Table of contents

Welcome to this book of notes **2**

1 Chapter 1 - Headings **3**
1.1 Models for binary/binomial response . . . 3
1.2 Section title . . . 4

2 Week 2 – Code demonstrations **6**
2.1 Features . . . 6
2.2 New Section Title . . . 7
2.3 Figures . . . 8

3 Links chapter **11**
3.1 Equations . . . 11
3.2 Handling videos . . . 12

4 Chapter 4 - Nicer boxes and Python **13**
4.1 Custom Numbered Blocks . . . 14
4.2 Demonstration of Custom Numbered Blocks . . . 16
4.3 Referencing the Blocks . . . 17
4.4 Customization . . . 17
4.5 More Python . . . 18

5 Chapter 5 **23**
5.1 R package: `webexercises` . . . 23
5.2 Numbas embedding . . . 24

6 Chapter 6 **26**
6.1 Invisible text . . . 26

7 Introduction to Generalised Linear Models **28**
7.1 Introduction and motivating examples . . . 28
7.2 What do the Bollywood box office and medical school admission examples have in common? . . . 45
7.3 Exponential family of distributions . . . 46
7.4 Generalised Linear Models . . . 49
7.5 Maximum likelihood estimation of GLM coefficients . . . 54
7.6 Inference for GLMs . . . 57

Welcome to this book of notes

1 Chapter 1 - Headings

Each row that begins with a # is a chapter heading. So if each chapter is a single file it's best to only use a single # line at the top of each document.

1.1 Models for binary/binomial response

This chapter contains sections and subsections. This is the intro. Since the chapter title begins with a `#`, sections begin with `##`, and subsections should begin with `###`.

Typical default numbering will look like this:

```
Chapter 1
Section 1.1
Subsection 1.1.1
Subsection 1.1.2
Section 1.2
Subsection 1.2.1
Subsection 1.2.2
Subsection 1.2.3
Section 1.3
```

The `crossref` option in your global `.yml` file does offer the chance to not number any chapters and just have sections in each chapter, if you really don't want chapters.

1.1.1 First subsection

Text goes here.

1.1.2 Second subsection

Text goes here. And we can reference whole sections if we wish, like Section 1.1. Indeed if you look in Chapter 3.

Let

$$D_3(y) \equiv \frac{\mathrm{d}^3 y}{\mathrm{d}x^3}$$

That was just an equation inside `$$` and `$$` tags. Equations must be defined like this, they can be labelled (unlike in LaTeX). You should also avoid using LaTeX environments which begin math mode, like `align`, instead use the `-ed` version `aligned` which assumes you're already inside math mode.

$$\int_0^\infty x^3 \, \mathrm{d}x \tag{1.1}$$

Here's some more text

- here's bullet point $x + y = z$
- another bullet

More text.

1.2 Section title

Here's some sample text from Advanced Predictive Models.

We will begin with models for binomial responses and we will look at exploratory plots of the data, different choices of link function and hypothesis tests about terms in the model. Finally we will examine measures of goodness of fit of the model. Let's begin with an example.

Beetle mortality

This is standard Quarto a "Tip Callout" box called via

```
:::{.callout-tip}
### Banner title like Bettle Mortality

Content here. This gives coloured boxes in HTML and LaTeX.
:::
```

However, see Chapter 4 for my recommended alternative to these callout boxes. Now for a default code `panelset` as Quarto calls it. For allowing dual code presentation. See Chapter 2 for my restyled version.

1.2.1 R

```
beetles <-
 ↪  read.csv(url("http://www.stats.gla.ac.uk/~tereza/rp/beetles.csv"))
beetles
```

```
    dose number killed
1 1.6907     59      6
2 1.7242     60     13
3 1.7552     62     18
4 1.7842     56     28
5 1.8113     63     52
6 1.8369     59     53
7 1.8610     62     61
8 1.8839     60     60
```

1.2.2 Python

```
import pandas as pd
beetles = pd.read_csv("beetles.csv")
beetles
```

This Python code was told not to evaluate. Note it calls a local csv file, which would need to be in the right place. Paths will be relative to the `.qmd` file being compiled.

2 Week 2 – Code demonstrations

Some equations, again. Notice this chapter begins with:

```
---
filters:
    - acc-tools
---
```

This loads the bonus accessibility features extension. There is a global code toggle button bottom left, and the R and P keyboard buttons will switch **all** code tabs simultaneously to the chosen language.

Documentation for it can be found here: UofGStats/acc-tools repo

$$\frac{\mathrm{d}^3 y}{\mathrm{d}x^3}$$

$$\int_0^\infty x^2 \, \mathrm{d}x$$

This equation is the same as Equation 1.1 from Chapter 1. Since we're in the book template, and all files are compiled simultaneously into one output we are allowed to link between chapters. This is the fundamental difference between the booktemplate and the weektemplate.

2.1 Features

- You'll first note in the bottom left is a global toggle button set. You can change all code blocks to R or to Python, by clicking; or by pressing `R` or `P` on the keyboard.
- There tabs labelled R and Python have been moved to the right, and when selected (for example by pressing tab) are much more clearly selected.
- Also the copy-code button is always visible.

2.2 New Section Title

Some more placeholder text here. Even in the book template every `.qmd` that wishes to use python needs to load it via `reticulate`. This is in the template:

````
```{r}
#| label: reticulate-setup
library(reticulate)
#These next two lines need to run ONCE on your machine
#reticulate::virtualenv_create("r-quarto")
#reticulate::py_install(c("pandas","seaborn","matplotlib","numpy"), envname =
 ↪  "r-quarto")
reticulate::use_virtualenv("r-quarto", required = TRUE)
```
````

Another standard callout box.

Beetle mortality

Jump to the Beetle code.
This is an R `.panel-tabset` environment. It's the recommended route to display more than one codeblock at once, switchable via a tab on the top edge.
To make life easier, my extension also adds a global-toggle button which can be used by clicking (see bottom left), or just pressing R or P to toggle **all** code blocks to R or Python respectively. I've also tidied up their appearance, made the tabs more prominent and easier to navigatve with a keyboard.
The code blocks themselves are just like usual RMarkdown blocks you define a language and decide if you want the code to be shown, and if you want the results to be rendered on the screen too.

2.2.1 R

The code in the R example is folded by default. Click on `Code` to reveal it.

```
beetles <-
 ↪  read.csv(url("http://www.stats.gla.ac.uk/~tereza/rp/beetles.csv"))
beetles
```

```
    dose number killed
1 1.6907     59      6
2 1.7242     60     13
3 1.7552     62     18
4 1.7842     56     28
5 1.8113     63     52
6 1.8369     59     53
```

```
7 1.8610     62     61
8 1.8839     60     60
```

2.2.2 Python

```
import pandas as pd
beetles = pd.read_csv("resources/data/beetles.csv")
beetles
```

```
     dose  number  killed
0  1.6907      59       6
1  1.7242      60      13
2  1.7552      62      18
3  1.7842      56      28
4  1.8113      63      52
5  1.8369      59      53
6  1.8610      62      61
7  1.8839      60      60
```

2.3 Figures

Quarto calls various different things Figures. See Quarto figures.

In particular, including external images (pre-generated) can be done in many ways and are regarded as figures. However, so are plots generated by R, and indeed any computational blocks of code you want it to run in R or Python.

As usual you use ```` ```{r} ```` to start an R block (which you want to run), and ```` ```{python} ```` for python.

To include pre-rendered images the typical format is inline like this:

```
![Elephant](elephant.png){fig-alt="A drawing of an elephant."}
```

The word Elephant from the first bracket becomes the caption and the `fig-alt` determines the alt-text for screenreaders. A lot more options for controlling size and layout are in the Quarto documentation.

Information on accessibility

For accessibility purposes there a number of actions you should always be taking, but may not be used to.

1. Always add an alt-text to your images. These are designed to be read aloud and describe what someone looking at the image is suppose to get from

it. There is lots of advice online for what good alt-text looks like. It's a particularly rich topic in maths/stats where a plot may need a moderately long description to provide for a non-sighted user what you wish sighted users to have got from it.

2. Always add alt-text to R or Python generated plots, the recommended approach has changed in recent years, now you use the `#|` syntax to add properties of your blocks.

Here's an example:

````
```{r}
#| label: fig-myplot
#| fig-cap: "Here's my caption"
#| fig-alt: "This is the possibly long alt text describing the plot"
#| code-fold: show

plot(cars)
```
````

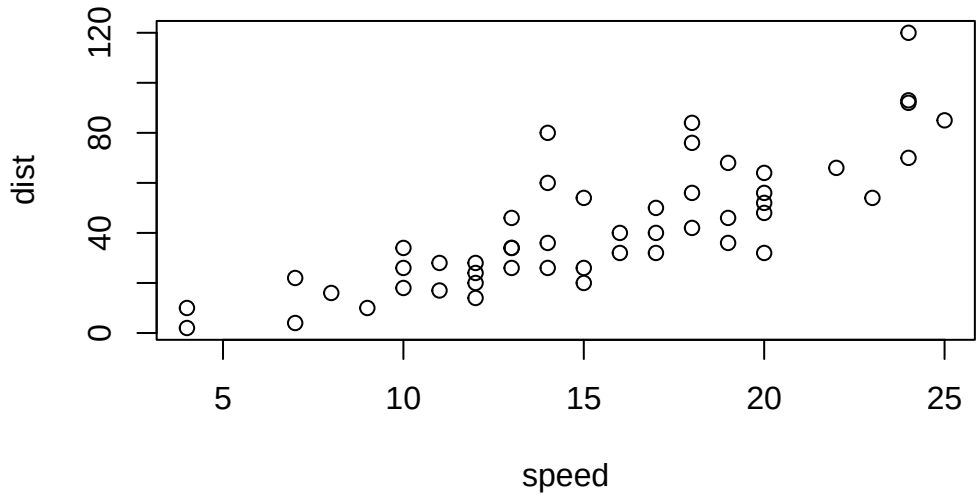


Figure 2.1: Here's my caption

Note that by default the code which generated the plot is also shown. This was made foldable (in the HTML version) with `#| code-fold: show` but could be hidden entirely with options like `#| echo: false`.

This figure caption is written in a dark font color. The Quarto default is often too pale and fails accessibility checks, so the acc-style files in my extension fix this, along with a few other things like it (table of contents contrast too).

2.3.1 Python

```
3+34
```

```
37
```

2.3.2 R

```
34+3
```

```
[1] 37
```

2.3.3 Python

```
5+4
```

```
9
```

2.3.4 R

```
4+5
```

```
[1] 9
```

Note that our plot before was called `#fig-myplot` so we can use `@fig-myplot` to reference Figure 2.1.

3 Links chapter

This chapter can even link back to other whole chapters Section 1.1. Or equations in other chapters Equation 1.1. This is only true for the book template.

3.1 Equations

There is good Quarto documentation on how to crossref things Quarto Cross References.

The key thing to note is that if you plan to reference a standard object like a heading/section, the internal label name is restricted to begin with a particular three-letter prefix (except equations which get just the two letters `eq`). Plus Quarto recommends against using underscores in labels, hyphens are great.

New prefixes can be created, as this template does below to create new figures for videos called `vid`. Various default ones already exist like Theorems, Lemmas etc.. this template will demo the `numbered-custom-boxes` in Chapter 4 as an alternative. This alternative gives fine-grained numbering control, so you'll need to avoid Quarto's numbering. So if you do follow this recommendation then you must avoid using standard Quarto names like `#thm-fermat` because Quarto will be confused about which method you wish. So in the examples presented in Chapter 4 I have just started all references with `#r` to avoid overlap, e.g. `#rthm-fermat`.

Here's an equation:

$$x^2 + y^2 = z^2 \tag{3.1}$$

This equation was created using Quarto labelling syntax, i.e. like this

```
$$
x^2 + y^2 = z^2
$$ {#eq-squares}
```

Then you type `@eq-squares` to get Equation 3.1.

This section was defined like this

```
## Equations {#sec-equations}
```

as such we can now type `@sec-equations` and it will render as Section 3.1.

Quarto even provides a clever 'hover to see a preview function'. (There is a way to disable it if you really wish)

The main difference/drawback to using the Quarto numbering system for equations is in equation blocks (equations on many consecutive lines), where LaTeX offers individually labelling of lines Quarto doesn't. There is a more complex workaround to delegate all labelling of equations to MathJax which can replicate LaTeX behaviour but it's easier to work around it.

3.2 Handling videos

One way to include videos is to use the Quarto syntax used to insert any figures. You can define a new type of figure. This template has defined a new float object called `vid`, then we can embed videos (using normal Quarto code) but label them and caption them like a figure but use `vid` instead of `fig`.

```
::: {#vid-vid3}
{{ video https://youtu.be/d2nG7Y17sQw
  title = "Binomial response models applied to beetle mortality data (9m59s)",
  <other customizations can be put here>
>}}

This is the caption.
:::
```

For more details: Quarto video manual

Here it is:

https://youtu.be/d2nG7Y17sQw

Video 3.1: This is the caption.

Referencing is simple, like this Video 3.1. Such video blocks must contain actual text (the caption is drawn from the final line). You must use the three-letter code defined in Quarto for your object type.

Personally I prefer to hide the videos from direct view to keep the window clean, so you'll see an alternative method in Chapter 4, where I put them inside folded boxes.

4 Chapter 4 - Nicer boxes and Python

This chapter shows off some nice coloured boxes. They come from the `custom-numbered-blocks` extension. First you need to install the extension `quarto add ute/custom-numbered-blocks`.

The following colour definitions need to be added somewhere (e.g. `_quarto.yml` or `_metadata.yml` or at the top of this `.qmd` file) to load the coloured boxes. In this template I have added a link to a file called `_numbered-boxes.yml` from inside `_quarto.yml`. In this way you just need to look insider `_numbered-boxes.yml` to edit the settings.

It looks approximately like this, you define groups of like-numbered boxes, and then define specific classes (and their names later). They can have separate colour schemes too.

```
filters:
  - custom-numbered-blocks # This loads the extension

# This is ute-hahn's nice coloured boxes: setup
custom-numbered-blocks:
  groups:
    thmlike:
      colors: [948bde, 584eab]
      boxstyle: foldbox.simple
    deflike:
      colors: [ffdc36, ffb948]
      boxstyle: foldbox.simple
    videolike:
      colors: [beb9eb, 877eda]
      boxstyle: foldbox.simple
      collapse: open
  classes:
    Theorem:
      group: thmlike
    Lemma:
      group: thmlike
    Corollary:
      group: thmlike
    Definition:
      group: deflike
    Video:
      group: videolike

crossref:
  chapters: true
```

4.1 Custom Numbered Blocks

We've defined several custom-numbered block types using the `custom-numbered-blocks` option. The key point is that these can be numbered and follow grouped numbering conventions.

Each block belongs to a **group** that determines its shared numbering.

All of these blocks are *foldable*, meaning they can be opened and closed. The `collapse` setting determines the starting state. Most boxes have been set to be open (`collapse: false`) at the group level, but this can be overridden at both lower levels. e.g. Corollary shares numbering with all the theorem-like boxes, which are all open by default. Corollaries could be all set to closed by default, or even on a case by case basis in the code.

Table 4.1: This is the table's caption

Box Name	Group	Title Background Color	Left Border Color	Default State
Theorem	`thmlike`	`#948bde`	`#584eab`	Open / Closeable
Lemma	`thmlike`	`#b2dac4`	`#004721`	Open / Closeable
Corollary	`thmlike`	`#7fbad7`	`#005398`	Open / Closeable
Definition	`deflike`	`#ffdc36`	`#ffb948`	Open / Closeable
Video	`videolike`	`#d18888`	`#660000`	Closed / Openable

These boxes also have dark-mode versions, see below. Notice this table had a caption, and it can be referenced with normal Quarto syntax: Table 4.1. They can also have styling added.

> Note: The two colors defined for each group represent the **left border** and the **title background**.

Here's an alternative to treating videos as float/figures, just stick them directly onto the page immediately (not as a floating object) but wrap them in a coloured box and use the title of the box to describe the video.

Video 1: Binomial response models applied to beetle mortality data (9m59)

There is some text here before the video
https://youtu.be/d2nG7Y17sQw

Example 1: Beetle mortality data

Here's some famous beetle data. The R code is folded (and thus hideable), the Python code isn't folded so cannot be folded away.

4.1.1 R

```
beetles <-
 ↪  read.csv(url("http://www.stats.gla.ac.uk/~tereza/rp/beetles.csv"))
beetles
```

```
    dose number killed
1 1.6907     59      6
2 1.7242     60     13
3 1.7552     62     18
4 1.7842     56     28
5 1.8113     63     52
6 1.8369     59     53
7 1.8610     62     61
8 1.8839     60     60
```

4.1.2 Python

```
import pandas as pd
beetles = pd.read_csv("resources/data/beetles.csv")
beetles
```

```
     dose  number  killed
0  1.6907      59       6
1  1.7242      60      13
2  1.7552      62      18
3  1.7842      56      28
4  1.8113      63      52
5  1.8369      59      53
6  1.8610      62      61
7  1.8839      60      60
```

Conventional tip callout

Some content

Corollary 1: Super

This is a fantastic Corollary.

There are no Corollary's better than Corollary 1. Note that referencing of these coloured boxes works like LaTeX's `\ref{}` command, thus it just renders the number, and you have to write the object name in front. This is in contrast to the `@cor-name` Quarto notation.

Full documentation on these coloured boxes is available at https://github.com/ute/custom-numbered-blocks. Including the new `\longref{}` syntax allowing the object name to be included in the reference. Like this Corollary 1.

The other way to embed a video is inside a fenced div and treat it like a figure, as we saw in other chapters.

Our video in this chapter was embedded inside a custom numbered block and it was called Video 1, in a moment we'll include another video. It will be called Video 2.

4.2 Demonstration of Custom Numbered Blocks

Below are examples of the custom-numbered blocks.
Each block is foldable (open or closed by default) and has its own colour scheme and group style as defined in your YAML configuration.

Notice Theorem, Corollary and Lemma has been defined to share a counter. Unlike Video and Definition which are different types (*vidlike* and *deflike*).

Theorem 2: Pythagoras' Theorem

In a right-angled triangle…

Corollary 3: Corollary to Pythagoras' Theorem

In a right-angled triangle, if the two shorted sides are equal, so that $a = b$, then the hypotenuse satisfies $c = a\sqrt{2}$.

Lemma 4: Euclid

If two lines are perpendicular to the same line, then they are parallel to each other.

Video 2: Video Demonstration

Here's a Youtube video:
https://youtu.be/d2nG7Y17sQw

Definition 1: Definition of Orthogonality

Two vectors $\mathbf{u}$ and $\mathbf{v}$ are said to be *orthogonal* if their dot product is zero:

$$\mathbf{u} \cdot \mathbf{v} = 0.$$

4.3 Referencing the Blocks

You can reference the blocks defined above using their labels:

- The main result is stated in Theorem 2.
- Checkout Corollary 3.
- The proof uses Lemma 4.
- Then there's Definition 1.
- Finally, see Video 2.

If we want we can wrap the `\ref{}` inside a hyperlink to get Theorem 2, Corollary 3, etc...

Update: The `custom-numbered-blocks` extension now offers `\longref{}` as a way to make the hyperlink include the word of the type of object, like `\longref{rthmpythag}` now rendering as Theorem 2.

4.4 Customization

Each box's definition uses two colours. In the file mentioned above the light-mode border and background are set. For those using dark-mode you also need to define a dark mode pair (so you'll need four total colours). See dark styles boxes stylesheet for how to define them as it's not available as a default setting.

4.5 More Python

Rendering documents containing R and Python is based on the `reticulate` and `knitr` R packages.

One way is to let Quarto manage a Python environment for you via `quarto install python`, but if you already have Python installed, the recommended approach now is to use a **persistent virtual environment**.

4.5.1 Setting up a persistent Python environment

Look at the top of each file and you'll find a chunk like this

````
```{r}
#| label:  reticulate-setup
#| include: false
library(reticulate)

# These next two lines need to run ONCE on your machine to create the
 ↪  environment
# reticulate::virtualenv_create("r-quarto")
# reticulate::py_install(c("pandas","seaborn","matplotlib","numpy"), envname =
 ↪  "r-quarto")

# Use the virtual environment in this document
reticulate::use_virtualenv("r-quarto", required = TRUE)
```
````

- **First time only:** Uncomment the `virtualenv_create` and `py_install` lines to set up the environment.
- **After setup:** Only the `use_virtualenv` line is needed.

The advantage of this approach is that libraries are installed once and persist across sessions, so you don't incur installation overhead every time you render the document.

If while developing your notes you realize you need to load another library, then re-run the `py_install` line to install this library into your `r-quarto` environment, e.g. by running these lines in the console:

```
library(reticulate)
reticulate::py_install(c("insert-new-library-here"), envname = "r-quarto")
```

4.5.2 Using Python in your Quarto document

Once the environment is set, you can run Python chunks as usual:

This will demonstrate the R/Python switcher again. Notice this page doesn't contain the global toggle button in the bottom left as it was disabled at the top of the `.qmd`, should you wish to disable it.

4.5.3 R

```r
x <- rnorm(50)
plot(x)
```

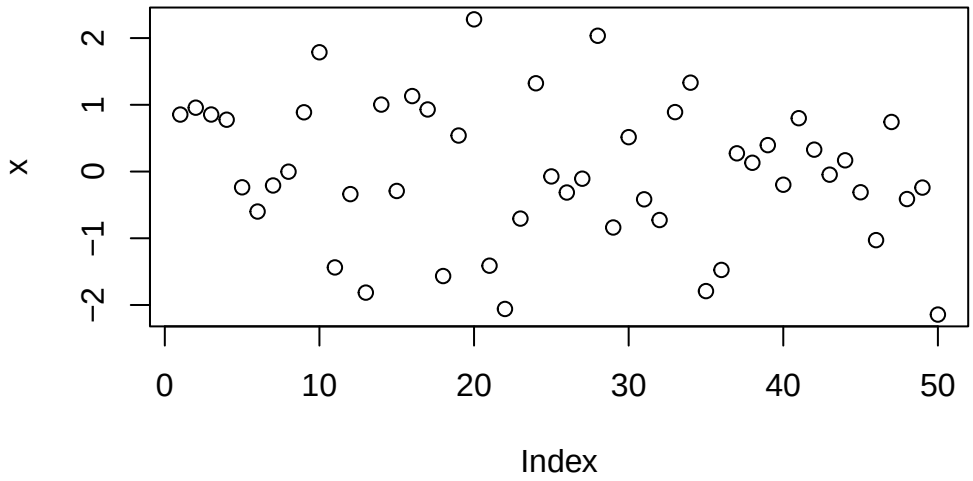


Figure 4.1: A base R plot

4.5.4 Python

```python
import numpy as np
import matplotlib.pyplot as plt

r = np.arange(0, 2, 0.01)
theta = 2 * np.pi * r
fig, ax = plt.subplots(
  subplot_kw = {'projection': 'polar'}
)
ax.plot(theta, r)
ax.set_rticks([0.5, 1, 1.5, 2])
```

```
ax.grid(True)
plt.show()
```

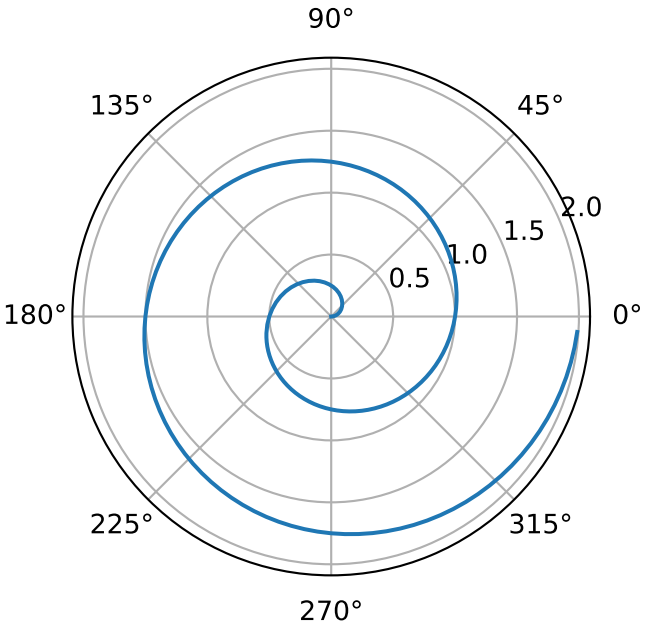


Figure 4.2: A line plot on a polar axis

4.5.5 R

```
x <- seq(-10,10, by = 0.1)
y <- x ^ 3
plot(x, y, type = "l")
```

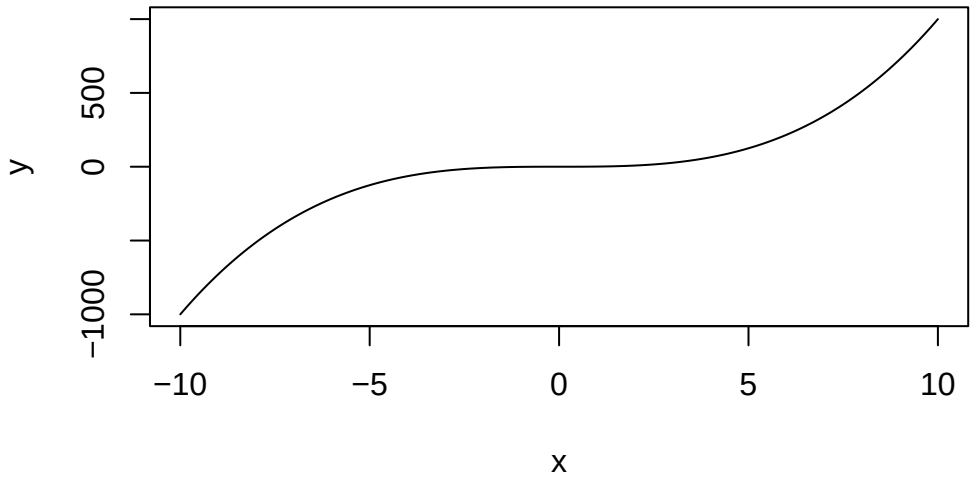


Figure 4.3: $f(x) = x^3$ between -10 and $+10$

4.5.6 Python

```
import matplotlib.pyplot as plt
import numpy as np

xpoints = np.arange(-10, 10, 0.1)
ypoints = np.power(xpoints, 3)

plt.plot(xpoints, ypoints)
plt.show()
```

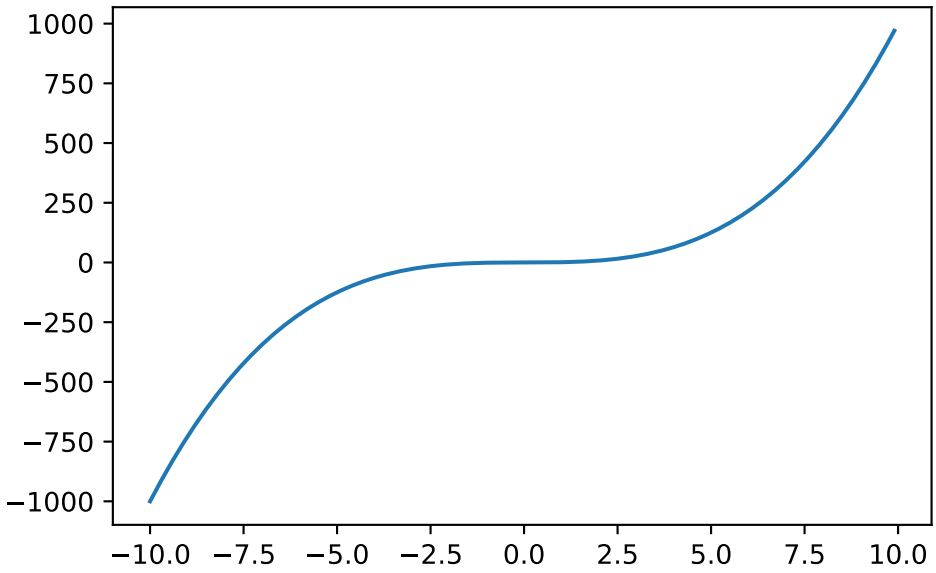


Figure 4.4: $f(x) = x^3$ between -10 and $+10$

The output images from these pieces of code have their own labels namely `@fig-rnorm` and `@fig-polar` which reference as Figure 4.1 and Figure 4.2.

5 Chapter 5

5.1 R package: `webexercises`

Some people are using external R packages like `webexercises`, they will just work out of the box. You load them as usual. I've added an extra function to attach accessibility aria labels to the entry boxes later.

````
```{r}
library(webexercises)

# To allow injection of aria-labels
in_labelled_fitb <- function(answer, label_text, ...) {
  html <- webexercises::fitb(answer, ...)
  # Insert aria-label into the <input ...> tag
  html <- sub(
    "<input ",
    paste0("<input aria-label=\"", label_text, "\" "),
    html
  )
  html
}
```
````

The `webexercises` package also comes with a css and js file, so you typically need to add the following to the `_metadata.yml` in the folder for the file being rendered (or in the global `_quarto.yml`) if you want it literally everywhere.

```
format:
  html:
    css:
      - include/webex.css
    include-after-body: include/webex.js
```

Here's an example someone may recognise.

Example 1

Consider a function taking arguments `a`, `b` and `c` and returning $(a + 2 \times b)^c$, defined as set out below.

```
func <-  function(a, b = 0, c = 1){
    (a + 2 * b)^c
}
```

Rewrite the calls below into named form (i.e. using `parameter = value`) and give the output of `func` without running it in R.

```
func(1, 2, 3)
```

This is the equivalent to _____ and it returns the value ____.

5.2 Numbas embedding

Another opportunity with HTML is to embed via an iframe to some external content.

I have created an extension which makes including iframes easier.

You need to first create your Numbas question, then preview it, and click *Download*. This provides a zip file, which you unzip and place the folder into `/resources/numbas/<your-chosen-name>`. Then in the code below you just point to the `index.html` for the question. Questions are of varying heights and widths too, so you can customize it in when you create the iframe.

5.2.1 Plain version (no web version)

```
{{< iframe path = "/resources/numbas/Q13/index.html"
  width = "90%"
  height = "900px"
  >}}
```

A question appears here in the HTML version.

5.2.2 Version with a hyperlink in the PDF

If you want the PDF version to contain an actual hyperlink then you need to get a link to a website version of your question, like this example below. You could try and always use

this link rather than download approach described above, but then you are dependent on the external website being live at all times.

```
{{< iframe path = "/resources/numbas/Q13/index.html",
  width = "90%",
  height = "900px",
  url = "https://numbas.mathcentre.ac.uk/question/119566/matrix-lin\
  ear-combination/embed/?token=e221efa1-3a50-429c-bb4a-8c63d2610114"
>}}
```

Link to Numbas question

> **Note**
>
> I inserted the \ symbol in the long url address, to render nicely for you, but users don't normally see it.
> When presenting the user never sees the code anyway.

5.2.3 Hiding a Numbas task

An alternative to having a large space-using question, is to hide inside a collapsed box like the following.

Task 1

A question appears here in the HTML version.

6 Chapter 6

6.1 Invisible text

I have a Quarto extension Invisible text extension to make hiding text easy. Mostly for use in gapped notes, or perhaps handouts.

It implemented by simply setting text to fully transparent (and not selectable). Screen-readers **will** be able to see it, and they will also get informative notifications about the visibly hidden text. In LaTeX/PDF it is rendered as white text.

Just install it as any other extension via

```
quarto add david-hodge/invis
```

and then you can use it via usage of the following shortcodes.

6.1.1 Simple hiding

`{{< invis start >}}` This starts invisibility
`{{< invis end >}}` This ends invisibility

By default these are designed to hide a single block/element on a page. It can hide a single equation, a single word, a single line or text, etc.. *However* it cannot hide multiple paragraphs or more complex elements.

6.1.2 Hiding larger elements

To hide more complex elements, use the `block` extra paramater.

`{{< invis start block >}}` This starts invisibility for a large element
`{{< invis end block >}}` This ends invisibility for a large element

There are limitations for complex multiple blocks (see below).

6.1.3 Examples

This first sentence is fully visible.

This third sentence is fully visible.

> **Warning**
>
> I don't imagine you would wish, but should you wish to hide some fundamental Quarto object like a code block and the resulting output you are recommended to use the built-in features like setting `echo: false` on the code block, or read about *Conditional Formatting* in Quarto.

You can use this `invis` extension to hide large blocks of content in the HTML, but for such elements they will likely appear in the PDF, since all we are doing in the PDF is setting standard text as white, and code and plots are not standard text.

```
plot(cars)
```

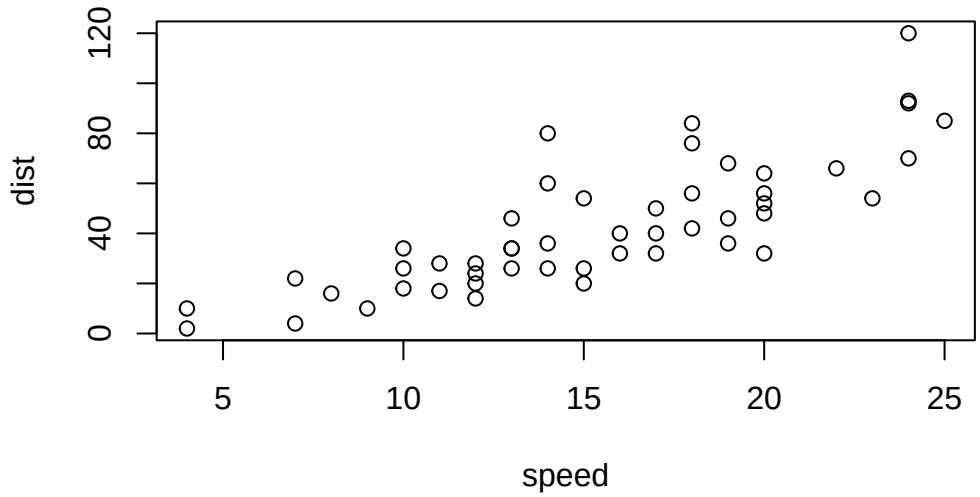


Figure 6.1: This is the caption

The code and output plot above are fully hidden in the HTML. Likely fully visible in the PDF. If you already have a favourite way of hiding in LaTeX this is likely usable, but too complex for this guide.

7 Introduction to Generalised Linear Models

Learning Outcomes

- See how a generalised linear model differs from a linear model, including the approach taken in logistic regression
- Meet the exponential family of distributions
- Be able to identify the key components of a generalised linear model (GLM)
- Appreciate the use of likelihoods, maximum likelihood estimation, and the definition of deviance in GLM modelling
- Appreciate the consequences of using maximum likelihood estimation for inference, including confidence intervals, and the use of deviance in likelihood ratio tests

These outcomes are further reinforced with concrete examples in coming weeks.

7.1 Introduction and motivating examples

In this course we will extend the theory of linear regression models to non-normal responses. Before we begin with introducing the class of models known as Generalised Linear Models (GLMs), we will briefly illustrate why linear models are not sufficient for all types of data. Throughout the course, we will see how we can deal with a variety of situations where the linear model may not be adequate.

The main objective of this week's learning material is to introduce Generalised Linear Models (GLMs), which extend the linear model framework to outcome variables that don't follow the normal distribution. GLMs can be used to model non-normal continuous outcome variables, but they are most frequently used to model **binary**, **categorical** or **count data**. We will focus on these latter types of outcome variables. To see why extensions to the normal linear model are needed, let's look at a couple of examples, one where the normal linear model is appropriate and one where it's not.

Example 1: Bollywood box office revenue

Possibly the simplest scenario of a predictive model is when we want to predict an outcome variable based on a predictor which displays a linear relationship to the variable of interest. Consider the following dataset on Bollywood film revenues (sourced from http://www.bollymoviereviewz.com) which contains data on 190 films made during the period 2013-2017. We would like to predict the gross revenue of a film from the film's budget. Both the gross revenue and the budget are measured in crore. Here are the first few rows of the data:

7.1.1 R

```
bollywood <- read.csv('https://github.com/UofGAnalyticsData/APM/raw/refs/h
 ↪  eads/main/bollywood_boxoffice.csv', fileEncoding =
 ↪  "latin1")
head(bollywood)
```

```
                       Movie  Gross Budget
1                 Ek Villain   95.64   36.0
2                 Humshakals   55.65   77.0
3                    Holiday  110.01   90.0
4                      Fugly   11.16   16.0
5                City Lights    5.19    9.5
6 Kuku Mathur Ki Jhand Ho Gayi   2.23    4.5
```

7.1.2 Python

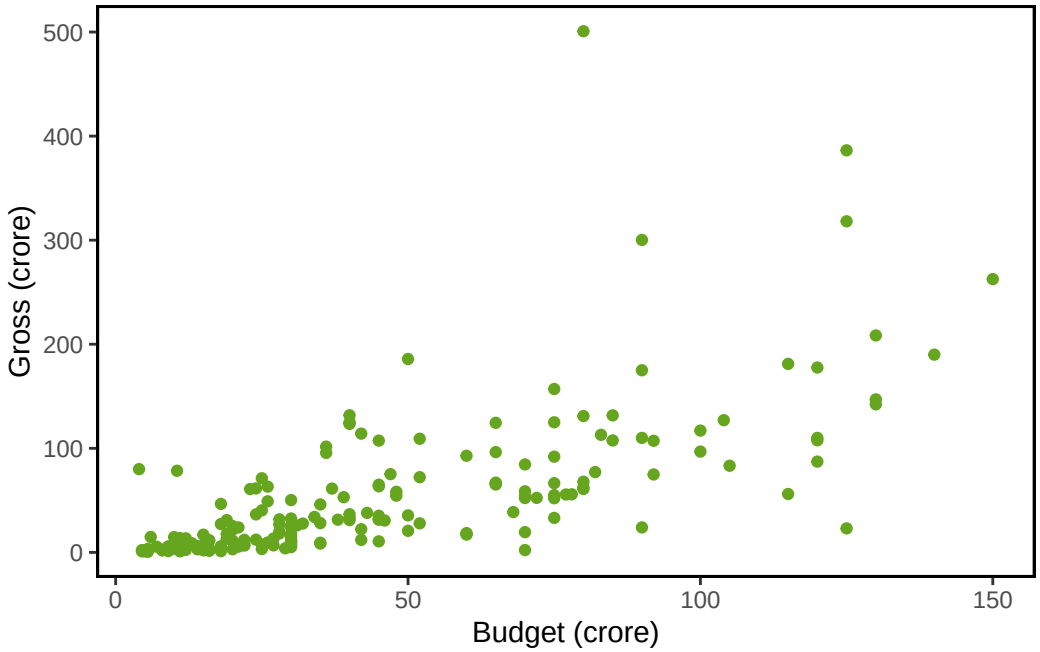
```
import pandas as pd
bollywood = pd.read_csv("resources/ForChX/bollywood_boxoffice.csv",
 ↪  encoding = 'latin-1')
bollywood.head()
```

```
          Movie   Gross  Budget
0   Ek Villain    95.64    36.0
1   Humshakals    55.65    77.0
2      Holiday   110.01    90.0
3        Fugly    11.16    16.0
4  City Lights     5.19     9.5
```

We can plot the gross revenue against the budget to explore the relationship between the two variables.

7.1.3 R

```
b.plot <- ggplot(data = bollywood, aes(y = Gross, x = Budget)) +
          geom_point(col = "#66a61e") +
          scale_x_continuous("Budget (crore)") + scale_y_continuous("Gross
           ↪  (crore)")
```



7.1.4 Python

```
sns.scatterplot(data=bollywood, x='Budget', y='Gross', color="#66a61e")
plt.xlabel("Budget (crore)")
plt.ylabel("Gross (crore)")
plt.show()
```

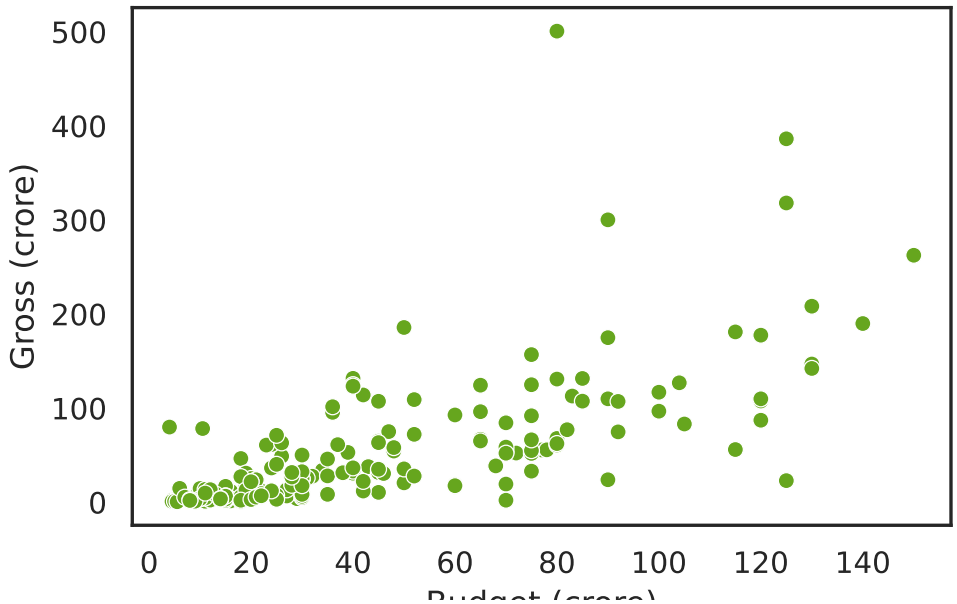
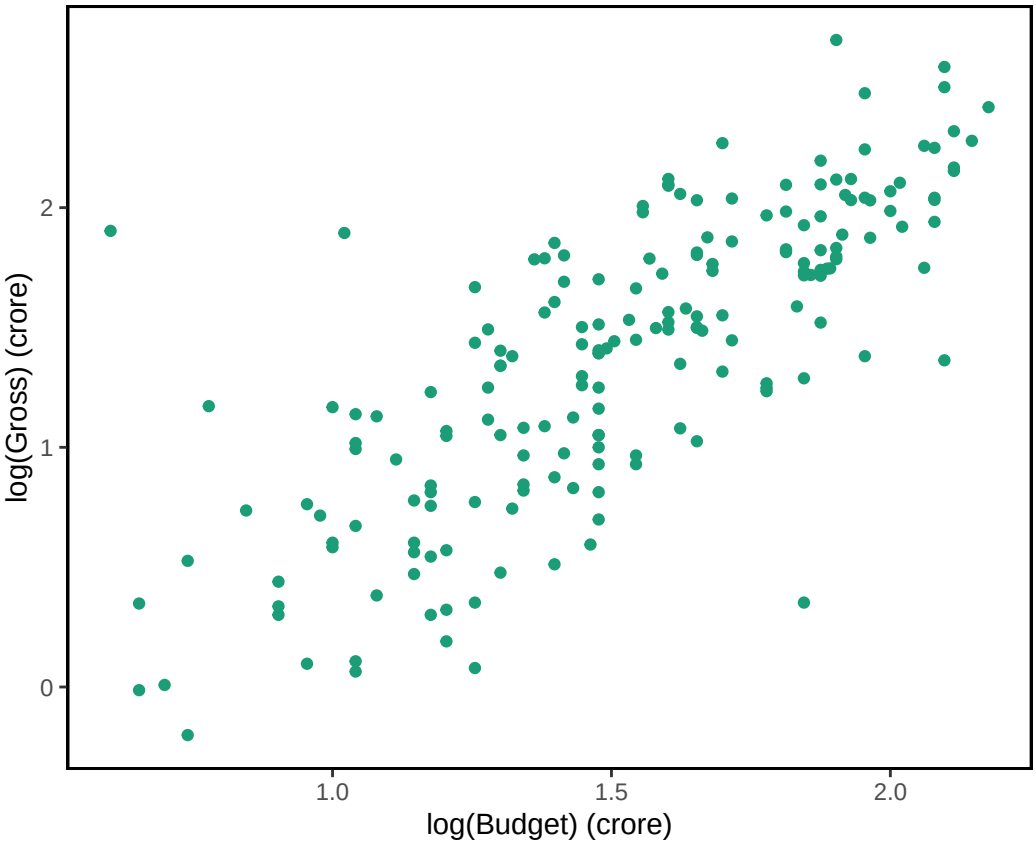


Figure 7.1: Hello

Looking at the scale of the values on both the horizontal and vertical axes, we might want to transform the data by taking logs.

7.1.5 R

```
b.plot.l <- ggplot(data = bollywood, aes(y = log10(Gross), x =
↪  log10(Budget))) +
        geom_point(col = "#1b9e77") +
        scale_x_continuous("log(Budget) (crore)") +
        scale_y_continuous("log(Gross) (crore)")
```

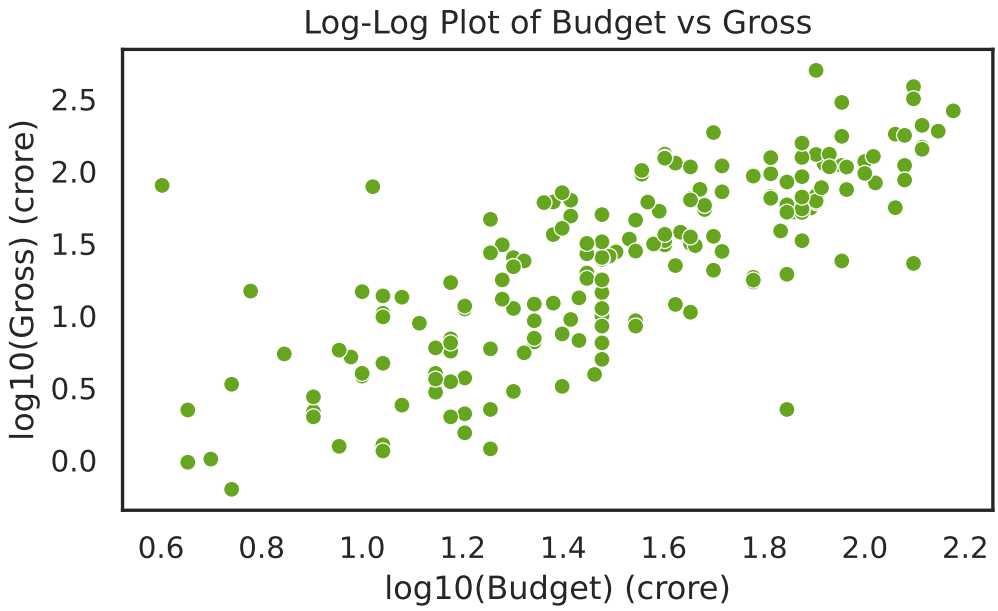


7.1.6 Python

```
import numpy as np

bollywood['log_Budget'] = np.log10(bollywood['Budget'])
bollywood['log_Gross'] = np.log10(bollywood['Gross'])

sns.set_style("white")
sns.scatterplot(data=bollywood, x='log_Budget', y='log_Gross',
 ↪  color="#66a61e")
plt.xlabel("log10(Budget) (crore)")
plt.ylabel("log10(Gross) (crore)")
plt.title("Log-Log Plot of Budget vs Gross")
plt.tight_layout()
plt.show()
```



Now let's fit a model with the $\log_{10}$ transformed gross revenue as the response (Y_i) and the $\log_{10}$ transformed budget (x_i) as the explanatory/predictor variable.

7.1.7 R

We can use the `lm()` function to fit this linear model in R.

```
bol.lm <- lm(log10(Gross) ~ log10(Budget), data = bollywood)
```

7.1.8 Python

```
import statsmodels.formula.api as smf
bol_lm = smf.ols('np.log10(Gross) ~ np.log10(Budget)',
 ↪  data=bollywood).fit()
```

The model equation in mathematical notation is

$$E(Y_i) = \mu_i = \beta_0 + \beta_1 x_i; \quad \text{where the } Y_i \text{ are independent } N(\mu_i, \sigma^2), \quad i = 1, ..., 190$$

The model fit is shown below:

7.1.9 R

```
summary(bol.lm)
```

```
Call:
lm(formula = log10(Gross) ~ log10(Budget), data = bollywood)

Residuals:
     Min       1Q   Median       3Q      Max
-1.45702 -0.24470  0.00807  0.24600  1.73413

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)   -0.62549    0.12338  -5.069 9.51e-07 ***
log10(Budget)  1.31955    0.07887  16.730  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.3921 on 188 degrees of freedom
Multiple R-squared:  0.5982,    Adjusted R-squared:  0.5961
F-statistic: 279.9 on 1 and 188 DF,  p-value: < 2.2e-16
```

7.1.10 Python

```
print(bol_lm.summary2())
```

```
                Results: Ordinary least squares
================================================================
Model:              OLS              Adj. R-squared:     0.596
Dependent Variable: np.log10(Gross)  AIC:                185.4400
Date:               2025-12-06 10:48 BIC:                191.9340
No. Observations:   190              Log-Likelihood:     -90.720
Df Model:           1                F-statistic:        279.9
Df Residuals:       188              Prob (F-statistic): 4.49e-39
R-squared:          0.598            Scale:              0.15376
----------------------------------------------------------------
                   Coef.  Std.Err.    t    P>|t|   [0.025  0.975]
----------------------------------------------------------------
Intercept         -0.6255   0.1234 -5.0695 0.0000 -0.8689 -0.3821
np.log10(Budget)   1.3195   0.0789 16.7298 0.0000  1.1640  1.4751
----------------------------------------------------------------
Omnibus:              14.570       Durbin-Watson:          1.711
Prob(Omnibus):        0.001        Jarque-Bera (JB):       38.796
Skew:                 0.181        Prob(JB):               0.000
Kurtosis:             5.184        Condition No.:          9
================================================================
```

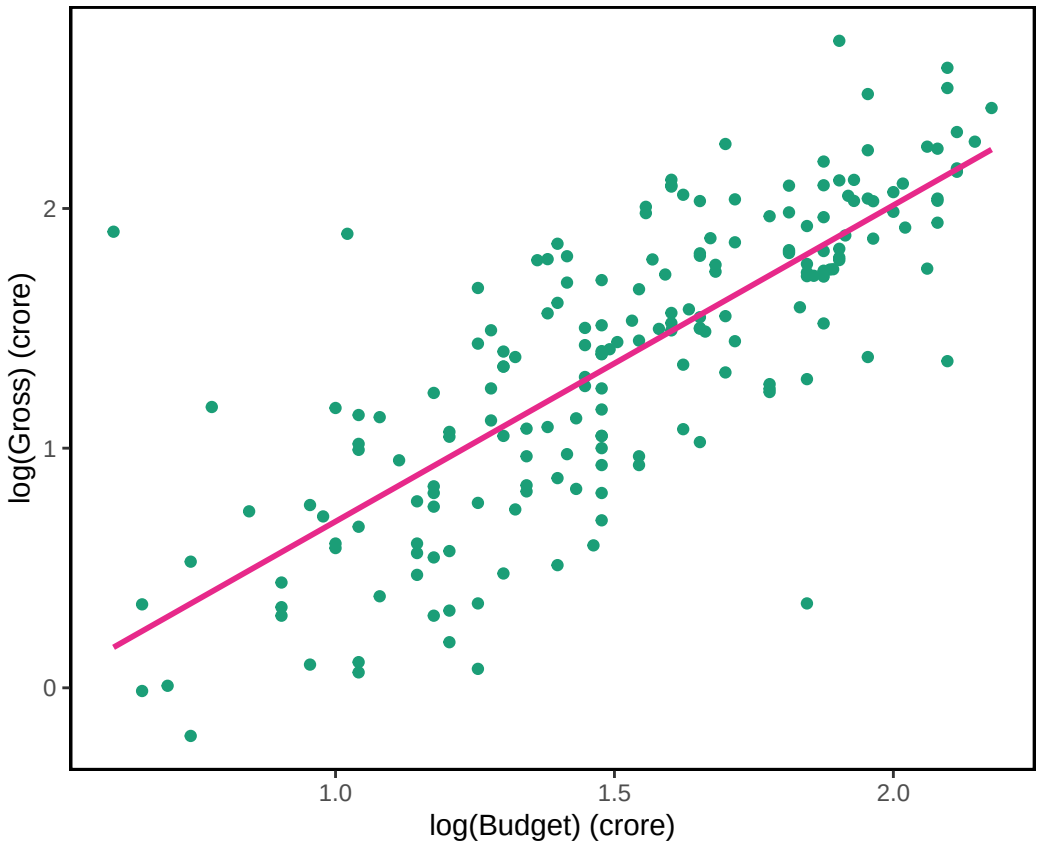
```
Notes:
[1] Standard Errors assume that the covariance matrix of the
errors is correctly specified.
```

We can visualise this regression model by plotting the data and fitted regression line:

7.1.11 R

```
b.plot.lm <- ggplot(data = bollywood, aes(y = log10(Gross), x =
 ↪  log10(Budget))) +
          geom_point(col ="#1b9e77") +
          scale_x_continuous("log(Budget) (crore)") +
          scale_y_continuous("log(Gross) (crore)") +
          geom_smooth(method = lm, colour="#e7298a", se=FALSE)
```

```
`geom_smooth()` using formula = 'y ~ x'
```

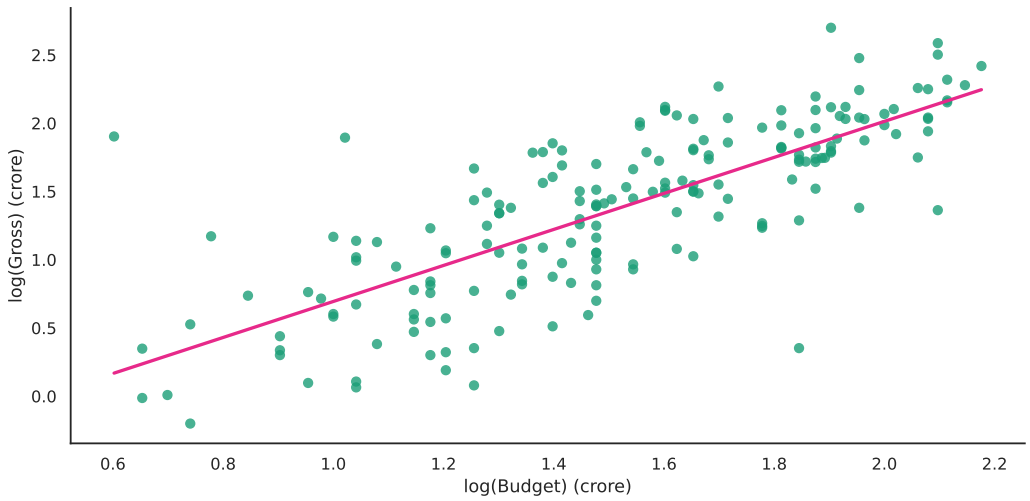


7.1.12 Python

```
sns.set_theme(style="white")
g = sns.lmplot(
    data=bollywood,
    x='log_Budget',
    y='log_Gross',
    scatter_kws={'color': '#1b9e77'},
    line_kws={'color': '#e7298a'},
    ci=None,
    aspect=2
)

g.set_axis_labels("log(Budget) (crore)", "log(Gross) (crore)");

plt.show()
```



Task 1

Using the fitted model equation, predict the gross revenue for a film with a budget of (i) 10, (ii) 50, and (iii) 100 crore.
Hint: Remember that the variables have been log-transformed.

Answer 1

For the *Bollywood box office revenue* example, we can write down the fitted model equation from the `summary(bol.lm)`:

$$\log_{10}(\text{Gross}) = -0.62549 + 1.31955 \times \log_{10}(\text{Budget})$$

We can use this equation to predict the gross revenue of a film by simply substituting the relevant budget value, and transforming the result from the log10 scale. Thus:

(i) budget = 10:

$$\log_{10}(\text{Gross}) = -0.62549 + 1.31955 \times \log_{10}(10) = 0.69406$$

$$\Rightarrow \text{Gross} = 10^{0.69406} = 4.94379$$

(ii) budget = 50:

$$\log_{10}(\text{Gross}) = -0.62549 + 1.31955 \times \log_{10}(50) = 1.616386$$

$$\text{Gross} = 10^{1.616386} = 41.34148$$

(iii) budget = 100:

$$\log_{10}(\text{Gross}) = -0.62549 + 1.31955 \times \log_{10}(100) = 2.01361$$

$$\Rightarrow \text{Gross} = 10^{2.01361} = 103.1834$$

Example 2: GPA and admission to medical school

Now let's look at a different kind of dataset, where the outcome we want to predict is not continuous-valued but binary. This is a dataset on admissions to US medical schools, which gives the admission status, GPA and standardised test scores for 55 medical school applicants from a liberal arts college in the US Midwest and it can be loaded from the `Stat2Data` package in R.

```
library(Stat2Data)
data(MedGPA)
```

The first few rows of the data are given below.

```
  Accept Acceptance Sex BCPM  GPA VR PS WS BS MCAT Apps
1      D          0   F 3.59 3.62 11  9  9  9   38    5
2      A          1   M 3.75 3.84 12 13  8 12   45    3
3      A          1   F 3.24 3.23  9 10  5  9   33   19
4      A          1   F 3.74 3.69 12 11  7 10   40    5
```

```
5       A           1   F 3.53 3.38  9 11  4 11   35   11
6       A           1   M 3.59 3.72 10  9  7 10   36    5
```

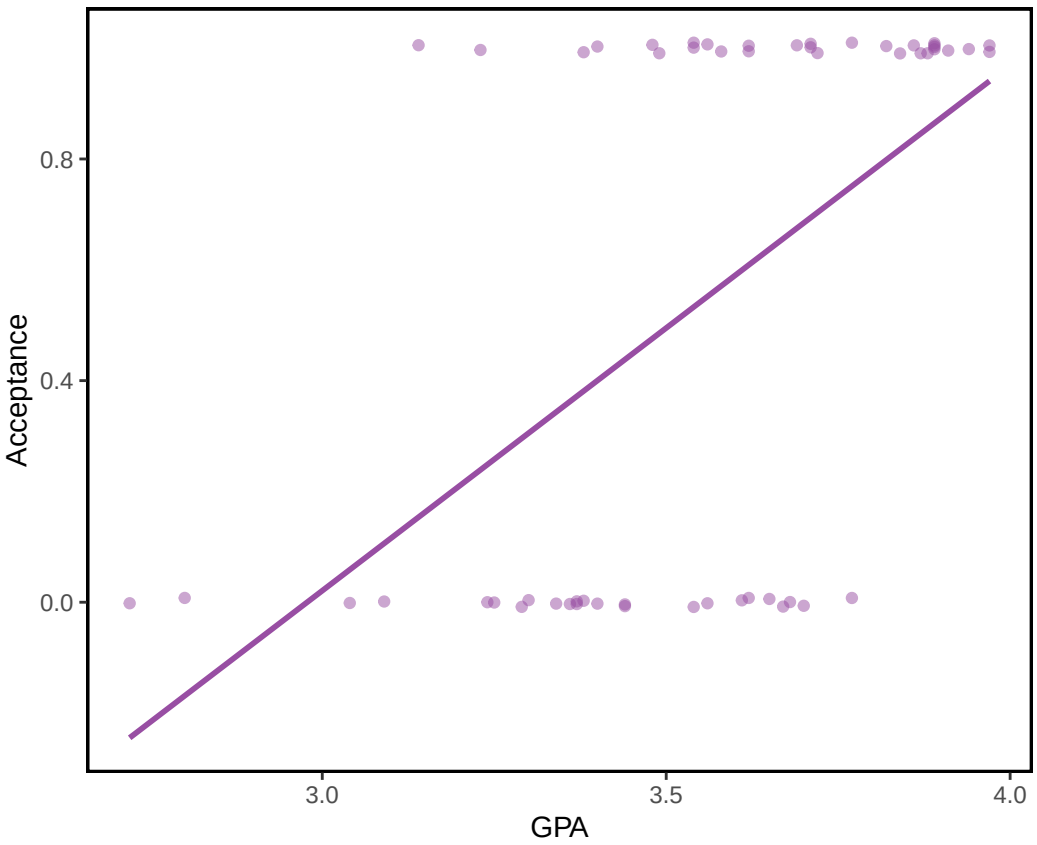
Let us look at a plot of acceptance against GPA, adding a bit of jitter to make overlapping points more visible.

```
medgpa.plot <- ggplot(data = MedGPA, aes(y = Acceptance, x = GPA)) +
               geom_jitter(width =0, height =0.01, alpha =0.5, colour
                ↪  ="#984ea3")
```

We can add the linear regression line for `Acceptance` as a function of `GPA` to the plot.

```
medgpa.plot + geom_smooth(method = "lm", se = FALSE,
                          fullrange = TRUE, colour = "#984ea3")
```

```
`geom_smooth()` using formula = 'y ~ x'
```



The R code for fitting the model and the model output is shown below.

```r
med.lm <- lm(Acceptance ~ GPA, data=MedGPA)
summary(med.lm)
```

```
Call:
lm(formula = Acceptance ~ GPA, data = MedGPA)

Residuals:
    Min      1Q  Median      3Q     Max
-0.7510 -0.3717  0.1352  0.3059  0.8464

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -2.8240     0.7226  -3.908 0.000266 ***
GPA           0.9483     0.2027   4.678 2.04e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.4267 on 53 degrees of freedom
Multiple R-squared:  0.2922,    Adjusted R-squared:  0.2788
F-statistic: 21.88 on 1 and 53 DF,  p-value: 2.043e-05
```

In mathematical notation, the value of the independent response Y_i is equal to 1 if the ith applicant is accepted and $Y_i = 0$ otherwise, where x_i refers to to the ith applicant's college GPA for $i = 1, \ldots, 55$. The normal linear model assumes that the Y_i are independent $N(\mu_i, \sigma^2)$ with $\mu_i = \beta_0 + \beta_1 x_i$, with the fitted model equation given by $\hat{\mu}_i = -2.8240 + 0.9483 x_i$.
One issue with this fit is that the predicted values of the response can take any real values, while acceptance can only take the value 0 or 1. And it is hard to argue that a variable taking only values 0 and 1 can be reasonably assumed to be normally distributed.

Key Idea: Mini introduction to generalised linear models

When faced with response variables which are clearly not normally distributed, and not even approximately normal, we take a multi-level approach.
We will introduce a **logistic regression model** to target the **probability of acceptance**. To do this we will setup our modelling a specfic way, making certain assumptions.
We begin by writing $p_i = P(Y_i = 1)$ to denote the probability of acceptance for the ith applicant. Then we assume that the Y_i are independent random variables which follow the $\text{Bin}(1, p_i)$ (or $\text{Bernoulli}(p_i)$) distribution. Our big idea is that our regression model will choose p_i to be determined by the following equation:

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_i,$$

i.e. we are still using a linear model underneath, but no longer directly fitting to the raw response variable values. The choice of function on the left will later be called a **link function**.
After some algebraic re-arrangement, this is equivalent to:

$$p_i = \frac{\exp(\beta_0 + \beta_1 x_i)}{1 + \exp(\beta_0 + \beta_1 x_i)}.$$

(*You should check you know how to do this!*)

Now we can return and apply this idea to our GPA data example.

Example 3

7.1.13 R

We can fit this model in R using the `glm()` function:

```
med.glm <- glm(Acceptance ~ GPA, data = MedGPA, family = binomial)
```

7.1.14 Python

```
import statsmodels.api as sm
import statsmodels.formula.api as smf

med_glm = smf.glm('Acceptance ~ GPA', data=MedGPA,
 ↪  family=sm.families.Binomial()).fit()
```

The argument `family=binomial` specifies that `Acceptance` follows a binomial distribution, with probability of success p_i (i.e. the probability of the ith applicant being accepted is p_i). In addition, the probability p_i is a function of x_i (i.e. the probability of the ith applicant being accepted is a function of that applicant's GPA).
The default **link function**, used is $\log\left(\frac{p_i}{1-p_i}\right)$. This function is known as the **logit function**. You can explicitly request it (unnecessarily) in R by writing `family = binomial(link="logit")` instead of `family = binomial`.
The model fit is shown below.

7.1.15 R

```
summary(med.glm)
```

```
Call:
glm(formula = Acceptance ~ GPA, family = binomial, data = MedGPA)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -19.207      5.629  -3.412 0.000644 ***
GPA            5.454      1.579   3.454 0.000553 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 75.791  on 54  degrees of freedom
Residual deviance: 56.839  on 53  degrees of freedom
AIC: 60.839

Number of Fisher Scoring iterations: 4
```

7.1.16 Python

```
print(med_glm.summary2())
```

```
                Results: Generalized linear model
=================================================================
Model:              GLM              AIC:            60.8390
Link Function:      Logit            BIC:            -155.5496
Dependent Variable: Acceptance       Log-Likelihood: -28.420
Date:               2025-12-06 10:48 LL-Null:        -37.896
No. Observations:   55               Deviance:       56.839
Df Model:           1                Pearson chi2:   51.4
Df Residuals:       53               Scale:          1.0000
Method:             IRLS
-----------------------------------------------------------------
               Coef.   Std.Err.    z    P>|z|   [0.025   0.975]
-----------------------------------------------------------------
Intercept    -19.2065   5.6292 -3.4119 0.0006 -30.2396 -8.1734
GPA            5.4542   1.5793  3.4535 0.0006   2.3588  8.5496
=================================================================
```

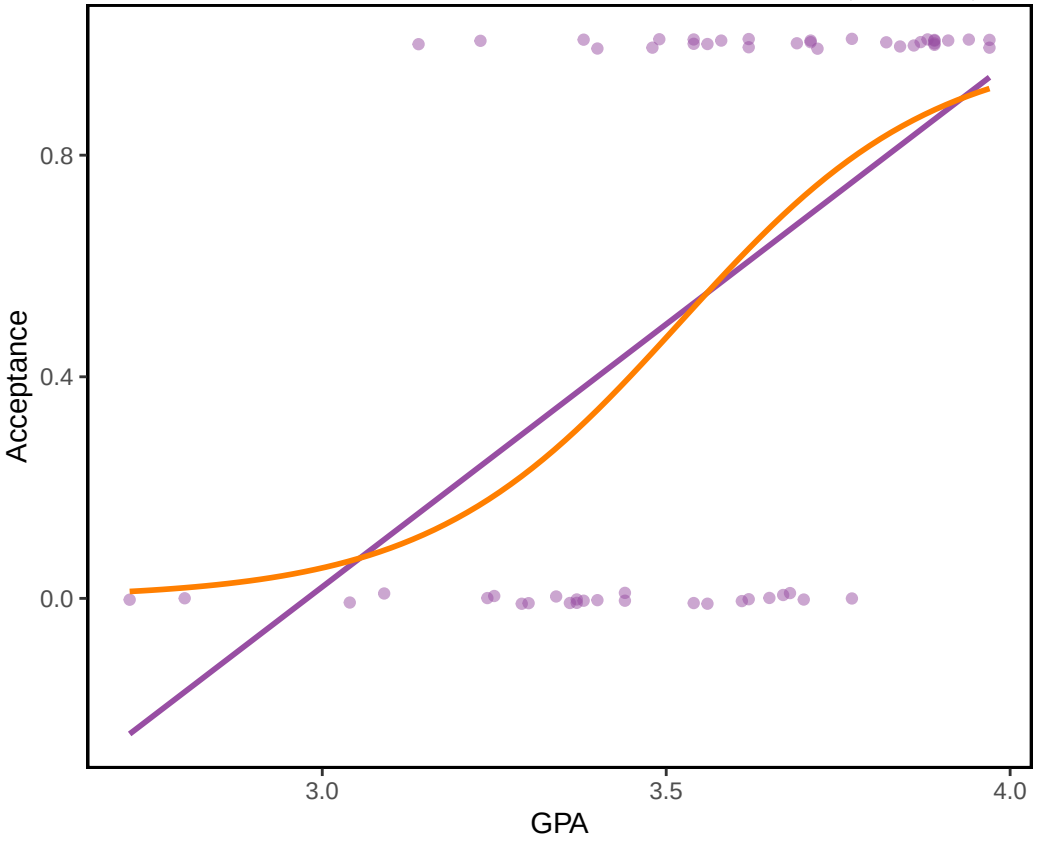
The regression equation for the fitted model is

$$\log\left(\frac{\widehat{p}_i}{1-\widehat{p}_i}\right) = -19.207 + 5.454x_i,$$

or equivalently

$$\hat{p}_i = \frac{\exp(-19.207 + 5.454x_i)}{1 + \exp(-19.207 + 5.454x_i)}.$$

The fitted curve for the probability of acceptance is shown below (in orange).



We can see that that this curve fits the data better than the linear regression line, and that it gives predicted probabilities between 0 and 1, as desired. We could add predictors to the model to improve predictive performance – we'll see more about that later.

The regression equation we have obtained allows us to predict the acceptance probability for a given GPA.

Task 2

Predict the acceptance probability for an applicant with a GPA of (i) 2.5, (ii) 3 (iii) 4. First do this "by hand" using the regression equation, then in R using the `predict()` function.

Hint: The `predict()` *function will return values on the linear predictor scale unless you specify* `type='response'` *which returns probabilities instead.*

Answer 2

In the *GPA and admission to medical school* example, we can write down the fitted model equation from the `summary(med.glm)`:

$$\log\left(\frac{p_i}{1-p_i}\right) = -19.207 + 5.454 \times \text{GPA}$$

From the fitted equation, we can obtain the acceptance probability by solving for p_i:

$$\hat{p}_i = \frac{\exp(-19.207 + 5.454 \times \text{GPA})}{1 + \exp(-19.207 + 5.454 \times \text{GPA})}$$

To predict the acceptance probability for an applicant we just need to substitute the specified GPA in the equation for $\hat{p}_i$:

(i) GPA $= 2.5$:

$$\hat{p}_i = \frac{\exp(-19.207 + 5.454 \times 2.5)}{1 + \exp(-19.207 + 5.454 \times 2.5)} \Rightarrow \hat{p}_i = 0.00378$$

(ii) GPA $= 3$:

$$\hat{p}_i = \frac{\exp(-19.207 + 5.454 \times 3)}{1 + \exp(-19.207 + 5.454 \times 3)} \Rightarrow \hat{p}_i = 0.05494$$

(iii) GPA $= 4$:

$$\hat{p}_i = \frac{\exp(-19.207 + 5.454 \times 4)}{1 + \exp(-19.207 + 5.454 \times 4)} \Rightarrow \hat{p}_i = 0.93143$$

Alternatively, we can use the `predict()` function in `R` as follows:

```
predict(med.glm, data.frame(GPA = c(2.5, 3, 4)), type = 'response')
```

```
          1           2           3
0.003791903 0.054992029 0.931512655
```

External link 1

https://goo.gl/mZHxyN

- Section 2.3 from *Mixed effects models and extensions in ecology with R -Zuur et al.* discusses the appropriateness of the assumptions of the linear model.

7.2 What do the Bollywood box office and medical school admission examples have in common?

In both cases we have independent observations and we want to predict an outcome of interest (gross revenue/acceptance) based on an explanatory variable (budget/GPA).

In both cases we have a regression equation allowing us to predict the response from a given value of the predictor. However, in one case the response is assumed to follow the normal distribution, in the other the binomial distribution.

In both cases we fit a model to the *mean* of the response:

- in the normal linear model the mean $E(Y_i) = \mu_i$ is assumed to be a linear function of x_i: $\mu_i = \beta_0 + \beta_1 x_i$, and
- in the logistic regression model the mean $\mu_i = E(Y_i) = p_i$ is modelled through the *logit link function*. That is, in logistic regression $\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_i$.

In slightly more general notation we can write $g(\mu_i) = x_i^\top \beta$ to cover both cases, where $\mu_i = E(Y_i)$ and $g(\mu_i)$ for each distribution is given in the following table.

Table 7.1: Summary table

Model	Random component	Systematic component	Link function
Normal model	$y_i \overset{\text{indep}}{\sim} N(\mu_i, \sigma^2)$, $E(Y_i) = \mu_i$	$x_i^\top \beta = \beta_0 + \beta_1 x_i$	Identity link: $g(\mu_i) = \mu_i$
Logistic regression model	$y_i \overset{\text{indep}}{\sim} Bin(1, p_i)$, $E(Y_i) = p_i$	$x_i^\top \beta = \beta_0 + \beta_1 x_i$	Logit link: $g(\mu_i) = \log\left(\frac{\mu_i}{1-\mu_i}\right) = \log\left(\frac{p_i}{1-p_i}\right)$

7.3 Exponential family of distributions

As you may already know, there is a framework for classifying distributions, inside which both the Normal (Gaussian) and Binomial distributions can be seen as specific examples from the larger **exponential family**.

So are the Poisson, the Negative Binomial, Gamma distribution, Bernoulli (covered by Binomial already) and many others.

Definition 1: Exponential family of distributions

Consider a random variable Y whose probability density function (p.d.f.) or probability mass function (p.m.f.) depends on parameter θ. The distribution belongs to the exponential family if it can be written as

$$f(y;\theta) = \exp\Big[a(y)b(\theta) + c(\theta) + d(y)\Big].$$

The term $b(\theta)$ is called the **natural parameter**. If $a(y) = y$ the distribution is said to be in **canonical form**.

Example 4: Normal distribution is a member of exponential family

Consider $Y \sim N(\theta, \sigma^2)$ with p.d.f.

$$f(y;\theta) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{1}{2\sigma^2}(y-\theta)^2\right], \quad -\infty < y < \infty.$$

If we are interested in estimating θ, the variance, σ^2, can be regarded as a nuisance parameter (which we may need to estimate at some other point).
Using the fact that $\blacksquare = \exp(\log(\blacksquare))$, we rewrite the p.d.f. as

$$f(y;\theta) = \exp\left[-\frac{y^2}{2\sigma^2} + \frac{y\theta}{\sigma^2} - \frac{\theta^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)\right]$$

we can see that this is of exponential family form with

Table 7.2: Normal distribution exponential family identification (for the case of fixed variance)

$a(y)$	$b(\theta)$	$c(\theta)$	$d(y)$
y	$\frac{\theta}{\sigma^2}$	$-\frac{\theta^2}{2\sigma^2} - \frac{1}{2}\log(2\pi\sigma^2)$	$-\frac{y^2}{2\sigma^2}$

So $a(y) = y$ meaning we can say it's in **canonical form**, and our **natural parameter** is $b(\theta) = \frac{\theta}{\sigma^2}$.

Task 3

Show that the binomial distribution $\text{Bin}(n, p)$ is a member of the exponential family.

Answer 3

Consider $Y \sim \text{Bin}(n, \theta)$ with p.m.f.

$$f(y;\theta) = \binom{n}{y}\theta^y(1-\theta)^{n-y} \text{for } y = 0, 1, \dots, n.$$

With extensive use of the same $\blacksquare = \exp(\log(\blacksquare))$ trick, plus the log laws, we rewrite the p.m.f. as

$$\begin{aligned} f(y;\theta) &= \exp\left[y\log\theta - y\log(1-\theta) + n\log(1-\theta) + \log\binom{n}{y}\right] \\ &= \exp\left[y\log\frac{\theta}{1-\theta} + n\log(1-\theta) + \log\binom{n}{y}\right] \end{aligned}$$

we see that this is a member of the exponential family in canonical form, and its natural parameter is given by: $b(\theta) = \log\left(\frac{\theta}{1-\theta}\right)$.

One of the reasons why seeing distrubutions as members of the exponential family is useful is that certain expectations and variances can be calculated for all such distributions simultaneously, using the general format.

In particular,

$$E[a(Y)] = -\frac{c'(\theta)}{b'(\theta)}$$

and

$$\text{Var}[a(Y)] = \frac{b''(\theta)c'(\theta) - c''(\theta)b'(\theta)}{[b'(\theta)]^3}.$$

These may prove useful at some point, along with further useful properties of exponential family distributions related to the **score function** (we provide the general definition first).

Definition 2: Score statistic

Writing $l(\theta; y)$ for the log-likelihood, we define the **score statistic** U, by

$$U \equiv \frac{\mathrm{d}l(\theta; y)}{\mathrm{d}\theta},$$

i.e. as the derivative of the log-likelihood $l(\theta; y)$ with respect to the parameter θ.

Recall, that for exponential family distributions the log-likelihood is always just

$$l(\theta; y) = a(y)b(\theta) + c(\theta) + d(y).$$

and so the **score statistic** is simply

$$U(\theta; y) = \frac{\mathrm{d}l(\theta; y)}{\mathrm{d}\theta} = a(y)b'(\theta) + c'(\theta)$$

The definition of the **score function** gives a new descriptive way to approach solving the maximum likelihood estimation problem. If we want to maximize the likelihood, then we typically just solve for the derivative being zero. So the **maximum likelihood estimate** $\widehat{\theta}$ for any distributions in the exponential family can be found by solving

$$U \equiv \frac{\mathrm{d}l(\theta; y)}{\mathrm{d}\theta} = 0.$$

We can also think of the score statistic, $U = a(Y)b'(\theta) + c'(\theta)$, as a random variable in its own right, which means we can calculate its expectation

$$\mathrm{E}(U) = b'(\theta)E[a(Y)] + c'(\theta) = b'(\theta)\left[-\frac{c'(\theta)}{b'(\theta)}\right] + c'(\theta) = 0,$$

which we note is always zero; and its variance

$$\mathrm{Var}(U) = [b'(\theta)^2]\mathrm{Var}[a(Y)].$$

This leads us to a very important concept in statistical inference, called **Fisher information**, which we can use to obtain standard errors for maximum likelihood estimates of GLM coefficients.

Definition 3: Fisher's information

The **Fisher Information**, denoted as $\mathcal{I}$, is given by:

$$\mathcal{I} \equiv \mathrm{Var}(U) = E(U^2) = E\left[\left(\frac{\mathrm{d}l(\theta; y)}{\mathrm{d}\theta}\right)^2\right] = E\left[\frac{\mathrm{d}^2 l(\theta; y)}{\mathrm{d}\theta^2}\right].$$

The variance of the maximum likelihood estimates tells us about the amount of *information* that an observed random variable carries about an unknown parameter in the model that is linked to a distribution.

As we will see shortly, the score and information play a key role in parameter estimation and in obtaining standard errors for the coefficient estimates of a GLM.

Having defined the exponential family of distributions, we are now ready to formally define a GLM.

External link 2

https://goo.gl/mZHxyN

- Chapter 8 from *Mixed effects models and extensions in ecology with R -Zuur et al.* contains a more in-depth discussion of the exponential family.

7.4 Generalised Linear Models

7.4.1 The definition

Definition 4: Generalised Linear Models

Let Y_i be independent responses from an exponential family distribution in canonical form and $\mu_i = E(Y_i)$, $i = 1, \dots, n$.
A **generalised linear model** (GLM) is a model of the form

$$g(\mu_i) = x_i^\top \beta \tag{7.1}$$

where β is a p-dimensional parameter vector, $x_i^\top$ is the ith row of the design matrix X, and $g()$ is a monotonic, differentiable function called the **link function**.

A GLM generalises the normal linear model by allowing:

- a response variable with a distribution other than normal, but still a member of the exponential family of distributions; and
- a non-trivial relationship between the response and the linear component of the form $g(\mu_i) = x_i^\top \beta$ where $g()$ is the **link function**.

Importantly, $g(m)$ doesn't need to be $g(m) = m$, which it was in our ordinarly linear models.

7.4.2 The random component

Suppose $Y_1, \ldots, Y_n$ are independent random variables which follow an exponential family distribution such that $f(y_i; \theta_i) = \exp[y_i b(\theta_i) + c(\theta_i) + d(y_i)]$ for $i = 1, \ldots, n$.

The joint p.d.f. (or p.m.f.) of the Y_i is

$$f(y_1, \ldots, Y_n; \theta_1, \ldots, \theta_n) = \prod_{i=1}^{n} \exp[y_i b(\theta_i) + c(\theta_i) + d(y_i)]$$

So,

$$f(y_1, \ldots, Y_n; \theta_1, \ldots, \theta_n) = \exp\left[\sum_{i=1}^{n} y_i b(\theta_i) + \sum_{i=1}^{n} c(\theta_i) + \sum_{i=1}^{n} d(y_i)\right] \tag{7.2}$$

The distribution of each Y_i is in canonical form and depends on a single parameter θ_i.

7.4.3 The systematic component

Associated with each y_i is a vector $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})^\top$ of values of p explanatory variables. The response, Y_i, depends on the explanatory variables through a linear component,

$$\eta_i = x_i^\top \beta = \beta_1 x_{i1} + \cdots + \beta_p x_{ip}$$

for $i = 1, \ldots, n$.

We will use the **link function** below to transform between the y_i and the η_i (though in some case we may use $y_i = \eta_i$), but the choice to use the values of the p covariates to determing the η is the **systematic component**.

The ith row of our design matrix, X, is $x_i^\top$; and $\beta = (\beta_1, \ldots, \beta_p)^\top$ is the parameter vector. Thus, as in linear models, the design matrix is given by

$$X = \begin{bmatrix} x_1^\top \\ \vdots \\ x_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & & \vdots \\ x_{n1} & \ldots & x_{np} \end{bmatrix}.$$

7.4.4 The link function

The parameters θ_i in Equation 7.2 are usually not of direct interest as they are not available to us. Instead, we are interested in a smaller set of parameters $(\beta_1, \ldots, \beta_p)$, and assume that Y_i depends on these through the linear predictor η_i. The link between the distribution of the Y_i and the linear predictor η_i is provided by the link function $g()$, for which $g(\mu_i) = \eta_i = x_i^\top \beta$. Here $\mu_i = E(Y_i)$ and $g()$ is a monotone, differentiable function. Although any one-to-one function could be used in principle, certain choices of link function can offer great simplification. In particular, the link function can be chosen so that the natural parameter, $b(\theta_i)$, is proportional to the linear component $\eta_i = x_i^\top \beta$. Such a link function is known as the **canonical link**. The following table shows the canonical link function for some of the most common distributions.

Distribution	Natural parameter, $b(\theta)$	Canonical link
Normal	$\frac{\theta}{\sigma^2}$	$g(\mu) = \mu$
Poisson	$\log \theta$	$g(\mu) = \log(\mu)$
Binomial	$\log\left(\frac{\theta}{1-\theta}\right)$	$g(\mu) = \log\left(\frac{\mu}{1-\mu}\right)$

External link 3

http://encore.lib.gla.ac.uk/iii/encore/record/C___Rb2939999?lang=eng

- Chapter 6 from *Extending linear models with R: generalized linear, mixed effects and nonparametric regression models - Faraway* gives an overview of GLMs and their properties.

External link 4

https://goo.gl/mZHxyN

- Sections 9.1 and 9.2 from *Mixed effects models and extensions in ecology with R -Zuur et al.* cover the general formulation of GLMs.

Let us now look at some examples of generalised linear models and their components.

Example 5: Normal linear model

Consider $E(Y_i) = \mu_i = x_i^\top \beta$ where the Y_i are independent $N(\mu_i, \sigma^2)$ for $i = 1, \ldots, n$. Here $g(\mu_i) = \mu_i$, the **identity** link. You may be more familiar with this model written as $Y = X\beta + \epsilon$ where $\epsilon = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$ with the ϵ_i independent, identically distributed $N(0, \sigma^2)$ random variables.
So the standard linear model is an example of a generalised linear model. In much the same way a square is a rectangle.

Example 6

Consider a model for whether students pass an end of semester exam at the first attempt. A Bernoulli variable Y takes the value 1 if they pass, and 0 if not. Each student's performance contains some uncertainty, and is modelled by

$$Y_i \sim \text{Bernoulli}(p_i),$$

where p_i is their probability of passing.
Student i attends a number of lectures, l_i, a number of labs m_i and spends a time t_i studying outside of classes.
We choose a model where these three covariates determine (through the **link function**) the probability of passing for each student, which as usual is the mean of Y_i.
Since we're working with a Bernoulli (or Binomial) our chosen linear model is that

$$g(p_i) = \log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 l_i + \beta_2 m_i + \beta_3 t_i.$$

So $g(p_i) = x_i^\top \beta$ with $x_i = (1, l_i, m_i, t_i)$ and $\beta = (\beta_0, \beta_1, \beta_2, \beta_3)$.

Example 7: A model for mortality rates

The number of deaths, Y, in a population in a year can be modelled by the Poisson distribution

$$f(y; \mu) = \frac{\mu^y e^{-\mu}}{y!}, \quad y = 0, 1, 2, \ldots.$$

However the population itself changes in size over time, and different groups in the population have different average death rates. *Recall that a sum of Poissons is still*

Poisson.
So we can model the expected total number of deaths per year in a population as $E(Y) = \mu$ and it can be modelled by

$$E(Y) = \mu = n\lambda^*$$

where n is the population size (in units of 100,000s) and λ^* is the death rate per 100,000 people.
To allow for different death rates by age, we can allow for the λ^* to vary for different sub-populations.
Let $Y_1, \ldots, Y_n$ be the numbers of deaths occurring in successive age groups. A popular model we assume is that $E(Y_i) = \mu_i = n_i \exp(\theta \times i)$ where $Y_i \sim \text{Poisson}(\mu_i)$ and

- $i = 1$ for the age group 30-34,
- $i = 2$ for age group 35-39,
- $\vdots$
- $i = 8$ for age group 65-69.

So in this example λ_i^* the death rate in sub-population i is being written as $e^{\theta \times i}$, and n_i denotes the number of people in that ith group (in our chosen units).
This model can be identified as a generalised linear model if we choose a log link function, i.e. $g(\mu_i) = \log \mu_i = \log n_i + \theta i$. Then we see that $g(\mu_i)$ our function of the distribution mean can be seen as a linear combination of covariates x_i and parameters β if we define them like this:

$$x_i^\top = (\log n_i, i) \text{ and } \beta = (1, \theta)^\top;$$

so then

$$g(\mu_i) = x^\top \beta.$$

Each subpopulation comes with two covariates: $\log(n_i)$ the log of its population, and it's label i. The only parameter to estimate is θ as we know exactly how $\log(n_i)$ affects $\log(\mu_i)$ already by our model assumption.
When we see this idea later we will call the term $\log n_i$ is an **offset**, we will especially see more about it when we talk about models for counts.

Task 4

Formulate the model used in the medical school admissions example as a GLM.

Answer 4

Random component: Let $Y_i = 1$ if the ith applicant is accepted to medical school and $Y_i = 0$ if not. We assume that the Y_i are independent responses from $\text{Bin}(1, p_i)$, with $E(Y_i) = p_i$ for $i = 1, \ldots, 55$.
Systematic component: $\beta_0 + \beta_1 x_i$ where x_i is the ith applicant's GPA and β_0 and β_1 are parameters to be estimated.
Link function: $g(p_i) = \log\left(\frac{p_i}{1-p_i}\right)$ (logit link)
Equation of the GLM:

$$\log\left(\frac{p_i}{1-p_i}\right) = \beta_0 + \beta_1 x_i.$$

7.5 Maximum likelihood estimation of GLM coefficients

In a generalised linear model we are interested in the parameters $\beta_1, \ldots, \beta_p$ that describe how the response depends on the explanatory variables. We use the observed $y_1, \ldots, y_n$ to maximise the log-likelihood function

$$l(\beta, y) = \sum_{i=1}^{n} y_i b(\theta_i) + \sum_{i=1}^{n} c(\theta_i) + \sum_{i=1}^{n} d(y_i) \tag{7.3}$$

obtained from equation (Equation 7.2). This depends on β through

$$\begin{aligned} \mu_i = E(Y_i) &= -\frac{c'(\theta_i)}{b'(\theta_i)}; \\ \text{Var}(Y_i) &= [b''(\theta_i)c'(\theta_i) - c''(\theta_i)b'(\theta_i)]/[b'(\theta_i)]^3; \\ g(\mu_i) = \eta_i &= x_i^\top \beta, \quad i = 1, \ldots, n. \end{aligned}$$

The maximisation procedure results in p simultaneous equations for $\hat{\beta}$, which are usually solved numerically using the **method of scoring** (also known as **Fisher's scoring method**) and an algorithm called **iteratively reweighted least squares**.

Supplement 1

Here we present in some detail how the maximum likelihood estimates of the coefficients of a GLM are obtained. Suppose that we are interested in estimating the parameter vector $(\beta_1, \ldots, \beta_p)^\top$ in a GLM. To find the maximum likelihood estimates $\hat{\beta}_j$ we need the scores (multivariate version of the score from Definition 2) expressed as functions of the β_j:

$$U_j \equiv \frac{\partial l}{\partial \beta_j} = \sum_{i=1}^{n} \left[\frac{\partial l_i}{\partial \beta_j} \right] = \sum_{i=1}^{n} \left[\frac{\partial l_i}{\partial \theta_i} \cdot \frac{\partial \theta_i}{\partial \mu_i} \cdot \frac{\partial \mu_i}{\partial \beta_j} \right] \tag{7.4}$$

For an exponential family distribution in canonical form, the components of Equation 7.4 are:

$$\frac{\partial l_i}{\partial \theta_i} = y_i b'(\theta) + c'(\theta) = b'(\theta) \left[y_i - \left(-\frac{c'(\theta)}{b'(\theta)} \right) \right] = b'(\theta)(y_i - \mu_i) \tag{7.5}$$

$$\frac{\partial \mu_i}{\partial \theta_i} = -\frac{c''(\theta_i)}{b'(\theta_i)} + \frac{c'(\theta_i) b''(\theta_i)}{[b'(\theta_i)]^2} = b'(\theta_i) \mathrm{Var}(Y_i)$$

$$\Rightarrow \frac{\partial \theta_i}{\partial \mu_i} = \frac{1}{b'(\theta_i) \mathrm{Var}(Y_i)} \tag{7.6}$$

$$\frac{\partial \mu_i}{\partial \beta_j} = \frac{\partial \mu_i}{\partial \eta_i} \cdot \frac{\partial \eta_i}{\partial \beta_j} = \frac{\partial \mu_i}{\partial \eta_i} x_{ij} = \frac{x_{ij}}{g'(\mu_i)} \tag{7.7}$$

Substituting Equation 7.5, Equation 7.6 and Equation 7.7 into Equation 7.4 the expression for the scores becomes

$$U_j = \sum_{i=1}^{n} \left[\frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} x_{ij} \frac{\partial \mu_i}{\partial \eta_i} \right] = \sum_{i=1}^{n} \left[\frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} \frac{x_{ij}}{g'(\mu_i)} \right]. \tag{7.8}$$

Note that the scores depend on β through $\mu_i = E(Y_i)$ and through $\mathrm{Var}(Y_i)$. The variance-covariance matrix of the U_j has terms $\mathcal{J}_{jk} = E(U_j U_k)$ and is known as the **(Fisher) information matrix**. This is the multivariate version of Definition 3. The elements of matrix $\mathcal{J}$ can be obtained from Equation 7.8:

$$\begin{aligned} \mathcal{J}_{jk} &= E\left\{ \sum_{i=1}^{n} \left[\frac{(y_i - \mu_i)}{\mathrm{Var}(Y_i)} x_{ij} \frac{\partial \mu_i}{\partial \eta_i} \right] \sum_{l=1}^{n} \left[\frac{(y_l - \mu_l)}{\mathrm{Var}(Y_l)} x_{lk} \frac{\partial \mu_l}{\partial \eta_l} \right] \right\} \\ &= \sum_{i=1}^{n} \frac{E[(Y_i - \mu_i)^2] x_{ij} x_{ik}}{[\mathrm{Var}(Y_i)]^2} \left(\frac{\partial \mu_i}{\partial \eta_i} \right)^2 \\ &= \sum_{i=1}^{n} \frac{x_{ij} x_{ik}}{\mathrm{Var}(Y_i)} \left(\frac{\partial \mu_i}{\partial \eta_i} \right)^2 = \sum_{i=1}^{n} \frac{x_{ij} x_{ik}}{\mathrm{Var}(Y_i) \left(g'(\mu_i) \right)^2} \end{aligned} \tag{7.9}$$

Here we have used the fact that $E[(Y_i - \mu_i)(Y_l - \mu_l)] = 0$ by the independence of the Y_i. Notice that the information matrix can be written as

$$\mathcal{J} = \mathcal{J}(\beta) = X^\top W X \tag{7.10}$$

where $X = \begin{bmatrix} x_1^\top \\ \dots \\ x_n^\top \end{bmatrix} = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}$, $W = \text{diag}(w) = \begin{bmatrix} w_1 & 0 & \dots & 0 \\ 0 & w_2 & & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \dots & 0 & w_n \end{bmatrix}$,

and

$$w_i = \frac{1}{\text{Var}(Y_i)\left(g'(\mu_i)\right)^2}, \qquad i = 1, \dots, n.$$

The information matrix $\mathcal{I}(\beta)$ depends on β through μ and through $\text{Var}(Y_i)$ for $i = 1, \dots, n$.
Equation 7.8 can be written as

$$U_j = \sum_{i=1}^{n}(y_i - \mu_i)x_{ij}w_i g'(\mu_i) = \sum_{i=1}^{n} x_{ij}w_i z_i \qquad j = 1, \dots, p \tag{7.11}$$

where $z_i = (y_i - \mu_i)g'(\mu_i)$, so the score can be expressed in vector-matrix form as

$$U(\theta) = X^\top W z. \tag{7.12}$$

Fisher's method of scoring is based on the estimating equation

$$\hat{\beta}^{(m)} = \hat{\beta}^{(m-1)} + [\mathcal{I}^{(m-1)}]^{-1}U^{(m-1)} \tag{7.13}$$

where $\hat{\beta}^{(m)}$ is the vector of estimates of $(\beta_1, \dots \beta_p)$ at the mth iteration, $[\mathcal{I}^{(m-1)}]^{-1}$ is the inverse of the information matrix with elements $\mathcal{I}_{jk}$ given by Equation 7.9, and $U^{(m-1)}$ is the vector of elements given by Equation 7.8, all evaluated at $\hat{\beta}^{(m-1)}$.
Substituting Equation 7.10 and Equation 7.12 in Equation 7.13 we obtain

$$\begin{aligned}\hat{\beta}^{(m)} &= \hat{\beta}^{(m-1)} + \left[X^\top W^{(m-1)}X\right]^{-1} X^\top W^{(m-1)} z^{(m-1)} \\ &= \left[X^\top W^{(m-1)}X\right]^{-1} X^\top W^{(m-1)} X\hat{\beta}^{(m-1)} + \left[X^\top W^{(m-1)}X\right]^{-1} X^\top W^{(m-1)} z^{(m-1)},\end{aligned}$$

this simplifies slightly to

$$\hat{\beta}^{(m)} = \left[X^\top W^{(m-1)}X\right]^{-1} X^\top W^{(m-1)}(\eta^{(m-1)} + z^{(m-1)}) \tag{7.14}$$

This is the same form as the normal equations for weighted least squares, except that it has to be solved numerically because η, z and W depend on $\hat{\beta}$.
This is why the method to obtain maximum likelihood estimators for GLMs is called **iteratively (re)weighted least squares (IRWLS)**. The procedure begins by

using an initial approximation $\hat{\beta}^{(0)}$ to obtain estimates of η, z and W. Then $\hat{\beta}^{(1)}$ is obtained from Equation 7.14 and is used to update η, z and W. The iterative process continues until the difference between successive approximations $\hat{\beta}^{(m-1)}$ and $\hat{\beta}^{(m)}$ is sufficiently small.

7.6 Inference for GLMs

We will now turn our attention to inference for GLMs, mainly through hypothesis tests and the construction of confidence intervals for the parameters of interest. For that we need some sampling distribution results.

7.6.1 Sampling distribution of the MLE

The asymptotic (large sample) distribution for $\hat{\beta}$ is

$$(\hat{\beta} - \beta)^\top \mathcal{I}(\hat{\beta})(\hat{\beta} - \beta) \overset{\text{approx}}{\sim} \chi^2(p) \tag{7.15}$$

or equivalently

$$\hat{\beta} \overset{\text{approx}}{\sim} N_p(\beta, \mathcal{I}^{-1}), \tag{7.16}$$

where $\chi^2(p)$ refers to the chi-squared distribution with p degrees of freedom and $N_p(\beta, \mathcal{I}^{-1})$ refers to a multivariate (p-dimensional) normal distribution with vector mean β and $p \times p$-dimensional variance-covariance matrix $\mathcal{I}^{-1}$.

In the one-dimensional case, what this says is that the MLE $\hat{\beta}$ is approximately normally distributed with mean β and variance $\mathcal{I}^{-1}$ (scalars).

This allows us to perform hypothesis tests and construct confidence intervals for the model parameters.

Definition 5: Wald statistic

The **Wald statistic**, also known as the **z-statistic**, for each of the model parameters $\{\beta_j\}$, $j = 1, \ldots, p$, is equal to the coefficient estimate, $\hat{\beta}_j$, over its standard error, $\text{se}(\hat{\beta}_j)$.

Under the null hypothesis $H_0 : \beta_j = 0$, and using the asymptotic normality result for the MLE, the Wald statistic is approximately distributed as standard normal. In other words, we have that

$$z = \frac{\hat{\beta}_j}{\text{se}(\hat{\beta}_j)} \overset{\text{approx}}{\sim} N(0,1).$$

This allows us to perform the **Wald test**, which compares the z-statistic to the extreme percentiles of a standard normal distribution.

Also using this asymptotic normality result, we can construct an approximate 95% confidence interval for β_j by taking

$$\hat{\beta}_j \pm 1.96\text{se}(\hat{\beta}_j).$$

Here 1.96 is the 97.5th percentile of the standard normal distribution.

Example 8: Hypothesis test and confidence interval for the GPA coefficient in the model for medical school admissions

Recall the logistic regression model for the medical school admissions data. In the output we see the MLEs of β_0 and β_1 in the `Estimate` column. These are obtained by solving the likelihood equations numerically using Fisher's scoring method (notice the `Number of Fisher Scoring iterations` information towards the end.)

```
summary(med.glm)
```

```
Call:
glm(formula = Acceptance ~ GPA, family = binomial, data = MedGPA)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -19.207      5.629  -3.412 0.000644 ***
GPA            5.454      1.579   3.454 0.000553 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 75.791  on 54  degrees of freedom
Residual deviance: 56.839  on 53  degrees of freedom
AIC: 60.839

Number of Fisher Scoring iterations: 4
```

We can also see the standard errors for $\hat{\beta}_0$ `(Intercept)` and $\hat{\beta}_1$ (GPA coefficient). These are obtained from the observed information matrix, which is computed during the iterative estimation procedure. Remember that the variance of β is estimated

by $\mathcal{J}^{-1}$, the inverse of the information matrix. The R function `vcov()`returns this estimated variance-covariance matrix of the model coefficients:

```
vcov(med.glm)
```

```
             (Intercept)       GPA
(Intercept)   31.682551 -8.873862
GPA           -8.873862  2.493774
```

The diagonal entries of this matrix are the estimated variances for $\hat{\beta}_0$ and $\hat{\beta}_1$ and the off-diagonal entries give the estimated covariance between $\hat{\beta}_0$ and $\hat{\beta}_1$. The standard errors shown in the output (`Std. Error` column) are the square roots of the estimated variances:

```
sqrt(diag(vcov(med.glm)))
```

```
(Intercept)         GPA
   5.628726    1.579169
```

The `z value` column gives the Wald statistics that test the hypotheses $H_0 : \beta_0 = 0$ and $H_0 : \beta_1 = 0$. We are almost never interested in testing whether the intercept term is zero or not, we focus instead on the coefficients of the explanatory variables in the model. So for $H_0 : \beta_1 = 0$ the z value equals 3.454 and is obtained by taking the coefficient estimate and dividing by its standard error

$$z = \frac{\hat{\beta}_1}{\text{se}(\hat{\beta}_1)} = \frac{5.454}{1.579} = 3.454.$$

Under H_0, z should approximately follow the standard normal distribution, $N(0, 1)$. The p-value shown as `Pr(>|z|)` in the output is the probability of obtaining a value as extreme as z or larger in absolute value, assuming the null hypothesis is true. A small p-value indicates that the z value is unlikely to come from a $N(0, 1)$ distribution and leads to rejecting H_0. As the p-value for our GPA coefficient is small (0.000553) we can therefore conclude that we reject our null hypothesis that the GPA coefficient is zero, ie. that the GPA term is worth keeping in the model.
A totally equivalent alternative would be to instead create the approximate 95% confidence interval for the GPA coefficient, by using $\hat{\beta}_j \pm 1.96\text{se}(\hat{\beta}_j)$:

```
5.454-1.96*1.579
```

```
[1] 2.35916
```

```
5.454+1.96*1.579
```

```
[1] 8.54884
```

The resulting interval is $(2.36, 8.55)$, and since it does not include zero, we conclude that the GPA coefficient is significant.

7.6.2 Deviance

To assess the adequacy of a model of interest, we compare it with the **saturated** (or **full**) model, which has the maximum number of parameters that can be estimated. For data with n observations, $y_1, \ldots, y_n$, each with a different parameter in $X^\top \beta$, the saturated model can be specified with n parameters.

Key Idea

Recall that for our GLM we are using Equation 7.1 to determine our best β values, and that we have one equation for each such i in $1, 2, \ldots, n$. So our $\beta_1, \beta_2, \ldots, \beta_p$ coefficients will typically be being chosen to try and match the observed means μ_i of our y_i values simultaneously across n equations. A **saturated** model includes as many coefficients as equations and thus allows for 'perfect' selection of all the μ_i values simultaneously. We still have uncertainty as our model for Y_i remains random, and note that having more than n coefficients doesn't help further, it's as if we can already essentially manually select all of our μ_i values.

If we have replicates (i.e. duplicated x rows), the maximum number of parameters in the saturated model can be less than n.

For the **saturated model** which uses m parameters (normally $m = n$) we write β_{max} and $\hat{\beta}_{\text{max}}$ for the corresponding parameter vector and its MLE.

Then let $L(\hat{\beta}_{\text{max}}; y)$ be the likelihood evaluated at $\hat{\beta}_{\text{max}}$, i.e. the likelihood for the **saturated** model; and let $L(\hat{\beta}; y)$ be the maximum value of the likelihood for our model of interest (where we used p covariates).

The **likelihood ratio**

$$\lambda = \frac{L(\hat{\beta}_{\text{max}}; y)}{L(\hat{\beta}; y)}$$

provides a measure of how well the model of interest fits compared with the full model. In practice, we often use the logarithm of the likelihood ratio (since it has more neatly

provable asymptotic properties),

$$\log \lambda = l(\hat{\beta}_{\text{max}}; y) - l(\hat{\beta}; y).$$

Large values of $\log \lambda$ suggest that the model of interest is a poor description of the data relative to the full model. Note the top of the fraction is always at least as large as the bottom so the minimum of λ is 1. How large a value of $\log \lambda$ is large? To answer this question we can obtain a critical region using the sampling distribution of $\log \lambda$. In fact, we will work with the quantity $2 \log \lambda$, which we call the **deviance**.

Definition 6: Deviance

The deviance, D, is defined as

$$D = 2 \log \lambda = 2 \left[l(\hat{\beta}_{\text{max}}; y) - l(\hat{\beta}; y) \right]$$

where $l(\hat{\beta}_{\text{max}}; y)$ is the maximised log-likelihood for the saturated model and $l(\hat{\beta}; y)$ is the maximised log-likelihood for the model of interest.

We won't go into the theory here, but this is a **likelihood-ratio test**, which you have likely studied elsewhere. Under various conditions (especially large n) this quantity D will have an approximately χ^2 distribution, with parameter equal to the difference in degrees of freedom between the two models considered.

This means that for a GLM that fits the data well, the approximate distribution of the deviance, D, is χ^2_{m-p} where m is the number of parameters in the saturated model and p is the number of parameters in the model of interest. We can therefore use this approximate distribution to evaluate whether the assumption here that our model does indeed fit the data well, was reasonable.

We are implicitly assuming that our top level **saturated** model is at least reasonable, in that our distributional assumptions for the Y_i are correct. Beyond this, the deviance allows us to see whether the restriction placed on our η_i values through limiting ourselves to just p covariates still does a good job of fitting the data.

Example 9: Deviance for a binomial model

Suppose $Y_1, \ldots, Y_n$ are independent with $Y_i \sim \text{Bin}(n_i, p_i)$ for $i = 1, \ldots, n$. The likelihood function looks like this

$$L(\beta; y) = \prod_{i=1}^{n} \binom{n_i}{y_i} p_i^{y_i} (1 - p_i)^{n_i - y_i}.$$

So the log-likelihood is

$$l(\beta; y) = \sum_{i=1}^{n} \left[y_i \log p_i - y_i \log(1-p_i) + n_i \log(1-p_i) + \log \binom{n_i}{y_i} \right] \tag{7.17}$$

For the full model all the p_i's can be separately determined, so $\beta = (p_1, \ldots, p_n)^\top$ and with algebra (which we'll skip) it can show that the optimal/maximizing choices are $\hat{p}_i = y_i/n_i$. This gives

$$l(\hat{\beta}_{\max}; y) = \sum \left[y_i \log\left(\frac{y_i}{n_i}\right) - y_i \log\left(\frac{n_i - y_i}{n_i}\right) + n_i \log\left(\frac{n_i - y_i}{n_i}\right) + \log \binom{n_i}{y_i} \right]. \tag{7.18}$$

For any model with $s < n$ parameters, the mle for p_i, which we call $\hat{p}_i$ will come from the fitted values $\hat{y}_i$ from the linear model. In the saturated model we were able to ensure $\hat{y}_i = y_i$ for all i, but now we likely cannot. So in Equation 7.17 we replace p_i with $\hat{p}_i = \frac{\hat{y}_i}{n_i}$ to obtain

$$l(\hat{\beta}; y) = \sum_{i=1}^{n} \left[y_i \log\left(\frac{\hat{y}_i}{n_i}\right) - y_i \log\left(\frac{n_i - \hat{y}_i}{n_i}\right) + n_i \log\left(\frac{n_i - \hat{y}_i}{n_i}\right) + \log \binom{n_i}{y_i} \right].$$

Thus, the deviance is

$$D = 2[l(\hat{\beta}_{\max}; y) - l(\hat{\beta}; y)] = 2 \sum_{i=1}^{n} \left[y_i \log\left(\frac{y_i}{\hat{y}_i}\right) + (n_i - y_i) \log\left(\frac{n_i - y_i}{n_i - \hat{y}_i}\right) \right]. \tag{7.19}$$

Example 10: Deviance for a normal linear model

Suppose $Y_1, \ldots, Y_n$ are independent with $Y_i \sim N(\mu_i, \sigma^2)$ and $E(Y_i) = \mu_i = X_i^\top \beta$ for $i = 1, \ldots, n$. The likelihood function is

$$L(\beta; y) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y_i - \mu_i)^2}{2\sigma^2} \right).$$

So, with a little algebra, the log-likelihood function is

$$l(\beta; y) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - \mu_i)^2 - \frac{1}{2} n \log(2\pi\sigma^2). \tag{7.20}$$

For the saturated model, with n parameters, we can essentially control all the $\mu_1, \ldots, \mu_n$. The MLEs are $\hat{\mu}_i = y_i$ and so the maximum value of the log-likelihood becomes

$$l(\hat{\beta}_{\max}; y) = -\frac{1}{2} n \log(2\pi\sigma^2). \tag{7.21}$$

For any other model with $p < n$ parameters, the MLE of β is $\hat{\beta} = (X^\top X)^{-1} X^\top y$. The corresponding maximised log-likelihood function is

$$l(\hat{\beta}; y) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i^\top \hat{\beta})^2 - \frac{1}{2} n \log(2\pi\sigma^2).$$

The deviance, $D = 2\left[l(\hat{\beta}_{\max}; y) - l(\hat{\beta}; y)\right]$ is therefore

$$D = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - x_i^\top \hat{\beta}\right)^2 = \frac{1}{\sigma^2} \sum_{i=1}^{n} \left(y_i - \hat{\mu}_i\right)^2. \tag{7.22}$$

It turns out that the (exact) distribution of D is $\chi^2(n-p)$. If the model fits the data well, then $D \sim \chi^2(n-p)$, and the expected value of D will be $n-p$, since the expectation of a chi-squared random variable is equal to its degrees of freedom. However, we are not able to use this chi-squared distribution directly, because the expression for the deviance contains the nuisance parameter σ^2. As you may remember from your linear regression courses, we end up using F tests instead.

Example 11: Deviance for a Poisson model

Let $Y_1, \ldots, Y_n$ be independent random variables with $Y_i \sim Po(\mu_i)$. Then the likelihood function is

$$L(\beta; y) = \prod_{i=1}^{n} \frac{\mu_i^{y_i} e^{-\mu_i}}{y_i!},$$

and so the log-likelihood function is

$$l(\beta; y) = \sum y_i \log \mu_i - \sum \mu_i - \sum \log(y_i!). \tag{7.23}$$

For the full model $\beta_{\max} = (\mu_1, \ldots, \mu_n)^\top$, $\hat{\mu}_i = y_i$, and the maximum value of the log-likelihood is

$$l(\hat{\beta}_{\max}; y) = \sum y_i \log y_i - \sum y_i - \sum \log(y_i!). \tag{7.24}$$

Suppose that for the model of interest with $p < n$ parameters the MLE, $\hat{\beta}$, can be used to obtain $\hat{\mu}_i$ and hence fitted values $\hat{y}_i = \hat{\mu}_i$ (because $E(Y_i) = \mu_i$). For the model of interest the maximum value of the log-likelihood is

$$l(\hat{\beta}; y) = \sum y_i \log \hat{y}_i - \sum \hat{y}_i - \sum \log(y_i!). \tag{7.25}$$

So by subtracting Equation 7.25 from Equation 7.24, the deviance is

$$D = 2[l(\hat{\beta}_{\max}; y) - l(\hat{\beta}; y)] = 2\left[\sum y_i \log\left(\frac{y_i}{\hat{y}_i}\right) - \sum(y_i - \hat{y}_i)\right] \tag{7.26}$$

For most models $\sum y_i = \sum \hat{y}_i$, so the deviance can be written as

$$D = 2\sum y_i \log\left(\frac{y_i}{\hat{y}_i}\right) = 2\sum o_i \log\left(\frac{o_i}{e_i}\right),$$

where o_i denotes the observed value and e_i the expected value of y_i. The deviance can be computed from the data and compared with the $\chi^2(n-p)$ distribution. Consider the data below for which the Y_i are assumed to be independent observations from a Poisson distribution.

y_i	2	3	6	7	8	9	10	12	15
x_i	-1	-1	0	0	0	0	1	1	1

We fit a model of the form $\mu_i = \beta_1 + \beta_2 x_i$. The fitted values are $\hat{y}_i = \hat{\beta}_1 + \hat{\beta}_2 x_i$ where $\hat{\beta}_1 = 7.45163$ and $\hat{\beta}_2 = 4.93530$. The deviance is 1.8947, which is small compared with $n - p = 7$, indicating no lack of fit.

Task 5

Verify the value 1.8947 for the deviance given in the above example.

Answer 5

We need to get the fitted values $\hat{y}_i$ for $i = 1, \ldots, 9$ and then substitute into the expression for the deviance.
Values here are displayed across two tables:

y_i	2	3	6	7	8
$\hat{y}_i$	2.51633	2.51633	7.45163	7.45163	7.45163
$y_i \log\left(\frac{y_i}{\hat{y}_i}\right)$	-0.45931	0.52743	-1.30004	-0.43766	0.56807

y_i	9	10	12	15
$\widehat{y}_i$	7.45163	12.38693	12.38693	12.38693
$y_i \log\left(\frac{y_i}{\widehat{y}_i}\right)$	1.69913	-2.14057	-0.38082	2.87115

So $\sum_{i=1}^{9} y_i \log\left(\frac{y_i}{\widehat{y}_i}\right) = 0.94738$ and $D = 2\sum_{i=1}^{9} y_i \log\left(\frac{y_i}{\widehat{y}_i}\right) = 1.89476$.
Notice that since we only have three distinct x_i values, we only have three distinct $\widehat{y}_i$ values. This is the restriction we are imposing by using a linear model (though technically we have replicates so we may have had no choice here), our model must use the same $\widehat{y}_i$ whenever $x_i = -1$, thus preventing us from matching $\widehat{y}_i$ exactly with each y_i – which would be ideal to minimize the likelihood.
In this case, since close x_i values has close y_i values, the likelihood after this restriction is not large compared with a case where we could set each $\widehat{y}_i = y_i$, so we didn't see a lack of fit using our linear model.

7.6.3 Hypothesis testing using the deviance

As we've already seen, we can test hypotheses about the p-dimensional parameter vector β by using the Wald statistic and the asymptotic distribution of the MLE.

Alternatively we can compare **nested** models M_0 and M_1 using the difference of their deviances. Let's consider two such nested models, with $q < p$ covariates:

Model $\mathbf{M_0}$ $H_0 : \beta = \beta_0 = (\beta_1, \ldots, \beta_q)^\top$.
Model $\mathbf{M_1}$ $H_1 : \beta = \beta_1 = (\beta_1, \ldots, \beta_p)^\top$.

We test H_0 against H_1 by considering

$$\begin{aligned} D_0 - D_1 &= 2\left[l(\widehat{\beta}_{\max}; y) - l(\widehat{\beta}_0; y)\right] - 2\left[l(\widehat{\beta}_{\max}; y) - l(\widehat{\beta}_1; y)\right] \\ &= 2\left[l(\widehat{\beta}_1; y) - l(\widehat{\beta}_0; y)\right] \end{aligned}$$

If both models describe the data well then $D_0 \sim \chi^2(n-q)$, $D_1 \sim \chi^2(n-p)$ and $D_0 - D_1 \sim \chi^2(p-q)$. If M_1 describes the data well but M_0 does not, then $D_0 - D_1$ will be larger than expected for a value from $\chi^2(p-q)$. So, reject H_0 if $D_0 - D_1 > \chi^2(1-\alpha; p-q)$ that is, if the difference in deviances exceeds the upper $100 \times \alpha\%$ point of the $\chi^2(p-q)$ distribution.

Example 12: Hypothesis test for the GPA coefficient in the model for medical school admissions, this time using deviances

Suppose that we want to test $H_0 : \beta_1 = 0$ in the medical school admissions example. We can perform this test using the deviances given in the output.

```
Call:
glm(formula = Acceptance ~ GPA, family = binomial, data = MedGPA)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -19.207      5.629  -3.412 0.000644 ***
GPA            5.454      1.579   3.454 0.000553 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 75.791  on 54  degrees of freedom
Residual deviance: 56.839  on 53  degrees of freedom
AIC: 60.839

Number of Fisher Scoring iterations: 4
```

Here D_0 is the null deviance, that is the deviance in the model that includes only the intercept (and no other predictors). D_1 is the residual deviance, that is the deviance of the model of interest (the model with GPA included as a predictor). Under H_0, $D_0 - D_1$ should be approximately distributed as χ^2_1. The 95th percentile of the $\chi^2(1)$ distribution, sometimes written $\chi^2(0.95; 1)$, is 3.84, and since $D_0 - D_1 = 75.791 - 56.839 = 18.952 > 3.84$ we can reject the null hypothesis. Again we conclude that GPA is a significant term in the model.