lower than 7 degrees Celsius.

One way we can do this is to split apart the original dataset into two datasets—one with all the observations on "cold" days (those below 7 degrees C) and one with all the observations on "hot" days (those above 8 degrees C). Then we can run the function to calculate attributable risk and attributable number on these two datasets separately, to get separate estimates for heat-attributable and cold attributable mortality.

```
obs_cold <- obs %>%
  filter(tmean < 7)
obs_hot <- obs %>%
  filter(tmean > 7)

# Estimate cold-related attributable risk over the study period
attrdl(x = obs_cold$tmean, cases = obs_cold$all,
       basis = temp_basis, model = dlnm_mod_1,
       cen = 7)
```

```
## [1] 0.007327224
```

```
# Estimate cold-related attributable number over the study period
attrdl(x = obs_cold$tmean, cases = obs_cold$all,
       basis = temp_basis, model = dlnm_mod_1,
       cen = 7, type = "an")
```

```
## [1] 2414.899
```

```
# Estimate heat-related attributable risk over the study period
attrdl(x = obs_hot$tmean, cases = obs_hot$all,
       basis = temp_basis, model = dlnm_mod_1,
       cen = 7)
```

```
## [1] 0.0235678
```

```
# Estimate cold-related attributable number over the study period
attrdl(x = obs_hot$tmean, cases = obs_hot$all,
       basis = temp_basis, model = dlnm_mod_1,
       cen = 7, type = "an")
```

```
## [1] 23484.98
```

Another way that you can do this is to use the `range` argument in `attrdl`, which allows you to focus on the impacts in only one range of the exposure. For example, you could estimate the heat-related attributable number by using a range from the baseline temperature to the maximum temperature in the observations:

```
attrdl(x = obs$tmean, cases = obs$all,
       basis = temp_basis, model = dlnm_mod_1,
       cen = 7, range = c(7, max(obs$tmean)), type = "an")
```

```
## [1] 23484.98
```

Based on these estimates, there were about 23,500 deaths over the approximately two-decade study period in London that we would expect to not have occurred if the temperature never exceeded 7 degrees C. This makes up about 2% of all mortality on days above 7 degrees C. On the other hand, based on this assessment, the impact from cold is much lower—only about 2,500 attributable

deaths over the approximately two-decade study period, making only about 0.7% of all the deaths on days colder than 7 degrees C.

There are a few caveats with these estimates. First, there are a lot fewer days that were below 7 degrees C in the study than that were above. This accounts for some of the difference in the total number of excess deaths attributable to heat versus cold over the study period. The location of the minimum mortality temperature, which we estimated from fitting a model with only immediate effects in this case, might shift as we include more lags, in which case we would recenter our estimates of attributable risk and attributable number in terms of the baseline temperature we compare to. This could change the numbers some just from having more days above or below the threshold.

Second, we know from fitting the distributed lag non-linear models in an earlier chapter that cold tends to have a more lagged effect on mortality risk, while the effect of heat is more immediate. By limiting the model to lag 0, we've likely missed a lot of the effect of cold on mortality risk. We'll address this in the next part of the exercise, where we'll expand to use a model with a distributed lag.

Finally, the choice of a baseline is tricky here. We're using the minimum mortality temperature, and that's a common choice. However, it's pretty unrealistic to think that there would be a way that this "safest" baseline scenario would ever be met—you'd never be able to enforce the temperature in a city staying constant year-round. Another approach could be to take a sample "mild" summer and "mild" winter for your study city, and compare the health impacts during more severe years to that milder case. This would require a bit more work to calculate, as your baseline temperature would be changing throughout each season. There's not a perfect solution to this issue of identifying a baseline, but it's important to be clear when you communicate the baseline you're using when you describe your results.

Going back to the `attrdl` function, you can use the `tot = FALSE` argument to check out some interesting patterns in attributable numbers over your study. This argument estimates the attributable risk or number for each separate day in the study. For example, you can use it to add a column to your original data with the estimated attributable number for that day:

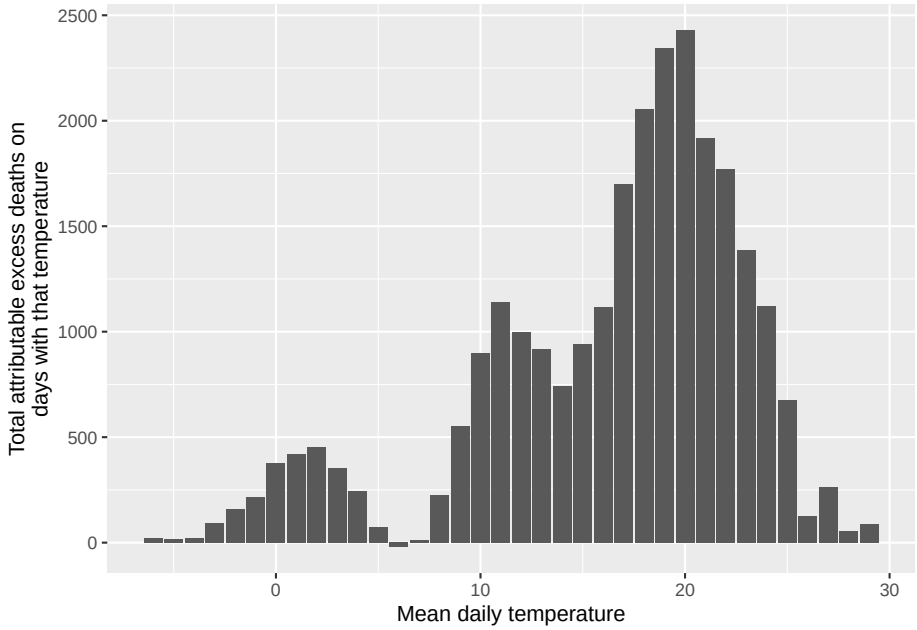
```
obs_an_added <- obs %>%
  select(date, tmean, all) %>%
  mutate(an = attrdl(x = obs$tmean, cases = obs$all,
                     basis = temp_basis, model = dlnm_mod_1,
                     cen = 7, type = "an", tot = FALSE))
obs_an_added
```

```
## # A tibble: 8,279 x 4
##    date       tmean   all      an
##    <date>     <dbl> <dbl>   <dbl>
##  1 1990-01-01  3.91   220  1.09
```

```
##  2 1990-01-02  5.55   257  0.0294
##  3 1990-01-03  4.39   245  0.756
##  4 1990-01-04  5.43   226  0.0685
##  5 1990-01-05  6.87   236 -0.0395
##  6 1990-01-06  9.23   235  1.74
##  7 1990-01-07  6.69   231 -0.0780
##  8 1990-01-08  7.96   235  0.546
##  9 1990-01-09  7.27   250  0.114
## 10 1990-01-10  9.51   214  1.84
## # ... with 8,269 more rows
```

With this, we can look at how the number of excess deaths vary by temperature. We can bin temperatures into broad bins and then plot the total number of attributable deaths for days with each temperature:

```
obs_an_added %>%
  mutate(temp_round = round(tmean)) %>%
  group_by(temp_round) %>%
  summarize(an_tot = sum(an)) %>%
  ggplot(aes(x = temp_round, y = an_tot)) +
  geom_col() +
  labs(x = "Mean daily temperature",
       y = "Total attributable excess deaths on\ndays with that temperature")
```



Remember that our exposure-response function increased dramatically at extreme temperatures, especially extreme heat. Here, however, we see the role that frequency of exposure plays in the ultimate impact. Because extremely hot

and extremely cold days are very rare, they don't account for much of the heat- or cold-attributable mortality. Instead, a lot more comes at temperatures that are a bit milder than those extremes but that occur much more often. This, of course, might change depending on the baseline we're using for comparison.

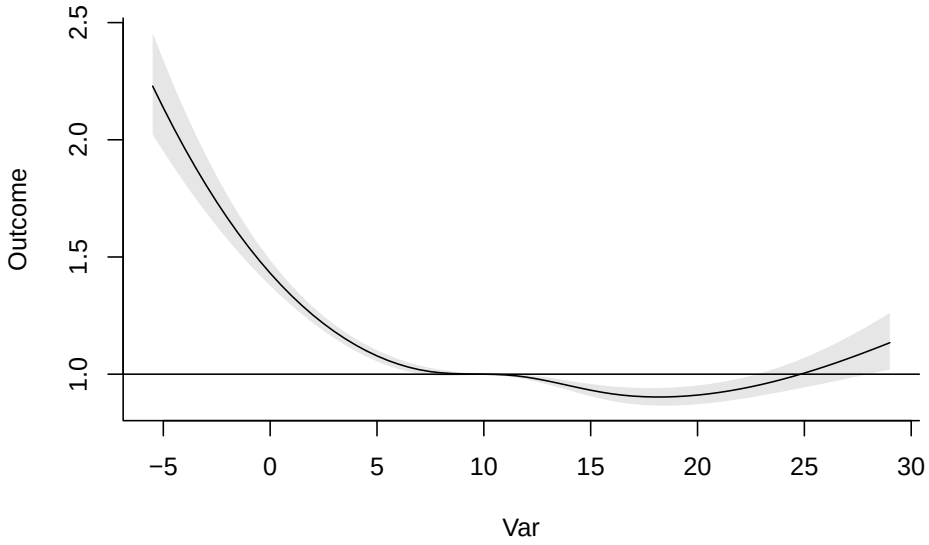
3. **Do the same calculation as in the previous part, but change the underlying model that you're using. In this case, use the `dist_lag_mod_5` you fit in Chapter 4. What differences are there in attributable risk and attributable number when you use a model that incorporates lagged effects (`dist_lag_mod_5`) compared to one that only fits the immediate association (`dlnm_mod_1`)?**

To start, re-run the code we used in Chapter 4 to fit `dist_lag_mod_5`, including the code to create the associated crossbasis function (`dl_basis_4`):

```
dl_basis_4 <- crossbasis(obs$tmean, lag = 30,
                         argvar = list(fun = "ns", df = 4),
                         arglag = list(fun = "ns", df = 6))
dist_lag_mod_5 <- glm(all ~ dl_basis_4 +
                        factor(dow, ordered = FALSE) +
                          ns(time, df = 158),
                        data = obs, family = "quasipoisson")
```

First, let's take a look to see if we should use a different baseline—in other words, is 7 degrees C still our best estimate of the minimum mortality temperature, now that we're incorporating lagged effects? If we use `crosspred`, we can check the exposure-response curve for the cumulative effects over all the fitted lags, to see what the minimum point is on that curve:

```
crosspred(dl_basis_4, dist_lag_mod_5, bylag = 1) %>%
  plot(ptype = "overall")
```



Now that we're considering lagged effects, the temperature with minimum mortality risk has gone up quite a lot, to 18 degrees C. We'll use that as our baseline in estimating attributable risk and number for heat and cold. We can already tell that this will increase our cold-attributable mortality by a bit most likely, since there will be a lot more days that are below this baseline temperature now. Following the same reasoning, this is also likely to decrease the heat-attributable mortality (although of course changes to the shape of the exposure-response curve will also affect our estimates).

The other thing that we need to think about is how to incorporate lagged effects of temperature. You can do this by thinking of temperatures at different lags as different exposures. The mathematics for this gets more complicated, but fortunately the `attrdl` function can handle those mechanics for us.

We do, however, have to decide whether we want to calculate these attributable impacts from lagged exposures under what's called a "backward perspective" or a "forward perspective". One of the required readings for this week has a full discussion of these two perspectives and the implications and assumptions of each (Gasparrini and Leone, 2014). Simply put, the forward perspective is describing how one day's temperature has impacts on its same day as well as following days, while the backward perspective calculates the impact that all the exposures in the lagged period before a certain day impact risk on that day.

There is a way to calculate the separate impacts of heat and cold under either of these perspectives, but the forward perspective can be prone to some bias (Gasparrini and Leone, 2014), so we'll use the backward perspective. Since we want to get estimates for heat and cold separately, we'll use the `range` argument in the `attrdl` function. (With the lagged effects, it would be tricky to try to do this by separating the dataset into hot and cold days, since you'll have some cases when you have temperatures that are both below and above the baseline

temperature in the lag period.) You can specify that you'd like to use the backward perspective with the argument `dir = "back"`:

```
# Estimate cold-associated attributable risk
attrdl(x = obs$tmean, cases = obs$all,
       basis = dl_basis_4, model = dist_lag_mod_5,
       cen = 18, range = c(min(obs$tmean), 18), dir = "back")
```

```
## [1] 0.09398662
```

```
# Estimate cold-associated attributable number
attrdl(x = obs$tmean, cases = obs$all,
       basis = dl_basis_4, model = dist_lag_mod_5,
       cen = 18, range = c(min(obs$tmean), 18), dir = "back", type = "an")
```

```
## [1] 124632.4
```

```
# Estimate heat-associated attributable risk
attrdl(x = obs$tmean, cases = obs$all,
       basis = dl_basis_4, model = dist_lag_mod_5,
       cen = 18, range = c(18, max(obs$tmean)), dir = "back")
```

```
## [1] 0.002380825
```

```
# Estimate heat-associated attributable number
attrdl(x = obs$tmean, cases = obs$all,
       basis = dl_basis_4, model = dist_lag_mod_5,
       cen = 18, range = c(18, max(obs$tmean)), dir = "back", type = "an")
```

```
## [1] 3157.129
```

When we use this model that includes distributed lag effects of temperature, you can see that the impact of cold has increased substantially and the impact of heat has decreased substantially. This is because cold tends to have effects that are lagged after the initial exposure, while heat tends to be more immediate, and can even have somewhat of a pattern reflective of mortality displacement (although this depends on the study city). Once we incorporate the delayed effects of cold, the minimum mortality temperature also increase substantially, so many more days are below, rather than above, the baseline temperature we're using for comparison as compared to the last part of the exercise.

These long lagged effects of temperature start to bring in some questions about whether the effect is specifically from the absolute value of the temperature, or whether they may be starting to pick up some impacts that are caused by seasonality, rather than absolute temperature. As you include lags up to a month or longer, it gets harder to separate the impact of temperature specifically from the impact of season. This question is very important when you start thinking about the impacts of climate change. With climate change, we expect in many places that winter days will, on average, have a higher temperature. If the mortality risk of cold is caused by the absolute temperature, this would result in

fewer cold-related deaths. However, if there are other seasonal factors (that we don't expect to change, or at least not as much), then the rising winter temperatures would have less of a beneficial effect on winter mortality risk. There are a number of interesting papers that focus on this question and its implications for climate change projections (Kinney et al., 2015; Ebi, 2015; Hajat and Gasparrini, 2016). Also, there are some papers that go more deeply into how to estimate minimum mortality temperatures (Lee et al., 2017; Tobıas et al., 2017), as well as what baseline temperature makes a suitable threshold for estimating attributable risk for cold-related mortality (Arbuthnott et al., 2018).

6.3 Quantifying potential health impacts under different scenarios

You can use measures of attributable risk and attributable number to communicate the estimated impacts of the exposure in the population that you studied. However, you can also use this same idea to investigate the potential health impacts of an exposure under different scenarios. The basic framework on these calculations was to compare one scenario (a baseline of no, limited, or "safest" exposure) to another scenario (the observed levels of exposure in the population). It's fairly straightforward to transfer this idea to broader applications—any case where you would like to compare one scenario of exposure to another scenario of exposure.

You could do this a few ways. One is that you could calculate attributable impacts under both scenarios in comparison to some baseline, and compare the final impact estimates. One of this week's readings (Vicedo-Cabrera et al., 2019) describes this process, adding a code tutorial in the supplement. It describes the process in the context of projecting the potential heat-related mortality impacts of climate change, compared to present-day impacts.

There are a few details that make this process a bit different from calculating attributable impacts based on the study data you used to fit the model. First, you do not have observations of exposure. Instead, you will use scenarios of exposure, generated under certain assumptions. For climate change projections, there are large climate modeling groups that have generated scenarios of future weather conditions under scenarios that are tied to assumptions about how drivers of climate change will change in the future. These scenarios can be used in place of exposure observations to project potential health impacts. Of course, there are more uncertainties that are introduced in this case, compared to observed exposures, since there is some uncertainty in those projections. Further, it is typically best to work with atmospheric scientists in deriving exposure scenarios from these projections, as you'll need expertise to appropriately convert the climate model output to an appropriate representation of temperature exposures in a future period.

Second, you will not have the observed number of cases (for example, deaths)

on each day of a hypothetical scenario. Instead, you will need to incorporate an estimate of mortality each day in the scenario. One approach is to use present-day average daily mortality counts, as this allows you to isolate the role of climate change, separate from changes in demographics. Further, if you are considering scenarios that happen in the near future, this is a very reasonable approach. However, if you're considering scenarios a few decades in the future, this won't give a good estimate of the impact we might actually expect. One approach to try to improve this estimate is to use demographic projections of population into the future. You can pair this with mortality rates in certain subsets of the population (these could be, for example, by age and race) to try to generate a more realistic estimate of future impacts.

Finally, when you use the estimate exposure-response functions that are estimated with present-day data to project impacts in a future period, there are a number of reasons that the exposure-response function might not provide as good of an estimate of how temperature would impact health. Exposure-response functions for temperature are not the same in every city. Instead, there's clear evidence that, in cities with cooler climates, heat tends to start affecting risk at a lower temperature, and, in cities with hotter climates, cold tends to start affecting risk at a higher temperature. In other words, there's not an "absolute" shape, tied to specific temperature values, that can universally describe how mortality risk changes with temperature. Further, there is evidence that these exposure-response functions have changed in the same city over time. This all suggests that there's likely some influence from adaptation on these exposure-response functions. For example, a decreasing risk from heat over the 20th century likely was, at least in part, caused by adaptation through the invention and increased use of air conditioning, which reduces exposure to heat for at least some of the population at least some of the time. Other changes can come from changes in the population structure. For example, we know that temperature-related risk tends to be higher in older versus younger adults, so if a population is aging, that could change the shape of its overall exposure-response function, even without other changes. These questions are an area of continuing development for climate epidemiologists. Good discussions are available in a number of papers, including Kinney et al. (2008), Arbuthnott et al. (2016), Gosling et al. (2017), and Kinney (2018).

Chapter 7

Longitudinal cohort study designs

7.1 Readings

The required readings for this chapter are:

- Andersson et al. (2019) Provides background on the study we'll use in this chapter for our example data
- Wong et al. (1989) An epidemiological study using the study data.

There are also some supplemental readings you may find useful. The following are a series of instructional papers on survival analysis, that are meant as general background on how to fit survival analysis models:

- Clark et al. (2003)
- Bradburn et al. (2003a)
- Bradburn et al. (2003b)

This article is a good summary for the limitations of Hazards Ratios and offers an alternative survival analysis approach enabling the plotting of adjusted cumulative incidence (and similarly survival rate) curves:

- Hernán (2010) (Note that there is an online erratum for this article: "On page 15, column 1, second full paragraph: '… and the average HR is ensured to reach the value 1.' should instead say, '… and the risk ratio is ensured to reach the value 1.' ")

These articles provide more background on the Framingham Heart Study:

- Dawber et al. (1951) A paper from the 1950s, this presents the rationale behind the design of the Framingham Heart Study
- Dawber et al. (2015) A reprint of the paper describing the first results to come out of the Framingham Heart Study

These articles provide some general reviews on our current understanding of the epidemiology and etiology of heart disease and its risk factors:

- Wong (2014) Review of large epidemiological studies on coronary heart disease and key findings
- Ziaeian and Fonarow (2016) Review on the epidemiology and etiology of heart failure
- Valenzuela et al. (2021) Review of risk factors for hypertension
- Zhou et al. (2021) Review on global epidemiology of hypertension

7.2 Longitudinal cohort data

Our example data for this chapter comes from the Framingham Heart Study. This is a cohort study that began in 1948 (Andersson et al., 2019). At the time (and continuing today), heart disease was the most common cause of death in the US—a notable shift from earlier times, when infectious diseases played a larger role in mortality. While heart disease was an important cause of death, however, very little was known about risk factors for heart disease, outside of a little concerning the role of some infectious and nutritional diseases that affected the heart (Dawber et al., 1951; Andersson et al., 2019). This study was designed to collect a group of people without evident cardiovascular disease (although this restriction was eased in practice) and track them over many years, to try to identify risk factors for developing cardiovascular disease. The study subjects were tracked over time, with data regularly collected on both risk factors and cardiovascular outcomes. The study was revolutionary in identifying some key risk factors for coronary heart disease, including elevated blood pressure, high cholesterol levels, being overweight, and smoking (Andersson et al., 2019). It was also revolutionary in identifying high blood pressure as a risk for other cardiovascular outcomes, including stroke and congestive heart failure (Andersson et al., 2019).

The original cohort included about 5,200 people from the town of Framingham, MA. The example data is a subset of data from this original cohort. You can download the example dataset for this class by clicking here and then saving the content of the page as a csv file (we recommend using the original filename of "frmgham2.csv"). There is also a codebook file that comes with these data, which you can download for this class by clicking here. This codebook includes some explanations about the columns in the data, as well as how multiple measurements from a single study subject are included in the data.

The data are saved in a csv format, and so they can be read into R using the `read_csv` function from the `readr` package (part of the tidyverse). You can use

the following code to read in these data, assuming you have saved them in a "data" subdirectory of your current working directory:

```
library(tidyverse) # Loads all the tidyverse packages, including readr
fhs <- read_csv("data/frmgham2.csv")
fhs
```

```
## # A tibble: 11,627 x 39
##    RANDID   SEX TOTCHOL   AGE SYSBP DIABP CURSMOKE CIGPDAY   BMI DIABETES BPMEDS
##     <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>  <dbl>
##  1   2448     1     195    39  106   70          0       0  27.0        0      0
##  2   2448     1     209    52  121   66          0       0  NA          0      0
##  3   6238     2     250    46  121   81          0       0  28.7        0      0
##  4   6238     2     260    52  105   69.5        0       0  29.4        0      0
##  5   6238     2     237    58  108   66          0       0  28.5        0      0
##  6   9428     1     245    48  128.  80          1      20  25.3        0      0
##  7   9428     1     283    54  141   89          1      30  25.3        0      0
##  8  10552     2     225    61  150   95          1      30  28.6        0      0
##  9  10552     2     232    67  183  109          1      20  30.2        0      0
## 10  11252     2     285    46  130   84          1      23  23.1        0      0
## # ... with 11,617 more rows, and 28 more variables: HEARTRTE <dbl>,
## #   GLUCOSE <dbl>, educ <dbl>, PREVCHD <dbl>, PREVAP <dbl>, PREVMI <dbl>,
## #   PREVSTRK <dbl>, PREVHYP <dbl>, TIME <dbl>, PERIOD <dbl>, HDLC <dbl>,
## #   LDLC <dbl>, DEATH <dbl>, ANGINA <dbl>, HOSPMI <dbl>, MI_FCHD <dbl>,
## #   ANYCHD <dbl>, STROKE <dbl>, CVD <dbl>, HYPERTEN <dbl>, TIMEAP <dbl>,
## #   TIMEMI <dbl>, TIMEMIFC <dbl>, TIMECHD <dbl>, TIMESTRK <dbl>, TIMECVD <dbl>,
## #   TIMEDTH <dbl>, TIMEHYP <dbl>
```

You can find full details on the structure of this data in the codebook. At a broad scale, note that it includes several health outcomes related to heart disease, which the codebook calls "events" (`DEATH`: indicator of death from any cause; `ANYCHD`: indicator of one of several types of events related to coronary heart disease; `HOSPMI`: Hopitalization for myocardial infarction [heart attack], etc.). The data also includes a number of risk factors that the study researchers hypothesized might be linked to cardiovascular disease (`CURSMOKE`: if the study subject is currently a smoker; `TOTCHOL`: serum total cholesterol; `SYSBP`, `DIABP`: measures of the systolic and diastolic blood pressure, respectively; `BMI`: Body Mass Index; etc.). Finally, there are some characteristics of the study subject, like age (`AGE`) and sex (`SEX`), as well as some variables that are connected to either the time of the record or the time of certain events (or of censoring, if the event did not happen during follow-up).

As you look through the data, pay attention to some features that are more characteristic of cohort studies, compared to features of the time series data we worked with in earlier chapters:

- One important difference compared to a time-series dataset is the `RANDID` variable. This is the unique identifier for unit for which we have repeated

observations for over time. In this case the `RANDID` variable represents a unique identifier for each study participant, with multiple observations (rows) per participant over time.
- The `TIME` variable indicates the number of days that have elapsed since beginning of follow-up of each observation. `TIME` is always 0 for the first observation of each participant (when `PERIOD` equals 1, for the first examination), and then for following measurements will track the time since follow-up started for that study participant.
- The number of observations varies between participants (typical of many cohort studies)
- The time spacing between observations is not constant. This is because the repeated observations in the Framingham Heart Study are the result of follow-up exams happening 3 to 5 years apart. Many longitudinal cohorts will instead have observations over a fixed time interval (monthly, annual, biannual etc), resulting in a more balanced dataset.
- Observations are given for various risk factors, covariates and cardiovascular outcomes. Some will be invariant for each participant over time (`SEX`, `educ`), while others will vary with each exam.

From a data management perspective, we might want to change all the column names to be in lowercase, rather than uppercase. This will save our pinkies some work as we code with the data! You can make that change with the following code, using the `str_to_lower` function from the `stringr` package (part of the `tidyverse`):

```
fhs <- fhs %>%
  rename_all(.funs = str_to_lower)
fhs
```

```
## # A tibble: 11,627 x 39
##    randid   sex totchol   age sysbp diabp cursmoke cigpday   bmi diabetes bpmeds
##     <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>  <dbl>
##  1   2448     1     195    39  106   70          0       0  27.0        0      0
##  2   2448     1     209    52  121   66          0       0  NA          0      0
##  3   6238     2     250    46  121   81          0       0  28.7        0      0
##  4   6238     2     260    52  105   69.5        0       0  29.4        0      0
##  5   6238     2     237    58  108   66          0       0  28.5        0      0
##  6   9428     1     245    48  128.  80          1      20  25.3        0      0
##  7   9428     1     283    54  141   89          1      30  25.3        0      0
##  8  10552     2     225    61  150   95          1      30  28.6        0      0
##  9  10552     2     232    67  183  109          1      20  30.2        0      0
## 10  11252     2     285    46  130   84          1      23  23.1        0      0
## # ... with 11,617 more rows, and 28 more variables: heartrte <dbl>,
## #   glucose <dbl>, educ <dbl>, prevchd <dbl>, prevap <dbl>, prevmi <dbl>,
## #   prevstrk <dbl>, prevhyp <dbl>, time <dbl>, period <dbl>, hdlc <dbl>,
## #   ldlc <dbl>, death <dbl>, angina <dbl>, hospmi <dbl>, mi_fchd <dbl>,
## #   anychd <dbl>, stroke <dbl>, cvd <dbl>, hyperten <dbl>, timeap <dbl>,
```

```
## #   timemi <dbl>, timemifc <dbl>, timechd <dbl>, timestrk <dbl>, timecvd <dbl>,
## #   timedth <dbl>, timehyp <dbl>
```

To look a bit more closely at how this dataset works, let's take a look just at the observations of the oldest study subjects at the first examination, those 69 or older (`age >= 69`) at their first examination (`period == 1`):

```
oldest_subjects <- fhs %>%
  filter(period == 1 & age >= 69) %>%
  pull(randid)
oldest_subjects
```

```
##  [1] 3603542 3762702 4726021 7198139 7351212 7568367 7659676 7695005 8723664
## [10] 9643995 9686180 9789948
```

Once we have the IDs of these study subjects (`oldest_subjects`), we can pull out the study data just for them. I'm limiting to a few columns: their ID (`randid`), the time of each examination (`time`), the number of each examination (`period`), whether they died during follow-up (`died`), and the number of days between the first examination and either death (if they died during follow-up) or censoring (i.e., stopped tracking the subject or lost them to follow-up) (`timedth`):

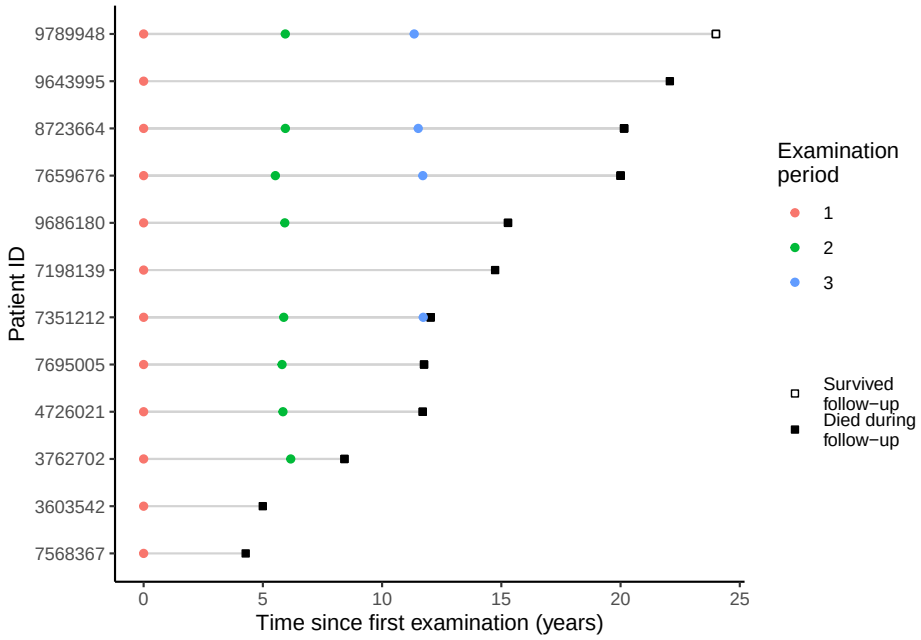
```
fhs %>%
  filter(randid %in% oldest_subjects) %>%
  select(randid, time, period, death, timedth)
```

```
## # A tibble: 24 x 5
##     randid  time period death timedth
##      <dbl> <dbl>  <dbl> <dbl>   <dbl>
##  1 3603542     0      1     1    1825
##  2 3762702     0      1     1    3076
##  3 3762702  2253      2     1    3076
##  4 4726021     0      1     1    4275
##  5 4726021  2134      2     1    4275
##  6 7198139     0      1     1    5384
##  7 7351212     0      1     1    4396
##  8 7351212  2146      2     1    4396
##  9 7351212  4283      3     1    4396
## 10 7568367     0      1     1    1563
## # ... with 14 more rows
```

We can explore this subset of the data by plotting, for each of these study subjects, the timing of each of their examination periods, whether they died during follow-up, and the timing of their death or censoring:

```
fhs %>%
  filter(randid %in% oldest_subjects) %>%
  select(randid, time, period, death, timedth) %>%
```

```
  mutate(randid = as_factor(randid),
         randid = fct_reorder(randid, timedth),  # Arrange by time to death
         period = as_factor(period),
         death = as_factor(death),
         timey = time / 365.25,                  # Convert from days to years
         timedthy = timedth / 365.25) %>%
  ggplot() +
  geom_segment(aes(x = 0, xend = timedthy,
                   y = randid, yend = randid), color = "lightgray") +
  geom_point(aes(x = timedthy, y = randid, fill = death), shape = 22) +
  geom_point(aes(x = timey, y = randid, color = period)) +
  theme_classic() +
  labs(x = "Time since first examination (years)",
       y = "Patient ID",
       color = "Examination\nperiod") +
  scale_fill_manual(name = "", values = c("white", "black"),
                    labels = c("Survived\nfollow-up", "Died during\nfollow-up"))
```



You can see that we have at least one examination (period 1) for each of the study subjects, and for some we have as many as three. One of these study subjects was tracked for almost 25 years without a recorded death (subject ID 9789948). All the other subjects in this subset died during follow-up. Some died within a few years of the first examination, and so did not survive to a second or later examination. Others survived longer but missed some later examinations. As you work with these data, keep in mind that they have multiple measurements

(rows) for some but not all of the study subjects, and that events are recorded both in terms of whether they happened during follow-up (e.g., `death`) and also how long after the first examination the event occurred or the data for the subject was censored (e.g., `timedth`).

Applied exercise: Exploring longitudinal cohort data

Read the example cohort data in R and explore it to answer the following questions:

1. What is the number of participants and number of observations in the `fhs` dataset?
2. Is there any missingness in the data?
3. How many participants died during the observation period? What is the distribution of age at time of death?
4. What is the distribution of BMI among MI cases and non-cases? How about between smokers and non-smokers?

Based on this exploratory exercise, talk about the potential for confounding when these data are analyzed to estimate the association between smoking and risk of incident MI.

Applied exercise: Example code

1. **What is the number of participants and the number of observations in the `fhs` dataset? (i.e what is the sample size and number of person-time observations)**

In the `fhs` dataset, the number of participants will be equal to the number of unique ID's (The `RANDID` variable which takes a unique value for each participant). We can extract this using the `unique` function nested within the `length` function

```
length(unique(fhs$randid))
```

```
## [1] 4434
```

If you'd like to use `tidyverse` tools to answer this question, you can do that, as well. The pipe operator (`%>%`) works on any type of object—it will take your current output and include it as the first parameter value for the function call you pipe into. If you want to perform operations on a column of a dataframe, you can use `pull` to extract it from the dataframe as a vector, and then pipe that into vector operations:

```
fhs %>%
  pull(randid) %>%
  unique() %>%
  length()
```

```
## [1] 4434
```

It's entirely a personal choice whether you use the `$` operator and "nesting" of function calls, versus `pull` and piping to do a series of function calls. You can see you get the same result, so it just comes down to the style that you will find easiest to understand when you look at your code later.

The number of person-time observations will be equal to the length of the dataset, since there's a row for every observation taken. The `dim` function gives us the length (number of rows) and width (number of columns) for a dataframe or any matrix like object in R.

```
dim(fhs)
```

```
## [1] 11627    39
```

We see that there are 11,626 observations, which is an average of approximately 2 to 3 observations per participant (11,626 / 4,434 = 2.6).

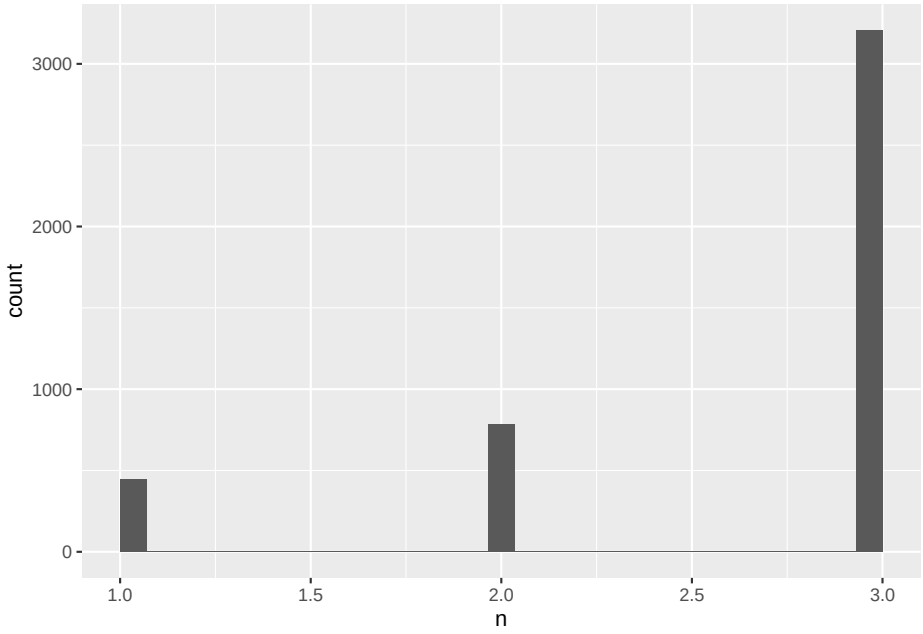
When you know there are repeated measurements, it can be helpful to explore how much variation there is in the number of observations per study subject. You could do that in this dataset with the following code:

```
fhs %>%
  # Group by the study subject identifier and then count the rows for each
  group_by(randid) %>%
  count() %>%
  # Reorder the dataset so the subjects with the most observations come first
  arrange(desc(n)) %>%
  head()
```

```
## # A tibble: 6 x 2
## # Groups:   randid [6]
##   randid     n
##    <dbl> <int>
## 1   6238     3
## 2  11252     3
## 3  11263     3
## 4  12806     3
## 5  14367     3
## 6  16365     3
```

You can visualize this, as well. A histogram is one good choice:

```
fhs %>%
  # Group by the study subject identifier and then count the rows for each
  group_by(randid) %>%
  count() %>%
  ggplot(aes(x = n)) +
  geom_histogram()
```



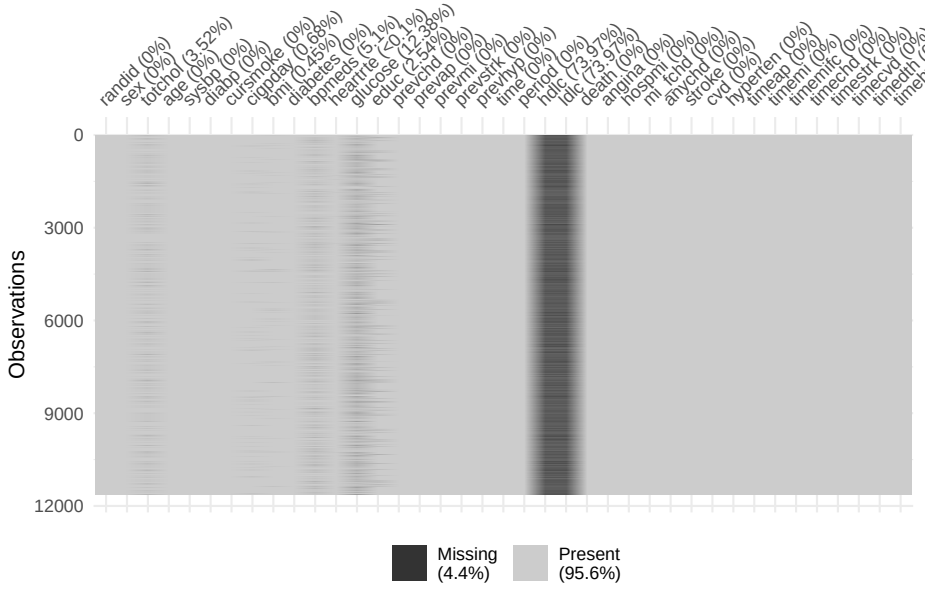
All study subjects have between one and three measurements. Most of the study subjects (over 3,000) have three measurements recorded in the dataset.

2. **Is there any missingness in the data?**

We can check for missingness in a number of ways. There are a couple of great packages, `visdat` and `naniar`, that include functions for investigating missingness in a dataset. If you don't have these installed, you can install them using `install.packages("naniar")` and `install.packages("visdat")`. The `naniar` package has a vignette with examples that is a nice starting point for working with both packages.

The `vis_miss` function from the `visdat` package shows missingness in a dataset in a way that lets you get a top-level snapshot:

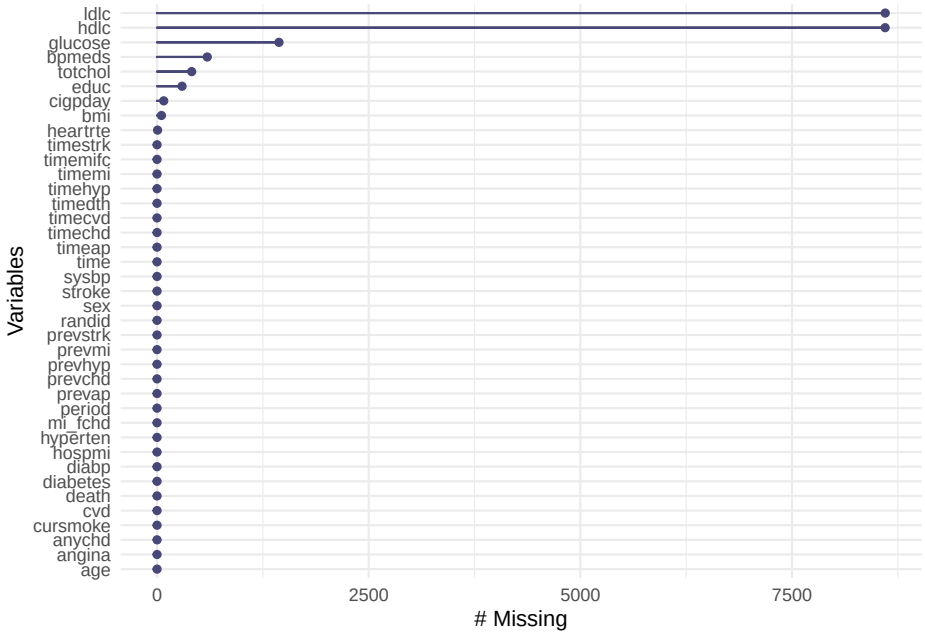
```
library(visdat)
vis_miss(fhs)
```



This shows you how much data is missing for each column in the data. For a smaller dataset, the design of the plot would also let you see how often missing data line up across several columns for the same observation (in other words, if an observation that's missing one measurement tends to also be missing several other measurements). In this case, however, there are so many rows, it's a bit hard to visually line up missingness by row.

Another was to visualize this is with `gg_miss_var` from the `naniar` package:

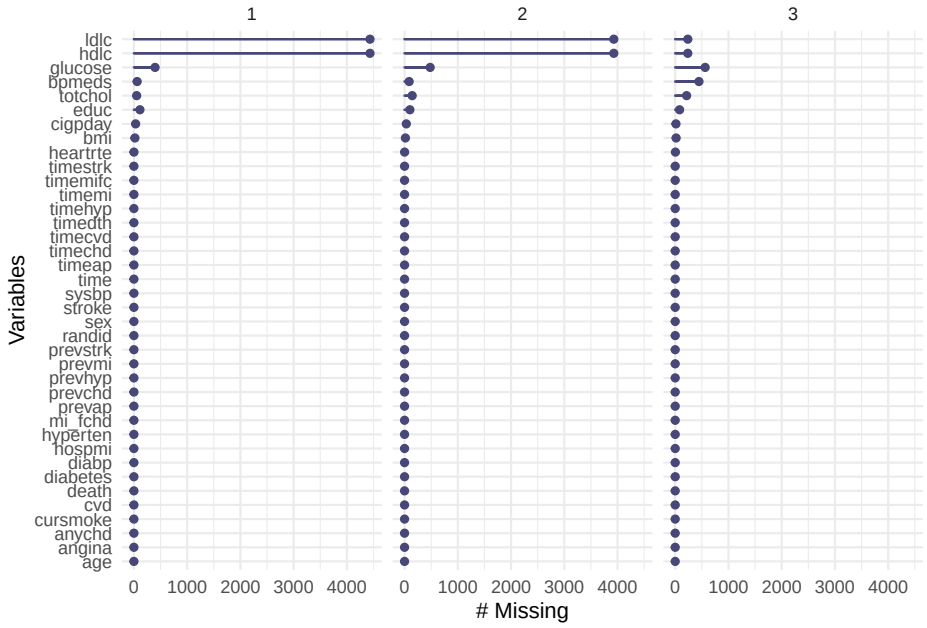
```
library(naniar)
gg_miss_var(fhs)
```



This output again focuses on missingness by column in the data, helping you identify columns where we might not have many non-missing observations. In this case, many of the columns have measurements that are available for all observations, with no missingness, including records of the subject's ID, measures of death, stroke, CVD and other events, age, sex, and BMI. Some of the measured values from visits are missing occasionally, like the total cholesterol, and glucose. Other measures asked of the participants (number of cigarettes per day, education) are occasionally missing. Two of the variables—`hdlc` and `ldlc` (High Density Lipoprotein Cholesterol and Low Density Lipoprotein Cholesterol, respectively)—are missing more often than they are available. If you read the codebook for the data, you'll see that this is because these measurements are only available at time period 3.

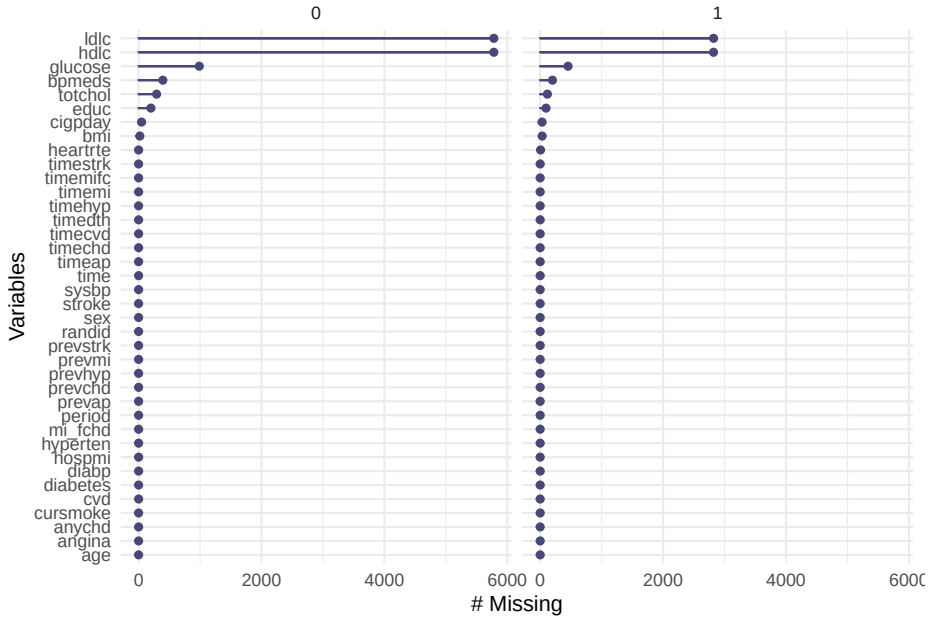
You can also do faceting with the `gg_miss_var` function. For example, you could see if missingness varies by the period of the observation:

```
gg_miss_var(fhs, facet = period)
```



You may also want to check if missingness varies with whether an observation was associated with death of the study subject:

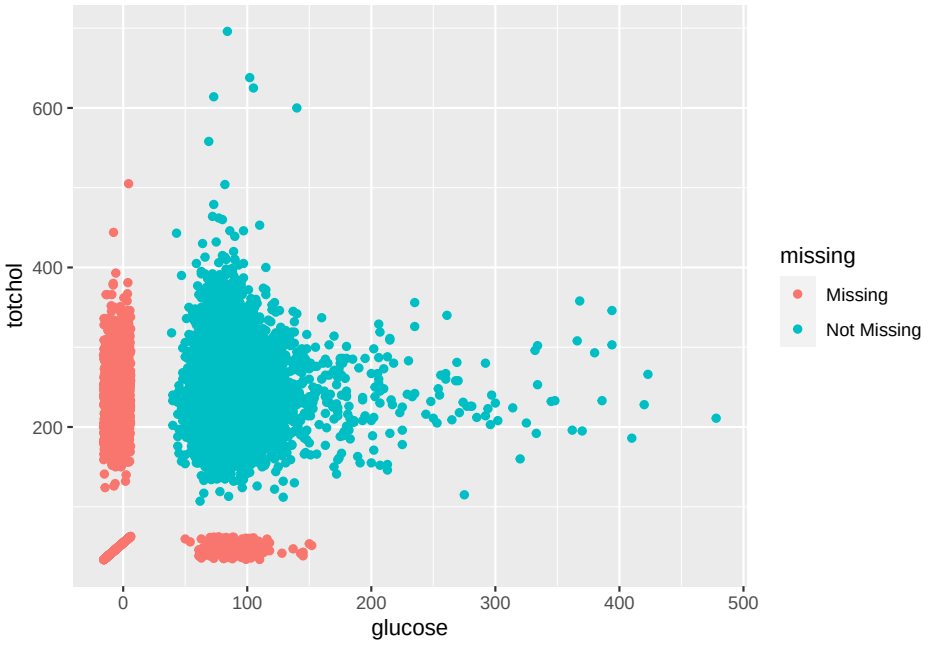
```
gg_miss_var(fhs, facet = death)
```



There are also functions in these packages that allow you to look at how miss-

ingness is related across variables. For example, both `glucose` and `totchol` are continuous variables, and both are occasionally missing. You can use the geom function `geom_miss_point` from the `naniar` package with a ggplot object to explore patterns of missingness among these two variables:

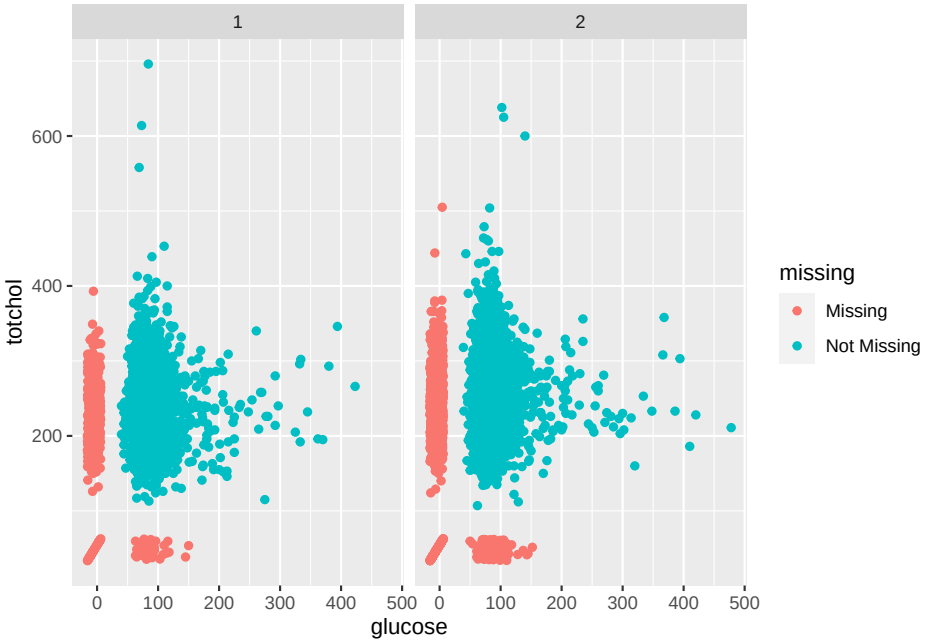
```
fhs %>%
  ggplot(aes(x = glucose, y = totchol)) +
  geom_miss_point()
```



The lower left corner shows the observations where both values are missing—it looks like there aren't too many. For observations with one missing but not the other (the points in red along the x- and y-axes), it looks like the distribution across the non-missing variable is pretty similar to that for observations with both measurements available. In other words, `totchol` has a similar distribution among observations where `glucose` is available as observations where `glucose` is missing, and the same for `glucose` for observations with and without missingess for `totchol`.

Since this function interfaces with `ggplot`, you can use any usual tricks with ggplot in association with it. For example, you can do things like facet by sex to explore patterns of missingness and codistribution at a finer level:

```
fhs %>%
  ggplot(aes(x = glucose, y = totchol)) +
  geom_miss_point() +
  facet_wrap(~ sex)
```



3. **How many participants died during the observation period? What is the distribution of age at time of death?**

The `death` variable in the `fhs` data is an indicator for mortality if a participant died at any point during follow-up. It is time-invariant: that is, it takes the value 1 if a participant died at any point during the follow-up period or 0 if they were alive at their end of follow-up, so we have to be careful on how to extract the actual number of deaths.

If you arrange by the random ID and look at `period` and `death` for each subject, you can see that the `death` variable is the same for all periods for each subject (this is what we mean by it being "time-invariant" in the data):

```
fhs %>%
  arrange(randid) %>%
  select(randid, period, death)
```

```
## # A tibble: 11,627 x 3
##    randid period death
##     <dbl>  <dbl> <dbl>
##  1   2448      1     0
##  2   2448      3     0
##  3   6238      1     0
##  4   6238      2     0
##  5   6238      3     0
##  6   9428      1     0
##  7   9428      2     0
```

```
##  8  10552      1     1
##  9  10552      2     1
## 10  11252      1     0
## # ... with 11,617 more rows
```

We need to think some about this convention of recording the data when we count the deaths.

It is often useful to extract the first (and sometimes last) observation, in order to assess certain covariate statistics on the individual level. We can create a dataset including only the first (or last) observation per participant from the `fhs` data using `tidyverse` tools. The `group_by` functions groups data by unique values of designated variables (here `randid`) and the `slice` function selects rows as designated. Here is an example of extracting the first row (`group_by(randid) %>% slice(1L)`) for each study subject:

```
fhs_first <- fhs %>%
  group_by(randid) %>%
  slice(1L) %>% # The L after the one clarifies that this is an integer
  ungroup()
fhs_first
```

```
## # A tibble: 4,434 x 39
##    randid   sex totchol   age sysbp diabp cursmoke cigpday   bmi diabetes bpmeds
##     <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>  <dbl>
##  1   2448     1     195    39  106     70        0       0  27.0        0      0
##  2   6238     2     250    46  121     81        0       0  28.7        0      0
##  3   9428     1     245    48  128.    80        1      20  25.3        0      0
##  4  10552     2     225    61  150     95        1      30  28.6        0      0
##  5  11252     2     285    46  130     84        1      23  23.1        0      0
##  6  11263     2     228    43  180    110        0       0  30.3        0      0
##  7  12629     2     205    63  138     71        0       0  33.1        0      0
##  8  12806     2     313    45  100     71        1      20  21.7        0      0
##  9  14367     1     260    52  142.    89        0       0  26.4        0      0
## 10  16365     1     225    43  162    107        1      30  23.6        0      0
## # ... with 4,424 more rows, and 28 more variables: heartrte <dbl>,
## #   glucose <dbl>, educ <dbl>, prevchd <dbl>, prevap <dbl>, prevmi <dbl>,
## #   prevstrk <dbl>, prevhyp <dbl>, time <dbl>, period <dbl>, hdlc <dbl>,
## #   ldlc <dbl>, death <dbl>, angina <dbl>, hospmi <dbl>, mi_fchd <dbl>,
## #   anychd <dbl>, stroke <dbl>, cvd <dbl>, hyperten <dbl>, timeap <dbl>,
## #   timemi <dbl>, timemifc <dbl>, timechd <dbl>, timestrk <dbl>, timecvd <dbl>,
## #   timedth <dbl>, timehyp <dbl>
```

Alternatively you can use the `slice_head` function, which allows us to slice a designated number of rows beginning from the first observation. Because we are piping this in the `group_by` function, we will be slicing rows beginning from the first observation for each `randid`:

```
fhs_first <- fhs %>%
  group_by(randid) %>%
  slice_head(n = 1) %>%
  ungroup()
```

We can similarly select the last observation for each participant:

```
fhs_last <- fhs %>%
  group_by(randid) %>%
  slice(n()) %>% # The `n()` function gives the count of rows in a group
  ungroup()
fhs_last
```

```
## # A tibble: 4,434 x 39
##    randid   sex totchol   age sysbp diabp cursmoke cigpday   bmi diabetes bpmeds
##     <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>  <dbl>
##  1   2448     1     209    52   121    66        0       0  NA          0      0
##  2   6238     2     237    58   108    66        0       0  28.5        0      0
##  3   9428     1     283    54   141    89        1      30  25.3        0      0
##  4  10552     2     232    67   183   109        1      20  30.2        0      0
##  5  11252     2      NA    58   155    90        1      30  24.6        0      0
##  6  11263     2     220    55   180   106        0       0  31.2        1      1
##  7  12629     2     220    70   149    81        0       0  36.8        0      0
##  8  12806     2     320    57   110    46        1      30  22.0        0      0
##  9  14367     1     280    64   168   100        0       0  25.7        0      0
## 10  16365     1     211    55   173   123        0       0  29.1        0      1
## # ... with 4,424 more rows, and 28 more variables: heartrte <dbl>,
## #   glucose <dbl>, educ <dbl>, prevchd <dbl>, prevap <dbl>, prevmi <dbl>,
## #   prevstrk <dbl>, prevhyp <dbl>, time <dbl>, period <dbl>, hdlc <dbl>,
## #   ldlc <dbl>, death <dbl>, angina <dbl>, hospmi <dbl>, mi_fchd <dbl>,
## #   anychd <dbl>, stroke <dbl>, cvd <dbl>, hyperten <dbl>, timeap <dbl>,
## #   timemi <dbl>, timemifc <dbl>, timechd <dbl>, timestrk <dbl>, timecvd <dbl>,
## #   timedth <dbl>, timehyp <dbl>
```

or using the `slice_tail` function:

```
fhs_last <- fhs %>%
  group_by(randid) %>%
  slice_tail(n = 1) %>%
  ungroup()
```

In this dataset we can extract statistics on baseline covariates (i.e., at the first examination) on the individual level, but also assess the number of participants with specific values, including `death = 1`. For example, we can use the `sum` function in base R, which generates the sum of all values for a given vector. In this case since each death has the value of 1, the `sum` function will give us the number of deaths in the sample.

```
sum(fhs_first$death)
```

```
## [1] 1550
```

Conversely using `tidyverse` tools we can extract the number of observations with `death = 1` using the `count` function:

```
fhs_first %>%
  count(death)
```

```
## # A tibble: 2 x 2
##   death     n
## * <dbl> <int>
## 1     0  2884
## 2     1  1550
```

Based on this analysis, 1,550 of the study subjects, or about 35% of them, died during follow-up for this study.

Note that, unlike in this sample data, in many datasets with longitudinal cohort data, survival or time-to-event outcomes will be recorded using time-varying conventions. For example, a variable for mortality will take the value of zero until the person-time observation that represents the time interval that the outcome actually happens in. For outcomes such as mortality this will typically be the last observation (since the subject won't be tracked after death). In those cases, it will be important to take the last observation for each subject to count the number of deaths; for this dataset, we've got more flexibility since they use time-invariant recording conventions for outcomes like death.

In order to estimate the distribution of age at death among those participants who died during follow-up we need to create a new age at death variable. First, we don't know the age at death of any study subjects who died after follow-up (which could be the end of the study or when they were lost to follow-up). Therefore, when we calculate, we should filter the dataset to only include subjects who died during follow-up (`filter(death == 1)` a bit later in the code).

Next, we need to use information in the dataset to calculate the age at the time of death for the study subjects who died during follow-up. The `age` variable in `fhs` represents the participant's age at each visit. Typically a death would happen between visits so the last recorded value for `age` would be less than the age at death. We will instead use the `timedth` variable to help us determine the actual age at death. The value of `timedth` is the number of days from beginning of follow-up until death for those with `death = 1`, while it is a fixed value of `timedth = 8766` (the maximum duration of follow-up) for those with `death = 0` (did not die during follow-up).

We can create a new age at death variable for those with `death = 1` using the `age` at baseline and `timedth` values:

```
fhs_first <- fhs_first %>%
  mutate(timedthy = timedth / 365.25, # time-to-death in years
         agedth = age + timedthy)

fhs_first
```

```
## # A tibble: 4,434 x 41
##    randid   sex totchol   age sysbp diabp cursmoke cigpday   bmi diabetes bpmeds
##     <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>    <dbl>   <dbl> <dbl>    <dbl>  <dbl>
##  1   2448     1     195    39  106     70        0       0  27.0        0      0
##  2   6238     2     250    46  121     81        0       0  28.7        0      0
##  3   9428     1     245    48  128.    80        1      20  25.3        0      0
##  4  10552     2     225    61  150     95        1      30  28.6        0      0
##  5  11252     2     285    46  130     84        1      23  23.1        0      0
##  6  11263     2     228    43  180    110        0       0  30.3        0      0
##  7  12629     2     205    63  138     71        0       0  33.1        0      0
##  8  12806     2     313    45  100     71        1      20  21.7        0      0
##  9  14367     1     260    52  142.    89        0       0  26.4        0      0
## 10  16365     1     225    43  162    107        1      30  23.6        0      0
## # ... with 4,424 more rows, and 30 more variables: heartrte <dbl>,
## #   glucose <dbl>, educ <dbl>, prevchd <dbl>, prevap <dbl>, prevmi <dbl>,
## #   prevstrk <dbl>, prevhyp <dbl>, time <dbl>, period <dbl>, hdlc <dbl>,
## #   ldlc <dbl>, death <dbl>, angina <dbl>, hospmi <dbl>, mi_fchd <dbl>,
## #   anychd <dbl>, stroke <dbl>, cvd <dbl>, hyperten <dbl>, timeap <dbl>,
## #   timemi <dbl>, timemifc <dbl>, timechd <dbl>, timestrk <dbl>, timecvd <dbl>,
## #   timedth <dbl>, timehyp <dbl>, timedthy <dbl>, agedth <dbl>
```

We can then get summary statistics on this new variable, filtering down to only the observations where death occurs during follow-up (this means that we're only calculating average age at death among those who died during follow-up—see the note at the end of this section related to that):

```
fhs_first %>%
  filter(death == 1) %>%
  summarize(min_agedth = min(agedth),
            mean_agedth = mean(agedth),
            max_agedth = max(agedth),
            missing_agedth = sum(is.na(agedth)))
```

```
## # A tibble: 1 x 4
##   min_agedth mean_agedth max_agedth missing_agedth
##        <dbl>       <dbl>      <dbl>          <int>
## 1       38.4        69.1       91.1              0
```

The earliest death was at 38 years and the latest at 91, among deaths that occurred during follow-up. On average, subjects who died during follow-up for this study died when they were about 69 years old. There were no missing ages

for study subjects who died during follow-up.

We can also check on these values by groups of interest such as sex:

```
fhs_first %>%
  filter(death == 1) %>%
  group_by(sex) %>%
  summarize(min_agedth = min(agedth),
            mean_agedth = mean(agedth),
            max_agedth = max(agedth))
```

```
## # A tibble: 2 x 4
##     sex min_agedth mean_agedth max_agedth
## * <dbl>      <dbl>       <dbl>      <dbl>
## 1     1       41.6        68.6       91.1
## 2     2       38.4        69.6       90.0
```

Of course, it's important to remember that these are summaries of the age at death *only* for the study subjects who died during follow-up. The mean age at death will be different across all our study subjects, but we don't have the information about age at death for those who died after censoring to include in calculating our summary statistics here. It is likely that they lived to older ages, if the most common reason for censoring is outliving follow-up. However, if they were censored because they dropped out of the cohort, then that might instead be associated with either longer or shorter average lifespans, in which case we don't have a great idea of which direction the average age of death would move if they were included. One thing that you could do is to average the age at either death or loss to follow-up—this would give you a lower bound on the average across the whole population, since you know that those who were censored survived to *at least* the age they were when they were censored.

4. **What is the distribution of BMI among MI cases and non-cases? How about between smokers and non-smokers**

Both BMI and smoking are recorded in a time-variant way; that is, their value can differ between different examinations for the same study subject. For example, you can look at a couple of study subjects:

```
fhs %>%
  filter(randid %in% c(6238, 16365)) %>%
  select(randid, period, bmi, cursmoke)
```

```
## # A tibble: 6 x 4
##   randid period   bmi cursmoke
##    <dbl>  <dbl> <dbl>    <dbl>
## 1   6238      1  28.7        0
## 2   6238      2  29.4        0
## 3   6238      3  28.5        0
## 4  16365      1  23.6        1
```

```
## 5  16365      2  27.5        0
## 6  16365      3  29.1        0
```

For the study subject with ID 6238, BMI changed a little bit across the three examinations, but not much, and the person remained a non-smoker. For the study subject with ID 16365, BMI increased steady across the examinations, and the person stopped smoking sometime after the first examination.

We will need to think about how to handle these variant values while we compare them to the invariant outcome (whether the subject was hospitalized for MI during follow-up). One thing that we could do is to see the association between average BMI for each study subject and whether they had a hospitalized MI event during follow-up:

```
bmi_vs_mi <- fhs %>%
  group_by(randid) %>%
  summarize(bmi = mean(bmi, rm.na = TRUE),
            hospmi = first(hospmi))
bmi_vs_mi
```

```
## # A tibble: 4,434 x 3
##    randid   bmi hospmi
##  *  <dbl> <dbl>  <dbl>
##  1   2448  NA        1
##  2   6238  28.9      0
##  3   9428  25.3      0
##  4  10552  29.4      0
##  5  11252  23.7      0
##  6  11263  30.9      0
##  7  12629  34.9      0
##  8  12806  22.0      0
##  9  14367  25.8      0
## 10  16365  26.7      0
## # ... with 4,424 more rows
```

Now we can compare the distribution of these average BMIs among study subjects who did and did not have a hospitalization for MI during follow-up. A simple was is by creating some summaries:

```
bmi_vs_mi %>%
  group_by(hospmi) %>%
  summarize(perc_25 = quantile(bmi, 0.25, na.rm = TRUE),
            mean = mean(bmi, na.rm = TRUE),
            median = median(bmi, na.rm = TRUE),
            perc_75 = quantile(bmi, 0.75, na.rm = TRUE))
```

```
## # A tibble: 2 x 5
##   hospmi perc_25  mean median perc_75
## *  <dbl>   <dbl> <dbl>  <dbl>   <dbl>
```

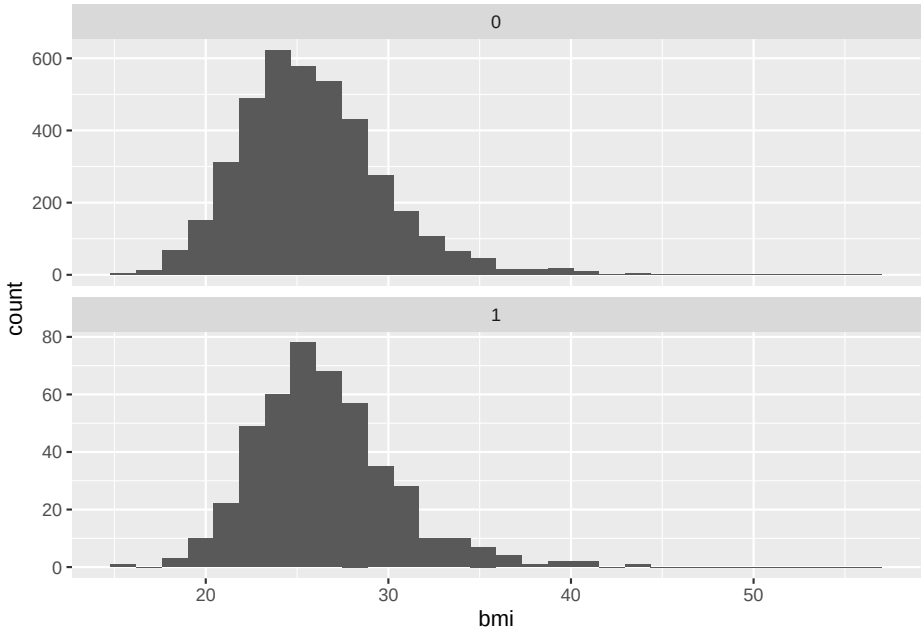
```
## 1      0    23.1  25.8   25.4    28.0
## 2      1    24.0  26.5   26.2    28.5
```

We can also create a plot to help explore this question:

```
bmi_vs_mi %>%
  ggplot(aes(x = bmi)) +
  geom_histogram() +
  facet_wrap(~ hospmi, ncol = 1, scale = "free_y")
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 44 rows containing non-finite values (stat_bin).
```

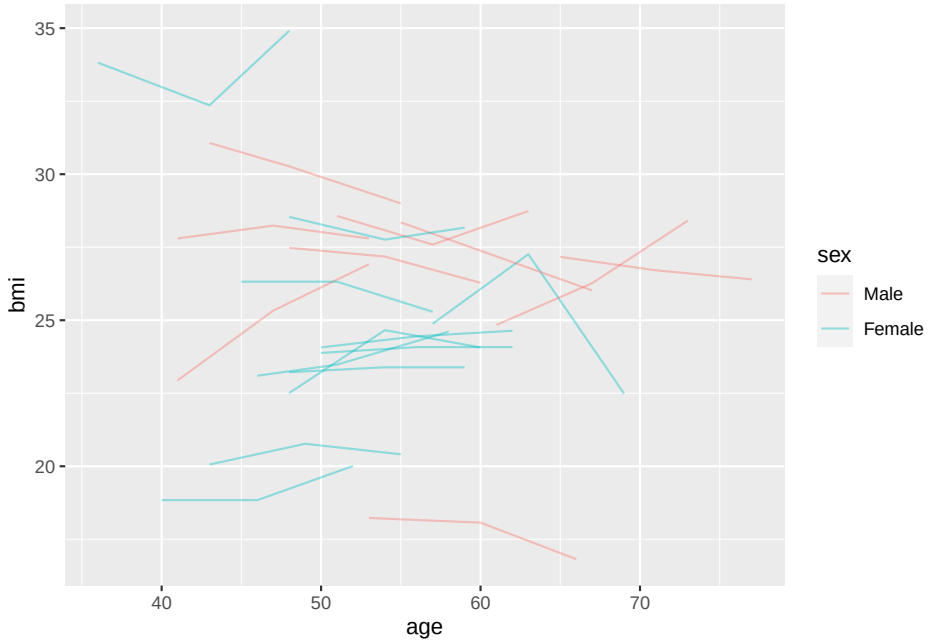


There is not a dramatic difference between the two groups in terms of the distribution of BMI.

We might want to check how stable BMI tends to be within each study subject, and check variation in BMI within subjects (at different timepoints) compared to between subjects, to see if the average might be a reasonable summary of BMI for a study subject. We can check this within just the study subjects with three examinations and without any missing measures of BMI at those examinations. We can check it with a random sample of 20 subjects (since this uses `sample`, the sample you get, and so the plot, will likely be different):

```
sample_check <- fhs %>%
  group_by(randid) %>%
  filter(max(period == 3) & !anyNA(bmi)) %>%
```

```
  pull(randid) %>%
  sample(size = 20)

fhs %>%
  filter(randid %in% sample_check) %>%
  mutate(sex = factor(sex, levels = c(1, 2), labels = c("Male", "Female"))) %>%
  ggplot(aes(x = age, y = bmi, color = sex, group = randid)) +
  geom_line(alpha = 0.4)
```



While there is some variation within study subjects in BMI, there tends to be more variation when comparing one study subject to another. Average BMI across the three examinations is therefore likely a reasonable measurement to use in exploratory analysis as we did in the previous plot.

Another alternative we could have considered is to use the BMI at the first examination as the measure of the BMI risk factor for each study subject; if you'd like, try that out to see if it changes our main conclusions.

We can use a similar approach to compare BMI and smoking. In this case, one way we could summarize smoking for each study subject is to determine if they were a current smoker at any of their examinations. If we do this, we see that there may be a small, but not dramatic, difference in BMI between smokers and non-smokers:

```
bmi_vs_smoke <- fhs %>%
  group_by(randid) %>%
```

```
  summarize(bmi = mean(bmi, rm.na = TRUE),
            smoke = max(cursmoke))

bmi_vs_smoke %>%
  group_by(smoke) %>%
  summarize(perc_25 = quantile(bmi, 0.25, na.rm = TRUE),
            mean = mean(bmi, na.rm = TRUE),
            median = median(bmi, na.rm = TRUE),
            perc_75 = quantile(bmi, 0.75, na.rm = TRUE))
```

```
## # A tibble: 2 x 5
##   smoke perc_25  mean median perc_75
## * <dbl>   <dbl> <dbl>  <dbl>   <dbl>
## 1     0    23.8  26.6   26.1    28.8
## 2     1    22.7  25.3   24.9    27.4
```

Again, we might want to check how "stable" smoking status is among study subjects, to get a better idea of how reasonable it is to use a single summary of smoking for each subject in this exploratory analysis.

```
fhs %>%
  group_by(randid) %>%
  summarize(always_no = max(cursmoke) == 0,
            always_yes = min(cursmoke) == 1,
            changes = !((sum(cursmoke) / n()) %in% c(0, 1))) %>%
  ungroup() %>%
  summarize(always_no = sum(always_no),
            always_yes = sum(always_yes),
            changes = sum(changes))
```

```
## # A tibble: 1 x 3
##   always_no always_yes changes
##       <int>      <int>   <int>
## 1      2100       1585     749
```

Less than 20% of the study subjects changed their smoking status over the course of the examinations. This isn't negligible, but it also means that for the majority of the study subjects, their smoking status was stable across all study examinations.

7.3 Coding a survival analysis

In the context of survival analysis, what is modeled is time to an event (also referred to as survival time or failure time). This is a bit different than the models in the linear or `glm` family that model an outcome that may follow a gaussian (linear regression), binomial (logistic model) or Poisson distribution.

Another difference is that the outcome (time to event) will not be determined in some participants, as they will not have experienced the event of interest during their follow-up. These participants are considered 'censored'. Censoring can occur in three ways:

- the participant does not experience the event of interest before the study end
- the participant is lost to follow-up before experiencing the event of interest
- the participant experiences a difference event that makes the event of interest impossible (for example if the event of interest is acute MI a participant that dies from a different cause is considered censored)

These are all types of right censoring and in simple survival analysis they are considered to be uninformative (typically not related to exposure). If the censoring is related to the exposure and the outcome, then you must adjust for censoring or it could confound the estimates from your model.

Let's assume that we are interested in all-cause mortality as the event of interest, and let's denote T as time to death, where we can assume that $T \geq 0$. We define the survival function as $S(t) = Pr[T > t] = 1 - F(t)$, where the survival function $S(t)$ is the probability that a participant survives past time t ($Pr[T > t] = 1$). $F(t)$ is the Probability Density Function, (sometimes also denoted as the the Cumulative Incidence Function, $R(t)$) or the probability that that an individual will have a survival time less than or equal to t ($[Pr(T \leq t)]$).

Time to event t is bounded by $[0, \infty)$ (i.e., the time could be as low as 0, but no lower, and has no upper bound) and $S(t)$ is non-increasing as t becomes greater. At $t = 0$, $S(t) = 1$ and conversely as t approaches ∞, $S(t) = 0$. A property of the survival and probabilty density function is $S(t) = 1–F(t)$: the survival function and the probability density function (or cumulative incidence function ($R(t)$) sum to 1.

Another useful function is the hazard Function, $h(t)$, which is the instantaneous potential of experiencing an event at time t, conditional on having survived to that time ($h(t) = \frac{Pr[t<T\leq t+\Delta t|T>t]}{\Delta t} = \frac{f(t)}{S(t)}$). The cumulative Hazard Function, $H(t)$ is defined as the integral of the hazard function from time 0 to time t, which equals the area under the curve $h(t)$ between time 0 and time t ($H(t) = \int_0^t h(u)du$). If we know any of $S(t)$, $H(t)$ or $h(t)$, we can derive the rest based on the following relationships:

$h(t) = \frac{\partial log(S(t))}{\partial t}$

$H(t) = -log(S(t))$ and conversely $S(t) = exp(-H(t))$

The `survival` package in R allows us to fit these types of models, including a very popular model in survival analysis, the Cox proportional hazards model. This is the model that was also applied in one of this chapter's required readings, Wong et al. (1989). There are also some simple non-parametric ways to explore survival times, which we'll also explore in the exercise.

Applied exercise: Survival curves and simple survival analysis

1. What does the survival curve for mortality look like with follow-up time as the time scale of interest? How about with age?
2. How do (survival) curves for mortality compare between males and females? How about for MI?
3. What is the Hazard Ratio for smoking and the risk of MI, from a Cox Proportional Hazards model?

1. What does the survival curve for mortality look like with follow-up time as the time scale of interest? How about with age?

A very simple way to estimate survival is the non-parametric Kaplan-Meier estimator.

In R we would estimate Survival $S(t)$ with all-cause mortality representing failure as follows:

```
library(survival)
S1 <- Surv(time = fhs_first$timedth,
           event = fhs_first$death)
```

The `Surv` function will be key as we look at time-to-event data. This function inputs two vectors: one for the `time` parameter that gives the follow-up time (either the time to the event, if the event happens during follow-up, or the time to censoring) and one for the `event` parameter, which gives a status indicator of whether the event happened during follow-up or whether then person was censored before the event happened. For our example data, we can use the column `timedth` to give the time until either death or censoring, and then the `death` column as an indicator of whether death happened before censoring.

The output of the `Surv` function is a special type of object in R with the class "Surv". If you look at the first few values, you can see that this object records both the follow-up time and the status at that follow-up time:

```
class(S1)
```

```
## [1] "Surv"
```

```
head(S1)
```

```
## [1] 8766+ 8766+ 8766+ 2956  8766+ 8766+
```

The numbers assigned to each individual represent their final measured time, with each number with a plus sign indicating that the participant was censored at that time without developing the outcome (haven't failed/died), while those without the plus sign are the times at which participants developed the outcome (failure/death).

We can use the `Surv` function inside the `survfit` function to estimate the Kaplan-Meier curve for a set of data. If we aren't considering any covariates,

we can include `~ 1` in the model equation, to estimate with only an intercept, rather than to explore differences in the curve based on covariates (we'll get to that idea later):

```
fit_time <- survfit(Surv(timedthy, death) ~ 1, data = fhs_first)
```

This creates an object of the `survfit` class, that we can then use in some special plotting functions to look at the curve its estimated. If you print this object, you can see that it gives us some information about the total number of study subjects as well as the total number of events during follow-up (in this case, deaths):

```
class(fit_time)
```

```
## [1] "survfit"
```

```
fit_time
```

```
## Call: survfit(formula = Surv(timedthy, death) ~ 1, data = fhs_first)
##
##       n  events  median 0.95LCL 0.95UCL
##    4434    1550      NA      NA      NA
```

You can also look at the structure of this object with `str`:

```
fit_time %>%
  str()
```

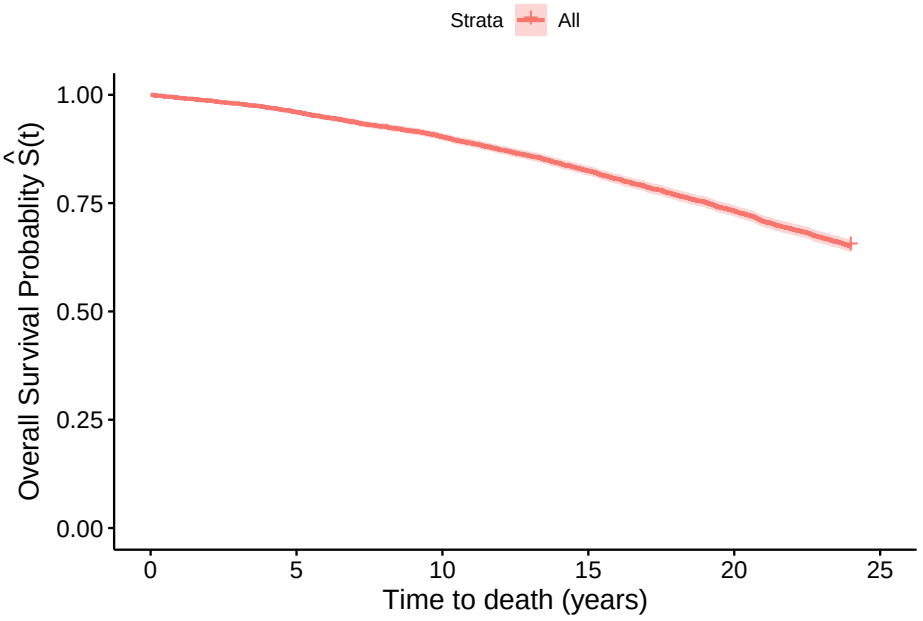
```
## List of 16
##  $ n        : int 4434
##  $ time     : num [1:1419] 0.0712 0.0931 0.1095 0.1232 0.1259 ...
##  $ n.risk   : num [1:1419] 4434 4433 4432 4431 4430 ...
##  $ n.event  : num [1:1419] 1 1 1 1 1 1 1 1 1 1 ...
##  $ n.censor : num [1:1419] 0 0 0 0 0 0 0 0 0 0 ...
##  $ surv     : num [1:1419] 1 1 0.999 0.999 0.999 ...
##  $ std.err  : num [1:1419] 0.000226 0.000319 0.000391 0.000451 0.000505 ...
##  $ cumhaz   : num [1:1419] 0.000226 0.000451 0.000677 0.000902 0.001128 ...
##  $ std.chaz : num [1:1419] 0.000226 0.000319 0.000391 0.000451 0.000505 ...
##  $ type     : chr "right"
##  $ logse    : logi TRUE
##  $ conf.int : num 0.95
##  $ conf.type: chr "log"
##  $ lower    : num [1:1419] 0.999 0.999 0.999 0.998 0.998 ...
##  $ upper    : num [1:1419] 1 1 1 1 1 ...
##  $ call     : language survfit(formula = Surv(timedthy, death) ~ 1, data = fhs_first
##  - attr(*, "class")= chr "survfit"
```

It's listing many different times during the follow-up. For each, it's determined the number of people in the study at risk at that time (not dead or censored) and the number of events at that time point, as well as some other values. These

calculations allow it to estimate the cumulative hazard at each time point during follow-up.

There are a number of ways you can plot this output to get a better idea of patterns in the survival curve estimated from the data. The `survminer` package allows you to do this in a way that interfaces well with `ggplot` (note also how you can use `expression` to include mathematical notation in an axis label):

```
library(survminer) # for plotting survival plots with ggplot2

fit_time %>%
  ggsurvplot(xlab = "Time to death (years)",
             ylab = expression(paste('Overall Survival Probablity ',
                                     hat(S)*"(t)")))
```

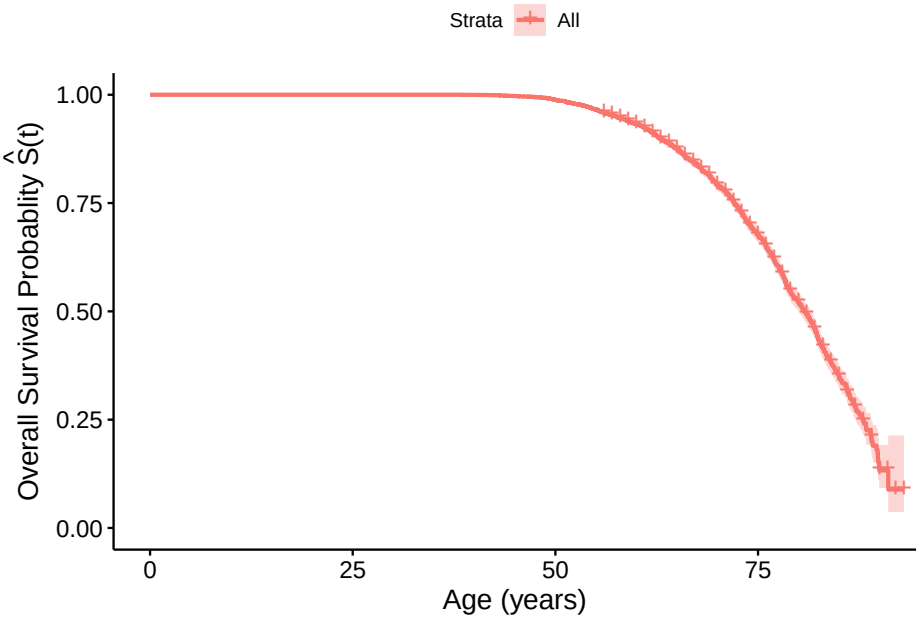


We can see that as follow-up time increases, survival decreases rather monotonically, steadily, and slowly over time. In other words, the number of people who have died increases as the length of follow-up increases, but not very quickly (which makes sense since the study population was largely healthy at the start of the study). Survival $\hat{S}(t)$ drops to about 0.65 at the end of follow-up, or in other words about 35% of participants have died, which is what is expected as we already know that 1,550 of 4,434 participants died during follow-up.

We can repeat this estimation with a different time-scale of interest. Other that follow-up times we may also be interested in survival and failure (mortality) with respect to age. We repeat the same code only changing the first argument in the `Surv` function, substituting time of death with respect to follow-up time

with age at death.

```
fit_age <- survfit(Surv(agedth, death) ~ 1, data = fhs_first)

fit_age %>%
  ggsurvplot(xlab = "Age (years)",
             ylab = expression(paste('Overall Survival Probablity ',
                                     hat(S)*"(t)")))
```



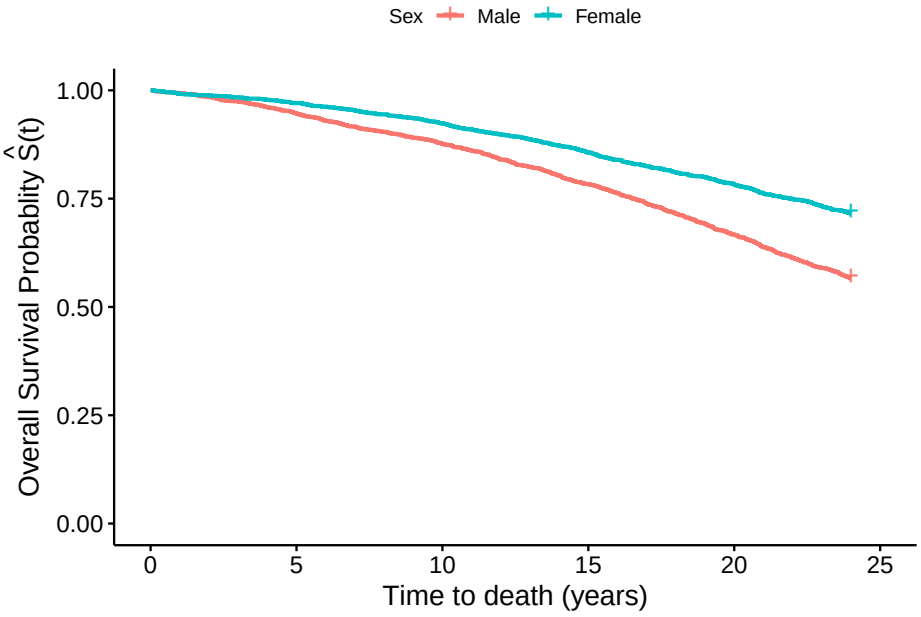
We see that the shape of this survival curve is different, with virtually no one dying until they reach their 40s (part of this is likely because this study focused on subjects in their 30s and older at the baseline examination period), and then a sharper drop in survival as age increases.

2. How do (survival) curves for mortality compare between males and females? How about for MI?

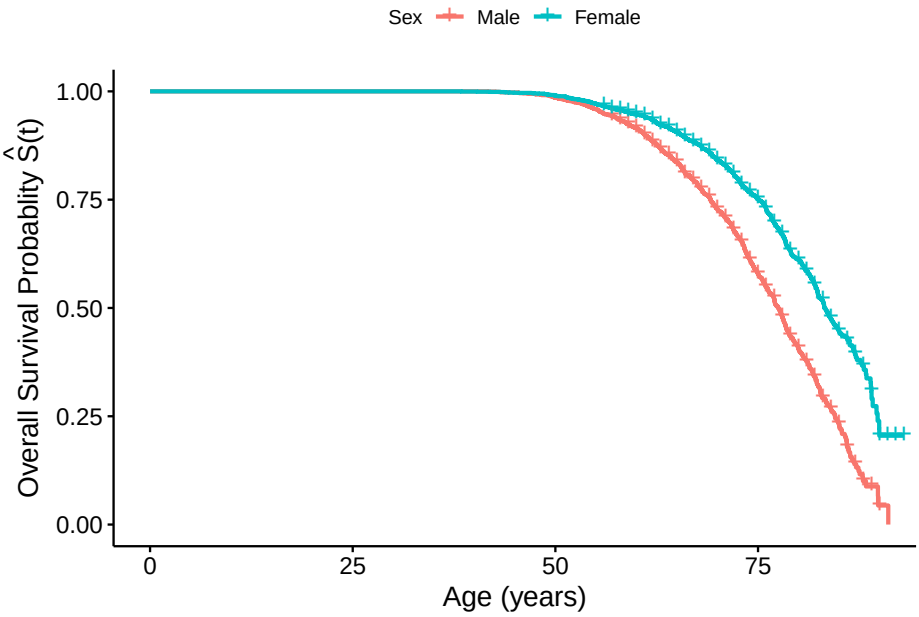
Kaplan-Meir curves like the above are useful in comparing the survival rate between two groups. For example if we wanted to compare the survival rates between males and females we would fit the same model as above with sex as an independent variable. For all-cause mortality, we can run these models both based on follow-up time and on age (in separate models):

```
fit_bysex <- survfit(Surv(timedthy, death) ~ sex, data = fhs_first)
fit_age_bysex <- survfit(Surv(agedth, death) ~ sex, data = fhs_first)

fit_bysex %>%
  ggsurvplot(xlab = "Time to death (years)",
```

```
            ylab = expression(paste('Overall Survival Probablity ',
                                    hat(S) * "(t)")),
            legend.labs = c("Male", "Female"),
  legend.title="Sex")
```



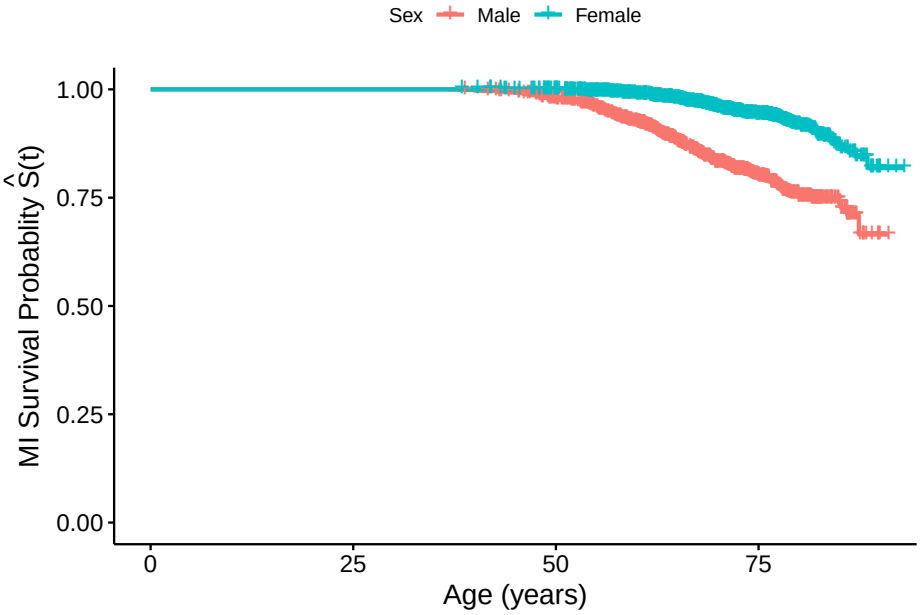

```
fit_age_bysex %>%
  ggsurvplot(xlab = "Age (years)",
            ylab = expression(paste('Overall Survival Probablity ',
                                    hat(S) * "(t)")),
            legend.labs = c("Male", "Female"),
  legend.title = "Sex")
```



You can now see that the survival rate for males drops quicker (at a younger age) than for females, and that it also drops more quickly for males if we're looking across follow-up time.

Similarly we can look at MI as the outcome. If we want to compare across age for the x-axis, then we need to calculate the age at the time of the first hospitalization for MI during follow-up. We can then put that variable in as the `time` parameter in `Surv` and use the status of whether the subject had an MI hospitalization by the end of follow-up for the `event` parameter, then plot as before:

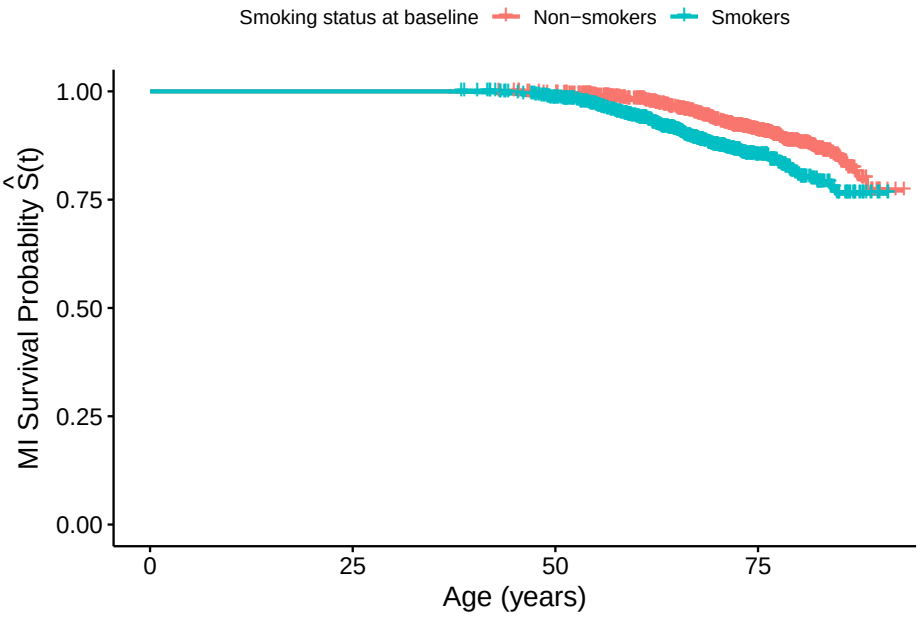
```
fhs_first <- fhs_first %>%
  mutate(timemiy = timemi / 365.25,
         agemi = age + timemiy)
fit_age_MIbysex <- survfit(Surv(agemi, hospmi) ~ sex, data = fhs_first )

fit_age_MIbysex %>%
  ggsurvplot(xlab = "Age (years)",
             ylab = expression(paste('MI Survival Probablity ',
                                     hat(S) * "(t)")),
             legend.labs=c("Male","Female"),
             legend.title="Sex")
```



Once again we see a difference in the survival rates (in this case, "survival" until first hospitalized myocardial infarction, even though the subject might survive the event) with age by sex, which is in line with what we already know from the literature.

We can actually approach survival rates by smoking status in the same manner (in this case, we'll use the subject's smoking status at the baseline examination):

```
fit_age_MIsmoking <- survfit(Surv(agemi, hospmi) ~ cursmoke, data = fhs_first )

fit_age_MIsmoking %>%
ggsurvplot(xlab = "Age (years)",
           ylab=expression(paste('MI Survival Probablity ',
                                 hat(S) * "(t)")),
           legend.labs=c("Non-smokers", "Smokers"),
           legend.title="Smoking status at baseline")
```



Once again we can observe that there is a difference in survival rates for MI, by smoking status at baseline.

The advantages of the Kaplan-Meier estimator for the survival function are its simplicity and the fact that it is a non-paramteric estimator. One limitation of Kaplan-Meier curves is that in this simple form of visualizing a survival rate, we cannot adjust for confounding by other variables, as the survival rates we are plotting here are marginal with respect to everything else. For example, we can compare survival rates among smokers and non-smokers, but we can't really simply plot a sex adjusted survival rate for each, as the baseline rate for males and females will differ. What we can estimate while adjusting for other covariates is a survival time ratio, which is actually estimated using the same model we've been fitting. The `survreg` function in the `survival` package will fit a failure time model, with time to event as the outcome of interest. Unlike the Kaplan-Meier estimator this will require us to make an assumption about the distribution of time-to-event. Usually time-to-event outcomes are assumed to belong to the *exponential*, *Weibull*, *log-normal (log(T) is normally distributed)* or *log-logistic* distributions.

The majority of survival analyses for longitudinal cohort data, however, has been dominated by the Cox proportional hazards model over the past few decades, and this is the type of model we will focus on for the rest of the chapter. The main advantage of the Cox proportional hazards model is that we don't have to make any distributional assumptions about the outcome or residuals. We simply model the instantaneous hazard of the outcome at specific time intervals as a function of covariates of interest, and the assumptions we have to make is that of 'proportional hazards'. This assumptions stipulates that the hazards

across levels of covariates of interest are proportional over time. In other words the ratio of the hazards across levels of covariates should be constant over time.

The Cox proportional hazards model in a simple form has this form:

$log(\lambda(t|X)) = log(\lambda_0(t)) + \beta_1 \times X$

where $\lambda(t)$ represents the hazard at time t, $\lambda_0(t)$ is the baseline hazard at time t, and β_1 is the log hazard for those with $X = 1$ compared to $X = 0$. The baseline hazard $\lambda_0(t)$ is similar to the intercept term in a linear model or glm and is the value of the hazard when all covariates equal 0. However, unlike the intercept term in a linear model or glm, $\lambda_0(t)$ is not estimated by the model. The above model can also be writen as

$\lambda(t|X) = \lambda_0(t) \times e^{\beta_1 \times X}$

e^{β_1} is the hazard ratio comparing those hose with $X = 1$ and $X = 0$

Using the `fhs` data we will fit a simple Cox proportional hazard for the effect of smoking on the hazard for MI.

```
coxph_mod1 <- coxph(Surv(timedth, death) ~ cursmoke, data = fhs_first)
```

If we look the estimated parameters of the model (we can use `broom` to pull out a tidy version of these summaries, just as we did with GLM models), there isn't an intercept term, as noted above, just an estimate for the covariate we included (`cursmoke` for smoking status at the baseline examination):

```
library(broom)
coxph_mod1 %>%
  tidy()
```

```
## # A tibble: 1 x 5
##   term     estimate std.error statistic p.value
##   <chr>       <dbl>     <dbl>     <dbl>   <dbl>
## 1 cursmoke   0.0867    0.0508      1.71  0.0880
```

The parameter for the covariate of interest is equivalent to the log of the hazard ratio comparing current smokers at baseline to non-smokers. We can extract the hazard ratio by exponentiating that parameter.

```
coxph_mod1 %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.09  0.987    1.20
```

We see that there is modest suggestive HR elevating the hazard for mortality, but the confidence interval includes the null (hazard ratio of 1).

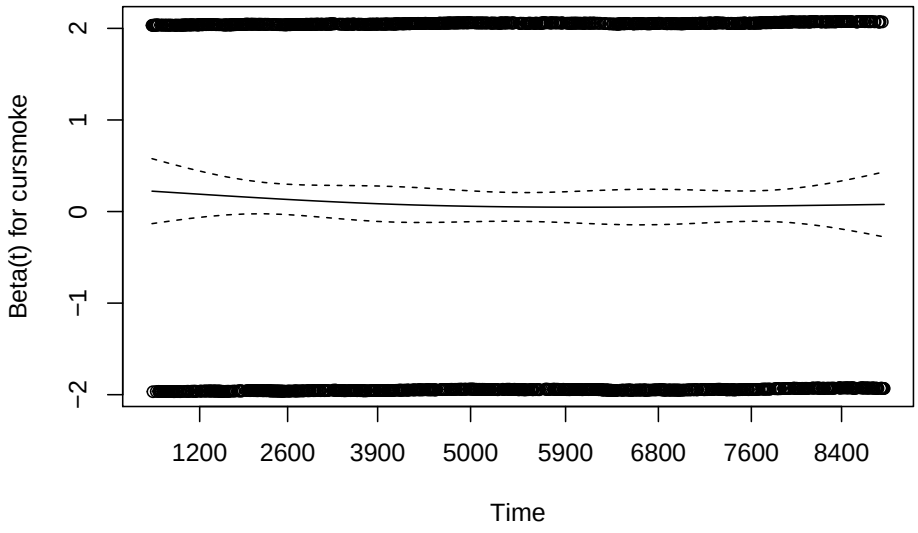
We have said that the main assumption we need to make here is that of proportional hazards. The `survival` package actually allows us to check this with the `cox.zph` function

```
phtest <- cox.zph(coxph_mod1)
phtest
```

```
##          chisq df    p
## cursmoke 0.511  1 0.47
## GLOBAL   0.511  1 0.47
```

The output of this function is the result of a χ^2 test for the proportional hazards assumption. If the p-value here was below 0.05 we would have to reject the null hypothesis that the proportional hazards assumption holds. We can also plot the parameter(s) of interest across time from this output. If the proportional hazards assumption holds (constant HR) then the parameter should resemble a horizontal line with respect to time.

```
plot(phtest)
```



Here we see that the line is basically horizontal. Now let's repeat the model adjusting for some covariates, specifically sex and age. We can include these in a model equation in `coxph` in a very similar way to how we built model equations for GLMs. The only difference is that the "outcome" part of the

formula should be a `Surv` object, rather than a direct (untransformed) measure from the original data:

```
coxph_mod2 <- coxph(Surv(timedth, death) ~ cursmoke + sex + age,
                    data = fhs_first)
coxph_mod2 %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term     estimate std.error statistic   p.value
##   <chr>       <dbl>     <dbl>     <dbl>     <dbl>
## 1 cursmoke   0.338    0.0535       6.32 2.64e- 10
## 2 sex       -0.540    0.0525     -10.3  9.35e- 25
## 3 age        0.0967   0.00327     29.6  3.79e-192
```

We can already see that the parameter for smoking is now quite a bit higher, but let's estimate the HR and 95% CI:

```
coxph_mod2 %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.40   1.26    1.56
```

The estimated Hazard Ratio is now 1.40 and the 95% CI does not include the null (in fact, the lower bound of the 95% CI is 1.26). We can determine that there was some confounding by these variables (sex and age at the baseline examination) leading the estimate from the previous model of the association between smoking and time to death to be biased towards the null. Let's test the proportional hazard assumption for this model:

```
phtest2 <- cox.zph(coxph_mod2)
phtest2
```

```
##          chisq df     p
## cursmoke 0.373  1 0.541
## sex      3.776  1 0.052
## age      3.046  1 0.081
## GLOBAL   7.747  3 0.052
```

We see that the assumption holds, though the results for the test for both sex and age is close to rejecting the assumption (p-values close to 0.05—if they were below 0.05, we'd reject the null hypothesis that the assumption holds).

Now let's see what happens if repeat the above model using age, rather than follow-up time, as the time-scale of interest in the survival function of the model:

```
coxph_modage1 <- coxph(Surv(agedth, death) ~ cursmoke + sex, data = fhs_first)
coxph_modage1 %>%
  tidy()
```

```
## # A tibble: 2 x 5
##   term     estimate std.error statistic  p.value
##   <chr>       <dbl>     <dbl>     <dbl>    <dbl>
## 1 cursmoke    0.373    0.0529      7.05 1.73e-12
## 2 sex        -0.539    0.0526    -10.2  1.46e-24
```

Notice that we did not also include age as an additional parameter in the model, since using it as the time-scale inherently adjusts for it. We can again convert the output to a hazard ratio and 95% CIs:

```
coxph_modage1 %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.45   1.31    1.61
```

We also see that the HR is similar as in the previous model (although slightly higher in this case).

Testing for the proportional hazards assumption in this model:

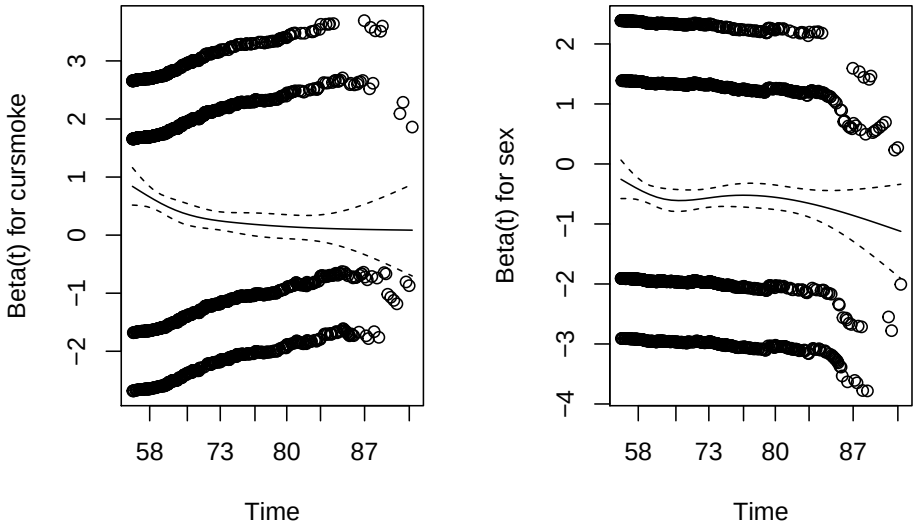
```
phtestage <- cox.zph(coxph_modage1)
phtestage
```

```
##           chisq df       p
## cursmoke 11.637  1 0.00065
## sex       0.657  1 0.41770
## GLOBAL   15.358  2 0.00046
```

Here we see that the assumption fails. If we plot the results of the models we can

also see that we no longer have that horizontal line and in the case of smoking, it deviates from the that line significantly.

```
par(mfrow=c(1,2)) # graphical parameters to plot two plots side by side (1 row, 2 columns)
plot(phtestage)
```



7.4 Handling complexity

You may have noticed that the models we've been fitting above only use the first observation from the dataset (including death and time of death variables). This would be fine if all covariates of interest were time-invariant (e.g. sex: its value at baseline would be representative for the duration of follow-up). However, this is a longitudinal dataset with multiple observations per participant, and as we have seen with BMI and smoking, some measurement values change over time. The time-varying nature of these values can be informative whether it has to do with an exposure, covariate, or outcome of interest, and we will now see how to leverage this extra bit of information in the survival analysis context. We will create a longitudinal dataset appropriate for analysis, and go through steps of dealing with repeated outcome measures, time-varying exposures and covariates, and handling multi-level exposures in this setting.

7.4.1 Recurrent outcome and time varying-exposures

We mentioned earlier how typically in longitudinal studies survival outcomes may be represented in a time-varying fashion. Typically these variables will take the value of 0 until someone becomes a case (i.e., an event like death happens to the person) in which case the value is 1 (and usually this will be the last observation for a given participant). We will go ahead and create these types of variables for death and incident MI in our longitudinal dataset.

Applied exercise: Survival analysis with time-varying values

1. Construct a longitudinal dataset with repeated (time-varying) values for death and MI hospitalization.
2. Fit a Cox model for the effect of smoking on mortality and MI using this dataset. How do the results compare to the results from the models in 7.3?
3. Construct an exposure variable representing the history of smoking (identify if someone is a current, former, or never smoker at each period). Repeat the model from 2, using this exposure instead. What added information does this model give us?

1. Construct a longitudinal dataset with repeated (time-varying) values for death and cardiovascular outcomes.

The original `fhs` dataset already has multiple observations per participant denoted by the `period` variable, with a range of 1–3. Covariate values can change from period to period as we've already seen, but the variables for death and cardiovascular outcomes were not time-varying, simply an indicator for the event occurring at any point during follow-up. We will create time-varying variables for these events, as well as time variables indicating the beginning and end of follow-up time in each period.

```
fhstv <- fhs %>%
  group_by(randid) %>%
  mutate(time_next = lead(time), ### bring forward the time from the next period
         time2 = ifelse(is.na(time_next) == FALSE,
                        time_next - 1,
                        timedth), ### create a variable indicating the last day of foll
         deathtv = ifelse(death == 1 & timedth <= time2, 1, 0),
         timemi2 = ifelse(time2 <= timemi, time2, timemi),
         hospmitv = ifelse(hospmi == 1 & timemi <= timemi2, 1, 0)) %>%
  ungroup()
```

We can look at each piece of this. First, we can check the time-related variables we've added:

```
fhstv %>%
  select(randid, period, time, time_next, time2)
```

```
## # A tibble: 11,627 x 5
##    randid period  time time_next time2
##     <dbl>  <dbl> <dbl>     <dbl> <dbl>
##  1   2448      1     0      4628  4627
##  2   2448      3  4628        NA  8766
##  3   6238      1     0      2156  2155
##  4   6238      2  2156      4344  4343
##  5   6238      3  4344        NA  8766
```

```
##  6   9428      1     0      2199  2198
##  7   9428      2  2199        NA  8766
##  8  10552      1     0      1977  1976
##  9  10552      2  1977        NA  2956
## 10  11252      1     0      2072  2071
## # ... with 11,617 more rows
```

You can see that we've grabbed the time of the start of the next period, then reduced this by one to get the last time in the current period. For example, study subject #2448 has examinations for periods 1 and 3. Since period 3 starts at 4,628 days, we can figure out that the first period's data represents all days up to day 4.627. For the last period, it represents measures from the time of the last examination to the time of death or loss to follow-up, so we can use the column with the time of death / follow-up in the original data to figure out this time.

Next, we can look at the variable we've added related to death. Let's filter to two study subjects who died during follow-up, to see how this element works:

```
fhstv %>%
  filter(randid %in% c(10552, 24721)) %>%
  select(randid, period, time, time2, death, timedth, deathtv)
```

```
## # A tibble: 5 x 7
##   randid period  time time2 death timedth deathtv
##    <dbl>  <dbl> <dbl> <dbl> <dbl>   <dbl>   <dbl>
## 1  10552      1     0  1976     1    2956       0
## 2  10552      2  1977  2956     1    2956       1
## 3  24721      1     0  2267     1    6411       0
## 4  24721      2  2268  4407     1    6411       0
## 5  24721      3  4408  6411     1    6411       1
```

The original `death` column is time-invariant—if the study subject died during follow-up, it will be "1" for all the subject's rows of data. With the new `deathtv` variable, it lines up the time of death with the time of each examination (`period`). For subject #10552, death occurred on day 2,956. This was after the study subject's first period of observation, which went from day 0 to day 1,976, but during the second period, which went from day 1,977 to day 2,956 (in other words, the second period ended with the subject's death). Therefore, `deathtv` is 0 in the first period (a death did not occur during this follow-up period for this subject) but 1 during the second period.

You can check the MI variables to see that they work in the same way:

```
fhstv %>%
  filter(randid %in% c(10552, 24721)) %>%
  select(randid, period, time, time2, hospmi, timemi, timemi2, hospmitv)
```

```
## # A tibble: 5 x 8
```

```
##   randid period  time time2 hospmi timemi timemi2 hospmitv
##    <dbl>  <dbl> <dbl> <dbl>  <dbl>  <dbl>   <dbl>    <dbl>
## 1  10552      1     0  1976      0   2956    1976        0
## 2  10552      2  1977  2956      0   2956    2956        0
## 3  24721      1     0  2267      0   6411    2267        0
## 4  24721      2  2268  4407      0   6411    4407        0
## 5  24721      3  4408  6411      0   6411    6411        0
```

We can check to see if our new variables are in agreement with the totals from above. Since these variables are created to only take the value of one once for each case, then their cumulative sum in the entire longitudinal dataset should equal our totals from above, when we calculated based on the first row of data for each study subject:

```
# Summarize by counting the number of "1"s for the time-variant death variable
fhstv %>%
  summarize(tot_deaths = sum(deathtv))
```

```
## # A tibble: 1 x 1
##   tot_deaths
##        <dbl>
## 1       1550
```

```
# Summarize similarly to early in the chapter, by taking the values from
# the first measurement for each study subject
fhstv %>%
  group_by(randid) %>%
  slice(1) %>%
  ungroup() %>%
  summarize(tot_deaths = sum(death))
```

```
## # A tibble: 1 x 1
##   tot_deaths
##        <dbl>
## 1       1550
```

2. Fit a Cox model for the effect of smoking on mortality and MI hospitalizations using this dataset. How do the results compare to the results from the models in 7.3

Using the dataset with the time-varying outcomes we created above, we will repeat the same model for death (and MI hospitalizations) as a function of smoking and age and sex. However, unlike those models in section 7.3 we are not interested in the hazard occurring between beginning and end of follow-up for each participant, but rather the hazard occurring during each period. In order to address this we will slightly modify the `Surv` function in our model to accommodate the specific interval for each period, by designating two time variables (`time` and `time2`) instead of one. If you look at the results of using `Surv` with two times, you can see that it now creates a range for each observation,

including an indicator of whether the person was still alive at the end of that range ("+" after the second number in the interval) or whether they died at the end time of the range:

```
Surv(fhstv$time, fhstv$time2, fhstv$deathtv) %>%
  head(20)
```

```
##  [1] (   0,4627+] (4628,8766+] (   0,2155+] (2156,4343+] (4344,8766+]
##  [6] (   0,2198+] (2199,8766+] (   0,1976+] (1977,2956]  (   0,2071+]
## [11] (2072,4284+] (4285,8766+] (   0,2177+] (2178,4350+] (4351,8766+]
## [16] (   0,2211+] (2212,8766+] (   0,2169+] (2170,4288+] (4289,8766+]
```

You can use this in a Cox regression model as before and extract estimates (log hazard ratio) for each coefficient:

```
coxph_modtv1 <- coxph(Surv(time, time2, deathtv) ~ cursmoke + age + sex,
                      data = fhstv)
coxph_modtv1 %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term     estimate std.error statistic   p.value
##   <chr>       <dbl>     <dbl>     <dbl>     <dbl>
## 1 cursmoke   0.378    0.0545       6.94 3.80e- 12
## 2 age        0.0871   0.00319     27.3  8.13e-164
## 3 sex       -0.546    0.0518     -10.5  5.42e- 26
```

And if we look specifically at the HR for smoking:

```
coxph_modtv1 %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.46   1.31    1.62
```

We see that our estimates are similar to the ones from the model looking only at baseline values and death, so it doesn't look like we've gained much in terms of information.

Now let's repeat this process with first MI hospitalization as the outcome:

```
coxph_modhospmitv1 <- coxph(Surv(time, timemi2, hospmitv) ~
                              cursmoke + age + sex, data = fhstv)
```

```
## Warning in Surv(time, timemi2, hospmitv): Stop time must be > start time, NA
## created
```

```
coxph_modhospmitv1 %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term     estimate std.error statistic  p.value
##   <chr>       <dbl>     <dbl>     <dbl>    <dbl>
## 1 cursmoke   0.482    0.107        4.52 6.08e- 6
## 2 age        0.0455   0.00610      7.45 9.50e-14
## 3 sex       -1.11     0.110      -10.1  4.77e-24
```

There is a warning here indicating that not all out time variables in each observation are in agreement with the rule `time2 > time1`. If that is the case, the Cox model cannot assess a hazard for a time interval with length $<= 0$. In our dataset we are including periods after a participant has experienced an MI hospitalization. The `Surv` function will create missing values for the hazard for these observations, so the estimation of model parameters is not affected, however, we should limit the dataset to the person-time actually at risk for incident MI hospitalization. Luckily the dataset includes a time-varying `prevmi` variable, indicating a prevalent MI for each period, which we can use to subset to the person-time that is still at risk for an incident MI hospitalization.

```
fhstv_incMI <- fhstv %>%
  filter(prevmi == 0)

coxph_modhospmitv2 <- coxph(Surv(time, timemi2, hospmitv) ~
                              cursmoke + age + sex, data = fhstv_incMI)
```

```
## Warning in Surv(time, timemi2, hospmitv): Stop time must be > start time, NA
## created
```

```
coxph_modhospmitv2 %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term     estimate std.error statistic  p.value
##   <chr>       <dbl>     <dbl>     <dbl>    <dbl>
## 1 cursmoke   0.487    0.107        4.57 4.88e- 6
## 2 age        0.0465   0.00611      7.61 2.69e-14
## 3 sex       -1.12     0.110      -10.2  3.00e-24
```

We see that the warning is still there. If we look into our data, we can check why this is:

```
fhstv_incMI %>%
  filter(timemi2 <= time) %>%
  pull(randid) # Identify study subjects where this is an issue
```

```
## [1] 1338446
```

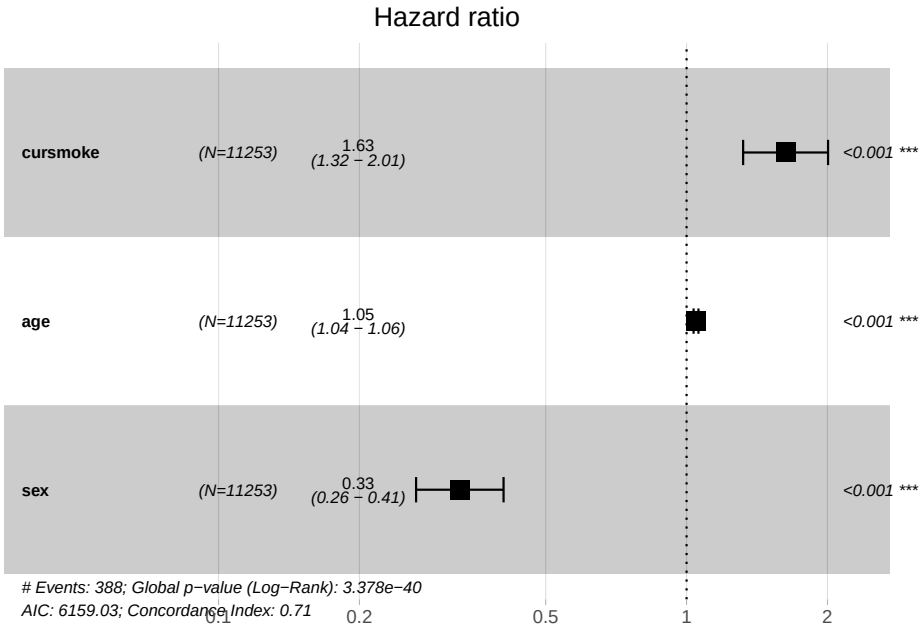
```
# Check out the data for this study subject
fhstv_incMI %>%
  filter(randid == '1338446') %>%
  select (randid, period, time, timemi2, timemi, hospmi, hospmitv)
```

```
## # A tibble: 2 x 7
##    randid period  time timemi2 timemi hospmi hospmitv
##     <dbl>  <dbl> <dbl>   <dbl>  <dbl>  <dbl>    <dbl>
## 1 1338446      1     0    1901   1902      0        0
## 2 1338446      2  1902    1902   1902      0        0
```

This is probably a single participant that was considered lost to follow-up, right after their second period visit, and was assigned the same time as the beginning of that period for all time-to-event values (without actually experiencing the outcomes). In other words they were censored at this exact time. Again, the model doesn't include this observation in the estimation as it creates a missing value for the hazard, so we don't have to worry about our results, but it's good to investigate issues like this. We can visualize the model results with a forest plot, which we can generate using `ggforest` from the `survminer` package we installed above. We also extract the HR and 95% CI for the variable of interest.

```
ggforest(coxph_modhospmitv2)
```

```
## Warning in .get_data(model, data = data): The `data` argument is not provided.
## Data will be extracted from model fit.
```




```
coxph_modhospmitv2 %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.63   1.32    2.01
```

We see that the hazard ratio comparing current smokers to current non-smokers is 1.63 with 95% CI: (1.32 - 2.01), indicating an elevated risk for MI hospitalization associated with current smoking.

However, the key in that interpretation is the word 'current'. In this model for each observation (conditional on sex and age) participants with smoking value of zero are treated equally regardless of whether they are never smokers or former smokers (the latter would have changed from `cursmoke=1` to `cursmoke=0` sometime during follow-up). Based on what we know about smoking history and health, we are therefore probably not making use of all the information we have with this approach.

3. Construct an exposure variable representing the history of smoking (identify if someone is a current, former and never smoker at each period). Repeat the model from 2, using this exposure instead. What added information does this model give us?

We can construct a new time-varying exposure variable that allows us to distinguish between those that are never smokers and those who are currently non-smoking, but used to at some point in the past. We've already identified those people that have had a change in current smoking status during follow-up above, which in theory would include people becoming smokers after starting as non-smokers, but also those who quit smoking and become former smokers.

We can add a column named `smoking` that takes three possible values: "never", "current", and "former". First, let's create a variable called `previous_smoking` that marks whether the subject was ever marked as a current smoking at previous examinations. For this we can use the `cummax` function after grouping by the subject ID—this function will take the maximum value up to (and including) that observation. Since smoking is marked with 1 and not smoking with 0 for `cursmoke`, this new variable marks if there have been *any* cases where `cursmoke` has been 1 up to the observation time:

```
fhstv <- fhstv %>%
  group_by(randid) %>%
  mutate(previous_smoking = cummax(cursmoke)) %>%
  ungroup()

fhstv %>%
  filter(randid %in% c(16365, 67905, 68397)) %>% # Show for some examples where smoking status ch
  select(randid, period, cursmoke, previous_smoking)
```

```
## # A tibble: 9 x 4
##   randid period cursmoke previous_smoking
##    <dbl>  <dbl>    <dbl>            <dbl>
## 1  16365      1        1                1
## 2  16365      2        0                1
## 3  16365      3        0                1
## 4  67905      1        1                1
## 5  67905      2        1                1
## 6  67905      3        0                1
## 7  68397      1        0                0
## 8  68397      2        1                1
## 9  68397      3        1                1
```

This `previous_smoking` variable will help make it easier to create a `smoking` variable of never, current, or former smoker:

```
library(forcats)
```

```
fhstv <- fhstv %>%
  group_by(randid) %>%
  mutate(smoking = case_when(
    cursmoke == 1 ~ "current",
    cursmoke == 0 & previous_smoking == 1 ~ "former",
    cursmoke == 0 & previous_smoking == 0 ~ "never",
    TRUE ~ NA_character_ # This is a catch-all, in case there are any cases missed by
                        ),
    smoking = as_factor(smoking),  # Change to a factor
    smoking = fct_relevel(smoking, "never", "former", "current")) # Get factor levels

fhstv %>%
  filter(randid %in% c(16365, 67905, 68397)) %>% # Show for some examples where smokin
  select(randid, period, cursmoke, previous_smoking, smoking)
```

```
## # A tibble: 9 x 5
## # Groups:   randid [3]
##   randid period cursmoke previous_smoking smoking
##    <dbl>  <dbl>    <dbl>            <dbl> <fct>
## 1  16365      1        1                1 current
## 2  16365      2        0                1 former
## 3  16365      3        0                1 former
## 4  67905      1        1                1 current
## 5  67905      2        1                1 current
## 6  67905      3        0                1 former
## 7  68397      1        0                0 never
## 8  68397      2        1                1 current
## 9  68397      3        1                1 current
```

Let's repeat the Cox model for all-cause mortality from above using this exposure:

```
coxph_modtv2 <- coxph(Surv(time, time2, deathtv) ~ smoking + age + sex,
                      data = fhstv)
coxph_modtv2 %>%
  tidy()
```

```
## # A tibble: 4 x 5
##   term           estimate std.error statistic   p.value
##   <chr>             <dbl>     <dbl>     <dbl>     <dbl>
## 1 smokingformer    0.144    0.0836       1.72 8.61e-  2
## 2 smokingcurrent   0.410    0.0577       7.10 1.28e- 12
## 3 age              0.0875   0.00320     27.3  3.28e-164
## 4 sex             -0.531    0.0525     -10.1  4.92e- 24
```

```
coxph_modtv2 %>%
  tidy() %>%
```

```
  filter(str_detect(term, "smoking")) %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 2 x 4
##   term              hr low_hr high_hr
##   <chr>          <dbl>  <dbl>   <dbl>
## 1 smokingformer   1.15  0.980    1.36
## 2 smokingcurrent  1.51  1.35     1.69
```

We see that the HR for current smokers vs non-smokers remains elevated, but the HR for former smokers is much smaller and the CI includes the null. It seems that the risk of mortality for former smokers compared to non-smokers is quite a bit lower than it is for current smokers.

If we repeat for MI hospitalization (we should recreate our incident MI dataset to include the new smoking variable):

```
fhstv_incMI <- fhstv %>%
  filter(prevmi == 0)

coxph_modhospmitv3 <- coxph(Surv(time, timemi2, hospmitv) ~
                              smoking + age + sex,
                            data = fhstv_incMI)
coxph_modhospmitv3 %>%
  tidy()
```

```
## # A tibble: 4 x 5
##   term            estimate std.error statistic  p.value
##   <chr>              <dbl>     <dbl>     <dbl>    <dbl>
## 1 smokingformer   -0.0588   0.186      -0.316 7.52e- 1
## 2 smokingcurrent   0.475    0.113       4.19  2.82e- 5
## 3 age              0.0464   0.00612     7.59  3.25e-14
## 4 sex             -1.12     0.111     -10.1   5.51e-24
```

```
coxph_modhospmitv3 %>%
  tidy()%>%
  filter(str_detect(term, "smoking")) %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
```

```
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 2 x 4
##   term              hr low_hr high_hr
##   <chr>          <dbl>  <dbl>   <dbl>
## 1 smokingformer  0.943  0.655    1.36
## 2 smokingcurrent 1.61   1.29     2.01
```

We can now see that the parameter for current smokers compared to never smokers is still elevated, with a HR = 1.60, but the hazard associated with a comparison of former smokers compared to never smokers is actually below the null, but the CI is very wide and includes the null.

7.4.2 Multi-level exposure

We've seen that changing smoking from a binary 'yes/no' variable to one with three categories (representing some history in the exposure) has added more information in our assessment for the potential effect of smoking on risk of MI. We actually have even more information on smoking, and specifically smoking intensity, through the `cigpday` variable. We will incorporate this in our model, as well as look at some other continuous exposures with respect to risk of MI.

Applied exercise: Survival analysis with time-varying values

1. Explore the role of smoking intensity on the hazard for MI hospitalizations, using the `cigpday` variable. What can we infer about the exposure-response?
2. Explore the relationship between some of the other predictors in the dataset (specifically `bmi` and `sysbp`) and the hazard for MI hospitalization. Investigate whether the exposure response for these variable deviates from (log-)linearity using spline functions.

1. Explore the role of smoking intensity on the hazard for MI hospitalizations, using the `cigpday` variable. What can we infer about the exposure-response?

Let's repeat our latest model for smoking and MI hospitalizations, including the `cigpday` variable

```
coxph_modhospmitv4 <- coxph(Surv(time, timemi2, hospmitv) ~
                              smoking + cigpday + age + sex,
                            data = fhstv_incMI)
coxph_modhospmitv4 %>%
  tidy()
```

```
## # A tibble: 5 x 5
##   term           estimate std.error statistic  p.value
##   <chr>             <dbl>     <dbl>     <dbl>    <dbl>
## 1 smokingformer  -0.0487    0.186      -0.261 7.94e- 1
```

```
## 2 smokingcurrent  0.256     0.166       1.54  1.23e- 1
## 3 cigpday         0.00992   0.00579     1.71  8.64e- 2
## 4 age             0.0461    0.00622     7.42  1.21e-13
## 5 sex            -1.10      0.114      -9.65  4.68e-22
```

```
coxph_modhospmitv4 %>%
  tidy()%>%
  filter(term == 'cigpday') %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term       hr low_hr high_hr
##   <chr>   <dbl>  <dbl>   <dbl>
## 1 cigpday  1.01  0.999    1.02
```

We now see that the parameter coefficient for current smoking is attenuated, while the continuous `cigpday` variable has a positive parameter coefficient (HR=1.009 with each cigarette per day increase in smoking intensity), but the CI includes the null).

We can actually combine these effects, so that for the current smokers we only get an effect for continuous cigarettes per day, by transforming our smoking variable into dummy variables for each level:

```
fhstv_incMI<-fhstv_incMI %>%
  mutate(smokingcurrent = ifelse(smoking == 'current', 1, 0),
         smokingformer = ifelse(smoking == 'former', 1, 0),
         smokingnever = ifelse(smoking == 'never', 1, 0))

coxph_modhospmitv5 <- coxph(Surv(time, timemi2, hospmitv) ~
                              smokingformer + smokingcurrent:cigpday + age + sex,
                            data = fhstv_incMI) ###leaving the dummy variable for 'never' out mak
coxph_modhospmitv5 %>%
  tidy()
```

```
## # A tibble: 4 x 5
##   term                    estimate std.error statistic  p.value
##   <chr>                      <dbl>     <dbl>     <dbl>    <dbl>
## 1 smokingformer           -0.106    0.182      -0.585 5.59e- 1
## 2 age                      0.0453   0.00620     7.31  2.64e-13
## 3 sex                     -1.10     0.114      -9.66  4.28e-22
## 4 smokingcurrent:cigpday   0.0163   0.00386     4.23  2.33e- 5
```

```
coxph_modhospmitv5 %>%
  tidy() %>%
  filter(term == 'smokingcurrent:cigpday') %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term                      hr low_hr high_hr
##   <chr>                  <dbl>  <dbl>   <dbl>
## 1 smokingcurrent:cigpday  1.02   1.01    1.02
```

The HR for each additional cigarette per day compared to never smokers is now 1.016 (95% CI: 1.008–1.023). If we want to estimate the effect for a smoker who smokes a pack of cigarettes a day then:

```
coxph_modhospmitv5 %>%
  tidy() %>%
  filter(term == 'smokingcurrent:cigpday') %>%
  mutate(hr = exp(20 * estimate),
         low_ci = 20 * (estimate - 1.96 * std.error),
         high_ci = 20 * (estimate + 1.96 * std.error),
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term                      hr low_hr high_hr
##   <chr>                  <dbl>  <dbl>   <dbl>
## 1 smokingcurrent:cigpday  1.39   1.19    1.61
```

The model we just fitted, however, assumes a log-linear exposure-response between smoking intensity and hazard for MI hospitalization. We can explore different categories of smoking intensity by creating a categorical variable, representing different smoking categories. The `case_when` function is helpful here to give a variable different values depending on when different logical conditions are met:

```
fhstv_incMI <- fhstv_incMI %>%
  mutate(smokingint = case_when(
    cigpday == 0 ~ "None",
    0 < cigpday & cigpday <= 5 ~ "Light",
    5 < cigpday & cigpday <= 10 ~ "Moderate",
    10 < cigpday & cigpday <= 20 ~ "Heavy",
```

```
    20 < cigpday ~ "Very Heavy",
    TRUE ~ NA_character_
                                ),
    smokingint = as_factor(smokingint),
    smokingint = fct_relevel(smokingint,
                             "None", "Light", "Moderate", "Heavy", "Very Heavy"))
```

You can see that this added variable gives us the desired ordered categories for smoking:

```
fhstv_incMI %>%
  select(randid, cigpday, smokingint)
```

```
## # A tibble: 11,253 x 3
## # Groups:   randid [4,348]
##    randid cigpday smokingint
##     <dbl>   <dbl> <fct>
##  1   2448       0 None
##  2   2448       0 None
##  3   6238       0 None
##  4   6238       0 None
##  5   6238       0 None
##  6   9428      20 Heavy
##  7   9428      30 Very Heavy
##  8  10552      30 Very Heavy
##  9  10552      20 Heavy
## 10  11252      23 Very Heavy
## # ... with 11,243 more rows
```

Now you can use this variable within the regression model:

```
coxph_modhospmitv6 <- coxph(Surv(time, timemi2, hospmitv) ~
                              smokingformer + smokingcurrent:smokingint + age + sex,
                            data = fhstv_incMI)
coxph_modhospmitv6 %>%
  tidy()
```

```
## # A tibble: 8 x 5
##   term                                estimate std.error statistic   p.value
##   <chr>                                  <dbl>     <dbl>     <dbl>     <dbl>
## 1 smokingformer                        -0.0463   0.186      -0.249  8.04e- 1
## 2 age                                   0.0471   0.00624     7.56   4.05e-14
## 3 sex                                  -1.08     0.113      -9.55   1.36e-21
## 4 smokingcurrent:smokingintNone        NA        0          NA     NA
## 5 smokingcurrent:smokingintLight       -0.257    0.313      -0.820  4.12e- 1
## 6 smokingcurrent:smokingintModerate     0.0545   0.244       0.223  8.23e- 1
## 7 smokingcurrent:smokingintHeavy        0.654    0.134       4.89   1.00e- 6
```

```
## 8 smokingcurrent:smokingintVery Heavy   0.559    0.159       3.51   4.50e- 4
```

We see that the category for "None" did not return a parameter, which should be expected as we don't expect any current smokers to be smoking zero cigarettes. We can extract the rest of the parameters and see what the HR is for each category compared to non-smokers. You can use a regular expression with `str_detect` to pull out the four parameter estimates we're interested in, so you can calculate the confidence intervals for all four at once:

```
coxph_modhospmitv6 %>%
  tidy() %>%
  filter(str_detect(term, "smokingcurrent:smokingint[HLMV]")) %>%
  mutate(hr = exp(estimate),
         low_ci = (estimate - 1.96 * std.error),
         high_ci = (estimate + 1.96 * std.error),
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 4 x 4
##   term                                  hr low_hr high_hr
##   <chr>                              <dbl>  <dbl>   <dbl>
## 1 smokingcurrent:smokingintLight     0.774  0.419    1.43
## 2 smokingcurrent:smokingintModerate  1.06   0.654    1.70
## 3 smokingcurrent:smokingintHeavy     1.92   1.48     2.50
## 4 smokingcurrent:smokingintVery Heavy 1.75  1.28     2.39
```

We see that only heavy and very heavy smokers have elevated HRs compared to non-smokers, with the estimates for light and moderate yielding CIs including the null.

2. Explore the relationship between some of the other predictors in the dataset (specifically `bmi` and `sysbp`) and the hazard for MI hospitalization. Investigate whether the exposure response for these variable deviates from (log-)linearity using spline functions.

We will now turn our attention to some other predictors for cardiovascular disease, namely BMI (`bmi` in our dataset) and systolic blood pressure (`sysbp` in out dataset). Below we fit a Cox model for BMI, adjusting for age and sex. BMI is a considered a fairly flawed proxy for obesity and/or healthy metabolic profiles, however unlike more accurate or informative exposure measures it is very easy to assess and can still be quite informative.

```
coxph_modhospmitvBMI <- coxph(Surv(time, timemi2, hospmitv) ~ bmi + age + sex,
                              data = fhstv_incMI)
coxph_modhospmitvBMI %>%
  tidy()
```

```
## # A tibble: 3 x 5
```

```
##   term  estimate std.error statistic  p.value
##   <chr>    <dbl>     <dbl>     <dbl>    <dbl>
## 1 bmi     0.0483   0.0122       3.96 7.55e- 5
## 2 age     0.0400   0.00596      6.71 1.89e-11
## 3 sex    -1.17     0.109      -10.7  8.84e-27
```

```
coxph_modhospmitvBMI %>%
  tidy()%>%
  filter(term == 'bmi') %>%
  mutate(hr = exp(estimate),
         low_ci = (estimate - 1.96 * std.error),
         high_ci = (estimate + 1.96 * std.error),
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term     hr low_hr high_hr
##   <chr> <dbl>  <dbl>   <dbl>
## 1 bmi    1.05   1.02    1.07
```

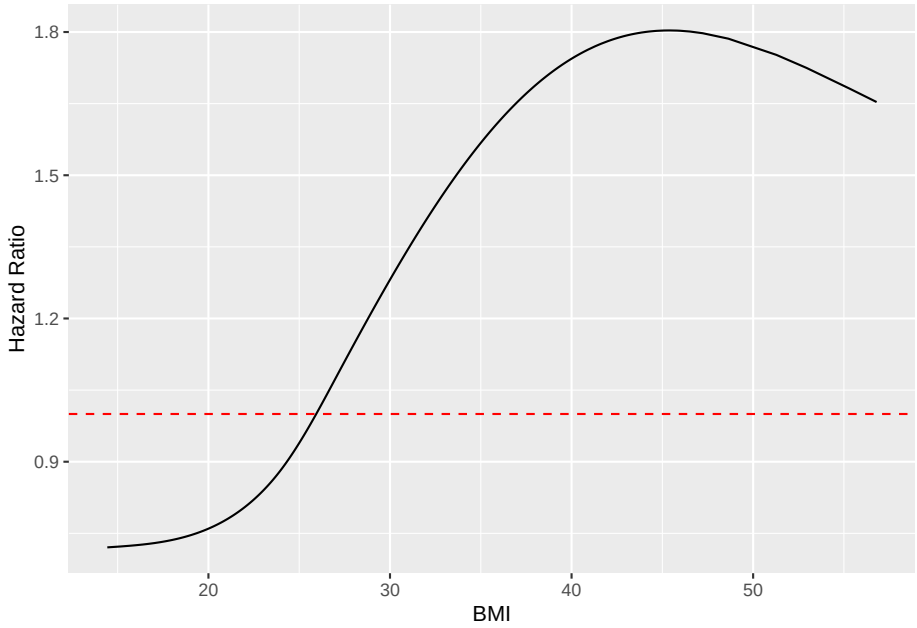
We see that there is an elevated HR for each unit increase in BMI (HR=1.05, 95% CI: 1.02–1.07). In order to explore whether the exposure-response relationship deviates from (log-)linearity, we will repeat the same model using a spline term for `bmi`.

```
library(splines)
coxph_modhospmitvBMIsp <- coxph(Surv(time, timemi2, hospmitv) ~
                                  ns(bmi, df = 3) + age + sex,
                                data = fhstv_incMI)
coxph_modhospmitvBMIsp %>%
  tidy()
```

```
## # A tibble: 5 x 5
##   term              estimate std.error statistic  p.value
##   <chr>                <dbl>     <dbl>     <dbl>    <dbl>
## 1 ns(bmi, df = 3)1    1.06     0.378       2.80  5.12e- 3
## 2 ns(bmi, df = 3)2    0.966    1.52        0.638 5.24e- 1
## 3 ns(bmi, df = 3)3    0.818    1.35        0.605 5.45e- 1
## 4 age                 0.0400   0.00596     6.71  1.94e-11
## 5 sex                -1.16     0.111     -10.4   2.10e-25
```

We can better summarize the relationship between BMI and the outcome from the above model by plotting it similar to our approach in the time-series example. We could use similar code as in 4.2, however the `termplot` functions simplifies some of the steps for us:

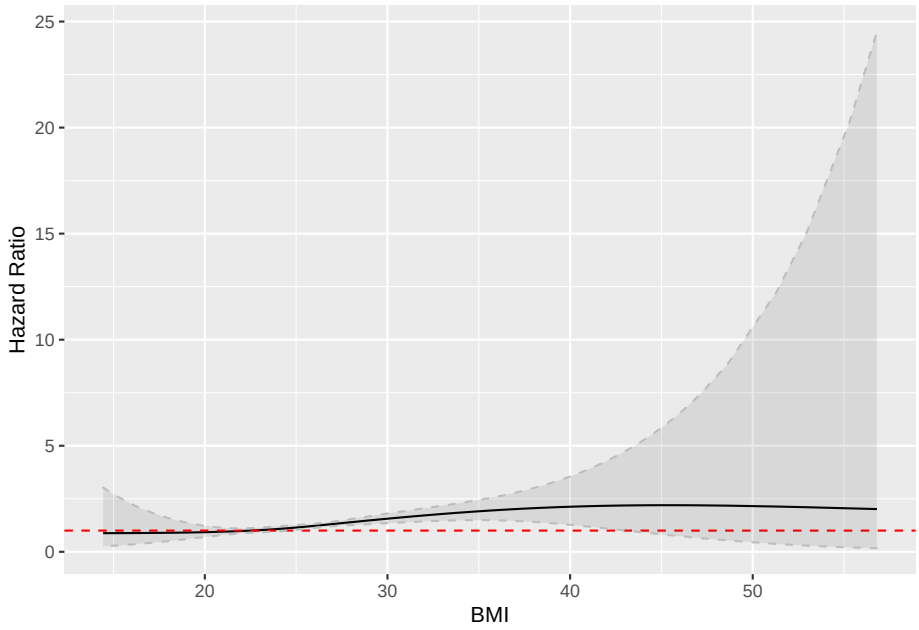
```
library(purrr)
predbmi <- termplot(coxph_modhospmitvBMIsp, se = TRUE, plot = FALSE) %>%
  pluck("bmi")
predbmi %>%
  ggplot(aes(x = x, y = exp(y))) +
  geom_line() +
  labs(x = "BMI",
       y = "Hazard Ratio") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)
```



We see that the relationship is mostly linear between values of about 25 and 40, but deviates from linearity at the lower and higher ends of the range in BMI. We would like to re-center this so that we are comparing to a BMI value that is of **lowest** risk (i.e., the plot crosses 1.0 at that BMI). The 'normal' range of BMI is 20–25, so let's center out graph around 22.5. We will also add confidence bands to our plot.

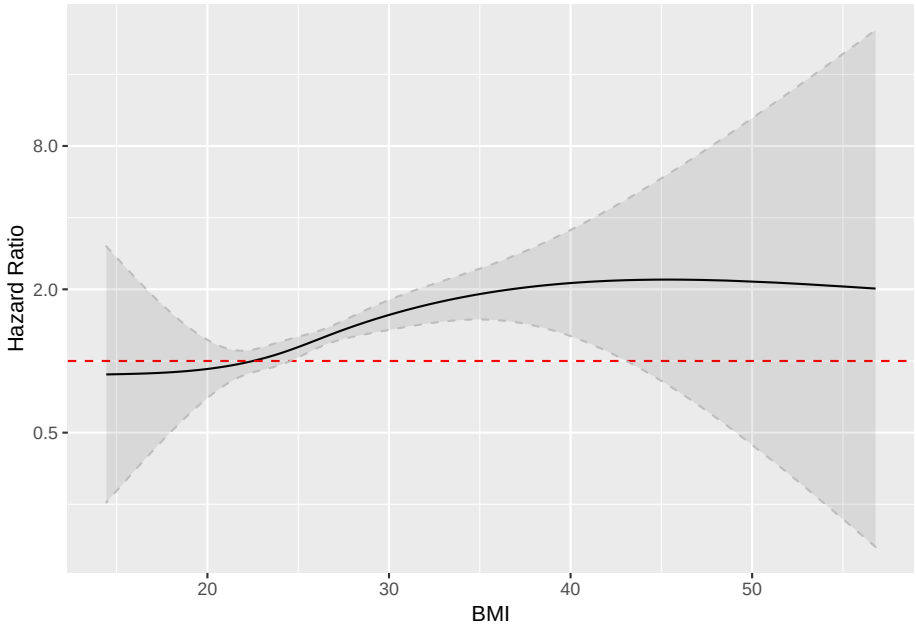
```
predbmi <- predbmi %>%
  mutate(center = y[x == 22.5],
         ylow = y - 1.96 * se,
         yhigh = y + 1.96 * se)

predbmi %>%
  ggplot(aes(x = x, y = exp(y - center))) +
  geom_line() +
  labs(x = "BMI",
```

```
       y = "Hazard Ratio") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)+
  geom_ribbon(aes(ymin = exp(ylow - center), ymax = exp(yhigh - center)),
              alpha = 0.1,
              linetype = "dashed",
              color = "grey")
```



We see that the width of the confidence interval is very large and makes reading the plot quite difficult. We can partially remedy this by log-transforming the y-axis.

```
predbmi %>%
  ggplot(aes(x = x, y = exp(y - center))) +
  geom_line() +
  labs(x = "BMI",
       y = "Hazard Ratio") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)+
  geom_ribbon(aes(ymin = exp(ylow - center), ymax = exp(yhigh - center)),
              alpha = 0.1,
              linetype = "dashed",
              color = "grey") +
  scale_y_continuous(trans = 'log2')
```



We now see that the HR is elevated as BMI increases until about BMI=40 with the effect estimate plateauing at higher values. The CI bands indicate a significant effect between BMI 25 and 42–43. Part of the reason for very wide confidence intervals at the far ranges of BMI is likely because data is sparse for those values, so another alternative you might explore would be to trim the data to focus on the range of BMIs that are more common among the study population.

Next, let's repeat the model, but now add a term for `sysbp`.

```
coxph_modhospmitvBMIBP <- coxph(Surv(time, timemi2, hospmitv) ~
                                  ns(bmi, df = 3) + sysbp + age + sex,
                                data = fhstv_incMI)
coxph_modhospmitvBMIBP %>%
  tidy()
```

```
## # A tibble: 6 x 5
##   term             estimate std.error statistic  p.value
##   <chr>               <dbl>     <dbl>     <dbl>    <dbl>
## 1 ns(bmi, df = 3)1   0.661    0.381       1.73  8.30e- 2
## 2 ns(bmi, df = 3)2  -0.360    1.53       -0.236 8.14e- 1
## 3 ns(bmi, df = 3)3  -0.694    1.38       -0.502 6.16e- 1
## 4 sysbp              0.0187   0.00216     8.67  4.23e-18
## 5 age                0.0246   0.00635     3.87  1.09e- 4
## 6 sex               -1.24     0.113     -11.1   2.05e-28
```

```
coxph_modhospmitvBMIBP %>%
    tidy()%>%
```

```
  filter(term == 'sysbp') %>%
  mutate(hr = exp(estimate),
         low_ci = (estimate - 1.96 * std.error),
         high_ci = (estimate + 1.96 * std.error),
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term     hr low_hr high_hr
##   <chr> <dbl>  <dbl>   <dbl>
## 1 sysbp  1.02   1.01    1.02
```

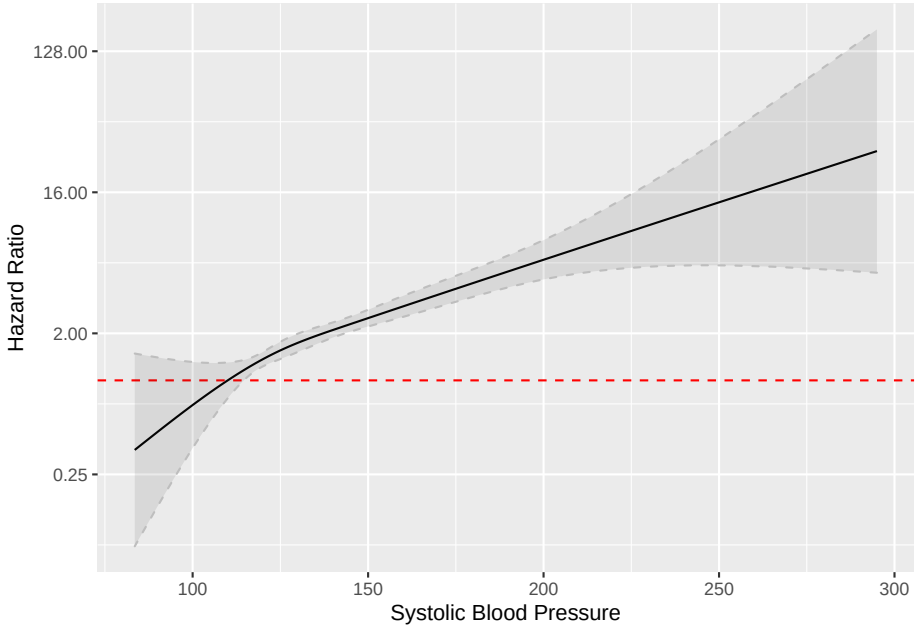
We see that there is an a HR of 1.019 (95% CI: 1.015 - 1.023) for each unit increase in systolic blood pressure. We can further explore this relationship by trying a spline term for blood pressure as well.

```
coxph_modhospmitvBMIBPsp <- coxph(Surv(time, timemi2, hospmitv) ~
                                    ns(bmi, df = 3) + ns(sysbp, df = 3) + age + sex,
                                  data = fhstv_incMI)
coxph_modhospmitvBMIBPsp %>%
  tidy()
```

```
## # A tibble: 8 x 5
##   term                estimate std.error statistic  p.value
##   <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
## 1 ns(bmi, df = 3)1      0.614    0.384       1.60  1.10e- 1
## 2 ns(bmi, df = 3)2     -0.374    1.53       -0.244 8.07e- 1
## 3 ns(bmi, df = 3)3     -0.617    1.39       -0.445 6.57e- 1
## 4 ns(sysbp, df = 3)1    2.03     0.414       4.91  9.32e- 7
## 5 ns(sysbp, df = 3)2    4.75     1.62        2.93  3.37e- 3
## 6 ns(sysbp, df = 3)3    3.77     1.04        3.62  2.96e- 4
## 7 age                   0.0243   0.00636     3.83  1.29e- 4
## 8 sex                  -1.24     0.113     -11.0   4.19e-28
```

```
# We will go ahead and plot this as well using 110 as the center
predsysbp <- termplot(coxph_modhospmitvBMIBPsp, se=TRUE, plot=FALSE) %>%
  pluck("sysbp")
predsysbp <- predsysbp %>%
  mutate(center = y[x==110],
         ylow = y - 1.96 * se,
         yhigh = y + 1.96 * se)

predsysbp %>%
  ggplot(aes(x = x, y = exp(y-center))) +
  geom_line() +
  labs(x = "Systolic Blood Pressure",
```

```
      y = "Hazard Ratio") +
  geom_hline(yintercept = 1.0, color = "red", linetype = 2)+
  geom_ribbon(aes(ymin = exp(ylow - center), ymax = exp(yhigh - center)),
              alpha = 0.1,
              linetype = "dashed",
              color="grey") +
  scale_y_continuous(trans = 'log2')
```



We see an almost monotonic increase in the HR (keep in mind the log-transformation on the y-axis) as blood pressure increases with the effect being significant for the range of exposure above ~110.

Both BMI and systolic blood pressure appear significant predictors for the hazard of MI hospitalization. You can explore other predictors in the dataset (including blood glucose, cholesterol, etc.) but keep in mind of the missingness in some of these variables and how might these affect your results.

7.4.3 Time-varying coefficients

We've already shown how some of the covariate values in the dataset change over time. Now consider the following model including smoking, and some of the other predictors from above:

```
coxph_modhospmitvall <- coxph(Surv(time, timemi2, hospmitv) ~
                                smoking + ns(bmi, df = 3) + ns(sysbp, df = 3) +
                                age + sex,
```

```
                              data = fhstv_incMI)
coxph_modhospmitvall %>%
  tidy()
```

```
## # A tibble: 10 x 5
##    term                estimate std.error statistic  p.value
##    <chr>                  <dbl>     <dbl>     <dbl>    <dbl>
##  1 smokingformer       -0.0592   0.187      -0.317 7.51e- 1
##  2 smokingcurrent       0.575    0.116       4.96  7.04e- 7
##  3 ns(bmi, df = 3)1     0.941    0.390       2.41  1.59e- 2
##  4 ns(bmi, df = 3)2     0.231    1.57        0.147 8.83e- 1
##  5 ns(bmi, df = 3)3    -0.387    1.43       -0.270 7.87e- 1
##  6 ns(sysbp, df = 3)1   2.06     0.414       4.97  6.69e- 7
##  7 ns(sysbp, df = 3)2   4.85     1.63        2.99  2.81e- 3
##  8 ns(sysbp, df = 3)3   3.78     1.05        3.60  3.22e- 4
##  9 age                  0.0321   0.00656     4.90  9.73e- 7
## 10 sex                 -1.16     0.115     -10.1   6.69e-24
```

```
coxph_modhospmitvall %>%
  tidy() %>%
  filter(term == 'smokingcurrent') %>%
  mutate(hr = exp(estimate),
         low_ci = (estimate - 1.96 * std.error),
         high_ci = (estimate + 1.96 * std.error),
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term              hr low_hr high_hr
##   <chr>          <dbl>  <dbl>   <dbl>
## 1 smokingcurrent  1.78   1.42    2.23
```

The HR for current smoking appears more elevated than in our simple models with only age and sex adjusted for. What is a possible explanation for this?

Keeping in mind the dynamic nature of some of these variable over time, what are some other potential explanations for these relationships? How does that affect our potential interpretation of the parameter coefficients for these variables? We will revisit this issue in chapter 8 of the book.

7.4.4 Using survey data (e.g. NHANES)

For this chapter, we've used data from the Framingham study as an example. Another very famous public health study in the United States is the National Health and Nutrition Examination study, more commonly known by its acronym of NHANES. We'll wrap up this chapter by talking a bit about this other famous

study, a branch of it that includes longitudinal data appropriate for a survival analysis like that covered in this chapter for Framingham, and how some quirks of the NHANES study design make the analysis of its data a bit more complex.

Just as the original Framingham study has evolved into a whole set of related studies, NHANES also began with one study and has evolved into a collection of studies. All are focused on nuitrition and health among Americans, and an enormous body of epidemiological literature has resulted from the collection of studies, covering a wide range of health outcomes and risk factors.

The core of the NHANES study collection is a survey study. These surveys are currently run continuously (every year), with around 5,000 participants a year. The surveys collect individual-level data (rather than the county- or city-level aggregated data we used for time series studies) that allow a cross-sectional view of associations between different risk factors and different measures of health.

These survey data therefore only offer a snapshot in time (the time of the survey). However, the NHANES project has supplemented the survey with some longitudinal data that can be used to study the time to events of interest. Following NHANES I, which was the first run of the survey, the study's researchers created the NHANES Epidemiologic Followup Study (NHEFS). They found as many people as they could who took the original survey (NHANES I) as adults (between ages 25 and 74) and began tracking their time to several health inputs, including mortality, and also collected updated information on risk factors and health status. The cohort was tracked during the 1980s and into the early 1990s, and follow-up for mortality among the cohort was tracked all the way until 2011. Much more information about this cohort, as well as links to available data, are available here.

The NHEFS cohort study from NHANES has been used for a wide variety of epidemiological studies. These include studies with a wide variety, including studies of folate and colon cancer risk (Su and Arab, 2001), hypertension and educational attainment (Vargas et al., 2000), chronic musculoskeletal pain and depressive symptoms (Magni et al., 1993), frequency of eating and weight change (Kant et al., 1995), physical activity and breast cancer (Breslow et al., 2001), white blood cell counts and stroke (Gillum et al., 1994), and risk factors for hip fracture (Mussolino et al., 1998).

While you can perform survival analysis on the NHEFS data, you have to take some extra steps. This is because the cohort was developed based on participants in a survey (NHANES I), and that survey selected participants in some ways that need to be addressed when you analyze the resulting data.

First, some groups were oversampled for NHANES I, including older adults, women of childbearing age, and people who lived in poorer areas. This oversampling has implications when it comes to generating estimates that generalize to the full US population, because it means that the study population isn't representative of the general population. To address this, you can use sampling weights within an analysis, which reweight the data from each study subject to

create estimates that are more appropriate to describe trends in the target population. Alternatively, in some cases it may be appropriate to do an unweighted analysis, while include variables related to the weight (e.g., age, gender, socioeconomic status) as covariates in the model (Korn and Graubard, 1991).

Second, NHANES I used a multi-stage sampling design to pick participants. It would have been impractical to randomly sample from everyone in the country, in part because the study included in-person interviews and it would be hard to get the interviewers to people spread around the country. Instead, they designed a strategy where they could send interviewers to fewer places, while still having a sample that could be used to represent the non-institutionalized US population (Korn and Graubard, 1991).

However, when a study population is selected based on this type of complex survey design, you may not be able to assume that observations are independent of each other. Instead, there may be some patterns of clustering in the data you collect—for example, people surveyed from the same sampling unit for a multi-stage sampling design may be more similar than people from different units. While this is unlikely to bias the main effect estimate, it is likely to cause you to estimate too-small standard errors (and so confidence intervals that are too tight), providing an overly optimistic view of the variance in your estimate (Korn and Graubard, 1991).

These characteristics of the NHEFS cohort were necessary to make the original survey practical enough that it could be carried out. While it does make the data more complicated to analyze, there are strategies that can be used to account for oversampling and the multi-stage sampling design. Some papers have described these approaches for the NHEFS cohort and provide a good roadmap if you'd like to explore these data yourself (Korn and Graubard, 1991; Ingram and Makuc, 1994). Further, the NHANES website includes extensive information about the NHANES series of studies, as well as tools, data, and tutorials, if you would like to try working with the data.

Chapter 8

Some approaches for confounding

8.1 Readings

The required readings for this chapter are:

- Hernán and M (2020a): IP weighting for confounding adjustment and the concept of a Marginal Structural Model
- Hernán and M (2020b): Propensity scores and outcome regression

There are also some supplemental readings you may find useful: The following is an instructional paper on constructing IP weights for regression:

- Cole and Hernán (2008)

This paper describes the use of propensity scores as an umbrella term for these types of approaches for covariate adjustment:

- Brookhart et al. (2013)

8.2 Inverse probability weighting

We've already fit some Cox models with multiple covariates, where the interpretation of each parameter coefficient is *conditional* on the other parameters in the model. Let's look back at our FHS data and reconstruct our simple models for current smoking status first unadjusted and then with age and sex also in the model.

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
```

```
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.40   1.26    1.56
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.40   1.26    1.56
```

The first model simply compares current smokers to current non-smokers in the population. It is not conditional on other variables, but also not adjusted for any potential confoudning by them. The interpretation of the (exponentiated) parameter for `cursmoke` in the adjusted model, is the Hazard Ratio comparing current smokers to current non-smokers *conditional* on age and sex. In other words this is the log-Hazard Ratio comparing currently smoking males of the same age to currently non-smoking males of the same age, and similarly currently smoking females of the same age to currently non-smoking females of the same age. This *conditioning* on other variables by including them in a regression model is often sufficient to adjust for any confounding by these variables and is the most widely used approach to adjust for confounding. In this example we see a much higher HR in the adjusted (conditional) model than the unadjusted.

One alternative approach to this is *inverse probability weighting* or IPW. IPW allows us to adjust for confounding without having to include additional variables in out final outcome model other than the exposure of interest. Instead we weigh each participant by the inverse of the probability that they had the exposure value they indeed had conditional on these other (potentially confounding variables). Typically the weight estimation will involve a model for the exposure conditional on the potential confounders. Let's go ahead and do this weight estimation for the above example using the `fhs_first` subset of the data. We will fit a logistic model for `cursmoke` conditional on age and sex:

```
model_IPW<-glm(cursmoke~age + sex, family='binomial', data=fhs_first)
model_IPW %>%
  tidy()
```

```
## # A tibble: 3 x 5
##   term        estimate std.error statistic  p.value
##   <chr>          <dbl>     <dbl>     <dbl>    <dbl>
## 1 (Intercept)   3.84     0.218        17.6 1.37e-69
## 2 age          -0.0514   0.00370     -13.9 8.06e-44
## 3 sex          -0.839    0.0634      -13.2 6.04e-40
```

We can see that these variables are significantly associated with current smoking status, however we are not going to be interpreting the parameter coefficients from this model. Instead, we will be using predictions based on this model to estimate our inverse probability weights:

```
fhs_first <-fhs_first %>%
  mutate(p.smok.obs=ifelse(cursmoke == 0,   ###estimate the conditional probability tha
```

```
         1 - predict(model_IPW, type = "response"),
         predict(model_IPW, type = "response")),
         w=1/p.smok.obs)#### The weights is the inverse of the probability of exposure value

fhs_first %>%
  summarize(mean_w=mean(w),
            sum_w=sum(w))
```

```
## # A tibble: 1 x 2
##   mean_w sum_w
##    <dbl> <dbl>
## 1   2.00 8871.
```

The mean weight value is 2, so by using the weights we'll on average be using 2 copies of each participant. Accordingly, the total sum of the weights is 8870 which is roughly double the number of participants we started with. This is because what the weights are doing in this example, is basically create a pseudopopulation where we have enough people so that everyone can be both exposed and unexposed while maintaining the distribution of covariates (here age and sex). The latter has to hold, because pseudopopulation has to be representative of the original target population. We can check this by looking at the distributions of age sex in both the weighted and unweighted data.

```
fhs_first %>%
  summarize()
```

```
## # A tibble: 1 x 0
```

The great thing about the pseudopopulaton is that the effect of `cursmoke` in no longer confounded by age and sex. We can go ahead and fit a Cox proportional hazards model for `cursmoke` using our weights, and without including age and sex in the model.

```
###Fit Cox weighted model for current smoking status
coxph_modIPW<- coxph(Surv(timedth, death) ~ cursmoke, weights=w,
                     data = fhs_first)
coxph_modIPW %>%
  tidy()
```

```
## # A tibble: 1 x 6
##   term     estimate std.error robust.se statistic       p.value
##   <chr>       <dbl>     <dbl>     <dbl>     <dbl>         <dbl>
## 1 cursmoke    0.313    0.0363    0.0532      5.88 0.00000000412
```

```
coxph_modIPW %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
```

```
         low_ci = estimate - 1.96 * std.error,
         high_ci = estimate + 1.96 * std.error,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.37   1.27    1.47
```

We see that the HR is very similar to before (they are not exaclty the same because of the non-collapsibility of the HR, more on this later). We also see that the CIs are actually narrower. This is because we are now using the pseudopopulation which is twice the number of the original population. A larger sample size means more power and by extension smaller standard errors, but in our case this larger sample size is artificial, not real and our CIs are misleadingly narrow. Rather than rely on the stanard erros from the model, we instead have to rely on robust variance estimators that counteract our artificially inflated sample size. We can do this, by using generalized estimation equations for clustered (correlated data). Our data are considered clustered, because of the multiple copies for each participant (`randid`). If we repeat the above model with a `cluster` term for `randid` this will invoke a robust sandwich variance estimation.

```
coxph_modIPW<- coxph(Surv(timedth, death) ~ cursmoke, weights=w, cluster = randid,
                     data = fhs_first)
coxph_modIPW %>%
  tidy()
```

```
## # A tibble: 1 x 6
##   term     estimate std.error robust.se statistic       p.value
##   <chr>       <dbl>     <dbl>     <dbl>     <dbl>         <dbl>
## 1 cursmoke    0.313    0.0363    0.0532      5.88 0.00000000412
```

We now have to use the `robust.se` rather than `std.error` to cnstuct our CIs:

```
coxph_modIPW %>%
  tidy() %>%
  filter(term == "cursmoke") %>%
  mutate(hr = exp(estimate),
         low_ci = estimate - 1.96 * robust.se,
         high_ci = estimate + 1.96 * robust.se,
         low_hr = exp(low_ci),
         high_hr = exp(high_ci)) %>%
  select(term, hr, low_hr, high_hr)
```

```
## # A tibble: 1 x 4
```

```
##   term        hr low_hr high_hr
##   <chr>    <dbl>  <dbl>   <dbl>
## 1 cursmoke  1.37   1.23    1.52
```

The CIs are now wider than before and more akin to the width from the original conditional model. This weighted model is essentially a Marginal Structural Cox Model. Marginal Structural Models (MSMs) refer to models that are *marginal* or *unconditional* with respect to some covariates, as is the case of our model with respect to age and sex. The interpretaton for the Cox MSM hazard ratios is no longer conditional on age and sex or in other words it is simple a comparison between current smokers and non-smokers (not within levels of age and sex).

8.3 Propensity scores

[Modeling for weights/propensity scores, involves machine learning]

Chapter 9

Mixed models

[Using a mixed modeling framework to help analyze repeated measures]

Chapter 10

Instrumental variables

Chapter 11

Causal inference

The readings for this chapter are:

The following are supplemental readings (i.e., not required, but may be of interest) associated with the material in this chapter:

- Samet (2016) A historical overview of how epidemiology tackled the question of causality for non-communicable diseases through research on the health effects of smoking

Bibliography

Andersson, C., Johnson, A. D., Benjamin, E. J., Levy, D., and Vasan, R. S. (2019). 70-year legacy of the framingham heart study. *Nature Reviews Cardiology*, page 1.

Arbuthnott, K., Hajat, S., Heaviside, C., and Vardoulakis, S. (2016). Changes in population susceptibility to heat and cold over time: assessing adaptation to climate change. *Environmental Health*, 15(1):73–93.

Arbuthnott, K., Hajat, S., Heaviside, C., and Vardoulakis, S. (2018). What is cold-related mortality? a multi-disciplinary perspective to inform climate change impact assessments. *Environment international*, 121:119–129.

Armstrong, B. (2006). Models for the relationship between ambient temperature and daily mortality. *Epidemiology*, pages 624–631.

Armstrong, B., Hajat, S., Kovats, S., Lloyd, S., Scovronick, N., and Wilkinson, P. (2012). Commentary: Climate changehow can epidemiology best inform policy? *Epidemiology*, 23(6):780–784.

Armstrong, B. G., Gasparrini, A., and Tobias, A. (2014). Conditional Poisson models: a flexible alternative to conditional logistic case cross-over analysis. *BMC Medical Research Methodology*, 14(1):122.

Balbus, J., Crimmins, A., Gamble, J., Easterling, D., Kunkel, K., Saha, S., and Sarofim, M. (2016). Introduction: Climate change and human health. In *The Impacts of Climate Change on Human Health in the United States: A Scientific Assessment*, chapter 1, pages 24–42. U.S. Global Change Research Program, Washington, DC.
****Chapter covering approach for quantitative health impact projections in the latest US Climate and Health Assessment**.

Barone-Adesi, F., Gasparrini, A., Vizzini, L., Merletti, F., and Richiardi, L. (2011). Effects of italian smoking regulation on rates of hospital admission for acute coronary events: a country-wide study. *PloS one*, 6(3):e17419.

Basu, R., Dominici, F., and Samet, J. M. (2005). Temperature and mortality among the elderly in the united states: a comparison of epidemiologic methods. *Epidemiology*, pages 58–66.

Benichou, J. (2006). Attributable risk. *Encyclopedia of Environmetrics*, 1.

Bernal, J. L., Cummins, S., and Gasparrini, A. (2017). Interrupted time series regression for the evaluation of public health interventions: a tutorial. *International Journal of Epidemiology*, 46(1):348–355.

Bhaskaran, K., Gasparrini, A., Hajat, S., Smeeth, L., and Armstrong, B. (2013). Time series regression studies in environmental epidemiology. *International Journal of Epidemiology*, 42(4):1187–1195.

Bor, J., Moscoe, E., Mutevedzi, P., Newell, M.-L., and Bärnighausen, T. (2014). Regression discontinuity designs in epidemiology: causal inference without randomized trials. *Epidemiology*, 25(5):729.

Bottomley, C., Scott, J. A. G., and Isham, V. (2019). Analysing interrupted time series with a control. *Epidemiologic Methods*, 8(1).

Bradburn, M. J., Clark, T. G., Love, S. B., and Altman, D. G. (2003a). Survival analysis part ii: multivariate data analysis–an introduction to concepts and methods. *British journal of cancer*, 89(3):431–436.

Bradburn, M. J., Clark, T. G., Love, S. B., and Altman, D. G. (2003b). Survival analysis part iii: multivariate data analysis–choosing a model and assessing its adequacy and fit. *British journal of cancer*, 89(4):605–611.

Breslow, R. A., Ballard-Barbash, R., Munoz, K., and Graubard, B. I. (2001). Long-term recreational physical activity and breast cancer in the national health and nutrition examination survey i epidemiologic follow-up study. *Cancer Epidemiology and Prevention Biomarkers*, 10(7):805–808.

Brookhart, M. A., Wyss, R., Layton, J. B., and Stürmer, T. (2013). Propensity score methods for confounding control in nonexperimental research. *Circulation: Cardiovascular Quality and Outcomes*, 6(5):604–611.

Casey, J. A., Karasek, D., Ogburn, E. L., Goin, D. E., Dang, K., Braveman, P. A., and Morello-Frosch, R. (2018). Retirements of coal and oil power plants in California: association with reduced preterm birth among populations nearby. *American Journal of Epidemiology*, 187(8):1586–1594.

Chang, W. (2018). *R graphics cookbook: practical recipes for visualizing data*. O'Reilly Media.

Clark, T. G., Bradburn, M. J., Love, S. B., and Altman, D. G. (2003). Survival analysis part i: basic concepts and first analyses. *British journal of cancer*, 89(2):232–238.

Cole, S. R. and Hernán, M. A. (2008). Constructing inverse probability weights for marginal structural models. *American journal of epidemiology*, 168(6):656–664.

Dawber, T. R., Meadors, G. F., and Moore Jr, F. E. (1951). Epidemiological

approaches to heart disease: the framingham study. *American Journal of Public Health and the Nations Health*, 41(3):279–286.

Dawber, T. R., Moore, F. E., and Mann, G. V. (2015). Ii. coronary heart disease in the framingham study. *International journal of epidemiology*, 44(6):1767–1780.

Dimick, J. B. and Ryan, A. M. (2014). Methods for evaluating changes in health care policy: the difference-in-differences approach. *Jama*, 312(22):2401–2402.

Dominici, F. and Peng, R. D. (2008a). *Exploratory Data Analysis*, chapter 5, pages 41–67. Springer.

Dominici, F. and Peng, R. D. (2008b). *Statistical Issues in Estimating the Health Effects of Spatial–Temporal Environmental Exposures*, chapter 4, pages 31–40. Springer.

Dominici, F. and Peng, R. D. (2008c). *Studies of Air Pollution and Health*, chapter 1, pages 1–6. Springer.

Dunn, P. K. and Smyth, G. K. (2018). *Statistical Models*, chapter 1, pages 1–30. Springer.

Ebi, K. L. (2015). Greater understanding is needed of whether warmer and shorter winters associated with climate change could reduce winter mortality. *Environmental Research Letters*, 10(11):111002.

Gasparrini, A. (2014). Modeling exposure–lag–response associations with distributed lag non-linear models. *Statistics in Medicine*, 33(5):881–899.

Gasparrini, A. and Armstrong, B. (2010). Time series analysis on the health effects of temperature: advancements and limitations. *Environmental research*, 110(6):633–638.

Gasparrini, A. and Leone, M. (2014). Attributable risk from distributed lag models. *BMC medical research methodology*, 14(1):1–8.

Gillum, R. F., Ingram, D. D., and Makuc, D. M. (1994). White blood cell count and stroke incidence and death: the nhanes i epidemiologic follow-up study. *American journal of epidemiology*, 139(9):894–902.

Gosling, S. N., Hondula, D. M., Bunker, A., Ibarreta, D., Liu, J., Zhang, X., and Sauerborn, R. (2017). Adaptation to climate change: a comparative analysis of modeling methods for heat-related mortality. *Environmental health perspectives*, 125(8):087008.

Greenland, S. and Robins, J. M. (1988). Conceptual problems in the definition and interpretation of attributable fractions. *American journal of epidemiology*, 128(6):1185–1197.

Haber, N. A., Clarke-Deelder, E., Salomon, J. A., Feller, A., and Stuart, E. A. (In Press). Covid-19 policy impact evaluation: A guide to common design issues. *American Journal of Epidemiology*.

Hajat, S. and Gasparrini, A. (2016). The excess winter deaths measure: why its use is misleading for public health understanding of cold-related health impacts. *Epidemiology (Cambridge, Mass.)*, 27(4):486.

Healy, K. (2018). *Data visualization: a practical introduction*. Princeton University Press.

Hernán, M. A. (2010). The hazards of hazard ratios. *Epidemiology (Cambridge, Mass.)*, 21(1):13.

Hernán, M. A. and M, R. J. (2020a). *IP weighting and marginal structural models*, chapter 12. Boca Raton: Chapman & Hall/CRC.

Hernán, M. A. and M, R. J. (2020b). *Outcome regression and propensity scores*, chapter 15. Boca Raton: Chapman & Hall/CRC.

Howe, K. B., Suharlim, C., Ueda, P., Howe, D., Kawachi, I., and Rimm, E. B. (2016). Gotta catch'em all! pokémon go and physical activity among young adults: difference in differences study. *bmj*, 355.

Imai, C., Armstrong, B., Chalabi, Z., Mangtani, P., and Hashizume, M. (2015). Time series regression model for infectious disease and weather. *Environmental research*, 142:319–327.

Ingram, D. and Makuc, D. (1994). Statistical issues in analyzing the nhanes i epidemiologic followup study. series 2: Data evaluation and methods research. *Vital and Health statistics. Series 2, Data Evaluation and Methods Research*, (121):1–30.

James, G., Witten, D., Hastie, T., and Tibshirani, R. (2013). *Linear Regression*, chapter 3, pages 59–126. Springer.

Kant, A., Schatzkin, A., Graubard, B., and Ballard-Barbash, R. (1995). Frequency of eating occasions and weight change in the nhanes i epidemiologic follow-up study. *International journal of obesity and related metabolic disorders: Journal of the International Association for the Study of Obesity*, 19(7):468–474.

Kinney, P. L. (2018). Temporal trends in heat-related mortality: implications for future projections. *Atmosphere*, 9(10):409.

Kinney, P. L., O'Neill, M. S., Bell, M. L., and Schwartz, J. (2008). Approaches for estimating effects of climate change on heat-related deaths: challenges and opportunities. *Environmental science & policy*, 11(1):87–96.

Kinney, P. L., Schwartz, J., Pascal, M., Petkova, E., Le Tertre, A., Medina, S., and Vautard, R. (2015). Winter season mortality: will climate warming bring benefits? *Environmental Research Letters*, 10(6):064016.

Korn, E. L. and Graubard, B. I. (1991). Epidemiologic studies utilizing surveys: accounting for the sampling design. *American journal of public health*, 81(9):1166–1173.

Lee, W., Kim, H., Hwang, S., Zanobetti, A., Schwartz, J. D., and Chung, Y. (2017). Monte carlo simulation-based estimation for the minimum mortality temperature in temperature-mortality association study. *BMC medical research methodology*, 17(1):1–10.

Linden, A. (2017). Challenges to validity in single-group interrupted time series analysis. *Journal of evaluation in clinical practice*, 23(2):413–418.

Lopez Bernal, J. A., Gasparrini, A., Artundo, C. M., and McKee, M. (2013). The effect of the late 2000s financial crisis on suicides in spain: an interrupted time-series analysis. *The European Journal of Public Health*, 23(5):732–736.

Lu, Y. and Zeger, S. L. (2007). On the equivalence of case-crossover and time series methods in environmental epidemiology. *Biostatistics*, 8(2):337–344.

Maciejewski, M. L. and Basu, A. (2020). Regression discontinuity design. *Jama*, 324(4):381–382.

Magni, G., Marchetti, M., Moreschi, C., Merskey, H., and Luchini, S. R. (1993). Chronic musculoskeletal pain and depressive symptoms in the national health and nutrition examination i. epidemiologic follow-up study. *Pain*, 53(2):163–168.

Mendola, P. (2018). Invited commentary: The power of preterm birth to motivate a cleaner environment. *American journal of epidemiology*, 187(8):1595–1597.

Mussolino, M. E., Looker, A. C., Madans, J. H., Langlois, J. A., and Orwoll, E. S. (1998). Risk factors for hip fracture in white men: the nhanes i epidemiologic follow-up study. *Journal of Bone and Mineral Research*, 13(6):918–924.

Northridge, M. E. (1995). Public health methods–attributable risk as a link between causality and public health action. *American journal of public health*, 85(9):1202–1204.

Robinson, D. (2014). broom: An r package for converting statistical analysis objects into tidy data frames. *arXiv preprint arXiv:1412.3565*.

Samet, J. M. (2016). Epidemiology and the tobacco epidemic: how research on tobacco and health shaped epidemiology. *American journal of epidemiology*, 183(5):394–402.

Sargent, R. P., Shepard, R. M., and Glantz, S. A. (2004). Reduced incidence of admissions for myocardial infarction associated with public smoking ban: before and after study. *Bmj*, 328(7446):977–980.

Steenland, K. and Armstrong, B. (2006). An overview of methods for calculating the burden of disease due to specific risk factors. *Epidemiology*, pages 512–519.

Su, L. J. and Arab, L. (2001). Nutritional status of folate and colon cancer risk: evidence from nhanes i epidemiologic follow-up study. *Annals of epidemiology*, 11(1):65–72.

Tobıas, A., Armstrong, B., and Gasparrini, A. (2017). Investigating uncertainty in the minimum mortality temperature. *Epidemiology*, 28:72–76.

Turner, S. L., Karahalios, A., Forbes, A. B., Taljaard, M., Grimshaw, J. M., Korevaar, E., Cheng, A. C., Bero, L., and McKenzie, J. E. (2021). Creating effective interrupted time series graphs: Review and recommendations. *Research synthesis methods*, 12(1):106–117.

Valenzuela, P. L., Carrera-Bastos, P., Gálvez, B. G., Ruiz-Hurtado, G., Ordovas, J. M., Ruilope, L. M., and Lucia, A. (2021). Lifestyle interventions for the prevention and treatment of hypertension. *Nature Reviews Cardiology*, 18(4):251–275.

Vargas, C. M., Ingram, D. D., and Gillum, R. F. (2000). Incidence of hypertension and educational attainment the nhanes i epidemiologic followup study. *American Journal of Epidemiology*, 152(3):272–278.

Venkataramani, A. S., Bor, J., and Jena, A. B. (2016). Regression discontinuity designs in healthcare research. *Bmj*, 352.

Vicedo-Cabrera, A. M., Sera, F., and Gasparrini, A. (2019). Hands-on tutorial on a modeling framework for projections of climate change impacts on health. *Epidemiology*, 30(3):321–329.

Wickham, H. and Grolemund, G. (2016). *R for data science: import, tidy, transform, visualize, and model data.* " O'Reilly Media, Inc.".

Wong, N. D. (2014). Epidemiological studies of chd and the evolution of preventive cardiology. *Nature Reviews Cardiology*, 11(5):276–289.

Wong, N. D., Cupples, A., Ostfeld, A. M., Levy, D., and Kannel, W. B. (1989). Risk factors for long-term coronary prognosis after initial myocardial infarction: the framingham study. *American Journal of Epidemiology*, 130(3):469–480.

Zhou, B., Perel, P., Mensah, G. A., and Ezzati, M. (2021). Global epidemiology, health burden and effective interventions for elevated blood pressure and hypertension. *Nature Reviews Cardiology*, pages 1–18.

Ziaeian, B. and Fonarow, G. C. (2016). Epidemiology and aetiology of heart failure. *Nature Reviews Cardiology*, 13(6):368–378.