

# Quantitative Methods for Linguistic Data

*Morgan Sonderegger, Michael Wagner, Francisco Torreira*

*Draft of October 2018*



# Contents

<b>Preface</b>	<b>5</b>
<b>1 Inferential statistics: Introduction</b>	<b>9</b>
1.1 Population vs. sample . . . . .	9
1.2 Confidence intervals . . . . .	14
1.3 $t$ distribution . . . . .	16
1.4 Other reading . . . . .	19
<b>2 Hypothesis testing</b>	<b>21</b>
2.1 Hypothesis testing: High-level . . . . .	21
2.2 $z$ -scores . . . . .	22
2.3 $t$ -tests . . . . .	23
2.4 Checking normality . . . . .	32
2.5 Non-parametric tests . . . . .	36
2.6 Other reading . . . . .	39
<b>3 Linear regression</b>	<b>41</b>
3.1 Regression: General introduction . . . . .	41
3.2 Simple linear regression . . . . .	44
3.3 Multiple linear regression . . . . .	55
3.4 Linear regression assumptions . . . . .	64
3.5 Model comparison . . . . .	86
3.6 Solutions . . . . .	103
<b>4 Categorical data analysis: Preliminaries</b>	<b>107</b>
4.1 Introduction . . . . .	107
4.2 Towards logistic regression . . . . .	111
4.3 Other readings . . . . .	116
4.4 Solutions . . . . .	117
<b>5 Logistic regression</b>	<b>119</b>
5.1 Simple logistic regression . . . . .	119
5.2 Evaluating logistic regression models . . . . .	127
5.3 Multiple logistic regression . . . . .	131
5.4 Model criticism for logistic regression . . . . .	139
5.5 Other readings . . . . .	142
5.6 Solutions . . . . .	143
5.7 Appendix: Other Generalized Linear Models . . . . .	143
<b>6 Practical Regression Topics 1: Multi-level factors, contrast coding, interactions</b>	<b>145</b>
6.1 Multi-level factors: Introduction . . . . .	146
6.2 Contrast coding . . . . .	146
6.3 Assessing a multi-level factor's contribution . . . . .	159

6.4 Practice with interactions . . . . .	161
6.5 Solutions . . . . .	165
<b>7 Linear mixed models</b>	<b>167</b>
7.1 Mixed-effects models: Motivation . . . . .	168
7.2 Linear mixed models 1: One grouping factor, random intercepts . . . . .	171
7.3 Linear mixed models 2: One grouping factor, random intercepts and slopes . . . . .	180
7.4 Linear mixed models 3: Two grouping factors . . . . .	185
7.5 Evaluating LMMs . . . . .	187
7.6 Linear mixed models 4: Multiple predictors . . . . .	193
7.7 More on random slopes . . . . .	202
7.8 Random effect correlations . . . . .	206
7.9 Model criticism for linear mixed models . . . . .	212
7.10 Random slopes for factors . . . . .	216
7.11 Other readings . . . . .	220
7.12 Appendix: Extra examples . . . . .	220
7.13 Appendix: Extended exercise . . . . .	222
7.14 Solutions . . . . .	224
<b>8 Mixed-effects logistic regression</b>	<b>227</b>
8.1 Preliminaries . . . . .	229
8.2 Basics . . . . .	231
8.3 Hypothesis testing . . . . .	235
8.4 Fixed and random effects . . . . .	238
8.5 MELR Practice . . . . .	239
8.6 Model criticism for mixed-effects logistic regression . . . . .	240
8.7 Evaluation measures . . . . .	245
8.8 Miscellaneous mixed-effects regression topics . . . . .	247
8.9 Other readings . . . . .	252
8.10 Appendices . . . . .	252
8.11 Solutions . . . . .	257
<b>9 Practical regression topics 2: Ordered factors, nonlinear effects, model predictions, post-hoc tests</b>	<b>261</b>
9.1 Introduction . . . . .	262
9.2 Ordered factors . . . . .	262
9.3 Nonlinear effects . . . . .	271
9.4 Predictions from mixed models . . . . .	286
9.5 Post-hoc tests and multiple comparisons . . . . .	290
9.6 Appendix: Model predictions for individual participants . . . . .	293
9.7 Appendix: Random slopes for factors . . . . .	295
9.8 Solutions . . . . .	297
<b>10 Appendix: Datasets and packages</b>	<b>299</b>
10.1 <code>english</code> lexical decision and naming latencies . . . . .	299
10.2 Dutch <code>regularity</code> . . . . .	299
10.3 European French phrase-medial vowel devoicing . . . . .	300
10.4 North American English <code>tapping</code> . . . . .	302
10.5 <code>halfrhyme</code> : English half-rhymes . . . . .	303
10.6 <code>givenness</code> data: the Williams Effect . . . . .	303
10.7 <code>alternatives</code> . . . . .	305
10.8 VOT . . . . .	305
10.9 Transitions . . . . .	306
10.10 Packages . . . . .	307

# Preface

This e-book grew out of lecture notes for the one-semester graduate course on methods for Experimental Linguistics given in the Department of Linguistics at McGill University. “Experimental Linguistics” is a cover term sometimes used for any linguistic study based on quantitative data collected from the world, whether from laboratory experiments, speech or text corpora, online surveys, or another source. From the beginning, the course was taken by a mix of linguists and language scientists from other fields (psychology, communication sciences and disorders, computer science), so we have attempted to make these materials useful and understandable for language scientists more broadly. While this book hopefully can stand alone, readers should bear in mind that it is still fairly tailored to the McGill course, in ways we describe below.

## What is in this book?

There are two sets of “Methods” for doing experimental studies:

1. Everything that goes into getting data, such as experimental design and common methodologies (lab studies), or methods for extracting data from naturalistic data (corpus studies).
2. Methods for visualization and quantitative analysis of data that has already been collected.

Very different types of training are required for (1) depending on the type of data, while there is a common set of methods for (2) across different types of data—especially regression modeling. Thus, the original course covered only (2), and the book shares this focus.

The book focuses on different kinds of regression models widely used in practice, and background needed to understand them:

- Basic inferential statistics and hypothesis testing (Chapters 1-2)
- Linear regression (Chapter 3)
- Categorical data analysis and logistic regression (Chapters 4-5)
- Practical topics for regression modeling, such as factors, contrast coding, interpreting interactions, and post-hoc tests (Chapters 6, 9)
- Mixed-effects linear and logistic regression models (Chapters 7-8)

This book is incomplete, in the sense of missing content from course lectures on additional topics:

- Beginning: Probability; Data summarization and visualization; Exploratory Data Analysis (beginning)
- End of book: Dimensionality reduction (PCA, LDA); Classification (CART trees, random forests, SVMs)

These will hopefully be incorporated into a future version.

The course was taught for the first time by Morgan and Michael in 2013, with most lecture notes drafted by Morgan, and most datasets taken from Michael’s prosody.lab. Other examples come from Morgan and Francisco’s work, and the notes were used and updated yearly in 2013–2017 in courses taught by subsets of us. The notes eventually contained much more than can be reasonably covered in a semester, and we thought it could be useful to release them as a “book”.

All three of us are linguists working primarily on speech sounds, and work with psychologists to varying degrees, hence the examples used in these notes come largely from studies of speech sounds and/or psycholinguistics.

### What do I need to know?

We assume readers have familiarity with statistics from a basic course (e.g. up to ANOVAs), and our goal is for readers to finish the book proficient in state-of-the-art statistical models (mixed-effect regression models) for analyzing linguistic data in the mid-2010s. We focus on conceptual understanding and practical skills by doing the course “in R”, but without actually providing instruction in R. Thus, this book assumes basic familiarity with R and R programming, for which many good tutorials now exist online. In practice, graduate and advanced undergraduate students with a variety of backgrounds in statistics and programming (including “none”) have done well in the McGill course.

### Caveats

Any citable source on quantitative methods, including those written by practitioners (and not statisticians, data scientists, etc.) is liable to be treated as a reference whose text contains Truth. We would like to emphasize that this “book” remains fundamentally a set of expanded lecture notes, and should not be thought of as an authoritative reference for several reasons:

1. *None of us are statisticians.* While one of us (Morgan) has some math/statistics training, we all are essentially self-taught data analysts. If you frequently use a particular tool or framework for data analysis, we recommend (eventually) consulting a more authoritative source.
2. *We only cover a fraction of the techniques and types of data* used in quantitative studies of linguistic data—those we happen to be most familiar with and deemed most important for a one-semester course.
3. *Statistical practice is not static.* This book emphasizes practical skills (which package to use, best practices for fitting and interpreting models), for which best practices are constantly evolving.

Another consequence of this book being extended lecture notes is that the References (Chapter 11) are incomplete.

Thus, this book certainly contains errors, is incomplete and out of date, and under-cites. We welcome any feedback on errors or ways the book can be improved!

### Datasets

The book uses publicly-available datasets and is written in RMarkdown, so that readers can follow along and replicate everything in R. The `.Rmd` source files used to generate each chapter are in this repository, which readers can consult to see exactly how anything was done.

The datasets are either from our own work—in which case they are available in Open Science Framework repositories—or were already included in the R package `languageR`. Each dataset is described in an appendix. Most datasets arise from collaborative work, and we are grateful to our collaborators for their willingness to post data publicly: Mirjam Ernestus (`devoicing`); Oriana Kilbourn-Ceron and Meghan Clayards (`tapping`); Laura Harder (`halfrhyme`); Max Bane and Peter Graff (`vot`); and Seán Roberts and Stephen Levinson (`transitions`). We are also grateful to R. Harald Baayen and his co-authors for making the datasets in `languageR` available in the first place.

### Thanks

Many people deserve thanks along the way from hastily-written slides to a book. Above all, students in each of the 2013–2017 McGill classes provided feedback on what worked and what didn’t, as did McGill Linguistics students who continued to use the materials for their research—especially Hye-Young Bang, Guilherme Garcia, Oriana Kilbourn-Ceron, Donghyun Kim, Jeff Lamontagne, and James Tanner. Guilherme Garcia and Michael McAuliffe offered R tutorials and general help in some years, and provided important feedback on course materials. David Flesicher and Vanna Willerton did substantial typesetting and editing legwork to edit the notes into a coherent whole. Colleagues who saw subsets of these notes encouraged Morgan to clean them up and release publicly, including Meghan Clayards, Tim O’Donnell, Jane Stuart-Smith, and

Alan Yu. More than citations convey, we are indebted to existing materials by R. Harald Baayen, Ben Bolker, Samprit Chatterjee & Ali Hadi, Florian Jaeger, Roger Levy, Shravan Vasishth, Hadley Wickham, and especially Andrew Gelman & Jennifer Hill (*Data analysis using regression and multilevel/hierarchical models*).

– Morgan Sonderegger, Michael Wagner, Francisco Torreira

October 2018



# Chapter 1

## Inferential statistics: Introduction

### Preliminary code

This code is needed to make other code below work:

```
library(ggplot2)
library(dplyr)

## loads transitions.txt from OSF project for Roberts, Torreira, & Levinson (2015) data
transitions <- read.delim(url("https://osf.io/4v8r7/download"))
```

This chapter introduces core concepts from inferential statistics:

- Population versus sample
- The sample mean and its sampling distribution
- Confidence intervals
- The  $t$  distribution

We assume you are familiar with:

- The `tapping` dataset, described here
- Basic probability
- Some data summarization and visualization techniques, ideally using `ggplot` and `tidyverse` packages such as `dplyr`.

These were described in earlier lectures that have yet to be turned into book chapters.

### 1.1 Population vs. sample

In an experiment, we are typically interested in **population** values of a parameter. For example, for the `tapping` dataset, we want to determine:

- $p_i, p_t$ : the tapping rates when `syntax` = *intransitive* or *transitive*
- $p_i - p_t$ : the difference between the rate of tapping in the two conditions.

However, we can only take a *sample* of size  $n$  and make an inference about the population. To estimate the two quantities above, we could use:

- The proportion of the  $n$  observations which were tapped, when `syntax` = *intransitive* or *transitive*.

- The difference between these two observed proportions

This is much of what we do in statistical analysis: estimate quantities of interest, whose true values we will never actually know, based on a finite sample.

### 1.1.1 Sample → population: High level

In inferential statistics, the general procedure is to use the sample to calculate *sample statistics*. These sample statistics are used to estimate the *population values* of the parameters we actually care about. Ideally the sample statistics should be *unbiased estimators* of the population values, defined as:

- The more data we have, the closer the sample statistic gets to the population value
- In the limit of an infinite sample, the sample statistic **is** the population value.

For example:

- $\mu$  and  $\sigma$  are often used to denote the population mean and standard deviation: the values we care about, but will never actually observe.
- $\bar{x}$  and  $s$  are often used to denote the sample statistics estimating  $\mu$  and  $\sigma$ .
- If you have  $n = 1000000000$ , we expect  $\bar{x}$  to be very, very close to  $\mu$ .

### 1.1.2 Sampling distribution of the sample mean

The most basic sample statistic is the *sample mean*, which is the average of  $n$  observations:

$$\bar{x} = \frac{\sum_{i=0}^n x_i}{n}$$

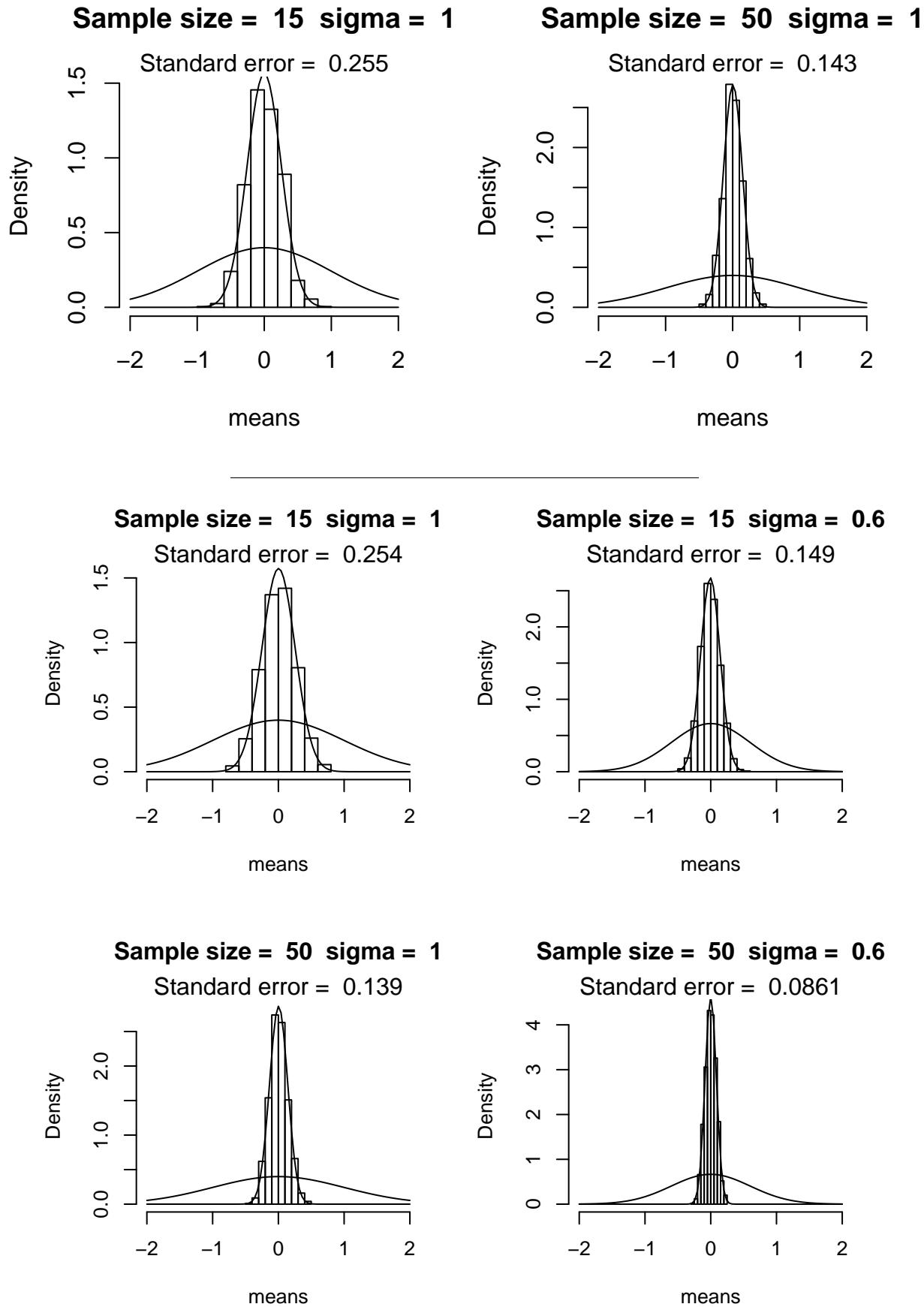
To understand how the sample mean is related to the population mean, let's explore using simulations where we know the population value.

We can do this by:

1. Draw a sample of  $n$  observations from a normal distribution with mean 0 and standard deviation  $\sigma$  (using the `rnorm` command)
2. Calculate the sample mean using this sample,  $\bar{x}$
3. Repeat (1)-(2) many times, and plot a histogram showing the distribution of sample mean values

(The code for this in the .Rmd file, but is hidden on this page.)

These plots show the histograms when the sample mean is calculated over 15 observations (left top:  $n = 15$ ) and over 50 observations (right top), varying the standard deviation  $\sigma$  between 1 and 0.5. The superimposed curve shows the normal distribution with  $\mu = 0$  and  $\sigma = 1$  or 0.5.



(These figures replicate Fig. 2.4 from Johnson (2008), using similar code to Johnson's.)

We see that the distribution of the sample mean gets narrower for larger  $n$ , and for smaller  $\sigma$ : the more observations in the sample, or the less variable the quantity is that we are measuring, the less variation there is in the mean value that we calculate based on the sample.

### Question

- Why is this, intuitively?

These simulations illustrate an important fact: **the sample mean is normally distributed** (for  $n$  observations from a  $N(\mu, \sigma)$  distribution), with mean  $\mu$ . It turns out that the standard deviation of the sample mean is  $\frac{\sigma}{\sqrt{n}}$ .

We can never observe  $\sigma$  directly (this is why we're estimating it). However, we can make an unbiased estimator:

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n-1}}$$

and use it to compute the *standard error* of the sample mean, defined as:

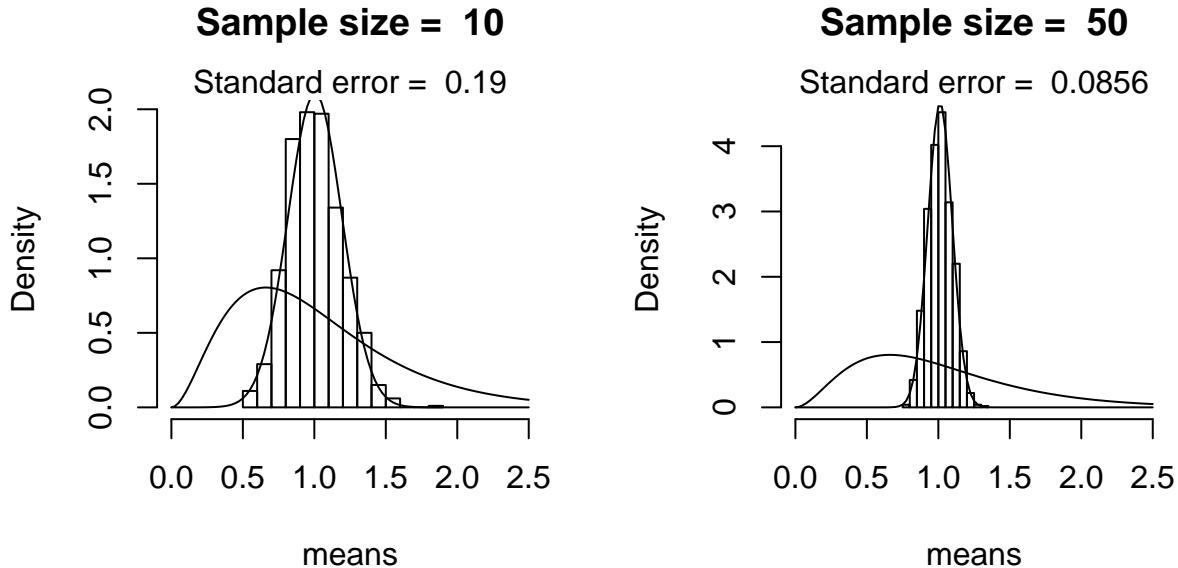
$$SE = \frac{s}{\sqrt{n}}$$

The standard error is an unbiased estimator of  $\frac{\sigma}{\sqrt{n}}$ . (And so  $\sqrt{n}SE$  is an unbiased estimator of  $\sigma$ .) The standard error quantifies how much error there is, on average (across many samples) in our estimate of the population mean using  $\bar{x}$ .

Something to note here is that **error scales as the square root of the sample size**: the SE has a  $\sqrt{n}$  in the denominator. This relationship will come up over and over in this course, and is the reason why collecting more data has diminishing returns. (Four times as much data is needed to halve the error, and so on.)

### 1.1.3 Sampling from a non-normal distribution

What happens if we take the sample mean for observations from a non-normal distribution? We can use the same simulation procedure, this time sampling from an 'F distribution'—the one used for ANOVAs. (Again, see the Rmd file for code.)



(These figures replicate Fig. 2.3 from Johnson (2008), using similar code to Johnson's.)

As you can see, the distribution of the sample mean is normal, even though the actual random variable whose mean is being estimated is not normal. This illustrates the **central limit theorem**:<sup>1</sup>

- For a large enough sample from a random variable with mean  $\mu$  and standard deviation  $\sigma$ , the (sampling) distribution of the sample mean
  - is approximately normally distributed with mean  $\mu$  and standard deviation of  $\frac{\sigma}{\sqrt{n}}$
  - regardless of the population distribution

One example of this is our estimate of  $p_t$  from  $n$  samples for the tapping data. Even though the actual variable being measured (how many times does a participant tap?) is binomially distributed, our **estimate** of  $p_t$  is normally distributed.

The central limit theorem is a very important mathematical result which allows us to apply the same tools of inferential statistics to many different kinds of data. Like gravity, it is so fundamental that it's easy to forget how much more complicated life would be without it.

## Exercises

### Exercise 1:

Consider an imaginary dataset of measurements where participants say “hi” every time they hear a beep, and we measure the latency between the beep and the beginning of “hi”.

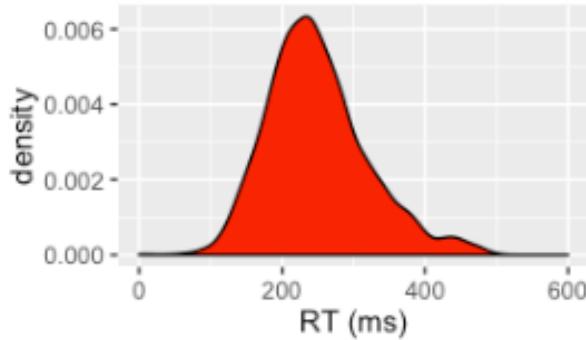
There are 100 participants, each of whom is measured once.

The sample mean and standard deviation are:

- $m = 200$  ms
- $s = 75$  ms

---

<sup>1</sup>More precisely, we are assuming the form of the CLT where the observations are independent and identically distributed. See e.g. the Wikipedia page for more details on variants of the CLT.



- What is the standard error of the sample mean?
- \* Hint:  $SE = \frac{s}{\sqrt{n}}$
- How many participants would we need in total to reduce the standard error by a factor of 2?
    - a. 200 participants
    - b. 400 participants
    - c. 1000 participants

### Exercise 2

This exercise uses the `transitions` dataset, described here.

This dataset describes approximately 20,000 transitions between conversational turns in the Switchboard Corpus of telephone calls, and can be used to analyze what factors affect the duration of transitions.

Using the `transitions` dataset, and `summarise()` (from `dplyr`), create a data frame with one column for each of:

- Mean transition duration (`dur`)
- Standard deviation of transition duration
- Number of observations
- Standard error of the mean transition duration

(Don't click unless you want to see the solution!)

```
## the transitions dataset was loaded at the beginning of this page

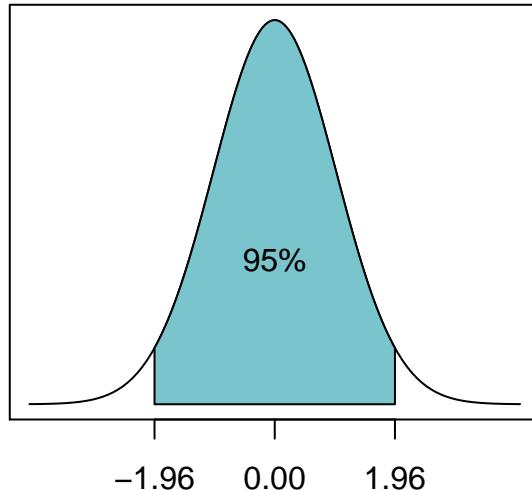
library(dplyr)
transitions %>% summarise(meanDur = mean(dur), n = n(), sd = sd(dur), se = sd/sqrt(n))

##      meanDur      n        sd        se
## 1 185.7611 21090 473.3708 3.259591
```

## 1.2 Confidence intervals

An important additional measure of uncertainty in the sample mean, related to the standard error, is the *confidence interval*. Recall that the sample mean is normally distributed with mean  $\mu$  and standard deviation of  $\frac{\sigma}{\sqrt{n}}$  (the standard error of the sample mean, or *SE*).

In addition, 95% of the area in a normal distribution lies within 1.96 standard deviations of its mean:



This motivates defining the 95% *confidence interval* of the sample mean:

- (sample mean -  $1.96 \times \text{SE}$ , sample mean +  $1.96 \times \text{SE}$ )

(You'll often see "mean  $\pm$  2 standard errors", or "mean  $\pm 2\sigma$ " used as "errorbars"; this is just because  $1.96 \approx 2$ .)

If we keep collecting samples, in the long run about 95% of these intervals would contain the true (population) mean within CI.

**Warning:** This does not mean that there is a 95% probability that the population mean lies within a given CI!

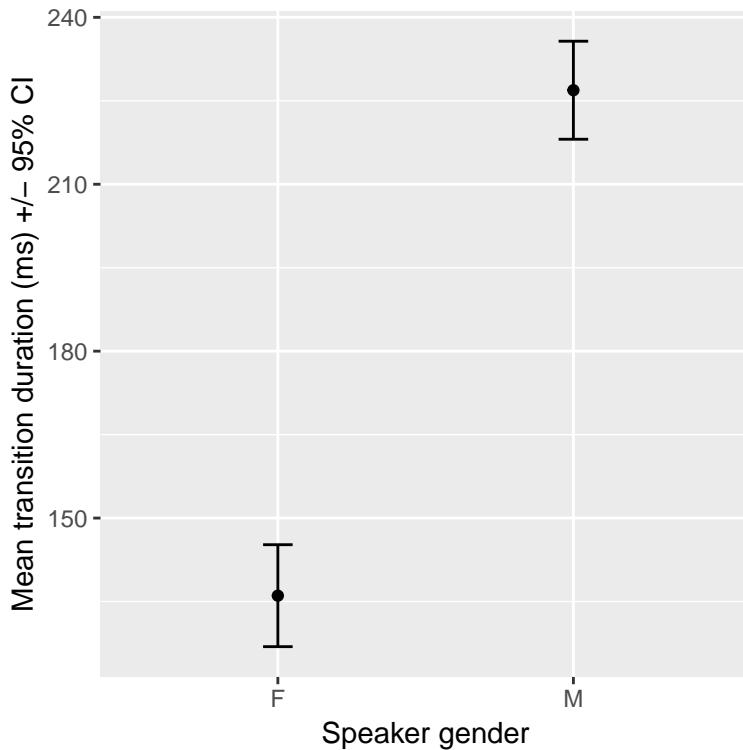
This is a common misconception of what confidence intervals mean.

### Example

Using the `transitions` dataset, let's look at the (sample) mean and 95% confidence intervals of transition duration as a function of speaker gender:

```
library(ggplot2)

transitions %>% group_by(sexB) %>%
  summarise(meanDur = mean(dur), se=sd(dur)/sqrt(n())) %>%
  ggplot(aes(sexB, meanDur)) +
  geom_point() +
  geom_errorbar(aes(ymin=meanDur-1.96*se, ymax=meanDur+1.96*se), width=0.1) +
  xlab("Speaker gender") +
  ylab("Mean transition duration (ms) +/- 95% CI")
```



**Q:** What does the errorbar here mean, exactly?

### 1.3 *t* distribution

We are usually interested in estimating not just means, but also  $\sigma$ 's. In particular, we're usually interested in  $\mu/(\sigma/\sqrt{n})$ : how far a mean is from zero, relative to the standard error of mean (a population value). For example, in the preceding plot, we are interested in how big the difference between male and female speakers is *relative to errorbar size*.

We can't actually observe  $\sigma$ , and thus  $\mu/(\sigma/\sqrt{n})$ . Instead we have to use  $s$ , which is an unbiased estimator of  $\sigma$ :

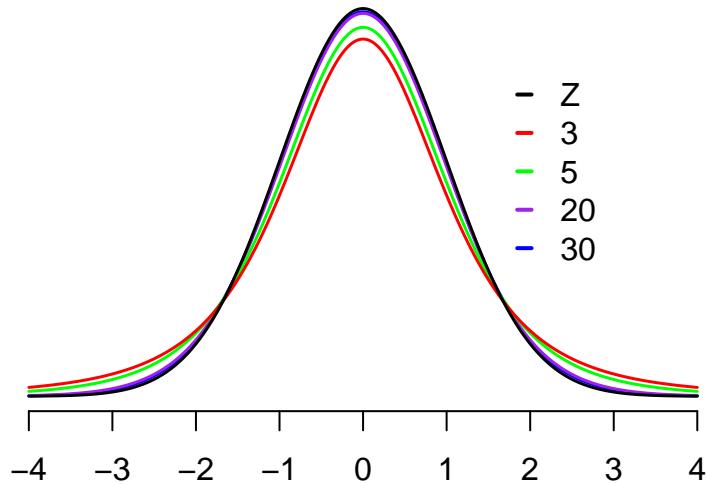
$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \mu)^2}{n-1}}$$

While  $\mu/(\sigma/\sqrt{n})$  is normally distributed, as the sample size becomes smaller,  $s$  becomes more uncertain (note the  $n-1$  in denominator), and our estimate of  $\mu/(\sigma/\sqrt{n})$  is not normally distributed.

To correct for this non-normal distribution, we use the *t distribution*: this is the distribution of  $m/(s/\sqrt{n})$ , and is sensitive to sample size. For a sample of  $n$  observations, the estimated standard error of the mean “follows a *t* distribution with  $n-1$  degrees of freedom”:

$$SE \approx \frac{s}{\sqrt{n}} \sim t_{n-1}$$

The *t* distribution looks similar to a normal distribution, but has “fatter tails” for smaller sample sizes ( $n < 30$ ):



Once  $n$  is larger than about 30, the  $t$  distribution is very similar to a normal distribution, which is the ‘Z’ line.<sup>2</sup>

### 1.3.1 $t$ -based confidence intervals

In a normal distribution, 95% of the distribution is within 1.96 standard deviations from the mean.

This code shows how many standard errors from the mean the upper bound of the 95%  $t$ -based CI is, as sample size increases:

```
alpha <- 0.05
n <- c(5, 15, 25, 50, 100, 500)

# two-sided t-distribution use 1 - alpha/2
1 - alpha/2

## [1] 0.975
cbind(n, qt(1 - alpha/2, n))

##          n
## [1,] 5 2.570582
## [2,] 15 2.131450
## [3,] 25 2.059539
## [4,] 50 2.008559
## [5,] 100 1.983972
## [6,] 500 1.964720
```

Because a  $t$  distribution is a little wider than a normal distribution, 95% confidence intervals based on the  $t$  distribution are slightly larger than those based on a normal distribution. Note that as  $n$  gets large we approach 1.96, as seen with the normal distribution. This illustrates that for large enough samples (about  $n > 30$ ), the  $t$  distribution is similar to the normal distribution.

When calculating errorbars (for plots, or to report in a paper), we usually should probably use  $t$ -based CI’s, but it’s easier to calculate CI’s based on a normal distribution (also called ‘ $z$ -based’), and the difference is minimal as long as  $n$  isn’t too small.

---

<sup>2</sup>This fact will come in handy when we cover more complex regression models, where it can be difficult to calculate the correct degrees of freedom.

### Example: Sample sizes in the `transitions` dataset

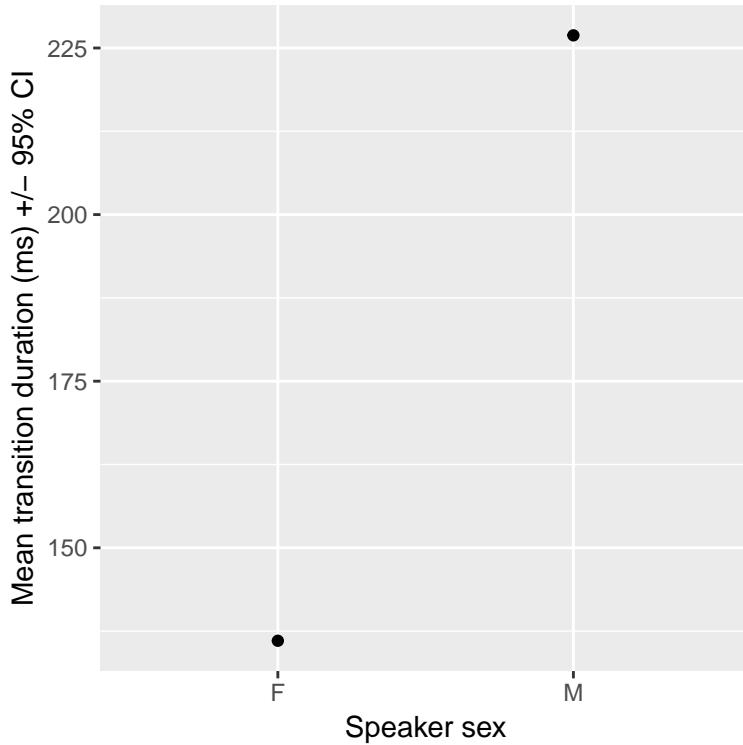
For female (F) and male (M) speakers:

```
table(transitions$sexB)

##
##      F      M
##  9550 11540
```

suppose now we which to construct a plot showing the mean transition duration (ms) .

```
transitions %>% group_by(sexB) %>%
  summarise(meanDur = mean(dur), se=n()/sd(dur)) %>%
  ggplot(aes(sexB, meanDur)) +
  geom_point() +
  xlab("Speaker sex") +
  ylab("Mean transition duration (ms) +/- 95% CI")
```

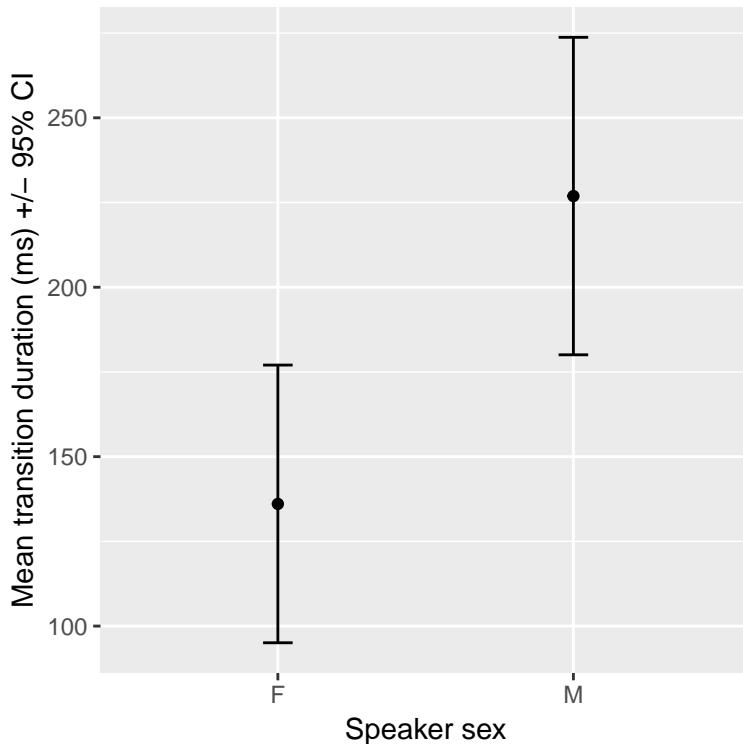


What critical value do we use if we wish to add 95% CIs in our plot using `geom_errorbar`? That is, how should we complete the following:

```
transitions %>% group_by(sexB) %>%
  summarise(meanDur = mean(dur), se=n()/sd(dur)) %>%
  ggplot(aes(sexB, meanDur)) +
  geom_point() +
  geom_errorbar(aes(ymin=meanDur-_____*se, ymax=meanDur+_____*se), width=0.1) +
  xlab("Speaker sex") +
  ylab("Mean transition duration (ms) +/- 95% CI")
```

As discussed above, we should probably use 1.96; the same as in Z-based confidence intervals:

```
transitions %>% group_by(sexB) %>%
  summarise(meanDur = mean(dur), se=n()/sd(dur)) %>%
  ggplot(aes(sexB, meanDur)) +
  geom_point() +
  geom_errorbar(aes(ymin=meanDur-1.96*se, ymax=meanDur+1.96*se), width=0.1) +
  xlab("Speaker sex") +
  ylab("Mean transition duration (ms) +/- 95% CI")
```



However, for smaller samples the confidence intervals will be wider!

## 1.4 Other reading

The concepts in this chapter are covered to varying degrees of detail, with R examples, in many sources. Some for general audiences:

- Dalgaard (2008): Ch. 3
- Crawley (2015): Ch. 1, 4, 5

and some specialized for language scientists/psychologists, such as:

- Vasishth and Nicenboim (2016)
- Navarro (2015): Ch. 10
- Vasishth (2014): Ch. 2



# Chapter 2

## Hypothesis testing

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(languageR)
library(ggplot2)
library(dplyr)

## loads transitions.txt from OSF project for Roberts, Torreira, & Levinson (2015) data
transitions <- read.delim(url("https://osf.io/4v8r7/download"))
```

In this chapter we will introduce:

- hypothesis testing, at a high level,
- $t$  tests, the most commonly-used hypothesis test, in some detail;
- other hypothesis tests, including non-parametric tests;
- and some necessary adjacent concepts, such as  $Z$ -scores.

### 2.1 Hypothesis testing: High-level

Based on a sample of  $n$  observations, we can compute the:

- *sample mean* ( $\bar{x}$ )
- and the *standard error* ( $SE = \frac{\sigma}{\sqrt{n}}$ )

concepts introduced in the previous chapter. The sample mean is normally distributed (by the central limit theorem), so  $\mu \pm 2\frac{\sigma}{\sqrt{n}}$  contains approximately 95% of the probability mass.

For concreteness, say we find that  $\bar{x}$  is 10 and  $SE = 5$ .

We don't know the population mean  $\mu$ , but suppose we have a **hypothesis** about it that is meaningful: say  $\mu = 0$ , which could mean "there is no effect" (of whatever we're measuring).

If we knew the population standard deviation  $\sigma$ , we could answer the question: "is the sample mean far enough away from 0 to be 95% sure that  $\mu \neq 0$ ?"

However, we don't know  $\sigma$ . Instead, we can approximate it by using the  $SE$  of the sample (see here). Then we can calculate the probability distribution of values the sample mean could take on, given  $\mu = 0$ , and

given some uncertainty in using  $SE$ —and use this distribution to answer our question. This procedure is called *hypothesis testing*.

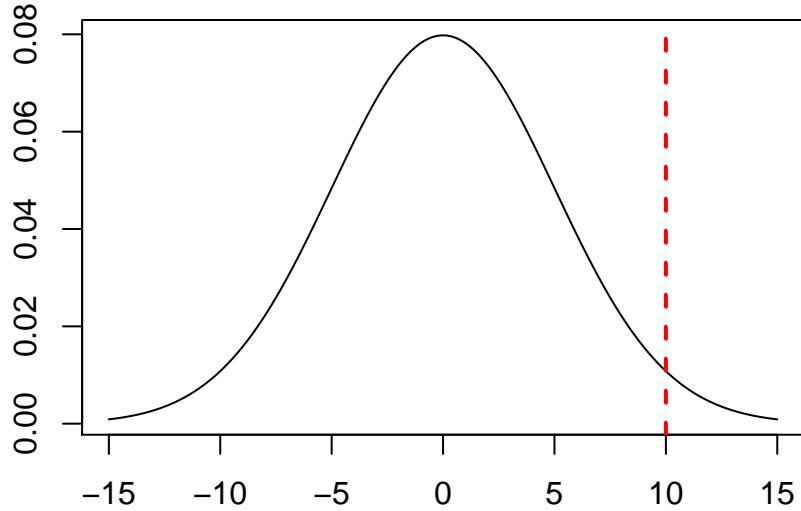
### Example

Suppose that for some sample:

- Sample mean = 10
- SE = 5
- Hypothesized  $\mu = 0$ .

In this example, we are assuming that we (magically) know the true value of the standard error (using  $\sigma$ ).

This plot shows how far out the sample mean (10) lies, for a normal distribution with mean 0 and standard deviation 5:



We would like to be able to answer: is the sample mean far enough away from  $\mu$  to be 95% sure that  $\mu$  isn't 0?

## 2.2 $z$ -scores

A  *$z$ -score* measures how many standard deviations an observation  $x_i$  is from the mean:

$$z_i = \frac{x_i - \mu}{\sigma}$$

If the observations are normally distributed, the  $z_i$  values are normally distributed as well, with mean 0 and standard deviation 1. (Note that  $\mu$  and  $\sigma$  here are the **population** parameters.)

When you draw from any random variable  $z$  with a normal distribution:

- The probability of a value with  $|z| > 1.96$  is 0.05
- Probability of a value with  $|z| > 2$  is 0.04
- Probability of a value with  $|z| > 3$  is 0.002

### Example

Returning to the question posed in our example above: “Is the sample mean far enough away from  $\mu$  to be 95% sure that  $\mu$  isn’t 0?”

The sample mean has standard deviation = SE. Thus, a  $z$  score for the sample mean we’ve computed is (sample mean - 0)/SE :

$$z = (10 - 0)/5 = 2$$

Thus, the probability of observing a sample mean at least this far from 0 is 0.04. The letter  $p$  (the *significance*) is conventionally used for “the probability of observing a value at least this big” in hypothesis testing, and written  $p = 0.04$ .

In this informal example, we assumed that we somehow knew SE—the population standard deviation—but this isn’t usually the case.

## 2.3 t-tests

### 2.3.1 Single-sample t-test: Setup

In the setting for a (single-sample)  $t$ -test, we assume that we have  $n$  observations of some normally distributed random variable, such as log-transformed reaction times (`english` dataset) or duration of vowels (`tapping` dataset). In reality the random variable has some mean and standard deviation  $\mu$  and  $\sigma$ , population values that we don’t know.

Our question is: is the mean of the variable ( $\mu$ ) different from some constant  $\mu_0$  (usually 0)?

To test this, as in the informal example above, we would like to calculate a  $z$  score:

$$z = \frac{\bar{x} - \mu_0}{\sigma/\sqrt{n}}$$

However, we typically do not know the population standard deviation  $\sigma$ , so we estimate it using the sample standard deviation:

$$s = \sqrt{\frac{\sum_{i=1}^n (x_i - \bar{x})^2}{n - 1}} \quad (2.1)$$

The  $t$ -statistic is:

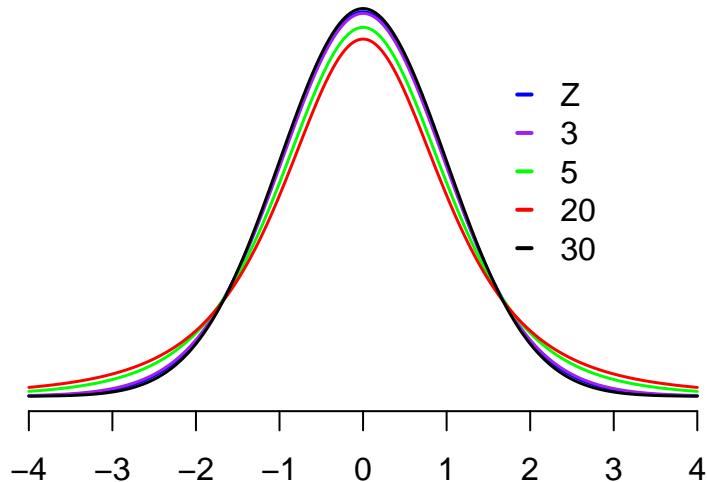
$$t = \frac{\bar{x} - \mu_0}{s/\sqrt{n}}$$

which is an estimate of the  $z$ -score that can be calculated just using the data in the sample.

The  $z$ -score and  $t$ -statistic are random variables:

- The  $z$ -score follows an  $N(0, 1)$  distribution (normal distribution with mean 0, standard deviation 1).
- The  $t$ -statistic follows the  $t$ -distribution with  $n - 1$  degrees of freedom.

This plot shows the  $t$  distribution for several different degrees of freedom values, as well as the standard normal distribution ( $Z$ ):



Note that the  $t$  distributions have “fatter tails” than the normal distribution. (Why, intuitively?)

$t$  in this case is the *test statistic*—a value we compute based on the sample, which we will then evaluate using a hypothesis test evaluating two options:

- The *null hypothesis* is  $H_0 : \mu = \mu_0$
- The *alternative hypothesis* is  $H_a : \mu \neq \mu_0$

That is: the population mean is either the “null value” (usually 0), or it isn’t.

The logic of hypothesis testing, in this case:<sup>1</sup>

- If  $H_0$  were true (“under the null hypothesis”),  $t$  would follow a  $t_{n-1}$  distribution.
- Calculate how likely we are to get a value of  $t$  at least as extreme as the value we observed, using the  $t$ -distribution. This is  $p$ , or the *p-value*.
- If  $p$  is less than the *significance level*  $\alpha$  we reject the null hypothesis.

$\alpha$  is a number between 0 and 1 parametrizing how certain we have to be to reject the null. Usually  $\alpha$  is taken to be 0.05 (95% certain), but this is just convention.

Note that “at least this extreme” refers to a *two-tailed* significance test, which is by far the most commonly-used in practice. (For example, the default for R’s `t.test` is a two-tailed test.)

It is also possible to carry out a *one-tailed* significance test, which asks: “How likely is a value at least this **positive** of the test statistic?” (Or “at least this negative.”) The Wikipedia page on one/two-tailed tests has a good visualization of the one-tailed test.

Among the reasons two-tailed significance tests are the default:

- They are “more conservative”:  $p$  values will be higher, making it less likely to reject the null hypothesis in a case where it is actually true (a “Type I error”).
- They do not require you to choose a direction (“positive” or “negative” above) to apply a one-sided test, instead remaining agnostic on the direction of any observed effect.

Classic textbooks at this point run through examples of calculating  $t$  statistics and carrying out  $t$  tests by hand, using tables to calculate  $p$  for a given sample. This is a bit of a contrived exercise in 2018, when  $t$ -tests can just be carried out automatically (even in Excel). But it remains very important to have a good **conceptual** understanding of how  $t$ -statistics and  $t$ -tests work (as well as the easier case of  $z$ -statistics and  $z$  tests), because the underlying concepts are fundamental to most statistical methods used in current practice in language sciences (e.g. any regression model).

<sup>1</sup>More precisely, *null-hypothesis significance testing* (NHST).

### 2.3.2 Hypothesis testing in general

More generally, (null-significance) hypothesis testing follows these steps:

0. Choose a significance level,  $\alpha$
1. Formulate a null hypothesis,  $H_0$
2. Formulate an alternative hypothesis,  $H_a$
3. Gather data, calculate a test statistic,  $T$
4. Determine the probability of obtaining  $T$  “or a more extreme value” under  $H_0$ , the  $p$ -value
5. If  $p \leq \alpha$ , reject  $H_0$

This procedure underlies most inferential statistics used in language sciences— $t$ -tests, ANOVAs, linear regressions, mixed-effects regressions—though the steps are not usually explicitly stated.

In particular, it is often assumed that:

- $\alpha = 0.05$
- The null hypothesis is “no difference” or “parameter is zero”
- A two-tailed test is used

unless stated otherwise. The reader of a paper or book (including this one) often must infer from context which test is being used: the type of data being analyzed and the results that are shown. For example, a comparison of two groups where  $t$ , d.f.,  $p$  are reported probably means a two-sided  $t$  test was used.<sup>2</sup>

#### Example: One-sample $t$ -test

For this example we use a new dataset: the Dutch verb regularity data (`regularity`) from the `languageR` package, described here. This dataset lists 700 Dutch irregular and regular verbs (the column `Regularity`), and includes variables which may help predict whether a verb is regular or not, including:

- What `Auxiliary` is used to form certain past/passive tenses (`hebben`, `zijn`, `zijnheb`)
- The verb’s frequency (`WrittenFrequency`)

The mean (log) frequency for `hebben` verbs (those where the auxiliary “hebben” is used) is:

```
d <- regularity %>% filter(Auxiliary == "hebben")
mean(d$WrittenFrequency)
```

```
## [1] 6.494323
```

Suppose for this example that we knew that this is the **true mean**—the population value.

For other verbs:

- `Auxiliary = zijn` (those where the auxiliary “zijn” is used): mean frequency is

```
d <- regularity %>% filter(Auxiliary == "zijn")
mean(d$WrittenFrequency)
```

```
## [1] 7.737747
```

- `Auxiliary = zijnheb` (those where either “zijn” or “hebben” can be used as auxiliaries): mean frequency is

---

<sup>2</sup>Also: with null hypothesis  $\mu_1 = \mu_2$ , not assuming equal variances in the two groups, not paired.

```
d <- regularity %>% filter(Auxiliary == "zijnheb")
mean(d$WrittenFrequency)

## [1] 6.857975
```

These mean values suggest that both *zijn* and *zijnheb* verbs have higher frequency on average than *hebben* verbs—with *zijn* verbs having the highest frequency—but we need to conduct hypothesis tests to conclude that these differences are statistically significant.

First, let's test whether *zijn* verbs have significantly different frequency from *hebben* verbs (with  $\alpha = 0.05$ ). In this case the null hypothesis is “*zijn* verbs have mean frequency = 6.494”, and we carry out a one-sample *t*-test (two-sided):

```
d <- regularity %>% filter(Auxiliary == "zijn")

t.test(d$WrittenFrequency, mu = 6.494)

##
##  One Sample t-test
##
## data: d$WrittenFrequency
## t = 2.1317, df = 19, p-value = 0.04631
## alternative hypothesis: true mean is not equal to 6.494
## 95 percent confidence interval:
##  6.516539 8.958954
## sample estimates:
## mean of x
## 7.737747
```

Which suggests we can reject the null hypothesis ( $p = 0.046$ ): *zijn* verbs have different frequencies from *hebben* verbs.<sup>3</sup>

To do the same test for *zijnheb* verbs:

```
d <- regularity %>% filter(Auxiliary == "zijnheb")

t.test(d$WrittenFrequency, mu = 6.494)

##
##  One Sample t-test
##
## data: d$WrittenFrequency
## t = 2.1505, df = 102, p-value = 0.03388
## alternative hypothesis: true mean is not equal to 6.494
## 95 percent confidence interval:
##  6.522260 7.193689
## sample estimates:
## mean of x
## 6.857975
```

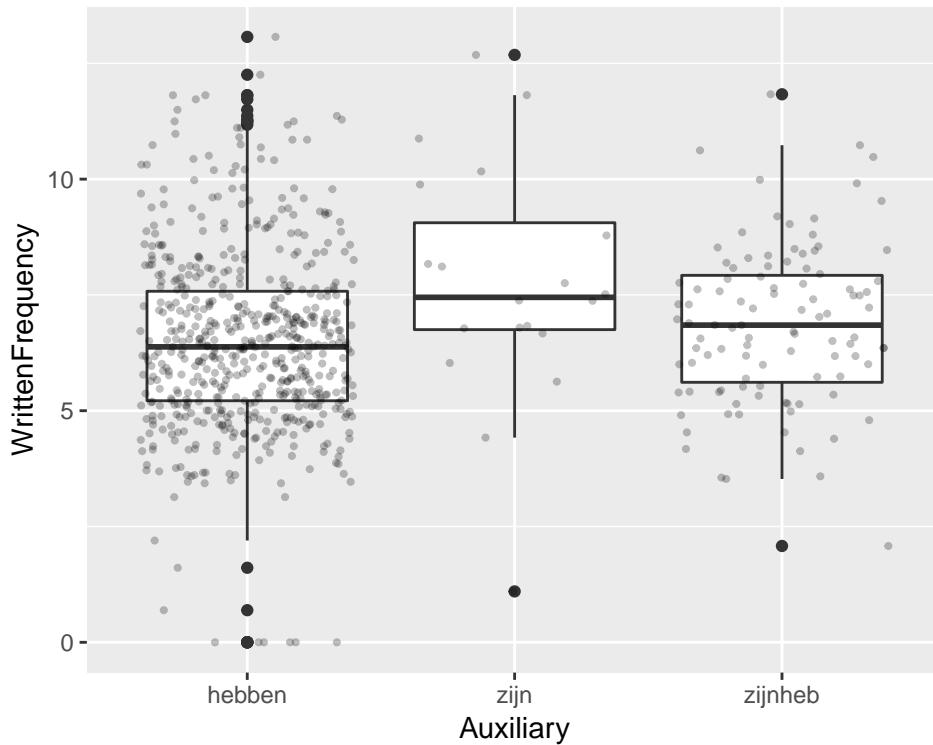
Which suggests that *zijnheb* verbs have different frequencies from *hebben* verbs ( $p = 0.034$ ).

Note that the  $p$  value is **higher** for the *zijn/hebben* comparison (less confident we can reject the null hypothesis) than for the *zijn/zijnheb* comparison, even though the difference in **sample means** is larger for *zijn/hebben*—as shown in the boxplot below.

---

<sup>3</sup>Note that we cannot technically conclude that *zijn* verbs have **higher** frequencies, because we carried out a two-sided test.

```
ggplot(regularity, aes(x=Auxiliary, y=WrittenFrequency)) +
  geom_boxplot() +
  geom_point(position="jitter", size=0.75, alpha=0.25)
```



**Question:**

Why is this?

### 2.3.3 Two-sample t-test

The one-sample *t*-test is mostly used as part of more complex statistical procedures (such as linear regression). More commonly used on its own is the *two-sample t-test*, to examine the difference in means between two groups.

As an example, for the `regularity` data, let's divide the data up into two “types”, by whether the `Auxiliary` is *hebben* or not:<sup>4</sup>

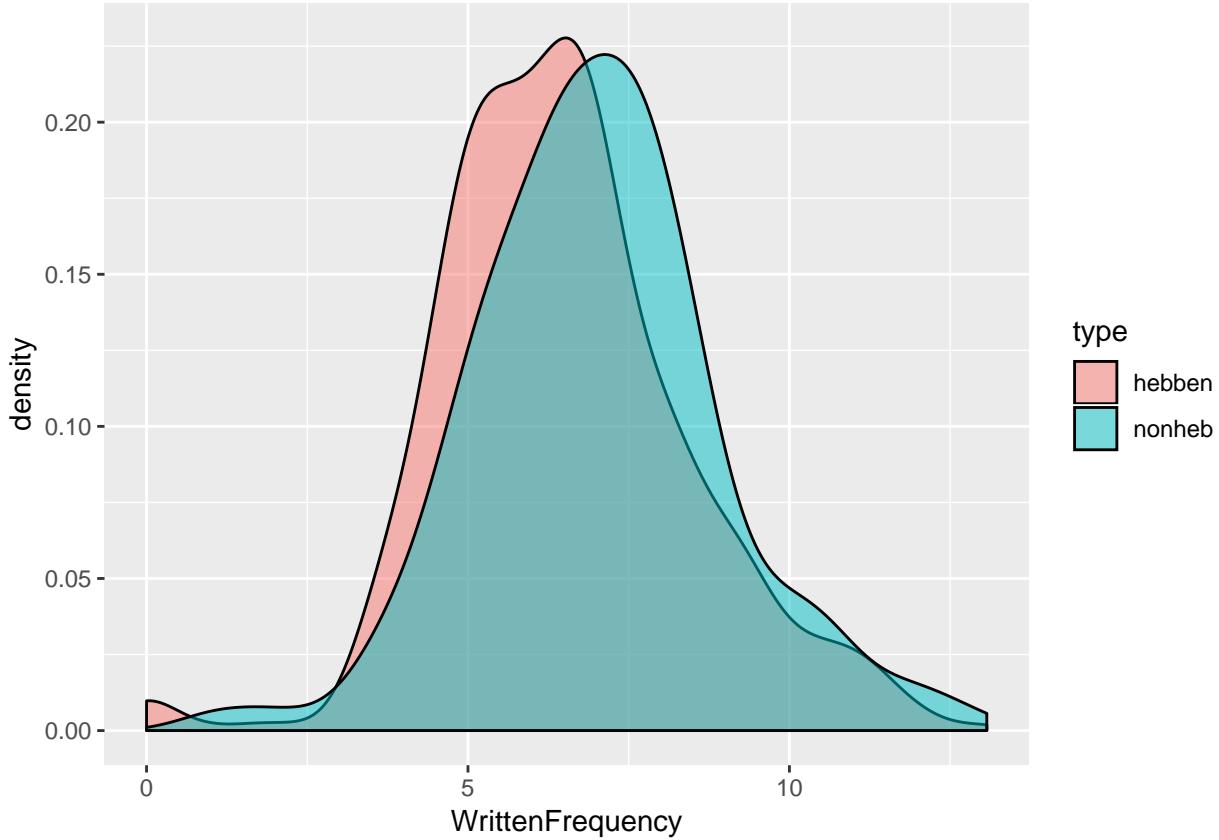
```
regularity <- mutate(regularity, type=factor(ifelse(Auxiliary %in% c("zijn", "zijnheb"), "nonheb", "hebbo
```

The two verb classes seem to have similar frequencies:

```
ggplot(aes(x=WrittenFrequency), data=regularity) + geom_density(aes(fill=type), alpha=0.5)
```

---

<sup>4</sup>This division makes sense because the auxiliary “*hebben*” can be thought of as the default case. See here (less technical) or “Non-finite forms” here (more technical) if interested.



but *haben* verbs may have lower frequencies on average. A two-sample *t*-test lets us test whether this difference is significant.

### 2.3.3.1 Setup

The two-sample *t*-test assumes two **independent** samples, each drawn from a normal distribution with the same standard deviation:

- Sample 1:  $n_1$  observations  $(x_1^1, \dots, x_1^{n_1})$  from  $N(\mu_1, \sigma)$
- Sample 2:  $n_2$  observations  $(x_2^1, \dots, x_2^{n_2})$  from  $N(\mu_2, \sigma)$

(This part can be skipped if math is not useful for your understanding.)

The sample means are the averages of the observations in each sample:

$$\bar{x}_1 = \frac{\sum_{i=1}^{n_1} x_1^i}{n_1}, \quad \bar{x}_2 = \frac{\sum_{i=1}^{n_2} x_2^i}{n_2}$$

and the difference in sample means is normally distributed (because  $\bar{x}_1$  and  $\bar{x}_2$  are). Recall that we want to calculate  $Z$ : the difference in sample means divided by its standard deviation. If we knew the standard deviation  $\sigma$ , the variance of the difference in sample means would be  $\sigma^2/n_1 + \sigma^2/n_2$ , and so

$$Z = \frac{\bar{x}_1 - \bar{x}_2}{\sigma \sqrt{(1/n_1 + 1/n_2)}}$$

As for the one-sample *t*-test, we don't know  $\sigma$  and must estimate it from the data in the two samples. The sample standard deviations  $s_1$  and  $s_2$  are calculated as in Equation (2.1). Since we are assuming the two

samples have equal variance, we calculate a single sample standard deviation  $s$ :

$$s = \sqrt{\frac{(n_1 - 1)s_1^2 + (n_2 - 1)s_2^2}{n_1 + n_2 - 2}}$$

That is, we are estimating the variance  $s^2$  as the average of the two sample variances  $s_1^2$  and  $s_2^2$ , weighted by their degrees of freedom—this turns out to be an unbiased estimator of  $\sigma$ .

The test statistic  $t$  corresponding to the difference in sample means is then:

$$t = \frac{\bar{x}_1 - \bar{x}_2}{s\sqrt{(1/n_1 + 1/n_2)}} \quad (2.2)$$

which turns out to follow a  $t$  distribution with  $n_1 + n_2 - 2$  degrees of freedom.

To carry out the two-sample hypothesis test:

- Null hypothesis: \*  $H_0 : \mu_1 - \mu_2 = 0$  (equal means)
- $H_a : \mu_1 - \mu_2 \neq 0$  (means are not equal)
- Choose  $\alpha$
- Compute  $t$
- Compute  $p$ -value by seeing how far out  $t$  is on a  $t_{n_1+n_2-2}$  distribution.

(And also calculate  $(1-\alpha)\%$  confidence intervals for the difference in means, if desired.)

As an aside: it is worth noting that there is more uncertainty when estimating the difference between the means of the two samples ( $\bar{x}_1$  and  $\bar{x}_2$ )—the denominator of Equation (2.2)—than when estimating either of these values alone.<sup>5</sup> This illustrates a very general fact: **the more things are being estimated to determine a quantity  $X$ , the more uncertainty there is in our estimate of  $X$ .**

### Example

To test whether the difference seen in the plot above is significant, we carry out a two-sample  $t$ -test of whether *hebben* and *non-hebben* verbs differ in frequency, assuming equal variances in each group:

```
t.test(regularity$WrittenFrequency ~ regularity$type, var.equal = TRUE)
```

```
## 
## Two Sample t-test
##
## data: regularity$WrittenFrequency by regularity$type
## t = -2.6406, df = 698, p-value = 0.008462
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.8834591 -0.1299488
## sample estimates:
## mean in group hebben mean in group nonheb
##           6.494323          7.001027
```

suggesting that *hebben* and *zijn/zijnheb* verbs have significantly different frequencies.

---

<sup>5</sup>That is, the standard error in the denominator is larger than  $s/\sqrt{n_1}$  or  $s/\sqrt{n_2}$

### 2.3.4 Unequal variances: Welch *t*-test

When testing whether two samples differ there is not usually any reason to think that the samples have equal variances. In practice one should assume by default that the samples have unequal variances—as R does in the `t.test` function.

This default is the *Welch t-test*, which corrects the degrees of freedom and SE calculations for unequal variances. (The formulas get complicated, which is why it is useful for exposition to assume equal variances.) To run the same *t*-test as above without assuming equal variances:

```
t.test(regularity$WrittenFrequency ~ regularity$type)
```

```
## 
## Welch Two Sample t-test
##
## data: regularity$WrittenFrequency by regularity$type
## t = -2.6688, df = 179.82, p-value = 0.008309
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## -0.8813494 -0.1320584
## sample estimates:
## mean in group hebben mean in group nonheb
##           6.494323          7.001027
```

(Note the Welch Two Sample t-test message prior to the results.)

This variant of the *t*-test—two-sample, unequal variances—is by far the most commonly used in practice, and is often just called “a *t*-test” in papers.

### 2.3.5 Assumptions behind *t*-test

There are two key assumptions made by *t*-tests (here we assume two-sample):

- **Normality:** Both groups are normally distributed
- **Independence:** Each observation is independently sampled

It is important to be aware of these assumptions—especially the second one, which is not the case for many very basic questions addressable with linguistic datasets.<sup>6</sup>

It is less clear how important the normality assumption is. If your samples are clearly not normally distributed—either by visual inspection, or for a priori reasons (e.g. a rating task)—can’t be normally distributed because of upper and lower bounds), you should use a non-parametric test instead (like Wilcoxon, discussed below). A common case in practice where *t* tests are suspect is when there are clear outliers in one or both samples. However, *t*-tests are relatively “robust”<sup>7</sup> to non-normally-distributed samples, and **many** papers especially in older literature report *t*-tests on (probably) non-normal observation distributions, in part because carrying out non-parametric tests was computationally difficult before the 1990s. One shouldn’t immediately discount the results of *t*-tests for the many types of linguistic data where normality is unlikely (like Likert scores, reaction times, durations of sounds/words), but do view these results critically—especially where significances are near the  $\alpha$  cutoff.

---

<sup>6</sup>Ex: do observations from participant Group A and participant Group B differ? In general participants are measured more than once, violating the independence assumption for *t*-tests.

<sup>7</sup>see e.g. discussion [here](<https://stats.stackexchange.com/questions/121852/how-to-choose-between-t-test-or-non-parametric-test-e-g-wilcoxon>), or Bland (2015) p. 168.

### 2.3.6 Paired *t*-test

A useful variant of the *t*-test, which deals with a particular violation of the independence assumption, is for *paired* data, where the two samples consist of pairs of observations (A and B), and the difference between A and B is of interest.

For example:

- For the **tapping** data: “tapping rate in intransitive items” and “tapping rate in transitive items” for the *i*th participant.
  - Can be used to ask: “does tapping rate differ by-participant between transitive and intransitive items?”
- For the **transitions** data:
  - “Mean transition duration during the first minute”, and “mean transition duration after the first minute”, for each conversation
  - Can be used to ask: “do floor transition times differ between the first minute of a conversation and the rest of the conversation?””

The *paired t-test*, essentially a one-sample *t*-test on pairwise differences between observations, is appropriate for this type of data.

#### Example

For the **transitions** example just given, we first set up a dataframe with three columns:

- Conversation id
- Mean transition duration during the first minute (= 60000 msec)
- Mean transition duration after the first minute

```
d <- transitions %>%
  group_by(file) %>%
  summarise(meanDurFirst = mean(dur[time < 60000]),
            meanDurAfter = mean(dur[time > 60000]))
d

## # A tibble: 349 x 3
##   file      meanDurFirst meanDurAfter
##   <fct>        <dbl>       <dbl>
## 1 sw3154.eaf    231.       245.
## 2 sw3155.eaf     19        153.
## 3 sw3156.eaf    131       -9.11
## 4 sw3159.eaf    166.      -28.3
## 5 sw3161.eaf   -352.       34.3
## 6 sw3168.eaf     31.7      120.
## 7 sw3169.eaf     NaN       202.
## 8 sw3171.eaf    -76.8     -120.
## 9 sw3174.eaf    378.       75.7
## 10 sw3182.eaf   330.       139.
## # ... with 339 more rows
```

Test whether `d$meanDurFirst - d$meanDurAfter` is statistically significantly different from 0 ( $\alpha = 0.05$ ):

```
t.test(d$meanDurFirst, d$meanDurAfter, paired=T)
```

```
##
## Paired t-test
##
## data: d$meanDurFirst and d$meanDurAfter
## t = 5.9893, df = 346, p-value = 5.283e-09
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## 42.10084 83.27295
## sample estimates:
## mean of the differences
## 62.6869
```

Thus, there is a significant difference ( $p < 0.0001$ ).

Is the difference positive or negative?

### 2.3.7 Reporting a hypothesis test

When reporting a hypothesis test's results in a paper, you include at least

- Test statistic value
- Any important parameter values (for  $t$ -test: degrees of freedom)
- The  $p$ -value

which are all needed to interpret your result.

It is also recommended to include appropriate descriptive statistics (for two-sample  $t$ -test: mean and SD of each group) when possible; this gives a sense of effect size, which is arguably as important as significance. Even better is to include a plot showing the relevant aspects of the data (like the histogram above, for a two-sample  $t$ -test), but this is not always possible when writing up results due to space limitations.

The most commonly-used format for reporting statistics is “APA style”, described various places online (e.g. here) without purchasing the APA style guide.

For example, one could report the two-sample  $t$ -test result above as:

Verb frequencies differ significantly by type ( $t(180) = -2.67, p = 0.0083$ ), with non-hebben verbs having higher frequency (mean=7.0, D=1.9) than hebben verbs (mean=6.5, SD=1.9).

Note how including the second clause, instead of the (more common format):

Verb frequencies differ significantly by type ( $t(180) = -2.67, p = 0.0083$ ), with non-hebben verbs having higher frequency than hebben verbs.

describes a crucial aspect of the result: even though the two verb groups differ significantly in frequency, the size of this between-group difference is very small in comparison to the **within-group** variation in frequency.

## 2.4 Checking normality

It is often useful to check whether a sample is normally distributed, for example when checking the normality assumption for  $t$ -tests.

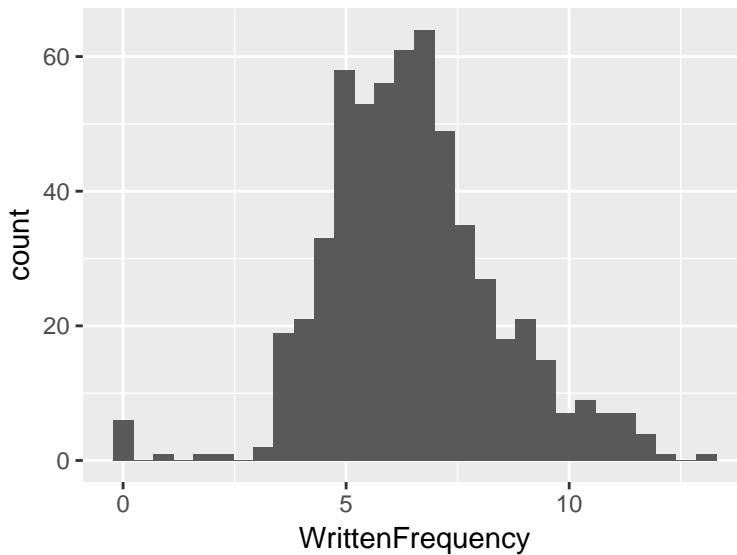
As a running example, we use the frequencies of verbs which take the *hebben* auxiliary in the **regularity** dataset.

### 2.4.1 Visual methods

The simplest method is to simply examine a histogram and eyeball whether it looks like a bell curve (= normally distributed):

```
hebben <- regularity %>% filter(Auxiliary=="hebben")
ggplot(hebben, aes(WrittenFrequency)) +
  geom_histogram()
```

## `stat\_bin()` using `bins = 30`. Pick better value with `binwidth`.



The histogram in this case is clearly non-symmetric, and thus not normal. But perhaps it is close to normal? Or does the right tail decay too slowly? Histograms are fine for detecting gross deviations from normality, but not more subtle ones.

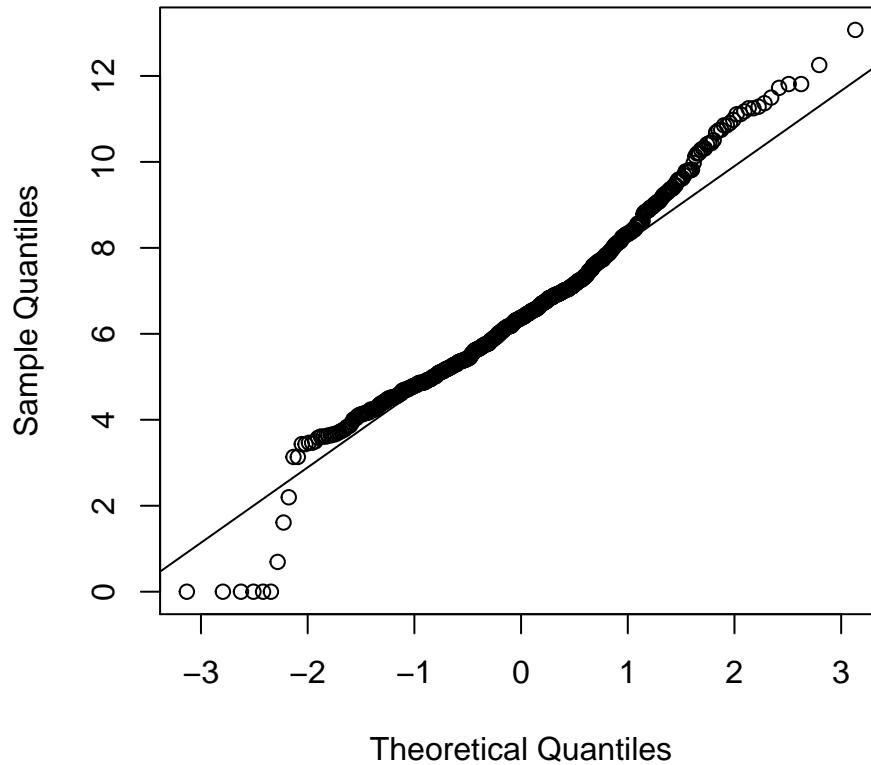
### 2.4.2 Q-Q plots

A better visual method for assessing normality is using a *quantile-quantile plot* (or *Q-Q plot*). For each observation, a Q-Q plot shows the **sample** quantile (the percentile of this point, in the sample) against the quantile that would be expected from a **normal distribution** with the same mean and standard deviation as the sample. If the sample were normally distributed, these two things would be the same, and the plot would just show a straight  $y = x$  line. The degree of deviation from this line thus captures the non-normality of the sample, and allows us to see which data points are “responsible” for deviations from normality.

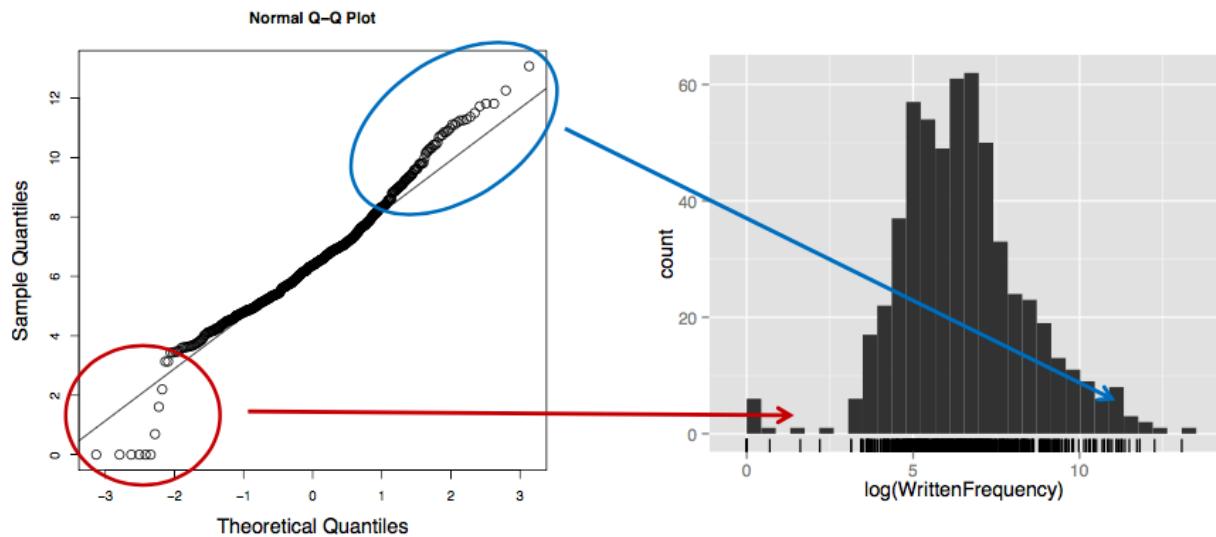
For the “hebben” frequency data:

```
qqnorm(hebben$WrittenFrequency) ## make a Q-Q plot
qqline(hebben$WrittenFrequency) ## plot the y=x line
```

### Normal Q-Q Plot



The deviations from the line correspond to parts of the histogram that are non-normal:



There are:

- Fewer data in left tail than expected in a normal distribution
- More data in right tail than expected in a normal distribution

### 2.4.3 Hypothesis test

Normality of a sample can also be assessed using the Shapiro-Wilk hypothesis test. The null hypothesis in this case is, “the sample is drawn from a normal distribution”.

For the “hebben” frequency sample:

```
shapiro.test(hebben$WrittenFrequency)

##
##  Shapiro-Wilk normality test
##
## data: hebben$WrittenFrequency
## W = 0.97396, p-value = 1.324e-08
```

The very low  $p$ -value suggests that the sample is **not** normally distributed, confirming our intuition from visual methods.

This test is less useful than it may seem, because:

- Any sufficiently large sample will show some deviation from normality—even when a Q-Q plot suggests the distribution is very close to normal.
  - We are never drawing data from a 100% normal distribution, in practice.
  - For this reason, `shapiro.test` in R only works when  $n < 5000$ .
  - See discussion here.
- A non-significant result does not let us conclude that the sample **is** drawn from a normal distribution.
- A significant result doesn’t say anything about what the (non-normal) distribution looks like.

For these reasons, visual methods are preferable to the Shapiro-Wilk test for assessing normality.

### 2.4.4 Other parametric tests

$z$ -tests and  $t$ -tests are called *parametric* hypothesis tests, because it is assumed that the sample is drawn from a distribution with a particular “functional form”—an equation characterized by a few parameters, like a normal distribution ( $\mu, \sigma, n$ ) or a binomial distribution ( $p, n$ ). For example, a two-sample  $t$ -test assumes that the samples are normally distributed with means  $\mu_1$  and  $\mu_2$ , and variances  $\sigma_1$  and  $\sigma_2$ —these are the parameters, which are referred to in the null ( $\mu_1 = \mu_2$ ) and alternative ( $\mu_1 \neq \mu_2$ ) hypotheses.

Some other parametric tests in common use:

- The  $F$ -test of the equality of **variance** between two samples
  - R: `var.test`
  - $F$  tests arise frequently in ANOVA analyses
- The proportion test to compare the probability of success in two groups
  - R: `prop.test`
  - Ex: compare the rate of tapping for “transitive” and “intransitive” sentences (`tapping` data)
- Pearson’s chi-squared test, which can be used to test the independence of two categorical variables (see the Categorical Data Analysis chapter)
  - R: `chisq.test`
  - Ex: assess whether tapping depends on syntactic boundary in the `tapped` dataset (test whether `syntax` and `tapped` are independent)

## 2.5 Non-parametric tests

In contrast, *non-parametric* hypothesis tests do not assume that the sample(s) being tested are drawn from a particular type of distribution (e.g. normal), and tend to have fewer other assumptions. There are non-parametric analogues to all commonly-used parametric hypothesis tests.

### 2.5.1 Wilcoxon tests

The most commonly-used non-parametric analogue to *t*-tests are the *Wilcoxon signed-rank test* (for one sample, or paired data) and the *Wilcoxon rank-sum test* (for two samples, a.k.a. “Mann-Whitney test”).<sup>8</sup>

It is easiest to think of all these as variants of “Wilcoxon tests”, analogous to “*t*-tests”, which have the same variants:

- One-sample
- Two-sample
- Paired

all of which are executed in R using `wilcox.test`, which automatically selects the correct Wilcoxon test depending on the arguments it is given.

Unlike *t*-tests, Wilcoxon tests do not assume sample(s) are drawn from a normal distribution—though they still have the same independence assumptions as *t*-tests.

### 2.5.2 Two-sample Wilcoxon test

The most commonly-used Wilcoxon test is the two-sample *rank-sum test*, which checks whether two samples were drawn from populations with the same distributions. This test is often described as checking whether the samples have different **medians** (as opposed to means, in the two-sample *t*-test), but it is actually more general. For example, two samples with similar medians but different **variances** will be significantly different using a Wilcoxon rank-sum test.

The null hypothesis for this test is that the two samples are drawn from identical population distributions. The test statistic is related to the sum of **ranks** of the data in sample 1, when all observations from both samples are combined and put in order. (The Wikipedia page gives more detail and intuitive examples, if you’re interested.)

The consequence of this rank-based test statistic that is important to remember is that **Wilcoxon tests are robust to outliers** and skewed distributions—like medians are.

#### Example: auxiliaries

We know from above that the frequencies of Dutch “hebben” verbs are not normally distributed (i.e.  $p < 0.0001$  in a Shapiro-Wilk test), so it was not actually appropriate to compare their frequencies with non-“hebben” verbs using a *t*-test.

Let’s compare the frequencies of these two verb groups again, using a Wilcoxon test:

```
wilcox.test(WrittenFrequency~type, regularity)
```

---

<sup>8</sup>The terminology is not standardized: the one-sample test is sometimes called “Wilcoxon *T*-test” (since it is analogous to a one-sample *t*-test), and the two sample test is also called “Mann-Whitney *U*-test” or “Mann-Whitney-Wilcoxon test” or a variation.

```
##  
## Wilcoxon rank sum test with continuity correction  
##  
## data: WrittenFrequency by type  
## W = 29315, p-value = 0.002444  
## alternative hypothesis: true location shift is not equal to 0
```

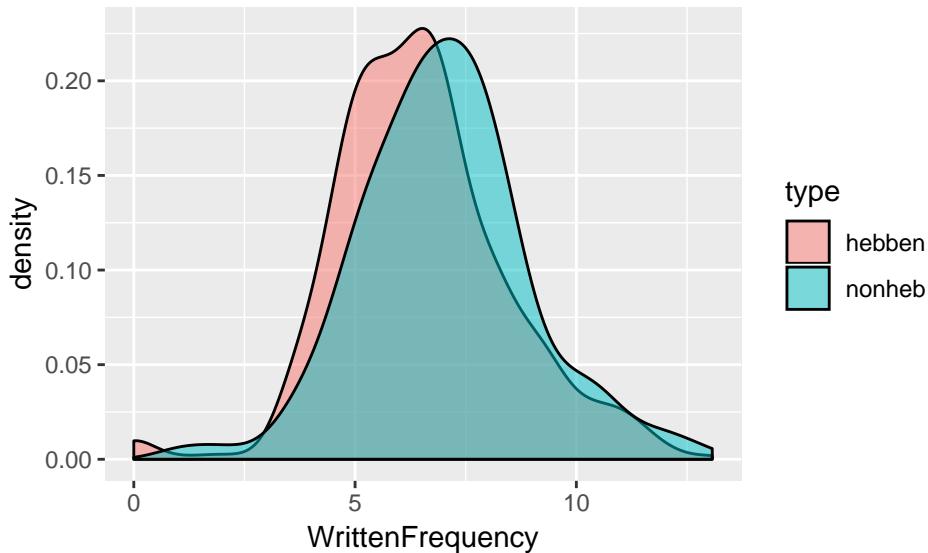
suggesting that *hebben* and *non-hebben* verbs have significantly different frequencies ( $p = 0.002$ ).<sup>9</sup>

To report this result in a paper, you would use the **median** as a descriptive statistic:

Verb frequencies differ significantly by type (Wilcoxon rank-sum:  $W = 29315$ ,  $p = 0.0024$ ), with non-hebben verbs having higher frequency (median=6.9) than hebben verbs (median=6.4).

You would also want to be sure to convey the amount of within-group variation, either by reporting a non-parametric measure of dispersion for each group (such as interquartile range) or including a visualization such as this density plot:

```
regularity %>% ggplot(aes(WrittenFrequency, fill=type)) +  
  geom_density(alpha=0.5)
```



### 2.5.3 Parametric versus non-parametric tests

For the verb frequency example, where the data is not normally distributed, the *t*-test has a higher  $p$ -value than the Wilcoxon test:

- Two-sample *t*-test:  $p = 0.0083$
- Two-sample Wilcoxon test:  $p = 0.0024$

When the samples being compared **are** normally distributed, the *t*-tests will have a (slightly) lower  $p$ -value:

```
set.seed(981)  
x1 <- rnorm(100, mean=1.5, sd=0.5)  
x2 <- rnorm(100, mean=1, sd=1.0)  
t.test(x1, x2)
```

<sup>9</sup>The alternative hypothesis refers to a “location shift” because the null hypothesis is actually “the two samples are drawn from distributions differing only by shifting one over by  $\mu$ ”, where  $\mu$  is a number specified in the test. By default  $\mu = 0$ , which makes the null hypothesis “the two samples are drawn from identical distributions”.

```

## Welch Two Sample t-test
##
## data: x1 and x2
## t = 5.3111, df = 147.12, p-value = 3.948e-07
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
## 0.3312187 0.7237747
## sample estimates:
## mean of x mean of y
## 1.489870 0.962373
wilcox.test(x1, x2)

##
## Wilcoxon rank sum test with continuity correction
##
## data: x1 and x2
## W = 6936, p-value = 2.254e-06
## alternative hypothesis: true location shift is not equal to 0

```

because the *t*-test is “more powerful” (less likely to miss inter-group differences that do in reality exist) for data that obeys the *t*-test’s assumptions.

In general, our advice is to use Wilcoxon tests rather than *t*-tests, because:

- They are robust to outliers.
- You don’t need to check for normality.
- If it “matters” for your result whether a Wilcoxon or *t*-test is used (e.g. *p*-value crosses the significance threshold  $\alpha$ ), the effect is unreliable anyway.

### 2.5.3.1 Type I and Type II error

When should you use parametric versus non-parametric hypothesis tests, more generally? It depends on what your data looks like, and how much you weight the risk of missing a true effect (*Type II error*, which is one minus *power*) versus the risk of detecting a spurious effect (*Type I error*). If you have a choice between a parametric and non-parametric test (such as *t*-tests and Wilcoxon tests):

- The parametric test will (often) be more powerful if its assumptions are met (lower Type II error), and potentially invalid otherwise—susceptible to detecting spurious effects (higher Type I error)<sup>10</sup>
- The non-parametric test potentially has less power to detect effects that in fact exist (higher Type II error), but requires fewer assumptions about the data, thus minimizing the risk of detecting spurious effects (lower Type I error)

The convention in cognitive/behavioral sciences is to prioritize minimizing Type I error over Type II error, which would suggest using the non-parametric test unless you are sure the parametric test is appropriate. However, the choice to minimize Type I error at the expense of power is mostly just tradition, and depending on your research questions it may make more sense to prioritize higher power (assuming the assumptions of the parametric test are met!). This tension between Type I error and Type II error when choosing between possible statistical tools—and to what extent currently standard methods are due to convention versus principled reasons—will come up repeatedly in this book.

---

<sup>10</sup>Using an “invalid” test could also lead to Type II errors, but we emphasize Type I errors for this discussion.

In general the more estimates/fewer assumptions go into a test statistic, the broader its distribution. This implies a larger  $p$ -value, and (assuming the null hypothesis is in fact false) means the test has less power to detect differences.

## 2.6 Other reading

Hypothesis testing is covered in detail with R examples in many textbooks, including those for general audiences, e.g.

- Dalgaard (2008): Ch. 3 & 5
- Crawley (2015): Ch. 5-6

and sources for language scientists/psychologists, such as:

- Vasishth and Nicenboim (2016)
- Navarro (2015): Ch. 10-11.6
- Vasishth (2014): Ch. 2



# Chapter 3

## Linear regression

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
require(languageR)
library(dplyr)
library(ggplot2)
library(arm)

## loads alternativesMcGillLing620.csv from OSF project for Wagner (2016) data
alt <- read.csv(url("https://osf.io/6qctp/download"))

## loads french_medial_vowel_devoicing.txt from OSF project for Torreira & Ernestus (2010) data
df <- read.delim(url("https://osf.io/uncd8/download"))

## loads halfrhymeMcGillLing620.csv from OSF project for Harder (2013) data
halfrhyme <- read.csv(url("https://osf.io/37uqt/download"))
```

**Note:** Most answers to questions/exercises not listed in text are in Solutions.

This chapter introduces linear regression. The broad topics we will cover are:

1. Regression: general introduction
2. Simple linear regression
3. Multiple linear regression
4. Linear regression assumptions, model criticism, and interpretation
5. Model comparison

### 3.1 Regression: General introduction

First, what is “regression”? Chatterjee and Hadi (2012) define it as “a conceptually simple method for investigating functional relationships among variables”:

- The variable to be explained is called the *response*, often written  $Y$
- The explanatory variables are called *predictors*, often written  $X_1, X_2, \dots$

Here is an example of a study that uses regression analysis:

## Effect of cow milk consumption on longitudinal height gain in children

Tomoo Okada

Department of Pediatrics  
Nihon University School of Medicine  
30-1, Oyaguchi-kamimachi  
Itabashi-ku  
Tokyo 173-8610  
Japan  
E-mail: tomo{at}med.email.ne.jp

**Dear Sir:**

Black et al (1) studied prepubertal children who had a long history of avoiding consumption of cow milk and found that such children tend to have short stature and high adiposity. Blanaru et al (2) confirmed that dietary arachidonic acid alters bone mass in piglets fed cow milk-based formula. We are very interested in their results because in a previous prospective study, we examined the effect of cow milk consumption on longitudinal height gain in children (3).

The subjects were 122 children (60 boys and 62 girls) aged  $9.5 \pm 0.2$  y ( $\bar{x} \pm SD$ ). Standing height and weight were measured, and relative weight was obtained according to the standard weight for sex, age, and height. Three years later, we recruited the subjects for the second part of the study, which included anthropometric measurements and the questionnaire about cow milk consumption. The question was "How much cow milk do you usually drink a day?" The possible answers were "<250 mL," "250-500 mL," "500-1000 mL," and ">1000 mL." We investigated the relation between cow milk consumption and longitudinal changes in height, weight, and relative weight.

**Questions:**

- What is the response variable in this study?
- What could be a predictor variable?

### 3.1.1 Linear models

The relationship between variables is captured by a regression *model*:

$$Y = f(X_1, X_2, \dots, X_p) + \epsilon$$

In this model,  $Y$  is approximated by a function of the predictors, and the difference between the model and reality is called the *error* ( $\epsilon$ ).

Throughout this book we will be dealing exclusively with *linear models* (including “generalized linear models”, like logistic regression), where the model takes the form:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \quad (3.1)$$

The  $\beta$ 's are called *regression coefficients*.

This turns out to be an extremely general class of model, which can be applied to a wide range of phenomena. Some important kinds of models that don't appear at first glance to fit Equation (3.1) can be linearized into this form.

### 3.1.2 Terminology

We will consider two broad types of regression in this book:

1. *linear regression*, where the response ( $Y$ ) is a continuous variable. An example would be modeling reaction time (RTlexdec) as a function of word frequency (WrittenFrequency) for the english dataset.
2. Later we will consider *logistic regression*, where the response ( $Y$ ) is binary: 0 or 1. An example would be modeling whether tapping occurs or not (tapping) as a function of vowelDuration and speakingRate in the tapping dataset.

Regression with just one predictor is called *simple*, while regression with multiple predictors is called *multiple*. The two examples just given would be “simple linear regression” and “multiple logistic regression”.

Predictors can be *continuous*, such as milk consumption, or *categorical*—also called “factors”—such as participant gender, or word type.

Certain special cases of linear models that are common go by names such as:

- *Analysis of variance*: continuous  $Y$ , categorical predictors. (Models variability between groups)
- *Analysis of covariance*: continuous  $Y$ , mix of categorical and continuous predictors.

We are not covering these cases in this book, but ANOVAs (and ANCOVAs) are widely used in language research, and you may have seen them before. ANOVAs can be usefully thought of as just a **special case** of regression, as discussed in Vasishth and Broe (2011), Levy (2012), Gelman and Hill (2007). Once you understand linear regression well, understanding ANOVA analyses is relatively straightforward.

### 3.1.3 Steps and assumptions of regression analysis

Regression analyses have five broad steps, as usefully discussed by Chatterjee and Hadi (2012):

1. Statement of the problem
  - Ex: Does milk consumption affect height gain?
2. Selection of potentially relevant variables
  - Ex: Height gain, daily milk consumption, age, and sex.
3. Data collection
  - Ex: data from an existing database.
4. Model specification & fitting
  - Ex: height gain =  $\beta_0 + \beta_1 \cdot$  milk consumption +  $\beta_2 \cdot$  sex + error

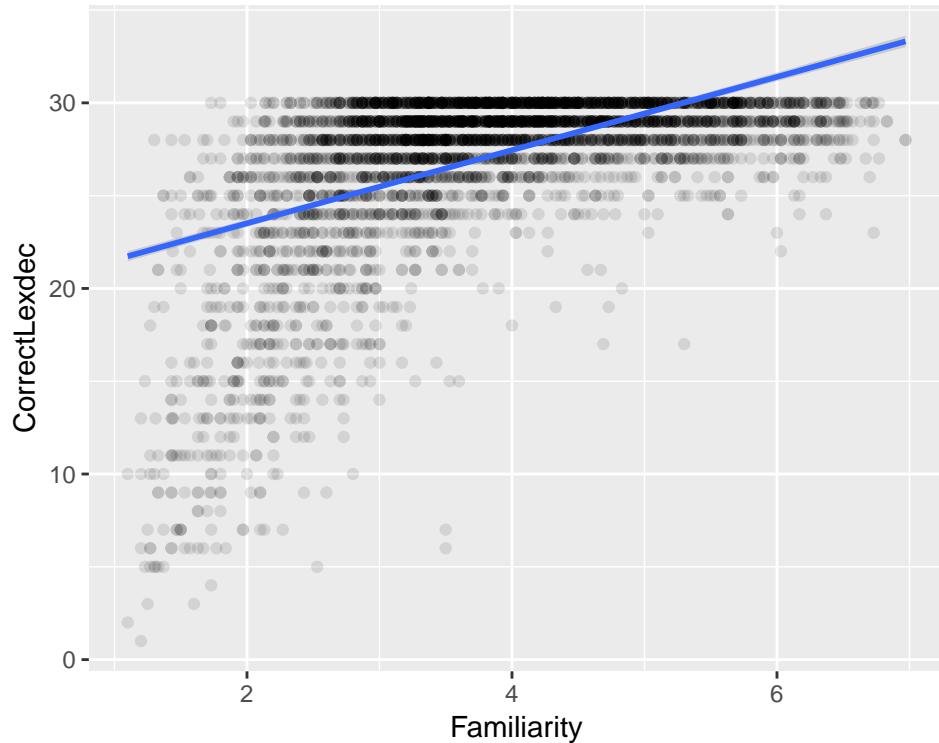
### 5. Model validation and criticism

It is important to remember that the **validity of a regression analysis depends on the assumptions of the data and model**.

For example, if you are modeling data where  $Y$  has a maximum value, fitting a simple linear regression (= a line) to this data doesn't make conceptual sense, a priori. Here's an example using the `english` dataset:

```
library(languageR) ## makes the 'english' dataset available

ggplot(aes(x = Familiarity, y = CorrectLexdec), data = english) + geom_point(alpha=0.1) + geom_smooth
```



Because `CorrectLexdec` has a maximum value of 30, fitting a line doesn't make sense—the predicted value when `Familiarity`=6 is above 30, but this is impossible given the definition of `CorrectLexdec`.

For now, we will not go into what the assumptions of linear regression are, and just assume that they are met. After introducing simple and multiple linear regressions, we'll come back to this issue in Section 3.4:

- What are the assumptions (of linear regression)?
- For each assumption, how do we determine whether it's valid?
- How much of a problem is it, and what can be done, if the assumption is not met?

Note that R almost never checks whether your data and model meet regression analysis assumptions, unlike other software (e.g. SPSS, sometimes).

## 3.2 Simple linear regression

The simplest application of simple linear regression (SLR) is to model an association between two continuous variables.

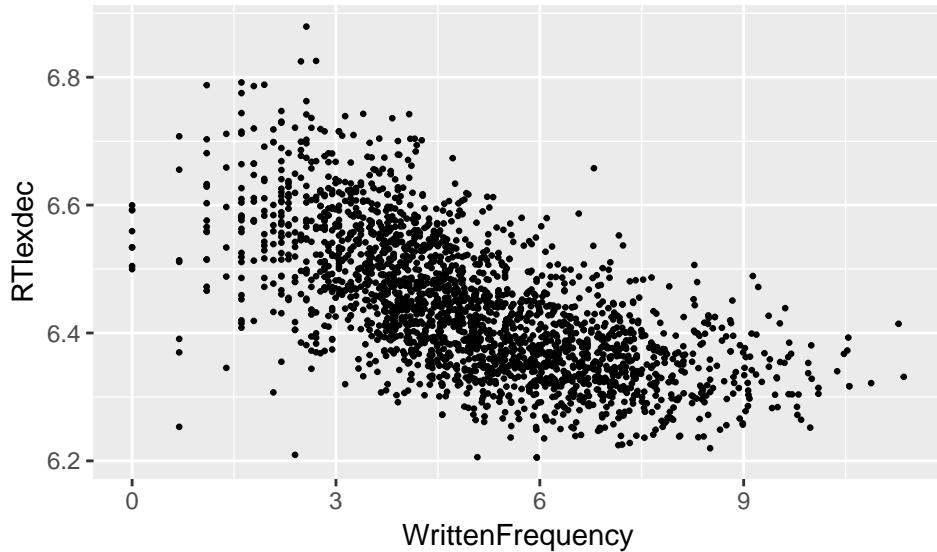
Example: `english` data, young participants only

- $X$ : `WrittenFrequency` (= predictor)

- $Y$ : `RTlexdec` (= response)

```
young <- filter(english, AgeSubject=='young')

ggplot(young, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5)
```

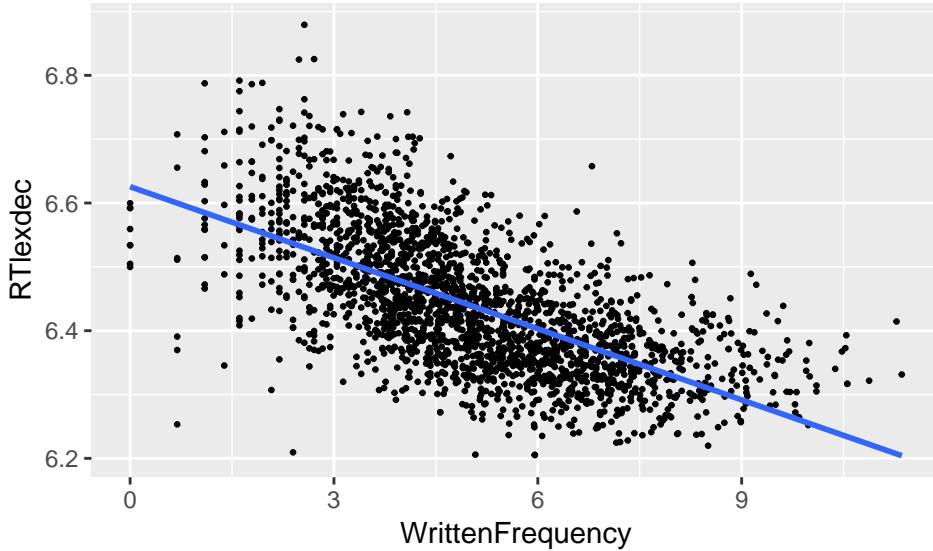


One way to describe this kind of association which you may be familiar with is a *correlation* coefficient which gives us two types of information about the relationship:

1. *direction*:  $r = -0.434$ 
  - $\Rightarrow$  negative relationship
2. *strength*:  $r^2 = 0.189$ 
  - $\Rightarrow$  weak relationship ( $0 \leq r^2 \leq 1$ )

A simple linear regression gives a *line of best fit*:

```
ggplot(young, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth(method="lm", se=F)
```



which gives some information not captured by a correlation coefficient:

- Prediction of  $Y$  for a given  $X$
- Numerical description of relationship

Regression gives both types of information, and more.

### 3.2.1 SLR: Continuous predictor

The formula for simple linear regression, written for the  $i^{\text{th}}$  observation, is:

$$y_i = \underbrace{\beta_0}_{\text{intercept}} + \underbrace{\beta_1}_{\text{slope}} x_i + \epsilon_i \quad (3.2)$$

In this expression:

- $\beta_0, \beta_1$  are *coefficients*
- For the  $i^{\text{th}}$  observation
  - $x_i$  is the value of the *predictor*
  - $y_i$  is the value of the *response*
  - $\epsilon_i$  is the value of the *error* (or *residual*)

This is our first linear model of a random variable ( $Y$ ) as a function of a predictor variable ( $X$ ). The actual model, written not for individual observations, is written:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

That is, we use notation like  $X$  for a variable, and notation like  $x_5$  for actual values that it takes on.

For example, for the `english` dataset, with  $Y = \text{RTlexdec}$  and  $X = \text{WrittenFrequency}$ :

```
head(dplyr::select(english, WrittenFrequency, RTlexdec))
```

```
##   WrittenFrequency RTlexdec
## 1          3.912023  6.543754
## 2          4.521789  6.397596
## 3          6.505784  6.304942
## 4          5.017280  6.424221
## 5          4.890349  6.450597
## 6          4.770685  6.531970
```

- $x_2=4.5217886$  (predictor value for second observation)
- $y_1=6.5437536$  (response value for first observation)

### Question

- What is  $y_5$ ?

## 3.2.2 SLR: Parameter estimation

To get a line of best fit, we want:  $\beta_0$  and  $\beta_1$ , the *population values*. Recall that we can't actually observe these (Sec.1.1), so we obtain *sample estimates*, written  $\hat{\beta}_0, \hat{\beta}_1$ .

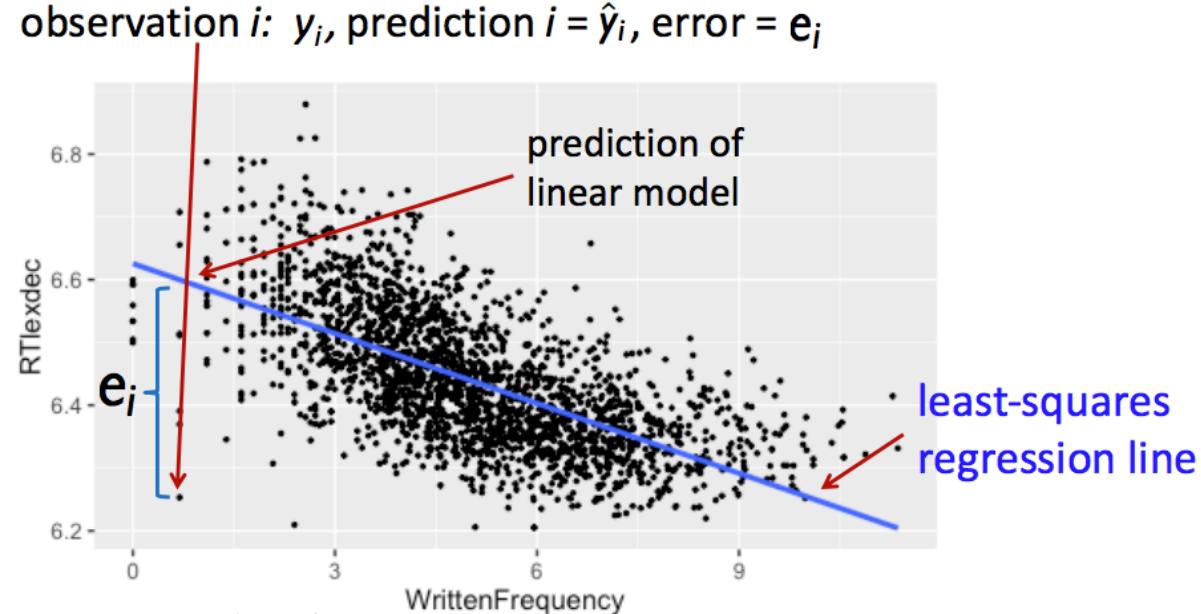
Once sample estimates are specified, Equation (3.2) gives *fitted values* for each observation, written  $\hat{y}_i$ :

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

Note that there are no residuals in this equation— $\epsilon_i$  are again population values, which we can't observe. Our estimates of the residuals, given an estimated line of best fit, are written  $e_i$  (*error*):

$$e_i = y_i - \hat{y}_i$$

This diagram shows the relationship between some of these quantities, for a single observation:



Our goal is to find coefficient values that minimize the difference between observed and expected values—the magnitudes of error ( $|e_i|$ ), which are minimized by minimizing the squared errors ( $e_i^2$ ).

We choose  $\hat{\beta}_0, \hat{\beta}_1$  that minimize the sum of the  $e_i^2$ ; these are called the *least-squares estimates*.

One useful property of the resulting regression line is that it always passes through the point  $(\text{mean}(X), \text{mean}(Y))$ . A consequence is that SLR is easily thrown off by observations which are outliers in  $X$  or  $Y$  (why?).

### Example

Here is an example of fitting an SLR model in R, of reaction time vs. frequency for young speakers in the `english` dataset:

```
young <- filter(english, AgeSubject == "young")
m <- lm(RTlexdec ~ WrittenFrequency, young)
```

The model output is:

```
m
```

```
##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency, data = young)
##
## Coefficients:
## (Intercept) WrittenFrequency
##           6.62556          -0.03711
```

The interpretation of the two coefficients is:

- $\beta_0$ : the predicted  $Y$  value when  $X = 0$

```
m$coefficients[1]
```

```
## (Intercept)
##       6.625556
```

- $\beta_1$ : predicted change in  $Y$  for every unit change in  $X$

```
m$coefficients[2]
```

```
## WrittenFrequency
##      -0.03710692
```

The regression line is:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i \quad (3.3)$$

In terms of the variables used in this example:

$$\text{Predicted RTlexdec}_i = 6.625 - 0.037 \cdot \text{WrittenFrequency}_i \quad (3.4)$$

### Questions:

What is the predicted `RTlexdec` when:

- `WrittenFrequency` = 5?
- `WrittenFrequency` = 10?

### 3.2.3 Hypothesis testing

The least-squared estimators are normally-distributed. These are sample estimates, so we can also approximate the standard errors of the estimators:

$$SE(\hat{\beta}_0) \quad SE(\hat{\beta}_1)$$

and apply  $t$ -tests to test for significance and obtain confidence intervals (CI) for the coefficients.

In particular, we are testing the null hypotheses of no relationship:

- $H_0 : \beta_1 = 0$

(and similarly for  $\beta_0$ ).

We then apply a  $t$ -test, using test statistic:

$$t_1 = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)} \quad (3.5)$$

$$df = n - 2$$

(where  $n$  = number of observations).

The resulting  $p$ -value tells us how surprised are we to get a slope this far from zero ( $\beta_1$ ) under  $H_0$ . (With this high a standard error, given this much data.)

To see the results of these hypothesis tests in R:

```
summary(lm(RTlexdec~WrittenFrequency, young))

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency, data = young)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34664 -0.05523 -0.00546  0.05167  0.34877
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.6255559  0.0049432 1340.34  <2e-16 ***
## WrittenFrequency -0.0371069  0.0009242  -40.15  <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08142 on 2282 degrees of freedom
## Multiple R-squared:  0.414, Adjusted R-squared:  0.4137
## F-statistic: 1612 on 1 and 2282 DF, p-value: < 2.2e-16
```

The  $t$ -values and associated  $p$ -values are in the **Coefficients:** table, where the first row is for  $\hat{\beta}_0$  and the second row is for  $\hat{\beta}_1$ .

#### Questions:

- Why does the column for the  $p$ -value say  $Pr(>|t|)$ ?

Having the SEs of the coefficients also lets us compute 95% confidence intervals for the least-squared estimators. Going from the null hypothesis ( $H_0$ ) above: if the 95% CI of the slope ( $\beta_1$ ) does not include 0, we can reject  $H_0$  with  $\alpha = 0.05$ .

In R, you can get confidence intervals for a fitted model as follows:

```
confint(lm(RTlexdec~WrittenFrequency, young))

##                   2.5 %      97.5 %
## (Intercept)    6.61586227  6.63524948
## WrittenFrequency -0.03891921 -0.03529463
```

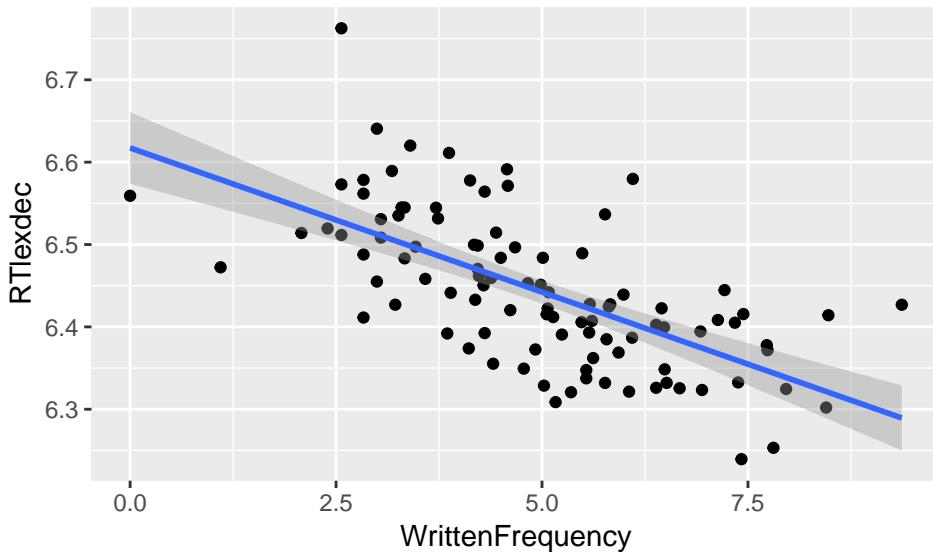
Neither CI includes zero, consistent with the very low  $p$ -values in the model table above.

### Example: Small subset

For the full dataset of `english` young speakers, it's a little silly to do hypothesis testing given how much data there is and the clarity of the pattern—the line of best fit has a tiny confidence interval. Just for exposition, let's look at the line of best fit for a subset of just  $n = 100$  points:

```
set.seed(2903) # This makes the following "random" sampling step always give the same result
young_sample <- young %>% sample_n(100)

ggplot(young_sample, aes(WrittenFrequency, RTlexdec)) +
  geom_point() +
  geom_smooth(method="lm")
```



### Questions:

- What does `geom_smooth(method="lm")` do?

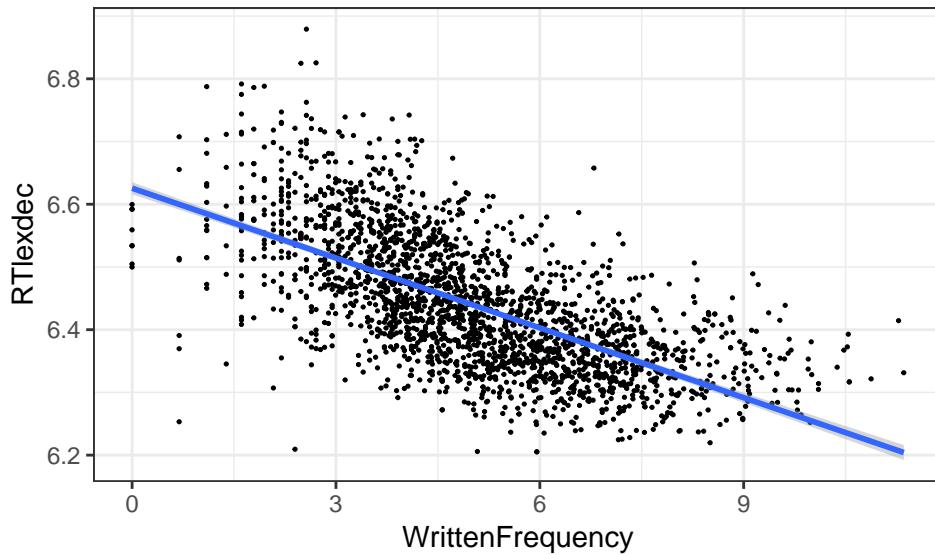
In this plot, the shading around the line is the 95% confidence interval. “Can we reject  $H_0$ ?” is equivalent to asking, “Can a line with 0 slope cross the shaded area through the range of  $x$  (and going through  $(\text{mean}(X), \text{mean}(Y))$ )?

### 3.2.4 Quality of fit

Here is the model we have been discussing, plotted on top of the empirical data:

```
young <- filter(english, AgeSubject == "young")

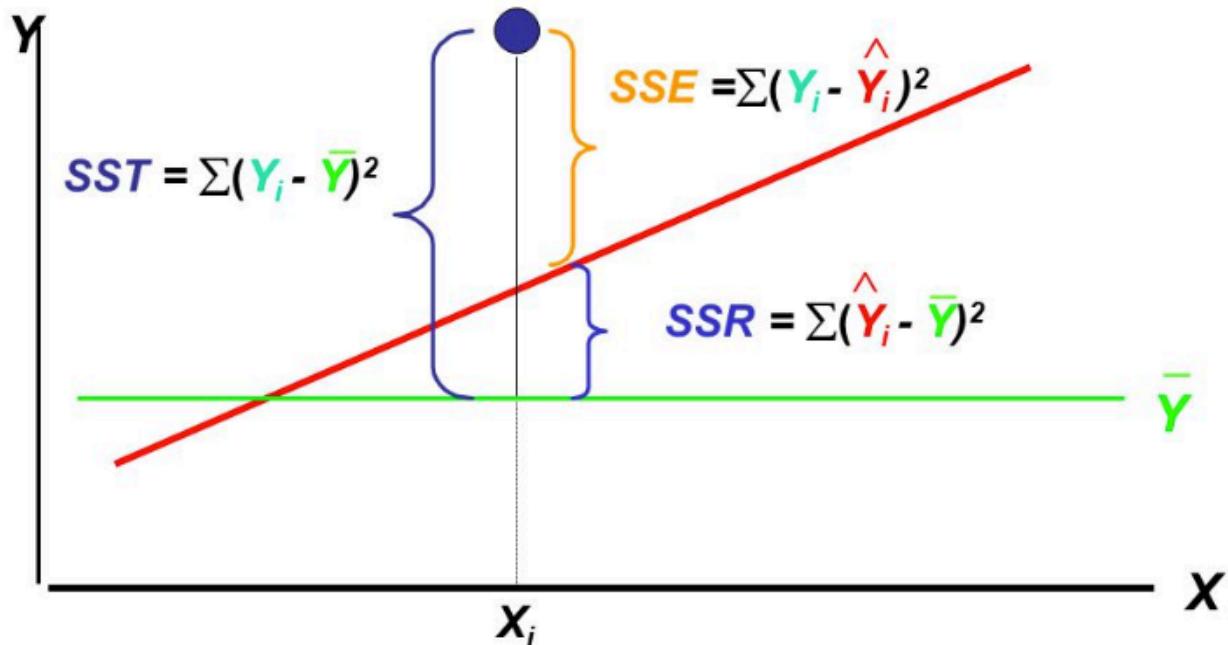
ggplot(young, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.25) +
  geom_smooth(method="lm") +
  theme_bw()
```



We often want a metric quantifying how well a model fits the data—the *goodness of fit*.

For simple linear regression, we can derive such a metric by first defining three quantities:

- $SST$ : Total sum of squares
- $SSR$ : Sum of squares due to regression
- $SSE$ : Sum of squares due to error



(Source: slides from *Business Statistics: A First Course (Third edition)*)

The fundamental equality is:  $SST = SSR + SSE$

Intuitively we want a measure of how much of SST is accounted for by SSR. This is  $R^2$ : the proportion of total variability in  $Y$  accounted for by  $X$ :

$$R^2 = \frac{SS_{\text{fit}}}{SS_{\text{total}}} \quad (3.6)$$

$R^2$  always lies between 0 and 1, which can conceptually be thought of as:

- 0: none of the variance in  $Y$  is accounted for by  $X$
- 1: all of the variance ““

It turns out  $R^2$  is also the square of (Pearson's) correlation,  $r$ .

### Example

Here is the SLR model fitted to a subset of 100 data points, repeated for convenience:

```
set.seed(2903)

d <- english %>% filter(AgeSubject=="young") %>% sample_n(100)
summary(lm(RTlexdec~WrittenFrequency, d))
```

```
##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.127857 -0.045072 -0.005826  0.042917  0.235034
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.61733   0.02190 302.236 < 2e-16 ***
## WrittenFrequency -0.03501   0.00419 -8.354 4.43e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07121 on 98 degrees of freedom
## Multiple R-squared:  0.4159, Adjusted R-squared:  0.41
## F-statistic: 69.79 on 1 and 98 DF,  p-value: 4.43e-13
```

In the model table, note the values of the sample statistic under `t value` and its significance in the `Pr(>|t|)` column in the `WrittenFrequency` row, as well as of the correlation statistic `Multiple R-squared`.

This is a hypothesis test for Pearson's  $r$  for the same data, checking whether it is significantly different from 0:

```
cor.test(d$WrittenFrequency, d$RTlexdec)

##
## Pearson's product-moment correlation
##
## data: d$WrittenFrequency and d$RTlexdec
## t = -8.354, df = 98, p-value = 4.43e-13
```

```
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
## -0.7467533 -0.5135699
## sample estimates:
##       cor
## -0.644931
```

Note that  $t$ ,  $p$ , and  $r$  (the square root of `Multiple R-squared`) are exactly the same. Thus, fitting a simple linear regression and conducting a correlation test give us two ways of finding the same information.

### 3.2.5 Categorical predictor

Simple linear regression easily extends to the case of a binary  $X$  (a *factor*).

#### Example

- `english` data
- Predictor: `AgeSubject`
- Response: `RTlexdec`

Everything is the same as for the case where  $X$  is continuous, except now we have:

- $x_i = 0$ : if `AgeSubject == "old"`
- $x_i = 1$ : if `AgeSubject == "young"`

The regression equation is exactly the same as Equation (3.2):

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i$$

Only the interpretation of the coefficients differs:

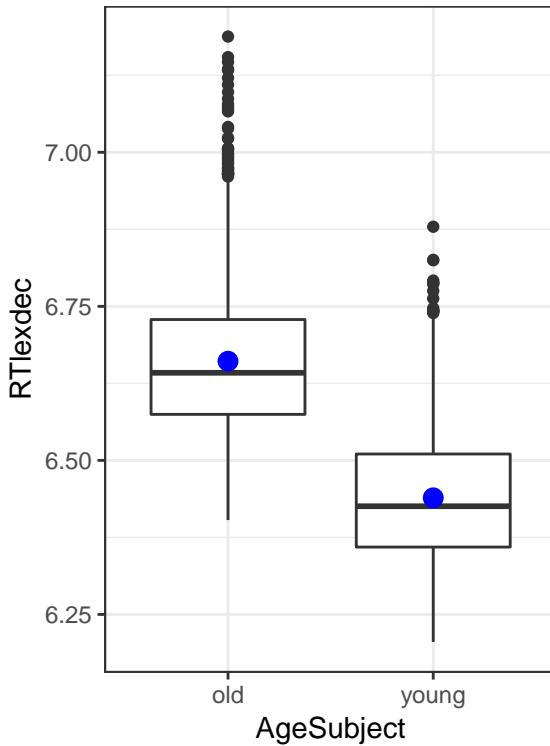
- $\beta_0$ : **mean** `RTlexdec` when `AgeSubject == "old"` (since  $x_i = 0 \iff \text{AgeSubject} == \text{"old"}$ )
- $\beta_1$ : **difference in mean** `RTlexdec` between young and old

Hypothesis tests,  $p$ -values, CIs, and goodness of fit work exactly the same as for a continuous predictor.

#### Example

Suppose we want to test whether the difference in group means is statistically significantly different from 0:

```
english %>% ggplot(aes(AgeSubject, RTlexdec)) +
  geom_boxplot() +
  stat_summary(fun.y="mean", geom="point", color="blue", size=3) +
  theme_bw()
```



#### Question:

- What goes in the blanks?

```
m1 <- lm(____ ~ ____ , english)
summary(m1)
```

Note the  $t$  and  $p$ -values for `AgeSubjectyoung`, we'll need them in a second.

### 3.2.6 SLR with a binary categorical predictor vs. two-sample $t$ -test

Conceptually, we just did the same thing as a two-sample  $t$ -test—tested the difference between two groups in the value of a continuous variable. Let's see what the equivalent  $t$ -test gives us:

```
t.test(RTlexdec ~ AgeSubject, english, var.equal=T)
```

```
##
##  Two Sample t-test
##
## data:  RTlexdec by AgeSubject
## t = 67.468, df = 4566, p-value < 2.2e-16
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  0.2152787 0.2281642
## sample estimates:
## mean in group old mean in group young
##          6.660958           6.439237
```

(The `var.equal` option forces the  $t$ -test to assume equal variances in both groups, which is one assumption of linear regression.)

You should find that both tests give identical  $t$  and  $p$ -values. So, a  $t$ -test can be thought of as a special case of simple linear regression.

### 3.2.6.1 Bonus: Linear vs. smooth regression lines

We have forced an SLR fit in the plots above using the `method='lm'` flag, but by default `geom_smooth` uses a *nonparametric smoother* (such as LOESS, the `geom_smooth` default for small samples):

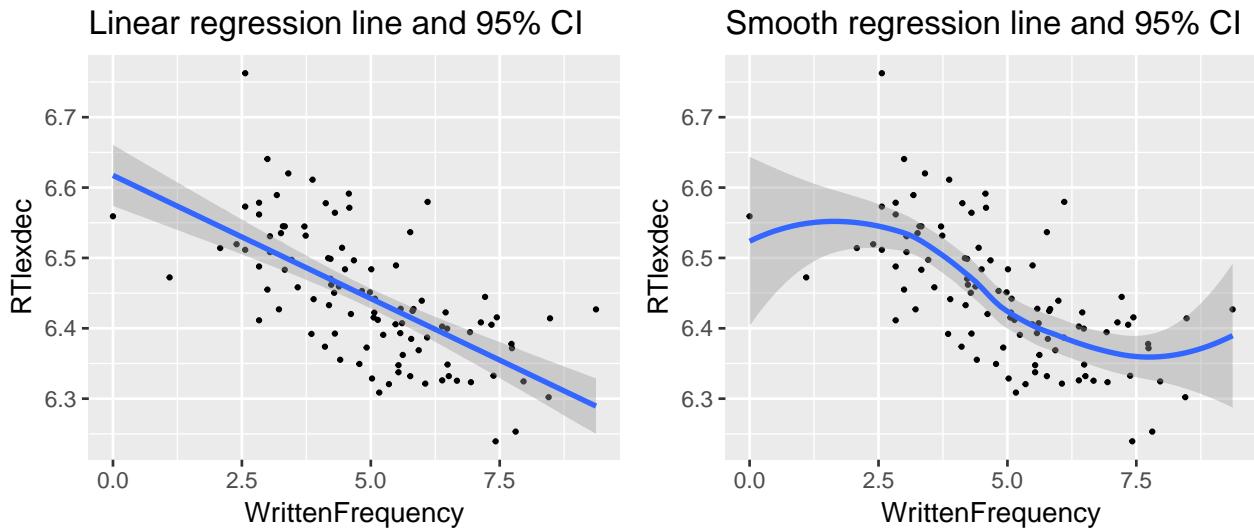
```
young <- filter(english, AgeSubject=='young')

set.seed(2903)
young_sample <- young %>% sample_n(100)

day7_plt1 <- ggplot(young_sample, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth(method="lm") +
  ggtitle("Linear regression line and 95% CI")

day7_plt2 <- ggplot(young_sample, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth() +
  ggtitle("Smooth regression line and 95% CI")

grid.arrange(day7_plt1, day7_plt2, ncol = 2)
```



Note the differences in the two plots:

- Linear/nonlinear
- CI widths related to distance from mean, versus **amount of data nearby**

(Hence “Local” in LOESS.)

## 3.3 Multiple linear regression

In *multiple linear regression*, we use a linear model to predict a continuous response with  $p$  predictors ( $p > 1$ ):

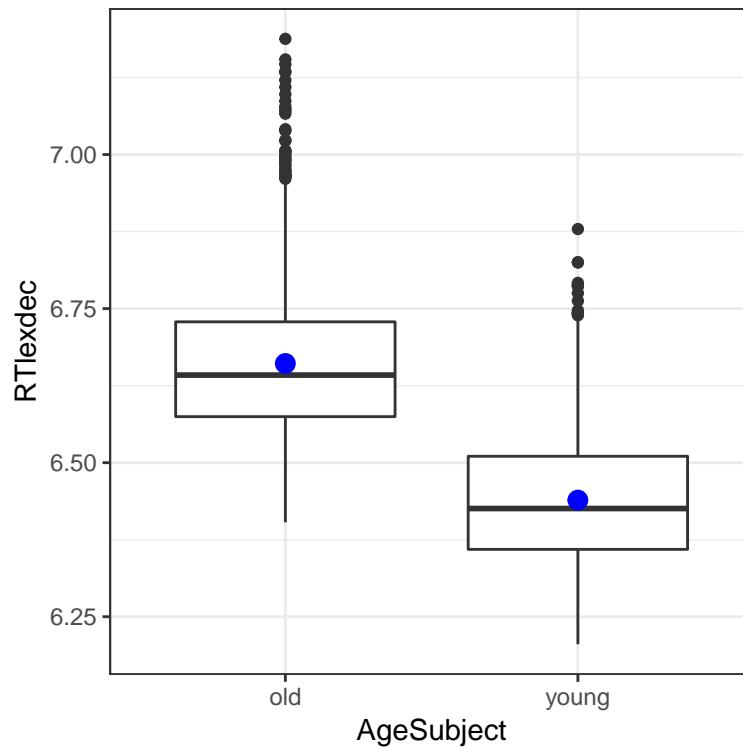
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

Each predictor  $X_i$  can be continuous or categorical.

**Example: RT ~ frequency + age**

For the `english` dataset, let's model reaction time as a function of word frequency and participant age. Recall that in addition to the word frequency effect, older speakers react more slowly than younger speakers:

```
english %>% ggplot(aes(AgeSubject, RTlexdec)) +
  geom_boxplot() +
  stat_summary(fun.y="mean", geom="point", color="blue", size=3) +
  theme_bw()
```



The response and predictors are:

- $Y$ : RTlexdec
- $X_1$ : WrittenFrequency (continuous)
- $X_2$ : AgeSubject (categorical) (0: old, 1: young)

Because  $p = 2$ , the regression equation for observation  $i$  is

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \epsilon_i$$

where  $x_{ij}$  means the value of the  $j^{\text{th}}$  predictor for the  $i^{\text{th}}$  observation.

To fit this model in R:

```
m2 <- lm(RTlexdec~WrittenFrequency+AgeSubject, english)
```

Summary of the model:

```
summary(m2)

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + AgeSubject, data = english)
##
## Residuals:
##    Min      1Q  Median      3Q     Max 
## -0.34622 -0.06029 -0.00722  0.05178  0.44999 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  6.8467921  0.0039792 1720.64 <2e-16 ***
## WrittenFrequency -0.0370103  0.0007033  -52.62 <2e-16 ***
## AgeSubjectyoung -0.2217215  0.0025930   -85.51 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08763 on 4565 degrees of freedom
## Multiple R-squared:  0.6883, Adjusted R-squared:  0.6882 
## F-statistic: 5040 on 2 and 4565 DF, p-value: < 2.2e-16
```

This model tells us that the least-squares solution for the regression line is:

$$RTlexdec = \underbrace{6.846}_{\beta_0} + \underbrace{(-0.037)}_{\beta_1} \cdot \text{WrittenFrequency} + \underbrace{(-0.221)}_{\beta_2} \cdot \text{AgeSubject} + \text{error}$$

#### Questions:

- What RT does the model predict for an observation with `WrittenFrequency=5` and `AgeSubject='old'`?

```
6.846 + (-0.037 * 5) + (-0.221 * 0)
```

```
## [1] 6.661
```

Note that in this MLR model, the interpretation of each coefficient is:

- $\beta_0$ : predicted value when all predictors = 0
- $\beta_1, \beta_2$ : change in a predictor **when others are held constant**

For example, the difference between old and young speakers in RT is 0.221, when word frequency is held constant.

### 3.3.1 Goodness of fit metrics

With  $R^2$  defined as for simple linear regression, in terms of sums of squares, the exact same measure works to quantify goodness of fit of a multiple linear regression:

$$R^2 = \frac{SS_{\text{fit}}}{SS_{\text{total}}}$$

sometimes called *multiple  $R^2$* .

An alternative to  $R^2$  when there's more than one predictor (MLR) is *adjusted  $R^2$* , defined as:

$$R^2 = 1 - \frac{SS_{\text{fit}}/(n-p-1)}{SS_{\text{total}}/(n-1)}$$

where  $p$  is the number of predictors and  $n$  is the number of observations.

In this expression, the sum-of-squares term can be thought of as a ratio comparing the amount of variance explained by two models: the “full” model (the one with  $p$  predictors) and the “baseline” model (the one with just the intercept). Each model’s sum of squares is scaled by its degrees of freedom; intuitively, this gives a measure of how much variance is explained **given the number of predictors**. (We expect that if you throw more predictors in a model, more variance can be explained, just by chance.)

The adjusted  $R^2$  measure only increases if the  $p$  additional predictors improve the model more than would be expected by chance.

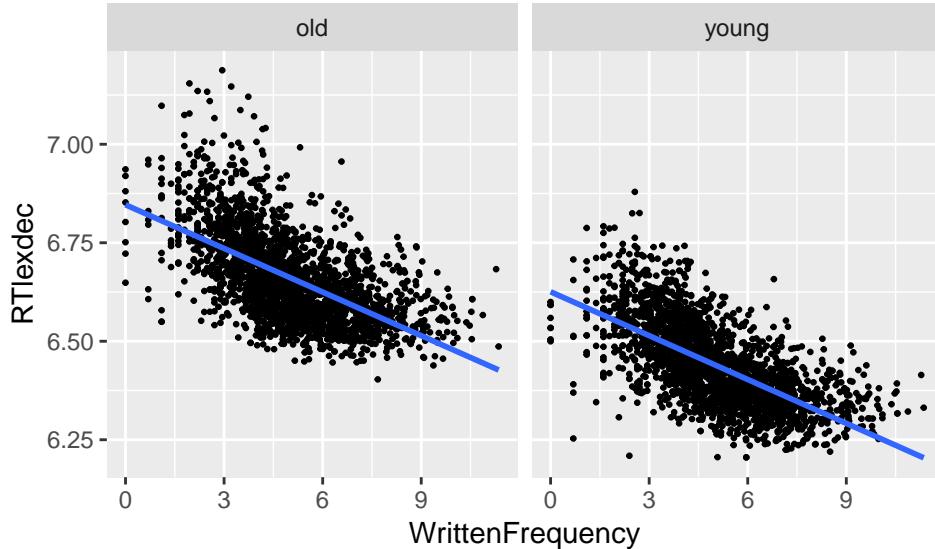
- **Pro:** Adjusted  $R^2$  is more appropriate as a metric for comparing different possible model—unlike “multiple  $R^2$ ”, adjusted  $R^2$  doesn’t automatically increase whenever new predictors are added.
- **Con:** Multiple  $R^2$  is no longer interpretable as “fraction of the variation accounted for by the model”.

R reports both adjusted and non-adjusted versions, as seen in the model summary above.

### 3.3.2 Interactions and factors

The models we have considered so far assume that each predictor affects the response **independently**. For example, in the example above ( $RT \sim \text{frequency} + \text{age}$ ), our model assumes that the slope of the frequency effect on RT is the same for old speakers as for young speakers. This looks like it might be approximately true:

```
ggplot(english, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth(method="lm", se=F) +
  facet_wrap(~AgeSubject)
```



in that there seems to be a similarly negative slope for both groups. That is, a model of this form seems approximately correct:

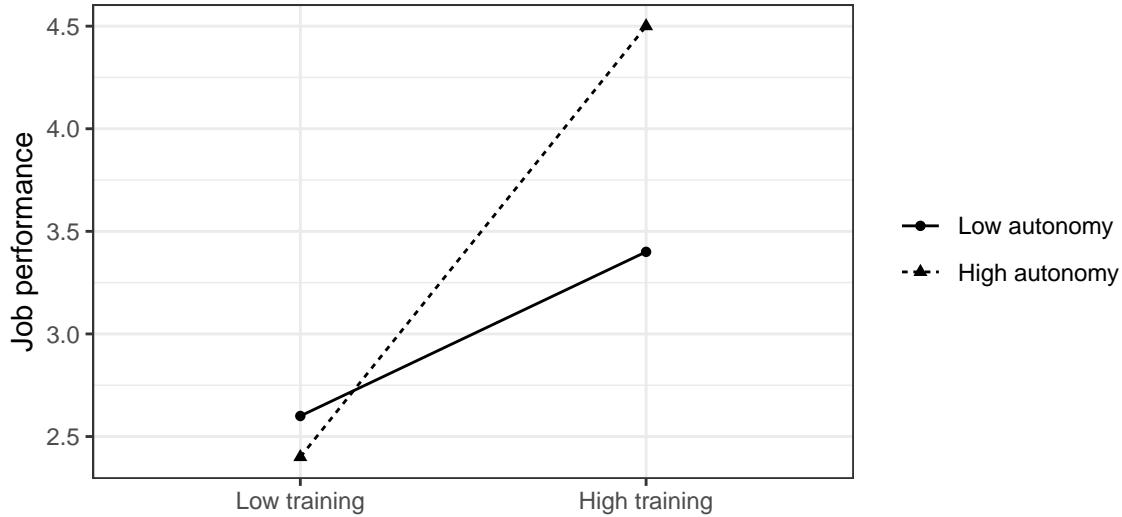
$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \epsilon$$

( $\epsilon$  = error).

Here is a (fake) example where the independence assumption is definitely not true:

- $Y$ : Job performance (continuous)
- $X1$ : Training (categorical)

- $X_2$ : Autonomy (categorical)



The effect of training on job performance is larger for high-autonomy participants. In this case, we say there is an *interaction* between training and autonomy: the value of one predictor modulates the effect of the other. This interaction is modeled by adding an extra term to the regression equation:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon$$

which is the *product* of the two terms. Note that

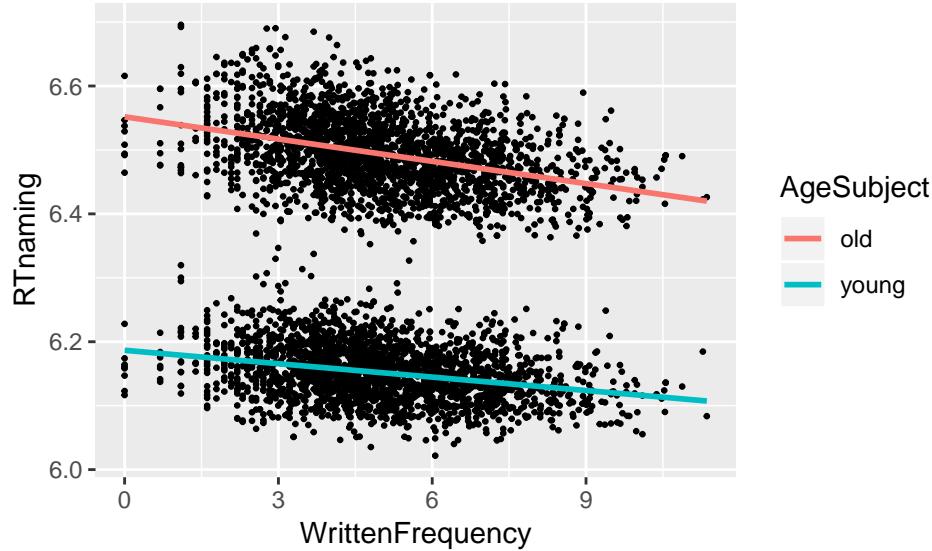
$$\beta_2 X_2 + \beta_3 X_1 X_2 = (\beta_2 + \beta_3 X_1) X_2$$

so the interaction coefficient  $\beta_3$  modulates the slope of  $X_2$ : depending on the value of  $X_1$ ,  $X_2$  has a different effect on  $Y$ .

### Example

Returning to the example above, suppose we'd like to know how much the slope of `WrittenFrequency` does actually differ between old and young speakers, and whether the difference is statistically significant ( $\alpha = 0.05$ ).

```
english %>% ggplot(aes(WrittenFrequency, RTnaming)) +
  geom_point(size=0.5) +
  geom_smooth(aes(group=AgeSubject, color=AgeSubject), method="lm", se=F)
```



$X_1:X_2$  means “interaction between  $X_1$  and  $X_2$ , and the notation used in R for interactions is  $X_1*X_2$ , which expands automatically to  $X_1 + X_2 + X_1:X_2$ . (The non-interaction terms are sometimes called *main effects*.) Note that in R these are equivalent:

```
lm(RTnaming ~ WrittenFrequency * AgeSubject, english)
lm(RTnaming ~ WrittenFrequency + AgeSubject + WrittenFrequency:AgeSubject, english)
```

To fit a model including an interaction between frequency and age:

```
m3 <- lm(RTnaming ~ WrittenFrequency * AgeSubject, english)
```

In the summary of this model, of interest is the WrittenFrequency:AgeSubjectyoung row, which is the interaction effect:

```
summary(m3)
```

```
##
## Call:
## lm(formula = RTnaming ~ WrittenFrequency * AgeSubject, data = english)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.160510 -0.033425 -0.002963  0.030855  0.181032
##
## Coefficients:
## (Intercept)          Estimate Std. Error t value Pr(>|t|)
## (Intercept)          6.5517608  0.0029118 2250.09 < 2e-16
## WrittenFrequency     -0.0116031  0.0005444 -21.31 < 2e-16
## AgeSubjectyoung      -0.3651823  0.0041179 -88.68 < 2e-16
## WrittenFrequency:AgeSubjectyoung  0.0046191  0.0007699    6.00 2.13e-09
##
## (Intercept)          ***
## WrittenFrequency     ***
## AgeSubjectyoung      ***
## WrittenFrequency:AgeSubjectyoung ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 0.04796 on 4564 degrees of freedom
## Multiple R-squared:  0.9278, Adjusted R-squared:  0.9278
## F-statistic: 1.956e+04 on 3 and 4564 DF, p-value: < 2.2e-16
```

We see that there is indeed a significant interaction between `WrittenFrequency` and `AgeSubject`.

#### Questions:

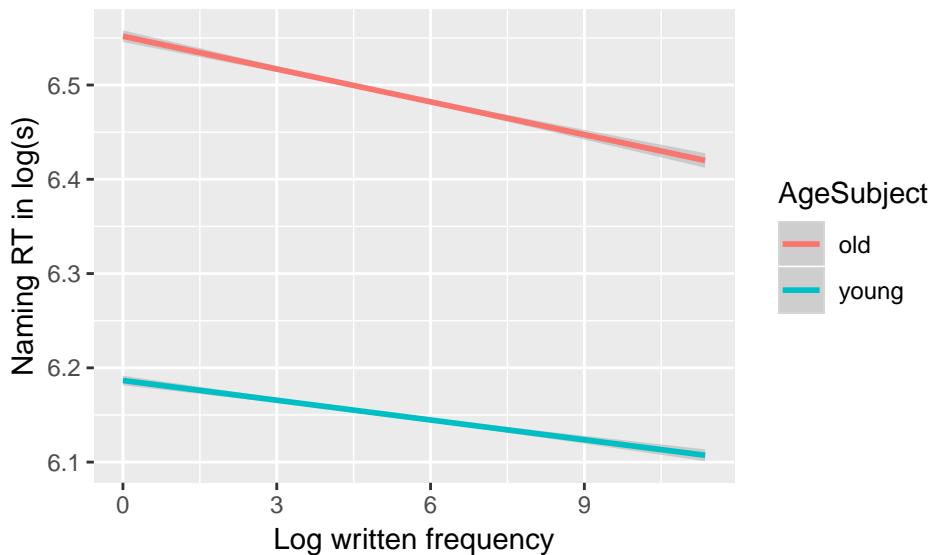
- What does it mean that this coefficient is positive?
- For this regression model including an interaction, what is the model for an observation with `WrittenFrequency=3` and `AgeSubject=='old'`? Solution.
- What is the model for an observation with `WrittenFrequency=3` and `AgeSubject=='young'`? Solution.

### 3.3.3 Plotting interactions

In general, making plots is indispensable for interpreting interactions. It is possible, with practice, to interpret interactions from the regression table, but examining a good plot is usually also necessary and much faster.

In later chapters we will cover how to actually visualize model predictions—exactly what the model predicts for different combinations of predictor values. You can usually get a reasonable approximation of this by making the relevant empirical plot, such as:

```
english %>% ggplot(aes(WrittenFrequency, RTnaming)) +
  geom_smooth(method='lm', aes(color=AgeSubject)) +
  xlab('Log written frequency') +
  ylab('Naming RT in log(s)')
```



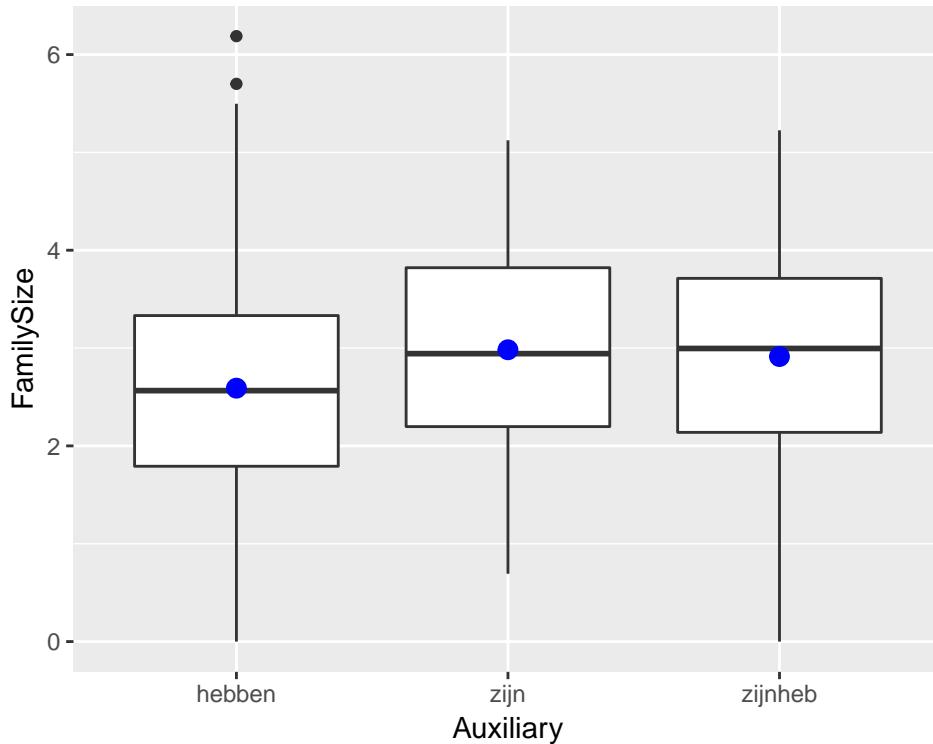
It is often better to use empirical plots to visualize interactions—even though, strictly speaking, you are not plotting the model's predictions.

- **Pros:** Empirical plots are more intuitive, and if you have a robust effect it should probably show up in an empirical plot.
- **Cons:** Empirical plots don't show actual model predictions, and in particular don't control for the effects of other predictors.

### 3.3.4 Categorical factors with more than two levels

We are often interested in categorical predictors with more than two levels. For example, for the Dutch regularity data, we might wonder whether the size of a verb's morphological family size is affected by what auxiliary it takes in the past tense.

```
regularity %>% ggplot(aes(Auxiliary, FamilySize)) +
  geom_boxplot() +
  stat_summary(fun.y="mean", geom="point", color="blue", size=3)
```



The relevant variable, `Auxiliary`, has three levels. Let's see how this kind of variable is dealt with in a regression model.

#### Exercise

1. Fit a regression model predicting `FamilySize` from `Auxiliary`.
2. What does the intercept ( $\beta_0$ ) represent?
3. What do the two coefficients for `Auxiliary` ( $\beta_1, \beta_2$ ) represent?

Hint: Compare  $\beta$  coefficients with group means, which you can check using `summarise()` from `dplyr`.

Solution to (1):

```
m4 <- lm(FamilySize ~ Auxiliary, regularity)
summary(m4)
```

```
##
## Call:
## lm(formula = FamilySize ~ Auxiliary, data = regularity)
##
## Residuals:
```

```

##      Min     1Q   Median     3Q    Max
## -2.9133 -0.7982 -0.0250  0.7442  3.5983
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.59000   0.04902 52.839 <2e-16 ***
## Auxiliaryzijn    0.39274   0.26780  1.467   0.1430
## Auxiliaryzijnheb 0.32329   0.12594  2.567   0.0105 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.177 on 697 degrees of freedom
## Multiple R-squared:  0.01169, Adjusted R-squared:  0.008855
## F-statistic: 4.123 on 2 and 697 DF, p-value: 0.0166

```

Solution to (2): The value of `FamilySize` when `Auxiliary`=“hebben”.

Solution to (3): The predicted difference in `FamilySize` between `Auxiliary`=“zijn” and “hebben”, and between `Auxiliary`=“zijnheb” and “hebben”.

### 3.3.5 Releveling factors

It's often useful for conceptual understanding to change the ordering of a factor's levels. For the `regularity` example, we could make `zijn` the base level of the `Auxiliary` factor:

```

regularity$Auxiliary <- relevel(regularity$Auxiliary, "zijn")

m5 <- lm(FamilySize ~ Auxiliary, regularity)
summary(m5)

##
## Call:
## lm(formula = FamilySize ~ Auxiliary, data = regularity)
##
## Residuals:
##      Min     1Q   Median     3Q    Max
## -2.9133 -0.7982 -0.0250  0.7442  3.5983
##
## Coefficients:
##                  Estimate Std. Error t value Pr(>|t|)
## (Intercept)      2.98274   0.26328 11.329 <2e-16 ***
## Auxiliaryhebben -0.39274   0.26780 -1.467   0.143
## Auxiliaryzijnheb -0.06946   0.28771 -0.241   0.809
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.177 on 697 degrees of freedom
## Multiple R-squared:  0.01169, Adjusted R-squared:  0.008855
## F-statistic: 4.123 on 2 and 697 DF, p-value: 0.0166

```

#### Questions:

- What is the interpretation of the intercept and the two `Auxiliary` coefficients in this new model?

## 3.4 Linear regression assumptions

Up to now, we have discussed regression models without worrying about the assumptions that are made by linear regression, about your data and the model. We will cover six main assumptions, the first four have to do with the form of the model and errors:

1. Linearity
2. Independence of errors
3. Normality of errors
4. Constancy of errors (*homoscedasticity*)

followed by two assumptions about the predictors and observations:

5. Linear independence of predictors
6. Observations have roughly equal influence on the model

We'll discuss each in turn.

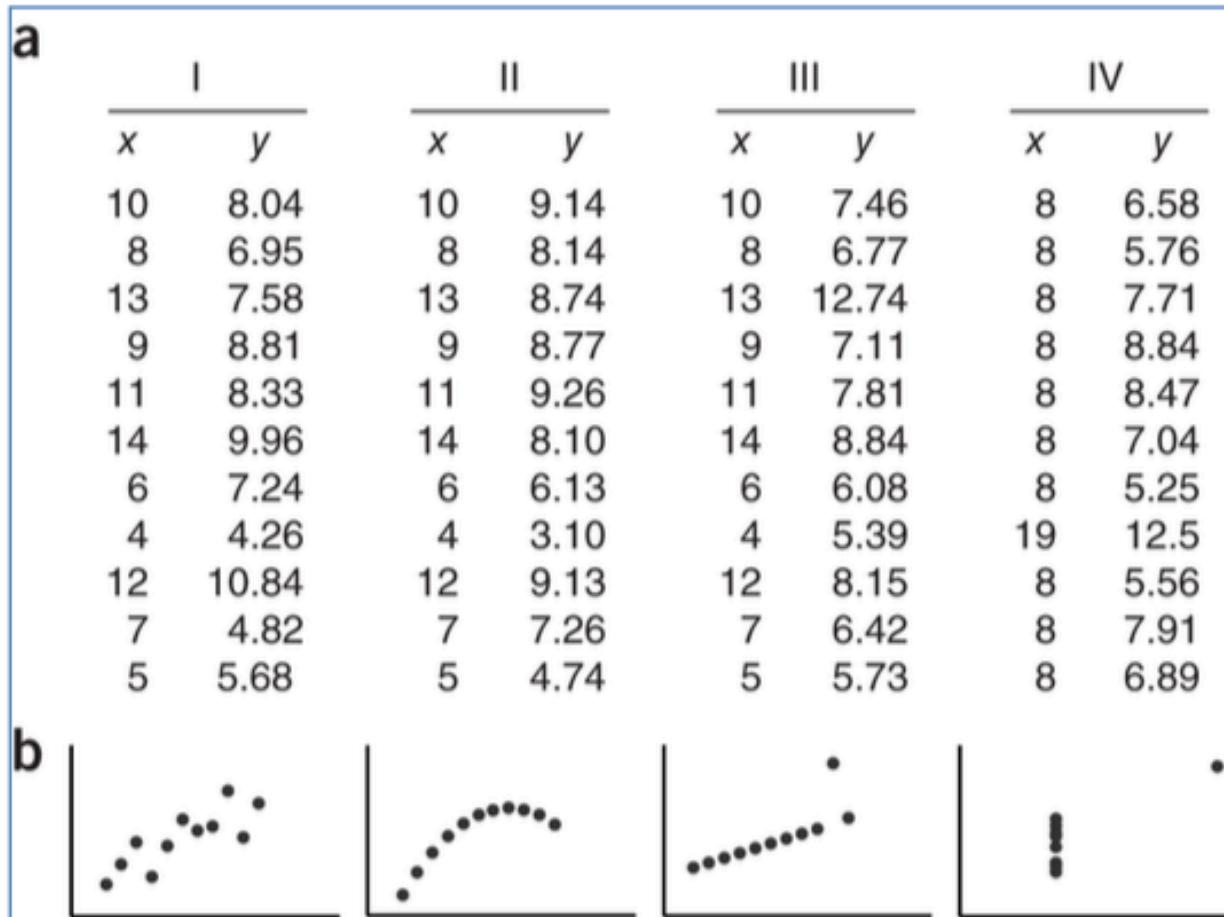
Our presentation of regression assumptions and diagnostics is indebted to Chapters 4, 6 and 9 of Chatterjee and Hadi (2012), where you can find more detail.

### 3.4.1 Visual methods

Visualization is crucial for checking model assumptions, and for data analysis in general. A famous example illustrating this is *Anscombe's quartet*: a set of four small datasets of  $(x, y)$  pairs with:

- The same mean and variance for  $x$
- The same mean and variance for  $y$
- A correlation( $x, y$ ) = 0.816
- The same regression line ( $y = 3 + 0.5 \cdot x$ )

in each case—and yet the datasets show qualitatively different patterns, as can be seen by plotting  $y$  against  $x$ :



(Source: unknown, but definitely taken from somewhere)

With more than one predictor it becomes difficult to check regression assumptions by just plotting the data, and visual methods such as *residual plots* (presented below) are crucial.

### 3.4.2 Assumption 1: Linearity

The first assumption of a linear regression model is that the relationship between the response ( $Y$ ) and predictors ( $X_i$ ) is... linear.

While obvious, this assumption is very important: if it is violated, the model's predictions can be in serious error.

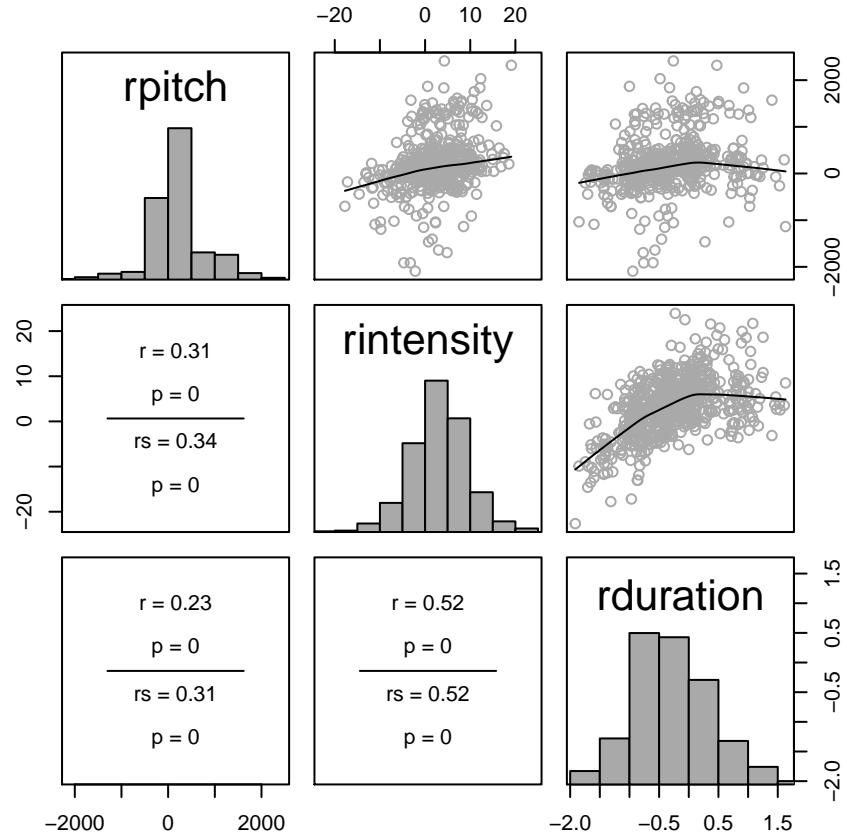
The linearity assumption can be partially checked by making a scatterplot of  $Y$  as a function of each predictor  $X_i$ . It is hard to exhaustively check linearity for MLR, because nonlinearity might only become apparent when  $Y$  is plotted as a function of several predictors.

#### Example

Consider relative pitch, intensity, and duration in the `alternatives` dataset.

We can make *pairwise plots* of these variables using the `pairsCor.fnc()` function `languageR`, to see if any of these variables might be a function of the other two.

```
pairscor.fnc(with(alt, cbind(rpitch, rintensity, rduration)))
```

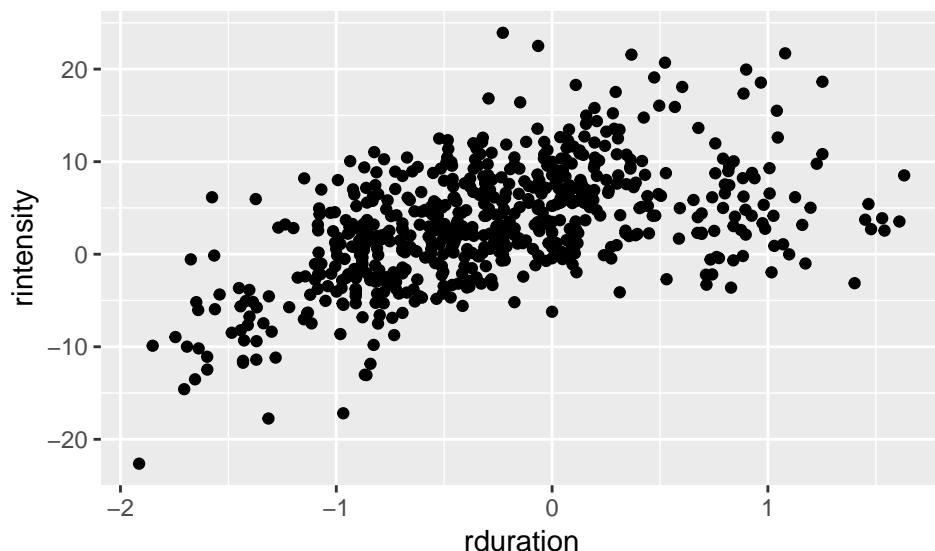


### Questions:

- Is this the case?

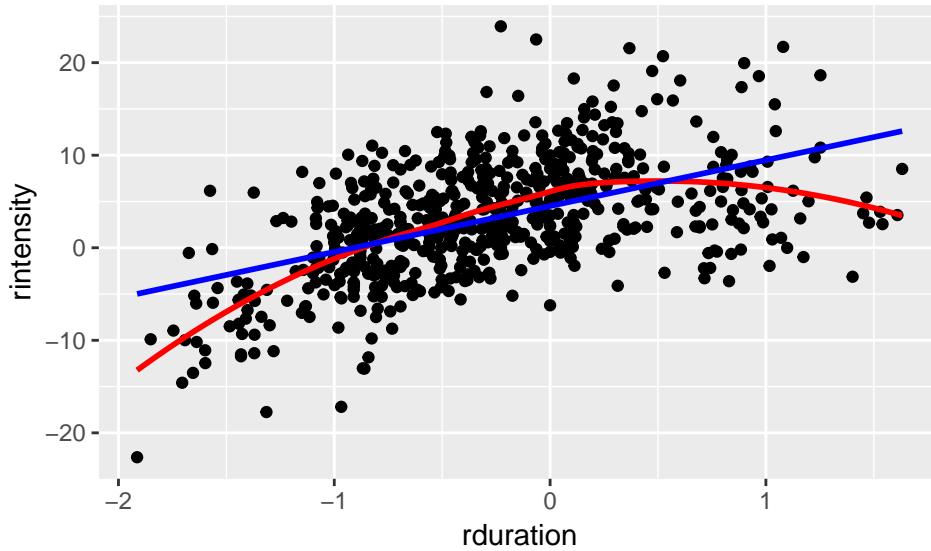
Let's examine the relationship between relative duration and relative intensity more closely:

```
alt %>% ggplot(aes(rduration, rintensity)) +
  geom_point()
```



We can try to fit a line to this data, but if we compare to using a nonlinear smoother, it seems clear that the relationship is not linear:

```
alt %>% ggplot(aes(rduration, rintensity)) +
  geom_point() + geom_smooth(col='red', se=F) +
  geom_smooth(method="lm", col="blue", se=F)
```



In particular, it looks like there is a quadratic trend. This means that we can in fact fit a linear regression, we just need to include coefficients for both `rduration` and its square, like so:

```
mq <- lm(rintensity ~ rduration + I(rduration^2), alt)
summary(mq)
```

```
##
## Call:
## lm(formula = rintensity ~ rduration + I(rduration^2), data = alt)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -16.3199  -3.5197  -0.2448   2.9515  19.1480 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept)  5.8280    0.2622  22.228  <2e-16 ***
## rduration   3.8841    0.3263  11.904  <2e-16 ***
## I(rduration^2) -3.1381    0.3429  -9.151  <2e-16 ***
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5.05 on 627 degrees of freedom
## Multiple R-squared:  0.358, Adjusted R-squared:  0.3559 
## F-statistic: 174.8 on 2 and 627 DF,  p-value: < 2.2e-16
```

We will cover more nonlinear functions of predictors in a later chapter. The important point here is that a model with just `rduration` as a predictor would have violated the linearity assumption, but a model with both `rduration` and `rduration^2` as predictors doesn't (arguably).

### 3.4.3 Assumption 2: Independence of errors

All regression equations we have considered contain an *error* or *residual* term:  $\epsilon_i$  for the  $i^{\text{th}}$  observation. A crucial assumption is that these  $\epsilon_i$  are **independent**: knowing the error for one observation shouldn't tell you anything about the error for another observation.

Unfortunately, violations of this assumption are endemic in realistic data. The simplest example is in time series, or longitudinal data—such as pitch measurements taken every 10 msec in a speech signal.

#### Questions:

- Can you think of why this might be the case?

In linguistics and psycholinguistics, violations of the independence assumption are common because most datasets include multiple observations per participant or per item (or per word, etc.). Crucially, violations of the independence assumption are often *anti-conservative*: CIs will be too narrow and  $p$ -values too small if the lack of independence of errors is not taken into account by the model.

Some solutions to these issues:

- **Paired-t-tests**, where applicable (binary predictor; two measures per participant)
- **Mixed-effects regression** (more general solution, major focus later this term)

Until we cover mixed-effects regression, we will be getting around the fact that the independence-of-errors assumption usually doesn't hold for linguistic data, in one of two ways:

1. Selectively using datasets where this assumption **does** hold, such as `regularity`.
2. Analyzing datasets where this assumption does **not** hold, such as `tapping` or `english`, using analysis methods that do assume independence of errors (such as linear regression), with the understanding that our regression models are probably giving results that are “wrong” in some sense.

### 3.4.4 Assumption 3: Normality of errors

The next major assumption is that the errors  $\epsilon_i$  are normally distributed, with mean 0 and a fixed variance. This assumption is impossible to check directly, because we never observe the true *errors*  $\epsilon_i$ , only the *residuals*  $e_i$ . The residuals are no longer normally distributed (even if the errors are), because some observations will be more influential than others in determining the fitted responses  $\hat{y}_i$  when fitting the least-squares estimates of the regression coefficients.

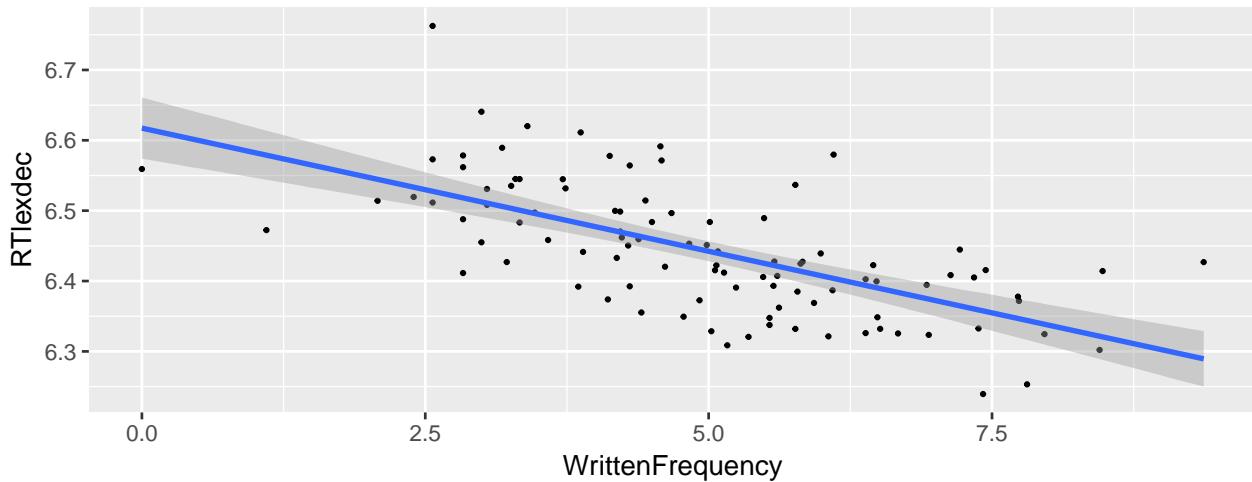
#### Example

```
young <- filter(english, AgeSubject=='young')

set.seed(2903)
young_sample <- young %>% sample_n(100)

ggplot(young_sample, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth(method="lm") +
  ggtitle("Linear regression line and 95% CI")
```

### Linear regression line and 95% CI



The width of the confidence interval increases for points further from (average of `WrittenFrequency`, average of `RTlexdec`), because these points are more influential, causing the variance of the residuals to increase—thus, the variance is not constant.

In order to correct for non-normality, the residuals are transformed in a way which accounts for the different influence of different observations (see e.g. Chatterjee and Hadi (2012) 4.3), to *studentized* or *standardized residuals*.<sup>1</sup> (In R, by applying `rstudent` or `rstandard` to a fitted model.)

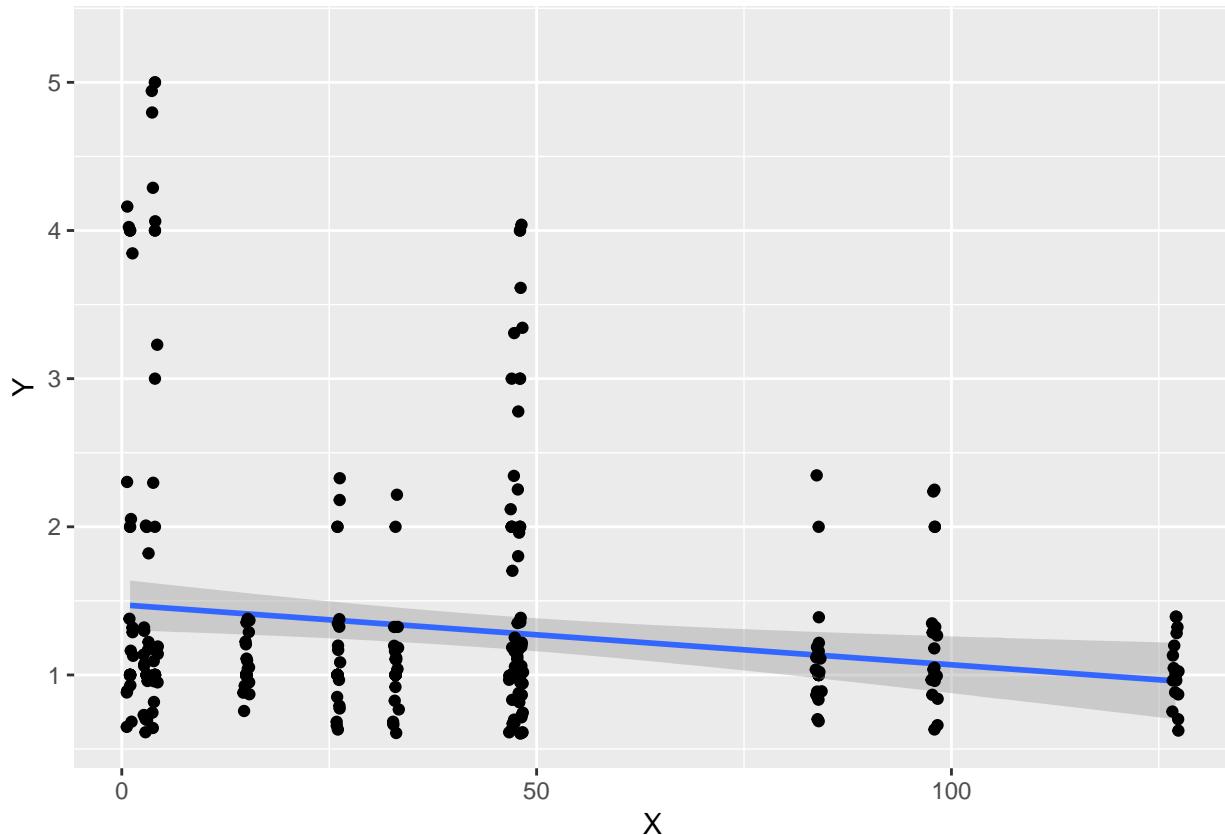
In general, we **check assumptions about errors by examining the distribution of standardized residuals**. This is because if the normality of errors assumption holds, then the standardized residuals will be normally distributed with mean 0 and fixed variance. So if they are not, we know the normality of errors assumption does not hold. (If they are, it's not a guarantee that the normality of errors assumption holds, but we hope for the best.)

### Example

This exercises uses the `halfrhyme` data, briefly described here. Let's abstract away from what the variables actually mean, and just think of them as  $Y$  and  $X$ :

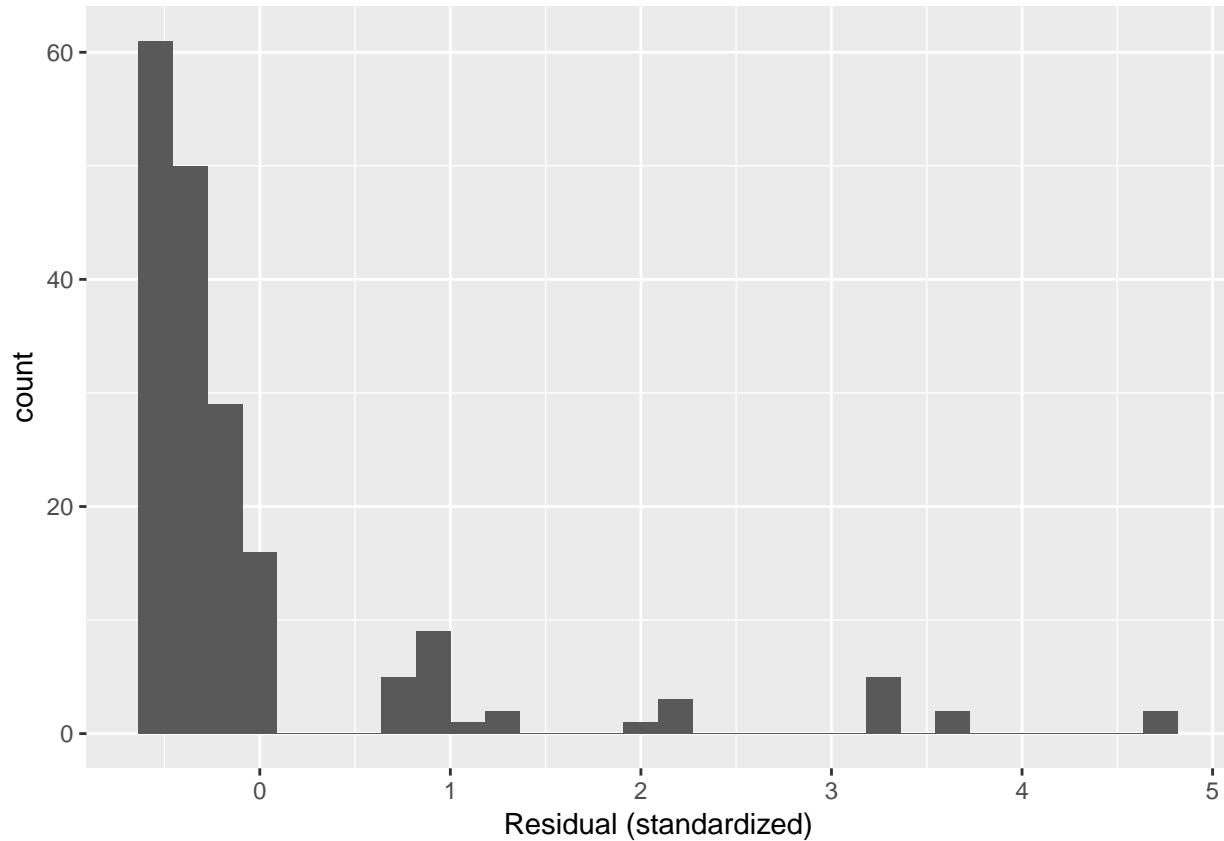
```
ggplot(aes(x=cohortSize, y=rhymeRating), data=filter(halfrhyme, conditionLabel=='bad')) +
  geom_point() + geom_smooth(method='lm') +
  geom_jitter() + xlab("X") + ylab("Y")
```

<sup>1</sup>The studentized and standardized residuals, or “externally studentized” and “internally studentized” residuals (in Chatterjee and Hadi (2012)), differ slightly in how they estimate the error variance: a leave-one-out estimate versus an estimate using all observations. This difference shouldn't matter much except when certain observations are highly influential or in small datasets.



The distribution of the standarized residuals for the regression of  $Y$  as a function of  $X$  is:

```
halfrhyme.sub <- filter(halfrhyme, conditionLabel=='bad' & !is.na(cohortSize))
mod <- lm(rhymeRating ~ cohortSize, data=halfrhyme.sub)
halfrhyme.sub$resid <- rstandard(mod)
ggplot(aes(x=resid), data=halfrhyme.sub) + geom_histogram() + xlab("Residual (standardized)")
```



#### Questions:

- Why do the residuals have this distribution?

This example illustrates probably the most common source of non-normal residuals: a highly non-normal distribution of the predictor or response.

##### 3.4.4.1 Effect and solution

Non-normality of residuals is a pretty common violation of regression assumptions. How much does it actually matter? Gelman and Hill (2007) (p. 46) argue “not much”, at least in terms of the least-squares estimates of the regression line (i.e., the regression coefficient values), which is often what you are interested in.

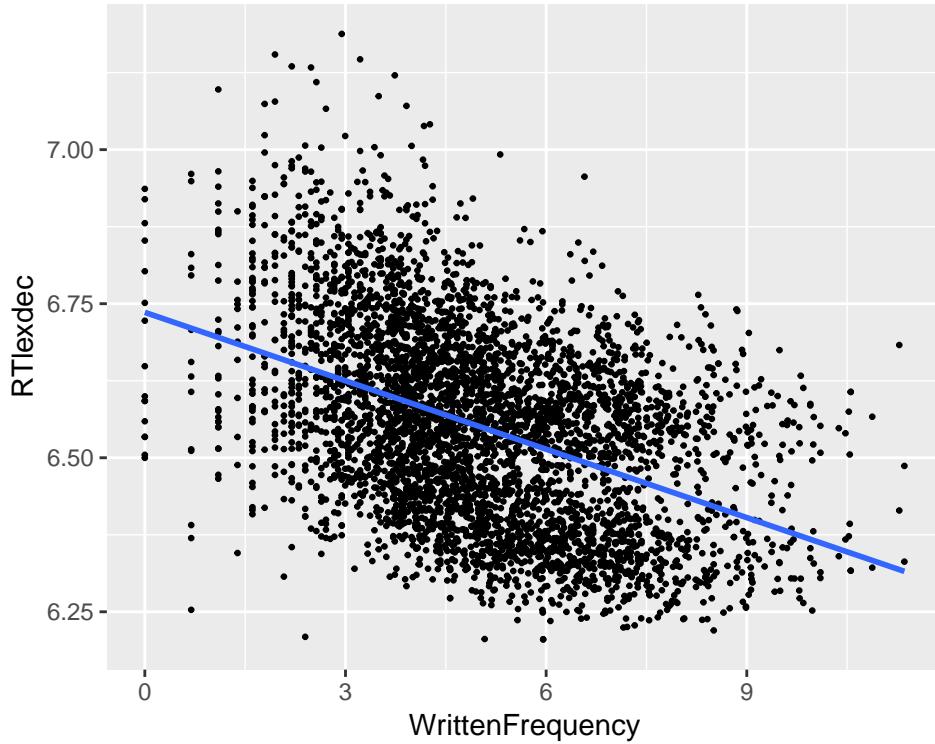
However, non-normality of residuals, especially when severe, can signal other issues with the data, such as the presence of outliers, or the predictor or response being on the same scale. (Example: using non-log-transformed word frequency as a predictor.) Non-normality of residuals can often be dealt with by **transforming the predictor or response** to have a more normal distribution (see Sec. 3.4.7).

Non-normality of residuals can also signal other errors, such as an important predictor missing.

#### Exercise

1. Using the `english` data, plot `RTlexdec` as a function of `WrittenFrequency`, and add a linear regression line.

```
ggplot(english, aes(x = WrittenFrequency, y = RTlexdec)) +
  geom_point(size = 0.5) +
  geom_smooth(method = "lm", se=F)
```



2. Do you think the residuals of this model are normally distributed? Why/why not?
3. Now plot a histogram of the standardized residuals of the mode. Does the plot confirm your first impressions?

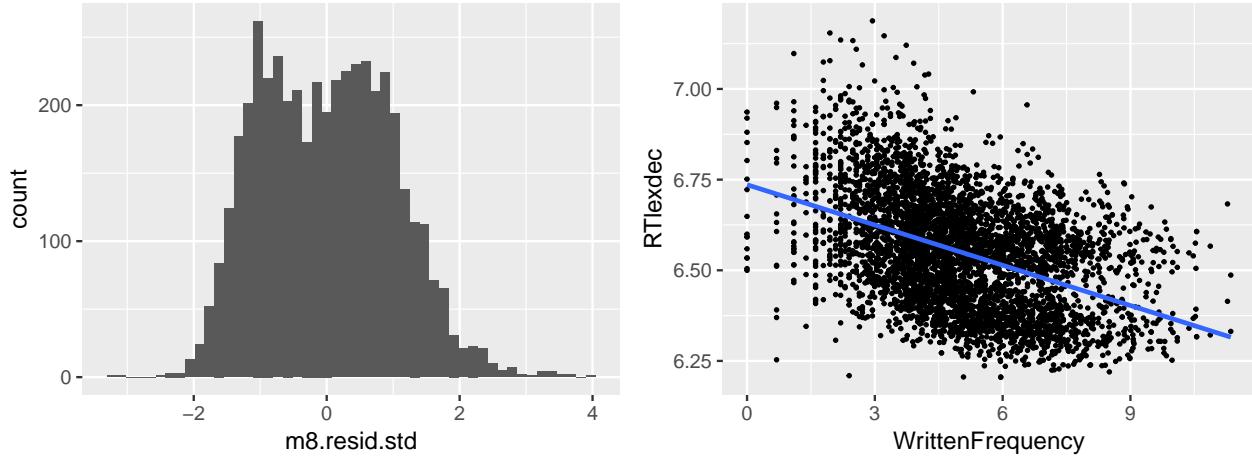
```
m8 <- lm(RTlexdec ~ _____, english)
m8.resid.std <- rstandard(_____)
hist(_____, breaks = 50)

day9_plt1 <- ggplot(english, aes(WrittenFrequency, RTlexdec)) +
  geom_point(size=0.5) +
  geom_smooth(method="lm", se=F)

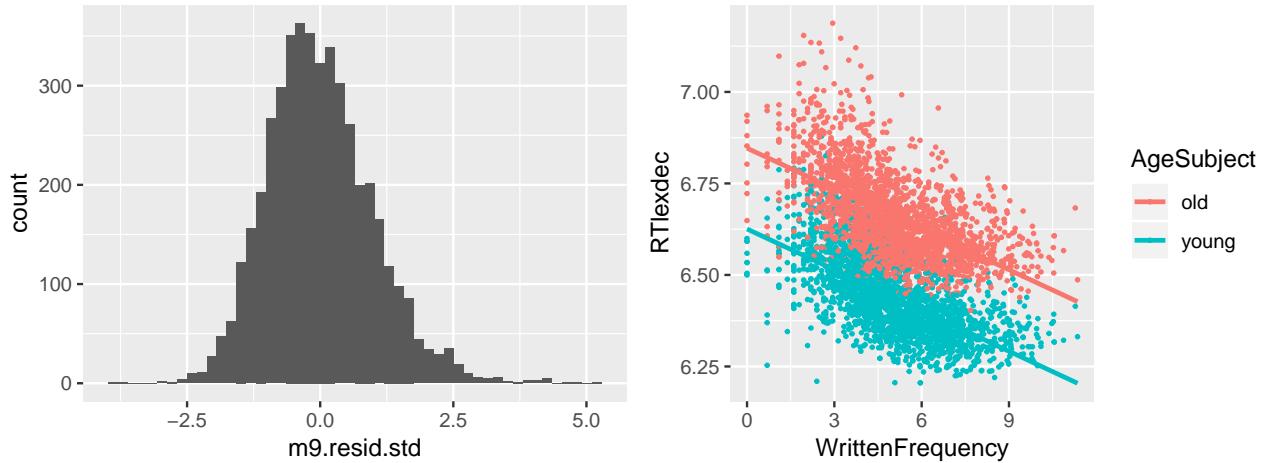
m8 <- lm(RTlexdec~WrittenFrequency, english)
m8.resid.std <- rstandard(m8)

day9_plt2 <- ggplot(data.frame(m8.resid.std), aes(x = m8.resid.std)) +
  geom_histogram(bins = 50)

grid.arrange(day9_plt2, day9_plt1, ncol = 2)
```



4. Now add `AgeSubject` to the model, and plot a histogram of its standardized residuals. What has changed? Why so?



This exercise shows one reason that examining the residual distribution is useful. If we didn't already know what the missing predictor was, the non-normality of the residual distribution gives us a way to look for an explanatory variable. (Look at observations in each mode of the distribution, see what they have in common.)

### 3.4.5 Assumption 4: Constancy of variance

*Homoscedasticity* is one of the trickier regression assumptions to think about: the assumption that  $\epsilon_i$  is normally distributed with the same variance, across all values of the predictors.

For example, in our example modeling reaction time as a function of subject age and word frequency, it is assumed that the amount of variability in reaction time is similar for old speakers and young speakers, for high frequency words and young speakers, for observations of high frequency words for old speakers, and so on.

In the example above from the `halfrhyme` data, the homoscedasticity assumption is violated: lower values of  $\hat{y}$  show higher variance in the residuals.

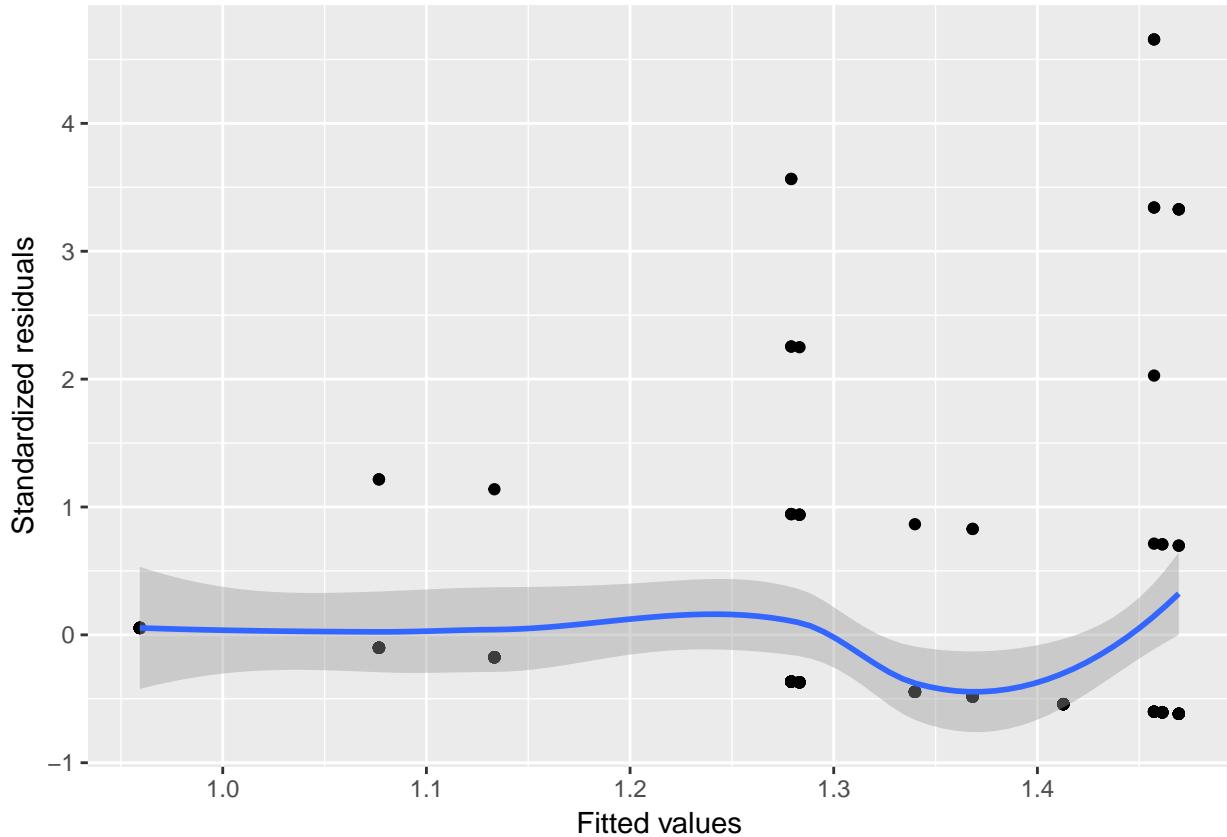
In this case, the model shows *heteroscedasticity*.

If homoscedasticity holds, **then** the standardized residuals are uncorrelated with the predictor values, and with the fitted values, and there should be a constant spread (variance) of residual values (y-axis) for each

fitted or predictor value (x-axis). Thus, it is common to plot (standardized) residuals versus fitted values and versus predictors. (The fitted values-residuals plot is one of the diagnostic plots that shows up if you `plot(mod)` in R, where `mod` is a fitted model.) The desired pattern is a flat line, with the same variance for different x-axis values.

For the `halfrhyme` data, the fitted value-residuals plot looks like:

```
halfrhyme.sub <- filter(halfrhyme, conditionLabel=='bad' & !is.na(cohortSize))
mod <- lm(rhymeRating ~ cohortSize, data=halfrhyme.sub)
halfrhyme.sub$resid <- rstandard(mod)
halfrhyme.sub$fitted <- fitted(mod)
ggplot(aes(x=fitted, y=resid), data=halfrhyme.sub) + geom_point() + geom_smooth() + xlab("Fitted values")
```



There is greater variance for higher fitted values, indicating heteroscedasticity.

#### 3.4.5.1 Effect and solution

In general, estimates of least-squares coefficients in the presence of heteroscedasticity are unbiased, but standard errors will be under- or over-estimated. This means that confidence intervals will be too narrow/wide and *p*-values too low/high.

Heteroscedasticity is endemic in some types of data, such as from lexical statistics (Baayen (2008), p. 35). In other types of data, such as economic data, heteroscedasticity is so common that dealing with it is a primary concern in statistical analysis. Heteroscedasticity is discussed less frequently than other regression assumptions for linguistic data, but it is unclear whether this is because heteroscedasticity is less common than in other types of data or just has not been focused on by language scientists.

One can often correct for heteroscedasticity by using various transformations of the response and predictors to get better estimates (Chatterjee and Hadi (2012), Ch. 4). For example, in the `halfrhyme` example, it

turns out that a stronger effect of  $X$  on  $Y$  (lower  $p$ -value) can be detected once variance is stabilized.

### 3.4.6 Interim summary

- **Linearity**
  - Serious violation if not met.
  - Fit data with non-linear trend (e.g. quadratic)
  - Transformed predictor/response to normality (e.g. log-transform)
- **Independence of error:**
  - In linguistic data: use mixed-effects regression<sup>2</sup>
- **Normality of errors:**
  - Not too serious violation if not met, but may signal issues with model/data
  - Remove outliers; transform  $X/Y$  to normality
- **Constancy of variance:**
  - Not commonly checked in linguistic data
  - Leads to uncertain regression estimates
  - Transform predictor/response to normality

### 3.4.7 Transforming to normality

Normality of the distribution of the response and predictors ( $Y$  and  $X_i$ ) is **not** an assumption of linear regression. This is a common misconception, perhaps because normality is an assumption of other basic statistical inference tools, such as  $t$ -tests.

However, there is still good reason to be circumspect if  $Y$  or  $X_i$  are not normally distributed, because this can often lead to violations of regression assumptions. This is why it is recommended to transform the predictors and response to normality to fix violations of the linearity, normality of errors, and homoscedasticity assumptions. Because non-normality of  $Y$  or  $X_i$  can easily lead to violations of regression assumptions, it is sometimes recommended to transform them to normality just to be safe. This makes it less likely that a regression assumption will be violated, but also changes the interpretation of the transformed variable, which may make it harder to interpret the model's results.

For linguistic data, *logarithmic* transformations are often useful when working with skewed distributions, because many kinds of linguistic data are roughly *log-normally* distributed, meaning the log-transformed variable is normally distributed. Some examples:

- Lexical statistics (e.g. lexical frequency, probability)
- Reaction times (e.g. naming latencies)
- Duration measures in phonetics (syllable, phrase durations)

Other transformations besides log are also used: reaction times are sometimes inverse or inverse-log-transformed ( $1/RT$ ,  $\log(1/RT)$ ), and durations are sometimes square-root-transformed.

---

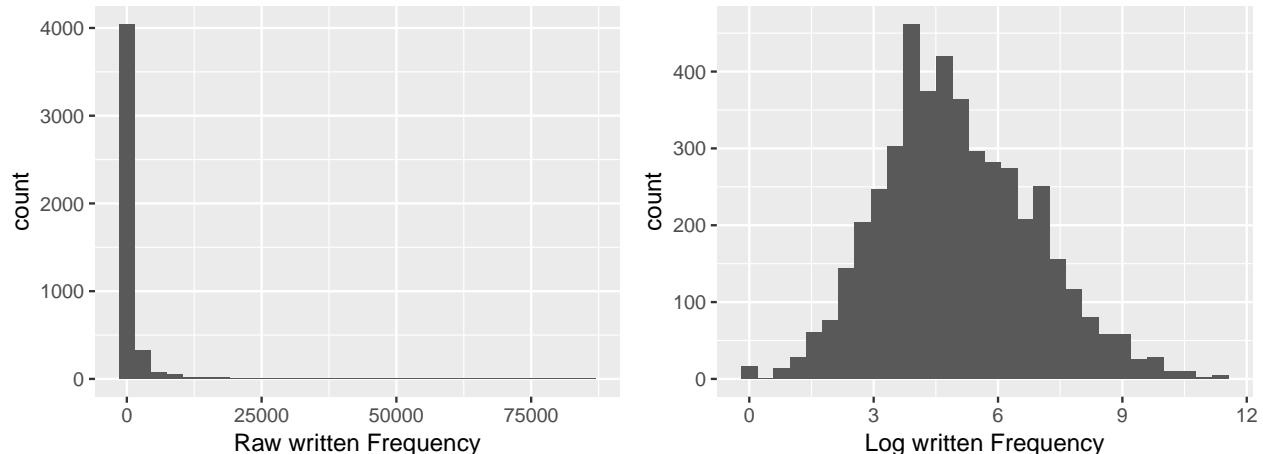
<sup>2</sup>Or another method that accounts for non-independence of errors, such as repeated measures ANOVA.

### Example: Distribution of raw vs. log lexical frequency

```
english$WrittenFrequency_raw <- exp(english$WrittenFrequency)
english$WrittenFrequency_log <- english$WrittenFrequency

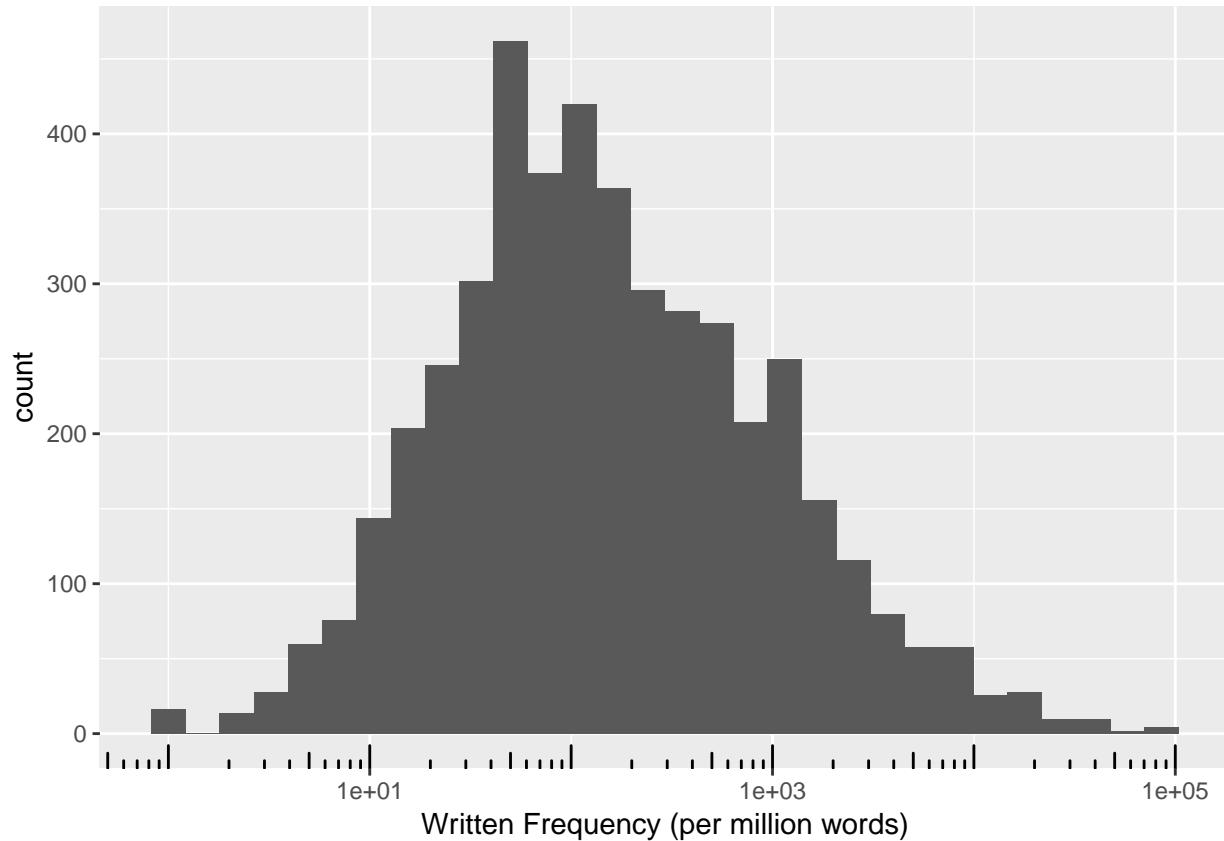
day10_plt1 <- ggplot(english, aes(x = WrittenFrequency_raw)) +
  geom_histogram() +
  xlab("Raw written Frequency")
day10_plt2 <- ggplot(english, aes(x = WrittenFrequency_log)) +
  geom_histogram() +
  xlab("Log written Frequency")

grid.arrange(day10_plt1, day10_plt2, ncol = 2)
```



R note: ggplot functions such as `scale_x_log10()` can be used to plot data in its raw units on a log scale, which is often more interpretable. Ex:

```
ggplot(english, aes(x = WrittenFrequency_raw)) + geom_histogram() + scale_x_log10() +
  xlab("Written Frequency (per million words)") + annotation_logticks(sides='b')
```



### Exercise

Consider the raw and log frequency measures for the `english` dataset, for young speakers:

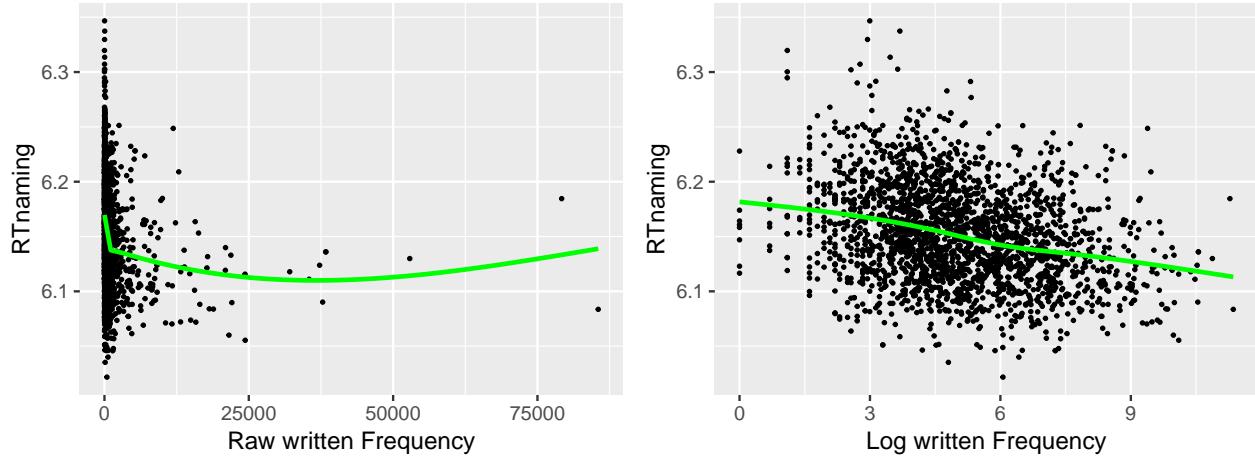
```
english$WrittenFrequency_raw <- exp(english$WrittenFrequency)
english$WrittenFrequency_log <- english$WrittenFrequency

young <- filter(english, AgeSubject=="young")
```

1. Plot RTnaming as a function of raw written frequency, and of log written frequency, with a green LOESS (smooth) regression line added to each plot (using `geom_smooth(color='green')`).

```
plt5 <- ggplot(young, aes(WrittenFrequency_raw, RTnaming)) +
  geom_point(size=0.5) +
  xlab('Raw written Frequency') +
  geom_smooth(se=F, color="green")

plt6 <- ggplot(young, aes(WrittenFrequency_log, RTnaming)) +
  geom_point(size=0.5) +
  xlab('Log written Frequency') +
  geom_smooth(se=F, color="green")
grid.arrange=plt5, plt6, ncol = 2)
```



2. Is the linearity assumption met in each case?

### 3.4.8 Assumption 5: Linear independence of predictors

A crucial assumption of linear regression is that the predictors are *linearly independent*, meaning it isn't possible to write one predictor as a linear function of the others. If you can do so, it's impossible to disentangle the effect of different predictors on the response. For example, temperatures in farenheit and centigrade are related by a linear function ( $F = 9/5C + 32$ ), so  $F$  and  $C$  are linearly dependent.

Another way of thinking of linear dependence is that in a linear regression where you predict one predictor as a function of the others, the  $R^2$  would be 1. Linear dependence of predictors will either give a model error or a weird model output if you fit a linear regression in R, because the math to find least-squares estimates of coefficients doesn't work out if there is linear dependence: there is no longer a unique optimal solution. (For example, if the slope of  $F$  and  $C$  in a model were  $\beta_1 = 2$  and  $\beta_2 = 0$ , then a different model using slopes of 0 and  $10/9$  would also work.)<sup>3</sup>

#### Exercise

Define a new variable in the `english` dataset that is the average of `WrittenFrequency` and `Familiarity`, then fit a linear regression of `RTlexdec` as a function of this new variable, `WrittenFrequency`, and `Familiarity`. What looks odd in the model output?

### 3.4.9 Collinearity

Full linear dependence of predictors is usually a sign that something is conceptually wrong with your data, or your model structure. In the farenheit/celsius example, it doesn't make conceptual sense to have both as predictors of anything.

However, it is very common for there to be **partial** dependence between predictors—that is,  $0 < R^2 < 1$  when you regress one predictor on the others. This is called *multicollinearity*, or just *collinearity*.<sup>4</sup> Collinearity is ubiquitous in linguistic data, and can significantly affect the estimates and interpretations of regression

<sup>3</sup>In technical terms: the *design matrix*  $X$  (where each column is the values of predictor  $i$ , across all observations) is not *full rank*, and the matrix  $X^T X$ , which is used to find the least-squares regression coefficient estimates, is *singular* (alternatively, there is a *singularity*) Even if you don't care about the math of linear regression, it is useful to know that when you encounter these terms in R output it usually means some variable can be perfectly predicted from the others, and this is a problem.

<sup>4</sup>Note that you can have  $R^2 < 1$  even when one predictor is completely predictable from other predictors—as long as this dependence is not a linear function. Sometimes it does not make conceptual sense to have two predictors related in this way in a model (e.g. VOT and  $\log(VOT)$ ), while in other settings it does (e.g. linear and quadratic terms for the same variable).

coefficients—particularly when collinearity is “high” (say,  $|R| > 0.8$ ) However, (high) **collinearity is not a violation of the assumptions of linear regression!**

This figure may be useful to get an intuitive sense of what collinearity is, if you think of X1-X4 as four predictors which affect Y, and may be highly interrelated (right figure) or independent (left figure).

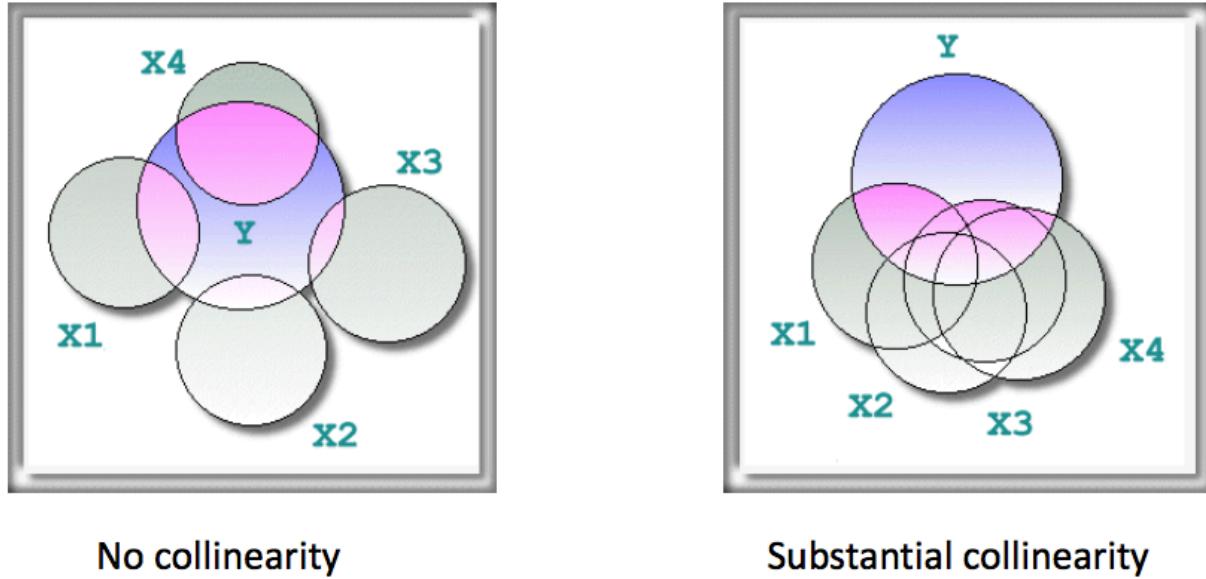


Figure 3.1:

(Source: [http://www.creative-wisdom.com/computer/sas/collinear\\_stepwise.html](http://www.creative-wisdom.com/computer/sas/collinear_stepwise.html))

There can be collinearity among several predictors—hence the term “multicollinearity”—where one predictor is a linear function of several others.

### Exercise

Suppose we are trying to predict the duration of the first vowel of every word in a dataset of conversational speech, using these four predictors:

1.  $\log(\text{speaking rate})$
2.  $\log(\# \text{ of syllables in the word})$
3.  $\log(\text{duration of word})$
4.  $\log(\text{first syllable duration})$

where “speaking rate” is defined as “syllables per second”.

Why are these predictors linearly dependent?

### Exercise

Let us simulate a small-scale experiment ( $n = 25$ ), using data from young speakers in the `english` dataset. We measure RT, take `Familiarity` norms from a previous study, and would like to control for the following variables:

- `WrittenFrequency`

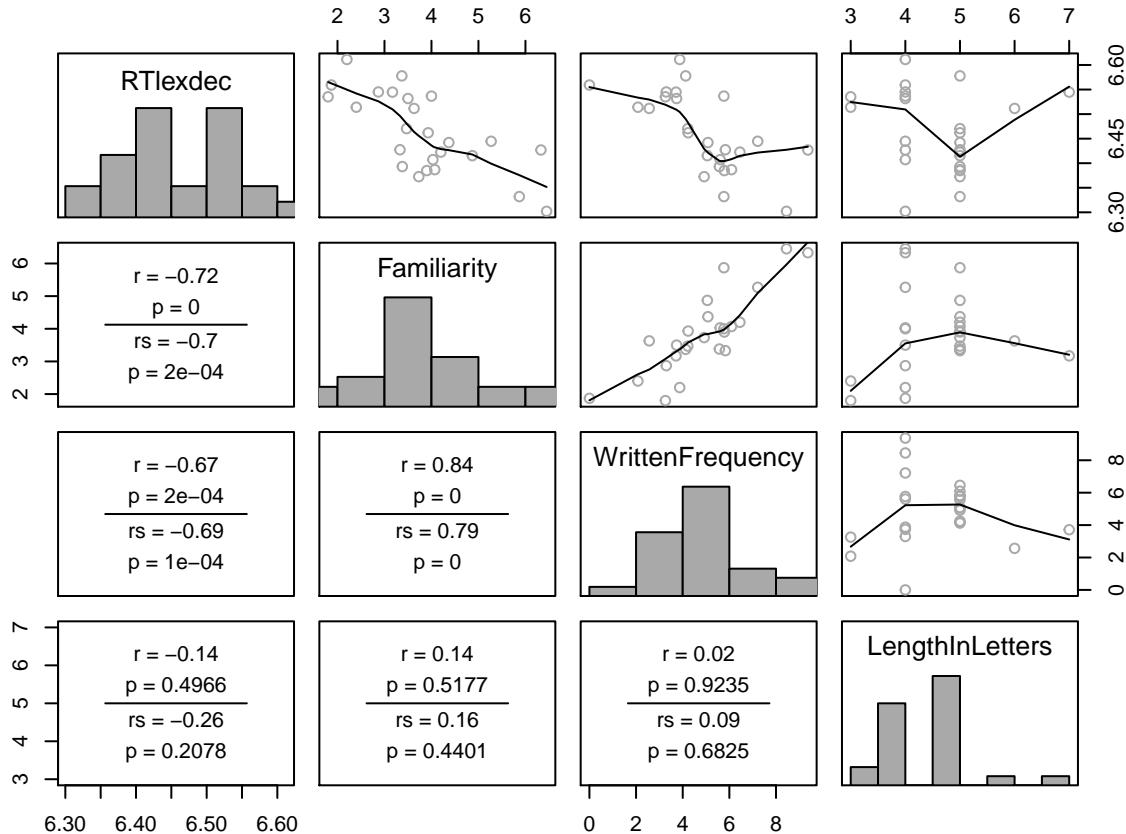
- LengthInLetters

Because both are expected to correlate with Familiarity, and to affect RT.

We first examine whether there is collinearity among predictors, then assess whether familiarity affects lexical decision RT.

```
young <- filter(english, AgeSubject == "young")
set.seed(2903)
d <- sample_n(young, 25)

pairscor.fnc(dplyr::select(d, RTlexdec, Familiarity, WrittenFrequency, LengthInLetters))
```



### Questions:

- What correlations are present in the data? Is there collinearity among predictors?

Now, fit a model of just familiarity on RT:

```
m.fam <- lm(RTlexdec ~ Familiarity, d)
summary(m.fam)
```

```
##
## Call:
## lm(formula = RTlexdec ~ Familiarity, data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.094956 -0.035753  0.002142  0.050448  0.093090
##
## Coefficients:
```

```

##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.645471  0.038231 173.83 < 2e-16 ***
## Familiarity -0.047697  0.009502  -5.02 4.44e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05699 on 23 degrees of freedom
## Multiple R-squared:  0.5228, Adjusted R-squared:  0.502
## F-statistic:  25.2 on 1 and 23 DF,  p-value: 4.442e-05

```

This model finds a highly significant effect of word familiarity on RT.

#### Questions:

- What is the interpretation of this model—what relationship is it capturing from the grid of plots above?

Now,

```
m.fam_freq <- lm(RTlexdec ~ Familiarity + WrittenFrequency + LengthInLetters, d)
summary(m.fam_freq)
```

```

##
## Call:
## lm(formula = RTlexdec ~ Familiarity + WrittenFrequency + LengthInLetters,
##      data = d)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.090416 -0.040823 -0.008442  0.042555  0.094739
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.66953   0.07094  94.013 <2e-16 ***
## Familiarity -0.03333   0.01821  -1.831  0.0814 .
## WrittenFrequency -0.01017   0.01117  -0.911  0.3728
## LengthInLetters -0.00643   0.01410  -0.456  0.6530
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.05838 on 21 degrees of freedom
## Multiple R-squared:  0.5429, Adjusted R-squared:  0.4776
## F-statistic: 8.313 on 3 and 21 DF,  p-value: 0.0007782

```

This model finds that **none** of the three variables significantly affect RT

#### Questions:

- Why has the **Familiarity** effect changed between the two models? (Examine both the coefficient estimates and SEs.)
- How is it possible that none of the three variables significantly affect RT, but together they do predict half the variation in RT ( $R^2 = 0.54$ )?

This kind of situation is called a *credit assignment problem*: we can tell that some combination of predictors **together** affects the response, but not whether an **individual** predictor does, after controlling for other predictors.

### 3.4.9.1 Effects of collinearity

In general, collinearity increases Type II error: the probability of concluding a predictor has no effect on the response, when it actually does. Collinearity does not in general affect the actual values of coefficient estimates, just their standard errors.<sup>5</sup>

Importantly, **collinearity is a property of the data**—not a violation of model assumptions. To the extent that an effect is “missed” (a Type II error) in the presence of collinearity, this is because of the structure of the data. In the example above, it makes intuitive sense that it is harder to detect a “real” effect of **Familiarity** when **frequency** is added as a predictor, because of the correlation between **Familiarity** and **frequency**.

Because collinearity isn’t a violation of model assumptions, high collinearity cannot be diagnosed using diagnostic plots, such as those considered for Assumptions 1-4 above.

### 3.4.9.2 Diagnosing collinearity

How can one then detect collinearity, and decide if it could be affecting (the standard errors of) regression coefficients?

Some warning signs that there may be substantial collinearity in your data (Chatterjee and Hadi (2012), 9.4):

- **Unstable coefficients:** large changes in values of the  $\hat{\beta}_i$  when predictors/data points are added/dropped.
- **Nonsensical coefficients:** signs of  $\hat{\beta}_i$  don’t conform to prior expectations.
- **Unexpected non-significance:** values of  $\hat{\beta}_i$  for predictors expected to be important (e.g. from EDA) have large SEs, low  $t$ -values, and high  $p$ -values.

These diagnostics all follow from the fact that when data is highly collinear, a linear relationship between predictors **almost** holds, so there are many regression coefficient estimates that give models **almost** as good as the least-squared estimates.

The degree of collinearity can be quantified using the *condition number* (`collin.fnc` in `languageR`), applied to the design matrix (where each column = values of one predictor, across the dataset). For example, for **frequency** and **AoA** in the example above, the condition number is:

```
library(languageR)
collin.fnc(dplyr::select(d, Familiarity, WrittenFrequency, LengthInLetters))$cnumber
## [1] 16.0162
```

As a rule of thumb, condition numbers can be interpreted as (Baayen (2008), p. 200, citing Belsley et al. (1980)):

- CN < 6: “no collinearity”
- CN < 15: “acceptable collinearity”
- CN > 30: “potential harmful collinearity”

### 3.4.9.3 Is collinearity a problem?

There are two philosophies here.

---

<sup>5</sup>However, when standard errors are high, the coefficient estimates themselves are by definition very uncertain, so in this sense collinearity “affects” coefficient values.

The first is represented by Chatterjee and Hadi (2012) and Baayen (2008): (high) collinearity **is** a problem—it causes unstable coefficient estimates, increases Type II error, and can slow down or foil model fitting. Therefore, it should be somehow dealt with, for example by changing the predictors included in the model (“residualizing” or dimensionality reduction strategies such as “principal components”).

The second is represented by Gelman and Hill (2007) and Levy (2012): collinearity **is not** a problem—the issues above reflect a lack of information in your data, and how hard it is to detect effects of these variables based on this dataset. Therefore, you should either fit the model with your original predictors (which acknowledges the lack of information), or collect more data.

While we think there is real merit to the second view, there are certainly (very common) circumstances in which it is appropriate to take steps to decrease collinearity. Most common: centering all predictor variables decreases collinearity, without any loss of information. “Residualizing” one predictor on another is generally not a good idea, because it complicates the interpretation of the regression coefficients in unintended ways (Wurm and Fisicaro, 2014).

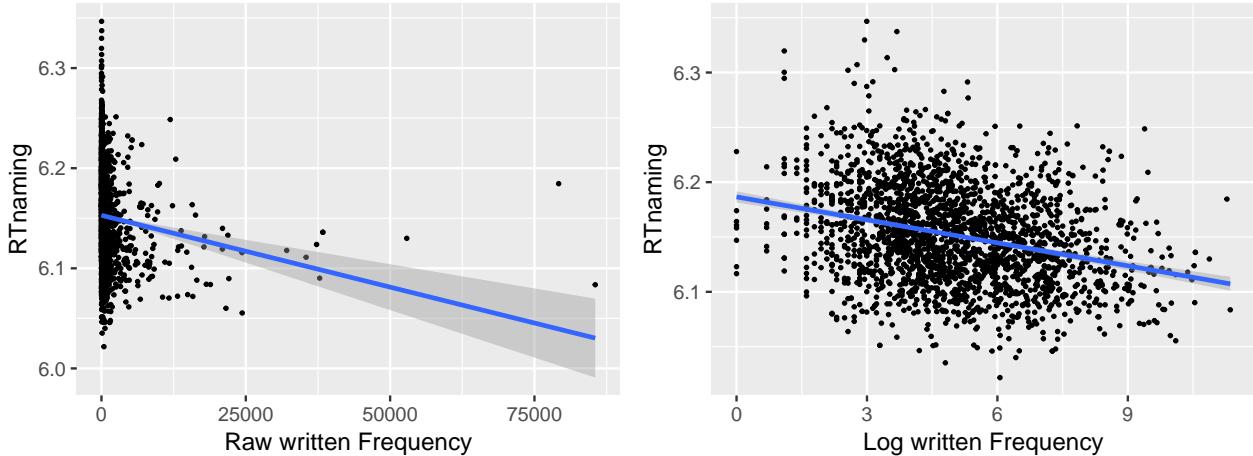
### 3.4.10 Assumption 6: Observations

Two linear regression assumptions about **observations** are that they are:

1. Equally **reliable**
2. Roughly equally **influential**

The first assumption is hard to check in practice, and we won’t consider it further. The second assumption is important: observations should have “a roughly equal role in determining regression results and in influencing conclusions” (Chatterjee and Hadi (2012), p. 88). This is because our goal in statistical analysis is usually to make inferences about **population** values of parameters (like the slope of a line of best fit), abstracting away from our finite sample. If certain observations are much more influential than others, they skew the regression results to reflect not the population, but the particular sample we happened to draw.

For example, consider the relationship between raw and log-transformed word frequency, and naming latency, in the `english` dataset. (The same data as in this example above, but now fitting a simple line of best fit in each case.)



When raw frequency is used as the predictor (left plot), the handful of “extreme” observations with frequency above 25000 have a much greater effect on the slope of the line than other observations, resulting in larger confidence intervals.

Often, a non-normally distributed response or predictors leads to some points influencing the model much more than others, and the problem can be fixed by transforming to normality—as for the word frequency

example (right plot). After log-transforming frequency, removing the seven most extreme X values hardly affects the fitted line.

The presence of highly-influential observations can lead to either Type I or Type II errors: the influential observations might be responsible for a spurious result, or they might obscure a pattern that would be clear if the influential observations were excluded.

This isn't always the case, though—often, some points are inherently more influential than others (say, a couple participants with behavior very different from others), and we need to decide how to proceed in fitting and interpreting the model.

### 3.4.11 Measuring influence

*Cook's distance* is a metric of how influential an observation is in a given model, defined as the product of two terms:

1. The (squared) standardized residual of the observation
2. A measure of the *leverage* of this observation—how much it affects the fitted values ( $\hat{y}_i$ )

Cook's distance (CD) is roughly **how much the regression coefficients change** when the model is fitted with all data except this observation. Data points with significantly higher CD than other points are highly influential, and should be flagged and examined as potential outliers.

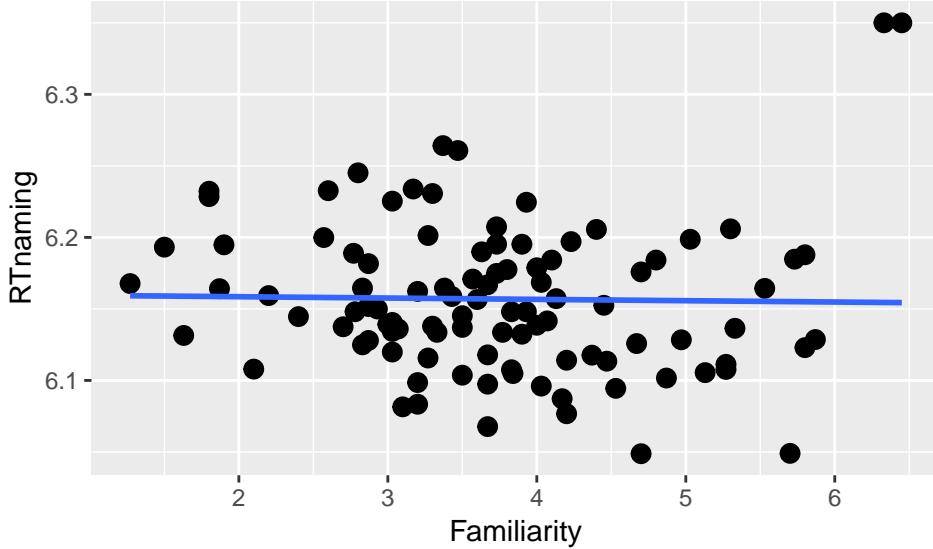
#### Example

Let's take a sample of the young-speaker subset of `english` and tweak it, to get an intuition of what it means for observations to have high influence.

```
set.seed(2903)
young_subset <- filter(english, AgeSubject=="young") %>%
  sample_n(100)

# Tweak data to illustrate point
young_subset[young_subset$Familiarity>6,]$RTnaming <- 6.35

ggplot(young_subset, aes(Familiarity, RTnaming)) +
  geom_point(size=3) +
  geom_smooth(method="lm", se=F)
```

**Questions:**

- Which points do you think have high influence on the model relating Familiarity and RTnaming?

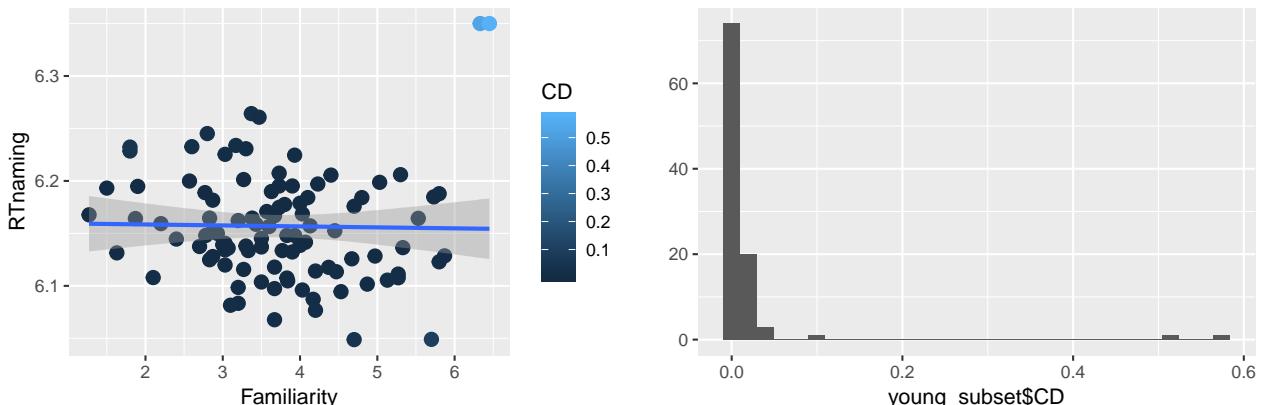
Let's check our intuition by coloring the points according to their CD values:

```
m <- lm(RTnaming ~ Familiarity, young_subset)
young_subset$CD <- cooks.distance(m)

p3 <- ggplot(young_subset, aes(Familiarity, RTnaming)) +
  geom_point(size=3, aes(color=CD)) +
  geom_smooth(method="lm")

p4 <- qplot(young_subset$CD)

grid.arrange(p3, p4, ncol = 2)
```



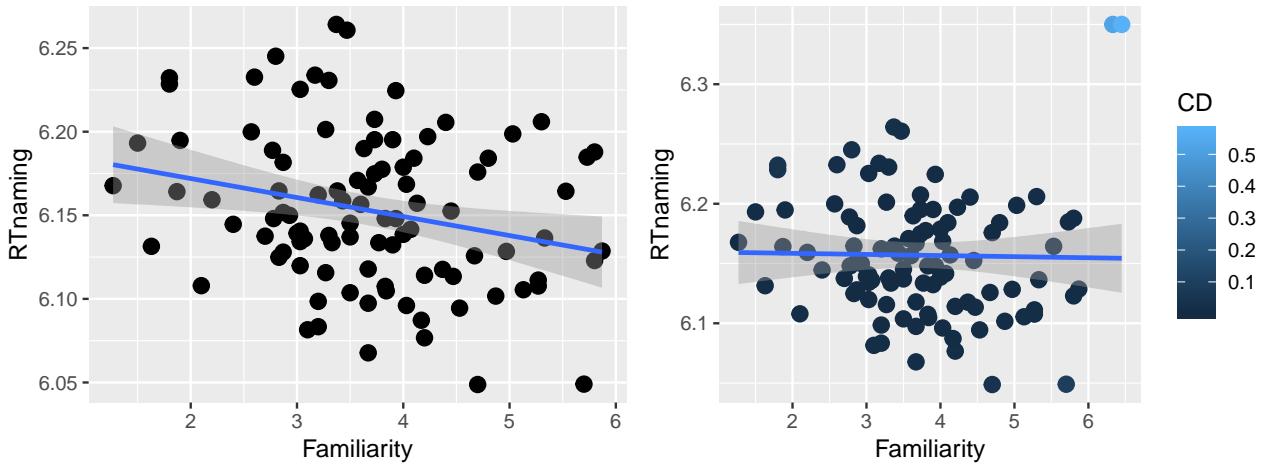
Indeed, two points are outliers in terms of Cook's distance. We expect to get a very different regression line by removing these two points (why?):

```
young_subset2 <- filter(young_subset, CD < 0.4)
```

```
p5 <- ggplot(young_subset2, aes(Familiarity, RTnaming)) +
```

```
geom_point(size=3) +
geom_smooth(method="lm")

grid.arrange(p5, p3, ncol = 2)
```



After removal of these two outliers, the regression line has negative slope instead of flat slope.

### 3.4.12 Outliers

Highly influential points are one example of *outliers*. It is also possible for points to have extreme values of the predictor or response, and yet not be highly influential. There are different philosophies on how to detect outliers (visual inspection versus numeric measures), and what to do with them. At a minimum:

- During data analysis, potential outliers should be flagged, examined, and considered for exclusion.
- Note what effect excluding outliers **would** have on the analysis.
- Report how outliers were dealt with, and ideally what effect they have on the analysis, in any write-up.

In many cases inspection of gross outliers will reveal data points that should be clearly excluded, such as a participant who always gave the same response, or data coding errors that cannot be corrected. In other cases the decision is more subjective.

### 3.4.13 Regression assumptions: Reassurance

It may seem a bit daunting that there are so many ways for a regression model to **not** be appropriate. It is important to take a step back, and remember: when doing data analysis, you should just satisfy the assumptions of the statistical tool being used as best you can, and then **fit and report your model anyway**. You should not fit regression models blindly, and it's important to be aware of the assumptions being made by these models, and their limitations. But the more you learn about regression (or any statistical technique), the easier it is to become paralyzed with fear by everything that could be wrong with your data. Keep in mind the dictum of George Box: "Essentially, all models are wrong, but some are useful." That is, every model is just that: a **model** of reality, that is only as useful as the insight it can offer about the questions of interest. Often, minor violations of model assumptions don't affect the qualitative conclusions you can draw from the model.

## 3.5 Model comparison

So far in our discussion of regression, we have assumed that the form of the model is given: we already know what predictors will be used, and what terms will be included in the model. In practice, with real data this is usually not clear—it is necessary to choose among several possible models of the same data, that differ in which terms they include. Choosing between models is called *variable selection* (or *model selection*). In order to perform variable selection, we need a way to perform *model comparison*: comparing two or more candidate models to assess which one is “better”, in the sense of how well it fits the data relative to its expressive power (e.g. number of predictors).

We wish to compare models of the same dataset, with:

- Same response ( $Y$ )
- Different sets of predictors ( $X_i$ )

Model comparison techniques differ on whether they can compare “nested” models only, or both nested and “non-nested” models.

### 3.5.1 Nested model comparison

Two models are *nested* if one is a subset of the other, in terms of the set of predictors included.

For example, these two models of RT (in the `english` dataset) are nested:

$$\begin{aligned} M_1 &: Y = \beta_0 + \beta_1 \cdot \text{WrittenFrequency} + \epsilon \\ M_2 &: Y = \beta_0 + \beta_1 \cdot \text{WrittenFrequency} + \beta_2 \cdot \text{Familiarity} + \epsilon \end{aligned}$$

$M_2$  is called the *full* or *superset* model, and  $M_1$  the *reduced* or *subset* model.

To compare these two models, we wish to perform a hypothesis test of:

$$H_0 : \beta_2 = 0$$

More generally, we wish to compare two nested models:

- $M_1$ : predictors  $X_1, \dots, X_q$
- $M_2$ : predictors  $X_1, \dots, X_p$  (where  $q < p$ )

By performing this hypothesis test:

$$H_0 : \beta_{p+1} = \beta_{p+2} = \dots = \beta_q = 0$$

We fit both models, and obtain a sum of squared residuals for each:  $RSS_1$ ,  $RSS_2$ , which serves as a measure of how well the model fits the data. The RSS of a regression model can be scaled by  $n - p - 1$ , where  $p$  is the number of predictors in the model, to give a measure of how well the model fits the data **given the sample size and number of predictors**:

$$RSS/(n - p - 1)$$

In addition, the difference in RSS values between two nested models, with  $q$  and  $p$  predictors ( $p > q$ ), can be scaled to give a measure of how much RSS has gone down, **relative to what's expected** given the additional predictors:

$$(RSS_1 - RSS_2)/(p - q)$$

Thus, this test statistic:

$$F = \frac{(RSS_1 - RSS_2)/(p - q)}{RSS_2/(n - p - 1)}$$

gives a measure of how much the unexplained variance is reduced in the full model, with respect to the reduced model. Intuitively,  $R$  divides how much effect “dropping” the  $q$  extra predictors has on explained variance. Note that  $RSS_2$  will always be smaller than  $RSS_1$ , since adding predictors to a model can’t give a **worse** fit to the data. What our hypothesis test checks is: does the superset model fit the data **significantly** better than the subset model, given the added complexity?

It turns out that this test statistic follows an  $F$  distribution (under the null hypothesis above) with  $p - q$ ,  $n - p$  degrees of freedom, written  $F_{p-q, n-p}$ . So, we can perform hypothesis testing using an  $F$ -test, which may be familiar if you have seen ANOVAs.

### Example

This model comparison addresses the question: does adding `AgeSubject` and `LengthInLetters` to `m1` significantly reduce its unexplained variance?

```
m1 <- lm(RTlexdec ~ WrittenFrequency, english)
m2 <- lm(RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters, english)
```

```
anova(m1, m2)
```

```
## Analysis of Variance Table
##
## Model 1: RTlexdec ~ WrittenFrequency
## Model 2: RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1    4566 91.194
## 2    4564 35.004  2      56.19 3663.1 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The highly significant  $F$ -test means that adding these two variables does significantly improve the model.

### Practical notes:

- In experimental literature (at least in linguistics), “model comparison” is often used as shorthand for “nested model comparison via an  $F$  test” when comparing two linear regressions, because this is the most common way to do so. We will sometimes use this shorthand as well.
- The results of such a hypothesis test are usually reported in parentheses, like “Subject age and the word’s length in letters together affect RT beyond word frequency ( $F(2, 4564) = 3663, p < 0.0001$ )”.

### Questions:

- What are the reduced model and full model in the  $F$  test reported at the bottom of every linear regression’s output?

```
summary(m2)
```

```
##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters,
##     data = english)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
## -0.34438 -0.06041 -0.00695  0.05241  0.45157
```

```

## 
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 6.8293072  0.0079946 854.245 <2e-16 ***
## WrittenFrequency -0.0368919  0.0007045 -52.366 <2e-16 ***
## AgeSubjectyoung -0.2217215  0.0025915 -85.556 <2e-16 ***
## LengthInLetters  0.0038897  0.0015428   2.521  0.0117 *  
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.08758 on 4564 degrees of freedom
## Multiple R-squared:  0.6887, Adjusted R-squared:  0.6885 
## F-statistic: 3366 on 3 and 4564 DF, p-value: < 2.2e-16

```

Hint: fit `m0 <- lm(RTlexdec ~ 1, english)` and compare to `m2`.

### 3.5.2 Non-nested model comparison

In non-nested model comparison, one model isn't a subset of the other. For example, the two models `RT ~ Frequency` and `RT ~ Familiarity` would be non-nested.

Non-nested models can no longer be compared using sums-of-squares (using an  $F$  statistic). Instead, a very different approach is used: *information criteria*. Instead of testing the hypothesis that certain model coefficients are zero, information criteria compare models based on two general goals:

1. Fit the data as well as possible
2. Have as few predictors as possible (“parsimony”)

Different information criteria measures combine model likelihood ( $L$ ) and number of predictors ( $p$ ; the fewer the better) into a single value, of which the most common are:

- Akaike information criterion (AIC):

$$AIC = 2p - 2 \log(L)$$

- Bayesian information criterion (BIC)

$$BIC = p \log(n) - 2 \log(L)$$

To apply either criterion, you calculate its value for each model in a set of candidate models, and pick the model with the lowest value.

Note that information criteria can be used to compare models regardless of whether they are nested or non-nested.

#### Example

```

m1 <- lm(RTlexdec ~ WrittenFrequency, english)
m2 <- lm(RTlexdec ~ WrittenFrequency + AgeSubject, english)
m3 <- lm(RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters, english)
m4 <- lm(RTlexdec ~ WrittenFrequency + LengthInLetters, english)

```

Using AIC and BIC to compare the models:

```
AIC(m1, m2, m3, m4)
```

```
##      df      AIC
## m1  3 -4908.994
## m2  4 -9274.590
## m3  5 -9278.948
## m4  4 -4909.437
```

```
BIC(m1, m2, m3, m4)
```

```
##      df      BIC
## m1  3 -4889.713
## m2  4 -9248.883
## m3  5 -9246.814
## m4  4 -4883.729
```

By AIC, we would choose `m3`, while by BIC we would choose `m2`. This illustrates a couple of points:

- Different model selection criteria don't necessarily give the same answer.
- BIC tends to choose more parsimonious models than AIC.

### 3.5.3 Variable selection

Now that we have seen some methods to compare models with different sets of predictors, we can turn to variable selection: how to decide what predictors to keep in a given model.

There is no single best way to do variable selection, out of context—each method has pros and cons, and what method to use depends on the goals of your study. Your goal could be:

1. **Prediction** (estimating  $Y$  for unseen data as accurately as possible)
2. **Explanation** (choosing the right model from one of several possible pre-specified choices)
3. **Exploratory** study

In #1 and #2, our higher-level goal is generalization about “the world” (unseen data). In #3, we are interested primarily in the data we have, and don't claim that our results generalize to the world. We'll assume going forward that our goal is #1 or #2—many studies in language sciences are in fact exploratory, but at present this isn't a goal that leads to publication.

It is important to know that different variable selection methods result in different final models, appropriate for different goals, because most papers that report variable selection simply say what they did, without justification. It is up to the reader to think, what might the final model be if a different variable selection method were used?

There are three components of any variable selection method:

1. How models are compared (e.g.  $F$  test on nested models, AIC)
2. How the quality of a single model is evaluated
3. How it's decided which of a set of models to choose, based on #1 and #2.

We'll consider a couple methods here, and cover others in later chapters.

#### 3.5.3.1 Method 1: Nested model comparison

As noted above, “model comparison” is often used as shorthand for “comparison of nested models using F tests”, when referring to linear regressions.

### Exercise

Which model of `m1`, `m2`, `m3` (from the example above) would you choose based on nested model comparison ( $F$  test)?

Note: `anova(x,y,z)` can be used as shorthand when `x`, `y`, `z` are nested models, and so on (`anova(w,x,y,z)...`).

#### 3.5.3.2 Method 2: Stepwise variable selection

A common type of variable selection uses “choose automatically” for Component 3 above (how do we decide which model to prefer). In *stepwise* variable selection, we decide whether to add or drop terms based on any model comparison procedure (Component 1): AIC, BIC,  $F$  test with  $p < 0.05$ , etc.

There are various flavors of stepwise model selection, such as *forwards*, where you start with an empty model (intercept only) and add terms, and *backwards*, where you start with a full model and drop terms.

#### Example: Stepwise backwards selection using AIC

This is what the `step` function in R does by default.

1. Start with the complete regression model (all possible predictors) and obtain AIC
2. At each “step” we try all possible ways of removing one of the predictors, and whichever yields the lowest AIC value is kept.
3. Repeat steps 1-2 until we end up with a model with lower AIC value than any of the other possible models that you could produce by deleting one of its predictors.

Stepwise model selection is very popular, probably because it is easy to do using statistical software. It can be helpful to select among a huge set of predictors as a first pass, but has serious drawbacks.

#### Exercise: 25 potential predictors

Fit the same model with 25 potential predictors to two random halves of the `english` data, then perform stepwise backwards selection:

```
## split the English data in half, randomly:
set.seed(2903)
english.1 <- english %>% sample_n(round(nrow(english)/2))
english.2 <- english[!(row.names(english) %in% row.names(english.1)),]

## fit full model to each half-dataset
m1 <- lm(RTlexdec ~ (WrittenFrequency + Familiarity + AgeSubject + LengthInLetters + FamilySize)^3, eng
m2 <- lm(RTlexdec ~ (WrittenFrequency + Familiarity + AgeSubject + LengthInLetters + FamilySize)^3, eng

## trace=0 suppresses output
m1.stepped <- step(m1, trace=0)
m2.stepped <- step(m2, trace=0)

## the two resulting models
summary(m1.stepped)

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + Familiarity + AgeSubject +
##     LengthInLetters + FamilySize + WrittenFrequency:Familiarity +
```

```

##   WrittenFrequency:AgeSubject + WrittenFrequency:LengthInLetters +
##   WrittenFrequency:FamilySize + Familiarity:AgeSubject + Familiarity:LengthInLetters +
##   Familiarity:FamilySize + AgeSubject:LengthInLetters + LengthInLetters:FamilySize +
##   WrittenFrequency:Familiarity:AgeSubject + WrittenFrequency:AgeSubject:LengthInLetters,
##   data = english.1)
##
## Residuals:
##       Min      1Q Median      3Q     Max
## -0.42008 -0.05175 -0.00326  0.04572  0.37444
##
## Coefficients:
##                               Estimate Std. Error
## (Intercept)                7.189260  0.045538
## WrittenFrequency          -0.084197  0.011610
## Familiarity                 -0.058426  0.014137
## AgeSubjectyoung            -0.413999  0.058362
## LengthInLetters             -0.004709  0.008692
## FamilySize                  -0.112072  0.018225
## WrittenFrequency:Familiarity        0.008855  0.001203
## WrittenFrequency:AgeSubjectyoung      0.045885  0.011738
## WrittenFrequency:LengthInLetters      0.004581  0.002131
## WrittenFrequency:FamilySize          0.005662  0.001863
## Familiarity:AgeSubjectyoung         0.030455  0.008445
## Familiarity:LengthInLetters         -0.006493  0.002830
## Familiarity:FamilySize             0.007705  0.003262
## AgeSubjectyoung:LengthInLetters     0.013042  0.011006
## LengthInLetters:FamilySize          0.006179  0.003240
## WrittenFrequency:Familiarity:AgeSubjectyoung -0.006894  0.001396
## WrittenFrequency:AgeSubjectyoung:LengthInLetters -0.003181  0.002059
##
## t value Pr(>|t|)
## (Intercept) 157.875 < 2e-16 ***
## WrittenFrequency -7.252 5.61e-13 ***
## Familiarity -4.133 3.71e-05 ***
## AgeSubjectyoung -7.094 1.74e-12 ***
## LengthInLetters -0.542 0.588063
## FamilySize -6.149 9.17e-10 ***
## WrittenFrequency:Familiarity 7.360 2.56e-13 ***
## WrittenFrequency:AgeSubjectyoung 3.909 9.53e-05 ***
## WrittenFrequency:LengthInLetters 2.150 0.031699 *
## WrittenFrequency:FamilySize 3.039 0.002405 **
## Familiarity:AgeSubjectyoung 3.606 0.000317 ***
## Familiarity:LengthInLetters -2.295 0.021842 *
## Familiarity:FamilySize 2.362 0.018245 *
## AgeSubjectyoung:LengthInLetters 1.185 0.236150
## LengthInLetters:FamilySize 1.907 0.056609 .
## WrittenFrequency:Familiarity:AgeSubjectyoung -4.940 8.39e-07 ***
## WrittenFrequency:AgeSubjectyoung:LengthInLetters -1.545 0.122548
##
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.07727 on 2267 degrees of freedom
## Multiple R-squared:  0.755, Adjusted R-squared:  0.7533
## F-statistic: 436.6 on 16 and 2267 DF, p-value: < 2.2e-16

```

```

summary(m2.steped)

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + Familiarity + AgeSubject +
##     LengthInLetters + FamilySize + WrittenFrequency:Familiarity +
##     WrittenFrequency:LengthInLetters + WrittenFrequency:FamilySize +
##     Familiarity:AgeSubject + Familiarity:LengthInLetters + Familiarity:FamilySize +
##     LengthInLetters:FamilySize + WrittenFrequency:LengthInLetters:FamilySize,
##     data = english.2)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -0.30263 -0.05327 -0.00526  0.04654  0.50932
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                7.0214673  0.0616493 113.894
## WrittenFrequency          -0.0338445  0.0137689 -2.458
## Familiarity                 -0.0768169  0.0139772 -5.496
## AgeSubjectyoung            -0.2047469  0.0113464 -18.045
## LengthInLetters              0.0228065  0.0137176  1.663
## FamilySize                  -0.0362279  0.0365663 -0.991
## WrittenFrequency:Familiarity      0.0072404  0.0009183  7.884
## WrittenFrequency:LengthInLetters   -0.0024614  0.0029950 -0.822
## WrittenFrequency:FamilySize       -0.0096833  0.0055442 -1.747
## Familiarity:AgeSubjectyoung      -0.0042687  0.0028607 -1.492
## Familiarity:LengthInLetters      -0.0044449  0.0028657 -1.551
## Familiarity:FamilySize           0.0137811  0.0031320  4.400
## LengthInLetters:FamilySize       -0.0059273  0.0082370 -0.720
## WrittenFrequency:LengthInLetters:FamilySize 0.0020024  0.0012937  1.548
## Pr(>|t|)
## (Intercept) < 2e-16 ***
## WrittenFrequency 0.0140 *
## Familiarity 4.32e-08 ***
## AgeSubjectyoung < 2e-16 ***
## LengthInLetters 0.0965 .
## FamilySize 0.3219
## WrittenFrequency:Familiarity 4.87e-15 ***
## WrittenFrequency:LengthInLetters 0.4113
## WrittenFrequency:FamilySize 0.0808 .
## Familiarity:AgeSubjectyoung 0.1358
## Familiarity:LengthInLetters 0.1210
## Familiarity:FamilySize 1.13e-05 ***
## LengthInLetters:FamilySize 0.4718
## WrittenFrequency:LengthInLetters:FamilySize 0.1218
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08024 on 2270 degrees of freedom
## Multiple R-squared:  0.7444, Adjusted R-squared:  0.7429
## F-statistic: 508.4 on 13 and 2270 DF,  p-value: < 2.2e-16

```

Compare the two resulting models:

- How similar or different are the resulting terms in the model, and coefficient values?
  - Why is the degree of (non-)similarity a problem?
- 

In general, fully automatic model selection procedures are **dangerous**: they can easily find spurious effects, because automatically dropping terms with high  $p$ -values leads to inflation (= lower values) of significances for remaining terms. Stepwise procedures can also (to a lesser extent) miss true effects. Some statisticians recommend that automatic model selection should never be used at all, as it is simply too easy and too dangerous. At a minimum, automatic model selection procedures should never be used **alone** for variable selection—that is, in the absence of another method, such as careful exploratory data analysis. Despite this, stepwise procedures are widely used in language research, and you should critically examine any model arrived at in this way.<sup>6</sup>

### 3.5.3.3 Method 3: Gelman and Hill (2007)

Gelman and Hill (2007) (p. 69) suggest a holistic approach to variable selection, that takes common sense into account in evaluating model quality (Component 2 above), in addition to quantitative model comparison (Component 1):

1. Include all predictors that, **for substantive reasons**, are expected to be important (e.g. speech rate when modeling vowel duration).
2. For predictors with large effects, consider interactions as well.
3. Now decide whether to exclude predictors, one at a time:
  - Not significant, coefficient has expected sign: Consider leaving in.
  - Not significant, wrong sign: Remove.
  - Significant, wrong sign: Think hard if something's wrong.
  - Significant, right sign: Keep in.

This approach requires you to have a sense of what the “right” and “wrong” sign of different coefficients are, either for substantive reasons (previous work suggests an effect direction), or from exploratory data analysis. If your sense of “right” and “wrong” for a given model coefficient is based **purely** on EDA, it’s important to remember that you may be simply modeling this dataset well (goal = exploratory analysis) rather than obtaining results that generalize to unseen data (goal = prediction or explanation).

#### Exercise: Phrase medial devoicing in European French

This dataset is described in the appendix. We are interested in whether function words are shorter than content words—considering just words which are a consonant-vowel sequence—after controlling for the consonant and vowel identities (since different Cs/Vs are intrinsically longer than others) and how fast the speaker is speaking.

In terms of the variables in this dataset:

- Is there is an effect of `func` on `syldur...`
- ... after controlling for `c1`, `v`, and `speebrate`?

#### Part 1: Model comparison

Use (nested) model comparison to compare these three models:

---

<sup>6</sup>For example: are there terms that should have been included, based on previous work—regardless of the result of a model comparison? Are there higher-order interactions with borderline significance, that could be spuriously included due to automated selection?

1. `syldur~(speechrate+c1+v)^2`
2. `syldur~(speechrate+c1+v)^2+func`
3. `syldur~(speechrate+c1+v)^2*func`

**Questions:**

- Which model ends up being chosen, and what do you conclude about the research question?

```
#model comparison
m1 <- lm(syldur~(speechrate+c1+v)^2, df)
m2 <- lm(syldur~(speechrate+c1+v)^2+func, df)
m3 <- lm(syldur~(speechrate+c1+v)^2*func, df)
anova(m1, m2, m3)

## Analysis of Variance Table
##
## Model 1: syldur ~ (speechrate + c1 + v)^2
## Model 2: syldur ~ (speechrate + c1 + v)^2 + func
## Model 3: syldur ~ (speechrate + c1 + v)^2 * func
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1     529 528092
## 2     528 518990  1     9102.4 9.3121 0.002392 **
## 3     523 511219  5     7770.7 1.5900 0.161123
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
summary(m2)

##
## Call:
## lm(formula = syldur ~ (speechrate + c1 + v)^2 + func, data = df)
##
## Residuals:
##       Min        1Q        Median        3Q        Max
## -130.682   -21.233    0.304    20.544   103.960
##
## Coefficients: (1 not defined because of singularities)
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 247.5718   23.8023 10.401 < 2e-16 ***
## speechrate -15.6166   3.2983 -4.735 2.82e-06 ***
## c1k         -89.2652  26.6617 -3.348 0.000872 ***
## c1p         -67.4682  39.4741 -1.709 0.088006 .
## cis          -27.7955  26.1708 -1.062 0.288683
## c1t         -68.6392  27.3605 -2.509 0.012416 *
## vu            4.6582  31.5829  0.147 0.882802
## vy           16.2882  20.2343  0.805 0.421194
## funcf      -13.8005  4.5350 -3.043 0.002458 **
## speechrate:c1k  9.5259  3.7368  2.549 0.011077 *
## speechrate:c1p  9.2280  5.8037  1.590 0.112428
## speechrate:cis  4.4121  3.5902  1.229 0.219637
## speechrate:c1t  7.5638  3.7154  2.036 0.042268 *
## speechrate:vu -0.6849  3.9631 -0.173 0.862851
## speechrate:vy -1.6412  2.6474 -0.620 0.535579
## c1k:vu        19.8107 22.1307  0.895 0.371104
## c1p:vu        -1.3369 25.6737 -0.052 0.958489
```

```

## c1s:vu      0.9348   22.3409   0.042 0.966639
## c1t:vu      NA       NA       NA       NA
## c1k:vy     -3.1728   13.8966  -0.228 0.819491
## c1p:vy     -9.7528   15.2713  -0.639 0.523338
## c1s:vy    -18.2987   11.6317  -1.573 0.116279
## c1t:vy     -5.4756   11.9582  -0.458 0.647219
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.35 on 528 degrees of freedom
## Multiple R-squared:  0.3012, Adjusted R-squared:  0.2734
## F-statistic: 10.84 on 21 and 528 DF,  p-value: < 2.2e-16

```

## Part 2: Stepwise backwards selection

Apply stepwise backwards model selection to this model:

- `syldur~(speechrate+c1+v+func)^2`

using `step`.

```

#Backwards selection
m = lm(syldur~(speechrate+c1+v+func)^2, df)
summary(step(m, trace=0))

##
## Call:
## lm(formula = syldur ~ speechrate + c1 + func + speechrate:c1 +
##     c1:func, data = df)
##
## Residuals:
##      Min        1Q        Median        3Q        Max 
## -129.449   -19.780    0.762    21.343   104.048 
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 255.813   20.798 12.300 < 2e-16 ***
## speechrate -16.508    2.991 -5.520 5.3e-08 ***
## c1k         -87.803   25.813 -3.401 0.000720 ***
## c1p         -71.475   37.858 -1.888 0.059566 .  
## c1s         -41.730   23.963 -1.741 0.082180 .  
## c1t         -64.292   24.776 -2.595 0.009719 ** 
## funcf       -21.398    5.558 -3.850 0.000132 *** 
## speechrate:c1k 10.350   3.559  2.908 0.003785 ** 
## speechrate:c1p  9.105   5.564  1.637 0.102310  
## speechrate:c1s  5.058   3.422  1.478 0.139894  
## speechrate:c1t  7.322   3.498  2.093 0.036832 *  
## c1k:funcf    -1.833   8.565 -0.214 0.830655  
## c1p:funcf      NA      NA      NA      NA      
## c1s:funcf     17.373   7.218  2.407 0.016427 *  
## c1t:funcf      NA      NA      NA      NA      
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.14 on 537 degrees of freedom
## Multiple R-squared:  0.2987, Adjusted R-squared:  0.283 
## F-statistic: 19.06 on 12 and 537 DF,  p-value: < 2.2e-16

```

**Questions:**

- Which model ends up being chosen, and what do you conclude about the research question?
- What would be different about your answer to the research question using stepwise backwards selection versus nested model comparison? Why?

**Part 3:** Gelman & Hill's method

This method requires more domain specific knowledge, so we'll walk through it together.

First, all four possible predictors are included as main effects—they are all expected to affect vowel duration for substantive reasons:

```
# G&H
m1 <- lm(syldur~speechrate+c1+v+func, df)
summary(m1)

##
## Call:
## lm(formula = syldur ~ speechrate + c1 + v + func, data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -131.151  -21.337    0.863   20.419   98.228
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 208.8600    7.9734  26.195 < 2e-16 ***
## speechrate   -9.5866    0.9546 -10.043 < 2e-16 ***
## c1k          -26.6525    5.8330  -4.569 6.07e-06 ***
## c1p          -11.0561    6.9893  -1.582  0.11426
## cis           -2.9871    5.3464  -0.559  0.57659
## c1t          -18.3920    5.8075  -3.167  0.00163 **
## vu            12.3446    6.1420   2.010  0.04494 *
## vy            -2.0698    3.4933  -0.592  0.55377
## funcf        -13.3345    3.2684  -4.080 5.19e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.47 on 541 degrees of freedom
## Multiple R-squared:  0.2787, Adjusted R-squared:  0.2681
## F-statistic: 26.13 on 8 and 541 DF, p-value: < 2.2e-16
```

Every predictor has a significant effect. Let's suppose that `speechrate` in particular has a large effect (we'll discuss soon how to assess effect size). We thus consider interactions with speech rate:

```
m2 <- lm(syldur~speechrate*(func+c1+v), df)
summary(m2)

##
## Call:
## lm(formula = syldur ~ speechrate * (func + c1 + v), data = df)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -130.275  -20.390    0.522   20.501  103.274
##
```

```

## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)            255.785   22.952 11.144 < 2e-16 ***
## speechrate           -16.386    3.274 -5.005 7.58e-07 ***
## funcf                 -29.289   17.059 -1.717 0.08658 .
## c1k                  -86.729   28.237 -3.071 0.00224 **
## c1p                  -66.594   38.304 -1.739 0.08269 .
## c1s                  -29.059   26.392 -1.101 0.27137
## c1t                  -60.538   27.829 -2.175 0.03004 *
## vu                   -6.158   27.997 -0.220 0.82599
## vy                   3.596   18.342  0.196 0.84463
## speechrate:funcf     2.364    2.384  0.992 0.32172
## speechrate:c1k       8.949    4.009  2.232 0.02604 *
## speechrate:c1p       8.235    5.645  1.459 0.14522
## speechrate:c1s       3.813    3.775  1.010 0.31291
## speechrate:c1t       6.203    3.965  1.564 0.11829
## speechrate:vu        2.217    3.953  0.561 0.57512
## speechrate:vy       -1.013    2.585 -0.392 0.69529
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.32 on 534 degrees of freedom
## Multiple R-squared:  0.2945, Adjusted R-squared:  0.2747
## F-statistic: 14.86 on 15 and 534 DF,  p-value: < 2.2e-16

```

There seems to be a significant interaction with preceding consonant, in particular preceding /k/ means less duration decrease with increasing rate. There isn't an immediately obvious explanation for this interaction, so it's not "wrong sign", and it's fine to leave in the model.

`func` also has a large effect, and is of primary interest, so we consider its potential interactions:

```
m3 <- lm(syldur~func*(speechrate+c1+v), df)
summary(m3)
```

```

##
## Call:
## lm(formula = syldur ~ func * (speechrate + c1 + v), data = df)
##
## Residuals:
##      Min      1Q      Median      3Q      Max
## -132.663 -21.550     1.495    18.620   101.766
##
## Coefficients: (4 not defined because of singularities)
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)            223.1977   11.2756 19.795 < 2e-16 ***
## funcf                 -46.3965   14.9707 -3.099 0.00204 **
## speechrate            -11.8593   1.4817 -8.004 7.47e-15 ***
## c1k                  -18.3159   7.5125 -2.438 0.01509 *
## c1p                 -11.6366   6.9622 -1.671 0.09522 .
## c1s                  -7.8271   5.5722 -1.405 0.16070
## c1t                 -13.3023   6.4106 -2.075 0.03846 *
## vu                   11.5966   6.8487  1.693 0.09098 .
## vy                   0.8120    4.3953  0.185 0.85350
## funcf:speechrate    3.4852    1.9453  1.792 0.07376 .
## funcf:c1k            0.3863   10.7540  0.036 0.97136
## funcf:c1p             NA        NA        NA        NA

```

```

## funcf:c1s      19.3740    8.7143   2.223  0.02661 *
## funcf:c1t       NA        NA        NA        NA
## funcf:vu        NA        NA        NA        NA
## funcf:vy        NA        NA        NA        NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.22 on 538 degrees of freedom
## Multiple R-squared:  0.2941, Adjusted R-squared:  0.2797
## F-statistic: 20.38 on 11 and 538 DF,  p-value: < 2.2e-16

```

There's a significant `func:c1` interaction. It's not clear what is the "right" or "wrong" sign here, but since `func` is of primary interest, it is prudent to include this term in the model, in case it's controlling for something that might otherwise spuriously give a significant `func` effect.

Our final model is then:

```
m4 <- lm(syldur~speechrate+c1+func+speechrate:c1+func:c1, df)
summary(m4)
```

```

##
## Call:
## lm(formula = syldur ~ speechrate + c1 + func + speechrate:c1 +
##     func:c1, data = df)
##
## Residuals:
##    Min      1Q  Median      3Q     Max
## -129.449 -19.780   0.762  21.343 104.048
##
## Coefficients: (2 not defined because of singularities)
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  255.813   20.798 12.300 < 2e-16 ***
## speechrate   -16.508    2.991 -5.520 5.3e-08 ***
## c1k          -87.803   25.813 -3.401 0.000720 ***
## c1p          -71.475   37.858 -1.888 0.059566 .
## c1s          -41.730   23.963 -1.741 0.082180 .
## c1t          -64.292   24.776 -2.595 0.009719 **
## funcf         -21.398    5.558 -3.850 0.000132 ***
## speechrate:c1k  10.350    3.559  2.908 0.003785 **
## speechrate:c1p   9.105    5.564  1.637 0.102310
## speechrate:c1s   5.058    3.422  1.478 0.139894
## speechrate:c1t   7.322    3.498  2.093 0.036832 *
## c1k:funcf      -1.833    8.565 -0.214 0.830655
## c1p:funcf        NA      NA      NA      NA
## c1s:funcf      17.373    7.218  2.407 0.016427 *
## c1t:funcf        NA      NA      NA      NA
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 31.14 on 537 degrees of freedom
## Multiple R-squared:  0.2987, Adjusted R-squared:  0.283
## F-statistic: 19.06 on 12 and 537 DF,  p-value: < 2.2e-16

```

### Questions:

- What do you conclude about the research question, based on the final model?

- What does the significant `func:c1` interaction mean for the research question? (Does the effect of `func` have the same direction regardless of `c1`?)

### 3.5.4 Interpretability issues

A crucial aspect of interpreting regression model results is being able to interpret and compare coefficient values.

Consider this model for the `english` data:

```
m6 <- lm(RTlexdec ~ WrittenFrequency + LengthInLetters + AgeSubject, english)
summary(m6)
```

```
##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + LengthInLetters +
##     AgeSubject, data = english)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34438 -0.06041 -0.00695  0.05241  0.45157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.8293072  0.0079946 854.245 <2e-16 ***
## WrittenFrequency -0.0368919  0.0007045 -52.366 <2e-16 ***
## LengthInLetters  0.0038897  0.0015428   2.521  0.0117 *
## AgeSubjectyoung -0.2217215  0.0025915 -85.556 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08758 on 4564 degrees of freedom
## Multiple R-squared:  0.6887, Adjusted R-squared:  0.6885
## F-statistic:  3366 on 3 and 4564 DF,  p-value: < 2.2e-16
```

Two issues make the coefficients difficult to interpret:

1. The **intercept value** is not interesting or meaningful
  - Predicted response when all  $X$  are 0.
  - What does `WrittenFrequency=0, LengthInLetters=0` mean?
2. **Coefficient values aren't comparable**, because predictors are on different scales.
  - Change of 1 in `AgeSubject` covers 100% of the data, while change of 1 in `WrittenFrequency` only covers part of the data.
  - We need comparable coefficient values to help assess *effect size*.

#### 3.5.4.1 Solutions

We can address these two issues by transforming predictors before fitting the model. There are two common ways to *standardize* predictors, considering for now just continuous predictors and binary factors (no multi-level factors):

1. *z-score*: subtract mean (center) and divide by standard deviation

- This measure is similar to Cohen's  $d$  for binary predictors (which we'll discuss soon)
  - `scale()` function in R
2. Alternative method (Gelman and Hill (2007) §4.2):
- Continuous predictors: center and divide by **2 SD**.
  - Binary predictors: transform to have mean 0 and difference of 1 between values (i.e.  $-0.5/0.5$  for balanced data)
  - Interpretation of “unit change of 1” is similar for continuous and binary predictors  $\Rightarrow$  can compare effect sizes.
  - Use `rescale()` in the `arm` package.

Method 2, while less standard, has advantages discussed by Gelman and Hill (2007), and we'll use it going forward.

New interpretations of coefficients:

- **Intercept**: predicted  $Y$  when all predictors are held at mean value
- **Main effect** coefficients for continuous predictors: predicted change in  $Y$  when predictor changed by 2 SD, with other predictors held at **mean** values.
- **Main effect** coefficients for binary predictors: difference between the two levels. (Which is the same as 2 SD.)

**Practical notes:**

- When you report standardization of variables in a paper, report the information needed to replicate what you did—not the same of particular software package or function used. For example, you'd write “continuous variables were standardized by centering and dividing by two standard deviations”, not “continuous variables were standardized using the `rescale()` function in the `arm` package”. The latter kind of report is common and unhelpful.
- “Rescaling”, “rescaled variables” etc. are **not** standard terminology, and should not be used in any write-up. Instead, refer to “standardized” variables, defining at some point what “standardized” means.

## Example

Model of `RTlexdec` as a function of `WrittenFrequency`, `LengthInLetters`, `AgeSubject`, using raw and standardized predictors:

Before standardizing:

```
summary(m6)

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + LengthInLetters +
##     AgeSubject, data = english)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34438 -0.06041 -0.00695  0.05241  0.45157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.8293072  0.0079946 854.245  <2e-16 ***
```

```

## WrittenFrequency -0.0368919  0.0007045 -52.366 <2e-16 ***
## LengthInLetters   0.0038897  0.0015428   2.521  0.0117 *
## AgeSubjectyoung  -0.2217215  0.0025915 -85.556 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08758 on 4564 degrees of freedom
## Multiple R-squared:  0.6887, Adjusted R-squared:  0.6885
## F-statistic:  3366 on 3 and 4564 DF,  p-value: < 2.2e-16

```

After standardizing:

```

english2 <- mutate(english,
  WrittenFrequency = arm::rescale(WrittenFrequency),
  LengthInLetters = arm::rescale(LengthInLetters),
  AgeSubject       = arm::rescale(AgeSubject))
m7 <- lm(RTlexdec ~ WrittenFrequency + LengthInLetters + AgeSubject, english2)
summary(m7)

```

```

##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + LengthInLetters +
##     AgeSubject, data = english2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34438 -0.06041 -0.00695  0.05241  0.45157
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 6.550097  0.001296 5055.020 <2e-16 ***
## WrittenFrequency -0.136025  0.002598 -52.366 <2e-16 ***
## LengthInLetters  0.006549  0.002598   2.521  0.0117 *
## AgeSubject       -0.221721  0.002592 -85.556 <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08758 on 4564 degrees of freedom
## Multiple R-squared:  0.6887, Adjusted R-squared:  0.6885
## F-statistic:  3366 on 3 and 4564 DF,  p-value: < 2.2e-16

```

Interpretations of coefficients in the model using standardized predictors:

- (Intercept) row: Mean RT at mean  $X_i$ , for both old and young participants
- WrittenFrequency, LengthInLetters rows: Predicted change between mean( $X_i$ ) and lower or upper bound of  $X_i$ 's range
- AgeSubject row: Difference in group means between young and old participants

#### Questions:

- How do these interpretations differ from the coefficient interpretations for the model using raw predictors (m6)?
- What is the relative importance of the three predictors for predicting reaction time?

### 3.5.5 Interim recipe: Building a multiple linear regression model

Caveat: there is no single recipe for building a regression model of a dataset to address a research question! Nonetheless, here is one possible recipe, which we have given as part of linear regression project directions in classes.<sup>7</sup>

#### 1. Preliminaries

- State the problem (what are the goals of this analysis?)
- Select the response(s) and relevant predictors
- Continuous variables: center and scale, possibly transform to normality
- Categorical variables: center / choose coding scheme

#### 2. Do EDA:

- What patterns are there that “should” come out in a statistical model?
- Identify potential outliers
- (if applicable) Possible interactions? By-subject, by-item plots.

#### 3. Choose models to fit, based on steps 1-2

#### 4. Fit candidate model(s)

#### 5. Model criticism

- Assess linearity assumption
- Examine distribution of (standardized) residuals
- Examine plots of residuals against fitted values, predictors
- Look for influential points
- Check for collinearity
- (Other steps possible—especially checking model *robustness*—but not covered in this book.)

#### 6. Revise: Based on step 5, possibly:

- Exclude some data
- Transform the predictor and/or response
- Then re-fit the model (Step 4)

#### 7. Iterate: repeat Steps 4-6 until model criticism is satisfactory

- (0–2 iterations)

## 3.6 Solutions

### 3.6.1 Multiple linear regression: Solutions

**Q:** What does it mean that this coefficient is positive?

**A:** The effect of `WrittenFrequency` is closer to zero (= less negative) for younger speakers.

---

<sup>7</sup>Note that this recipe explicitly includes exploratory data analysis, which may not be appropriate depending on your analytic philosophy or conventions in your subfield.

### Regression equation 1

$$\text{Predicted RTlexdec} = \underbrace{6.552}_{\hat{\beta}_0} + (\underbrace{-0.011}_{\hat{\beta}_1} \cdot 3) + (\underbrace{-0.365}_{\hat{\beta}_2} \cdot \underbrace{0}_{\text{AgeSubject} == 'old'}) + (\underbrace{-0.005}_{\hat{\beta}_3} \cdot \underbrace{0}_{\text{AgeSubject} == 'old'}) = 6.519$$

### Regression equation 2

$$\text{Predicted RTlexdec} = \underbrace{6.552}_{\hat{\beta}_0} + (\underbrace{-0.011}_{\hat{\beta}_1} \cdot 3) + (\underbrace{-0.365}_{\hat{\beta}_2} \cdot \underbrace{1}_{\text{AgeSubject} == 'young'}) + (\underbrace{-0.005}_{\hat{\beta}_3} \cdot \underbrace{1}_{\text{AgeSubject} == 'young'}) = 6.139$$

## 3.6.2 Linear regression assumptions: Solutions

**Q:** Is this the case?

**A:** It looks like `rintensity` and `rpitch` have a roughly linear relationship, while the relationships between `rpitch` and `rduration` as well as between `rduration` and `rpitch` are nonlinear.

---

**Q:** Why do the residuals have this distribution?

**A:** The distribution of  $Y$  is highly right-skewed: there is a lot of data with  $Y \approx 1$ , and a long tail of data with  $Y \in [2, 5]$  (examine `hist(data$rhymeRating)`). As a result, the fitted line is near  $Y = 1$  for most values of  $X$ , so the residuals also have a right-skewed distribution: points with  $Y \approx 1$  correspond to residuals near the mode, and other points correspond to residuals in the right tail. In short, the residual distribution is highly skewed because the distribution of the response ( $Y$ ) is highly skewed.

---

**Q:** Is the linearity assumption met in each case?

**A:** No (left), Yes (right). The linearity assumption isn't met when  $Y$  is untransformed word frequency — or at least it's unclear — whereas the assumption looks safe for log-transformed  $Y$ .

---

## 3.6.3 Model comparison: Solutions

**Q:** What are the reduced model and full model in the  $F$  test reported at the bottom of every linear regression's output?

**A:** The full model is the model you fitted in the linear regression. The reduced model is the model including only an intercept. For example, in this model:

```
m2 <- lm(RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters, english)
summary(m2)
```

```
##
## Call:
## lm(formula = RTlexdec ~ WrittenFrequency + AgeSubject + LengthInLetters,
##      data = english)
##
## Residuals:
##      Min        1Q    Median        3Q       Max
## -0.34438 -0.06041 -0.00695  0.05241  0.45157
##
```

```

## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)           6.8293072  0.0079946 854.245 <2e-16 ***
## WrittenFrequency   -0.0368919  0.0007045 -52.366 <2e-16 ***
## AgeSubjectyoung    -0.2217215  0.0025915 -85.556 <2e-16 ***
## LengthInLetters     0.0038897  0.0015428   2.521  0.0117 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.08758 on 4564 degrees of freedom
## Multiple R-squared:  0.6887, Adjusted R-squared:  0.6885
## F-statistic:  3366 on 3 and 4564 DF,  p-value: < 2.2e-16

```

The *F* test is for the comparison with the model `m0 <- lm(RTlexdec ~ 1, english)`.

---

**Q:** Which model of `m1`, `m2`, `m3` (from the example above) would you choose based on nested model comparison (*F* test)?

**A:** Model 3 > Model 2 > Model 1

---

**Q:** How do these interpretations differ from the coefficient interpretations for the model using raw predictors (`m6`)? What is the relative importance of the three predictors for predicting reaction time?

**A:**

- Original model: change in RT per change of 1 in word frequency
- New model: change in RT per change of 2 SD in word frequency

The interpretation of the `LengthInLetters` coefficient is:

- Original model: change in RT for each additional letter in the word
- New model: change in RT per change of 2 SD word length (in letters)

The interpretation of the `AgeSubject` coefficient is the same in the two models. Although `AgeSubject` is a two-level factor in one model and a centered numeric variable in the other model, the regression coefficient captures the difference between the two levels (old and young speakers) in each case.

The relative importance of the three predictors follows from the magnitudes of the coefficients in the model using standardized predictors: `AgeSubject > WrittenFrequency > LengthInLetters`.



# Chapter 4

## Categorical data analysis: Preliminaries

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(languageR)
library(ggplot2)
library(dplyr)
library(arm)
library(boot)

## loads alternativesMcGillLing620.csv from OSF project for Wagner (2016)
## "Information structure and production planning"
alternatives <- read.csv(url("https://osf.io/6qctp/download"), stringsAsFactors = FALSE)
```

**Note:** Answers to some questions/exercises not listed in text are in Solutions

### 4.1 Introduction

So far in this book, we have mostly considered data analysis where the outcome is a continuous variable, like reaction time or vowel duration. In linear regression, the outcome ( $Y$ ) is modeled as a function of one or more categorical and continuous predictors ( $X_i$ ).

We now turn to *categorical data analysis*, where the outcome being modeled is a categorical variable. We assume that you have had some exposure to some basic categorical data analysis topics:

- *Contingency tables* (e.g. `xtabs` in R)
  - Including the notion of **observed** and **expected** values based on a contingency table, and “observed/expected ratios” (O/E ratios).
- *Tests of independence* of categorical variables
  - $\chi^2$  tests
  - Fisher’s exact test

which we refresh briefly below. If you haven't seen these topics before, some places to read more are given below.

### 4.1.1 2x2 contingency tables

Contingency tables show the number of observations for each combination of values of categorical variables. A 2x2 contingency table in particular shows the number of observations for each combination of two categorical variables,  $X_1$  and  $X_2$ .

#### Example

For this subset of the `alternatives` data:

- `prominence` = Adjective or Noun
- `context` = Alternative or NoAlternative

A contingency table showing the number of observations for each combination of `prominence` and `context`:

```
alternatives.sub <- filter(alternatives, context %in% c('Alternative', 'NoAlternative') & prominence %in% c('Adjective', 'Noun'))
xtabs(~prominence + context, data=alternatives.sub)

##          context
## prominence Alternative NoAlternative
##   Adjective      120          55
##   Noun            85         155
```

#### Observed and expected values

In the contingency table above, it looks like `prominence` isn't independent of `context`: `Adjective` prominence is much more likely relative to `Noun` prominence in `Alternative` context. We would like to formally test this hypothesis of non-independence. We can do so using this null hypothesis:

- $H_0$ : `prominence` and `context` occur independently in this data.

Under  $H_0$ , we can work out:

1. The estimated  $P(\text{context} = \text{alternative})$
2. The estimated  $P(\text{prominence} = \text{adjective})$
3. The expected count in each cell, given the total number of observations  $n$ .

For example, for the contingency table above, we can calculate the probabilities #1 and #2:

```
tab <- xtabs(~prominence + context, data=alternatives.sub)
## total number of observations
n <- sum(tab)

## P(context = alternative)
pAlt <- sum(tab[, 'Alternative'])/n

## p(prominence = adjective)
pAdj <- sum(tab['Adjective',])/n
```

```
## print these probabilities
pAlt

## [1] 0.4939759
pAdj

## [1] 0.4216867
```

The estimated count in the `context=Alternative & prominence=Adjective` cell is then:

$$n \cdot P(\text{'context' = 'Alternative'}) \cdot P(\text{'prominence' = 'Adjective'}) = 415 \cdot 0.4939759 \cdot 0.4216867 \quad (4.1)$$

$$= 86.4457831 \quad (4.2)$$

### Exercise 1

Calculate the estimated counts for the other three cells.

#### 4.1.2 The chi-squared test

For a 2x2 contingency table, we have:

- The *observed* counts in each cell, denoted  $O_i$
- The *expected* counts in each cell, denoted  $E_i$ , calculated assuming independence ( $H_0$ ).

We can then define *Pearson's test statistic*:

$$X^2 = \sum_{i=1}^n \frac{(O_i - E_i)^2}{E_i}$$

which measures how much the observed and expected values differ, across the whole contingency table.  $X^2$  is 0 when the observed and expected values are exactly the same, and increases the more the observed and expected values differ.

Pearson's test statistic **approximately** follows a  $\chi^2$  distribution (pronounced “chi-squared”) with one degree of freedom, denoted  $\chi^2(1)$ , under conditions described below. Thus, we can test the null hypothesis that  $X_1$  and  $X_2$  are independent by comparing the value of  $X^2$  to the  $\chi^2(1)$  distribution.

The same methodology applies for testing independence of two categorical variables with any number of levels. For  $X_1$  and  $X_2$  with  $p$  and  $q$  levels,  $X^2$  can be calculated using Eq. (4.1.2), and the null hypothesis tested using a  $\chi^2((p-1)(q-1))$  distribution.

### Examples

1. Are `prominence` and `context` independent for the contingency table above?

```
chisq.test(tab)

##
## Pearson's Chi-squared test with Yates' continuity correction
##
## data: tab
## X-squared = 43.189, df = 1, p-value = 4.969e-11
```

2. Are **Regularity** and **Auxiliary** independent in the Dutch **regularity** data?

```
xtabs(~Auxiliary + Regularity, regularity)

##             Regularity
## Auxiliary irregular regular
##   zijn          12      8
##   hebben        108    469
##   zijnheb       39     64
chisq.test(xtabs(~Auxiliary + Regularity, regularity))

## Warning in chisq.test(xtabs(~Auxiliary + Regularity, regularity)): Chi-
## squared approximation may be incorrect

##
## Pearson's Chi-squared test
##
## data: xtabs(~Auxiliary + Regularity, regularity)
## X-squared = 34.555, df = 2, p-value = 3.136e-08
```

Note that in the **regularity** example, there is a warning message, related to the “approximately” condition above. The “ $\chi^2$  approximation” (that the test statistic  $X^2$  follows a  $\chi^2$  distribution) is only valid when the expected count in each cell is “big enough”—otherwise,  $p$  values are anti-conservative. Common rules of thumb for “big enough” are 5 or 10 observations per cell.

It is common to have fewer than 10 observations in some cell in a contingency table, making  $\chi^2$  tests frequently inappropriate. Widespread use of the  $\chi^2$  test is to some extent a holdover from when it was computationally difficult to compute “exact” tests (= no approximation to distribution of test statistic), which require simulation, and there is little reason to use the  $\chi^2$  approximation today. For example, you can run a “chi-squared test” but without assuming the  $\chi^2$  approximation by using the **simulate.p.value** flag to **chi.sq**:

```
chisq.test(xtabs(~Auxiliary + Regularity, regularity), simulate.p.value=TRUE)
```

```
##
## Pearson's Chi-squared test with simulated p-value (based on 2000
## replicates)
##
## data: xtabs(~Auxiliary + Regularity, regularity)
## X-squared = 34.555, df = NA, p-value = 0.0004998
```

which gets rid of the error.

It is still worth knowing about the  $\chi^2$  test and its limitations because it is still frequently used, and in older literature is very widely used. For example, if you are reading a paper where the crucial result relies on a  $\chi^2$  test with  $p = 0.02$  for a contingency table with 5 observations in one cell, you should be suspicious. If the contingency table has 20+ observations per cell, you shouldn’t be.

### 4.1.3 Fisher’s exact test

A good alternative to the  $\chi^2$  test is *Fisher’s exact test*, which as an “exact test” does not place any assumptions on counts per cell. Fisher’s test asks a slightly different question from a  $\chi^2$  test:

- Given the *marginal counts* (row and column totals) in the contingency table, if  $X_1$  and  $X_2$  were independent, how likely would an arrangement of the data at least this extreme be?

For example, for testing independence of `Auxiliary` and `Regularity` for the Dutch `regularity` data:

```
xtabs(~Auxiliary + Regularity, regularity)
```

```
##             Regularity
## Auxiliary irregular regular
##   zijn          12      8
##   hebben        108    469
##   zijnheb       39     64
```

Fisher's test asks: for 159 irregular verbs, assuming independence, how likely would we be to have  $\geq 108$  `hebben`,  $\leq 12$  `zijn`,  $\leq 39$  `zijnheb`, and so on.

To perform Fisher's exact test in R:

```
fisher.test(xtabs(~Auxiliary + Regularity, regularity))
```

```
##
## Fisher's Exact Test for Count Data
##
## data: xtabs(~Auxiliary + Regularity, regularity)
## p-value = 1.52e-07
## alternative hypothesis: two.sided
```

As expected, whether a verb is irregular and which `Auxiliary` it takes are not independent.

## 4.2 Towards logistic regression

Intuitively, in logistic regression (next chapter) we will predict the probability that some event happens (e.g. tapping) as a function of predictors, given a bunch of data where each observation corresponds to observing whether the event happened or not, once ( $Y = 0$  or  $1$ ). It would be tempting to just apply the tool we've learned so far—linear regression—to this task. However, it's not obvious **what to predict** in such a model:

- We can't do a linear regression using  $Y$  as the response, because  $Y$  only takes on the values 0 and 1. We want to model a **probability**.
- However, we can't do a linear regression using a probability as a response, because (among other reasons) probabilities are bounded by 0 and 1 and linear regression assumes that the response variable can be any number. (We don't want the model to be able to predict “probability = 2”.)

In order to predict probabilities using a regression model, we need:

**Goal 1:** A way to think of probabilities on a continuous, unbounded scale.

**Goal 2:** A way to estimate these probabilities, such that the sample statistic is normally distributed.

Goal 2 is necessary so that we can apply all the statistical inference machinery we have used so far to do things like conduct hypothesis tests.

The answer turns out to be to use “log-odds”, for which we must first discuss “odds”.

### 4.2.1 Odds

*Odds* are a way of thinking about probability, as in “how likely is this event to happen versus not happen?”

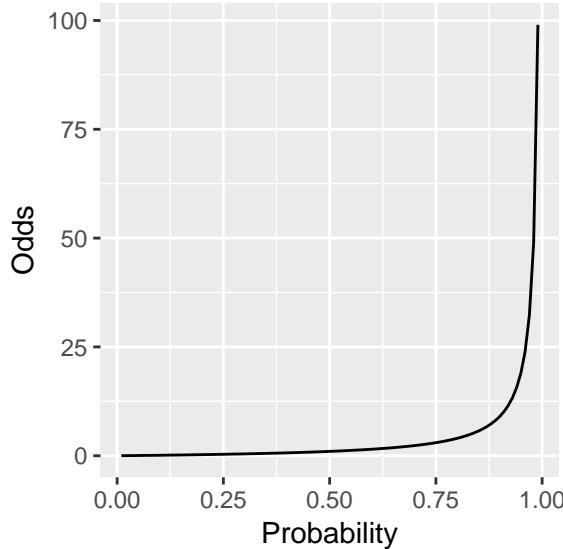


Figure 4.1: Odds as a function of probability.

**Questions:**

Intuitively, what does it mean to say the odds are 3:1 (“three-to-one”) that it will be sunny tomorrow?

The relationship between probabilities ( $p \in [0, 1]$ ) and odds are:

$$\text{Odds}(p) = \frac{p}{1-p}$$

For the example above: if the probability that it is sunny tomorrow is 0.75, then the odds of it being sunny tomorrow are 3 ( $0.75/(1 - 0.75) = 3$ ). The odds of it *not* being sunny tomorrow are 1/3 ( $0.25/(1 - 0.25)$ ), pronounced “one-to-three”, or “three-to-one against sun tomorrow.”

Figure 4.1 shows odds as a function of probability.

Odds may be more intuitive than probabilities, but they do not meet Goal 1 (unbounded scale) because odds are always positive.

### 4.2.2 Log-odds

The *log-odds*, corresponding to a probability  $p$  are:<sup>1</sup>

$$\text{log-odds}(p) = \log \frac{p}{1-p}$$

The following figure shows log-odds as a function of probability.

Log-odds meet our goals: they range from  $-\infty$  to  $\infty$ , and are nearly linear as a function of  $p$ , so long as  $p$  isn’t too close to 0 or 1. It also turns out that log-odds satisfy our Goal 2 (normally-distributed sampling statistic), as discussed below.

Log-odds do have an important drawback: they are unintuitive, compared to odds or probability. With some practice you can learn to think in log-odds.

---

<sup>1</sup>Note that log here is base  $e$ , where  $e = 2.718\ldots$  is the base of the *natural logarithm*.

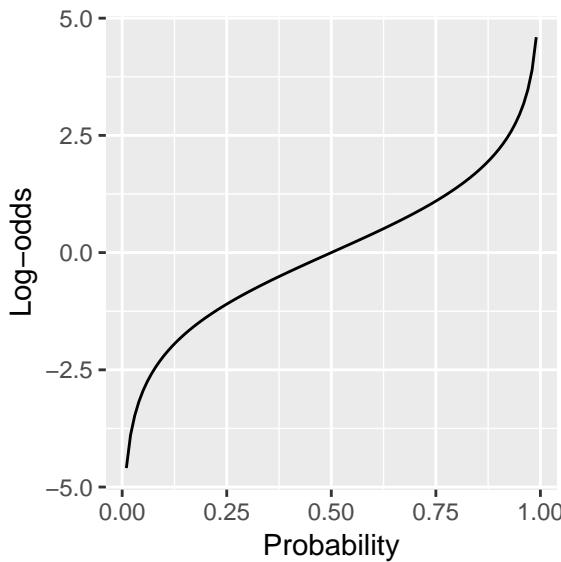


Figure 4.2: Log-odds as a function of probability

**Exercise 2: log-odds practice**

What are the log-odds corresponding to these probabilities?

- $p = 0.5$
- $p = 0.73$
- $p = 0.88$
- $p = 0.953$

```
## [1] 0
## [1] 0.9946226
## [1] 1.99243
## [1] 3.009467
```

This example illustrates:

- Probability space is contracted in log-odds, as we approach 0 or 1: a change of 1 in log-odds corresponds to a smaller and smaller change in  $p$ .
- Log-odds of -3 to 3 encompasses 90% of probability space ( $\approx p \in (0.05, 0.95)$ ).

Note that 4 is huge in log-odds ( $p \approx 0.983$ ).

**Questions:**

What is  $p$  corresponding to log-odds of -4?

```
## [1] 0.01798621
```

**4.2.3 Odds ratios**

Let's start with an example from the `tapping` data, looking at how the likelihood of tapping depends on syntax:

- $Y = \text{tapped}$  (1 = yes, 0 = no)
- $X = \text{syntax}$  (1 = transitive, 0 = intransitive)

Suppose that the proportions of cases in each cell for our data are:

	$Y = 1$	$Y = 0$
$X = 1$	0.462	0.037
$X = 0$	0.43	0.07

We would like a measure of how much more likely tapping is when  $X = 1$  than when  $X = 0$ . One way to do this is to calculate the odds of tapping in each case, then take their ratio:

- when  $X = 1$ :  $\frac{0.462}{0.037} = 12.49$
- when  $X = 0$ :  $\frac{0.43}{0.07} = 6.14$
- Ratio:  $\frac{12.49}{6.14} = 2.03$

Thus, the odds of tapping are roughly doubled for transitive verbs, relative to intransitive verbs.

To define this *odds ratio* more generally, suppose that two binary random variables  $X$  and  $Y$  co-occur with these (population) probabilities:

	$Y = 1$	$Y = 0$
$X = 1$	$p_{11}$	$p_{10}$
$X = 0$	$p_{01}$	$p_{00}$

The odds that  $Y = 1$  are then:

- When  $X = 1$ :  $\frac{p_{11}}{p_{10}}$
- When  $X = 0$ :  $\frac{p_{01}}{p_{00}}$

and the *odds ratio*, as a population statistic, is defined as:

$$OR = \frac{p_{11}/p_{10}}{p_{01}/p_{00}}$$

The odds ratio describes how much the occurrence of one (binary) event depends on another (binary) event, and is often used as an intuitive measure:

- “Your odds of developing lung cancer are halved if you quit smoking.”
- “Marie’s odds of winning the election tripled from January to March.”

Note that odds ratios are interpreted multiplicatively: it makes sense to say “ $X$  raises the odds of  $Y$  by a factor of 2”, but not “ $X$  raises the odds of  $Y$  by 2”.

Odds ratios are in part a way of thinking about changes in probability. For the tapping example:

$$P(\text{tapped} | \text{transitive}) = 0.926, \quad P(\text{tapped} | \text{intransitive}) = 0.86$$

Thus, the change in probability from intransitive to transitive sentences is 0.066. However, odds ratios are *not* equivalent to changes in probability, because with odds ratios, what a change from  $p$  to  $p + \Delta p$  corresponds to depends on the probability ( $p$ ) you start at. For example, Figure 4.3 shows the odds ratio corresponding to a change in probability of 0.1, as a function of the starting probability.

For a change in probability from 0.5 to 0.6, the odds increase by about 1.5x, while for a change in probability from 0.85 to 0.95 the odds increase by 3.35x.

To summarize:

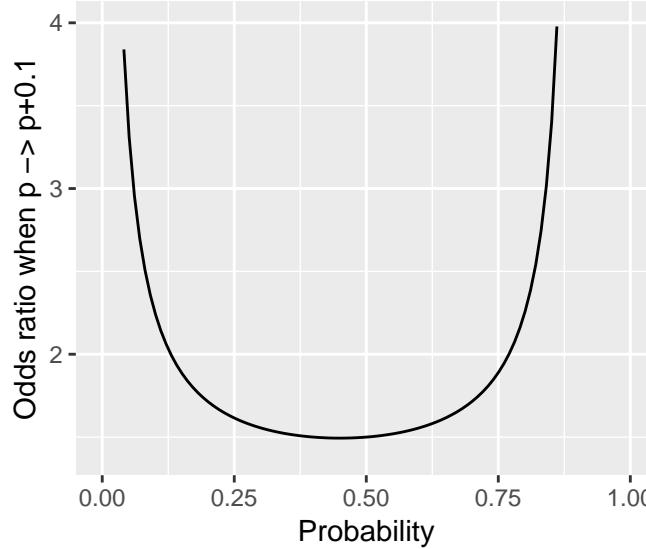


Figure 4.3: Odds ratio when a probability ( $p$ ) is increased by 0.1, as a function of  $p$

- Odds have a one-to-one relationship with probabilities
- Changes in odds don't straightforwardly correspond to changes in probabilities.

The non-equivalence between changes in odds and changes in probability is important to remember when interpreting logistic regressions, which are in log-odds space.

#### 4.2.4 Log odds: sample and population

The odds ratio is a useful *summary statistic* of the degree of dependence between two binary variables, analogous to correlations ( $r$ ,  $\rho$ ) for summarizing dependence between two continuous variables. Because we never actually observe the population value of the odds ratio—which is based on probabilities of each combination of  $X$  and  $Y$ —we need a *sample statistic* to estimate it from observed counts for each  $X/Y$  combination.

Suppose we observe the following counts:

		$Y = 1$	$Y = 0$
		$n_{11}$	$n_{10}$
$X = 1$	$n_{01}$		
	$n_{00}$		

To calculate the empirical odds ratio, we estimate each probability from the observed counts. For example:

$$P(Y = 1|X = 1) \approx \frac{n_{11}}{n_{11} + n_{10}}, \quad P(Y = 0|X = 1) \approx \frac{n_{10}}{n_{11} + n_{10}} \quad (4.3)$$

$$\implies Odds(Y = 1|X = 1) = \frac{n_{11}}{n_{10}} \quad (4.4)$$

Similarly estimating  $Odds(Y = 1|X = 1)$ , then taking the logarithm of  $Odds(Y = 1|X = 1)/Odds(Y = 1|X = 0)$ , gives the sample change in log odds:

$$\hat{L} = \log \left[ \frac{n_{11}/n_{10}}{n_{01}/n_{00}} \right]$$

It turns out that  $\hat{L}$  is normally distributed with mean  $\log(OR)$ , where  $OR$  is the population value of the odds ratio. Thus, if we want to do inferential statistics on the effect of binary  $X$  on binary  $Y$ , it makes sense to work with **log-odds**, rather than probabilities.

#### 4.2.4.1 Calculation and interpretation

- The function from probability to log-odds is called *logit* ( $\text{logit}(p)$ )
- The function from log-odds to probability is called *inverse logit* ( $\text{logit}^{-1}(x)$ ), a.k.a. the “logistic function” ( $1/(1 + e^{-x})$ ).

Both can be calculated using R functions, such as `logit` and `invlogit` in the `arm` package.

Note that **adding** in the log-odds scale corresponds to **multiplying** in the odds scale, by a power of  $e$ .

For example, changing log-odds by 1 corresponds to multiplying the odds ratio by  $e$ :

$$\begin{aligned}\log(OR_1) = 0.69 &\implies OR_1 = e^{0.69} = 2 \\ \log(OR_2) = 1.69 &\implies OR_2 = e^{1.69} = 5.42 = 2 \cdot e\end{aligned}$$

#### Exercise 3

Suppose that in the `tapping` data, one participant’s tapping rate (i.e. probability of tapping) is 0.90 in fast speech and 0.70 in slow speech.

- What are the participant’s **odds** of tapping in fast and slow speech (two numbers)?

```
## [1] 9
## [1] 2.333333
• What is the odds ratio of tapping (in fast vs. slow speech) (one number)?
## [1] 3.857143
• What is the equivalent change in log-odds?
## [1] 1.349927
```

### 4.3 Other readings

Most introductory statistics books cover contingency tables, chi-squared tests, and Fisher exact tests, including:

- General audience: Dalgaard (2008) Ch. 8, Agresti (2007) Ch. 1-2
- For psychologists and language scientists: Navarro (2015) Ch. 12, Johnson (2008) Ch. 3.1, 5.1-5.3

The Agresti book is a particularly good and accessible resource for categorical data analysis generally. (Agresti (2003) is Agresti’s more technical/comprehensive book, which is the reference on CDA.)

Some discussion of odds and odds ratios is given by Agresti (2007) Ch. 2, Gelman and Hill (2007) 5.2, Crawley (2015) Ch. 14.

## 4.4 Solutions

### 4.4.1 Solutions to Exercise 1:

Estimated Counts for:

- Prominence = Noun, Context = Alternative cell = 118.5542
- Prominence = Adjective, Context = NoAlternative cell = 88.55422
- Prominence = Adjective, Context = NoAlternative cell = 121.4458



# Chapter 5

## Logistic regression

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(languageR)
library(ggplot2)
library(dplyr)
library(arm)
library(boot)

## loads givennessMcGillLing620.csv from OSF project for Wagner (2012) data
givenness <- read.csv(url("https://osf.io/q9e3a/download"))

## make standardized (numeric/centered/scaled) versions of 'givenness' dataset predictors:
givenness <- mutate(givenness,
                     conditionLabel.williams = arm::rescale(conditionLabel),
                     clabel.williams = arm::rescale(conditionLabel),
                     npType.pronoun = arm::rescale(npType),
                     npType.pron = arm::rescale(npType),
                     voice.passive = arm::rescale(voice),
                     order.std = arm::rescale(order),
                     stressshift.num = (as.numeric(stressshift) - 1),
                     acoustics.std = arm::rescale(acoustics)
)
```

**Note:** Answers to some questions/exercises not listed in text are in Solutions

### 5.1 Simple logistic regression

Analogously to “simple linear regression”, which is predicting a continuous  $Y$  from a single predictor  $X$ , in *simple logistic regression* we predict a binary  $Y$ .

Notation:

- Our sample consists of  $n$  observations:  $(x_1, y_1), \dots, (x_n, y_n)$
- The **response**  $Y$  is binary: each  $y_i = 1$  or  $0$ .

- There is a single **predictor**  $X$ , which may be continuous or discrete.

We model the **log-odds** of  $Y = 1$  vs.  $Y = 0$  for the  $i^{\text{th}}$  observation:

$$\log \left[ \frac{P(y_i = 1)}{P(y_i = 0)} \right] \equiv \text{logit}(P(y_i = 1))$$

as a function of the single predictor:

$$\text{logit}(P(Y = 1)) = \beta_0 + \beta_1 X$$

This is the model in “logit space”, where the right-hand side of the equation (called the *linear predictor*) looks the same as for simple linear regression except for the lack of an error term, while the left-hand side of the equation looks similar to SLR modulo the logit function (called the *link function*).

The same model can be written in “probability space”:

$$P(Y = 1) = \text{logit}^{-1}(\beta_0 + \beta_1 X) \quad (5.1)$$

$$= \frac{1}{1 + e^{-(\beta_0 + \beta_1 X)}} \quad (5.2)$$

The model in Eq. (5.1) predicts the log-odds of  $Y = 1$  for a single observation as:

$$\text{logit}(\hat{p}_i) = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

### 5.1.1 Hypothesis testing

The test statistic

$$z = \frac{\hat{\beta}_1}{SE(\hat{\beta}_1)} \quad (5.3)$$

turns out to be approximately normally distributed (e.g. Agresti (2003) 5.5.2).<sup>1</sup> This means we can perform a *Wald test* of the null hypothesis to show that there is no relationship between  $P(Y = 1)$  and  $X$ :

$$H_0 : \hat{\beta}_1 = 0$$

and get a  $p$ -value and confidence interval for the slope ( $\beta_1$ ). The same is true for the intercept ( $\beta_0$ ).

### 5.1.2 Interpreting the coefficients: Logit, odds, and probability

The regression coefficients,  $\beta_0$  and  $\beta_1$ , can be interpreted in terms of log-odds, odds, or probability:

Intercept:

- $\beta_0$ : **predicted log-odds** of  $Y = 1$  when  $X = 0$
- $e^{\beta_0}$ : predicted odds of  $Y = 1$  when  $X = 0$
- $\text{logit}^{-1}(\beta_0)$ : probability of  $Y = 1$  when  $X = 0$

<sup>1</sup>This “large-sample approximation” is what is assumed by `glm` in R in calculating logistic regression coefficient significances. However the Wald test becomes unreliable for small enough samples ( $n < 100$  is sometimes given) or very large  $|\beta|$ , and for these reasons Agresti (2003) (5.2.1) recommends always using a likelihood ratio test to assess coefficient significances. The LR and Wald tests will give very similar significances for large samples.

Slope:

- $\beta_1$ : predicted change in log-odds of  $Y = 1$  for a unit change in  $X$
- $e^{\beta_1}$ : predicted amount odds of  $Y = 1$  is multiplied by for a unit change in  $X$
- The meaning of  $\beta_1$  in probability depends on the value of  $X$

### 5.1.3 Logistic regression as a GLM

Logistic regressions are fit in R using the `glm()` function with the option `family="binomial"`.

Why? Logistic regression is one type of *generalized linear model* (GLM): a family of models that look like linear regression, but with different choices for each part of Eq. (5.1): link function, linear predictor, probability distribution over  $Y$ . Logistic regression assumes a binary response  $Y$ , and uses the logit link function. Because the logit link is the most commonly used for binomially-distributed data (of which a binary response is a special case, with  $n = 1$ ), specifying `family="binomial"` gives a logit link by default (it's the same as writing `family = binomial(link = "logit")`).<sup>2</sup> There are many types of GLM, including some described in an Appendix below, but we'll only discuss logistic regression in this book.

#### Example 1: Single continuous predictor

In the `givenness` data (described in detail here), whether stress shifted is captured by the continuous variable `acoustics`, which is a composite of prosodic measurements. We check using a simple logistic regression how the perceptual measure is related to the acoustic measure. That is:

- $Y$  is `stressshift`
  - = 1 if shifted, 0 if not
- $X$  is `acoustics`
  - (centered + scaled)

To fit and summarize this model:

```
mod1 <- glm(stressshift ~ acoustics.std, data=givenness, family="binomial")
summary(mod1)
```

```
##
## Call:
## glm(formula = stressshift ~ acoustics.std, family = "binomial",
##      data = givenness)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -1.6271   -0.8924   -0.6532    1.1213    2.2418
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.6897    0.1167  -5.908 3.47e-09 ***

```

<sup>2</sup>There is a further complication here: we have presented logistic regression as modeling data with a binary outcome, that is a *Bernoulli* random variable  $y_i$  for the  $i^{\text{th}}$  observation, where we model  $P(y_i = 1)$ . Why is the option to `glm()` then `family = "binomial"`, given that a binomial distribution refers to  $n \geq 0$  trials with probability  $p$  of any outcome being 1? (The Bernoulli case is when  $n = 1$ .) It turns out that logistic regression can be characterized either as the probability of a binary outcome for each observation, or as modeling the number of “hits” for a Bernoulli distribution over  $n$  outcomes, for each set of observations that share the same predictor values. The `glm()` with `family = "binomial"` can handle data in either format ( $y_i$  binary or giving the number of 0’s and 1’s for a set of predictor values), and if the response variable in the data passed to `glm()` only has two values, R assumes that you’re modeling a binary outcome.

```

## acoustics.std   1.6371      0.2588    6.325 2.54e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24  on 381  degrees of freedom
## Residual deviance: 448.07  on 380  degrees of freedom
## AIC: 452.07
##
## Number of Fisher Scoring iterations: 3

```

A few notes about the model output, compared to the output of the linear regression models we've seen so far:

- There are no references to “residuals” or “residual standard error”. This is because the logistic regression equation Eq. (5.1) has no error term, so there are no residuals.
- There are no “R-squared” values.  $R^2$  is not a well-defined concept for logistic regression, for which different measures of goodness of fit are used (discussed below).
- The model table contains `z statistic` and `Pr(>|z|)` columns rather than `t` and `Pr(>|t|)` columns. This is because the test statistic  $\hat{\beta}_i/SE(\hat{\beta}_i)$  for logistic regression ( $z$  in Eq. (5.3)) looks like a z-score and follows a normal distribution, while the test statistic follows a  $t$  distribution for linear regression.

### 5.1.3.1 Interpreting the model: Odds and log-odds

#### Questions:

- What are the log-odds of stress shifting when `acoustics` = 0 ? (Remember this means “mean value of `acoustics`”, because it’s centered.)

```
## [1] -0.6896608
```

- What is  $P(\text{stress shift})$  when `acoustics` = 0 ?

```
## [1] 0.3341085
```

- Increasing `acoustics` by 1 (= 2 SD) corresponds to increasing the log-odds of stress shifting by \_\_\_\_\_ ?

```
## [1] 1.637052
```

- Increasing `acoustics` by 1 (= 2 SD) corresponds to multiplying the odds of stress shifting by \_\_\_\_\_ ?

```
## [1] 5.139996
```

### 5.1.3.2 Interpreting the model: Probability

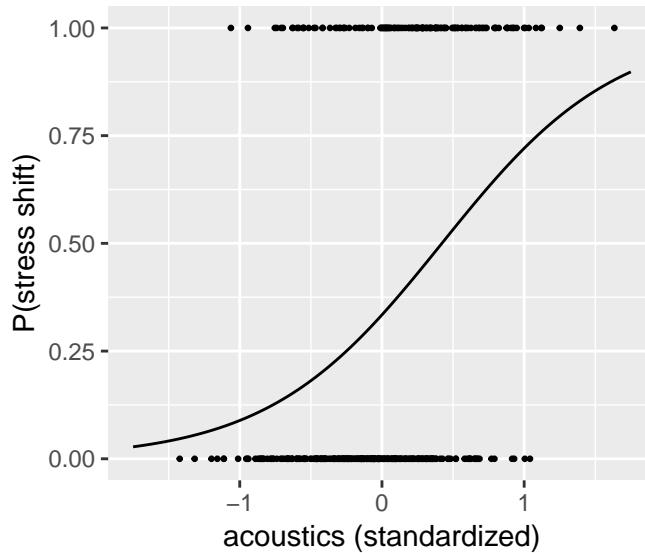
We can plot the predicted  $P(\text{stress shift})$  as a function of `acoustics` to get a sense of what the model predicts:

```

## set up dataframe with range of acoustics.std we want to predict over
newdata <- data.frame(acoustics.std=seq(-1.75,1.75, by=0.01))
## get the model's predictions in log-odds space
newdata$predLogit <- predict(mod1, newdata=newdata)
## transform those predictions to probability space
newdata <- newdata %>% mutate(predP=invlogit(predLogit))

```

```
ggplot(aes(x=acoustics.std, y=predP), data=newdata) +
  geom_line() +
  geom_point(data=givenness, aes(y=(as.numeric(stressshift)-1)), size=0.5) +
  xlab("acoustics (standardized)") +
  ylab("P(stress shift)")
```

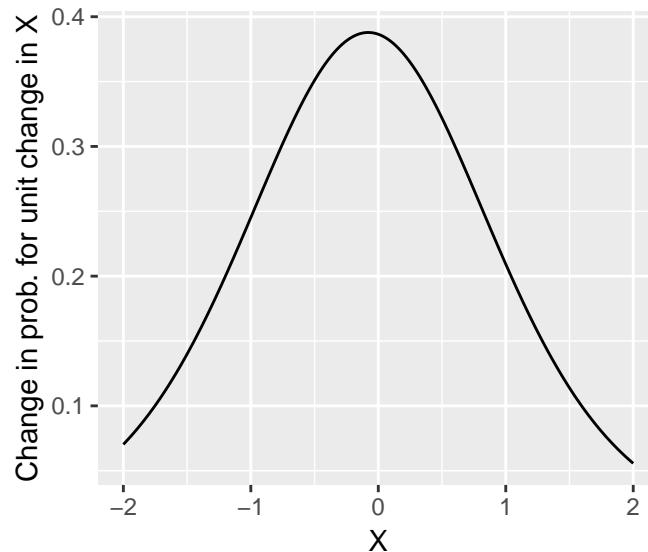


As expected, there is a strong relationship between the acoustic proxy for stress shift, and whether stress is perceived as having shifted.

Let's work out in more detail what “increase in log-odds” means in probability space for this model:

- When `acoustics` = 0:
  - log-odds = -0.68
  - odds = 0.506
  - $P(\text{stress shift}) = 0.336 = \text{logit}^{-1}(-0.68)$
- When `acoustics` = 1:
  - log-odds = 0.947
  - odds =  $2.58 = e^{0.947}$
  - $P(Y = 1) = 0.720 = \text{logit}^{-1}(0.947)$

Thus, the change in probability  $P(Y = 1)$  when you increase  $X$  by 1 is 0.348 here ( $X : 0 \rightarrow 1$ ) However, in general what change in probability results when  $X$  is increased by 1 (a “unit change”) will depend on where you start from:



Note that the greatest change in probability occurs when  $P(Y = 1|X) = 0.5$  (at  $X = 0.69 \implies \beta_0 + \beta_1 X = 0$ ). The change in probability here ( $\approx 0.4$ ) is roughly 1/4 of the fitted slope value:

$$\frac{\beta_1}{4} = \frac{1.637}{4} = 0.41$$

It turns out that this relationship holds more generally, and the *divide-by-4 rule* can be used to roughly interpret logistic regression slopes in terms of probability (Gelman and Hill (2007), p. 82): the slope (in log-odds) is about 4x the maximum change in probability.

### Example 2: Single categorical predictor

Let's now predict the probability of stress shifting (`stressshift`) for the `givenness` data, as a function of the NP type (`npType`). This predictor is “dummy coded”, with values:

- $X = 0$ : Full NP
- $X = 1$ : Pronoun

The regression model is still Eq. (5.1).

**Questions:** What are the interpretations of the regression coefficients?

- Intercept ( $\beta_0$ )
- Slope ( $\beta_1$ )

To fit and summarize this model:

```
mod2 <- glm(stressshift ~ npType, data=givenness, family="binomial")
summary(mod2)
```

```
##
## Call:
## glm(formula = stressshift ~ npType, family = "binomial", data = givenness)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.014  -1.014  -0.850   1.350   1.545
##
## Coefficients:
```

```

##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.8321    0.1587 -5.244 1.57e-07 ***
## npTypepronoun 0.4353    0.2159  2.016  0.0438 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24  on 381  degrees of freedom
## Residual deviance: 492.14  on 380  degrees of freedom
## AIC: 496.14
##
## Number of Fisher Scoring iterations: 4

```

The model predicts  $P(\text{stress shift})$  to be:

- Full NPs:  $\text{logit}^{-1}(-0.832) = \mathbf{0.303}$
- Pronouns:  $\text{logit}^{-1}(-0.832 + 0.435) = \mathbf{0.402}$

We can compare to the observed proportion of stress-shifted observations for each NP type:

```

## Observed condition means:
prop.table(xtabs(~npType + stressshift, data=givenness), margin=1)

```

```

##           stressshift
## npType      noshift     shift
##   full      0.6968085 0.3031915
##   pronoun  0.5979381 0.4020619

```

and see that the model matches these proportions exactly (`shift` column). This makes sense, as predicting these two numbers is all the model has to do for simple logistic regression with one categorical predictor.

#### 5.1.4 Differences from linear regression: Fitting and interpretation

There are two important differences between logistic regression and linear regression, for our purposes:

1. The **relationship between the response and what's being modeled**. In linear regression, these are the same thing:  $Y$  is the response variable, which is also what the regression is modeling. In logistic regression, these are not the same thing: we observe  $Y$ , which takes on values 0 or 1, but we model the *expected value of  $Y$*  ( $E(Y)$ ): the probability that  $Y = 1$ .
2. The **presence of an error term**. In linear regression, there is an error term  $\epsilon_i$ , capturing the difference between the fitted value ( $\hat{y}_i$ ) and the actual value ( $y_i$ ) for each observation, along with the linear predictor ( $\beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip}$ ). In logistic regression, there is no error term: the right-hand side of Eq. (5.1) predicts  $P(Y = 1)$  using just the linear predictor.

Why? For a Bernoulli random variable  $Y$  (basically, a coin that comes up “heads” with some probability  $p$ ), the variance and expectation only depend on one unknown parameter,  $p$ :

$$E(Y) = p, \quad \text{Var}(Y) = p(1 - p)$$

So unlike for linear regression, where the mean ( $\mu$ ) and variance ( $\sigma$ ) in our model of the response can be tweaked independently to better fit the data, in logistic regression these two things are not independent. By fitting the mean (probability that  $Y = 1$ ), we have already determined the variance.

Both differences make interpreting and fitting logistic regression models less straightforward than linear regression models.

### 5.1.5 Fitting a logistic regression model

For linear regression, it was possible to estimate the regression coefficients by “least squares”: minimizing the difference between the model’s prediction and the observed data ( $\epsilon_i^2$ ), across all points. Least-squares has a *closed-form* solution: the regression coefficients can be solved for by plugging the predictor and response values for all observations into an equation.

For logistic regression, there are no residuals, so we can’t use a least-squares method. Instead, we use a likelihood-based method: determine the probability of the observed data, given values of the regression coefficients (the data *likelihood*), then find the regression coefficient values that maximize this probability.

For simple logistic regression: for any  $\beta_0$  and  $\beta_1$ , the likelihood of the data is:

$$L(\beta_0, \beta_1) = \prod_{i=1}^n [\text{logit}^{-1}(\beta_0 + \beta_1 x_i)]^{y_i}$$

We choose  $\hat{\beta}_0, \hat{\beta}_1$  that maximize the likelihood: the *maximum likelihood* (ML) estimates. There is no closed form solution for  $\hat{\beta}_0, \hat{\beta}_1$ , which must instead be estimated numerically, using an estimation algorithm.<sup>3</sup>

### 5.1.6 Interpretation

You of course don’t need to understand the math to fit `lm()` or `glm()` models—R will do this for you automatically. However, it can be helpful to know some basic aspects of how these models are fitted to interpret model output, and some otherwise cryptic error messages. For example:

- (Dispersion parameter for binomial family taken to be 1) in logistic regression output: related to the relationship between mean and variance—the amount of variability in the data is assumed to be similar to what you’d expect based on the estimated mean. If it’s not, your data is “over-dispersed” or “under-dispersed”.<sup>4</sup>
- Number of Fisher Scoring iterations: X in logistic regression output: refers to the numeric algorithm used to fit logistic regression models (see footnote 3).
- There is no  $F$  statistic or  $p$  value in logistic regression output: this is because there are no residuals, so the “full” and “reduced” models cannot be compared via their residual sums of squares (which are used to conduct an  $F$  test for linear regression).
- If you try to fit a logistic regression model with just  $Y = 1$  response values (no  $Y = 0$  values), you’ll get the message `glm.fit: algorithm did not converge`:

```
badmodel <- glm(stressshift ~ acoustics.std, data=filter(givenness, stressshift=='shift'), family="binom")
```

```
## Warning: glm.fit: algorithm did not converge
```

This means that the numeric algorithm used to maximize likelihood couldn’t find a good solution for the regression coefficients. This makes sense, because the model is trying to fit 100% probabilities (so log-odds =  $\infty$ ), and has no data on the basis of which it can estimate the slope ( $\beta_1$ ).

---

<sup>3</sup>`glm()` uses the *Fisher scoring algorithm* by default, hence the reference to “Number of Fisher Scoring iterations” in the `summary()` output of a model fitted with `glm()`.

<sup>4</sup>Dispersion is not a concern for logistic regression models where each observation is just 0 or 1, the only kind of logistic regression covered in this chapter. (As opposed to more complex flavors, like observing proportions between 0 and 1 over multiple observations.)

## 5.2 Evaluating logistic regression models

We would like a measure of goodness-of-fit for logistic regression models: how well does the model predict  $Y$ , compared to a baseline model? For linear regression, we used  $R^2$  to quantify how similar the model's predictions ( $\hat{y}_i$ ) were to the observations ( $y_i$ ), relative to a baseline model where the model's prediction is always the grand mean— $R^2$  was simply the correlation between observed and predicted values.

However, for logistic regression we cannot directly compare model predictions ( $p_i$ : (log-odds of) probabilities) to observations ( $y_i$ : 0 or 1), so we need a different measure of goodness-of-fit. We consider three commonly used options:

1. Likelihood ratio test
2. Classification accuracy
3. Pseudo- $R^2$

to compare a logistic regression model to a baseline (intercept-only model). We assume here that the logistic regression model has just one predictor  $X$ , and denote this model by  $M_1$  and the baseline model by  $M_0$ .

### 5.2.1 Likelihood ratio test

This option is analogous to the  $F$  test for linear regression models, where the full model is compared to the intercept-only model, and we test whether the change in sum-of-squares is significant, given the added predictors in the full model.

Logistic regression models don't have sums-of-squares because they don't have residuals. Instead, we compare the **likelihood** (Eq. (5.1.5)) of the two models: is the change in likelihood between  $M_0$  and  $M_1$  significant?

Most methods involving likelihood use log-transformed likelihood, mostly to avoid numerical issues when computing with very small numbers. Instead of working with straight log-likelihood, it is also customary to define the *deviance* of a model  $M$  (Agresti (2007) 3.4.3), as

$$D = -2 \log L(M)$$

Thus, a better model (higher likelihood) has lower deviance.

The difference in deviance ( $\Delta D$ ) between models with ( $M_1$ ) and without ( $M_0$ ) the single predictor is related to the log of their *likelihood ratio*:

$$\Delta D = -2 \log \frac{L(M_0)}{L(M_1)}$$

For large enough sample size,  $\Delta D$  follows a  $\chi^2(1)$  distribution if the slope of  $X$  ( $\beta_1$ ) is 0. Thus, we can use  $\Delta D$  as a test statistics for the null hypothesis that  $\beta_1 = 0$ . This kind of test is called a *likelihood ratio test*. The  $p$  value of this test gives a measure of goodness of fit, while  $\Delta D$  gives a sort of measure of variance accounted for (if you think of deviance as capturing “variance” from the perfect model).

For our Example 1, an LR test is conducted in R as follows:

```
mod1 <- glm(stressshift ~ acoustics.std, data=givenness, family="binomial")
mod0 <- glm(stressshift ~ 1, data=givenness, family="binomial")
## LR test of the effect of acoustics.std in mod1
anova(mod0, mod1, test='Chisq')
```

```

## Analysis of Deviance Table
##
## Model 1: stressshift ~ 1
## Model 2: stressshift ~ acoustics.std
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      381     496.24
## 2      380    448.07  1    48.17 3.909e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Note that for simple linear regression, this is just another way to test the hypothesis that  $\beta_1 = 0$ , in addition to the  $z$  test reported in the regression summary.

```
summary(mod1)
```

```

##
## Call:
## glm(formula = stressshift ~ acoustics.std, family = "binomial",
##      data = givenness)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q      Max
## -1.6271 -0.8924 -0.6532  1.1213  2.2418
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.6897    0.1167 -5.908 3.47e-09 ***
## acoustics.std 1.6371    0.2588  6.325 2.54e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24 on 381 degrees of freedom
## Residual deviance: 448.07 on 380 degrees of freedom
## AIC: 452.07
##
## Number of Fisher Scoring iterations: 3

```

However, for multiple regression models with more than one predictor, the LR test gives a measure of overall model fit, and is not testing the same thing as the  $z$  tests reported by `summary(model)` for each predictor.

### 5.2.2 Classification accuracy

A second way to assess model quality comes from thinking of the model as predicting  $Y$ , as follows:

$$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_i > 0.5 \\ 0 & \text{if } \hat{p}_i \leq 0.5 \end{cases}$$

That is, just predict  $Y = 1$  if the predicted log-odds are positive, and  $Y = 0$  if the predicted log-odds are negative. We can then define *classification accuracy* as the percentage of observations where the predicted and observed values are the same ( $\hat{y}_i = y_i$ ).

Classification accuracy always needs to be compared to a baseline, usually “how often would we classify correctly if we chose the most common case?” Formally, if  $n_1$  and  $n_2$  are the number of observations where

$y_i = 0$  and  $y_i = 1$  in the data, the baseline classification accuracy is:

$$\max\left(\frac{n_1}{n}, \frac{n_2}{n}\right)$$

Here are functions for computing the classification accuracy and baseline accuracy of a logistic regression model:

```
## function for computing accuracy of a logistic regression model
## (on the dataset used to fit the model)
## lrMod = fitted model
## responseVar = name of response variable for lrMod
##
## adapted from: https://www.r-bloggers.com/evaluating-logistic-regression-models/
lrAcc <- function(lrMod, responseVar){
  ## convert response variable into a factor if it's not one
  if(!is.factor(model.frame(lrMod)[,responseVar])){
    model.frame(lrMod)[,responseVar] <- as.factor(model.frame(lrMod)[,responseVar])
  }
  ## model predictions in log-odds
  preds = predict(lrMod, newdata=model.frame(lrMod))
  ## transform to 0/1 prediction
  preds <- ((sign(preds)/2)+0.5)

  ## response variable values, transformed to 0/1
  y <- (as.numeric(model.frame(lrMod)[,responseVar])-1)

  ## how often is prediction the same as the actual response
  acc <- sum(preds==y)/length(preds)

  return(acc)
}

## baseline accuracy for a logistic regression model lrMod
## with a given response variable
baselineAcc <- function(lrMod, responseVar){
  response <- model.frame(lrMod)[,responseVar]
  tab <- table(response)
  return(max(tab)/sum(tab))
}
```

## baseline accuracy

For our Example 1 (`stressshift ~ acoustics.std`), the classification accuracy is

```
lrAcc(mod1, 'stressshift')
```

```
## [1] 0.7015707
```

Compared to a baseline of 0.65 (using `baselineAcc(mod1, 'stressshift')`). The model performs better than the baseline, but its performance is less impressive than it seems (70%) given that the most common case (not shifting stress) accounts for 65% of the data.

### 5.2.3 Pseudo- $R^2$

There is no equivalent to  $R^2$  for logistic regression, meaning a quantity with similar properties and multiple interpretations:

- Fraction reduction in sum-of-squares (there is no sum-of-squares)
- Degree of “variance” accounted for (“variance” isn’t well-defined)
- Squared correlation between fitted and observed values (different scales: probabilities versus 0/1)

Nonetheless, we might want an  $R^2$ -like quantity that at least has similar properties, if we find such measures easier to interpret than classification accuracy or an LR test result. A number of *pseudo- $R^2$*  measures exist, of which the two most common are:

- **Cox-Snell** pseudo- $R^2$ : a value  $\geq 0$
- **Nagelkerke** pseudo- $R^2$ : a value between 0 and 1 (like  $R^2$  for linear regression)

Both measures are related to the likelihood ratio of the full and reduced (intercept-only) model. Pseudo- $R^2$  measures should not be taken too seriously, but can be useful, for example for comparing goodness of fit between logistic regression and linear regression models. We won’t consider these methods further, but they are reported in some papers.

#### Example

How well does our Example 2 model (`stressshift ~ npType`) do at predicting whether stress shifts?

```
mod2 <- glm(stressshift ~ npType, data=givenness, family="binomial")
```

By the likelihood ratio test:

```
## LR test of the effect of npType in mod2
anova(mod2,mod0, test='Chisq')
```

```
## Analysis of Deviance Table
##
## Model 1: stressshift ~ npType
## Model 2: stressshift ~ 1
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       380    492.14
## 2       381    496.24 -1   -4.0972  0.04295 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

there is a (barely) significant effect of `npType`, corresponding to a difference in deviance of  $\Delta D = 4.1$ .

However, classification accuracy:

```
## accuracy of mod2
lrAcc(mod2, 'stressshift')
```

```
## [1] 0.6465969
## it's the same as the baseline's accuracy
baselineAcc(mod1, 'stressshift')
```

```
## [1] 0.6465969
```

does not differ between the baseline and full models. This example illustrates a couple of points:

- Different methods for comparing two models won't necessarily give the same qualitative answers—as we already saw with AIC versus BIC-based model selection.
- Classification accuracy is a blunter tool than an LR test—it does not reflect effects that are small compared to baseline accuracy, because in order to affect classification accuracy an effect has to be big enough to change the sign of the predicted log odds.

## 5.3 Multiple logistic regression

Like for linear regression, generalizing from one predictor to multiple predictors is straightforward for logistic regression.

We assume there are  $p$  predictors ( $X_1, \dots, X_p$ ) of a binary response  $Y$ , where each  $X_i$  can be continuous or categorical. The logistic regression model is now:

$$\text{logit}(P(Y = 1)) = \beta_0 + \beta_1 X_1 + \dots + \beta_p X_p$$

Rewriting this model in terms of odds by exponentiating each side:

$$\frac{P(Y = 1)}{P(Y = 0)} = e^{\beta_0} e^{\beta_1 X_1} \dots e^{\beta_p X_p}$$

or, rewriting with  $\alpha_i = e^{\beta_i}$ :

$$\frac{P(Y = 1)}{P(Y = 0)} = \alpha_0 \alpha_1^{X_1} \dots \alpha_p^{X_p}$$

That is, each predictor's effect on the odds are **multiplicative**. A predictor that has no effect corresponds to multiplying by 1 ( $\alpha_i = 1$ ), while predictors with “negative” effects decrease the odds ( $\alpha_i < 1$ ) and predictors with “positive” effects increase the odds ( $\alpha_i > 1$ ).

### 5.3.1 Likelihood ratio test: General case

The likelihood ratio test generalizes to the case of comparing two nested logistic regression models, with  $p$  and  $q$  predictors:

$$M_0 : X_1, \dots, X_p \tag{5.4}$$

$$M_1 : X_1, \dots, X_p, X_{p+1}, \dots, X_q \tag{5.5}$$

With the difference in deviance again defined as

$$\Delta D = -2 \log \frac{L(M_0)}{L(M_1)}$$

$\Delta D$  approximately follows a  $\chi^2(q - p)$  distribution (Agresti (2003) 3.4.4), if

$$\beta_{p+1} = \beta_{p+2} = \dots = \beta_q = 0$$

Thus, we can use  $\Delta D$  as a test statistic for the null hypothesis that all the added predictors have no effect:

$$H_0 : \beta_{p+1} = \beta_{p+2} = \dots = \beta_q = 0$$

### 5.3.2 Worked example

In this example, we model the probability of shifting stress (`stressshift`) for the `givenness` data, as a function of four predictors—all standardized (see code chunk at the beginning of this chapter):

1. Condition: `conditionLabel.williams`
2. NP Type: `npType.pronoun`
3. Voicing: `voice.passive`
4. Trial number: `order`

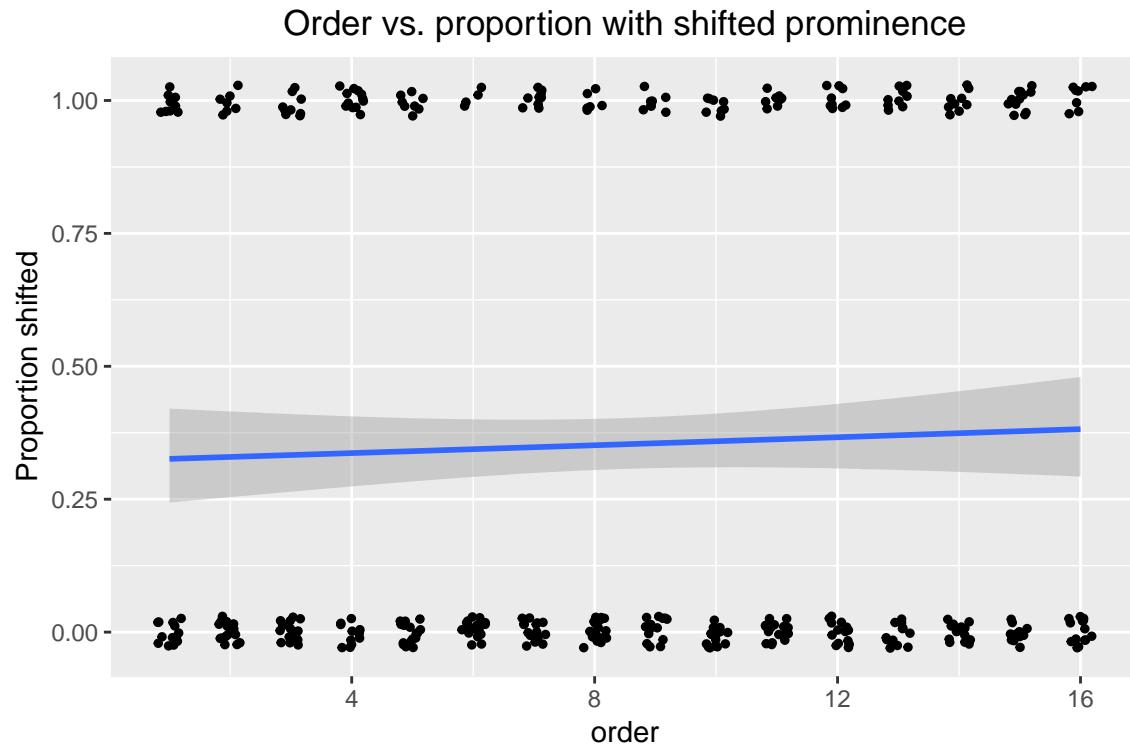
We will carry out a full analysis (except model criticism):

- Exploratory data analysis, motivating which terms to include in a model
- Fit a first model
- Variable selection: adding and dropping terms
- Model evaluation

#### Exploratory data analysis

In this empirical plot of how `order` affects the probability of stress shifting, our actual observations appear as points at  $y = 0$  or  $y = 1$ , while the prediction from a simple logistic regression (on just `order`) appears as the fitted blue line:

```
ggplot(aes(x=order, y=stressshift.num), data=givenness) +
  geom_smooth(method="glm", method.args = list(family = "binomial")) +
  geom_jitter(position=position_jitter(width=0.2,height=0.03),size=1) +
  ylab("Proportion shifted") +
  ggtitle("Order vs. proportion with shifted prominence") +
  theme(plot.title = element_text(hjust = 0.5))
```



Proportion tables showing how the probability of stress shifting depends on each categorical predictor (plots would be better):

- conditionLabel:

```
prop.table(xtabs(~conditionLabel+stressshift, data=givenness), margin=1)
```

```
##           stressshift
## conditionLabel  noshift      shift
##          Contrast 0.91919192 0.08080808
##          Williams 0.35326087 0.64673913
```

- npType:

```
prop.table(xtabs(~npType+stressshift, data=givenness), margin=1)
```

```
##           stressshift
## npType     noshift      shift
##   full     0.6968085 0.3031915
##   pronoun 0.5979381 0.4020619
```

- voice:

```
prop.table(xtabs(~voice+stressshift, data=givenness), margin=1)
```

```
##           stressshift
## voice     noshift      shift
##   active   0.7005348 0.2994652
##   passive 0.5948718 0.4051282
```

The expected directions of effects of the predictors on percent stress-shifted are:

- order: positive
- conditionLabel.williams, npType.pronoun, voice.passive: positive

## Model

We fit a first multiple logistic regression model using just main effects of the four predictors:

```
mod3 <- glm(stressshift ~ npType.pron + clabel.williams + voice.passive + order.std,
             family = "binomial",
             data = givenness)
summary(mod3)

##
## Call:
## glm(formula = stressshift ~ npType.pron + clabel.williams + voice.passive +
##       order.std, family = "binomial", data = givenness)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.8358 -0.5249 -0.3509  0.7644  2.6344
##
## Coefficients:
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.0092    0.1580 -6.389 1.67e-10 ***
## npType.pron   0.5985    0.2746  2.179  0.0293 *
## clabel.williams 3.1848    0.3179 10.018 < 2e-16 ***
## voice.passive  0.8026    0.2803  2.863  0.0042 **
## order.std      0.3044    0.2742  1.110  0.2669
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24  on 381  degrees of freedom
## Residual deviance: 335.90  on 377  degrees of freedom
## AIC: 345.9
##
## Number of Fisher Scoring iterations: 5
```

Note that because the predictors are standardized, we can interpret their values as relative effect sizes: the `conditionLabel` effect is much stronger than the others, then `voice > npType > order`. The direction of all effects is positive, as predicted.

Let's interpret a couple terms from the model, in terms of log-odds and probability:

### 1. Intercept

- $\beta_0 = -1.01$ , corresponds to probability 0.267
- Thus, 27% stress shift is predicted overall (all predictors held at mean values)

### 2. Slope for `conditionLabel.williams`

- Coefficient = 3.18
- Interpretation: the odds of stress shifting are multiplied by **24.0** ( $e^{3.18}$ ) in Williams condition (with other predictors at mean values). This is a huge effect!
- Corresponds to a maximum change of 0.80 in probability (divide-by-4-rule:  $3.18/4=0.80$ ).

### Exercise:

What are the predicted log-odds of stress shifting when:

- `order` = mean value
- `npType` = pronoun
- `conditionLabel` = Williams
- `voice` = active

Hint: assume that the three categorical variables just take on values -0.5 and 0.5.

```
## [1] 0.6818186
```

### Model evaluation

To evaluate the model in terms of how much likelihood improves from a null model, we first set the null model:

```
mod3.0 <- glm(stressshift ~ 1, family = "binomial", data = givenness)
```

Then carry out the likelihood ratio test:

```
anova(mod3, mod3.0, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: stressshift ~ npType.pron + clabel.williams + voice.passive +
##           order.std
## Model 2: stressshift ~ 1
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       377     335.90
## 2       381     496.24 -4    -160.34 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Note that the degrees of freedom of this comparison is  $p - q = 4$ , since there are four predictors in the full model. The difference in deviance is  $\Delta D = 160.34$ . Since we find  $\text{Pr}(>\text{Chisq}) < 2.2e-16$  we may conclude that our four terms make a very significant contribution to the overall model likelihood.

To assess model quality using classification accuracy:

```
## classification accuracy
lrAcc(mod3, 'stressshift')
```

```
## [1] 0.8089005
```

Compared to baseline accuracy:

```
## baseline accuracy
baselineAcc(mod3, 'stressshift')
```

```
## [1] 0.6465969
```

Thus, the four predictors improve classification accuracy by 15%.

### Variable selection

In this exercise we will follow the variable selection guidelines from Gelman & Hill discussed in the previous chapter.

First two steps:

1. Include main effects for predictors expected to affect the response (done)
2. Consider interactions between terms with large effects (we do this now)

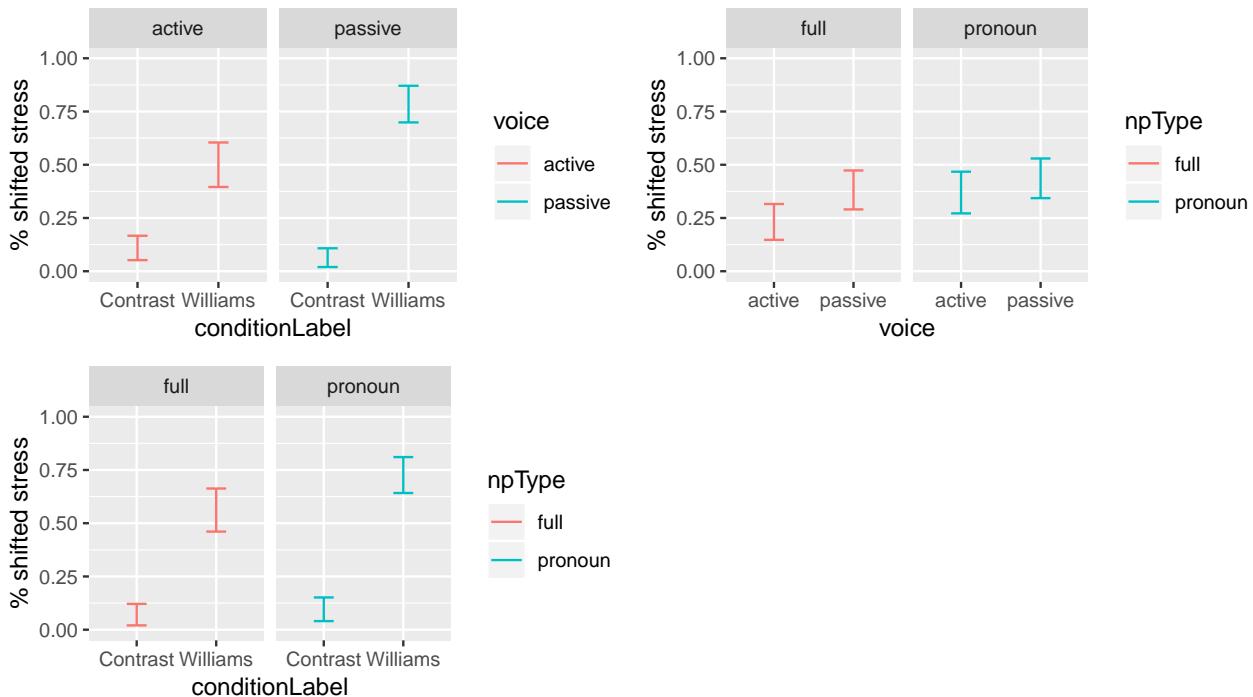
These empirical plots, for each pair of variables with the largest effects, tell us which interactions are most suggestive:

```
day14plt1 <- ggplot(aes(x=conditionLabel, y=stressshift.num), data = givenness) +
  stat_summary(fun.data="mean_cl_boot", geom="errorbar", width=0.25, aes(color=voice)) +
  facet_wrap(~voice) +
  ylab("% shifted stress") +
  ylim(0,1)

day14plt2 <- ggplot(aes(x=voice, y=stressshift.num), data = givenness) +
  stat_summary(fun.data="mean_cl_boot", geom="errorbar", width=0.25, aes(color=npType)) +
  facet_wrap(~npType) +
  ylab("% shifted stress") +
  ylim(0,1)

day14plt3 <- ggplot(aes(x=conditionLabel, y=stressshift.num), data = givenness) +
  stat_summary(fun.data="mean_cl_boot", geom="errorbar", width=0.25, aes(color=npType)) +
  facet_wrap(~npType) +
  ylab("% shifted stress") +
  ylim(0,1)

grid.arrange(day14plt1, day14plt2, day14plt3, ncol = 2)
```



Let's try adding the `voice:conditionLabel` interaction, which looks the largest:

```
mod3 <- glm(stressshift ~ npType.pron + clabel.williams + voice.passive + order.std,
             family = "binomial",
             data = givenness)

mod3.1 <- glm(stressshift ~ npType.pron + clabel.williams * voice.passive + order.std,
```

```

        data = givenness,
        family = "binomial")
anova(mod3, mod3.1, test = 'Chisq')

## Analysis of Deviance Table
##
## Model 1: stressshift ~ npType.pron + clabel.williams + voice.passive +
##          order.std
## Model 2: stressshift ~ npType.pron + clabel.williams * voice.passive +
##          order.std
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      377    335.90
## 2      376    325.81  1   10.085 0.001495 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

This model comparison suggests we should include the `voice:conditionLabel` interaction in the model.

Why, pedantically:

- $\Delta D = 10.08$
- Probability of  $\Delta D$  at least this larger under  $\chi^2(1)$  distribution is 0.0015
- $\Rightarrow$  reject  $H_0$  that  $\beta_{\text{voice:conditionLabel}} = 0$
- $\Rightarrow$  include `voice:conditionLabel` interaction in the model.

We can carry out similar tests for the other two possible two-way interactions, and get:

```

mod3.2 <- glm(stressshift ~ npType.pron + clabel.williams + voice.passive + order.std + npType.pron * c
                data = givenness,
                family = "binomial")
anova(mod3, mod3.2, test="Chisq")

```

```

## Analysis of Deviance Table
##
## Model 1: stressshift ~ npType.pron + clabel.williams + voice.passive +
##          order.std
## Model 2: stressshift ~ npType.pron + clabel.williams + voice.passive +
##          order.std + npType.pron * clabel.williams
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      377    335.90
## 2      376    335.31  1   0.5867  0.4437

mod3.3 <- glm(stressshift ~ npType.pron + clabel.williams+voice.passive + order.std + voice.passive * np
                data = givenness,
                family = "binomial")
anova(mod3, mod3.3, test="Chisq")

```

```

## Analysis of Deviance Table
##
## Model 1: stressshift ~ npType.pron + clabel.williams + voice.passive +
##          order.std
## Model 2: stressshift ~ npType.pron + clabel.williams + voice.passive +
##          order.std + voice.passive * npType.pron
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1      377    335.90
## 2      376    334.99  1   0.9108  0.3399

```

Thus,  $p > 0.3$  for the `npType:conditionLabel` and `voice:npType` interactions, so we don't add these interactions to the model.

The new model is:

```
summary(mod3.1)

##
## Call:
## glm(formula = stressshift ~ npType.pron + clabel.williams * voice.passive +
##       order.std, family = "binomial", data = givenness)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -1.9933 -0.4987 -0.3543  0.6757  2.6080
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.9742    0.1625 -5.993 2.06e-09 ***
## npType.pron   0.6302    0.2810  2.242  0.02493 *
## clabel.williams 3.2054    0.3230  9.923 < 2e-16 ***
## voice.passive  0.3209    0.3239  0.991  0.32172
## order.std     0.3650    0.2876  1.269  0.20437
## clabel.williams:voice.passive  1.9850    0.6370  3.116  0.00183 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24 on 381 degrees of freedom
## Residual deviance: 325.81 on 376 degrees of freedom
## AIC: 337.81
##
## Number of Fisher Scoring iterations: 5
```

Examining the coefficient values, and comparing to our prior expectations—from EDA and from the study design—there are no coefficients to consider removing by Gelman & Hill's guidelines. (Make sure you understand the different reasons we don't drop the `order.std` or `voice.passive` main effects.)

The classification accuracy for this new model is:

```
lrAcc(mod3.1, 'stressshift')
```

```
## [1] 0.8062827
```

Recall that without the `voice:conditionLabel` interaction, we had the classification accuracy:

```
lrAcc(mod3, 'stressshift')
```

```
## [1] 0.8089005
```

So accuracy actually **decreases** slightly when this interaction is added. This can happen, because improving model likelihood and improving classification accuracy are not the same thing.<sup>5</sup>

---

<sup>5</sup>It's worth noting that classification accuracy is very similar for the two models—close enough to be effectively the same, taking finite sample size into account. For a proportion estimated at  $\hat{p} = 0.80$  (like classification accuracy, here) from  $n = 382$  observations, standard error is  $\sqrt{p(1-p)/n} = 0.02$ , so the 95% CIs of classification accuracy of the two models overlap.

## 5.4 Model criticism for logistic regression

Regression assumptions and model diagnostics are just as necessary to assess a fitted model (“model criticism”) as for linear regression, where we covered these topics in depth. The assumptions and diagnostics differ somewhat for logistic regression, but not at a qualitative level. For example:

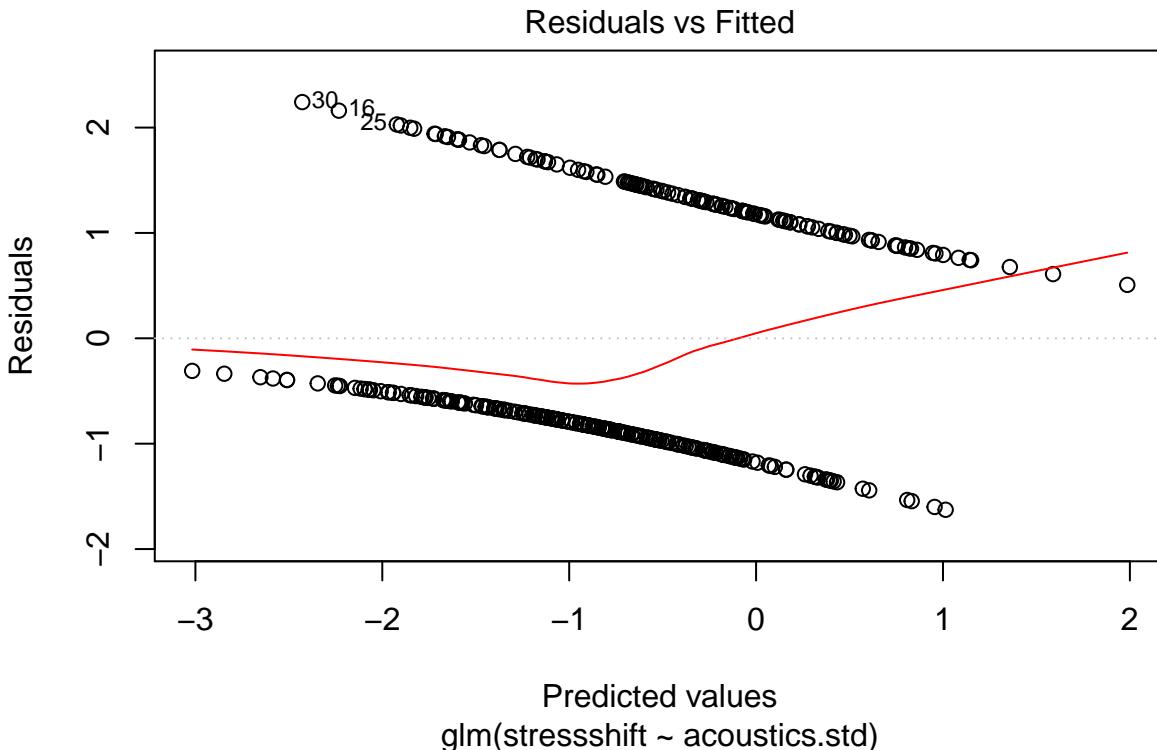
- Linearity: The predictors are assumed to be linearly related to **log-odds of  $Y = 1$**  (rather than to  $Y$  itself, for linear regression).
- Collinearity: The predictors are assumed to not be linearly dependent, and the presence of collinearity causes similar issues for regression coefficients as for linear regression.
- Residual plots and measures of influence: conceptually very similar for logistic regression, except “residuals” needs to be defined differently because there is no error term ( $e_i$ ) for logistic regression.

We will only briefly discuss these topics here, but you should read more (e.g. Chatterjee and Hadi (2012), Baayen (2008), Gelman and Hill (2007)) if you are using logistic regression to analyze data.

### 5.4.1 Residual plots

The **raw** residuals for logistic regression would consist of the model prediction (log-odds, or probabilities) minus the observation (0 or 1) for point  $i$ . R will give you these residuals if you call `resid` on a logistic regression model, but any plot using these residuals is not very informative, because the model’s predictions and the observations are conceptually different (as discussed in Sec. 5.1.4). For example, here is a fitted-residuals plot for a model we fit above, of the kind used to assess homoscedasticity for linear regression models:

```
plot(mod1, which = 1)
```



We get a pattern of two lines in plots using raw residuals, corresponding to 0/1 observations. This plot is not very useful for assessing whether there is constant “variance” in some sense throughout the dataset.

There are several alternative residuals that can be defined for logistic regression, including:

- Deviance residuals (use `glm.diag()` in the `boot` package)
- Binned residuals (use `binned.resids()` in the `arm` package)

Using either kind of residuals, you can evaluate a logistic regression model using similar diagnostic plots as for linear regression: Q-Q plots, fitted-residual plots, and plots of each predictor versus residuals.

### Example: Binned residuals versus expected values

*Binned residuals* are the average (raw) residual over observations within a certain range of expected values (log-odds). That is, you chop the plot immediately above into vertical slices, and average the “Residuals” value within each slice to get a single  $y$ -axis number for each slice.

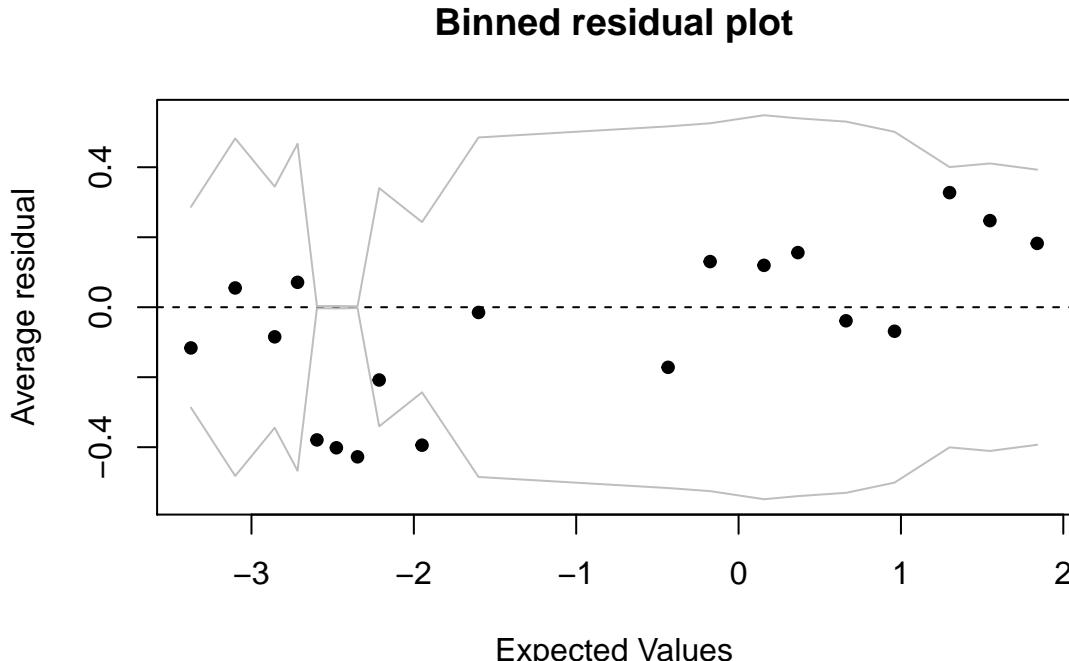
We expect these residuals to be roughly normally distributed with mean 0 and constant variance across different expected values—that is, we expect an expected-value/binned-residual plot to look like the ideal fitted/residual plot for a linear regression.

Consider the final model we ended up with in the example above:

```
ex1.mod <- glm(stressshift ~ npType.pron + conditionLabel.williams + voice.passive + order.std + condit
                 family="binomial",
                 data=givenness)
```

To make a plot of expected values (= fitted values) versus binned residuals:

```
## binned residual plot
binnedplot(predict(ex1.mod), resid(ex1.mod))
```



The diagnostic plot doesn't suggest any major issue (like a “funnel” shape would), but the observations around expected value = -2.5 merit further investigation.

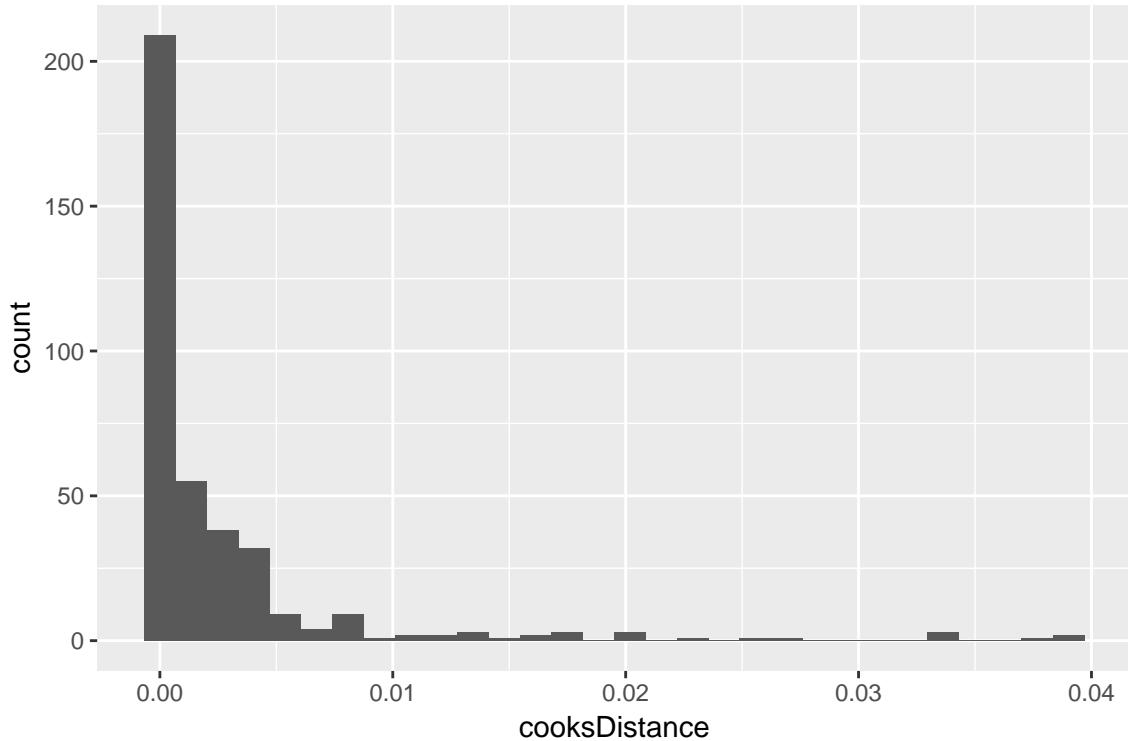
### 5.4.2 Cook's distance

Cook's distance (CD) can be defined for each observation in a logistic regression model, similarly to linear regression. In R, the `glm.diag()` function in the `boot` package computes CD. Similarly to linear regression, the distribution of CD values can be plotted and used to identify observations with extreme values.

Here is an example using the multiple logistic regression model considered above:

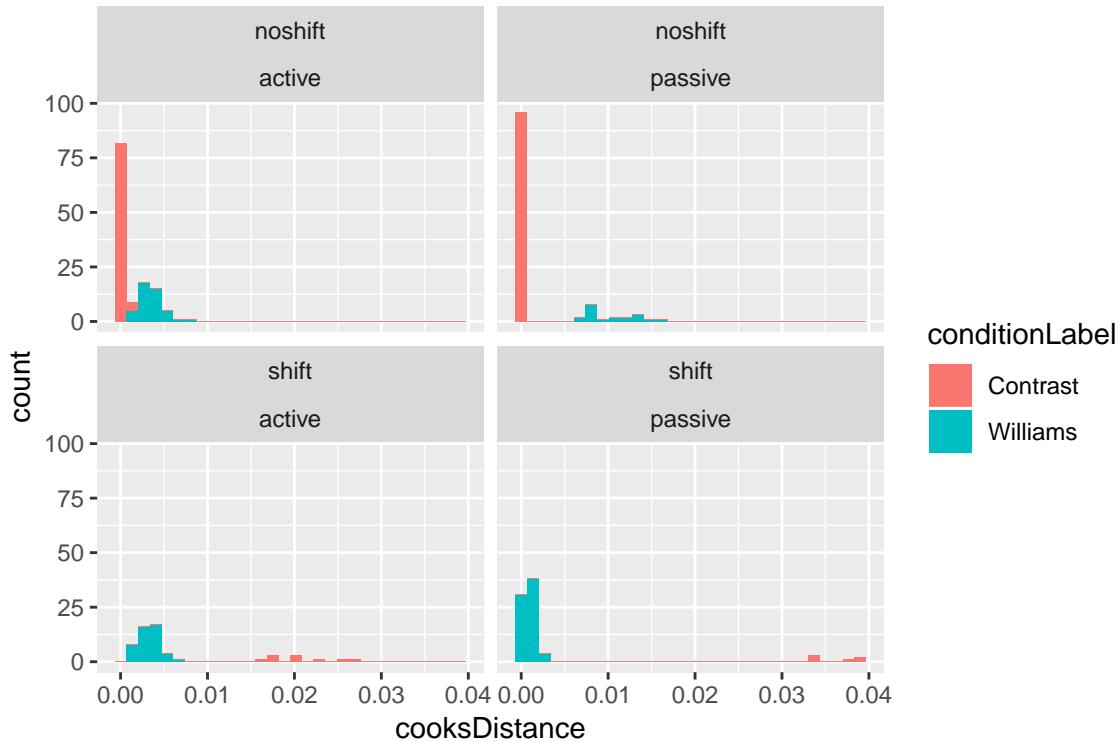
```
## add a column showing Cook's distance values to dataframe
givenness <- mutate(givenness, cooksDistance=glm.diag(ex1.mod)$cook)

ggplot(aes(x=cooksDistance), data=givenness) +
  geom_histogram()
```



It turns out that the points with highest Cook's Distance are passive voice observations in contrast condition where stress was shifted:

```
ggplot(aes(x=cooksDistance), data=givenness) +
  geom_histogram(aes(fill=conditionLabel)) +
  facet_wrap(~stressshift+voice)
```



In fact, all shifted observations in the *contrast* condition are influential, as well as many non-shifted observations in the *williams* condition (top-right panel). This is related to the fact that the Williams effect in this data is very strong: basically, any observation that is unexpected—does not show the Williams effect in the *williams* condition, or does show the effect in the control (*contrast*) condition—is influential, because it goes against the overwhelming majority of the data.

This example illustrates why it may be preferable to use visual inspection of the distribution of CD values to flag “highly influential” points, rather than a rigid cutoff (Chatterjee and Hadi (2012), 4.9.1). One common suggested cutoff is  $4/n$  ( $n$  = number of observations), which would be 0.01 for this dataset. This cutoff would flag as influential *most* data where stress doesn’t shift as expected by the Williams effect. This is correct, in some sense (those points are more influential than others), but not helpful in terms of deciding which observations are “highly influential”—these are the exact observations which let us estimate the size of the Williams effect.

## 5.5 Other readings

Some good sources for general audiences covering the same logistic regression topics in more detail:

- Gelman and Hill (2007) Ch. 4.6, 5.1-5.5 (The rest of Ch. 5, on additional topics, is excellent.)
- Chatterjee and Hadi (2012) 12.1-12.5
- Agresti (2007) Ch. 4-5

For language scientists and psychologists specifically, we are not familiar with in-depth treatments. Some shorter ones:

- Johnson (2008) 5.4-5.7 (5.7 is interesting if you are familiar with GoldVarb from sociolinguistics)
- Jaeger (2008)
- Baayen (2008) 6.3.1

## 5.6 Solutions

**Q:** What are the interpretations of the regression coefficients? \* Intercept ( $\beta_0$ ) \* Slope ( $\beta_1$ )

**A:**  $\beta_0$  is the probability of a stress shift when the NP type is ‘Full NP’.  $\beta_1$  is the change in probability between when the NP type is ‘Full NP’ or ‘Pronoun’.

## 5.7 Appendix: Other Generalized Linear Models

Logistic regression is the only type of generalized linear model we will cover in this book. However, other types of generalized linear model are very useful and cover many other types of data. For example:

- **Multinomial regression:** response = multiple discrete outcomes
  - Example: Verb **regularity** data (outcomes: “zijn”, “zijnheb”, “hebben”)
  - Multinomial regression models are approximately equivalent to “maximum entropy models” widely used in linguistics (especially phonology)
- **Poisson regression:** response = counts (0, 1, 2, 3, ...)
  - Example: Number of words in a lexicon with a given phonotactic shape, or a number of times a word occurs in a corpus (e.g. Baayen (2001))
- **Linear model log link:** response = positive values only, somewhat right-skewed data.
  - Example: Many phonetic parameters (e.g. VOT)
  - Apparently this model is often more appropriate for log-normal data than just transforming the response to  $\log(Y)$  and modeling that; see here.



# Chapter 6

## Practical Regression Topics 1: Multi-level factors, contrast coding, interactions

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(languageR)
library(ggplot2)
library(dplyr)
library(arm)
library(boot)

## loads givennessMcGillLing620.csv from OSF project for Wagner (2012) data
givenness <- read.csv(url("https://osf.io/q9e3a/download"))

givenness <- mutate(givenness,
                     conditionLabel.williams = arm::rescale(conditionLabel),
                     clabel.williams = arm::rescale(conditionLabel),
                     npType.pronoun = arm::rescale(npType),
                     npType.pron = arm::rescale(npType),
                     voice.passive = arm::rescale(voice),
                     order.std = arm::rescale(order),
                     stressshift.num = (as.numeric(stressshift) - 1))
##
## note: "stress shift" generally called "prominence shift" etc. in the text

## loads alternativesMcGillLing620.csv from OSF project for Wagner (2016) data
alternatives <- read.csv(url("https://osf.io/6qctp/download"))

## loads french_medial_vowel_devoicing.txt from OSF project for Torreira & Ernestus (2010) data
devoicing <- read.delim(url("https://osf.io/uncd8/download"))
```

**Note:** Answers to some questions/exercises not listed in text are in Solutions

## 6.1 Multi-level factors: Introduction

Thus far, we have only used factors with two levels as predictors. In general, we want to be able to include factors with  $> 2$  levels in regression models, such as part of speech (noun, verb, adjective, ...), native language ( $L_1 = \text{English}$ , French, Mandarin, other), and so on.

In this chapter we introduce techniques needed to effectively fit and interpret regression models which include multi-level factors:

- **Contrast coding**, which captures how the different levels of the factor affect the response
- **Hypothesis testing** to assess whether a multi-level factor helps predict the response, overall (without reference to particular levels)
- **Interpreting interaction terms** involving multi-level factors, which are present in most models fitted in practice.

## 6.2 Contrast coding

A factor  $X$  with  $k$  levels corresponds to  $k - 1$  categorical predictors, called *contrasts*. There are different ways of mapping  $k$  levels into  $k - 1$  categorical predictors, corresponding to hypotheses that you want to test about how the factor affects the response. These different choices of contrasts (or “coding schemes”) result in different interpretations of:

1. The intercept
2. Regression coefficients for  $X$  in terms of levels of  $X$

That is, **the meaning of the intercept and the meaning of the regression coefficients corresponding to  $X$  depend on the coding scheme**. This is fundamentally why factors with more than two levels are confusing—there is no single way to interpret them.

### 6.2.1 First examples

#### givenness data: two-level factor

Let’s begin with a linear regression example using the **givenness** data, with a single predictor: the two-level factor `conditionLabel`, which is of primary interest. This factor has two levels: *Contrast* and *Williams*. (Note that while discussing contrast coding, we will use *italics* to refer to levels of a factor, and `teletype` to refer to actual names of factors or other predictors.)

```
mod <- lm(acoustics ~ conditionLabel, data=givenness)
summary(mod)

##
## Call:
## lm(formula = acoustics ~ conditionLabel, data = givenness)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.31843 -0.56264  0.01977  0.53651  2.50851
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.87510   0.05708 -15.331 < 2e-16 ***
##
```

```
## conditionLabelWilliams  0.31774    0.08224   3.863 0.000131 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8032 on 380 degrees of freedom
## Multiple R-squared:  0.03779,   Adjusted R-squared:  0.03526
## F-statistic: 14.93 on 1 and 380 DF,  p-value: 0.0001315
```

The `conditionLabelWilliams` row of the regression table gives the coefficient value for the single *contrast* corresponding to the two-level factor, which has two values: 0 and 1.

To see this, examine the *contrast matrix*:

```
contrasts(givenness$conditionLabel)
```

```
##           Williams
## Contrast      0
## Williams     1
```

which says there is a single contrast, which R (confusingly) calls `Williams`—the name of the second level. This numeric variable takes on the value 1 when the level of `condition` is *Williams* and 0 when the level of `condition` is *Contrast*.

In the regression above:

- The interpretation of the intercept is: value of `acoustics` when `conditionLabel = Contrast`
- The interpretation of the regression coefficient `conditionLabelWilliams` is: difference in `acoustics` between `conditionLabel = Contrast` and `conditionLabel = Williams`.

These are the interpretations of (1) and (2) resulting from this choice of contrast, which is called “dummy coding” (see Sec. 6.2.3.1 for more detail). R uses dummy coding by default for factors.

In previous analyses of the `givenness` data, we have often used a “standardized” version of `conditionLabel`, called `clabel.williams`:

```
mod <- lm(acoustics ~ clabel.williams, data=givenness)
summary(mod)

##
## Call:
## lm(formula = acoustics ~ clabel.williams, data = givenness)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -2.31843 -0.56264  0.01977  0.53651  2.50851
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.72205   0.04109 -17.570 < 2e-16 ***
## clabel.williams 0.31774   0.08224   3.863 0.000131 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8032 on 380 degrees of freedom
## Multiple R-squared:  0.03779,   Adjusted R-squared:  0.03526
## F-statistic: 14.93 on 1 and 380 DF,  p-value: 0.0001315
```

a numeric variable which takes on values of roughly -0.5 and 0.5. This is equivalent to using a different contrast coding scheme, where the two values of the `conditionLabel` contrast are (roughly) -0.5 and 0.5.

(To see this, we can actually set the contrast values to -0.5 and 0.5 for the `conditionLabel` factor, and carry out a regression using `conditionLabel` as a factor:

```
## sets contrast levels to -0.5 and 0.5
contrasts(givenness$conditionLabel) <- contr.sum(2)/2

## carry out the regression and see results
mod <- lm(acoustics ~ conditionLabel, data=givenness)
summary(mod)

##
## Call:
## lm(formula = acoustics ~ conditionLabel, data = givenness)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -2.31843 -0.56264  0.01977  0.53651  2.50851 
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)    
## (Intercept) -0.71623   0.04112 -17.417 < 2e-16 ***
## conditionLabel1 -0.31774   0.08224 -3.863 0.000131 *** 
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
##
## Residual standard error: 0.8032 on 380 degrees of freedom
## Multiple R-squared:  0.03779, Adjusted R-squared:  0.03526 
## F-statistic: 14.93 on 1 and 380 DF, p-value: 0.0001315
```

Note the results are identical to using `clabel.williams` as the predictor.)

The interpretations of (1) and (2) are now:

- Intercept: **Average value of acoustics** (across both levels of `conditionLabel`)
- Regression coefficient for `conditionLabel`: difference in `acoustics` between `conditionLabel = Contrast` and `Williams`.

That is, the interpretation of the intercept has changed from dummy coding, but the interpretation of the contrast's regression coefficient has not changed.

### Example: Three-level factor with `devoicing` dataset

As another example, consider a simple linear regression of the effect of vowel (`v`: three levels = *i*, *u*, *y*) on syllable duration (`syldur`) in the `devoicing` dataset:

```
mod <- lm(syldur ~ v, data=devoicing)
summary(mod)
```

```
##
## Call:
## lm(formula = syldur ~ v, data = devoicing)
##
## Residuals:
##       Min     1Q   Median     3Q    Max 
## -108.006 -26.925   1.075  23.055 115.994 
##
```

```

## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 123.006    2.050 59.999 <2e-16 ***
## vu          15.963    6.686  2.388  0.0173 *
## vy         -4.082    3.304 -1.235  0.2173
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 36.56 on 547 degrees of freedom
## Multiple R-squared:  0.01559,   Adjusted R-squared:  0.01199
## F-statistic: 4.331 on 2 and 547 DF,  p-value: 0.01361

```

The interpretations of (1) and (2) are:

- Intercept: value of `syldur` when `vowel = i`
- Regression coefficients for `syldur`:
  - Contrast 1: difference in `syldur` between `vowel = u` and `i`
  - Contrast 2: difference in `syldur` between `vowel = u` and `y`

The `v` variable has been turned into two numeric variables:

- Contrast 1: 1 when `vowel=u`, 0 otherwise
- Contrast 2: 1 when `vowel=y`, 0 otherwise

as can be seen in the two columns of the contrast matrix:

```
contrasts(devoicing$v)
```

```

##   u y
## i 0 0
## u 1 0
## y 0 1

```

## 6.2.2 Basic interpretation of contrasts

These examples illustrate the most crucial aspect of contrasts: **contrasts correspond to different hypotheses you're testing about how the response differs between groups.**<sup>1</sup>

For example, based on our research questions, we may be interested in:

- “What is the difference in  $Y$  between level 1 and level 3 of the factor?”
- “What is the difference in  $Y$  between level 3 and the previous levels (1-2) of the factor?”

Different contrast coding schemes are used to test different hypotheses. Only  $k - 1$  such hypotheses can be included at once in a model, for a factor with  $k$  levels, in order for the predictors to not be linearly dependent (= one can be perfectly predicted from the others). As a result, contrasts do **not** correspond to “the value of  $Y$  at level  $i$  of the factor” (e.g. “`syldur` when `v` is  $i$ ”)—they always correspond to **differences** between levels. This is a frequent point of confusion.

Contrast coding is a powerful and useful tool. Unfortunately, interpreting contrasts is confusing, the terminology for talking about contrasts is only partially standardized, and there are few good and comprehensive readings on contrast coding—the only one we are aware of is Schad et al. (2018), who go into both mathematical and practical detail. Another useful exposition by UCLA’s statistical consulting service, which focuses

---

<sup>1</sup>Contrasts also correspond to different interpretations of the intercept, but this is often less important in practice.

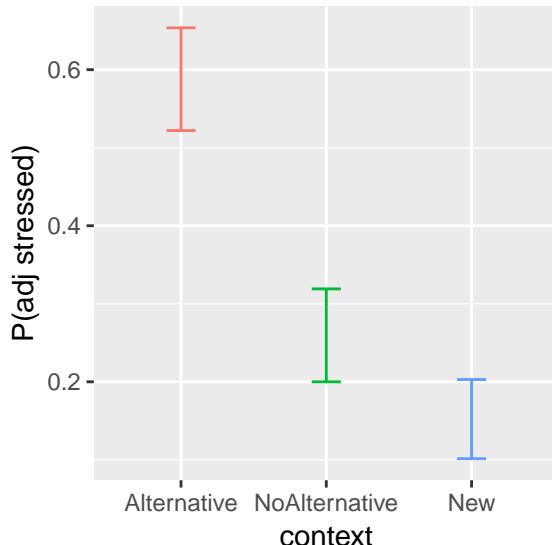


Figure 6.1: Effect of ‘context’ on proportion of tokens with shifted prominence, for the ‘alternatives’ data.

largely on practical implementation using R is here.

We will cover a few common contrast coding schemes, through examples.

### Example

For this example, we will be using the `alternatives` dataset, described in detail here. Each row corresponds to a single sentence read by a participant in a speech production experiment, and the important columns are:

- Predictor: `context`, with levels *New*, *NoAlternative*, *Alternative*
- Response: `shifted`, a binary (0/1) factor indicating whether prominence was shifted in the sentence read by the speaker

We first do some data preparation:

```
## reorder factor levels to make conceptual sense: "NoAlternative" is conceptually between "Alternative"
alternatives <- mutate(alternatives, context=factor(context, levels=c("Alternative", "NoAlternative", "New")))

## remove rows where response is NA (this is just due to an issue with this dataset)
alternatives <- filter(alternatives, !is.na(prominence))

## add a 0/1 variable where 1 = adj prominence (shifted), 0 = N prominence
alternatives <- alternatives %>% mutate(shifted = -1*as.numeric(factor(prominence))+2)
```

The basic results are:

```
## plot of the basic pattern
ggplot(aes(x=context, y = shifted), data=alternatives) +
  stat_summary(fun.data="mean_cl_boot", geom='errorbar', width=0.2, aes(color=context)) +
  ylab("P(adj stressed)") +
  theme(legend.position="none")
```

The probability of shifting prominence decreases from *Alternative* to *NoAlternative* to *New*.

For exemplifying different contrast coding schemes, we will need the condition means: the **log-odds** of shifting prominence in the empirical data, in each context:

- `context = Alternative`: 0.345
- `context = NoAlternative`: -1.036
- `context = New`: -1.736

**Questions:**

- What are the equivalent probabilities?

(This is just practice in going from log-odds to probabilities.)

```
## [1] 0.585
## [1] 0.262
## [1] 0.15
```

For practice with data summarization: this hidden table shows how to calculate empirical probabilities and log-odds for each condition, using `dplyr` functions:

```
## empirical p and log-odds of shifting prominence:
conditionMeans <- alternatives %>% group_by(context) %>% summarise(p=mean(shifted), logOdds=log(p/(1-p)))
conditionMeans

## # A tibble: 3 x 3
##   context      p  logOdds
##   <fct>     <dbl>    <dbl>
## 1 Alternative 0.585    0.345
## 2 NoAlternative 0.262   -1.04
## 3 New         0.150   -1.74
```

### 6.2.3 Contrast coding schemes

#### 6.2.3.1 Dummy coding

We first consider *dummy coding*, also known as *treatment coding*.

In dummy coding:

- The **intercept** corresponds to level 1
- **Contrast 1** corresponds to level 2 minus level 1
- **Contrast  $k$**  corresponds to level  $k + 1$  minus level 1.

For example, for a four-level factor coded using dummy contrasts, the first/second/third contrast correspond to the difference between level 4/3/2 and level 1.

**Questions:**

- What does the hypothesis test  $H_0 : \beta_1 = 0$  correspond to asking? (What levels are hypothesized to have the same value of  $Y$ ?)

The *contrast matrix* for dummy coding for a three-level factor is:

	Contrast1	Contrast2
<i>Level 1</i>	0	0
<i>Level 2</i>	1	0

	Contrast1	Contrast2
Level 3	0	1

The contrast matrix shows the mapping from a categorical predictor with  $k$  levels, to a set of  $k - 1$  numeric variables (the contrasts). In the example above, observations from *Level 1* have values 0 for both the first and second contrast.

The contrast matrix looks similar for factors with more levels , such as  $k = 4$  or  $k = 5$ :

```
contr.treatment(4)
```

```
## 2 3 4
## 1 0 0 0
## 2 1 0 0
## 3 0 1 0
## 4 0 0 1
```

```
contr.treatment(5)
```

```
## 2 3 4 5
## 1 0 0 0 0
## 2 1 0 0 0
## 3 0 1 0 0
## 4 0 0 1 0
## 5 0 0 0 1
```

(Rows = factor levels; columns = contrasts)

By default, R assumes that factor levels go in alphabetical order. In our data, we have re-leveled `context` so that *Alternative* < *NoAlternative* < *New*. That is, *Alternative* is the “base level” (level 1). A logistic regression of `shifted` on `context` using dummy coding gives:

```
summary(glm(shifted ~ context, data=alternatives, family='binomial'))
```

```
##
## Call:
## glm(formula = shifted ~ context, family = "binomial", data = alternatives)
##
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -1.3269   -0.7793   -0.5696    1.0349    1.9487
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) 0.3448     0.1418   2.432   0.015 *
## contextNoAlternative -1.3809     0.2115  -6.529 6.61e-11 ***
## contextNew   -2.0813     0.2409  -8.639  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 789.96 on 621 degrees of freedom
## Residual deviance: 694.53 on 619 degrees of freedom
## AIC: 700.53
##
```

```
## Number of Fisher Scoring iterations: 4
```

The coefficient for contrast is the `contextNoAlternative` row. The coefficient value is the estimated difference, between `context = NoAlternative` and `context = Alternative`, in the log-odds of stress shifting.

### Questions:

- What is the interpretation of the second contrast's coefficient (`contextNew`)?

We can compare these coefficient values to the corresponding differences in empirical means (log-odds, given for each level in a previous example):

- $\hat{\beta}_1 = -1.38$ 
  - *NoAlternative* mean - *Alternative* mean = -1.381
- $\hat{\beta}_2 = -2.08$ 
  - *New* mean - *Alternative* mean = -2.081
- $\hat{\beta}_0 = 0.344$ 
  - *Alternative* mean = 0.345

From the *p*-values for these coefficients, we can conclude:

- *NoAlternative* and *Alternative* differ significantly
- *New* and *NoAlternative* also differ significantly
- *Alternative* is significantly different from 0.

Dummy coding is what R uses by default for factors, perhaps because this coding scheme is most traditional or easiest to understand.

However, dummy coding has important drawbacks: each contrast doesn't sum to zero (for balanced data), which means there is collinearity between contrasts—for no good reason. Intuitively, we would like  $k - 1$  independent contrasts for a factor with  $k$  levels, but dummy contrasts aren't independent—if you know that contrast 1 = 1 (for some observation), you already know that the level isn't 1 (the base level), which gives you information about contrast 2.

A more practical drawback is that dummy contrasts are not “centered”: the intercept for dummy coding is level 1, but this is often not of direct interest. It often makes more sense for the intercept to be interpretable as some sort of grand mean—the “average value” across the dataset. (In *simple coding*, the intercept is the grand mean (average of levels 1, 2, ...), but the contrasts have the same interpretation as in dummy coding.)

In general, it is not recommended to use dummy coding without good reason, e.g. you are actually interested in the value of level 1 and differences between level  $k$  and level 1.

#### 6.2.3.2 Contrast matrix → interpretation

In general, you can't read off from the contrast matrix the **interpretations** of the intercept and contrasts in an actual regression model. For example, it is not obvious from the two columns of the contrast matrix shown for dummy coding above ((0, 1, 0) and (0, 0, 1)) that the first contrasts should be interpreted as “difference between level 2 and level 1”.

Besides just memorizing the interpretation of each coding scheme, you can solve for the interpretations by taking the “generalized inverse” of the contrast matrix, using the `ginv` function in the MASS package.<sup>2</sup>

This example (for dummy coding a three-level factor) shows how to obtain a matrix from which we can read off intercept and contrast interpretations from each row:

---

<sup>2</sup>For the mathematical details of contrast coding, including why taking the generalized inverse gives contrast interpretations, see Schad et al. (2018) or Sec. 6.2 of Venables and Ripley (2002).

```
ginv(contr.treatment(3))
```

```
##      [,1] [,2] [,3]
## [1,]    0    1    0
## [2,]    0    0    1
```

The three rows of this matrix correspond to the interpretation of the intercept and contrasts:

1. Row 1 = Intercept:

- *Level 1* (vector  $(1, 0, 0)$ ) means “*Level 1* minus  $0 \times$  *Level 2* minus  $0 \times$  *Level 3*”)

2. Row 2 = Contrast 1:

- *Level 2 - Level 1* (vector  $(-1, 1, 0)$ ) means “*Level 2* minus *Level 1*”)

3. Row 3 = Contrast 2:

- *Level 3 - Level 1*

You can get a similar matrix for any contrast matrix  $X$  using `ginv(X)`.

### 6.2.3.3 Sum coding

Next, we consider *sum coding*, also known as *deviation coding*. For a factor with  $k$  levels:

- The interpretation of the **intercept** is: mean of *level 1*, ..., *level k* (“grand mean”)
- Contrast 1: *level 1* – grand mean
- Contrast 2: *level 2* – grand mean
- Contrast  $k$ : *level k* – grand mean

For example, for a factor with three levels, the first contrasts corresponds to “difference between *level1* and the mean of *levels 1/2/3*”.

#### Questions:

- What does the hypothesis test  $H_0 : \beta_1 = 0$  correspond to? (What is equal to what?)

The contrast matrix for sum coding with  $k = 3$  levels is:

	Contrast1	Contrast2
<i>Level 1 (Alternative)</i>	1	0
<i>Level 2 (NoAlternative)</i>	0	1
<i>Level 3 (New)</i>	-1	-1

In R, this is `contr.sum(3)`. You can see how the contrast matrix looks for a factor with  $k$  levels by extrapolating from  $k = 4$  and  $k = 5$ :

```
contr.sum(4)
```

```
##      [,1] [,2] [,3]
## 1     1    0    0
## 2     0    1    0
## 3     0    0    1
## 4    -1   -1   -1
```

```
contr.sum(5)
```

```
##   [,1] [,2] [,3] [,4]
## 1    1    0    0    0
## 2    0    1    0    0
## 3    0    0    1    0
## 4    0    0    0    1
## 5   -1   -1   -1   -1
```

Sum contrasts have the advantage that each contrast is **centered** (for balanced data): the contrast sums to zero across the dataset. This tends to reduce collinearity, and allows for easier interpretation of main effects in the presence of interactions (as “omnibus” effects). It does not eliminate collinearity, because knowing the value of one contrast still tells you something about the value of the others. (Why?)

An aside: as above, the interpretations of the intercept and contrasts are not obvious from the contrast matrix, but we can get these interpretations by taking the generalized inverse:

```
ginv(contr.sum(3))
```

```
##          [,1]      [,2]      [,3]
## [1,] 0.6666667 -0.3333333 -0.3333333
## [2,] -0.3333333  0.6666667 -0.3333333
```

where we see:

- Row 1: intercept
  - $1/3 \times$  each level = **grand mean**
- Row 2:  $(2/3, -1/3, -1/3) = (1, 0, 0) - (1/3, 1/3, 1/3)$ 
  - Difference between *level 1* and grand mean
- Row 3:  $(-1/3, 2/3, 1/3) = (0, 1, 0) - (1/3, 1/3, 1/3)$ 
  - Difference between *level 2* and grand mean

### Example

Let’s refit the simple logistic regression above, but now coding `context` using sum contrasts:

```
## sum-coded version of context
alternatives$context.sum <- alternatives$context
contrasts(alternatives$context.sum) <- contr.sum(3)
summary(glm(shifted ~ context.sum, data=alternatives, family="binomial"))

##
## Call:
## glm(formula = shifted ~ context.sum, family = "binomial", data = alternatives)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -1.3269  -0.7793  -0.5696   1.0349   1.9487
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.80925   0.09584  -8.444   <2e-16 ***
## context.sum1  1.15409   0.12604   9.157   <2e-16 ***
## context.sum2 -0.22684   0.13190  -1.720   0.0855 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 789.96 on 621 degrees of freedom
## Residual deviance: 694.53 on 619 degrees of freedom
## AIC: 700.53
## 
## Number of Fisher Scoring iterations: 4

```

Remembering that the ordering of the levels of `context` is *Alternative*, *NoAlternative*, *New*:

#### Questions:

- What are the interpretations of:
- The intercept?
- `context.sum1`?
- `context.sum2`?
- The coefficient for the second contrast isn't significant. What corresponding difference is not significant (roughly, "is very small"), in the empirical data (Figure 6.1)?

**Example:** interpretation of contrast 1:

- Level 1 (= *Alternative*) condition mean = 0.344
- Average of condition means:  $(0.344 + -1.036 + -1.736)/3 = -0.809$
- Difference between these:  $0.344 - -0.809 = 1.15$
- $\hat{\beta}_1 = 1.15$

#### 6.2.3.4 Helmert coding

We next consider *Helmert coding*, where each contrast corresponds to the difference between *level k* and the previous levels.<sup>3</sup>

- The interpretation of the **intercept** is: mean of *level 1*, ..., *level k* ("grand mean", as for sum coding)
- Contrast 1:  $\frac{1}{2} \times (\text{level 2} - \text{level 1})$
- Contrast 2:  $\frac{1}{3} \times (\text{level 3} - (\text{mean of level 1 and level 2}))$
- etc.

The contrast matrix for Helmert contrasts with  $k = 3$  levels is:

	Contrast1	Contrast2
<i>Level 1 (Alternative)</i>	-1	-1
<i>Level 2 (NoAlternative)</i>	1	-1
<i>Level 3 (New)</i>	0	2

For  $k = 4$  and  $k = 5$ :

```
contr.helmert(4)
```

```
## [,1] [,2] [,3]
```

---

<sup>3</sup>Confusingly, this coding scheme is also sometimes also called *reverse Helmert* due to disagreement on which direction is "reverse" (comparing to previous or later levels?).

```
## 1   -1   -1   -1
## 2    1   -1   -1
## 3    0    2   -1
## 4    0    0    3
contr.helmert(5)

## [,1] [,2] [,3] [,4]
## 1   -1   -1   -1   -1
## 2    1   -1   -1   -1
## 3    0    2   -1   -1
## 4    0    0    3   -1
## 5    0    0    0    4
```

Helmert contrasts are *orthogonal*: knowing the value of one contrast doesn't tell you anything about the value of another contrast. For example, knowing the difference between *level 1* and *level 2* (contrast 1) tells you nothing about the difference between *level 3* and previous levels (contrast 2). This property is desirable: orthogonality minimizes collinearity between contrasts.

Again, to derive the interpretation of Helmert contrasts:

```
ginv(contr.helmert(3))
```

```
##          [,1]      [,2]      [,3]
## [1,] -0.5000000  0.5000000  0.0000000
## [2,] -0.1666667 -0.1666667  0.3333333
```

where we see:

- Row 1: intercept
  - $1/3 \times$  each level = **grand mean**
- Row 2:  $\frac{1}{2} \times [(0, 1, 0) - (1, 0, 0)]$ 
  - $\frac{1}{2}$  difference between *level 2* and *level 1*
- Row 3:  $\frac{1}{3} \times [(0, 0, 1) - (1, 1, 0) \times \frac{1}{2}]$ 
  - $\frac{1}{3}$  difference between *level 3* and mean of the previous levels

### Example

Let's refit the simple logistic regression above, but now coding `context` using Helmert contrasts.

```
## Helmert-coded version of context
alternatives$context.helm <- alternatives$context
contrasts(alternatives$context.helm) <- contr.helmert(3)

summary(glm(shifted ~ context.helm, data=alternatives, family="binomial"))

##
## Call:
## glm(formula = shifted ~ context.helm, family = "binomial", data = alternatives)
##
## Deviance Residuals:
##       Min        1Q     Median        3Q       Max
## -1.3269  -0.7793  -0.5696   1.0349   1.9487
##
## Coefficients:
```

```

##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.80925   0.09584 -8.444 < 2e-16 ***
## context.helm1 -0.69047   0.10575 -6.529 6.61e-11 ***
## context.helm2 -0.46362   0.07388 -6.275 3.49e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 789.96 on 621 degrees of freedom
## Residual deviance: 694.53 on 619 degrees of freedom
## AIC: 700.53
##
## Number of Fisher Scoring iterations: 4

```

#### Questions:

- What are the interpretations of:
- The intercept? (same as for sum coding)
- `context.helm1`?
- `context.helm2`?

(What two differences between levels are significant?)

Having all Helmert contrast coefficients significant, and in the same direction, is **consistent with** (but does not prove) that *level 1 < level 2 < level 3*. To actually show that each level is “less” than the subsequent level (= the response is significantly lower), you need to either apply post-hoc tests or use “successive difference” contrasts (see below).

#### 6.2.3.5 Other coding schemes

Many other contrast coding schemes exist. A couple useful ones:

- Successive-difference coding ( $1 < 2, 2 < 3, \dots$ ) (`contr.sdif` in MASS library)
- User-defined contrast (any  $k - 1$  functions of the  $k$  levels)

The UCLA page mentioned above explains different contrast schemes well, using R examples.

#### 6.2.3.6 Practical advice

Two pieces of practical advice:

1. **Don't use dummy-coded contrasts** (R's default), except in special circumstances (e.g. you are actually interested in level 1's value).
  - The fact that the contrasts are neither centered nor orthogonal can easily lead to unnecessary interpretability, collinearity, and model convergence issues.
  - Because dummy coding is the default in R, not using it requires either changing the default contrast coding R uses, or changing the contrasts manually for every factor.
2. **Use a “centered” contrast coding scheme**, such as sum, Helmert, or simple contrast coding.
  - Sum-coded contrasts are not orthogonal (bad), but easier to interpret (good). Helmert-coded contrasts are orthogonal (good), but harder to interpret (bad).

If you have a multi-level factor in your data you will have to use contrast coding. But contrasts can be difficult to think about, including deciding which coding scheme to use. Using and interpreting contrasts gets easier with practice, and it's important to keep the big picture in mind: contrast coding schemes are just fancy ways of **testing hypotheses about the relationship between a factor and the response**. As such, which coding scheme you use is just a matter of convenience, depending on which hypotheses you want to test, and practical considerations (e.g. collinearity). There is no “wrong” contrast coding scheme, just better and worse ones for a particular case.

Importantly, different contrast coding schemes give the **same regression model**, in the sense that it makes the same predictions ( $Y$ ) given predictor values ( $X$ ). However, the values and significances of regression coefficients change when different coding schemes are used—as we saw above, fitting the same model using three different coding schemes.

### 6.3 Assessing a multi-level factor's contribution

For continuous predictors and factors with two levels, the question “does this predictor significantly affect the response?” is answered by the significance of the regression coefficient. For a factor with  $3+$  levels, we would like to ask the same question (e.g. “does `context` affect the likelihood of shifting prominence?”), but no one coefficient's  $p$ -value gives the answer, because the factor's effect on the response is **jointly** captured by all its contrasts.

Instead, we assess whether a factor with  $3+$  levels affects the response by viewing this as a special case of **model comparison** (discussed here). For a factor with  $k$  levels, the  $k - 1$  contrasts correspond to  $k - 1$  predictors (that is, numeric variables  $X_i$ ). Suppose there are  $p - 1$  additional predictors in the regression model. We can then compare:

- Full model  $M_1$ : factor + other predictors (degrees of freedom  $df = n + p + k - 2$ )
- Reduced model  $M_0$ : other predictors, only ( $df = n + p - 1$ )

using the methods previously discussed: an  $F$  test for linear regression (seen in this example), or a likelihood ratio test for logistic regression. In either case, the difference in degrees of freedom between the two models is just the number of contrasts ( $k - 1$ ).

#### Example

Does `context` significantly affect whether prominence is shifted, in `alternatives`?

In this case,  $p = 1$  (intercept-only model) and  $k = 3$ . Thus, the change in  $df$  between the models is 2. To carry out the model comparison:

```
## likelihood ratio test to check whether context significantly affects prominence shift
m1 <- glm(shifted ~ context, data=alternatives, family="binomial")
m0 <- glm(shifted ~ 1, data=alternatives, family="binomial")
anova(m0,m1, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: shifted ~ 1
## Model 2: shifted ~ context
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       621    789.96
## 2       619    694.53  2    95.432 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Recall that `test="Chisq"` specifies a likelihood ratio test ( $\chi^2$ ) between models. Under `Pr(>Chi)` we see that `context` does indeed significantly affect prominence shift (`Pr(>Chisq) < 2.2e-16`).

Importantly, the result of model comparison does not depend on the coding scheme used! Thus, it is also not affected by collinearity between contrasts (which is a consideration in which contrast scheme to choose).

For example, let us examine how `shifted` depends on `context`:

```
## the regression coefficients are very different for the two contrasts for context
## under different coding schemes...
mod2 <- glm(shifted ~ context, data=alternatives, family="binomial")
mod2.helm <- glm(shifted ~ context.helm, data=alternatives, family="binomial")
mod2.sum <- glm(shifted ~ context.sum, data=alternatives, family="binomial")
mod0 <- glm(shifted ~ 1, data=alternatives, family="binomial")
```

the full models with different coding schemes for `context` give the same result for the likelihood ratio test, which asks whether context significantly contributes:

```
anova(mod0, mod2, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: shifted ~ 1
## Model 2: shifted ~ context
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       621    789.96
## 2       619    694.53  2    95.432 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
anova(mod0, mod2.helm, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: shifted ~ 1
## Model 2: shifted ~ context.helm
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       621    789.96
## 2       619    694.53  2    95.432 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
anova(mod0, mod2.sum, test="Chisq")
```

```
## Analysis of Deviance Table
##
## Model 1: shifted ~ 1
## Model 2: shifted ~ context.sum
##   Resid. Df Resid. Dev Df Deviance Pr(>Chi)
## 1       621    789.96
## 2       619    694.53  2    95.432 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

## 6.4 Practice with interactions

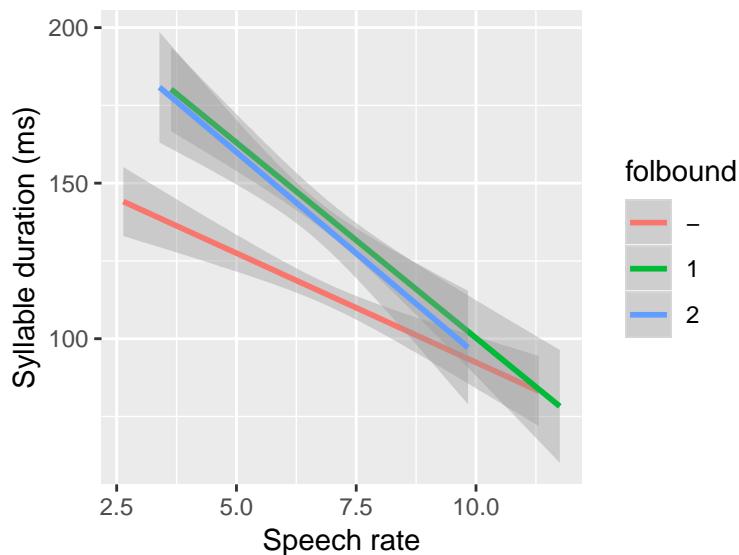
Interpreting interaction terms in a regression model gets trickier for factors with 3+ levels, and for interactions between 3+ predictors (“three-way” etc. interactions). It is useful to interpret model coefficients together with empirical plots. We give a couple examples here.

### Example 1: Two-way interaction with multi-level factor

**Data:** vowel devoicing data (described in detail here)

Our question is: does the effect of `speechrate` on `syldur` depend on following prosodic boundary type (`folbound`)?

```
ggplot(aes(x=speechrate,y=syldur), data=devoicing) +
  geom_smooth(method='lm', aes(color=folbound)) +
  xlab("Speech rate") +
  ylab("Syllable duration (ms)")
```



Note that `folbound` has the levels `-` (no boundary), `1` (weak boundary), and `2` (strong boundary).

### Questions:

- Choose a coding scheme (sum? Helmert?) so that one contrast = “- versus 2/ 3”
  - Hint: you’ll first need to change factor levels so that 3 < 2 < -
- Carry out a linear regression of `speechrate` on `syldur\*folbound`

Answer (in comments), and implementation:

```
## we should choose *Helmert contrasts*:
## contrast 1: difference between 3 and 2
## contrast 2: difference between - and 2/3

## change factor levels for folbound so highest>lowest
devoicing <- mutate(devoicing, folbound=factor(folbound, levels=c("2", "1", "-")))

## Choose a contrast scheme for folbound so that one contrast is 2/1 vs - (the distinction seen in the p-values)
contrasts(devoicing$folbound) <- contr.helmert(3)
```

Before fitting the model, let's figure out what results we "should" get, by determining the interpretation of each coefficient of the `speechrat*folbound` interaction. In terms of the above plot:

- Contrast 1: difference in slope of the line between 2 and 1
- Contrast 2: difference in slope of the line between - and 2/1

### Questions:

- What directions do we expect?

The fitted model is:

```
summary(lm(syldur ~ speechrat*folbound, data=devoicing))

##
## Call:
## lm(formula = syldur ~ speechrat * folbound, data = devoicing)
##
## Residuals:
##      Min       1Q   Median       3Q      Max 
## -109.769  -21.641    0.858   21.919  111.630 
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)    
## (Intercept) 204.4614   8.6510  23.635 < 2e-16 ***
## speechrat   -10.8602   1.2524 -8.671 < 2e-16 ***
## folbound1    0.4630   12.2837  0.038  0.96995  
## folbound2   -20.9193   4.9541 -4.223 2.83e-05 ***
## speechrat:folbound1  0.2153   1.7835  0.121  0.90398  
## speechrat:folbound2   1.9173   0.7130  2.689  0.00739 ** 
## ---      
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 32.47 on 544 degrees of freedom
## Multiple R-squared:  0.2276, Adjusted R-squared:  0.2205 
## F-statistic: 32.06 on 5 and 544 DF,  p-value: < 2.2e-16
```

Focusing just on the last two rows:

- `speechrat:folbound1`: contrast 1 is not significant, reflecting the very small difference between the slopes of `speechrat` for 1 and 2.
- `speechrat:folbound2`: contrast 2 is positive and highly significant, reflecting the less-negative slope of `speechrat` for - compared to 1 and 2.

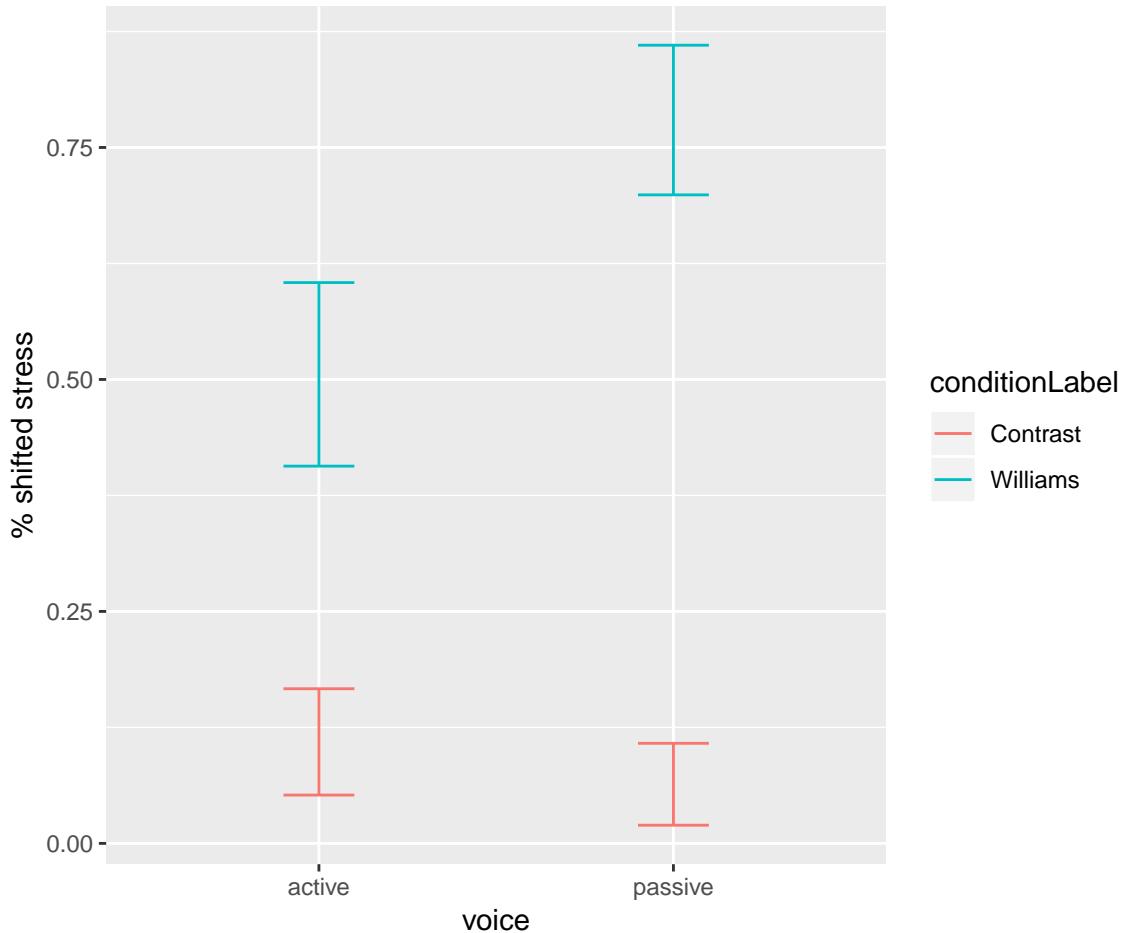
We can thus say that the effect of speech rate does differ by prosodic boundary type. This could be reported in a paper as: > "The effect of speech rate differed by prosodic boundary type: speech rate had a stronger effect for null boundaries than for non-null (i.e. weak or strong) boundaries ( $\hat{\beta} = 1.91$ ,  $t = 2.7$ ,  $p = 0.007$ ), but did not significantly differ between weak and strong boundaries ( $p = 0.9$ )."

### Example 2: Three-way interaction

#### Data: givenness

Recall that for this data, the strength of the Williams effect (the `conditionLabel` slope) depends on `voice`:

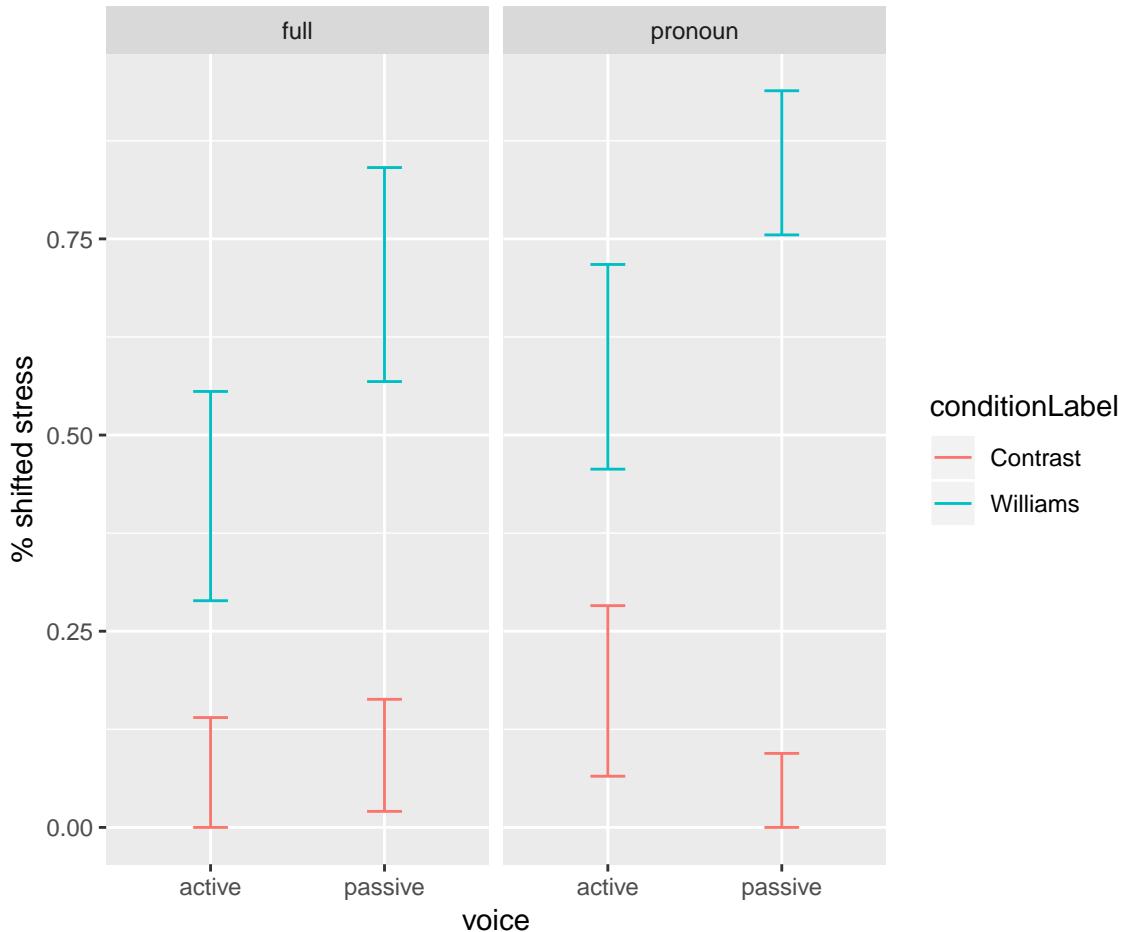
```
ggplot(aes(x=voice, y = stressshift.num), data=givenness) +
  stat_summary(fun.data="mean_cl_boot", geom='errorbar', width=0.2, aes(color=conditionLabel)) +
  ylab("% shifted stress")
```



Let's now ask: does this dependence differ between full NPs and pronouns? This question is addressed by including a **3-way interaction** in the model: `conditionLabel:voice:npType`.

In the empirical data:

```
ggplot(aes(x=voice, y = stressshift.num), data=givenness) +
  stat_summary(fun.data="mean_cl_boot", geom='errorbar', width=0.2, aes(color=conditionLabel)) +
  ylab("% shifted stress") + facet_wrap(~npType)
```



this three-way interaction corresponds to asking: is there a qualitative difference between the left and right panels? (That is: a difference in how `voice` modulates the `conditionLabel` effect.)

#### Questions:

- What is the expected *sign* of the three-way interaction term? What is its interpretation?  
("For full NPs, A has a bigger effect on... compared to ...")

Check your intuition against the last row of the fitted model:

```
summary(glm(formula = stressshift ~ conditionLabel.williams * npType.pron * voice.passive, family = "binomial", data = givenness))
```

```
## 
## Call:
## glm(formula = stressshift ~ conditionLabel.williams * npType.pron *
##       voice.passive, family = "binomial", data = givenness)
## 
## Deviance Residuals:
##      Min        1Q     Median        3Q       Max
## -1.9728   -0.5746   -0.3518    0.5553    2.5601
## 
## Coefficients:
## (Intercept)           Estimate Std. Error
## -0.9807          0.1704
## conditionLabel.williams 3.2322          0.3347
## npType.pron            0.4295          0.3403
```

```

## voice.passive          0.3168   0.3392
## conditionLabel.williams:npType.pron      0.7082   0.6684
## conditionLabel.williams:voice.passive    1.9230   0.6662
## npType.pron:voice.passive      -0.8359   0.6776
## conditionLabel.williams:npType.pron:voice.passive  2.1099   1.3308
##                                     z value Pr(>|z|)
## (Intercept)                  -5.756 8.61e-09 ***
## conditionLabel.williams      9.658 < 2e-16 ***
## npType.pron                   1.262  0.20691
## voice.passive                0.934  0.35028
## conditionLabel.williams:npType.pron      1.060  0.28937
## conditionLabel.williams:voice.passive    2.887  0.00389 **
## npType.pron:voice.passive      -1.234  0.21731
## conditionLabel.williams:npType.pron:voice.passive  1.585  0.11289
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
## Null deviance: 496.24  on 381  degrees of freedom
## Residual deviance: 323.94  on 374  degrees of freedom
## AIC: 339.94
##
## Number of Fisher Scoring iterations: 5

```

The three-way `conditionLabel.williams:npType.pron:voice.passive` interaction effect is:

- Positive
- Not significant ( $p = 0.11$ ).

Note that the `conditionLabel.williams` and `conditionLabel.williams:voice.passive` effects are significant. Thus, there is a Williams effect (averaging across other variables), and this effect is stronger for passive-voice items. This modulation seems slightly larger for full NPs, but is not significantly larger.

## 6.5 Solutions

**Q:** What does the hypothesis test  $H_0 : \beta_1 = 0$  correspond to asking? (What levels are hypothesized to have the same value of  $Y$ ?)

**A:** “Do *level 1* and *level 2* have the same value (for the response)?”

---

**Q:** What is the interpretation of the second contrast’s coefficient (`contextNew`)?

**A:** Difference in log-odds of prominence shifting between `context=New` and `context=Alternative`.

---

**Q:** What does the hypothesis test  $H_0 : \beta_1 = 0$  correspond to? (What is equal to what?)

**A:** “Do *level 1* and the grand mean differ?”

---

**Q:** What are the interpretations of:

- The Intercept?

- `context.sum1?`
- `context.sum2?`

The coefficient for the second contrast isn't significant. What corresponding difference is not significant (roughly, "is very small"), in the empirical data?

**A:**

- Intercept: grand mean
- `context.sum1`: difference between *Alternative* and grand mean
- `context.sum2`: difference between *NoAlternative* and grand mean

So, the latter difference isn't significantly different from zero. You can see this in the empirical plot: the value of green isn't so different from the mean of green, red, and blue.

---

**Q:** What directions do we expect?

**A:** The slope of the speech rate effect seems to be shallower (= less negative) for - than for  $2/1$ . Thus, we expect a **positive** value for the Contrast 2 regression coefficient (- slope minus  $2/1$  slopes). There doesn't look to be much difference between the 2 and 1 slopes, so we expect the Contrast 1 regression coefficient ( $2$  slope minus  $1$  slope) to be near zero.

---

**Q:** What is the expected *sign* of the three-way interaction term? What is its interpretation? ("For full NPs, A has a bigger effect on... compared to ...")

**A:** Positive. "The difference in Williams effect size between passive and active voice items is larger for pronouns than for full NPs."

# Chapter 7

## Linear mixed models

### Preliminary code

This code is needed to make other code below work:

```
library(ggplot2)
library(dplyr)
library(languageR)
library(Hmisc)
library(arm)

## loads givennessMcGillLing620.csv from OSF project for Wagner (2012) data
givenness <- read.csv(url("https://osf.io/q9e3a/download"))

## define numeric versions of factors, for convenience
givenness <- mutate(givenness,
  conditionLabel.williams = arm::rescale(conditionLabel),
  npType.pronoun = arm::rescale(npType),
  npType.pron = arm::rescale(npType),
  voice.passive = arm::rescale(voice),
  order.std = arm::rescale(order),
  stressshift.num = (as.numeric(stressshift) - 1))

## make non-mixed-effect model prediction for examples below (just one prediction per
## level of conditionLabel)
newdata <- data.frame(conditionLabel.williams=sort(unique(givenness$conditionLabel.williams)))

mod1a <- lm(acoustics ~ conditionLabel.williams, data=givenness)

newdata$pred <- predict(mod1a, newdata=newdata)

newdata$conditionLabel <- factor(levels(givenness$conditionLabel))

## loads halfrhymeMcGillLing620.csv from OSF project for Harder (2013) data
halfrhyme <- read.csv(url("https://osf.io/37uqt/download"))
```

**Note:** Answers to some questions/exercises not listed in text are in Solutions

## 7.1 Mixed-effects models: Motivation

Data analysis can be split into two parts: exploratory (EDA), and confirmatory (CDA).<sup>1</sup> In this book, EDA always accompanies CDA. But the ultimate goal of a study of linguistic data is usually **confirmatory data analysis**: we want to make generalizations about units drawn from a population, based on finite data. As such, we want our statistical analysis techniques—such as regression models—to generalize to new data, not just describe the sample we happened to draw.

A key assumption of regression models, which we discussed in the context of linear regression, is *independence of errors*. When we fit a regression model of  $Y$  as a function of some predictors, it is assumed that the “errors” for each observation—how off the model’s prediction is, after taking predictor values into account—are independent. That is, knowing how off the model is for observation 1 doesn’t tell us anything about how off it is for observation 10, and so on.

However, at least for linguistic data, usually the **unit** over which we want to generalize is not observations; it is some higher-level grouping, such as participants, “items”, sentences, and so on. We usually take more than one observation per unit level (Baayen et al., 2008):

- Multiple observations per participant (since we’ve already paid them to do the experiment)
- Multiple observations per item (because designing items is time-consuming)

and so on.

But in general, observations from within a unit level do not have independent errors! For example, in a study of lexical decision reaction time as a function of participant age and word frequency (such as the [english dataset](#engdata)), certain participants will be characteristically fast (low RT), compared to other participants of the same age—perhaps because they just had coffee, are highly motivated to finish the experiment, or some other reason. Similarly, errors won’t be independent from multiple observations of the same word. Certain words will just take longer to recognize, beyond the effect of word frequency—such as (orthographically) longer words, or words less familiar to an undergraduate participant population.

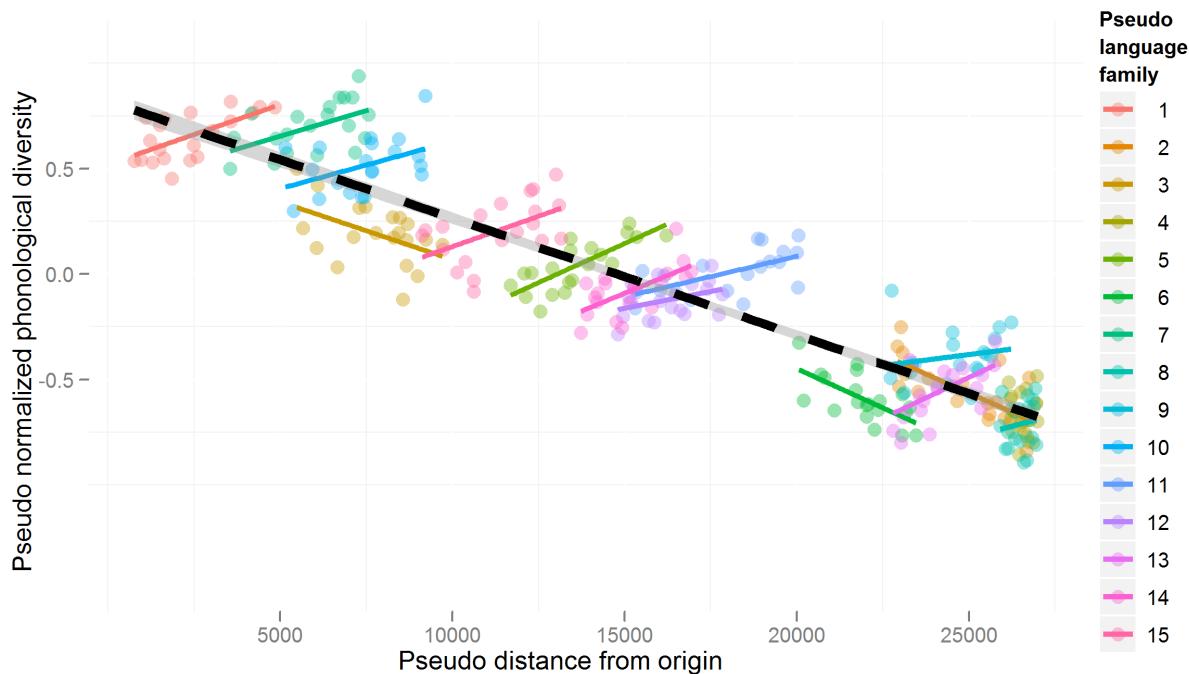
Grouping by one or more units, or *grouping factors*, is the norm in linguistic data—as well as in most data from behavioral sciences.

We discussed many different assumptions made by regression models in the linear regression chapter, but the independence-of-errors assumption is particularly crucial. Let’s see a few examples of what can happen when grouping by unit is not taken into account.

---

<sup>1</sup>This distinction is discussed, along with an introduction to EDA and motivation for its importance in data analysis, in a lecture that has not yet been turned into a chapter of this book. Until it is, this section just assumes you have read about EDA and CDA somewhere, such as the first sections of Behrens (1997). Tukey’s original manifesto for combining EDA and CDA (Tukey, 1980) is still valuable.

### 7.1.1 Simpson's paradox



Source: <https://hlplab.wordpress.com/2011/05/31/mixed-models-and-simpsons-paradox/>

This data is from a study examining the relationship between a language's geographic distance from a fixed point ( $X$ ) and its phonological diversity ( $Y$ ). Languages are grouped into families (the “grouping factor”), because related languages will have very similar values of  $X$  and  $Y$ , for reasons not of interest here. Of interest is how  $X$  affects  $Y$ —after accounting for family differences.

Without taking grouping into account, the relationship between  $X$  and  $Y$  would look negative. However, once grouping is taken into account, we can see that the relationship between  $X$  and  $Y$  is actually positive, within language families. (In addition, there is a negative relationship between a family's mean  $X$  and mean  $Y$ .)

That is: we would make the **opposite conclusion** if grouping information were not taken into account. This situation, where a trend is reversed or disappears within groups compared to across groups, is called *Simpson's paradox*. A famous example related to graduate admissions at UC Berkeley is described here, with useful visualizations.

Simpson's paradox is an extreme example, where the actual effect direction is flipped. More commonly, not taking grouping into account can result in Type I errors (spurious effects) or Type II errors (missing an effect that is really there):

- Type I error: A couple items drive an otherwise null effect
- Type II error: Consistent effect within items, but not in pooled data

In a future version of these notes there will be actual illustrative examples, but for now one can see these possibilities by mentally adjusting the Simpson's paradox example shown above:

- Type I: Keep all green points where they are; move points of all other colors so they have y-axis value of 0.5
- Type II: Move each cluster of points vertically so it's centered at y-axis = 0.5.

In technical terms, Type I errors can result from non-independence of errors because the *degrees of freedom* assumed by the model are wrong. Intuitively,  $n$  different observations from the same item should not count as  $n$  independent pieces of information about how a property of items (like word frequency) affects the response—as  $n$  different observations all from different items would count.

### 7.1.2 Repeated-measure ANOVAs

One technique for analyzing grouped data is *repeated measure ANOVAs* (RM-ANOVA). We will not discuss how to do RM-ANOVA's, but they are widely used.<sup>2</sup> For experimental linguistics in particular, especially psycholinguistics, RM-ANOVAs were the norm for statistical analysis from the mid 1970s to the early 2010s. Thus, some context is helpful.

An RM-ANOVA analysis assumes (relatively) equal sample size in different levels of the same grouping factor (e.g. same number of observations per participant), and that there is a single grouping factor (e.g. participants or items). The most common case for linguistic data is to have two or more grouping factors, such as participant and item. The analyst carries out one RM-ANOVA per grouping factor: separate “by-participant” and “by-item” RM-ANOVAs.<sup>3</sup> Each RM-ANOVA results in an  $F$  statistic value, often called  $F_1$  (“by-participant”) and  $F_2$  (“by-item”).  $F_1$  and  $F_2$  are combined into a single statistic,  $\text{min}F$ . In practice, the by-participant and by-item RM-ANOVAs are often computed and interpreted without considering  $\text{min}F$ , as in the example below.

Here is a sample RM-ANOVA analysis report (from Salverda et al. (2003)):

**The fixation proportion to the competitor picture, averaged over the 300–900 ms time window, was 27% in the monosyllabic-word condition and 24% in the carrier-word condition. A one-way ANOVA (monosyllabic condition vs. carrier condition) on the average fixation proportions revealed that this difference was significant by participants but not by items ( $F_1(1, 29) = 5.9, P < 0.05$ ;  $F_2(1, 27) = 1.5, P > 0.10$ ), suggesting large variability across items. A two-way (Condition  $\times$  Time Window [300–550 ms vs. 550–900 ms]) ANOVA revealed a significant effect of Window ( $F_1(1, 29) = 65.7, P < 0.001$ ;  $F_2(1, 27) = 19.1, P < 0.001$ ), an effect of Condition significant only by participants ( $F_1(1, 29) = 5.2, P < 0.05$ ;  $F_2(1, 27) = 1.4, P > 0.10$ ), and no interaction ( $F_1$  and  $F_2 < 1$ ).**

Note that the report discusses significance “by participant” versus “by item”, referring to the  $F_1$  and  $F_2$  RM-ANOVA results for the same effect.

RM-ANOVA is a perfectly fine methodology for analyzing grouped data, especially for relatively simple experiments with balanced designs. Statistical power will be lower than for an equivalent mixed model, but ideally power should be high anyway in a simple experiment. Some discussion of RM-ANOVAs, including contextualization with respect to mixed models, is given by Barr et al. (2013); Baayen (2008), Sec. 7.2.1; Baayen et al. (2008).<sup>4</sup>

Mixed-effects regression models, or *mixed models*, are another way of analyzing grouped data. Mixed models are much more powerful than RM-ANOVA, and have become the standard for analyzing grouped data in many areas of language sciences, as well as other fields. The advantages of mixed models over RM-ANOVAs for analyzing grouped data include:

<sup>2</sup>See for example: Johnson (2008), Sec. 7.8 of this tutorial (by Jonathan Baron and Yuelin Li), these notes by Brian Dillon, many other tutorials online.

<sup>3</sup>Grouping factors are often called *random factors* in RM-ANOVA analyses.

<sup>4</sup>Mixed models have become (by 2017) sufficiently popular that many language scientists, including journal article reviewers, think that an analysis of grouped data that doesn't use mixed models is somehow “incorrect” or “not standard”. It is important to remember that there is always more than one valid way to analyze the same data. Mixed models are often appropriate for analyzing grouped data, but simpler methods—such as RM-ANOVA or even paired  $t$  tests—may also be suitable in many cases, especially for simple statistical analyses where a mixed model may be overkill.

- More than one grouping unit can be included in the same model
- Unequal number of observations per level is OK
- **Explicitly model variability** among levels of the same grouping factor
  - Ex: By participant variability (“individual differences”)
  - This means the analyst can explore and test questions about the variability itself (as opposed to effects averaging across participants, etc.), which is not possible using RM-ANOVA, or other methods we’ve covered so far.

## 7.2 Linear mixed models 1: One grouping factor, random intercepts

We will introduce each part of mixed models through a series of examples, using just one predictor ( $X$ ) to keep things simple.

We start with the simplest case of mixed models—where there is variability between groups just in the value of the intercept. This first example uses the `givenness` dataset, taking into account by-participant variability (grouping factor = `participant`).

First, some notation:

- $n$ : number of observations.
- The grouping factor is `participant`.
  - There are  $J$  levels/groups (i.e.,  $J$  participants)
  - Observation  $i$  is in group  $j[i]$  ( $j[i] \in \{1, \dots, J\}$ )
- The response is `acoustics` ( $Y$ )
  - $y_i$ : value for the  $i^{\text{th}}$  observation
- There is a single predictor, `conditionLabel.williams` ( $X$ )
  - $x_i$ : value for the  $i^{\text{th}}$  observation
  - $0.5 = \text{Williams}$ ,  $-0.5 = \text{Contrast}$

(Italics are used to refer to levels of a factor, like *Williams* for the `conditionLabel` factor.)

We first consider a simple linear regression model *without* by-participant variability (Model 1A), then introduce the first mixed model (Model 1B).

### 7.2.1 Model 1A: Simple linear regression

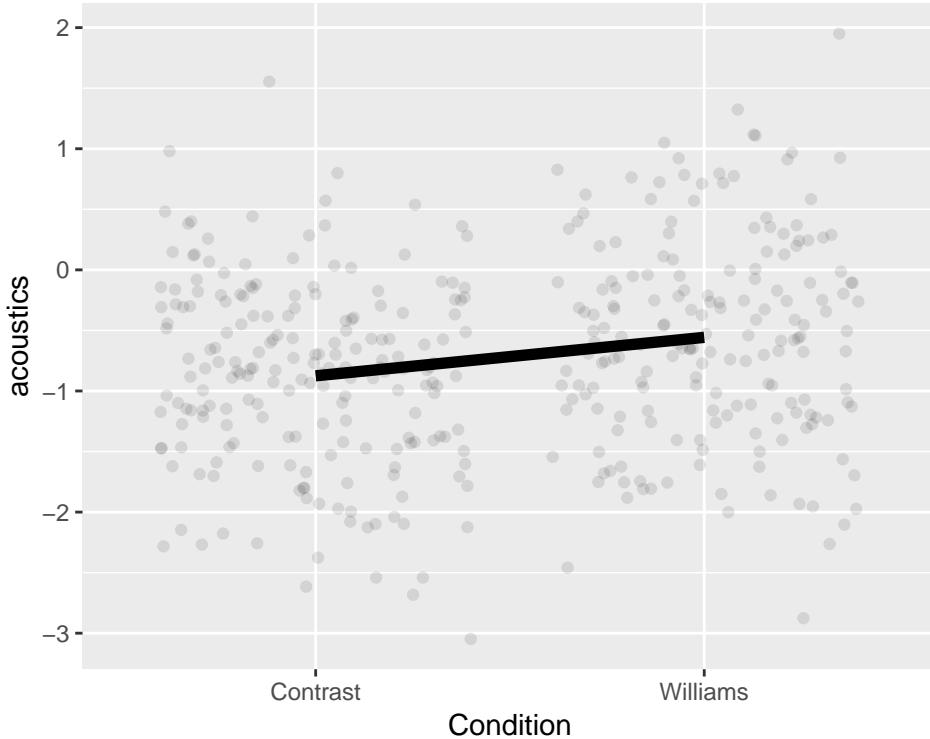
The model for simple linear regression is:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon_i, \quad i = 1, 2, \dots, n$$

The errors are assumed to be normally distributed (and independent):

$$\epsilon_i \sim N(0, \sigma^2)$$

This model assumes the same slope and intercept for **all groups** (i.e., all participants), as schematized here:



The same `acoustics` value is predicted for all observations in each condition, regardless of which participant an observation comes from.

To fit and summarize this model:

```
mod1a <- lm(acoustics ~ conditionLabel.williams, data = givenness)
summary(mod1a)
```

```
##
## Call:
## lm(formula = acoustics ~ conditionLabel.williams, data = givenness)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.31843 -0.56264  0.01977  0.53651  2.50851
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.72205    0.04109 -17.570 < 2e-16 ***
## conditionLabel.williams 0.31774    0.08224   3.863 0.000131 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8032 on 380 degrees of freedom
## Multiple R-squared:  0.03779,   Adjusted R-squared:  0.03526
## F-statistic: 14.93 on 1 and 380 DF,  p-value: 0.0001315
```

Since the data is grouped by participant (and item), this model violates the **independent errors assumption** of linear regression. Participants might differ in:

1. The mean value of `acoustics`
2. The effect of `conditionLabel.williams`

Our goal in fitting the model is to assess whether there's an effect of `conditionLabel.williams`, beyond individual differences:

$$H_0 : \beta_1 = 0$$

Either type of differences between participants could lead to falsely rejecting or accepting  $H_0$ .

Mixed models deal with non-independence of errors by using two types of coefficients:

1. *Random effects*: Coefficients which vary between groups
2. *Fixed effects*: Coefficients which don't vary between groups

Fixed effects are basically what we have referred to as "regression coefficients" in the linear regression and logistic regression models covered in previous chapters.

The next two models we consider have two different kinds of random effects:

- Model 1B: Intercept varies between participants
- Model 1C: Intercept and slope vary between participants

### 7.2.2 Model 1B: Random intercept only

This model makes predictions as in Fig. 7.1: one line per participant, differing in intercept but not in slope.

The regression model for a simple linear regression with a *random intercept* is:

$$y_i = \beta_0 + \alpha_{j[i]} + \beta_1 x_i + \epsilon_i, \quad i = 1, 2, \dots, n$$

$\alpha_{j[i]}$  is the random intercept, which captures the offset of group  $j[i]$  from the grand mean ( $\beta_0$ ). The random intercept values for different participants are assumed to be normally distributed:

$$\alpha_j \sim N(0, \sigma_s^2), \quad j = 1, 2, \dots, J$$

where  $\sigma_s$  captures the amount of between-participant variation in the intercept value.

The errors  $\epsilon_i$  are still assumed to be normally distributed, as in a non-mixed-effects linear regression:

$$\epsilon_i \sim N(0, \sigma_e^2), \quad j = 1, 2, \dots, n$$

where we now write the error variance as  $\sigma_e$  (instead of  $\sigma$ ), to distinguish it from  $\sigma_s$ .

This regression equation implies different models of how  $Y$  depends on  $X$ , for different participants. For example, for participant #5:

$$Y = \beta_0 + \alpha_5 + \beta_1 X + \epsilon$$

where  $(\beta_0 + \alpha_5)$  is the value of the intercept for participant 5.

The model "across participants", or "for an average participant", is:

$$Y = \beta_0 + \beta_1 X + \epsilon$$

Here, the intercept is  $\beta_0$ : the overall intercept, averaged across participants. This model is identical to Model 1A—simple logistic regression, without random effects.

What is the same in both models is the slope of  $X$  (and the variance of the error term).

Fitting Model 1B means estimating:

1. The *fixed effect coefficients*:

- $\beta_0$ : grand mean
  - $\beta_1$ : slope of `conditionLabel.williams`
2. The *variance components*:
- $\sigma_s^2$ : variability of intercept offset across participants
  - $\sigma_e^2$ : residual error variance

### Detour: Fitting linear mixed-effects models

We will use the `lme4` package for fitting (linear) mixed effects models, which is the most widely-used R package for such models. `lme4` is both widely used and frequently updated. (Because of the latter, it's important to cite the exact version of `lme4` used when reporting a mixed-effects model in a paper. The version of `lme4` used to fit models in this chapter is 1.1.18.1.) See Bates et al. (2014) for the mathematical details of fitting these models.

Functions in this package fit mixed models by maximizing either:

- *maximum likelihood (ML)*, which gives biased variance component estimates, or
- *restricted ML (REML)*, which gives unbiased variance component estimates, and is typically the default

The difference between ML and REML only matters for small sample sizes. Technically you must use ML fits for model comparison to make sense, and if you try to compare two `lme4` models using `anova()` in R, the models will automatically be re-fit using ML before comparing (as of 2018).<sup>5</sup>

#### 7.2.2.1 Fitting Model 1B

We fit the mixed model described above using the `lmer` function (from the `lme4` package):

```
mod1b <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant), data = givenness)
```

where `(1|participant)` is `lme4` notation for a by-participant random intercept (1 means “intercept”, `|participant` means “grouped by participant”).

**Note:** You do **not** need to write `lme4::lmer` in general, just `lmer`. The notation `lme4::lmer` is used here to make sure we use the `lmer` function from the `lme4` package, rather than the redefined version of `lmer` from the `lmerTest` package discussed in Sec. 7.5.3.3.

The model's output is:

```
summary(mod1b)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: acoustics ~ conditionLabel.williams + (1 | participant)
##   Data: givenness
##
## REML criterion at convergence: 897.9
##
## Scaled residuals:
##       Min     1Q Median     3Q    Max
## -3.0538 -0.7129  0.0083  0.6540  3.3136
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   Groups      Name        Variance Std.Dev.
```

<sup>5</sup>More precisely: to compare two models with different fixed-effect terms, using any likelihood-based method (such as a LR test or AIC), the models must be fitted using maximum likelihood (Faraway, 2016; Zuur, 2009). See here for some discussion.

```

##  participant (Intercept) 0.08937  0.299
##  Residual                 0.55800  0.747
## Number of obs: 382, groups: participant, 27
##
## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)              -0.71856   0.06916 -10.39
## conditionLabel.williams  0.32626   0.07677   4.25
##
## Correlation of Fixed Effects:
##             (Intr)
## cndtnLbl.wl  0.002

```

Under Random effects:

- $\hat{\sigma}_s^2 = 0.089$ : estimated **by-participant variability** in the intercept.
- $\hat{\sigma}_e^2 = 0.558$ : estimated **residual error variance**

Under Fixed effects: estimates of the fixed-effect coefficients:

- $\hat{\beta}_0 = -0.71$ : intercept
- $\hat{\beta}_1 = 0.326$ : slope

### 7.2.2.2 Interpretation

The model predicts that for an “average participant”, the relationship between condition and **acoustics** is:

$$\text{acoustics} = -0.71 + 0.326 \times \text{conditionLabel.williams}$$

Because the random intercepts are normally distributed, we expect (approximately) 95% of participants to have intercepts within  $2\sigma_s$  of the overall intercept,  $\beta_0$ . Thus, the model predicts that 95% of participants in the population have intercepts between -1.309 and -0.113:

- lower bound:  $\hat{\beta}_0 - 2 \cdot \hat{\sigma}_s = -0.71 - 2 \cdot 0.299 = -1.309$
- upper bound:  $\hat{\beta}_0 + 2 \cdot \hat{\sigma}_s = -0.113$

(In addition,  $\approx 95\%$  of observations are predicted to have an error between -1.5 and 1.5,  $= \pm 2 \cdot \hat{\sigma}_e$ .)

We can compare the residual error (how much variance is “left over”) in the simple linear regression Model (1A) and the random-intercept model (1B):

- Model 1A:  $\hat{\sigma}_e^2 = 0.803^6$
- Model 1B:  $\hat{\sigma}_e^2 = 0.558$

The residual error variance is much smaller in the random-intercept model. This is because in the random-intercept model, the error of the simple linear regression model has been partitioned into **participant-level** error and **observation-level** error, captured by  $\sigma_s$  and  $\sigma_e$ . Intuitively, some error has been “given” to the participant level.

Note that the **fixed-effect** coefficients  $\hat{\beta}_0$  and  $\hat{\beta}_1$  (the “intercept” and “slope”) are nearly the same in Model 1A and Model 1B.

#### Questions:

- What does this mean? (That the fixed-effect coefficients are the same in the predictions made by Model 1A and Model 1B?)

---

<sup>6</sup>This is the “Residual standard error” in the R summary of the model output (`summary(mod1a)`).

### 7.2.2.3 By-participant predictions

Although the random effects  $\alpha_1, \dots, \alpha_J$  aren't fitted parameters, we can extract the following from the model:

- Estimates of the random effects, the “best linear unbiased predictors” (or *BLUPs*)
- Standard errors of the BLUPs

This allows us to extract the model predictions for each participant:

```
ranef(mod1b)$participant # ranef() found in library(lme4), library(arm)
```

```
##      (Intercept)
## 24    0.085333501
## 297   0.036820472
## 432   -0.411839925
## 524   0.116659623
## 529   -0.485395105
## 530   -0.077424563
## 540   0.244536341
## 541   0.193392070
## 542   -0.126468368
## 544   0.012462384
## 547   -0.335559107
## 548   0.489860310
## 549   -0.135853392
## 550   -0.047183013
## 552   -0.032362240
## 553   0.239715782
## 554   -0.045139278
## 555   -0.204230783
## 556   -0.104644687
## 557   0.199844626
## 558   0.004117646
## 559   -0.336348686
## 560   0.307842484
## 561   0.530048433
## 562   -0.163493454
## 563   -0.072168272
## 564   0.117477201
```

```
se.ranef(mod1b)$participant # se.ranef found in library(arm)
```

```
##      (Intercept)
## 24    0.1583862
## 297   0.1620712
## 432   0.1620712
## 524   0.1853443
## 529   0.1660260
## 530   0.1620712
## 540   0.1660260
## 541   0.1660260
## 542   0.1583862
## 544   0.1748900
## 547   0.1620712
## 548   0.1660260
## 549   0.1583862
```

```

## 550  0.1620712
## 552  0.1748900
## 553  0.1660260
## 554  0.1620712
## 555  0.1620712
## 556  0.1620712
## 557  0.1660260
## 558  0.1660260
## 559  0.1702852
## 560  0.1660260
## 561  0.1620712
## 562  0.1583862
## 563  0.1798897
## 564  0.1702852

```

These values might be useful if you are interested in which participants have particularly high or low baseline values of your dependent variable, for example.

A simpler way to get predictions is to just use `predict(yourmodel, newdata = dataframeToPredictFor)`. (`predict` is a useful function in general, which also works with `lm` and `glm`.)

For example:

```

## MODEL 1B: random intercept only
mod1b <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant), data=givenness)

## set up a dataframe for which the model should predict new values:
## each level of conditionLabel, for each participant.

## it's easiest to understand this if we first refit a version of mod1b using the *factor* version of c

mod1b.1 <- lme4::lmer(acoustics ~ conditionLabel + (1|participant), data=givenness)

## set up a dataframe to predict values for: one row per participant/cond label pair
newdata <- data.frame(expand.grid(conditionLabel=unique(givenness$conditionLabel),
                                    participant=unique(givenness$participant)))

## get the predicted value for each case
newdata$pred <- predict(mod1b.1, newdata=newdata)

## plot the model's prediction for each participant:
ggplot(aes(x=conditionLabel, y=pred), data=newdata) +
  geom_line(aes(x=conditionLabel,y=pred, group=participant)) +
  xlab("Condition") +
  ylab("Model prediction")

```

We can also get 95% CIs, as in the example in Sec. 7.12.

Let's compare the predictions made by Model 1A (simple linear regression) and Model 1B (random intercept), for each participant:

```

## get the same predictions but for model 1. first fit a version with the factor for conditionlabel:

mod1a.1 <- lm(acoustics ~ conditionLabel, data=givenness)

newdata$pred.mod1a <- predict(mod1a.1, newdata=newdata)

```

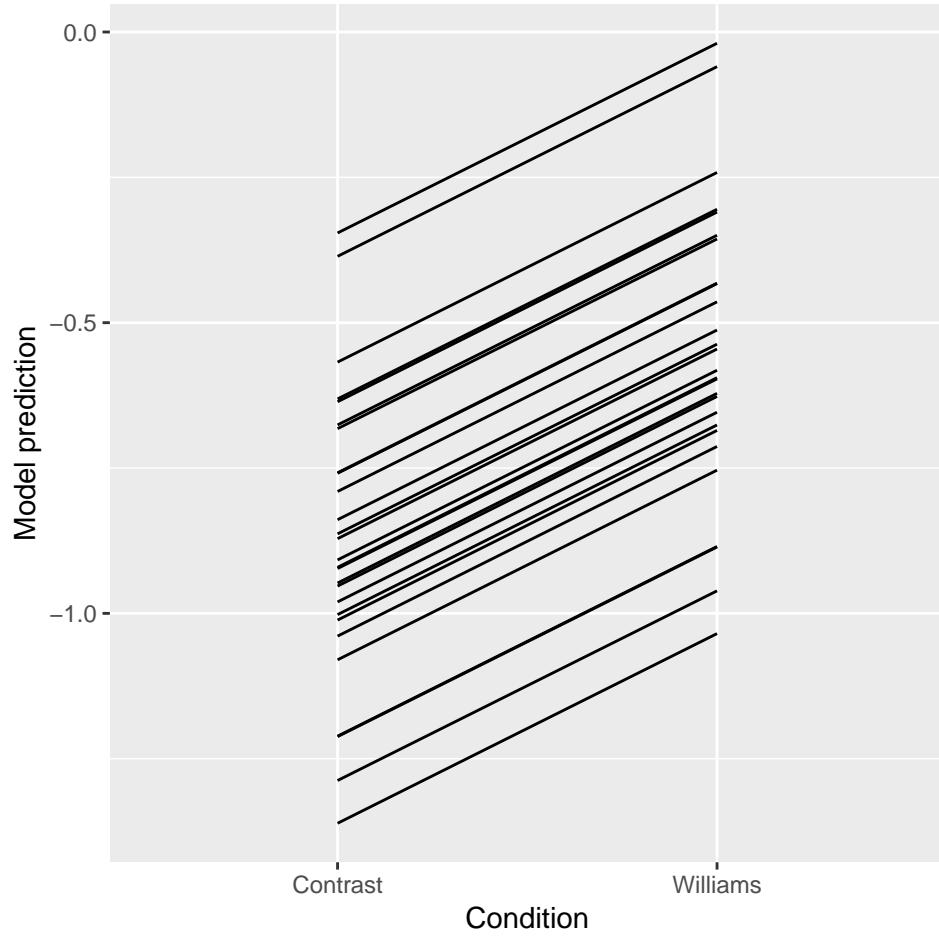


Figure 7.1: Predictions for each participant from Model 1B.

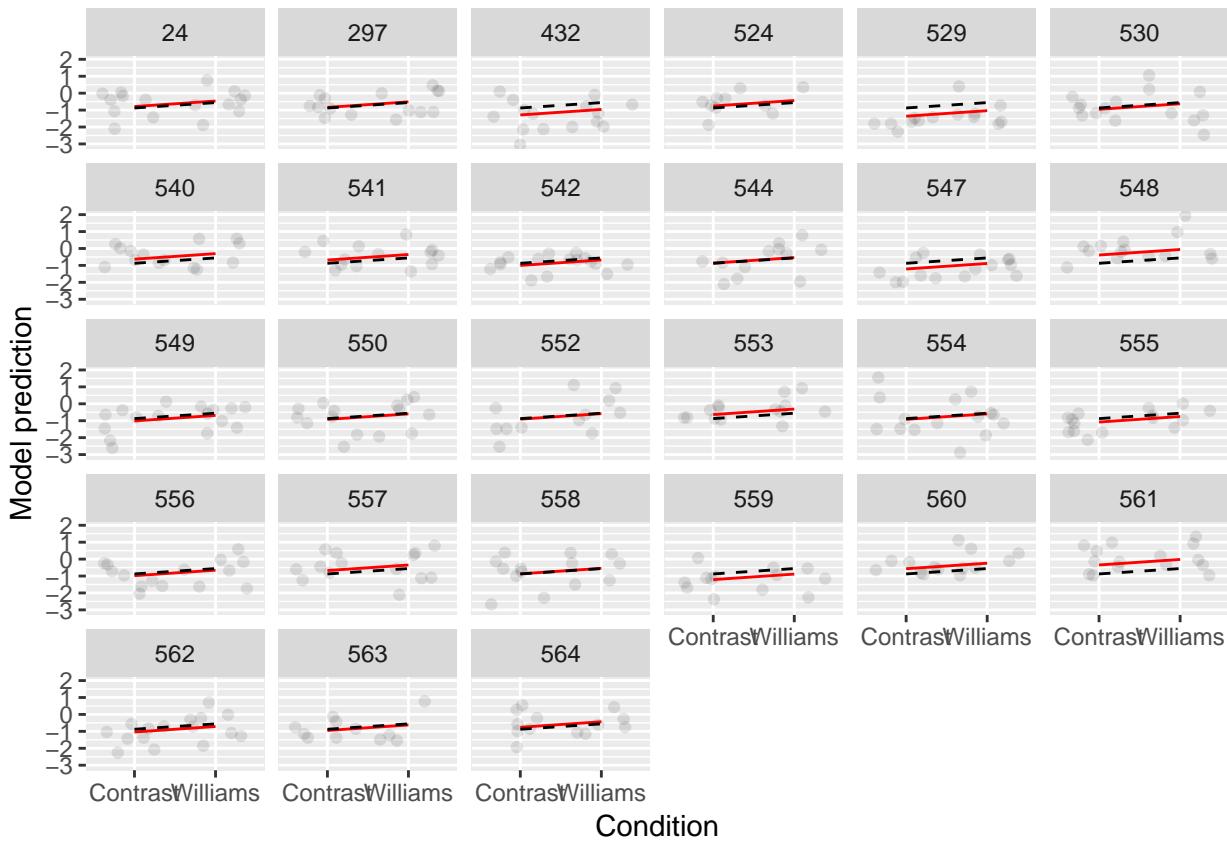


Figure 7.2: Model 1A shown in black, Model 1B shown in red.

```
ggplot(aes(x=conditionLabel, y=acoustics), data=givenness) +
  geom_jitter(alpha=0.1) +
  geom_line(aes(x=conditionLabel,y=pred, group=participant), color='red', data=newdata) +
  geom_line(aes(x=conditionLabel,y=pred.mod1a, group=participant), lty=2, data=newdata) +
  xlab("Condition") +
  ylab("Model prediction") +
  facet_wrap(~participant)
```

Model 1A always makes the same prediction—the black line—which moves vertically for different participants (red lines).

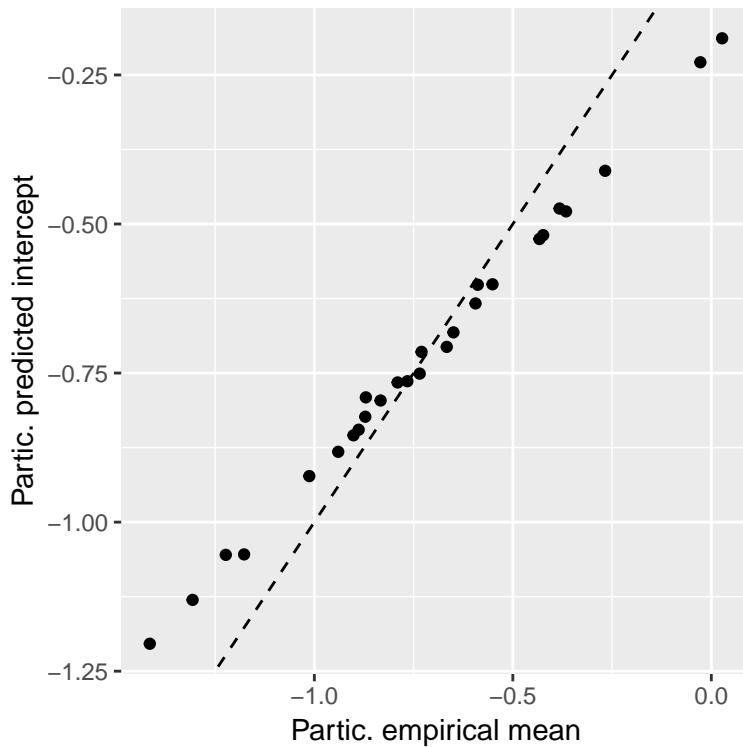
To get a sense of what the random-intercept model is doing, we can compare the predicted intercept (participant  $i$ :  $\hat{\beta}_0 + \hat{\alpha}_{j[i]}$ ) with the empirical mean for each participant:

```
## empirical mean acoustic values by participant
df <- givenness %>% group_by(participant) %>% summarise(meanAcoustics=mean(acoustics))

## participant random effects
df2 <- ranef(mod1b)$participant
df$modPred <- fixef(mod1b)[1] + df2[match(df$participant, rownames(df2)),1]

ggplot(aes(x=meanAcoustics, y=modPred), data=df) +
  geom_point() +
  geom_abline(lty=2) +
  xlab("Partic. empirical mean") +
```

```
ylab("Partic. predicted intercept")
```



The predicted intercepts are closer to the grand mean than the empirical values, a phenomenon called *shrinkage*, because the (absolute value of the) random intercepts are “shrunk” from the empirical means. Why?

- The predicted value (random effect) is a weighted average of the participant’s (empirical) mean and the grand mean.
- More observations  $\Rightarrow$  greater weight given to empirical mean

Shrinkage improves generalization of the model to data from new participants. But importantly, it also means that BLUPs are **not the fitted values** for each participant. They are *intentionally* not near the empirical means. In contrast, the fixed-effect coefficients ( $\hat{\beta}_0, \hat{\beta}_1$ ) should be close to the values you’d estimate from empirical data.

Chapter 12 of Gelman and Hill (2007) discusses these points.

### 7.3 Linear mixed models 2: One grouping factor, random intercepts and slopes

Model 1B allowed the intercept to differ by participant—each participant’s “baseline” is different. In addition, the **slope** of an effect could differ by participant, which is captured in a mixed model by a *random slope* term. Fig. 7.2 shows what the predictions from this kind of model would look like.

For the **givenness** example used for Models 1A-1B, there turns out to be no detectable by-participant variation in the slope of `conditionLabel.williams`—see extra examples in Sec. 7.12. Thus, we use a different dataset, `halfrhyme`, where there is clear by-participant variation.

We have not discussed this dataset’s interpretation, and you can just think of  $Y$  and  $X$  as arbitrary variables if it’s helpful.

We will fit a model with:

- Response ( $Y$ ): `rhymeRating`
- Fixed effect ( $X$ ): `relDuration` (of vowel)
- Random effects:
  - By-participant intercept
  - By-participant random slope of `relDuration`

As in Models 1A and 1B, there is a single grouping factor: `participant`.

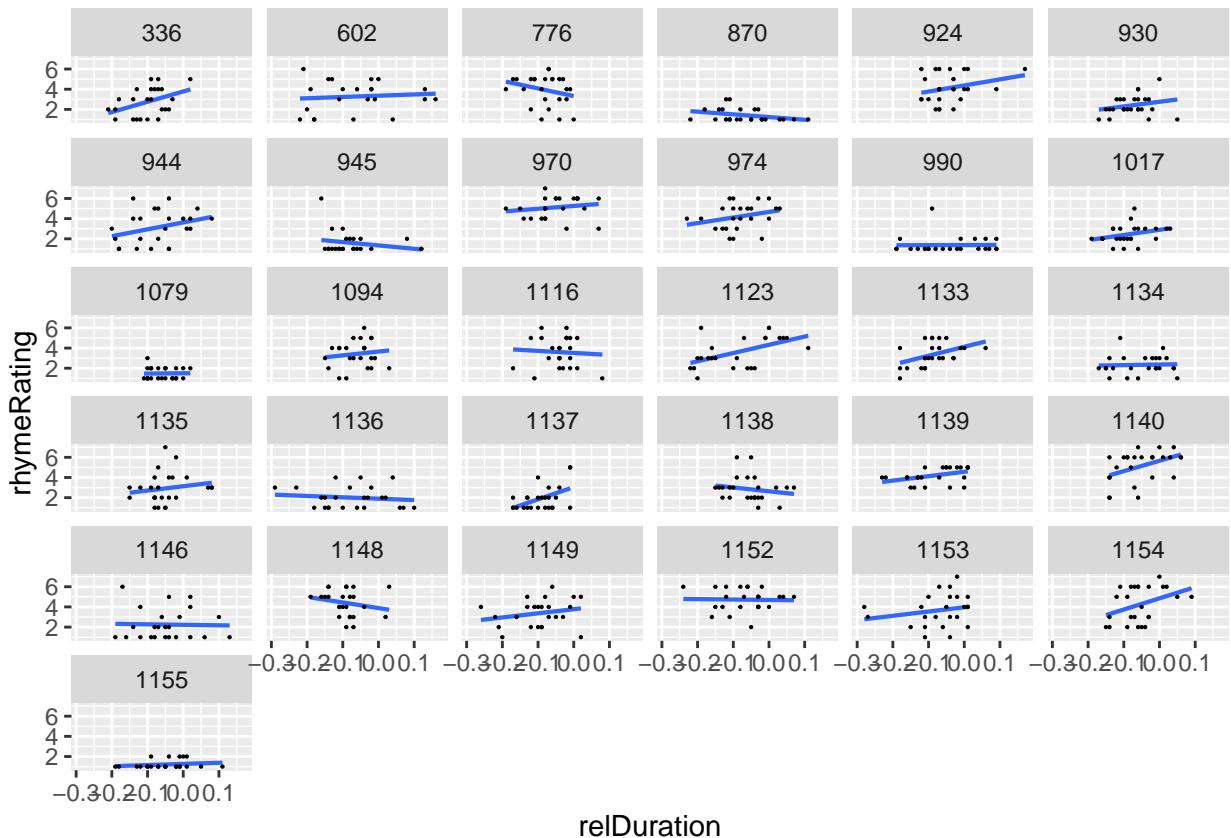
We first have to restrict to a subset of the data where `relDuration` is defined:

```
halfrhyme.orig <- halfrhyme
halfrhyme <- subset(halfrhyme, conditionLabel=='voice')
```

From the empirical data, it does look like participants might vary in both the intercept and the slope of the `relDuration` effect:

```
halfrhyme$participant <- as.factor(halfrhyme$participant)

ggplot(aes(x=relDuration, y=rhymeRating), data=halfrhyme) +
  geom_smooth(method='lm', se=F, size=0.75) +
  geom_point(size=0.1) +
  facet_wrap(~participant) +
  theme(legend.position='none') +
  xlab("relDuration")
```



### 7.3.1 Model 1C

The model for a simple linear-mixed effects regression with a random intercept and random slope for a single grouping factor is:

$$y_i = \beta_0 + \alpha_{j[i]} + (\beta_1 + \gamma_{j[i]})x_i + \epsilon_i, \quad 1, 2, \dots, n$$

As in previous models:

- $\beta_0, \beta_1$  are the fixed effects (intercept and slope)
- Errors are normally distributed:

$$\epsilon_i \sim N(0, \sigma_e^2), \quad i = 1, \dots, n$$

For random effects:

- $\alpha_{j[i]}$  is the **random intercept** term, which has the same interpretation as in Model 1B.
- $\gamma_{j[i]}$  is the **random slope** term, which captures how much each participant's slope (for  $X$ ) differs from the "average" slope across participants ( $\beta_1$ ).

The by-participant offsets in intercept and in the slope of  $X$  are normally distributed:

$$\begin{aligned} \alpha_j &\sim N(0, \sigma_{p,0}^2), \quad j = 1, \dots, J \\ \gamma_j &\sim N(0, \sigma_{p,1}^2) \end{aligned}$$

Fitting Model 1C means estimating:

- Fixed-effect coefficients:  $\beta_0, \beta_1$
- Variance components:
  - Random effect variances  $\sigma_{p,0}^2, \sigma_{p,1}^2$
  - Error variance  $\sigma_e^2$

### 7.3.2 Fitting Model 1C

To fit the model:

```
mod1c <- lme4::lmer(rhymeRating ~ relDuration + (1 + relDuration || participant),
  data=halfrhyme)
```

where  $(1 + relDuration || participant)$  is **lme4** notation for:

- Random intercept (1 means "intercept")
- Random slope of **relDuration** (random effects go to the left of the "pipe" symbol |)
- Grouped by participant (grouping factor goes to the right of the pipe)
- The random intercept and slope are **uncorrelated**—indicated by the double pipe ||. (We'll discuss this further when we introduce correlated random effects.)

$(1 + relDuration || participant)$  is read "uncorrelated by-subject random intercept and by-subject random slope".

The model's output is:

```

summary(mod1c)

## Linear mixed model fit by REML ['lmerMod']
## Formula:
## rhymeRating ~ relDuration + ((1 | participant) + (0 + relDuration |
##     participant))
## Data: halfrhyme
##
## REML criterion at convergence: 2578.7
##
## Scaled residuals:
##    Min      1Q  Median      3Q     Max
## -2.4022 -0.6091 -0.1126  0.6138  3.5726
##
## Random effects:
## Groups      Name      Variance Std.Dev.
## participant (Intercept) 1.267    1.126
## participant.relDuration 4.482    2.117
## Residual            1.552    1.246
## Number of obs: 756, groups: participant, 31
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 3.3003    0.2125 15.533
## relDuration 2.7249    0.7816  3.486
##
## Correlation of Fixed Effects:
##          (Intr)
## relDuration 0.192

```

The fixed effect coefficients, and the by-participant and **Residual** rows of “Random effects” have the same interpretation as for the random-intercept-only model (Model 1B). What is new is the random slope variance (second row of “Random effects”), which is estimated to be  $\hat{\sigma}_{p,1}^2 = 4.482$ .

The interpretation of this “random slope” term is:

- $\hat{\sigma}_{p,1} = 2.12$ : degree of variability among participants in the slope of **relDuration**
- $\approx 95\%$  of participants (in the population) have slope of **relDuration** **between -1.51 and 6.96**  
 $= \hat{\beta}_1 \pm 2 \cdot \hat{\sigma}_{p,1}$

#### Questions:

- How much variability is there in participants’ intercept values?

Again, we can use **raneff()** or **predict()** to get by-participant model predictions, using the estimated random effects (BLUPs).

To get model predictions for each participant, and visualize them:

```
## get model predictions for mod1c for each participant
```

```

## first, set up a prediction frame, say from min to max values of relDuration in the data, for each pa
newdata <- data.frame(
  expand.grid(
    relDuration=seq(min(halfrhyme$relDuration),
                   max(halfrhyme$relDuration), by=0.01),
    participant=unique(halfrhyme$participant)
  )
)
```

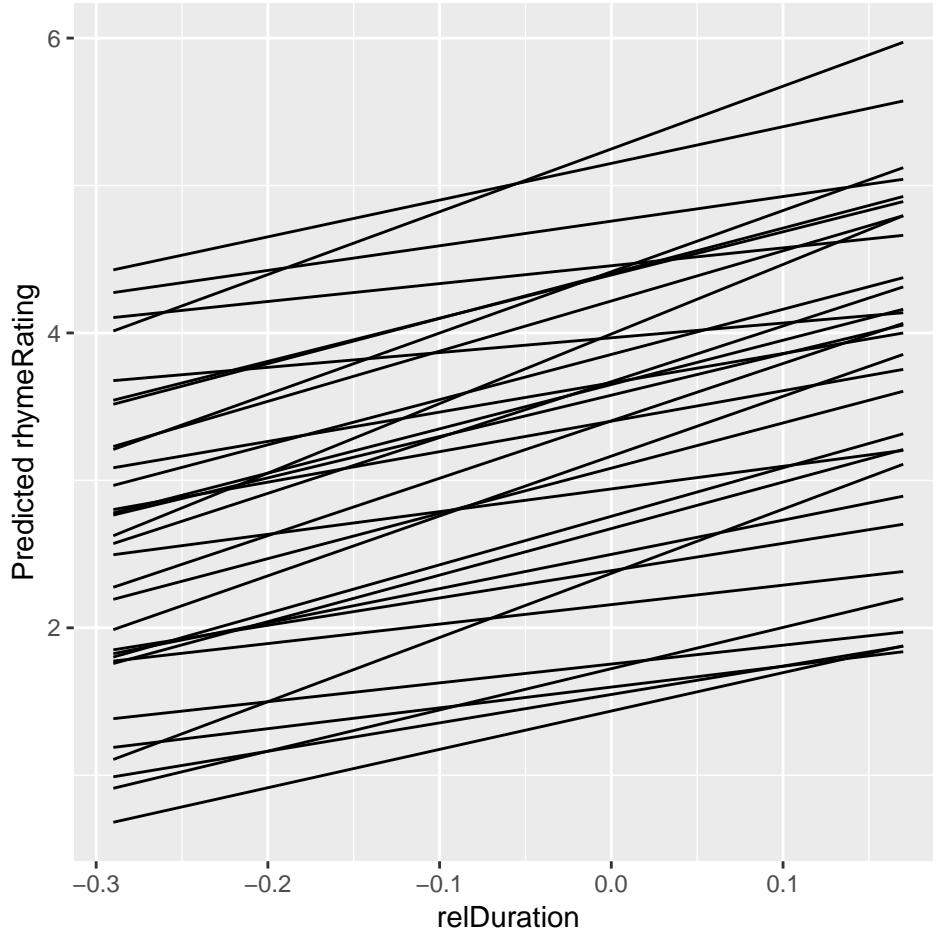


Figure 7.3: Predictions for each participant from Model 1C.

```

        )
    )

## get the predicted value for each case
newdata$pred <- predict(mod1c, newdata=newdata)

## plot the model's prediction for each participant:
ggplot(aes(x=relDuration, y=pred), data=newdata) +
  geom_line(aes(x=relDuration,y=pred, group=participant)) +
  xlab("relDuration") +
  ylab("Predicted rhymeRating")

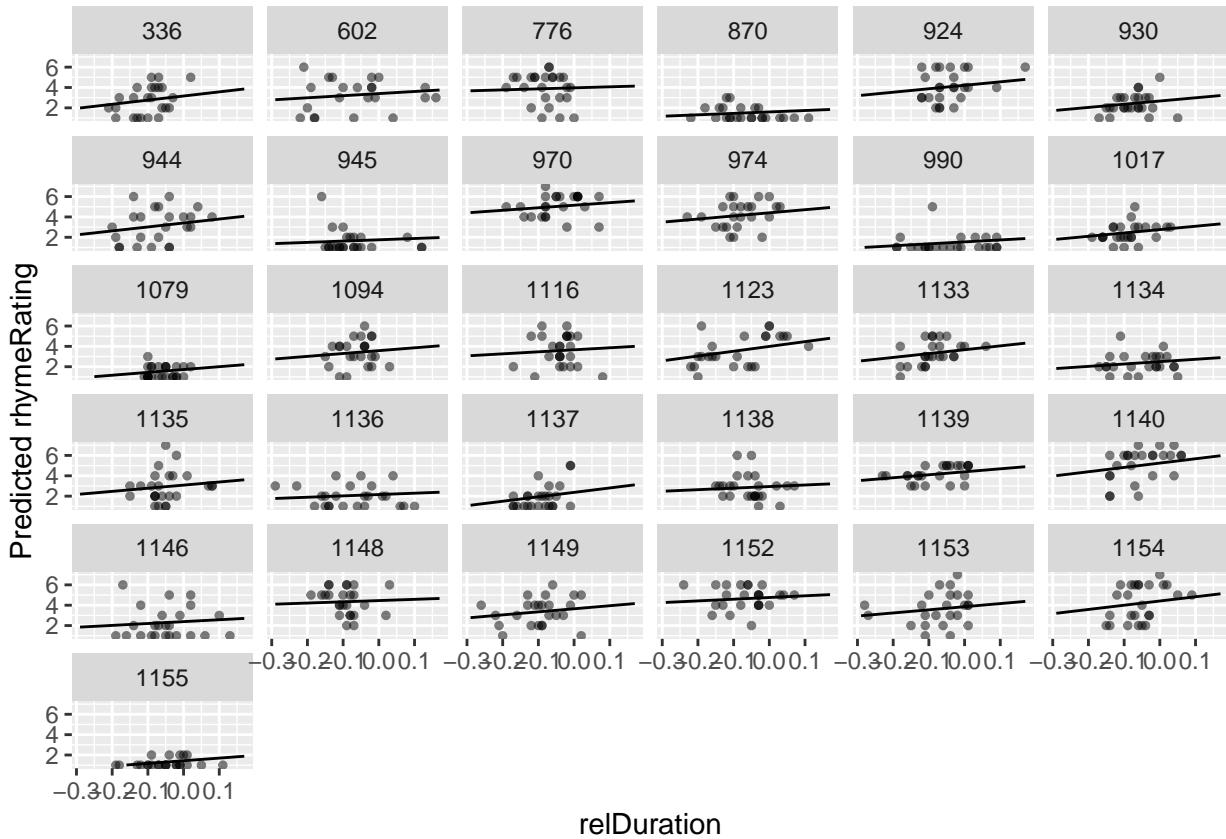
```

Plot model predictions and empirical data for each participant:

```

ggplot(aes(x=relDuration, y=pred), data=newdata) +
  geom_line(aes(x=relDuration,y=pred, group=participant)) +
  geom_point(aes(x=relDuration,y=rhymeRating),data=halfrhyme, alpha=0.5, size=1) +
  ylim(1,7) +
  xlab("relDuration") +
  ylab("Predicted rhymeRating") +
  facet_wrap(~participant)

```



## 7.4 Linear mixed models 3: Two grouping factors

Often in linguistic data, there is more than one grouping factor. Some common cases:

1. Laboratory experiments: both participants **and** items are sampled from larger populations.<sup>7</sup>
2. Corpus data: both speakers/authors **and** words are sampled from larger populations.

We will focus on the participant/item case (#1) for exposition.

Most experiments have clear by-participant and by-item variability. In RM-ANOVA analyses, this is dealt with by fitting two separate “by-participant” and “by-item” models, an awkward solution that can lower statistical power. In mixed-effects models, it is possible to account for both kinds of variability simultaneously, by including by-participant and by-item random effects.

This is a case of *crossed* random effects: multiple grouping factors, which vary independently. (As opposed to “nested” random effects, such as by-school variability and by-participant within school.) Crossed random effect structure is necessary to model even simple linguistic experiments. The facility of fitting crossed random effects in `lme4` makes it well-suited for modeling linguistic data.<sup>8</sup>

We introduce crossed random effects for the same case as in Models 1A and 1B: `acoustics ~ conditionLabel.williams`, for the `givenness` data.

<sup>7</sup>Not accounting for by-item variability was famously termed the “language-as-fixed-effect fallacy” by Clark (1973).

<sup>8</sup>In many other fields where mixed models are used (e.g. education, ecology), crossed random effects only come up in complex designs. This is why `lme4` is particularly well-suited for modeling linguistic data—in part as a result of collaboration between its primary architect, the statistician Doug Bates, and the psycholinguist R. Harald Baayen. Many mixed-model software packages assume only one grouping factor, or nested grouping factors.

### 7.4.1 Model 2A: By-participant and by-item random intercepts

The model for this case is:

$$y_i = \beta_0 + \alpha_{par,j[i]} + \alpha_{item,k[i]} + \beta_1 x_i + \epsilon_i$$

The fixed effects ( $\beta_0, \beta_1$ ) and errors ( $\epsilon_i \sim N(0, \sigma_e)$ ) are as in previous models.

The random effects are:

- $\alpha_{par,j[i]}$ : by-participant random intercept
- $\alpha_{item,k[i]}$ : by-item random intercept

The random intercepts are assumed to be normally distributed:

$$\begin{aligned} \alpha_{par,j[i]} &\sim N(0, \sigma_{par}^2), \quad j = 1, \dots, J \\ \alpha_{item,k[i]} &\sim N(0, \sigma_{item}^2), \quad j = 1, \dots, K \end{aligned}$$

where  $\sigma_{par}^2$  and  $\sigma_{item}^2$  quantify the degree of variability among participants and items in the intercept value.

**Questions:**

- What is the interpretation of each random intercept in this example? For example, what do  $\alpha_{par,2}$  and  $\alpha_{item,5}$  mean?

#### 7.4.1.1 Fitting Model 2A

To fit this model:

```
mod2a <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant) + (1|item), data=givenness)
```

The model's output is:

```
summary(mod2a)
```

```
## Linear mixed model fit by REML ['lmerMod']
## Formula: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
##           item)
## Data: givenness
##
## REML criterion at convergence: 887
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7608 -0.6026 -0.0187  0.6293  3.2908
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   participant (Intercept) 0.09266  0.3044
##   item         (Intercept) 0.04108  0.2027
##   Residual                 0.51732  0.7192
##   Number of obs: 382, groups: participant, 27; item, 16
##
```

```

## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)           -0.71655   0.08585 -8.347
## conditionLabel.williams 0.33771   0.07406  4.560
##
## Correlation of Fixed Effects:
##             (Intr)
## cndtnLbl.wl 0.001

```

**Questions:**

- Is there more variability among participants or items?
- This pattern is common in laboratory experiments. Why?

**Exercise 1**

Based on your understanding of the mixed models presented so far (1B, 1C), you should be able to answer these questions.

- In Model 2A:
  - 95% of subjects have intercepts between \_\_\_\_\_ and \_\_\_\_\_
  - 95% of items have intercepts between \_\_\_\_\_ and \_\_\_\_\_
- What is the residual error for these three models, fit with the same fixed effects (`acoustics ~ conditionLabel.williams`):
  - Model 1A (simple linear regression) (fit `mod1a`, then `summary(mod1a)`)
  - Model 1B (by-participant random intercept)
  - Model 2A (by-participant and by-word random intercept)
- Why does the pattern you see make sense?

## 7.5 Evaluating LMMs

### 7.5.1 Hypothesis testing

In non-mixed models, we see a test statistic and  $p$ -value for each term in the regression model, corresponding to the hypothesis test that the term is different from zero.

We might expect to see a test statistic and  $p$ -value for each random-effect and fixed-effect term in a mixed model, but we don't. For example, in Model 2A:

```
summary(mod2a)
```

```

## Linear mixed model fit by REML ['lmerMod']
## Formula: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
##           item)
## Data: givenness
##
## REML criterion at convergence: 887
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max

```

```

## -2.7608 -0.6026 -0.0187  0.6293  3.2908
##
## Random effects:
##   Groups      Name      Variance Std.Dev.
##   participant (Intercept) 0.09266  0.3044
##   item        (Intercept) 0.04108  0.2027
##   Residual           0.51732  0.7192
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##                               Estimate Std. Error t value
## (Intercept)            -0.71655   0.08585 -8.347
## conditionLabel.williams 0.33771   0.07406  4.560
##
## Correlation of Fixed Effects:
##                   (Intr)
## cndtnLbl.wl  0.001

```

- Under **Random effects**, there are no test statistics or  $p$ -values.
- Under **Fixed effects**, there are test statistics but no  $p$ -values.

How can we assess whether a fixed-effect or random-effect term contributes **significantly** to a mixed-effects model?

This turns out to be a contentious question, for practical and philosophical reasons—especially for fixed effects, which we usually care about more.<sup>9</sup> Some (technical) discussion is given here.

The upshot is that there are several ways to calculate  $p$ -values, which vary along an axis from “more approximate, but faster to compute” to “exact, but very time consuming to compute”.

In the opinion of one author (Morgan), any method in Sec. 7.5.3 more complex than “Use  $t$  statistic” is probably fine—and as usual, if how you calculate significance matters much to the conclusion you make, you probably shouldn’t put much stock into the effect anyway. However, it’s good to be aware of the issues, not least because other researchers (including reviewers) think it is important to use a more precise method.

### 7.5.2 Significance of a random effect term

Hypothesis testing for random effects turns out to follow a similar logic to testing the effect of multiple terms in a logistic regression model, where we used a likelihood ratio test, as in Sec. 5.2.1.

For a large enough dataset, the difference in deviance between a model with and without  $k$  random effect terms approximately follows a  $\chi^2$  distribution with  $df = k$ . This means we can use a likelihood ratio test to assess whether these terms significantly contribute.

#### Example

Does the by-item random intercept in Model 2A significantly contribute to model likelihood?

---

<sup>9</sup>The short version of “practical” is: if we want to use a  $t$  test to calculate the significance of fixed effects—as we did for regression coefficients in a non-mixed-effects linear regression—it is unclear what the degrees of freedom should be, because it’s unclear how many independent pieces of information  $n$  observations from the same group level (e.g., from a single participant) give: somewhere between 1 (because they’re all from 1 participant) and  $n - 1$  (the answer in linear regression). The short version of “philosophical” is: mixed-effects models can be thought of as Bayesian models, and in Bayesian statistics  $p$ -values are not a meaningful concept.

```

mod2a <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant) + (1|item), data=givenness)
mod2a.1 <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant), data=givenness)

anova(mod2a, mod2a.1, test='Chisq')

## refitting model(s) with ML (instead of REML)

## Data: givenness
## Models:
## mod2a.1: acoustics ~ conditionLabel.williams + (1 | participant)
## mod2a: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
## mod2a:      item)
##          Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## mod2a.1  4 899.07 914.86 -445.54    891.07
## mod2a     5 890.58 910.31 -440.29    880.58 10.49      1     0.0012 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

Thus, `acoustics` differs significantly between items ( $\chi^2(1) = 10.49, p = 0.0012$ ).

Note that this method doesn't work for testing the significance of a random effect in a model with just **one** random effect, such as Model 1B, because the models being compared are not of the same type (one is a mixed model and one isn't, so the likelihoods are not directly comparable). In this case we can instead use an “exact restricted likelihood ratio test”, implemented as `exactRLRT()` in the `RLRsim` package. This is not a common case.

### 7.5.3 Significance of fixed effects

Several options are available for calculating  $p$ -values for fixed effect terms, including the following, listed in increasing order of precision/computation time:

1. Use  $t$  statistic / Wald test
2. Likelihood ratio test
3. Satterthwaite approximation (`lmerTest`)
4. Parametric bootstrap

We show an example of how to do each, with some discussion.

#### 7.5.3.1 $t$ -statistic

The first method is simply to assume that the  $t$ -statistic for a coefficient—its estimated value divided by its standard error—follows a normal distribution (equivalent to assuming high  $df$ ), and calculate a two-sided Wald test on  $|t|$ . For example, for  $t = -2.03$ , the  $p$ -value would be:

```
2*(1-pt(abs(-2.03), df=1000))
```

```
## [1] 0.04262085
```

This is not a good way to get a  $p$ -value—it's very approximate. However, it has a useful corollary: observing that 95% of a  $t$  distribution with high  $df$  has  $|t| < 2$  gives a simple **rule of thumb**: fixed effects with  $|t| > 2$  are (roughly!) significant at the  $p < 0.05$  level.

Thus, if  $|t|$  is much larger than 2 (say  $|t| > 4$ ), the effect is highly significant and you don't need to bother using a more exact method to get a  $p$ -value.

In the literature,  $|t| > 2$  is sometimes used as a rough “is this term significant?” criterion, without comment.

### 7.5.3.2 Likelihood ratio tests

This works exactly like testing the significance of 1+ random effect terms. You fit a model with and without 1+ fixed-effect terms, and compare them with a likelihood ratio test using the `anova()` command. For large enough datasets, the difference in deviance between the two models follows a  $\chi^2$  distribution under the null hypothesis that the fixed-effect coefficients (for the omitted predictors) are zero.

For example, to test whether `conditionLabel.williams` significantly contributes to Model 2A, using an LR test:

```
mod2a <- lme4::lmer(acoustics ~ conditionLabel.williams + (1|participant) + (1|item), data=givenness)

mod2a.1 <- lme4::lmer(acoustics ~ 1 + (1|participant) + (1|item), data=givenness)

anova(mod2a, mod2a.1)

## refitting model(s) with ML (instead of REML)

## Data: givenness
## Models:
## mod2a.1: acoustics ~ 1 + (1 | participant) + (1 | item)
## mod2a: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
## mod2a:     item)
##       Df   AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## mod2a.1 4 908.69 924.48 -450.35    900.69
## mod2a    5 890.58 910.31 -440.29    880.58 20.11      1 7.313e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is a significant effect of `conditionLabel.williams` ( $\chi^2(1) = 20.1, p < 0.001$ ).

### 7.5.3.3 Satterthwaite approximation

An option which gives a reasonably good  $p$ , in reasonable computation time, is using the *Satterthwaite approximation* to compute (approximately correct)  $df$  for each fixed-effect coefficient, which is then used to conduct a two-sided  $t$  test. This method has become popular due in part to its handy implementation in the `lmerTest` package, which redefines the `lmer` command to calculate these  $df$  and  $p$ -values, and add them to the standard `lmer` output.

For example, re-running Model 2A with `lmerTest`:

```
library(lmerTest)

mod2a <- lmer(acoustics ~ conditionLabel.williams + (1|participant) + (1|item), data=givenness)
summary(mod2a)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
##     item)
##     Data: givenness
##
## REML criterion at convergence: 887
##
```

```

## Scaled residuals:
##      Min     1Q Median     3Q    Max
## -2.7608 -0.6026 -0.0187  0.6293  3.2908
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   participant (Intercept) 0.09266  0.3044
##   item         (Intercept) 0.04108  0.2027
##   Residual            0.51732  0.7192
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##                               Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)              -0.71655   0.08585 27.88067 -8.347 4.57e-09
## conditionLabel.williams  0.33771   0.07406 342.45831  4.560 7.14e-06
##
## (Intercept) *** 
## conditionLabel.williams ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cndtnLbl.wl
## cndtnLbl.wl  0.001

```

We recommend (in 2018) calculating *p*-values for fixed-effect terms in linear mixed models using the Satterthwaite approximation, as an easy and reasonably accurate option.

#### Notes:

- After loading `lmerTest`, your `lmer` models will take about twice as long to run. This is only an issue if you are fitting complicated models or analyzing large datasets.
- If you want to use the `lme4` of `lmer` after loading `lmerTest`, you use `lme4::lmer`. (For example, to run a model that doesn't take twice as long.)

#### 7.5.3.4 Parametric bootstrap

*Parametric bootstrapping* (PB) is a very accurate method for calculating p-values, which also takes a very long time. One useful implementation of PB is in the `afex` package; for more general PB computations the `bootMer()` function of `lme4` can be used.

Roughly, what PB is doing is simulating many new datasets from your model, assuming a given fixed effect coefficient is set to `zero` (but all other coefficients are kept at their fitted values). It then fits the original model to each of the new datasets, resulting in a distribution of values for the fixed-effect coefficient of interest. The proportion of these re-fitted models where the coefficient (which **should** be zero) is as large or larger than the original estimate of the coefficient's value (the first time you ran the model, on the real data) is the p-value. This is a very direct implementation of the meaning of Type 1 error ("how often would I wrongly conclude the coefficient was at least this large, if I redid the experiment many times?").

Thus, PB effectively needs to re-run your model `nsim` times, and the higher `nsim` is, the more accurate the p-values are. Here's an example for Model 2A, with `nsim = 1000`:

```

library(afex)
mod2a.pb <- mixed(acoustics ~ conditionLabel.williams + (1|participant) + (1|item), args_test = list(nsim = 1000))

## Fitting 2 (g)lmer() models:

```

```
## [..]
## Obtaining 1 p-values:
## [.]
```

This takes a couple minutes on a laptop. Results:

```
mod2a.pb
```

```
## Mixed Model Anova Table (Type 3 tests, PB-method)
##
## Model: acoustics ~ conditionLabel.williams + (1 | participant) + (1 |
## Model:      item)
## Data: givenness
##          Effect df     Chisq p.value
## 1 conditionLabel.williams  1 20.11 ***  .0010
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1
```

The  $p$ -value is very small in this model, similar to when the Satterthwaite approximation is used, but now it is only 0.001. The true  $p$ -value is probably much lower than 0.001, but 0.001 is the lowest value we can get using `nsim = 1000`. (Why?)

You can try running with `nsim = 1000000` to get a more accurate  $p$  value while you do something else for a couple hours.

#### 7.5.4 Evaluating goodness of fit

As for all regression models, we would like a measure of goodness-of-fit for linear mixed-effects models (LMMs). However, there is no simple measure for LMMs with the same properties of  $R^2$  for linear regression, including interpretability as “proportion of variation explained.” This is because in LMMs there is more than one kind of variation that can be “explained”:

- Residual variation: error in each observation
- Variation among speakers in intercept
- Variation among speakers in the slope of  $X$
- etc.

Thus, capturing “goodness of fit” for LMMs requires more than one measure. Snijders and Bosker (2011) give useful discussion.

That said, one simple and common recipe is just to take the squared correlation of an LMM’s predictions ( $\hat{y}_i$ ) with the observations ( $y_i$ ), and call this “ $R^2$ ”.

This measure means “amount of variability accounted for by the model in this dataset”, without distinguishing between fixed and random effect terms. Thus,  $R^2$  is often rather high for LMMs, simply because much variability in the data comes down to by-participant and by-item variability—even if the model would not actually have much predictive power on new data (*unseen* participants or items).

For models we have considered so far of `givenness ~ conditionLabel.williams`:

- Model 1A:

```
cor(predict(mod1a), givenness$acoustics)^2
```

```
## [1] 0.03779243
```

- Model 1B:

```
cor(predict(mod1b), givenness$acoustics)^2
```

```
## [1] 0.2192169
```

- Model 2A:

```
cor(predict(mod2a), givenness$acoustics)^2
```

```
## [1] 0.3054109
```

$R^2$  increases with the number of random-effect terms. This makes sense as a measure of goodness-of-fit for this dataset: more variability in the data is “explained”—as being due to participant or item variability, as opposed to noise. It does not make sense if  $R^2$  is viewed as a measure of predictive power: the three models have almost identical fixed-effect coefficients ( $\hat{\beta}_0, \hat{\beta}_1$ ), meaning they will make near-identical predictions for unseen data!

## 7.6 Linear mixed models 4: Multiple predictors

So far we have considered four types of linear mixed-effects models:

- Linear regression, no random effects (Model 1A)
- By-participant random intercept (Model 1B)
- By-participant random intercept and slope (Model 1C)
- By-participant and by-item random intercepts (Model 2A)

all with a single predictor.

We now turn to LMMs with multiple predictors:

- Model 3A: random intercepts only
- Model 3B: random intercepts + slopes

We fit Models 3A and 3B to the same data, to demonstrate LMMs with multiple predictors (= “multiple fixed effects”).

- Data: `givenness`
- Response: `acoustics`
- Fixed effects:
  1. `conditionLabel.williams`
  2. `npType.pron`
  3. `voice.passive`
  4. `order`
  5. `conditionLabel:npType` (interaction)

These fixed effects are one possible set one could arrive at via exploratory data analysis to analyze the data with the goal of testing whether the Williams effect exists (see dataset description). The `conditionLabel.williams` term is of primary interest.

We will denote the fixed-effect coefficients for (1)–(5) as  $\beta_{\text{conditionLabel}}$ , and so on.

### 7.6.1 Types of predictors

For fitting and interpreting mixed models, it is important to think of predictors in terms of their *level*. A predictor which describes something about participants (e.g. participant gender) is *participant-level*, a predictor which describes a property of items (e.g. word frequency) is *item-level*, and so on. Participant-level predictors are sometimes called “between-participant”, because they do not vary within participants, while a predictor that varies within participant is “within-participant” (and similarly for “between-item”, etc.). (We will not use within/between-X terminology, but you may be familiar with it, and it’s widely used in the literature.) A predictor which has a different value for every observation—that is, varies within item and participant—is called *observation-level*.

For the `givenness` data:

- Every item appears in two conditions (*Williams, contrast*) and two NP types (*pronoun, full NP*).
- Every item has **one** voice (*active or passive*)
- Thus, `voice` is an item-level predictor
- All other predictors vary within participants and items  $\implies$  observation-level predictors:
  - `conditionLabel.williams, order, npType.pron`
  - `conditionLabel:npType` (if both A and B are observation level, so is A:B—why?)
- There are no participant-level predictors.

### 7.6.2 Model 3A: Random intercepts only

This “intercepts only” model includes by-participant and by-item random intercepts, as well as the fixed effects described above. To fit the model:

```
## Model 3A: multiple predictors, by-item and by-partic random effects
## (p-values from lmerTest)
library(lmerTest)
mod3a <- lmer(acoustics ~ conditionLabel.williams*npType.pron + voice.passive + order.std + (1|participant)
```

The model’s output is:

```
library(lmerTest)
summary(mod3a)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## acoustics ~ conditionLabel.williams * npType.pron + voice.passive +
##     order.std + (1 | participant) + (1 | item)
## Data: givenness
##
## REML criterion at convergence: 758.6
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.7638 -0.6535 -0.0102  0.5750  3.5847
##
## Random effects:
##   Groups      Name      Variance Std.Dev.
##   participant (Intercept) 0.11624  0.3409
##   item        (Intercept) 0.03825  0.1956
```

```

## Residual           0.34715  0.5892
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##                               Estimate Std. Error    df t value
## (Intercept)             -0.71517   0.08728 29.38442 -8.194
## conditionLabel.williams 0.32769   0.06074 338.37005  5.395
## npType.pron              0.77529   0.06072 338.22707 12.768
## voice.passive            0.05087   0.11572 12.11117  0.440
## order.std                -0.12730   0.10819 17.79197 -1.177
## conditionLabel.williams:npType.pron 0.31916   0.12126 337.32828  2.632
##                               Pr(>|t|)
## (Intercept)             4.45e-09 ***
## conditionLabel.williams 1.29e-07 ***
## npType.pron               < 2e-16 ***
## voice.passive            0.66795
## order.std                0.25483
## conditionLabel.williams:npType.pron 0.00888 **
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cndtnL. npTyp. vc.pss ordr.s
## cndtnLbl.wl  0.001
## npType.pron  0.001 -0.015
## voice.passv  0.009  0.005 -0.023
## order.std   -0.004  0.015 -0.032  0.107
## cndtnLb.:T. -0.006 -0.004  0.003  0.003 -0.021

```

Comparing significances and directions of the fixed effects, Model 3A is mostly similar to a model without the random intercepts:

```
summary(lm(acoustics ~ conditionLabel.williams*npType.pron + voice.passive + order.std, data=givenness))
```

```

##
## Call:
## lm(formula = acoustics ~ conditionLabel.williams * npType.pron +
##     voice.passive + order.std, data = givenness)
##
## Residuals:
##      Min       1Q       Median      3Q      Max
## -1.94657 -0.49206  0.00032  0.46756  2.31921
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -0.72335   0.03582 -20.193 < 2e-16
## conditionLabel.williams 0.30626   0.07169   4.272 2.46e-05
## npType.pron              0.75743   0.07171  10.563 < 2e-16
## voice.passive            0.05866   0.07203   0.814  0.4159
## order.std                -0.19556   0.07211  -2.712  0.0070
## conditionLabel.williams:npType.pron 0.31954   0.14346   2.227  0.0265
##
## (Intercept)             ***
## conditionLabel.williams ***
## npType.pron               ***

```

```

## voice.passive
## order.std                      **
## conditionLabel.williams:npType.pron *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Residual standard error: 0.7 on 376 degrees of freedom
## Multiple R-squared:  0.2768, Adjusted R-squared:  0.2672
## F-statistic: 28.78 on 5 and 376 DF,  p-value: < 2.2e-16

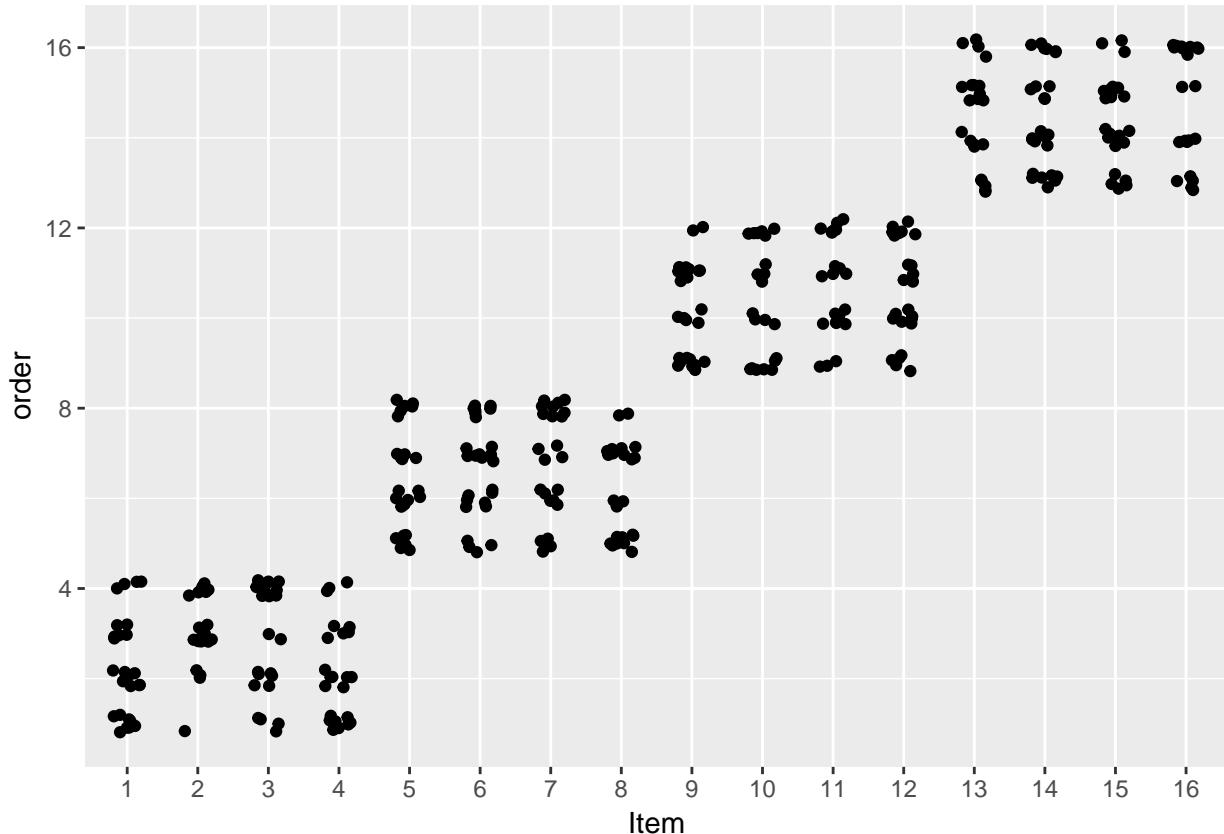
```

With one exception: the `order` effect, which has smaller effect size and is no longer significant in the mixed model. To see why, note how items were presented in the experiment:

```

ggplot(data=givenness, aes(x=as.factor(item), y=order)) +
  geom_point(position=position_jitter(w=0.2, h=0.2)) +
  xlab("Item")

```



Items were presented in four blocks—something that isn’t accounted for unless the regression model allows for by-item variability. Not accounting for such variability in the multiple linear regression model leads to a spurious `order` effect.

In this case, the random effects in the mixed model are simply accounting for information we forgot to include in the fixed-effects-only model. (We could have included “block” as a factor.) But more generally, one useful function of random effects is to account for by-participant and by-item variability **beyond sources included as predictors in the model**—either because you don’t know what these sources are (the usual case), or because you forgot to include them (this example). Doing so helps avoid spurious effects.

### Bonus: Thinking through model predictions

Thinking through mixed-model predictions is confusing at first, but very useful. Once you understand “simple” models like this one, it isn’t hard to generalize to more complex models.

A couple examples of the predictions the model makes:

1. Model prediction for an “average subject and average item”:

$$y_i = \hat{\beta}_0 + \hat{\beta}_{\text{clabel}} \cdot \text{clabel}_i + \hat{\beta}_{\text{npType}} \cdot \text{npType}_i + \\ \hat{\beta}_{\text{order}} \cdot \text{order}_i + \hat{\beta}_{\text{clabel:npType}} \cdot \text{clabel}_i \text{npType}_i$$

(Where `clabel` is an abbreviation for `conditionLabel`.)

This looks exactly like you’d expect given the model’s fixed effects formula (`clabel + npType + order + voice + clabel:npType`), except that there is no term for `voice`. This is because `voice` is an item-level predictor, so for an “average item”, there is no voice effect—the average is over both *active* and *passive* voice items.

2. Model prediction for Subject 5, item 2 (where `voice` = passive for this item):

$$y_i = (\hat{\beta}_0 + \alpha_5 + \delta_2 + 0.5 \cdot \hat{\beta}_{\text{voice}}) + \\ \hat{\beta}_{\text{clabel}} \cdot \text{label}_i + \hat{\beta}_{\text{npType}} \cdot \text{npType}_i + \hat{\beta}_{\text{order}} \cdot \text{order}_i + \\ \hat{\beta}_{\text{clabel:npType}} \cdot \text{clabel}_i \text{npType}_i$$

In this equation, the “intercept” is:

$$(\hat{\beta}_0 + \alpha_5 + \delta_2 + 0.5 \cdot \hat{\beta}_{\text{voice}})$$

this is the **predicted mean value for this participant and item**: the grand mean, with offsets for this participant, this item, and this `voice` value.

Interpretation aside: because `voice` is item-level, the by-item random intercept is not the offset from the grand mean. It is the offset of item 2 among `voice` = passive items.

### 7.6.3 Model 3B: Random intercepts and all possible random slopes

Model 3A had random intercepts (by-participant and by-item) only. We now add “all possible random slopes”. What this means is:

- By-participant random slopes for all predictors that vary within participants (i.e. not participant-level predictors)
- By-item random slopes for all predictors that vary within items (i.e. not item-level predictors)

For example, suppose that the model included a predictor for participant gender. This predictor estimates the difference between male and female participants. The effect of this predictor cannot vary within participants—each participant has only one value of gender, so it doesn’t make sense to estimate “difference between male and female for participant 3”. Thus, there can be no by-participant random slope term. More generally, there can be no by-participant random slopes for participant-level predictors.

By similar logic, there can be no by-item random slopes for item-level predictors.

Thus, the current model could include:

- By-participant random slopes for all predictors
- By-item random slopes for all predictors **except voice**.

However, for simplicity we leave out the random slope for `order`. The fixed-effect term is not significant, and we don't care about the  $\beta_{\text{order}}$  estimate anyway. (We will discuss motivation for which random slope terms to include, soon.)

Technical note: we are again using “uncorrelated” random effects, denoted by the double-pipe (||) symbol. (We will discuss what this means soon as well.)

To fit the model:

```
## Model 3B
mod3b <- lmer(acoustics ~ conditionLabel.williams*npType.pron + voice.passive + order.std +
               (1 + conditionLabel.williams*npType.pron + voice.passive || participant) +
               (1 + conditionLabel.williams*npType.pron || item), data=givenness)
```

In the model formula:

- `(1 + conditionLabel.williams*npType.pron + voice.passive || participant)` are the by-participant random effects
- `(1 + conditionLabel.williams*npType.pron || item)` are the by-item random effects.

The model output is:

```
summary(mod3b)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## acoustics ~ conditionLabel.williams * npType.pron + voice.passive +
##   order.std + (1 + conditionLabel.williams * npType.pron +
##   voice.passive || participant) + (1 + conditionLabel.williams *
##   npType.pron || item)
## Data: givenness
##
## REML criterion at convergence: 722.6
##
## Scaled residuals:
##   Min     1Q   Median     3Q    Max
## -2.97005 -0.61342 -0.00089  0.55381  2.98506
##
## Random effects:
##   Groups      Name           Variance Std.Dev.
##   participant (Intercept) 0.11589  0.3404
##   participant.1 conditionLabel.williams 0.01278  0.1131
##   participant.2 npType.pron       0.01298  0.1139
##   participant.3 voice.passive    0.08281  0.2878
##   participant.4 conditionLabel.williams:npType.pron 0.03131  0.1770
##   item        (Intercept) 0.04089  0.2022
##   item.1      conditionLabel.williams 0.00000  0.0000
##   item.2      npType.pron       0.20854  0.4567
##   item.3      conditionLabel.williams:npType.pron 0.00000  0.0000
##   Residual                0.26762  0.5173
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
```

```

##                               Estimate Std. Error      df t value
## (Intercept)                 -0.71909   0.08702 29.69336 -8.263
## conditionLabel.williams      0.30897   0.05894 19.87033  5.242
## npType.pron                  0.78377   0.12831 15.26567  6.108
## voice.passive                0.06037   0.12802 16.52472  0.472
## order.std                     -0.12414   0.10625 18.30847 -1.168
## conditionLabel.williams:npType.pron 0.31522   0.11341 21.95021  2.780
##                               Pr(>|t|)
## (Intercept)                 3.45e-09 ***
## conditionLabel.williams     4.03e-05 ***
## npType.pron                  1.85e-05 ***
## voice.passive                0.6434
## order.std                     0.2576
## conditionLabel.williams:npType.pron 0.0109 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##              (Intr) cndtL. npTyp. vc.pss ordr.s
## cndtnLbl.wl  0.002
## npType.pron  0.000 -0.005
## voice.passv  0.008  0.004 -0.009
## order.std   -0.004  0.019 -0.013  0.097
## cndtnLb.:T. -0.005  0.001  0.002  0.003 -0.019

```

Compare to the results of Model 3A:

```
summary(mod3a)
```

```

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## acoustics ~ conditionLabel.williams * npType.pron + voice.passive +
##       order.std + (1 | participant) + (1 | item)
## Data: givenness
##
## REML criterion at convergence: 758.6
##
## Scaled residuals:
##    Min     1Q   Median     3Q    Max
## -2.7638 -0.6535 -0.0102  0.5750  3.5847
##
## Random effects:
## Groups      Name        Variance Std.Dev.
## participant (Intercept) 0.11624  0.3409
## item        (Intercept) 0.03825  0.1956
## Residual            0.34715  0.5892
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##                               Estimate Std. Error      df t value
## (Intercept)                 -0.71517   0.08728 29.38442 -8.194
## conditionLabel.williams      0.32769   0.06074 338.37005  5.395
## npType.pron                  0.77529   0.06072 338.22707 12.768
## voice.passive                0.05087   0.11572 12.11117  0.440

```

```

## order.std              -0.12730   0.10819 17.79197 -1.177
## conditionLabel.williams:npType.pron  0.31916   0.12126 337.32828  2.632
##
## (Intercept)          4.45e-09 ***
## conditionLabel.williams    1.29e-07 ***
## npType.pron            < 2e-16 ***
## voice.passive          0.66795
## order.std               0.25483
## conditionLabel.williams:npType.pron  0.00888 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr) cndtL. npTyp. vc.pss ordr.s
## cndtnLbl.wl  0.001
## npType.pron  0.001 -0.015
## voice.passv  0.009  0.005 -0.023
## order.std   -0.004  0.015 -0.032  0.107
## cndtnLb.:T. -0.006 -0.004  0.003  0.003 -0.021

```

Comparing the model with random slopes (3B) to the model without random slopes (3A):

- Fixed-effect coefficient values are similar
- $t$  values for coefficients which were significant ( $|t| >> 2$ ) are still high.

This means there are solid overall effects (of `conditionLabel`, `npType`, and their interaction), after accounting for between-item and between-participant variability.

Note that the  $t$  value for `npType` is much lower in Model 3B (6.1 versus 12.8)—though still highly significant. This lower  $t$  value is OK, and in fact a good thing, as explained in Sec. 7.7.

#### 7.6.4 Assessing variability

Random slopes capture variability among participants or items in the size of an effect. Thus, we can test whether participants/items significantly differ in an effect by performing model comparison of models with and without the random slope term.

The random slope variances for `conditionLabel.williams` in Model 3B are:

- By-participant: 0.0127849
- By-item: 0

These are both very small (corresponding to  $\sigma = 0.1130703$  and 0), compared to the fixed effect coefficient for `conditionLabel.williams`—in fact, no variability among items is detected. But let's check whether there is significant by-participant and by-item variability anyway, as an example.

By participant:

```

## model comparisons to check whether conditionLabel.williams variability
## by-participant

mod3b.no.partic.slope <- update(mod3b, . ~ . - (0+conditionLabel.williams|participant))
anova(mod3b, mod3b.no.partic.slope)

## refitting model(s) with ML (instead of REML)

```

```

## Data: givenness
## Models:
## mod3b.no.partic.slope: acoustics ~ conditionLabel.williams + npType.pron + voice.passive +
## mod3b.no.partic.slope:      order.std + (1 | participant) + (0 + npType.pron | participant) +
## mod3b.no.partic.slope:      (0 + voice.passive | participant) + (0 + conditionLabel.williams:npType.p
## mod3b.no.partic.slope:      participant) + (1 | item) + (0 + conditionLabel.williams |
## mod3b.no.partic.slope:      item) + (0 + npType.pron | item) + (0 + conditionLabel.williams:npType.p
## mod3b.no.partic.slope:      item) + conditionLabel.williams:npType.pron
## mod3b: acoustics ~ conditionLabel.williams * npType.pron + voice.passive +
## mod3b:      order.std + (1 + conditionLabel.williams * npType.pron +
## mod3b:      voice.passive || participant) + (1 + conditionLabel.williams *
## mod3b:      npType.pron || item)
##                               Df   AIC   BIC logLik deviance Chisq Chi Df
## mod3b.no.partic.slope 15 735.9 795.08 -352.95    705.9
## mod3b                  16 737.8 800.92 -352.90    705.8 0.0993     1
##                               Pr(>Chisq)
## mod3b.no.partic.slope
## mod3b                  0.7527

```

Thus, there is no significant by-participant variability in the Williams effect ( $\chi^2(1) = 0.09, p = 0.75$ )

By item:

```

## by-item
mod3b.no.item.slope <- update(mod3b, . ~ . - (0+conditionLabel.williams|item))
anova(mod3b, mod3b.no.item.slope)

## refitting model(s) with ML (instead of REML)

## Data: givenness
## Models:
## mod3b.no.item.slope: acoustics ~ conditionLabel.williams + npType.pron + voice.passive +
## mod3b.no.item.slope:      order.std + (1 | participant) + (0 + conditionLabel.williams |
## mod3b.no.item.slope:      participant) + (0 + npType.pron | participant) + (0 + voice.passive |
## mod3b.no.item.slope:      participant) + (0 + conditionLabel.williams:npType.pron |
## mod3b.no.item.slope:      participant) + (1 | item) + (0 + npType.pron | item) + (0 +
## mod3b.no.item.slope:      conditionLabel.williams:npType.pron | item) + conditionLabel.williams:npTyp
## mod3b: acoustics ~ conditionLabel.williams * npType.pron + voice.passive +
## mod3b:      order.std + (1 + conditionLabel.williams * npType.pron +
## mod3b:      voice.passive || participant) + (1 + conditionLabel.williams *
## mod3b:      npType.pron || item)
##                               Df   AIC   BIC logLik deviance Chisq Chi Df
## mod3b.no.item.slope 15 735.8 794.98 -352.9    705.8
## mod3b                  16 737.8 800.92 -352.9    705.8     0     1
##                               Pr(>Chisq)
## mod3b.no.item.slope
## mod3b                  1

```

So there is also no significant by-item variability in the Williams effect ( $\chi^2(1) = 0, p = 1$ )

For the sake of this example, let's calculate the predicted Williams effect (`conditionLabel.williams` slope) for each participant anyway:

```

## examine distribution of participant clabel.wiliams coefficents:
participantRanefs <- ranef(mod3b)$participant

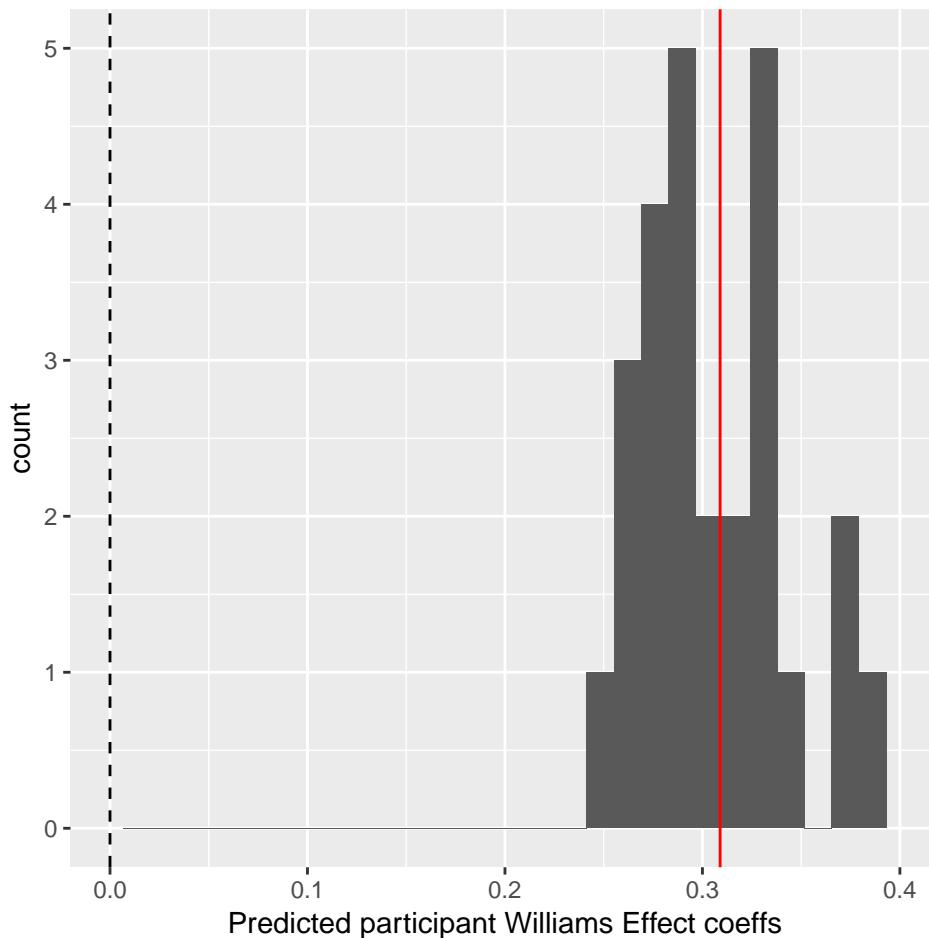
## fixed effect for conditionLabel
beta <- fixef(mod3b)[['conditionLabel.williams']]

```

```
## each participant's offset
offsets <- participantRanefs$conditionLabel.williams

participantEffects <- (beta + offsets)

ggplot(aes(x=participantEffects), data=data.frame(participantEffects=participantEffects)) +
  geom_histogram() +
  xlim(0,0.4) +
  geom_vline(aes(xintercept=0),lty=2) +
  geom_vline(aes(xintercept=beta),color='red') +
  xlab("Predicted participant Williams Effect coeffs")
```



In this plot, the red line is the fixed effect coefficient value—the overall effect, across participants—and the dotted line is at 0. Thus, there is a clear Williams effect (red line far from zero), but minor by-participant variability. Participants differ in the **magnitude** but not **direction** of the Williams effect.

## 7.7 More on random slopes

Random slopes are crucial to using mixed models, but can be confusing to use and interpret.

### 7.7.1 What does adding a random slope term do?

To see why random slopes are so important, let's consider an example where (unlike Model 3B) there is significant by-participant variability in an effect: the `halfrhyme` example used in Model 1C.

As a reminder, this model predicts `rhymeRating` with a fixed effect of `relDuration`, and a by-participant random intercept and random slope of `relDuration`.

Let Model 1D be the same model, without the random slope:

```
## Model 1D: half-rhyme data model of 1 var, with by-partic random intercept only
mod1d <- lmer(rhymeRating ~ relDuration + (1 | participant), data=halfrhyme)
```

To get a sense of what a random slope does, we compare the two models:

- Model 1D: By-subject **random intercept** (only)
- Model 1C: By-subject **random intercept, random slope of relDuration**

(We ignore by-item variability.)

The intercept-only model:

```
summary(mod1d)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: rhymeRating ~ relDuration + (1 | participant)
##   Data: halfrhyme
##
## REML criterion at convergence: 2580
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -2.4624 -0.6545 -0.1025  0.6348  3.6438
##
## Random effects:
##   Groups   Name        Variance Std.Dev.
##   participant (Intercept) 1.222    1.106
##   Residual           1.575    1.255
##   Number of obs: 756, groups: participant, 31
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  3.2918    0.2088  33.0815 15.765 < 2e-16 ***
## relDuration  2.6600    0.6703 728.5548  3.968 7.96e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr)
## relDuration 0.219
```

models `rhymeRating` as a linear function of `relDuration`, with the **same slope** for every participant. Thus, every participant's data is contributing to estimating one number:  $\beta_1$ , the slope for `relDuration`. (Setting aside estimating the intercept terms:  $\beta_0$  and random intercepts.)

The intercept + slope model:

```
summary(mod1c)

## Linear mixed model fit by REML ['lmerMod']
## Formula:
## rhymeRating ~ relDuration + ((1 | participant) + (0 + relDuration |
##     participant))
## Data: halfrhyme
##
## REML criterion at convergence: 2578.7
##
## Scaled residuals:
##    Min      1Q  Median      3Q     Max
## -2.4022 -0.6091 -0.1126  0.6138  3.5726
##
## Random effects:
## Groups       Name        Variance Std.Dev.
## participant (Intercept) 1.267    1.126
## participant.relDuration 4.482    2.117
## Residual           1.552    1.246
## Number of obs: 756, groups: participant, 31
##
## Fixed effects:
##             Estimate Std. Error t value
## (Intercept) 3.3003    0.2125 15.533
## relDuration 2.7249    0.7816  3.486
##
## Correlation of Fixed Effects:
##          (Intr)
## relDuration 0.192
```

models `rhymeRating` as a linear function of `relDuration`, with a **different slope** for every participant. Thus, every participant's data is contributing to estimating two numbers:  $\beta_1$ , and the offset of that participant's slope from  $\beta_1$ . Intuitively, this results in the model being less certain about the **overall** slope ( $\beta_1$ ).

We can see the result by comparing the `relDuration` fixed-effect rows for the two models:

Model	Estimate	Std. Error	t value
Intercept-only	2.66	0.21	3.99
Intercept + slope	2.72	<b>0.78</b>	3.49

Both models show similar estimated slopes ( $\hat{\beta}_1$ ) for `relDuration`. But the intercept + slope model has a much larger standard error for this effect, resulting in a smaller *t* value, and a less significant effect. (Recall that higher  $|t| \implies$  more significant.)

### 7.7.2 Discussion: Adding a random slope

It makes sense that the standard error of the `relDuration` fixed effect goes up when a random slope is added: the model detects significant variability among participants in the effect of `relDuration`, which makes it less sure of the **overall** effect of `relDuration`.

This is a good thing, from the perspective of not finding spurious effects. In general, participants (and items) will differ in the effect of a given predictor,  $X$ . Thus, if a by-participant (or item) random slope for  $X$  is not included, we are underestimating the uncertainty in the fixed effect for  $X$ , and can easily falsely conclude

there is a significant effect—for example, based on a subset of participants who show large effects. In general, not including a random slope for  $X$  is anti-conservative (for evaluating whether the fixed-effect coefficient of  $X$  is 0)—it **increases Type I error**.

Similar logic holds for by-item random slopes, and so on: whenever the effect of a predictor  $X$  could vary among levels of a grouping factor  $Z$ , it is anti-conservative to not include a by- $Z$  random slope for  $X$ . This issue is discussed at length by Barr et al. (2013).

On the other hand, adding a random slope for  $X$  also **increases Type II error**—that is, lowers power to detect a non-zero (fixed) effect of  $X$ . This is especially true if there is little by-participant variability in the effect of  $X$ . This issue is emphasized by Matuschek et al. (2017) and Bates et al. (2015).

These points seem to imply that adding random slopes can be both good (lower Type I error) and bad (higher Type II error). What practical advice can be given on when to include a given random slope term? The more general issue is: how do we decide on a random effect structure? This usually means, what random slope terms (and correlation terms—see below) should we include in our model?

This is a **model selection** problem (with respect to random effect terms)—and as we saw when discussing model selection for fixed effects in the context of multiple linear regression (Sec. 3.5), there is no “best” answer. What model selection procedure you use depends on the goals of your analysis. There are two broad perspectives:

- **Perspective 1** (Barr et al. (2013) advice):<sup>10</sup> include random slope(s) for any fixed effect coefficient you care about
  - Ideal: “Maximal” random effect structure, meaning all possible random effect terms (modulo issues with model convergence)
  - Guards against *Type 1 errors*
- **Perspective 2** (Bates et al. (2014) advice): only include random slopes that contribute significantly to model likelihood, using a likelihood ratio test
  - (a.k.a. random effect terms “justified by the data”)
  - Guards against *Type II errors*

Type I and Type II error always trade off in model selection. Thus, there is no “correct” answer—it depends on whether you care more about Type I and Type II error. No consensus exists on how to arrive at a random-effect structure, and this is a major issue for users fitting these models in practice, since often the “maximal” random effect structure is too complex for the data, and leads to fitting issues.

Still, a few guidelines can be given:

1. It is crucial to **consider** random slope terms for all fixed effects of interest for your research questions—either by including them in the model (Perspective 1), or testing whether they should be added (Perspective 2). Otherwise, you run the risk of seriously inflated  $p$ -values.
2. If a random slope for an interaction is included, it is crucial to include the corresponding random slopes for all subsets of the interaction—for the same reason that you must include all subset of an interaction as fixed effects. (For example, a by-participant  $X:Y$  random slope  $\implies$  by-participant  $X$  and  $Y$  random slopes must be included.)
3. It is not as important to consider random slope terms for fixed effects not of interest, such as those included as controls.
4. It is not as important to consider random slope terms for fixed effects which do not reach significance in an intercepts-only model. Adding these random slopes will often have little effect on estimates of other fixed effects.

---

<sup>10</sup>See also Schielzeth and Forstmeier (2009), who had earlier emphasized the importance of random slopes for analyzing ecological data.

5. Only so many random slope terms can be properly estimated, given the size and structure of your dataset. Thus, it's important to prioritize random slopes you **must** consider (#1 and #2).

#### Extended exercise: LMM with random slopes

See the Appendix, Sec. 7.13.

## 7.8 Random effect correlations

In examples so far, we have always used “uncorrelated” random effects, written using the `||` notation in `lme4`. For example, in Model 1C, the random effect term is `(1 + relDuration.std || participant)`.

Uncorrelated random effects assume that there is no relationship between different random-effect terms for the same grouping factor. In Model 1C, it is assumed that there is no relationship between

1. a participant's offset from the overall intercept (random intercept)
2. a participant's offset from the overall slope of `relDuration` (random slope)

If there were a relationship between (1) and (2), it would take the form of a positive or negative **correlation**—hence the name “uncorrelated random effects”.

#### Questions:

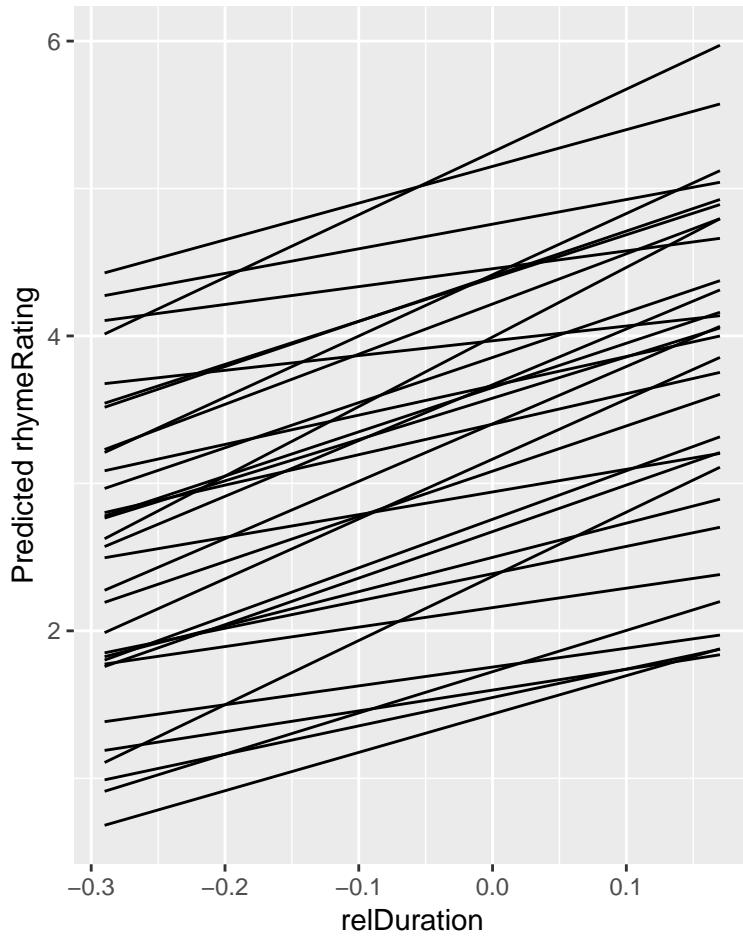
- What would it mean, intuitively, if there were a strong positive relationship (“positive correlation”) between (1) and (2)? (“Participants who \_\_\_\_\_ also have a higher \_\_\_\_\_.”)

To see whether this assumption is realistic, we can examine the prediction of Model 1C for each subject:

```
## model predictions by-subject for Model 1C:
newdata <- data.frame(
  expand.grid(relDuration=seq(
    min(halfrhyme$relDuration), max(halfrhyme$relDuration), by=0.01),
    participant=unique(halfrhyme$participant)))

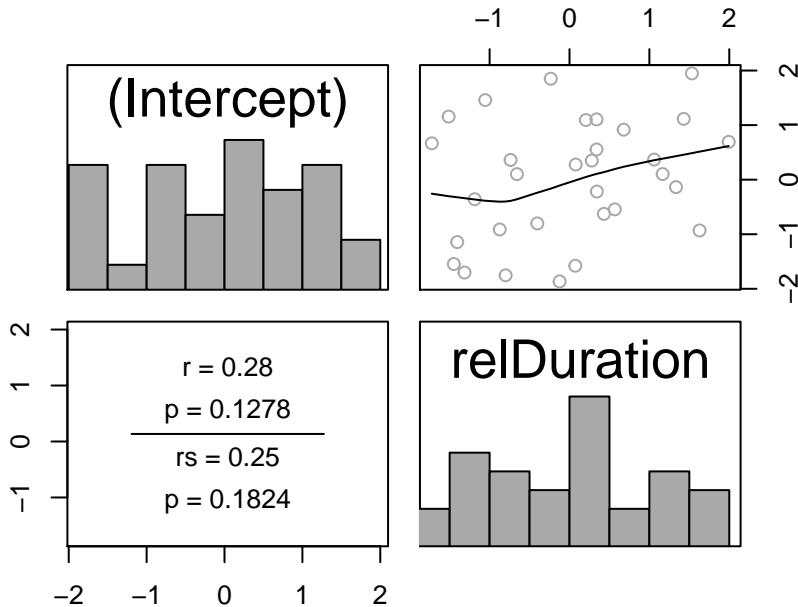
## get the predicted value for each case
newdata$pred <- predict(mod1c, newdata=newdata)

## plot the model's prediction for each participant:
ggplot(aes(x=relDuration, y=pred), data=newdata) +
  geom_line(aes(x=relDuration, y=pred, group=participant)) +
  xlab("relDuration") +
  ylab("Predicted rhymeRating")
```



A relationship between the height of a line (= random intercept) and its slope (= random slope) doesn't jump out, but we can check this more carefully by plotting the relationship between each participant's offsets for the intercept and the `relDuration` slope:

```
mod1c.ranefs <- ranef(mod1c)$participant  
pairscor.fnc(mod1c.ranefs)
```



This plot suggests there may be a weak positive correlation ( $r = 0.28, p = 0.13$ ): participants who rate half-rhymes better (higher `rhymeRating`) may also have a larger effect of the acoustic cue (`relDuration`) on their rating. Alternatively, this may be a spurious correlation, not significantly different from no correlation. We can test whether there's evidence for a "real" correlation between participants' slopes and intercepts by including this term in the mixed model.

### 7.8.1 Model 1E: Correlated random slope & intercept

The regression model is now:

$$y_i = \beta_0 + \alpha_{j[i]} + (\beta_1 + \gamma_{j[i]}) x_i + \epsilon_i, \quad i = 1, \dots, n$$

Where the random effects are:

- $\alpha_{j[i]}$ : random intercept coefficient
- $\gamma_{j[i]}$ : **random slope** coefficient

and the random intercept and slope follow a multivariate normal distribution:

$$\begin{pmatrix} \alpha_j \\ \gamma_j \end{pmatrix} \sim N \left( \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \begin{pmatrix} \sigma_{s,0} & \rho \\ \rho & \sigma_{s,1} \end{pmatrix} \right), \quad j = 1, \dots, J$$

where:

- $\sigma_{s,0}$  is the amount of by-subject variability in the intercept
- $\sigma_{s,1}$  is the amount of by-subject variability in the slope of `relDuration`
- $\rho$  is the **correlation between random intercept and random slope**, across participants.

The parameters estimated in this model are:

1. Two fixed-effect coefficients:

- $\beta_0$
- $\beta_1$

2. Four variance components:

- Random intercept:  $\sigma_{s,0}^2$
- Random slope (of `relDuration`):  $\sigma_{s,1}^2$
- Correlation between them:  $\rho$
- Residual error variance:  $\sigma_e^2$

$\rho$  is the only new parameter in this model, compared to Model 1C. The only difference between the two models is:

- Model 1C assumes that  $\rho = 0$
- Model 1E fits  $\rho \in (-1, 1)$

In R, the “single pipe” notation (`|`) is used for correlated random effects.

To fit Model 1E:

```
mod1e <- lmer(rhymeRating ~ relDuration + (1 + relDuration | participant), data=halfrhyme)
```

Note the syntax:

- Correlated random effects: `(1 + A + B | participant)`
- Uncorrelated random effects: `(1 + A + B || participant)`

Recall that the formula for Model 1C was:

```
mod1c <- lmer(rhymeRating ~ relDuration + (1 + relDuration || participant), data=halfrhyme)
```

The new model’s output is:

```
summary(mod1e)
```

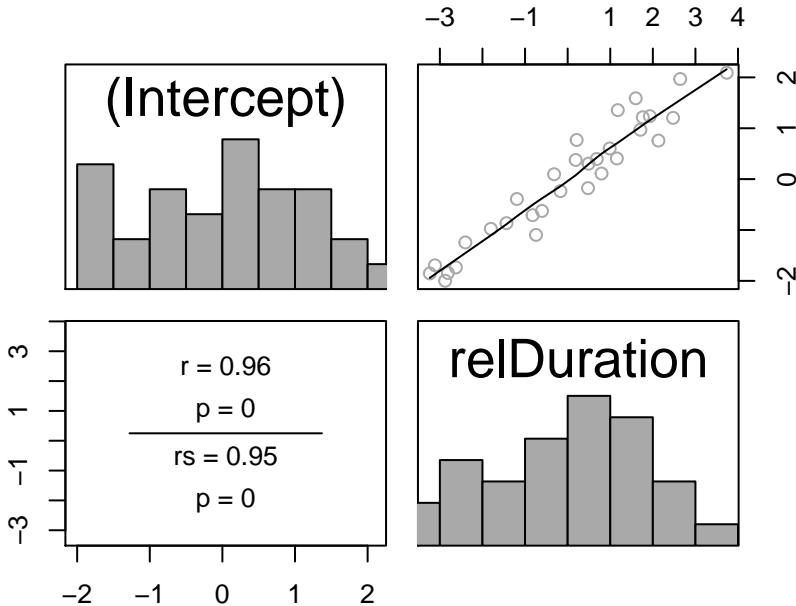
```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula: rhymeRating ~ relDuration + (1 + relDuration | participant)  
##   Data: halfrhyme  
##  
## REML criterion at convergence: 2571.9  
##  
## Scaled residuals:  
##     Min      1Q  Median      3Q     Max  
## -2.4723 -0.6129 -0.1562  0.5897  3.4874  
##  
## Random effects:  
##   Groups      Name        Variance Std.Dev. Corr  
##   participant (Intercept) 1.496    1.223  
##                  relDuration 5.728    2.393    0.82  
##   Residual           1.548    1.244  
## Number of obs: 756, groups: participant, 31  
##  
## Fixed effects:  
##             Estimate Std. Error      df t value Pr(>|t|)  
## (Intercept)  3.3068    0.2291 30.4810 14.433 3.62e-15 ***  
## relDuration  2.8161    0.8017 24.8569  3.513  0.00172 **  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##
```

```
## Correlation of Fixed Effects:
##           (Intr)
## relDuration 0.591
```

We see a positive correlation term under Random effects/Corr ( $\rho = 0.82$ )—the direction guessed from the scatterplot of the Model 1C random effects.

We can repeat that plot using the random effects from Model 1E, to see the relationship predicted by this model between participant offsets for intercept and `relDuration` slope:

```
mod1e.ranefs <- ranef(mod1e)$participant
pairscor.fnc(mod1e.ranefs)
```



A strong relationship is predicted—compare to the plot for Model 1C. It seems that the weak positive correlation observed in that plot hinted at a much stronger relationship that is detected by including a correlation term. To evaluate whether this correlation is significantly different from zero, we can do a model comparison:

```
anova(mod1c, mod1e)
```

```
## refitting model(s) with ML (instead of REML)

## Data: halfrhyme
## Models:
##   mod1c: rhymeRating ~ relDuration + ((1 | participant) + (0 + relDuration |
##     participant))
##   mod1e: rhymeRating ~ relDuration + (1 + relDuration | participant)
##             Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
##   mod1c  5 2588.7 2611.8 -1289.3    2578.7
##   mod1e  6 2583.7 2611.5 -1285.8    2571.7 6.9728      1  0.008276 **
##   ---
##   Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There is a significant improvement in model likelihood ( $\chi^2(1) = 7.0, p = 0.008$ ), meaning there is a significant positive correlation between participants' random intercepts and (`relDuration`) slopes.

Despite the correlation term, the fixed effects for the two models (with and without this term) are very similar:

```

## model 1C (no correlation)
summary(mod1c)$coefficients

##             Estimate Std. Error   t value
## (Intercept) 3.300261  0.2124740 15.532541
## relDuration 2.724922  0.7816163  3.486265

## model 1E (correlation)
summary(mod1e)$coefficients

##             Estimate Std. Error      df   t value   Pr(>|t|)
## (Intercept) 3.306768  0.2291117 30.48098 14.432996 3.624195e-15
## relDuration 2.816111  0.8016868 24.85693  3.512732 1.720269e-03

```

in terms of both coefficient estimates and standard errors.

The variance components differ somewhat:  $\rho > 0$  for Model 1E (obviously), and in addition the `relDuration` random slope is larger.

### 7.8.2 Dicussion: Adding a correlation

Comparing Model 1C to Model 1E: the **fixed-effect** coefficients are very similar, while the **random effects** change to some extent. The model with a correlation term is slightly “better” in terms of likelihood (significant increase), as a result of change in the random effects.

Empirically, this often happens when random-effect correlation terms are added to a model, at least when all predictors have been centered and are orthogonal: adding correlation terms can lead to a significantly better model fit, but has little effect on the fixed-effect estimates, which are usually what we are interested in.

This observation turns out to matter a lot in practice, by the following logic. Including (or at least considering) random **slope** terms is very important, for reasons described above. However, “maximal random effect structure” (Barr et al., 2013)—meaning all possible random slopes, and all possible correlations between random effect terms—often leads to a model that will not converge (or does so very slowly), because the number of terms to be estimated grows quadratically with the number of fixed-effect predictors included: ( $\frac{k(k+1)}{2}$  random effect terms for  $k$  predictors). For example, “maximal” random effect structure for five predictors gives 16 terms to be estimated: 1 intercept, 5 slopes, and 10 correlations. Usually, you do not have enough data to fit all these parameters—which are mostly correlation terms.<sup>11</sup> Researchers following the advice to construct “maximal” models typically use correlated random-effect structure, as this is R’s default (using `(1+X+Y+...|participant)` syntax). They often find that in practice these models do not converge, or have unrealistic correlation parameter estimates (near 1 or -1), and assume that the problem has to do with adding random **slopes**. In our experience, this is usually not the case: the model is probably too complex to support the many **correlations** between random-effect terms, which are less important than random slopes. Even if the maximal model does fit (and doesn’t have unrealistic correlation estimates), it may do so extremely slowly, hindering the data analysis process which inevitably involves fitting several models.

To deal with these issues, it is often useful to **first try models with uncorrelated random effects**—including slopes, but no correlations. You can then add in correlations as needed (e.g. significantly increases model likelihood), or fit a final “full” model with all random slopes if your data supports it. In practice, models with all possible random slopes but without correlation terms usually **do** converge, and fit fairly quickly. (For example, a model with five predictors would have just 6 random effect terms in the “maximal” model without correlations.)

<sup>11</sup>This is one motivation for Bayesian mixed models: see Vasishth and Nicenboim (2016).

Note that this is all the personal opinion of one author (Morgan). Although these issues sound esoteric, modeling issues (non-convergence or degenerate random effects: correlation terms near 1 or -1) due to complex random-effects structure is one of the most frequent issues that researchers have with using mixed models in practice for linguistic data.

## 7.9 Model criticism for linear mixed models

So far, we have neglected performing model criticism when fitting linear mixed models.

However, model criticism is just as important for linear mixed models as for (non-mixed) linear regressions, we just don't go into it in detail here because many aspects of model criticism for LMMs are similar to the linear regression case.

- Checking for outliers (in predictors, response)
- Residual plots: model predictions versus residuals, QQ plot, fixed-effect predictors versus residuals.

A new type of model criticism involves **random effects**: each random-effect term is assumed to be normally distributed, and whether this assumption is (roughly) correct should be checked.

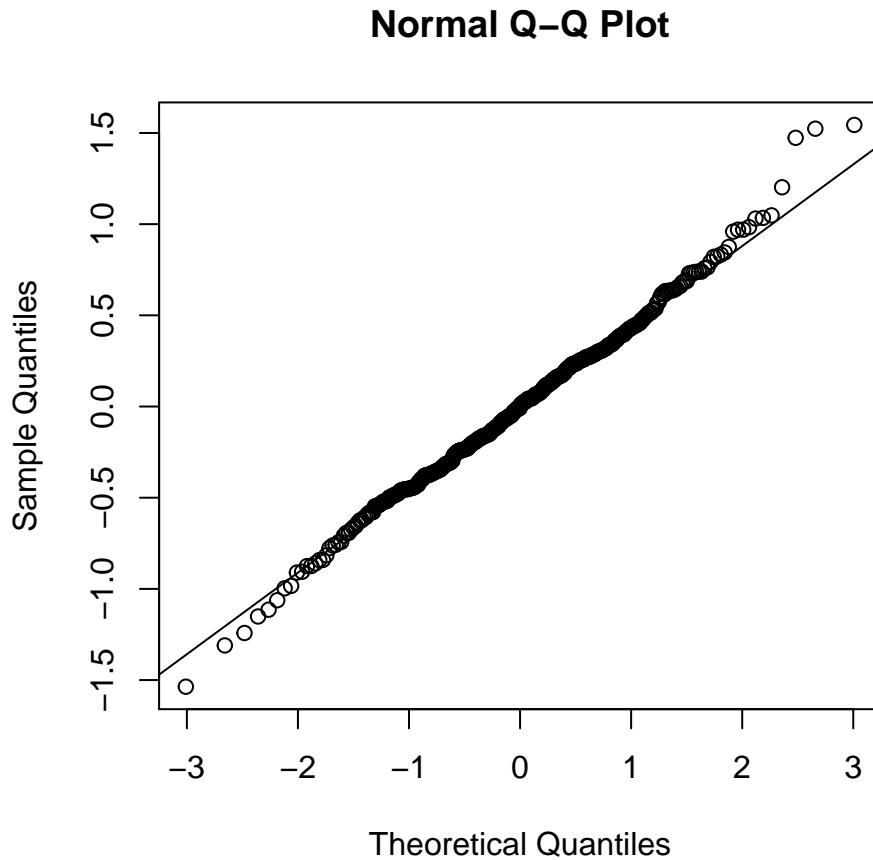
### 7.9.1 Model 3B: Residual plots

The logic of checking residual plots for an LMM is similar to linear regressions. For example, for normality of residuals: the model assumes that the residuals are normally distributed:

$$\epsilon_i \sim N(0, \sigma_e^2)$$

If this is true for an LMM, then the *Pearson residuals* of the model are normally distributed (calculated in R using `residuals(myMixedModel)`). So we can check a QQ plot to see if the residuals are not normally distributed (which would mean the assumption is false):

```
## QQ plot for Model 3B
qqnorm(residuals(mod3b))
qqline(residuals(mod3b))
```



The distribution of (Pearson) residuals looks normal, except for some points which should be flagged as potential outliers.

(It turns out that removing these outliers **strengthens** all “significant” effects—those with  $t > 2$ .)

We can also plot residuals versus fitted values (`fitted(mod3b)`) or predictor values—similar diagnostics to the same plots for linear regressions.

### 7.9.2 Model 3B: Random effect distribution

A key assumption of mixed models is that random effects are drawn from a particular distribution. For our purposes, it is always assumed that the random effects are drawn from a **normal** distribution, with mean 0 and some variance.

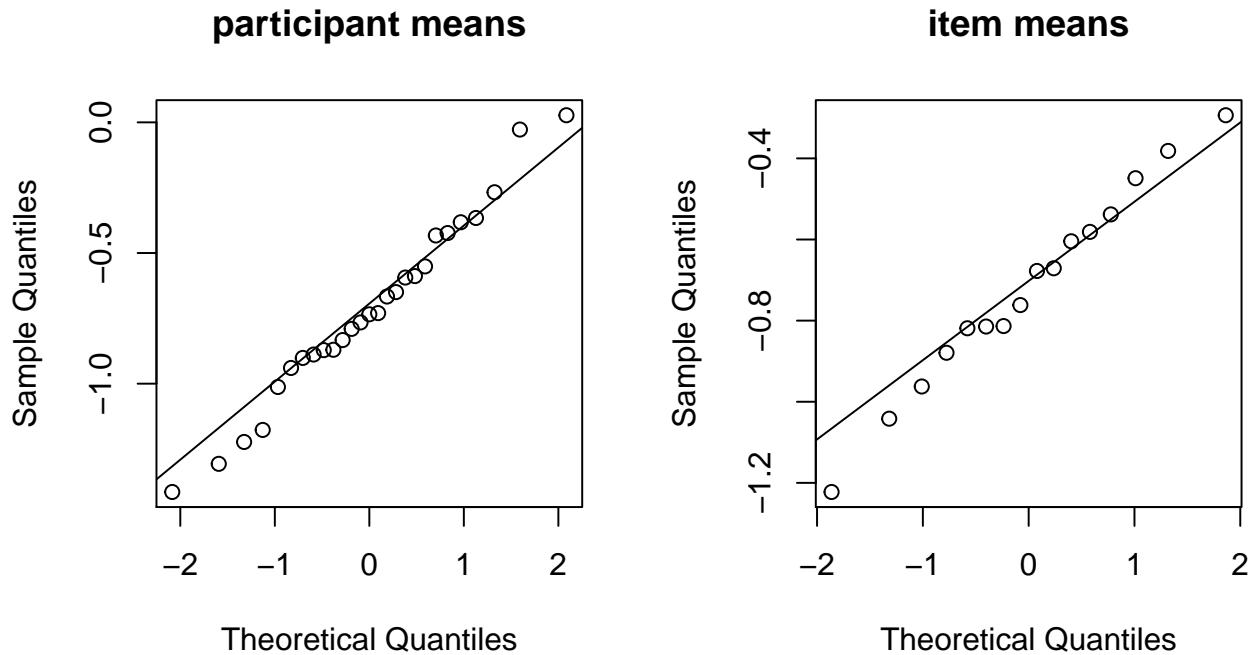
We can check this assumption by examining the distribution of either the empirical data (empirical participant means or item means), or the actual random-effect estimates (the BLUPs). Like checking residual plots, checking normality of random effects has benefits besides making sure your model is OK—it also often shows unusual participants or items, who should be investigated further.

To examine empirical participant and item means for `acoustics` for the `givenness` data:

```
givennessByPartic <- givenness %>% group_by(participant) %>% summarise(mean=mean(acoustics))
givennessByItem <- givenness %>% group_by(item) %>% summarise(mean=mean(acoustics))

par(mfrow=c(1,2))
qqnorm(givennessByPartic$mean, main = "participant means")
qqline(givennessByPartic$mean)
```

```
qqnorm(givennessByItem$mean, main = "item means")
qqline(givennessByItem$mean)
```



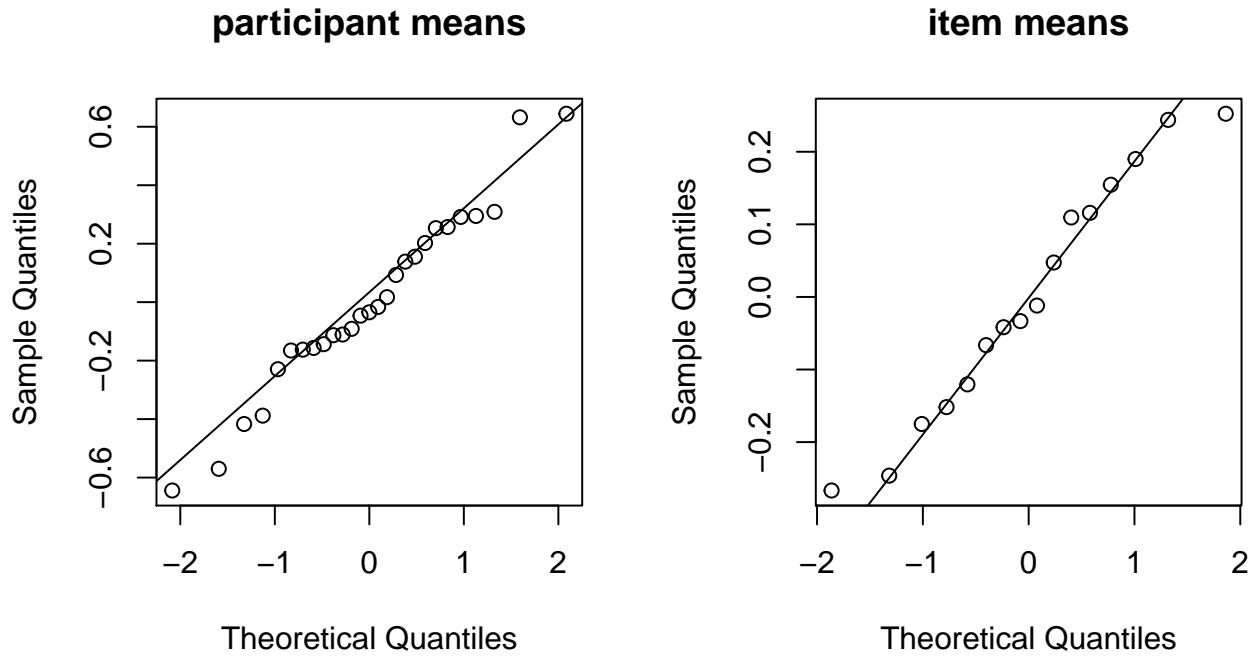
These distributions look basically normal, possibly with a weird participant or two (left plot).

We can also examine Q-Q plots for the by-participant and by-item random intercepts:

```
par(mfrow=c(1,2))

givennessByParticRanef <- ranef(mod3b)$participant[['(Intercept)']]
qqnorm(givennessByParticRanef, main = "participant means")
qqline(givennessByParticRanef)

givennessByItemRanef <- ranef(mod3b)$item[['(Intercept)']]
qqnorm(givennessByItemRanef, main = "item means")
qqline(givennessByItemRanef)
```



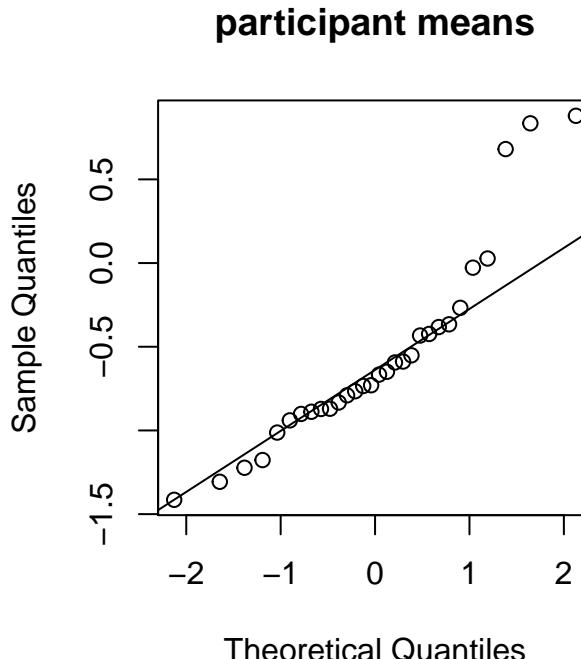
Again, there do not seem to be any particularly odd participants or items, though a couple participants may warrant a closer look.

**Detour:** What would serious non-normality of random effects look like? As an example, let's add three weird participants to dataset, who have **positive** mean value of acoustics (opposite of other participants), and examine a Q-Q plot for participant means:

```
par(mfrow = c(1,2))
# find participant summary stats to generate weird participant
n_weird <- 3
n_obs <- floor(nrow(givenness)/length(unique(givenness$participant))) # avg number of observations
mn_acoustics <- mean(givenness$acoustics)
sd_acoustics <- sd(givenness$acoustics)
# generate weird participants
set.seed(2903)
acoustics_w <- rnorm(n_weird * n_obs, mean = -mn_acoustics, sd = sd_acoustics)
participant_w <- rep(paste0("wp_", seq.int(n_weird)), each = n_obs)
# place into data frame
df_w <- data.frame(acoustics = acoustics_w, participant = participant_w)
# merge with givenness data
givenness_w <- rbind(dplyr::select(givenness, acoustics, participant), df_w)

givennessByPartic_w <- givenness_w %>% group_by(participant) %>% summarise(mean=mean(acoustics))

## make particiaption means plot
qqnorm(givennessByPartic_w$mean, main = "participant means")
qqline(givennessByPartic_w$mean)
```



There are now three outlier participants, visible at the top right of the plot. If a mixed model is fitted to this data, the estimated by-participant random intercept variance ( $\hat{\sigma}_{s,0}^2$ ) will be too high. (Because the model assumes random effects are normally distributed, the only way to account for a few outlier participants or items is to increase the random effect variance.)

## 7.10 Random slopes for factors

Technical issues, related to R's implementation of factors, come up if you want to use factors as predictors in models with uncorrelated random effects. This is an important issue in practice if you're building models with uncorrelated random effects. We advise skipping this section until this issue actually comes up in practice for you.

### 7.10.1 Model with random-effect correlations

We have so far only used factors with two levels as predictors in mixed-effects models, which we've handled by coding the factor as a single numeric variable. We have not considered factors not converted to numeric predictors, including factors with  $> 2$  levels. Using such factors in mixed models can be tricky, especially if you want to use uncorrelated random effects.

As an example, we will use the `halfrhyme` data. To load the data and code contrasts:

```
## go back to the full dataset -- all conditionLabel values
halfrhyme <- halfrhyme.orig

halfrhyme <- mutate(halfrhyme,
  conditionLabel =
    factor(conditionLabel, levels = c("bad", "voice", "good")))
contrasts(halfrhyme$conditionLabel) <- contr.helmert(3)
```

Our example models rhyme rating as a function of experimental condition:

- Response: `rhymeRating`

- Fixed effect: `conditionLabel` (single predictor)
  - Three levels: *bad, voice, good*
  - Coded using two Helmert contrasts.
- Random effects: by-participant intercept and slope for `conditionLabel`

**Questions:**

- What is the interpretation of each contrast?

Suppose we fit the model with “maximal” random effects:

```
modEx2 <- lmer(rhymeRating ~ conditionLabel + (1 + conditionLabel | participant), data = halfrhyme)
summary(modEx2)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: rhymeRating ~ conditionLabel + (1 + conditionLabel | participant)
##   Data: halfrhyme
##
## REML criterion at convergence: 3826.5
##
## Scaled residuals:
##   Min     1Q Median     3Q    Max
## -4.6442 -0.4262 -0.0567  0.5333  3.8688
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   participant (Intercept) 0.26067  0.5106
##             conditionLabel1 0.20112  0.4485   0.91
##             conditionLabel2 0.05675  0.2382  -0.73 -0.55
##   Residual           1.25335  1.1195
## Number of obs: 1205, groups: participant, 31
##
## Fixed effects:
##   Estimate Std. Error       df t value Pr(>|t|)
## (Intercept)  3.63504  0.09918 30.04544 36.652 < 2e-16 ***
## conditionLabel1 0.89985  0.09086 29.89488  9.904 5.94e-11 ***
## conditionLabel2 1.42441  0.05166 30.13975 27.573 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cndtL1
## conditnLb11  0.677
## conditnLb12 -0.511 -0.335
```

The fixed-effect and random-effect table rows for `conditionLabel1` and `conditionLabel2` describe the slopes for the two contrasts of `conditionLabel`:

- Fixed effects: “overall” slope for each contrast
- Random effects: degree of by-participant variability in the slope for each contrast (under `Variance` and `Std. Dev`)

The correlation terms capture how much different random effect terms are correlated, across participants. For example, speakers who give a higher `rhymeRating` on average (the random intercept) also tend to show a larger rating difference between *bad* and *voice* rhymes (the first contrast).

Although there don't seem to be any issues with fitting this model—it converges, and no random effect correlations are too high—often fitting maximal models with multi-level factors *does* lead to such issues, particularly if a non-orthogonal contrast coding scheme is used (such as dummy coding, R's default).

### 7.10.2 Models without random-effect correlations

If we instead try to fit a model with uncorrelated random effects, we get a warning from R:

```
modEx3 <- lmer(rhymeRating ~ conditionLabel + (1 + conditionLabel||participant), data=halfrhyme)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly uniden
## - Rescale variables?

summary(modEx3)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula:
## rhymeRating ~ conditionLabel + (1 + conditionLabel || participant)
##   Data: halfrhyme
##
## REML criterion at convergence: 3826.5
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.6442 -0.4262 -0.0567  0.5333  3.8688
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   participant (Intercept) 8.833e-05 0.009398
##   participant.conditionLabelbad 1.623e-01 0.402815
##           conditionLabelvoice 1.230e+00 1.109149  0.66
##           conditionLabelgood  1.320e-01 0.363301 -0.23  0.37
##   Residual                1.253e+00 1.119530
## Number of obs: 1205, groups: participant, 31
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept) 3.63504   0.09918 30.04544 36.652 < 2e-16 ***
## conditionLabel1 0.89985   0.09086 29.89489  9.904 5.94e-11 ***
## conditionLabel2 1.42441   0.05166 30.13975 27.573 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cndtL1
## conditnLb11  0.677
## conditnLb12 -0.511 -0.335
## convergence code: 0
## Model is nearly unidentifiable: large eigenvalue ratio
## - Rescale variables?
```

which tells us something is wrong, and we should not trust this model. A “nearly unidentifiable” model usually means “you've included variables that are not independent”.

In this case, the issue is with the random effects. Rather than the random-effect table showing one row for each of the two `conditionLabel` contrasts, it contains one row per *level* of `conditionLabel`. R is incorrectly interpreting the notation `(1 + conditionLabel || participant)` as 3 independent levels, rather than 2 contrasts.<sup>12</sup>

This issue will arise whenever a factor is used in an uncorrelated random effect formula (one that contains `||`, or that uses `(0+X|group)` notation). To include factors in a model with uncorrelated random effects, we must first turn them into numeric variables—that is, the contrasts. (This is what R usually does under the hood, but not in this case.) We manually make a numeric variable for each contrast:

```
halfrhyme$clabel.c1 <- model.matrix(~conditionLabel, halfrhyme)[,2]
halfrhyme$clabel.c2 <- model.matrix(~conditionLabel, halfrhyme)[,3]
```

then include these variables in the random-effect formula, rather than the actual factor:

```
modEx4 <- lmer(rhymeRating ~ conditionLabel + (1 + clabel.c1 + clabel.c2 || participant), data=halfrhyme)
summary(modEx4)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: rhymeRating ~ conditionLabel + (1 + clabel.c1 + clabel.c2 || participant)
## Data: halfrhyme
##
## REML criterion at convergence: 3860
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.2957 -0.5188 -0.0262  0.5393  3.8347
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   participant (Intercept) 0.30396  0.5513
##   participant.1 clabel.c1  0.23903  0.4889
##   participant.2 clabel.c2  0.06519  0.2553
##   Residual            1.24808  1.1172
## Number of obs: 1205, groups: participant, 31
##
## Fixed effects:
##             Estimate Std. Error       df t value Pr(>|t|)
## (Intercept) 3.63410  0.10596 30.02772 34.297 < 2e-16 ***
## conditionLabel1 0.90151  0.09733 30.01033  9.262 2.63e-10 ***
## conditionLabel2 1.42488  0.05420 30.28367  26.289 < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cndtL1
## conditnLbl1 -0.061
## conditnLbl2  0.043  0.059
```

This model has no convergence issues, and shows the expected random effect structure: one row for each of the `conditionLabel` contrast random slopes, and no correlations between random-effect terms.

Comparing the fixed effects of this “maximal model with uncorrelated random effects” to the actual “maximal

---

<sup>12</sup>This is a bug, but one that follows from the way uncorrelated random effects are treated in `lmer` and how R deals with factors in model formulas without intercept terms.

model” (model `modEx2` above), we see they are very similar—modulo the lack of correlations.<sup>13</sup>

Summary:

- To include a random slope for any factor (even if just 2 levels) in a model with uncorrelated random effects, you need to code as numeric variable(s).
- For a factor with 2 levels, you can just use a centered version (e.g. `arm::rescale`), which is essentially sum coding.
- For factors with more levels (or to use dummy coding for a two-level factor), you manually extract the contrasts to make numeric variables.

## 7.11 Other readings

Mixed-effects models are widely used in behavioral and social sciences. Many full-length treatments and shorter introductions are available, including:

- Books
  - About half of Gelman and Hill (2007) is focused on mixed models, called “hierarchical” (different name, same models).
  - Textbooks on statistics for linguists, such as Baayen (2008), Levy (2012), Johnson (2008) have mixed model sections.
  - Speelman et al. (2018) is an edited volume of case studies which apply mixed-effects models to linguistic data.
- Shorter
  - Bodo Winter’s introductory tutorials on linear mixed models
  - First two lectures of Martijn Wieling’s “advanced regression for linguists” course.
- Troubleshooting/specific questions:
  - Ben Bolker’s FAQ for mixed models, largely focused on R.
  - The `r-sig-mixed-models` mailing list. (Search the archives)

## 7.12 Appendix: Extra examples

### 7.12.1 Predicting confidence intervals by simulation

To get 95% CIs of our Model 1B:

```
## a simple recipe for simulating 95% confidence intervals over model predictions

## set up dataframe to predict for (every participant, every value of the predictor)
newdata <- data.frame(expand.grid(conditionLabel=unique(givenness$conditionLabel),
                                    participant=unique(givenness$participant)))

## (simulate 10k times from the model, for newdata)
preds <- simulate(mod1b.1, newdata=newdata, nsim=10000)
```

---

<sup>13</sup>The random effect variances are slightly larger in the “uncorrelated” model, indicating that the amount of variability in the intercept and slopes is being over-estimated—some of this variability is actually shared between random-effect terms, as captured by the correlation terms.

```
## lower and upper 95% CIs
newdata$lower <- apply(preds, 1, function(x){quantile(x,0.025)})
newdata$upper <- apply(preds, 1, function(x){quantile(x,0.975)})
newdata$pred <- apply(preds, 1, function(x){quantile(x,0.50)})

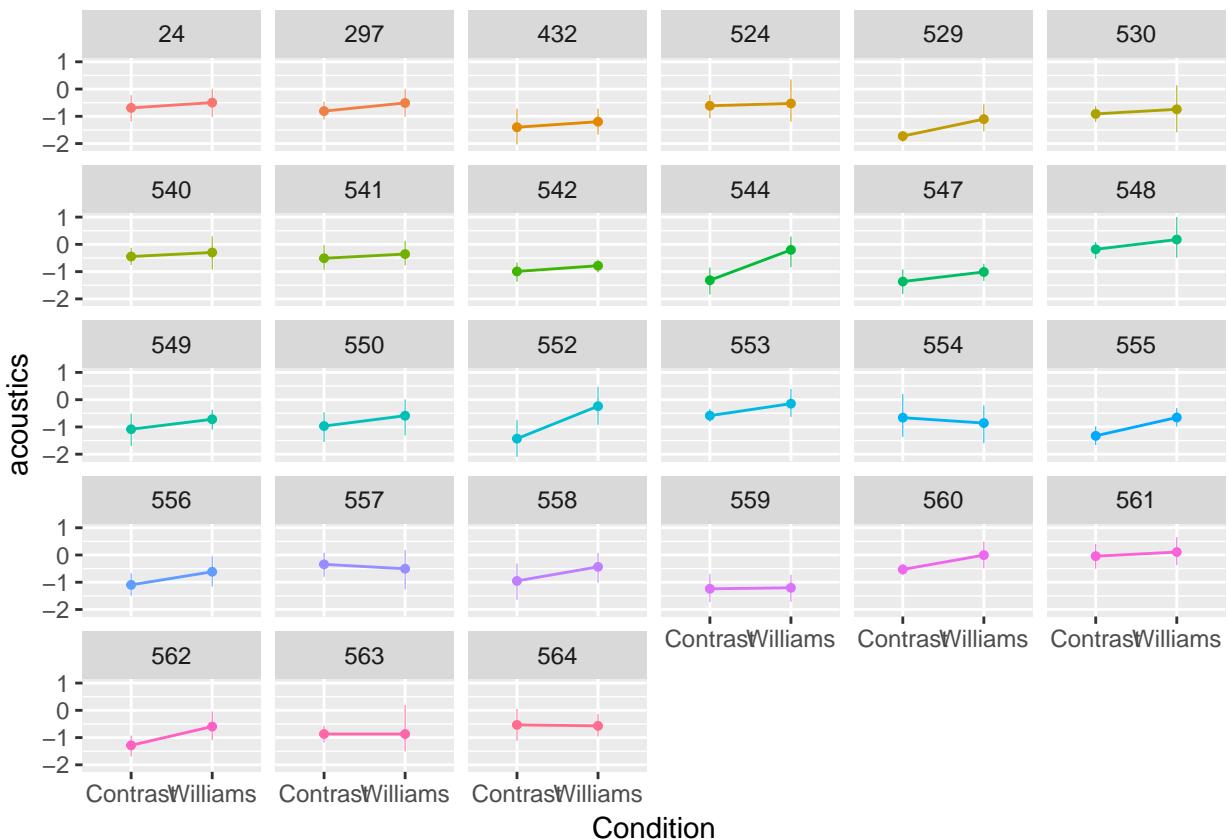
## newdata now contains lower and upper bounds of 95% CI, with 'prediction' = median from simulations
```

### 7.12.2 Random intercept and slope model for givenness data

We would like to check whether the `conditionLabel.williams` effect differs by participant as well:

```
givenness$participant <- as.factor(givenness$participant)
```

```
ggplot(aes(x=conditionLabel, y=acoustics), data=givenness) +
  stat_summary(fun.data="mean_cl_boot", aes(color=participant), size=0.1) +
  stat_summary(fun.y=mean, geom='line', aes(group=participant,color=participant)) +
  facet_wrap(~participant) +
  theme(legend.position='none') +
  xlab("Condition")
```



To do so, we fit a model with a by-participant random intercept and a by-participant random slope for `conditionLabel.williams` (as explained in Sec. 7.3):

```
mod1c.not <- lmer(acoustics ~ conditionLabel.williams + (1 + conditionLabel.williams||participant), data=)
```

`(1 + conditionLabel.williams || participant)` is the term for an (uncorrelated) by-subject random intercept and by-subject random slope.

Model output:

```
summary(mod1c.not)
```

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula:  
## acoustics ~ conditionLabel.williams + (1 + conditionLabel.williams ||  
## participant)  
## Data: givenness  
##  
## REML criterion at convergence: 897.9  
##  
## Scaled residuals:  
##    Min     1Q Median     3Q    Max  
## -3.0538 -0.7129  0.0083  0.6540  3.3136  
##  
## Random effects:  
## Groups      Name           Variance Std.Dev.  
## participant (Intercept) 0.08937  0.299  
## participant.conditionLabel.williams 0.00000  0.000  
## Residual      0.55800  0.747  
## Number of obs: 382, groups: participant, 27  
##  
## Fixed effects:  
##                           Estimate Std. Error      df t value Pr(>|t|)  
## (Intercept)            -0.71856   0.06916  26.21655 -10.39 8.66e-11  
## conditionLabel.williams  0.32626   0.07677 356.75848    4.25 2.73e-05  
##  
## (Intercept) ***  
## conditionLabel.williams ***  
## ---  
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1  
##  
## Correlation of Fixed Effects:  
##          (Intr)  
## cndtnLbl.wl  0.002
```

Under `Random effects` we find the variance estimator for the by-participant random slope to be effectively 0 ( $6.255 \cdot 10^{-16}$ ). Thus, the model's predictions are the same as Model 1B—there is no (detectable) by-subject variability in the effect of `conditionLabel.williams`.

## 7.13 Appendix: Extended exercise

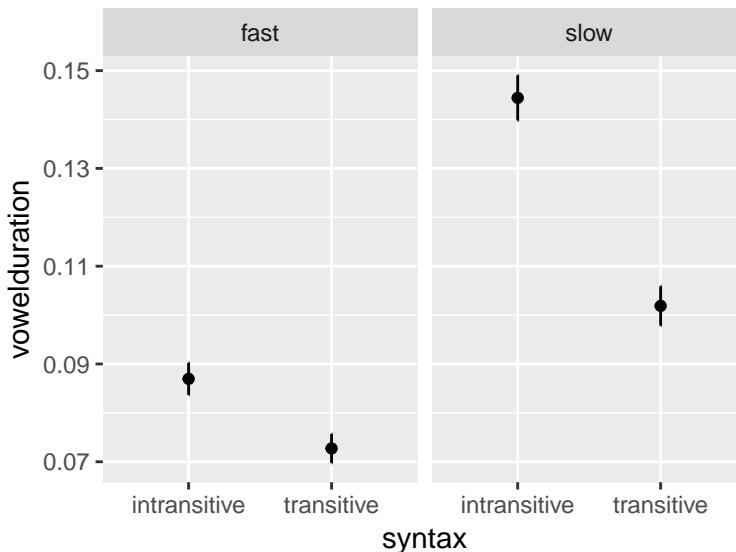
In this exercise you'll build a linear mixed-effects model using:

- Data: tapping data

of how `vowelduration` (response) depends on the predictors `syntax` and `speecharate`.

- Response: `vowelduration`

The empirical effect is:



The fixed effects to be included in the model are `syntax`, `speechrate`, and their interaction.

#### Part 1

- Make normalized predictors (values:  $\approx -0.5, +0.5$ ):
  - `syntax.trans` (+ = transitive, - = intransitive)
  - `speechrate.slow` (+ = slow, - = fast)
- For fixed effects `syntax.trans`, `speechrate.slow`, `syntax.trans:speechrate.slow`:
  - What are all possible random slope terms?
  - Hint: both by-participant and by-item random effects are possible, but some random slopes are not possible.

#### Part 2:

Fit an LME for these fixed effects

- With by-item, by-participant random intercepts and all possible slopes
- Uncorrelated random effects

#### Part 3:

- Interpret the results of the model: are the fixed-effect coefficients in the expected direction, given the empirical data?
- Compare the random slope sizes to the fixed-effect sizes for the same predictors (e.g. compare the fixed effect for `speechrate.slow` to the the by-participant random slope effect “standard deviation” for `speechrate.slow`). Do the three effects ever switch directions (for some participants, items)?

**Solution** model:

```
## Linear mixed model fit by REML. t-tests use Satterthwaite's method [  
## lmerModLmerTest]  
## Formula:  
## vowelduration ~ syntax.trans * speechrate.slow + (1 + syntax.trans *  
##     speechrate.slow || item) + (1 + syntax.trans * speechrate.slow ||  
##     participant)  
## Data: tapped  
##
```

```

## REML criterion at convergence: -2911.9
##
## Scaled residuals:
##      Min    1Q Median    3Q   Max
## -3.8002 -0.5148 -0.0247  0.5248  5.3484
##
## Random effects:
## Groups           Name        Variance Std.Dev.
## participant     syntax.trans:speechrate.slow 0.000e+00 0.000000
## participant.1   speechrate.slow            4.459e-04 0.021115
## participant.2   syntax.trans            4.842e-05 0.006958
## participant.3   (Intercept)          3.966e-04 0.019916
## item             syntax.trans:speechrate.slow 0.000e+00 0.000000
## item.1           speechrate.slow          1.245e-04 0.011158
## item.2           syntax.trans            4.895e-05 0.006996
## item.3           (Intercept)          1.428e-03 0.037785
## Residual          (Intercept)          7.831e-04 0.027983
## Number of obs: 721, groups: participant, 23; item, 8
##
## Fixed effects:
##                   Estimate Std. Error      df t value
## (Intercept)       0.101448  0.014029  8.381835  7.232
## syntax.trans     -0.028215  0.003546  9.113087 -7.956
## speechrate.slow  0.043487  0.006269 18.830504  6.937
## syntax.trans:speechrate.slow -0.029350  0.004173 631.448408 -7.033
##                  Pr(>|t|)
## (Intercept)       7.08e-05 ***
## syntax.trans      2.15e-05 ***
## speechrate.slow  1.37e-06 ***
## syntax.trans:speechrate.slow 5.25e-12 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) syntx. spchr.
## syntax.trns  0.000
## spechrt.slw  0.000  0.002
## syntx.trn:.  0.001 -0.001  0.000

```

The rest of the solution is omitted.

## 7.14 Solutions

**Q:** What does this mean? (That the fixed-effect coefficients are the same in the predictions made by Model 1A and Model 1B?)

**A:** The predictions of Model 1A (simple linear regression) are (basically) the same as the Model 1B makes for an “average speaker”.

**Q:** How much variability is there in participants’ intercept values?

**A:** The estimated by-participant intercept variance is  $\hat{\sigma}_{s,0} = 1.267$ .

---

**Q:** What is the interpretation of each random intercept in this example? For example, what do  $\alpha_{par,2}$  and  $\alpha_{item,5}$  mean?

**A:** The random intercepts are the offsets to the intercept for each participant and item.  $\alpha_{par,2}$  is the offset for participant 2 (the difference between their intercept value and  $\beta_0$ ), and  $\alpha_{item,5}$  is the offset for item 5.

---

**Q:**

- Is there more variability among participants or items?
- This pattern is common in laboratory experiments. Why?

**A:** There is more variability among participants. This pattern is common in laboratory experiments because items are constructed by the researcher to vary in a particular way, while the participants in an experiment are a truly random sample. This pattern might reverse if we had all linguistics majors whose first language is English read a carefully chosen word list, then examined vowel formants in these words, for example.

---

**Exercise 1:**

- In Model 2A:
  - 95% of subjects have intercepts between \_\_\_\_\_ and \_\_\_\_\_
  - 95% of items have intercepts between \_\_\_\_\_ and \_\_\_\_\_
- What is the residual error for these three models, fit with the same fixed effects (`acoustics ~ conditionLabel.williams`):
  - Model 1A (simple linear regression) (fit `mod1a`, then `summary(mod1a)`)
  - Model 1B (by-participant random intercept)
  - Model 2A (by-participant and by-word random intercept)
- Why does the pattern you see make sense?

**A:** (Not shown: 95% bounds, and the actual residual error for each model.) The pattern is that the more random intercept terms are included, the more the residual error goes down. This is because the error in the original simple regression model has been “parcelled out” to variability among participants and items.

---

**Q:** What would it mean, intuitively, if there were a strong positive relationship (“positive correlation”) between (1) and (2)? (“Participants who \_\_\_\_\_ also have a higher \_\_\_\_\_.”)

**A:** Participants with higher values of `rhymeRating` (the response) also have a steeper slope of `relDuration` (the predictor).

---

**Q:** What is the interpretation of each contrast?

**A:** Contrast 1: *bad* versus *voice*. Contrast 2: *good* versus *bad/voice*.



# Chapter 8

## Mixed-effects logistic regression

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(dplyr)
library(ggplot2)
library(arm)
library(languageR)
library(influence.ME)

## loads tappedMcGillLing620.csv from OSF project for Kilbourn-Ceron et al (2017) data
tapped <- read.csv(url("https://osf.io/pwdzx/download"))

## there is some missing data for 'tapped' variable -- exclude to avoid warnings later
tapped <- filter(tapped, !is.na(tapped))

tapped <- mutate(tapped, tapped.num=as.numeric(tapped)-1)
tapped$released <- factor(tapped$released, levels=c("Unreleased", "Released"))
tapped <- mutate(tapped,
                 syntax.trans = arm::rescale(syntax),
                 speechrate.slow = arm::rescale(speechrate)
)

## loads alternativesMcGillLing620.csv from OSF project for Wagner (2016) data
alternatives <- read.csv(url("https://osf.io/6qctp/download"))

## loads givennessMcGillLing620.csv from OSF project for Wagner (2012) data
givenness <- read.csv(url("https://osf.io/q9e3a/download"))

givenness <- mutate(givenness,
                    conditionLabel.williams = arm::rescale(conditionLabel),
                    clabel.williams = arm::rescale(conditionLabel),
                    npType.pronoun = arm::rescale(npType),
                    npType.pron = arm::rescale(npType),
```

```

        voice.passive = arm::rescale(voice),
        order.std = arm::rescale(order),
        stressshift.num = (as.numeric(stressshift) - 1)
    )

## function for computing accuracy of a logistic regression model
## (on the dataset used to fit the model)
## lrMod = fitted model
## responseVar = name of response variable for lrMod
## use.ranef = TRUE or FALSE (should we predict for grouping factor levels in this data, or for an "ave"
##
## adapted from: https://www.r-bloggers.com/evaluating-logistic-regression-models/
lrAcc <- function(lrMod, responseVar, use.ranef=TRUE){
    ## convert response variable into a factor if it's not one
    if(!is.factor(model.frame(lrMod)[,responseVar])){
        model.frame(lrMod)[,responseVar] <- as.factor(model.frame(lrMod)[,responseVar])
    }

    ## model predictions in log-odds
    if(use.ranef){
        preds = predict(lrMod, newdata=model.frame(lrMod))
    } else{
        ## predict for average participant/item/other
        preds = predict(lrMod, newdata=model.frame(lrMod), re.form=NA)
    }
    ## transform to 0/1 prediction
    preds <- ((sign(preds)/2)+0.5)

    ## response variable values, transformed to 0/1
    respVarValues <- model.frame(lrMod)[,responseVar]
    ## if the response is already 0/1, just use it
    if(is.numeric(respVarValues)){
        y <- respVarValues
    } else{
        ## otherwise, transform to 0/1
        y <- (as.numeric(model.frame(lrMod)[,responseVar])-1)
    }

    ## how often is prediction the same as the actual response
    acc <- sum(preds==y)/length(preds)

    return(acc)
}

## baseline accuracy for a logistic regression model lrMod
## with a given response variable
baselineAcc <- function(lrMod, responseVar){
    response <- model.frame(lrMod)[,responseVar]
    tab <- table(response)
    return(max(tab)/sum(tab))
}

```

**Note:** Answers to some questions/exercises not listed in text are in Solutions.

## 8.1 Preliminaries

*Mixed-effects logistic regression* (MELR) is to logistic regression as linear mixed-effects models are to linear regression. MELRs combine pieces we have seen previously in chapters on logistic regression and linear mixed-effects models:

### 1. Logistic regression

- Binary response  $Y$ 
  - Ex: `tapped = 1 or 0`, in the `tapping` dataset.
- Model log-odds that  $Y$  happens.

### 2. Fixed effects

- “Predictors” from logistic regression, now called “fixed effects”
  - Ex: `speechrate`, `vowelduration`, `syntax`
- Capture effect of each predictor across participants/items.

### 3. Random effects

- Capture by-participant/item variability
- Ex: in overall tapping rate (“random intercept”), in the effect of a predictor across participants (“random slope”) or items.

Remember that logistic regression models the **log-odds** of a “hit” ( $Y = 1$ ) as a function of predictors. Not the number of hits, the probability of a hit, or anything else.

### 8.1.1 Motivation

To get a sense of what kind of variability would be captured by the random effects in MELR models, we can examine by-participant and by-item variability in two datasets where the outcome is a binary (0/1) variable, just plotting the empirical data.

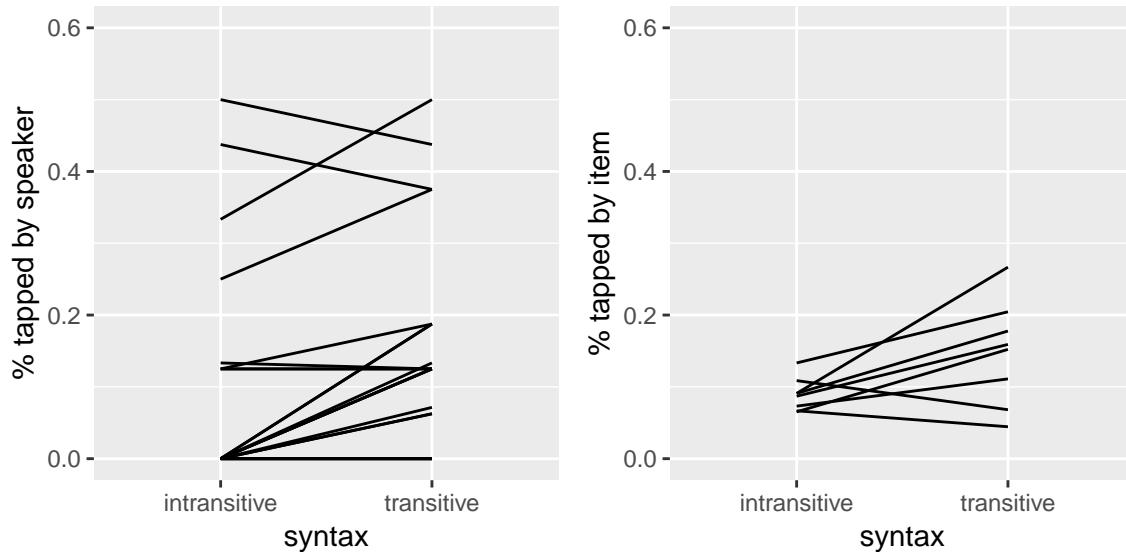
#### Tapping data

From the `tapping` dataset, here are by-participant and by-item plots for the effect of `syntax`, the predictor of primary interest, on the probability of `tapping`:

```
library(gridExtra)
day19Plot1 <- ggplot(aes(x=syntax, y=tapped.num), data=tapped) +
  stat_summary(fun.data="mean_cl_boot", geom='line', aes(group=participant)) +
  coord_cartesian(ylim=c(0,0.6)) +
  ylab("% tapped by speaker")

day19Plot2 <- ggplot(aes(x=syntax, y=tapped.num), data=tapped) +
  stat_summary(fun.data="mean_cl_boot", geom='line', aes(group=item)) +
  coord_cartesian(ylim=c(0,0.6)) +
  ylab("% tapped by item")

grid.arrange(day19Plot1, day19Plot2, ncol = 2)
```



### Questions:

- Does it look like there is by-participant variability in the intercept, slope, or both? What about by-item variability?

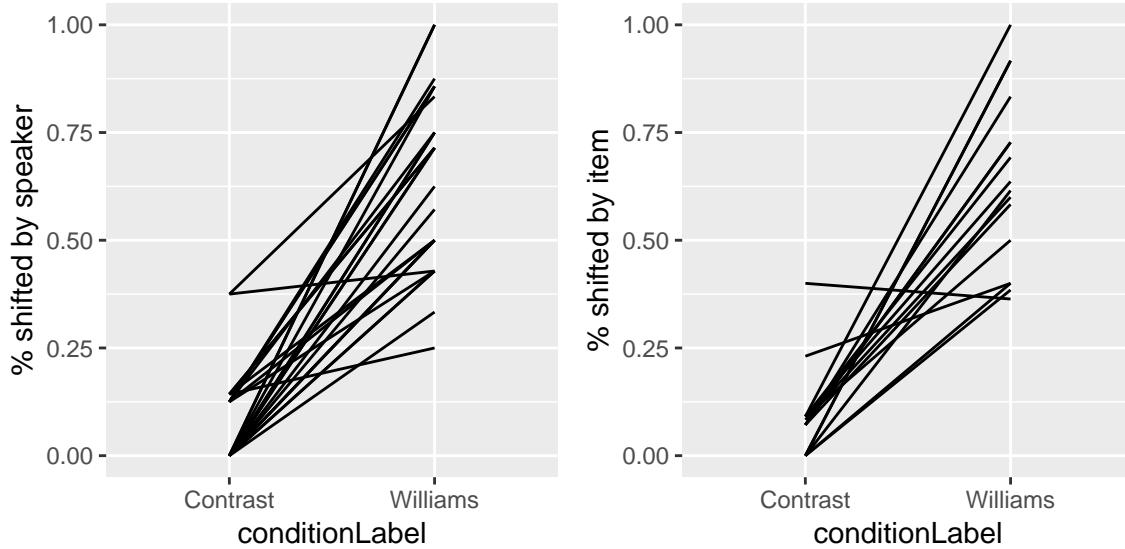
### Givenness data

For the `givenness` dataset, here are by-participant and by-item plots for the Williams effect (the effect of `conditionLabel` on probability of `stressshift.num`), which was of primary interest:

```
day19Plot3 <- ggplot(aes(x=conditionLabel, y=stressshift.num), data=givenness) +
  stat_summary(fun.data="mean_cl_boot", geom='line', aes(group=participant)) +
  coord_cartesian(ylim=c(0,1.0)) +
  ylab("% shifted by speaker")

day19Plot4 <- ggplot(aes(x=conditionLabel, y=stressshift.num), data=givenness) +
  stat_summary(fun.data="mean_cl_boot", geom='line', aes(group=item)) +
  coord_cartesian(ylim=c(0,1.0)) +
  ylab("% shifted by item")

grid.arrange(day19Plot3, day19Plot4, ncol = 2)
```



**Questions:**

- Again, for participants and items: do you think there is variability in the intercept? Slope? Both?

## 8.2 Basics

Formally, we assume that there are  $n$  observations. **Fixed effects** and **random effects** are defined in the same way as for linear mixed-effects models:

- $\beta_0$ : the fixed-effect coefficient for the intercept
- $\beta_1, \dots, \beta_k$ : the fixed effect coefficients for  $k$  predictors

Random effects are defined with respect to one or more *grouping levels*. For example, there could be data from  $J$  participants.

Mixed-effect logistic regressions are one kind of *generalized linear mixed model* (GLMM)—analogously to logistic regression being one kind of generalized linear model. To fit a MELR model in the `lme4` package, you use the `glmer()` function (generalized linear mixed effects regression), with a `family=binomial()` argument, similarly to fitting a logistic regression using the `glm()` function.<sup>1</sup>

All pieces of an MELR model are familiar from our units on linear mixed models (Chapter 7) and logistic regression (Chapter 5), so we will dive right in to a realistic example model that uses:

- Crossed random effects (participants, items)
- Multiple predictors

### 8.2.1 Model 1: givenness data, crossed random effects (intercepts + slopes)

We again consider the `givenness` data, using the binary annotation by a research assistant for whether stress shifted from the noun phrase (1/0 = shifted/not shifted).

- The response is `stressshift`

The data is grouped by:

---

<sup>1</sup>The `family='binomial'` argument specifies this regression uses a “logit link”, as for logistic regressions (see Baayen (2008)).

- participants ( $1, \dots, J$ )
- items ( $1, \dots, K$ )

We use these fixed effect terms, to build a relatively simple model (no interactions):

- `clabel.williams` (varies across participants and items)
- `npType.pron` (varies across participants and items)
- `voice.passive` (varies across participants, not across items)

Recall that `clabel.williams` and `npType.pron` are observation-level (they vary across participants and items) while `voice.passive` is item-level (it varies across participants, but not items).

The random effects contain intercepts and all possible random slopes:

- Intercepts: by-participant, by-word
  - Slopes:
    - Participant: `clabel.williams`, `npType.pron`, `voice.passive`
    - Item: `clabel.williams`, `npType.pron`
- 

It may be helpful for your understanding to explicitly write out what the equation for this model is. To do so, we define:

- $\alpha_{0,j}$  : random intercept for participant  $j$
- $\delta_{0,k}$  : random intercept for item  $k$
- $\alpha_{clabel,j}$  : random slope of `clabel` for participant  $j$
- $\delta_{clabel,k}$  : random slope of `clabel` for item  $k$
- ... similar for other random slopes.

The model for observation  $i$  is then:

$$\begin{aligned} \text{logit}(P(y_i = 1)) = & \beta_0 + \alpha_{0,j[i]} + \delta_{0,k[i]} + \\ & (\beta_{clabel} + \alpha_{clabel,j[i]} + \delta_{clabel,k[i]}) \cdot \text{clabel}_i + \\ & (\beta_{npType} + \alpha_{npType,j[i]} + \delta_{npType,k[i]}) \cdot \text{npType}_i + \\ & (\beta_{voice} + \alpha_{voice,j[i]}) \cdot \text{voice}_k[i] \end{aligned}$$

In this formula:

- $\beta_0 + \alpha_{0,j[i]} + \delta_{0,k[i]}$  is our **intercept** for participant  $j[i]$  and item  $k[i]$
- $\beta_{clabel} + \alpha_{clabel,j[i]} + \delta_{clabel,k[i]}$  is our **slope for clabel.williams** for participant  $j[i]$ , item  $k[i]$
- $\beta_{npType} + \alpha_{npType,j[i]} + \delta_{npType,k[i]}$  is our **slope for npType.pron** for participant  $j[i]$ , item  $k[i]$
- $\beta_{voice} + \alpha_{voice,j[i]}$  is our **slope for voice.passive** for participant  $j[i]$

(Recall that observation  $i$  is from participant  $j[i]$  and item  $k[i]$ .)

---

The terms fitted in this model—and hence shown in the regression model output—are:

- 4 fixed-effect coefficients:  $\beta_0$ ,  $\beta_{clabel}$ ,  $\beta_{npType}$ ,  $\beta_{voice}$

- For 3 predictors
- 7 variance components:
  - Random intercepts: 2 terms
  - Random slopes: 5 terms
  - No correlations

Note that unlike linear mixed-effects models, there is no residual error variance term, for the same reasons as for logistic regression (see Sec. 5.1.4).

### 8.2.1.1 Fitting the model

To fit this model in R:

```
lrMod1 <- glmer(stressshift ~ clabel.williams + npType.pron + voice.passive +
  (1 + clabel.williams + npType.pron || item) +
  (1 + clabel.williams + npType.pron + voice.passive || participant),
  data=givenness,
  family="binomial",
  control=glmerControl(optimizer = "bobyqa"))
```

Note that the `control=glmerControl(optimizer = "bobyqa")` option is not necessary, but often facilitates model convergence.<sup>2</sup>

The model output is:

```
summary(lrMod1)

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##   Family: binomial ( logit )
## Formula: stressshift ~ clabel.williams + npType.pron + voice.passive +
##           (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
##             npType.pron + voice.passive || participant)
##   Data: givenness
## Control: glmerControl(optimizer = "bobyqa")
##
##       AIC     BIC   logLik deviance df.resid
##     349.6   393.0   -163.8    327.6     371
##
## Scaled residuals:
##       Min     1Q Median     3Q    Max
## -2.2442 -0.3174 -0.1881  0.4603  4.5955
##
## Random effects:
##   Groups            Name        Variance Std.Dev.
##   participant      voice.passive 1.648e+00 1.284e+00
##   participant.1   npType.pron   5.823e-01 7.631e-01
##   participant.2   clabel.williams 5.013e-15 7.080e-08
##   participant.3 (Intercept)   1.685e-01 4.105e-01
##   item            npType.pron   5.394e-01 7.344e-01
##   item.1          clabel.williams 1.619e+00 1.273e+00
##   item.2          (Intercept)   2.624e-14 1.620e-07
```

---

<sup>2</sup>This option changes the optimization function R uses to find the parameters that maximize likelihood.

```

## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.1383   0.2076 -5.483 4.18e-08 ***
## clabel.williams 3.7711   0.5676  6.644 3.05e-11 ***
## npType.pron    0.7917   0.3987  1.986  0.0471 *
## voice.passive  0.6496   0.4149  1.566  0.1174
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) clbl.w npTyp.
## clabl.wllms -0.458
## npType.pron -0.141  0.174
## voice.passv -0.036  0.082  0.028

```

### 8.2.1.2 Interpreting the model

The fixed-effect and random-effect terms are interpreted similarly to a linear mixed-effects model, with the difference that the model predicts changes in log-odds.

Some examples of fixed effects:

- The intercept is  $\hat{\beta}_0 = -1.13$ :
  - Interpretation: for an average subject and average item, with all predictors held at mean values:
$$P(\text{shift stress}) = \text{logit}^{-1}(-1.13) = 0.24$$
- The slope for npType is  $\hat{\beta}_{\text{npType}} = 0.79$ 
  - Interpretation: for an average subject and item, the log-odds of shifting stress are 0.79 higher for pronouns than for full NPs.
  - (Alternatively: the odds of shifting stress are 2.20 times higher ( $= e^{0.79}$ ) for pronouns than for full NPs.)

For the random effects, first considering random intercepts:

- By-participant random intercept variance:  $\hat{\sigma}_{s,0} = 0.41$ 
  - 95% of participants have “baseline” log-odds of shifting stress in (-1.96, -0.31)
    - \* ( $= \hat{\beta}_0 \pm 1.96 \cdot \hat{\sigma}_{s,0}$ )
    - \* Alternatively, 95% of participants have baseline probability of shifting stress in (0.12, 0.42) ( $= \text{invlogit}(-1.96)$ ,  $\text{invlogit}(-0.31)$ ).
- In contrast, there is basically no variability among items in “baseline” log-odds of shifting stress ( $\hat{\sigma}_{w,0} \approx 0$ ).

For random slopes, consider the condition label effect (i.e. the “Williams effect”). The degree of variability among participants and items in the clabel.williams slope is:

$$\hat{\sigma}_{\text{clabel,partic}}, \quad \hat{\sigma}_{\text{clabel,item}}$$

To visualize the predicted Williams effect for different participants and items:

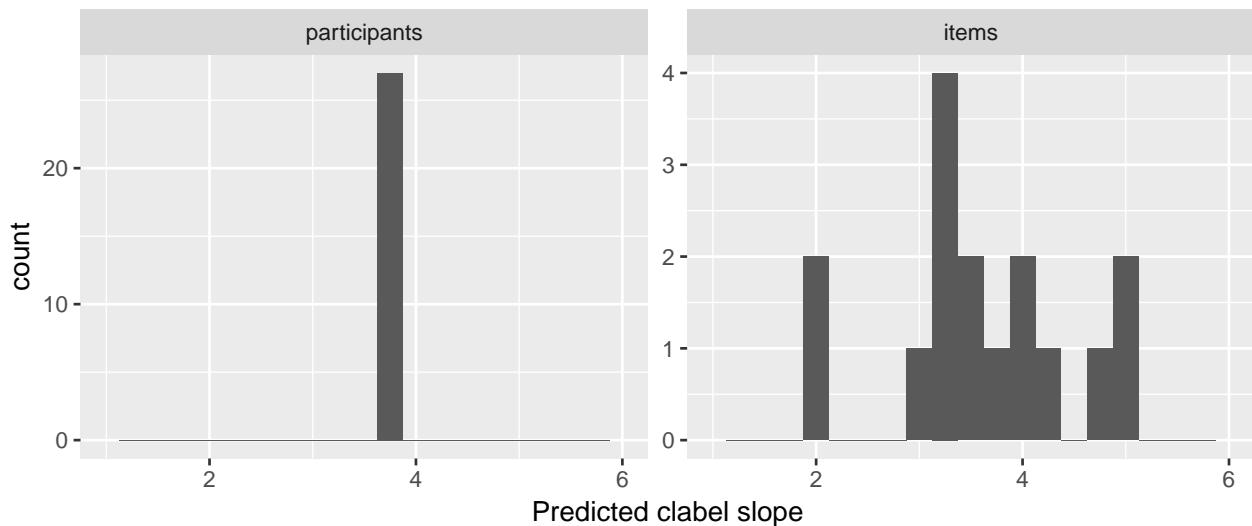
```

particSlopes <- fixef(lrMod1)['clabel.williams'] + ranef(lrMod1)$participant$clabel.williams
itemSlopes <- fixef(lrMod1)['clabel.williams'] + ranef(lrMod1)$item$clabel.williams

lrMod1SlopeDf <- rbind(data.frame(type='participants', slope=particSlopes),
                         data.frame(type='items', slope=itemSlopes))

ggplot(aes(x=slope), data=lrMod1SlopeDf) +
  geom_histogram(binwidth=0.25) +
  facet_wrap(~type, scales="free") +
  xlim(1,6) +
  xlab("Predicted clabel slope")

```



The model predicts that the size of the Williams effect varies among items—but not participants—and is robust, in the sense of having the same direction across all participants and items.

Thus: participants (but not items) differ in the intercept, while items (but not participants) differ in the Williams effect (the `clabel.williams` slope).

## 8.3 Hypothesis testing

### 8.3.1 Fixed effects

Recall that for linear mixed-effects models, there is not a consensus on how to calculate p-values and whether doing so is a good idea (see Sec. 7.5.3), so no test statistic or *p*-value is given by default in the `lmer` output. One rough method was to assume that the estimated coefficient value divided by its standard error was normally distributed, and use this to calculate a *p*-value (for a two-sided “Wald test”).

For generalized linear mixed-effects models, this is a somewhat more kosher method, and `glmer()` shows by default:

- The test statistic  $z$  ( $= \frac{\hat{\beta}}{SE(\hat{\beta})}$ )
- The *p*-value for a two-sided Wald test.<sup>3</sup>

<sup>3</sup>Meaning, how likely is it that  $|z|$  would be at least this large, under the null hypothesis, assuming that  $z$  is normally

However, these  $p$ -values are still approximations—they are only correct in the limit of a large number of observations and large number of groups. **These Wald-test based  $p$ -values are good enough for our purposes**, but better methods would be:

- A likelihood ratio (LR) test: slightly better, fast
- Parametric bootstrap: much better, very slow

See the discussion for linear mixed models in Baayen (2008) for more detail on both, and Ben Bolker's GLMM page for more recent discussion. You may want to use a “better” method when reporting  $p$ -values in a paper. Here are a few examples:

### Likelihood ratio test for one term

To calculate an LR test  $p$ -value for just the `npType.pron` fixed-effect term in Model 1:

```
lrMod1.sub <- update(lrMod1, . ~ . - npType.pron)
anova(lrMod1, lrMod1.sub)
```

```
## Data: givenness
## Models:
## lrMod1.sub: stressshift ~ clabel.williams + voice.passive + (1 + clabel.williams +
## lrMod1.sub:      npType.pron || item) + (1 + clabel.williams + npType.pron +
## lrMod1.sub:      voice.passive || participant)
## lrMod1: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod1:      (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
## lrMod1:      npType.pron + voice.passive || participant)
##          Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## lrMod1.sub 10 351.18 390.63 -165.59   331.18
## lrMod1     11 349.64 393.04 -163.82   327.64 3.5413      1   0.05986 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The  $p$ -value ( $p = 0.059$ ) is slightly higher than for the Wald test used in `lrMod1` ( $p = 0.047$ ).

### Likelihood ratio test for each term

The `afex` package lets you quickly compute an LR-based  $p$ -value for each term:

```
library(afex)

## fit model and get LR-based p-values:
lrMod1.lr <- mixed(stressshift ~ clabel.williams + npType.pron + voice.passive + (1 + clabel.williams +
                                         (1 + clabel.williams + npType.pron + voice.passive||participant),
                                         data=givenness,
                                         family="binomial", method='LRT',
                                         control=glmerControl(optimizer = "bobyqa"))

## Fitting 4 (g)lmer() models:
## [....]
## see the results:
lrMod1.lr

## Mixed Model Anova Table (Type 3 tests, LRT-method)
##
```

---

distributed?

```

## Model: stressshift ~ clabel.williams + npType.pron + voice.passive +
## Model:      (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
## Model:      npType.pron + voice.passive || participant)
## Data: givenness
## Df full model: 11
##          Effect df    Chisq p.value
## 1 clabel.williams  1 26.67 *** <.0001
## 2     npType.pron   1     3.54 +    .06
## 3   voice.passive  1     2.36     .12
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

```

### Parametric bootstrap (PB) test for each term

This example uses a version of Model 1 with random intercepts only, because using random slopes takes a very long time. Note this is only for pedagogical purposes—it is never a good idea to report a model where the only random effects are random intercepts!

```

library(afex)

## use multiple cores, if your machine has them.
require(parallel)
(nc <- detectCores()) # number of cores

## [1] 8

cl <- makeCluster(rep("localhost", nc)) # make cluster

## fit a *random-intercepts only* model (in general, not OK, but it would take much longer to fit the m
## This still takes 5 minutes on my 8-core computer:

lrMod1.pb <- mixed(stressshift ~ clabel.williams + npType.pron + voice.passive +
                     (1|item) + (1|participant),
                     data=givenness,
                     family="binomial", method='PB', args_test=list(nsim=1000, cl=cl),
                     control=glmerControl(optimizer = "bobyqa"), cl=cl)

## Fitting 4 (g)lmer() models.
## Obtaining 3 p-values:
## [...]
##  look at the p-values (nb: 0.001 = minimum p-value when nsim=1000)
lrMod1.pb

## Mixed Model Anova Table (Type 3 tests, PB-method)
##
## Model: stressshift ~ clabel.williams + npType.pron + voice.passive +
## Model:      (1 | item) + (1 | participant)
## Data: givenness
##          Effect df    Chisq p.value
## 1 clabel.williams  1 150.61 ***   .0010
## 2     npType.pron   1     5.06 *    .03
## 3   voice.passive  1     7.43 **   .009
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '+' 0.1 ' ' 1

```

### 8.3.2 Random effects

It may also be of interest to test whether random-effect terms significantly improve the model. As for linear mixed models, in the large-sample limit the difference in deviance between the model with and without  $k$  random effect terms is approximately  $\chi^2(k)$  distributed. Thus, we can use a likelihood ratio test to assess the null hypothesis that (all of) the random effect terms have zero variance.

#### Example

Do items significantly differ in the `clabel` effect (i.e. the Williams effect), in Model 1?

```
lrMod1.1 <- glmer(stressshift ~ clabel.williams + npType.pron + voice.passive +
  (1 + npType.pron || item) +
  (1 + npType.pron + voice.passive + clabel.williams || participant),
  data=givenness,
  family="binomial",
  control=glmerControl(optimizer = "bobyqa"))
anova(lrMod1, lrMod1.1)

## Data: givenness
## Models:
## lrMod1.1: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod1.1:     (1 + npType.pron || item) + (1 + npType.pron + voice.passive +
## lrMod1.1:     clabel.williams || participant)
## lrMod1: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod1:     (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
## lrMod1:     npType.pron + voice.passive || participant)
##           Df   AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## lrMod1.1 10 352.24 391.69 -166.12    332.24
## lrMod1   11 349.64 393.04 -163.82    327.64 4.5994      1   0.03198 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

This result could be reported as: “A likelihood ratio test showed that items differed significantly in the effect of condition label ( $\chi^2(1) = 4.6$ ,  $p = 0.031$ ).”

## 8.4 Fixed and random effects

In a model with random slopes, it is not clear what the best way is to assess “does predictor  $X$  significantly improve the model?” Testing the fixed-effect only (as above) is the most common method—but this does not take into account variability in the effect among participants or items. It is possible for a predictor to show no **overall** effect (across participants and items), but to show significant variability among participants or items—indeed, this may be **why** there is no significant overall effect.

We can instead assess the contribution of predictor  $X$  by doing model comparison (via a LR test) with a model where all terms involving  $X$  are excluded (fixed and random effects). For example, to assess the contribution of `voice` to Model 1, we would exclude the fixed effect and the by-participant random effect:

```
## all terms involving `voice` excluded
lrMod1.1 <- glmer(stressshift ~ clabel.williams + npType.pron +
  (1 + clabel.williams + npType.pron|| item) +
  (1 + npType.pron + clabel.williams || participant),
  data=givenness,
  family="binomial",
```

```

control=glmerControl(optimizer = "bobyqa"))
anova(lrMod1, lrMod1.1)

## Data: givenness
## Models:
## lrMod1.1: stressshift ~ clabel.williams + npType.pron + (1 + clabel.williams +
## lrMod1.1:     npType.pron || item) + (1 + npType.pron + clabel.williams || 
## lrMod1.1:     participant)
## lrMod1: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod1:     (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
## lrMod1:     npType.pron + voice.passive || participant)
##          Df    AIC    BIC  logLik deviance Chisq Chi Df Pr(>Chisq)
## lrMod1.1  9 353.08 388.59 -167.54    335.08
## lrMod1   11 349.64 393.04 -163.82    327.64 7.4458      2    0.02416 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

So adding information about item voicing (the `voice` predictor) does significantly improve the model ( $\chi^2(2) = 7.4$ ,  $p = 0.024$ )—even though there is no overall effect (fixed effect:  $p = 0.11$ ). This is because there is a lot of between-participant variability in the effect.

## 8.5 MELR Practice

This exercise gives practice fitting and interpreting mixed-effects logistic regression models.

### 8.5.1 Exercise 1: tapping

For the `tapping` data, loaded and processed at the beginning of this chapter, the `response` is `tapped`. We will take the `predictors` to be:

- `syntax.trans`: whether `syntax` is *transitive* (0.5) or *intransitive* (-0.5)
- `speecharate.slow`: whether `speecharate` is *slow* (0.5) or *fast* (-0.5)

#### Part 1: Fitting

Fit an MELR model of this data, with:

- Fixed effects for both predictors
- By-participant and by-item random intercepts
- By-participant and by-item random slopes for both `syntax.trans` and `speecharate.slow`

(Make sure you understand: why is it OK to include these random-slope terms? Is either `syntax.trans` or `speecharate.slow` participant-level or item-level?)

How does this model assume tapping rate depends on syntax and speech rate, intuitively?

#### Part 2: Interpretation

- Do participants or items vary more in the intercept?
- How much do participants vary in the slope of `syntax.trans`?
- Do the model results for `syntax.trans` fit with what we saw in the empirical data?

## 8.6 Model criticism for mixed-effects logistic regression

Model criticism tools are mostly similar to those used for logistic regression (see discussion in Sec. 5.4), with one addition for mixed-effects models: **examining random-effect distributions**, to address whether

- they are approximately normal
- they show evidence of outlier participants or items.

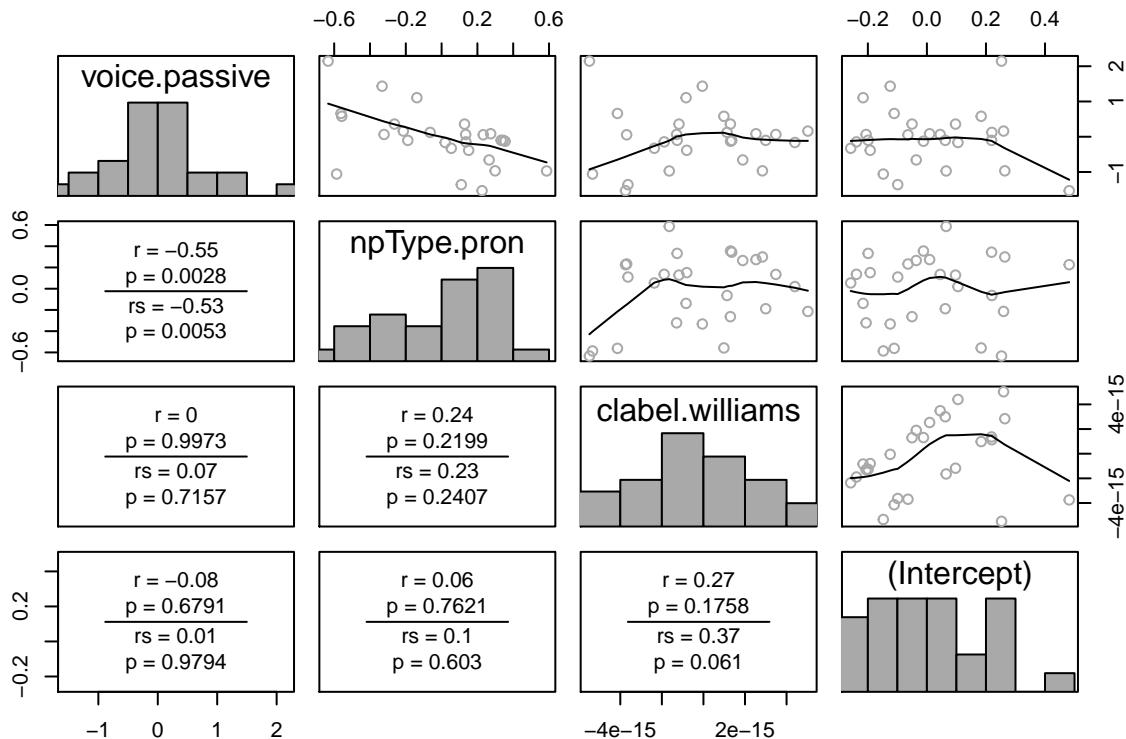
### 8.6.1 Random-effect distributions

We use Model 1 as an example. Recall that the random effects are:

- By-participant: intercept, slopes of `clabel.williams`, `npType.pron`, `voice.passive`
- By-item: intercept, slopes of `clabel.williams` and `npType.pron`

In the pairwise plot of by-participant random effects:

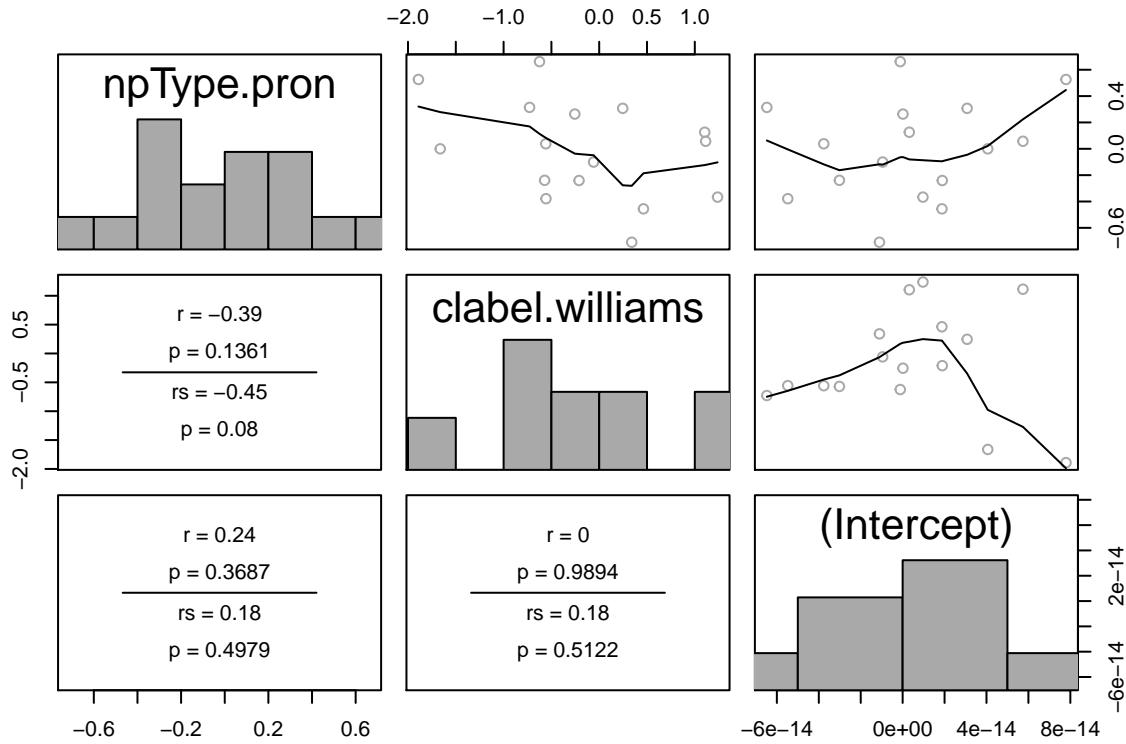
```
pairsCor.fnc(ranef(lrMod1)$participant)
```



There are no obvious outliers. One possible outlier is the participant with the highest random intercept value (lower-right plot, right-most point).

In the pairwise plot of by-item random effects:

```
pairsCor.fnc(ranef(lrMod1)$item)
```



There are no obvious outliers.

All histograms in these two plots look roughly like normal distributions, suggesting no substantial deviations from the normality-of-random-effects assumption. This isn't surprising, given that the model is assuming random effects are normally distributed. It would be better to check the distributions of participant and item empirical means, and of empirical slopes for each predictor. We leave this as an exercise.

### 8.6.2 Residual plots

We will skip these, for lack of time. Residual plots for MELR are similar to logistic regression, and similar issues arise (see Sec. 8.6.2).

- Raw residuals are uninformative, since the model predicts probabilities but observations are 0 or 1.
- Work-arounds exist: binned residuals (calculated using functions in the `arm` package) or deviance residuals.

### 8.6.3 Influence

For non-mixed-effects regression models, we used **Cook's distance** as a measure of influence, roughly defined as "how much do the model coefficients ( $\hat{\beta}$ 's) change when observation  $i$  is removed?"

Cook's distance can be similarly defined for mixed-effects models, at each "level" of the model:

- How much do the fixed-effect coefficients ( $\hat{\beta}$ 's) change when observation  $i$  is removed?
- or " " when participant  $j$  is removed?
- or " " when item  $k$  is removed?

That is, there can be influential items, participants (etc.), or observations.

Unlike for non-mixed models, there is no formula to compute Cook's distance: you have to actually refit the model without a given observation/participant/item to check how much the model changes. The `influence` function in the `influenceME` package does this for you. (It also works for linear mixed-effects models.)

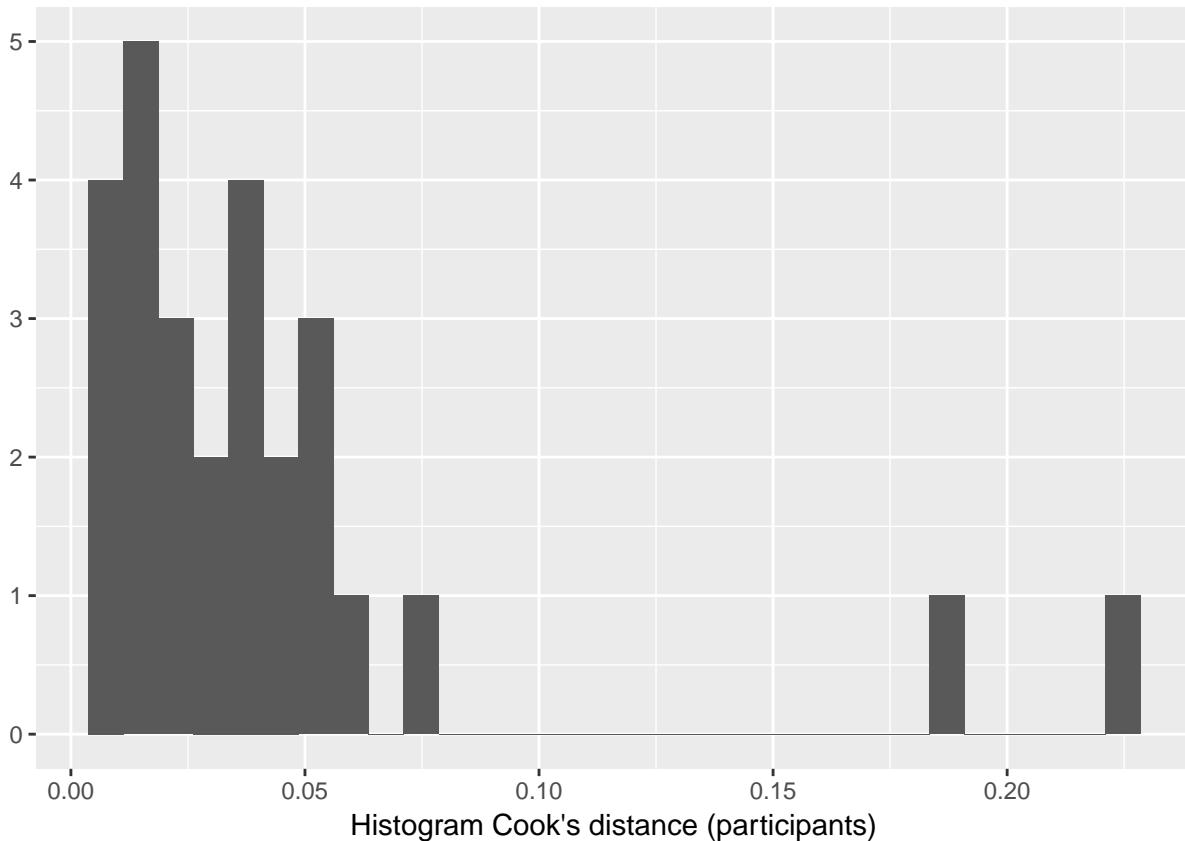
**Warning:** because `influence()` refits the model  $J$  times for  $J$  levels (e.g. participants), or  $n$  times for  $n$  observations, it can take a very long time.

#### Example: Compute influence of each participant for Model 1

```
m1.influence.partic <- influence(lrMod1, group="participant")

# Create data frame with cooks distance for each participant label:
cd=cooks.distance(m1.influence.partic)

# Histogram of cook's distance:
qplot(cd) + xlab("Histogram Cook's distance (participants)")
```



Two participants seem to have an unusual influence (as measured by cook's distance). How do you know which ones?

- The row names in the object `cd` tell you the participant number.
- You could just look at the individual values (type `cd` at R prompt), but it's easier if you sort these values in descending order (the command used here assures that we keep the row names with the the participant number):

```
cd[order(-cd), ,drop = FALSE]
```

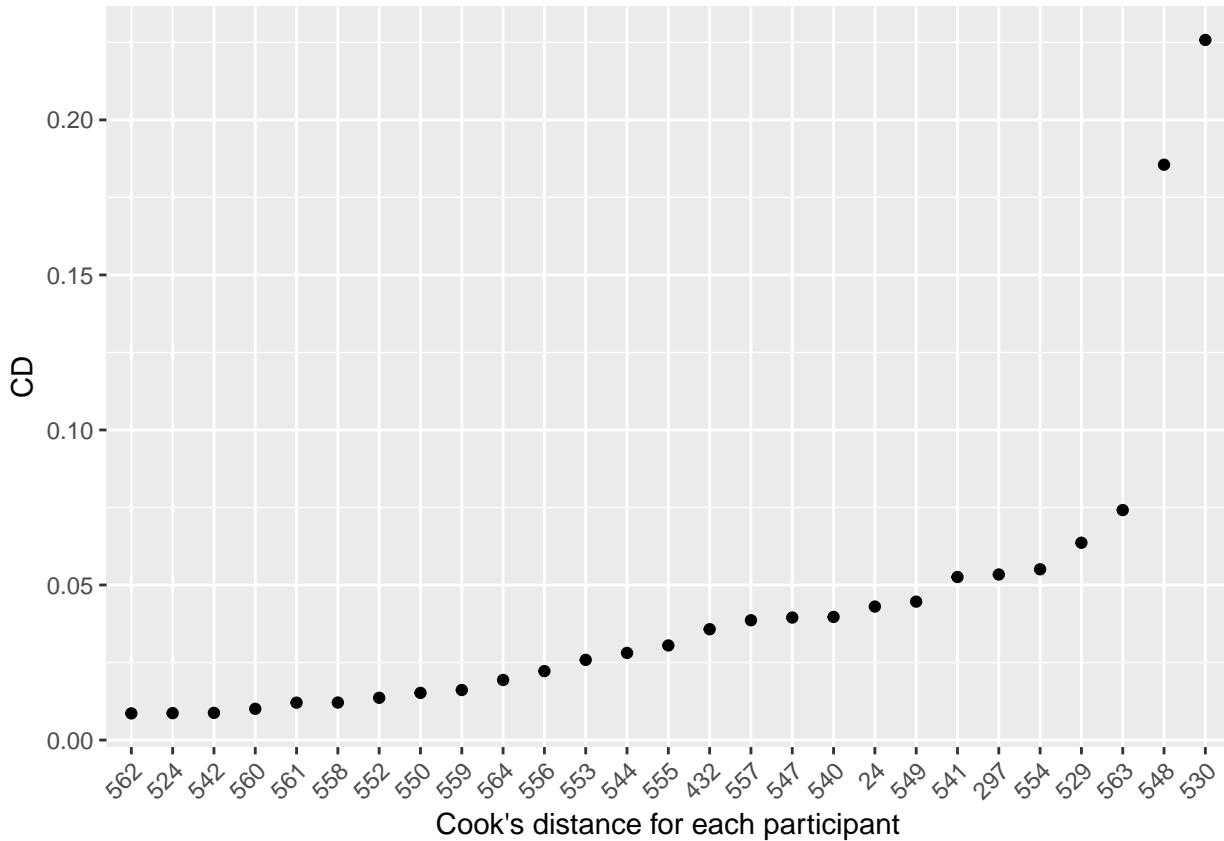
```
## [,1]
```

```
## 530 0.225778367
## 548 0.185550214
## 563 0.074174496
## 529 0.063628074
## 554 0.055073386
## 297 0.053395685
## 541 0.052571319
## 549 0.044627482
## 24 0.043046353
## 540 0.039707331
## 547 0.039506196
## 557 0.038630256
## 432 0.035733805
## 555 0.030503148
## 544 0.028093074
## 553 0.025852398
## 556 0.022235841
## 564 0.019375321
## 559 0.016134504
## 550 0.015203804
## 552 0.013620053
## 558 0.012093557
## 561 0.012054054
## 560 0.010068154
## 542 0.008765176
## 524 0.008675731
## 562 0.008593942
```

It's even easier if you plot the individual distances:

```
# Cook's distance for each individual participant, ordered by distance:
cooksDistance=data.frame(participant=rownames(cd),CD=cd)

ggplot(data=cooksDistance,aes(x = reorder(participant,CD),y = CD)) + geom_point() + xlab('Cook\`s distan
```



There are two very influential participants: 548 and (most influential) 530. These turn out to be the two participants with the strongest `voice` effects.<sup>4</sup>

We could also examine the influence of each item or each observation:

```
influence(lrMod1, group="item")
influence(lrMod1, obs=T)
```

### Detour: What can an influential participant do?

Participant 530 was noted above as a potential outlier—s/he has the highest Cook's distance. It is good practice to see what happens if influential participants (etc.) are omitted.

Compare the models fitted to the dataset with and without this participant:

```
lrMod1.no530 <- update(lrMod1, . ~ ., data=filter(givenness, participant!=530))

summary(lrMod1)$coefficients
```

```
##             Estimate Std. Error   z value Pr(>|z|)
## (Intercept) -1.1383258  0.2076053 -5.483124 4.178800e-08
## clabel.williams 3.7710505  0.5675843  6.644036 3.052075e-11
## npType.pron    0.7917211  0.3987366  1.985574 4.708063e-02
## voice.passive   0.6496316  0.4149409  1.565600 1.174422e-01
```

<sup>4</sup>To see this, plot participant's Cook's distance values against their `voice` random slope values (`raneff(lrMod1)$participant[['voice.passive']]`).

```
summary(lrMod1.no530)$coefficients

##             Estimate Std. Error   z value   Pr(>|z|)
## (Intercept) -1.1848192  0.2138651 -5.540030 3.024197e-08
## clabel.williams  3.8105677  0.5956800  6.397005 1.584543e-10
## npType.pron     0.9129111  0.3427896  2.663182 7.740552e-03
## voice.passive    0.3137257  0.3848432  0.815204 4.149556e-01
```

We see that excluding this participant:

- Strengthens the effects of `clabel.williams` and `npType.pron` (higher effect size, lower  $p$ -value)
- Lessens the effect of `voice.passive` (much lower effect size, higher  $p$ -value)

Thus, excluding this participant makes the results a little clearer (the non-significant result is now very weak; the significant results are even stronger), but doesn't change any qualitative conclusion. One could either exclude this participant or not in reporting the model, but the **existence** of this influential participant should be reported regardless.

This example illustrates why examining influence can be beneficial, beyond simply checking whether model assumptions are satisfied: you may find that some participants or items behave very differently from others, and are obscuring clearer results.

## 8.7 Evaluation measures

As for logistic regression, mixed-effects logistic regression models are evaluated by comparing **baseline** and **full** models. We considered these comparison methods:

1. Likelihood ratio test
2. Classification accuracy

For mixed-effects models, the baseline model is usually taken to be the **intercepts-only model**, which contains random intercepts for each grouping factor (e.g. participant, item) and the fixed-effect intercept ( $\beta_0$ ).

For our Model 1, the baseline intercepts-only model would be:

```
m0 <- glmer(stressshift ~ (1|item) + (1|participant), data=givenness, family="binomial")
```

This baseline can be interpreted as “participants and items vary in ( $P(\text{shift stress})$ )”, or “how well can the response (`stressshift`) be predicted, without adding any information about the predictors?” Note that for **new data** (unseen participants or items), the baseline model will just predict the most common outcome—which is exactly what the baseline model for (non-mixed-effects) logistic regression does.

Comparing the baseline model to the full model asks: how much does the model improve by adding information about the predictors? Note that this means both fixed-effect coefficients and random slopes—and either could significantly improve likelihood.

### 8.7.1 Evaluation measure 1: Likelihood ratio test

As we've now seen in many cases, the difference in deviance between the baseline ( $M_0$ ) and full ( $M_1$ ) models:

$$\Delta D = -2 \log \frac{L(M_0)}{L(M_1)}$$

can be used for a hypothesis-test for large enough samples. If any set of  $k$  terms (fixed-effects  $\beta$ 's and/or random-effect  $\sigma$ 's) are 0,  $\Delta D$  follows a  $\chi^2(k)$  distribution. Thus, we can test the hypothesis that these  $k$  terms are 0 using a likelihood ratio test.

For our Model 1, a LR test comparing the full model to the baseline gives:

```
anova(m0, lrMod1)
```

```
## Data: givenness
## Models:
## m0: stressshift ~ (1 | item) + (1 | participant)
## lrMod1: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod1:      (1 + clabel.williams + npType.pron || item) + (1 + clabel.williams +
## lrMod1:      npType.pron + voice.passive || participant)
##          Df    AIC   BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## m0      3 502.24 514.08 -248.12    496.24
## lrMod1 11 349.64 393.04 -163.82    327.64 168.6     8 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Thus, adding these three predictors significantly improves the model ( $\Delta D = 168.8$ ,  $df = 8$ ,  $p < 0.0001$ ).

### 8.7.2 Evaluation measure 2: Classification accuracy

Classification accuracy is defined similarly to (non-mixed-effects) logistic regression. For each observation, the model is used to predict the probability that the response is 1 ( $Y = 1$ ) for observation  $i$ , and this probability is used to predict the value of  $y_i$ :

$$\hat{y}_i = \begin{cases} 1 & \text{if } \hat{p}_i > 0.5 \\ 0 & \text{if } \hat{p}_i \leq 0.5 \end{cases}$$

*Classification accuracy* is the percent of these predictions that are the same as the observed responses  $y_i$ .

As for non-ME case, a model's classification accuracy must be compared against the baseline model's accuracy.

For our Model 1:

```
lrAcc(lrMod1, 'stressshift')
```

```
## [1] 0.8638743
```

classification accuracy is 86%, compared to 65% for the baseline model:

```
baselineAcc(lrMod1, 'stressshift')
```

```
## [1] 0.6465969
```

Note that the 86% classification accuracy uses the random-effect estimates for the participants and items in this data. That means it does not give a good estimate of what the model's accuracy would be on *unseen* data (new participants and items), which is what we usually care about. We can obtain this accuracy using model predictions for an “average” participant and item, which is equivalent to setting all random effect terms to zero. The `use.ranef=FALSE` flag does this for the `lrAcc()` function. Classification accuracy for an average participant/item is:

```
lrAcc(lrMod1, 'stressshift', use.ranef=FALSE)
```

```
## [1] 0.7879581
```

Thus, classification accuracy still improves over the baseline (14% increase), but by less than if computed over the participants/items in this dataset.

Note that all these measures are **observation-level**: they give the percent of observations classified correctly. This number is easy to report, but it should be remembered that we are usually interested in generalizing to new levels of the grouping factors (e.g. new participants and items)—not observations, per se. It is also possible to examine accuracy by participant and item, to get a sense of how well the model will predict an unseen level. We leave an example for a future version of this book.

## 8.8 Miscellaneous mixed-effects regression topics

There are two important practical issues that arise frequently while fitting mixed-effects models:

1. Factors and random effects

- The same issue arises for MELR models as for linear mixed-effects models, discussed in Sec. 7.10: problems arise if you want to use factors in models with uncorrelated random effects. An example for MELR models is in an appendix.

2. Correlations among random effects

- We discuss this issue next.

### 8.8.1 Random-effect correlation issues

Let's consider Model 1 with correlations between random effects added, which we call **Model 2**:

```
lrMod2 <- glmer(stressshift ~ clabel.williams + npType.pron + voice.passive +
  (1 + clabel.williams + npType.pron|item) +
  (1 + clabel.williams + npType.pron + voice.passive|participant),
  data=givenness, family="binomial", control=glmerControl(optimizer = "bobyqa"))
```

The model's random-effect terms are:

```
summary(lrMod2)$varcor
```

## Groups	Name	Std.Dev.	Corr
## participant	(Intercept)	0.67835	
##	clabel.williams	1.19637	-0.726
##	npType.pron	1.30652	-0.218 0.657
##	voice.passive	1.32666	-0.141 -0.094 -0.796
## item	(Intercept)	0.45694	
##	clabel.williams	1.51394	-1.000
##	npType.pron	0.85705	1.000 -1.000

Several correlations are 1 or -1, indicating perfect correlation between random-effect terms. This is highly unlikely to reflect reality (e.g. a speaker's William effect is 100% predictable from their rate of stress shifting). Instead, usually **correlations near 1 or -1 signal a problem** in fitting the model. Models with **perfect** random-effect correlations (or where some random effect variances are 0, but this seems to be less of a problem) are “singular fits”: see technical/practical discussion by Ben Bolker here

(Near-)perfect random effect correlations can result from:

1. Uncentered or collinear predictors
2. A problem in how the model is specified
3. There not being enough data to estimate the correlation terms.

Case 1 is sometimes resolvable by transforming your data: centering continuous variables and using contrast-coding schemes where contrasts are sum-to-zero and orthogonal. Case 2 means you did something wrong—such as trying to fit an uncorrelated random effect for a factor  $X$  using the formula  $(1+X|participant)$ , or using predictors that are linearly dependent in the random-effect formula.

In Case 3, the model is said to be *overparametrized*: the random-effect structure of the model is too complex for the amount of data—either in terms of observations, or the number of levels of a grouping factor. That is, too many random effect terms are being estimated given the dataset size. (Intuitively: if there are 10 speakers and 10 random effect terms, the model is roughly trying to fit 10 variables using 10 numbers—for which there will be no unique ‘best’ solution.)

Overparametrization is very common (in 2018) in linguistic data, and is probably responsible for the perfect correlations in Model 2:

- There are 16 items, from which 6 by-item random-effect terms are being estimated (1 intercept, 2 slopes, 3 correlations).
- There are 27 participants, from which 10 by-participant “ “ (1 intercept, 3 slopes, 6 correlations).

The issue of high random effect correlations:

- occurs easily (in 2018) for the dataset sizes/number of grouping factor levels typical in linguistic data.
- is often problematic (e.g. can cause Type I errors), even if the model converges.

It is not clear that there is a general solution to deal with this issue, without moving to Bayesian mixed-effects models (see below). One “backwards elimination” option, similar to that recommended by Barr et al. (2013), is:

1. Iteratively remove correlations until remaining correlations are sensible.
2. Check using model comparison that the difference in likelihood ratio from the model with all correlations (even if it didn’t converge) isn’t significant.

An alternative “forward” method is:

1. Start with the no-correlations model (with all possible random slopes).
2. Add correlations iteratively, from examining pairwise plots of random effects, using model comparison.

Ben Bolker’s FAQ here gives some other possibilities.

The “backwards” and “forwards” strategies often work, but they are ad-hoc solutions. We first give an example of the “backward” strategy, then show an alternative (fit a Bayesian MELR model) that is cleaner, at the expense of fitting a model whose architecture is quite different from the original (problematic) model.

### 8.8.1.1 Example: Dropping correlation terms

First, drop one of the perfect correlations:

```
1rMod2.red1 <- update(1rMod2, . ~ . - (1+clabel.williams+npType.pron|item) + (1+clabel.williams|item) +
```

The resulting model still has a perfect correlation among the by-item random effects:

```
summary(1rMod2.red1)$varcor
```

## Groups	Name	Std.Dev.	Corr
## participant	(Intercept)	0.60099	
##	clabel.williams	1.05584	-0.642
##	npType.pron	1.30842	-0.070 0.655
##	voice.passive	1.29183	-0.201 -0.157 -0.827
## item	(Intercept)	0.41505	

```
##          clabel.williams 1.40683 -1.000
##  item.1      npType.pron     0.70982
```

Next, drop this correlation:

```
lrMod2.red2 <- update(lrMod2, . ~ . - (1+clabel.williams+npType.pron|item) + (1|item) + (0+clabel.willi...
```

The resulting model has no more near-perfect correlations:

```
summary(lrMod2.red2)$varcor
```

```
## Groups      Name      Std.Dev. Corr
## participant (Intercept) 0.54756
##           clabel.williams 1.02080 -0.549
##           npType.pron     1.33817  0.007  0.644
##           voice.passive   1.30948 -0.327 -0.082 -0.807
## item       (Intercept) 0.00000
## item.1     clabel.williams 1.13985
## item.2     npType.pron     0.70400
```

Finally, do a likelihood ratio test comparing the models with and without the correlation terms:

```
anova(lrMod2, lrMod2.red2)
```

```
## Data: givenness
## Models:
## lrMod2.red2: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod2.red2: (1 + clabel.williams + npType.pron + voice.passive | participant) +
## lrMod2.red2: (1 | item) + (0 + clabel.williams | item) + (0 + npType.pron |
## lrMod2.red2: item)
## lrMod2: stressshift ~ clabel.williams + npType.pron + voice.passive +
## lrMod2: (1 + clabel.williams + npType.pron | item) + (1 + clabel.williams +
## lrMod2: npType.pron + voice.passive | participant)
##          Df    AIC    BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## lrMod2.red2 17 352.88 419.95 -159.44   318.88
## lrMod2      20 353.31 432.22 -156.66   313.31 5.5704      3     0.1345
```

The models do not differ significantly ( $\Delta D = 5.6$ ,  $df = 3$ ,  $p = 0.13$ ), suggesting that removing the correlation terms cannot have made much difference. Crucially, none of the fixed-effect terms—which are usually of primary interest—have qualitatively changed.

We don't give an example here of the “forward strategy”. What's crucial to remember when worrying about random-effect correlations is that **whether/which correlations added is less important than including random slopes**, as we discussed for LMEMs in Sec. 7.8.2.

### 8.8.1.2 Example: Bayesian MEMs

Another way to deal with overparametrized models is to fit a different class of model entirely: a *Bayesian mixed-effects model*. In Bayesian regression models, each parameter has a “prior distribution” specifying how likely different parameter values are thought to be, a priori; these distributions are updated using the actual data, resulting in a “posterior distribution”, which describes how likely different parameter values are given the observed data *and* the prior. These distributions can be summarized to give familiar-looking model output: a most-likely value for each parameter (the “fixed-effect estimate”: usually the posterior mode (MAP)), a “*p*-value” of sorts estimated by where 0 lies on the posterior distribution for a parameter, and so on.

The crucial aspect of Bayesian models for present purposes is that with properly chosen priors, they cannot give a non-nonsensical result due to overparametrization (perfect random-effect correlations)—having less

data to estimate a parameter just means the prior will have more influence on the posterior distribution for that parameter (and hence its MAP estimate).<sup>5</sup>

For Bayesian mixed-effects models, the standard prior (Wishart) used for the random-effects covariance matrix (all random effects variances + correlations) is an “informative” prior where correlation parameters closer to 0 are progressively more likely. This pulls the estimates for these correlation parameters towards zero, and avoids perfect (1 or -1) values.

The `blme` package extends the `lme4` package to fit Bayesian mixed-effects models (Chung et al. (2013)). The `lmer` and `glmer` functions (from the `lme4` package) become functions called `blmer` and `bglmer`. Particular choices of prior for the fixed and random effects are made by default, and you can just use the same model formula as for your `lmer` and `glmer` model, adding a `b`. For our Model 2:

```
library(blme)

lrMod2.bayesian <- bglmer(stressshift ~ clabel.williams + npType.pron + voice.passive +
  (1 + clabel.williams + npType.pron|item) +
  (1 + clabel.williams + npType.pron + voice.passive|participant),
  data=givenness, family="binomial", control=glmerControl(optimizer = "bobyqa"))
```

The resulting model has no near-perfect correlations:

```
summary(lrMod2.bayesian)

## Cov prior : participant ~ wishart(df = 6.5, scale = Inf, posterior.scale = cov, common.scale = TRUE)
##           : item ~ wishart(df = 5.5, scale = Inf, posterior.scale = cov, common.scale = TRUE)
## Prior dev  : -2.976
##
## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [bglmerMod]
## Family: binomial ( logit )
## Formula: stressshift ~ clabel.williams + npType.pron + voice.passive +
##           (1 + clabel.williams + npType.pron | item) + (1 + clabel.williams +
##           npType.pron + voice.passive | participant)
## Data: givenness
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC  logLik deviance df.resid
## 360.5    439.4   -160.3    320.5     362
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -2.05023 -0.24401 -0.07532  0.29963  2.57711
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   participant (Intercept) 1.1974   1.0943
##             clabel.williams 3.1481   1.7743  -0.62
##             npType.pron     3.4775   1.8648  -0.18  0.50
##             voice.passive   3.7961   1.9484   0.03  0.06 -0.68
##   item       (Intercept) 0.7312   0.8551
##             clabel.williams 4.6235   2.1502  -0.79
##             npType.pron     1.9166   1.3844   0.56 -0.73
##   Number of obs: 382, groups: participant, 27; item, 16
```

---

<sup>5</sup>That said, Bayesian MEMs can still be overparametrized (too complex for the data)—the resulting model will be more influenced by the priors, and will not generalize well to new data. See Bates et al. (2015) for discussion.

```

## 
## Fixed effects:
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.9235    0.5112 -3.763 0.000168 ***
## clabel.williams 5.8044    1.1396  5.093 3.52e-07 ***
## npType.pron    0.9937    0.6655  1.493 0.135425
## voice.passive  1.2633    0.6879  1.836 0.066286 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## 
## Correlation of Fixed Effects:
##          (Intr) clbl.w npTyp.
## clabl.wllms -0.799
## npType.pron  0.027  0.011
## voice.passv -0.151  0.235 -0.199

```

This seems like a great solution—no need for hackily removing terms, as in the “backwards” method above. You should just be aware that these (Bayesian) models are quite different from non-Bayesian ME models, and it’s up to you whether you are comfortable using them without learning more about how they work. The resulting models are not simply (g)lmer fits with different correlation parameters—as you can see by comparing the fixed and random effects for lrMod2.bayesian (above) and lrMod2:

```

print(summary(lrMod2), corr=FALSE)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: stressshift ~ clabel.williams + npType.pron + voice.passive +
##           (1 + clabel.williams + npType.pron | item) + (1 + clabel.williams +
##           npType.pron + voice.passive | participant)
## Data: givenness
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC      logLik deviance df.resid
## 353.3    432.2    -156.7     313.3      362
##
## Scaled residuals:
##      Min      1Q Median      3Q      Max
## -2.1302 -0.2737 -0.1186  0.4182  3.7528
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   participant (Intercept) 0.4602   0.6783
##             clabel.williams 1.4313   1.1964  -0.73
##             npType.pron     1.7070   1.3065  -0.22  0.66
##             voice.passive   1.7600   1.3267  -0.14 -0.09 -0.80
##   item       (Intercept) 0.2088   0.4569
##             clabel.williams 2.2920   1.5139  -1.00
##             npType.pron     0.7345   0.8571   1.00 -1.00
## Number of obs: 382, groups: participant, 27; item, 16
##
## Fixed effects:
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.5286    0.3601 -4.245 2.19e-05 ***
## clabel.williams 4.5928    0.8186  5.611 2.02e-08 ***

```

```

## npType.pron      0.8047    0.4997   1.610   0.1073
## voice.passive   0.9398    0.4722   1.990   0.0466 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

- Fixed effects estimates/SEs consistently larger/smaller in the Bayesian model
- Random effect variances/correlations consistently further/closer to 0 in the Bayesian model

It is possible that the Bayesian model is actually giving more accurate estimates—the point here is that it gives **different** estimates from the `(g)lmer` model.

## 8.9 Other readings

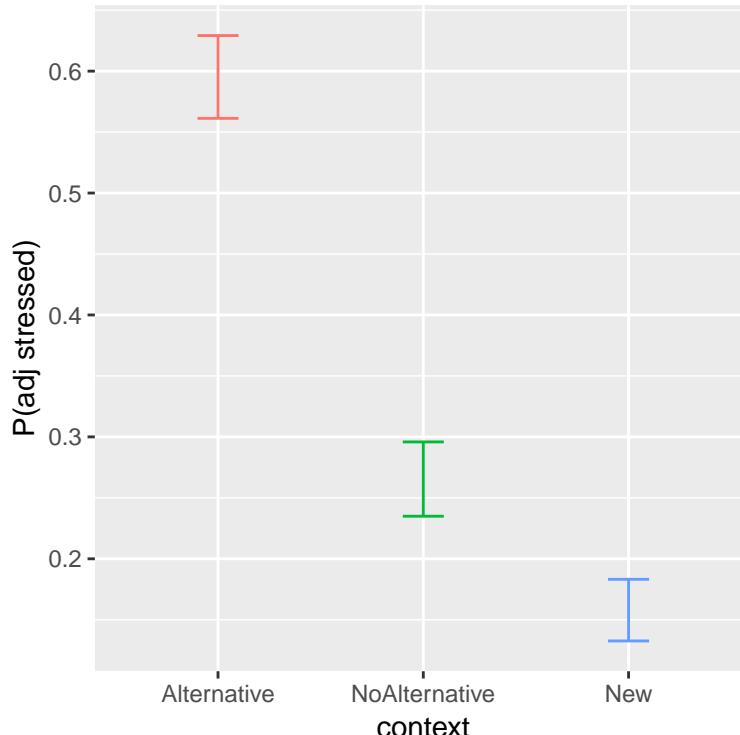
On MELR models: Ch. 14 of Gelman and Hill (2007) is a good general reference, while Jaeger (2008) and Ch 7.4 of Johnson (2008) are shorter introductions for language scientists in particular.

To learn more about Bayesian MEMs: Sorensen et al. (2015) and Nicenboim and Vasishth (2016) give (longer/shorter) introductions for language scientists. Bayesian MEMs are one type of Bayesian data analysis, a broader set of methods to which Kruschke (2014) and McElreath (2015) give accessible introductions using R.

## 8.10 Appendices

### 8.10.1 Appendix: Random slopes for factors

How to deal with factors with more than 2 levels is an important issue in practical applications of MELR models. We show an example using the `alternatives` dataset. Recall the basic results, for how the probability of shifting stress (to `prominence = Adjective`, from `Noun`) depends on `context`:



The probability of shifting stress decreases from *Alternative* to *NoAlternative* to *New*. We previously examined this data using a logistic regression, and will now use a mixed-effects logistic regression model.

In this model:

- The response is **prominence** (*adjective* or *noun*)
- The predictor is **context** (*Alternative*, *NoAlternative*, *New*)
- We include all possible random slopes and intercepts (by-participant and by-item), **with** correlations

```
summary(alternativesMod1)
```

```
## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: prominence ~ context + (1 + context | item) + (1 + context |
##           participant)
## Data: alternatives
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC  logLik deviance df.resid
## 654.6    721.1   -312.3    624.6     607
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -3.5691 -0.5550  0.2148  0.5648  2.5891
##
## Random effects:
##   Groups      Name        Variance Std.Dev. Corr
##   participant (Intercept) 0.6585734 0.81153
##             context1    0.0917516 0.30291   0.06
##             context2    0.0169077 0.13003  -0.24  0.95
##   item       (Intercept) 0.6781250 0.82348
##             context1    0.0008455 0.02908  0.81
##             context2    0.4283623 0.65449  0.62  0.96
## Number of obs: 622, groups: participant, 18; item, 12
##
## Fixed effects:
##            Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.2516    0.3652   3.427  0.00061 ***
## context1     0.8659    0.1492   5.803  6.5e-09 ***
## context2     0.8132    0.2662   3.054  0.00226 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cntxt1
## context1  0.092
## context2  0.566  0.062
```

The random-effect correlations in this model have the issue discussed above: some correlations are near perfect ( $r \approx 0.95$ ), indicating that the model is probably overparametrized. We can deal with this by omitting the correlation terms.

### 8.10.2 Appendix: Multi-level factors and uncorrelated random effects

(Only look through this section if you have a factor with multiple levels and you run into correlations that are too high.)

As with linear mixed-effects models, in order to fit a model with no correlations between random effects, any factor must be converted into numeric variables corresponding to the contrasts (the notation `||` will not work when a factor has >2 levels!):

```
alternatives$context1 <- model.matrix(~context, alternatives)[,2]
alternatives$context2 <- model.matrix(~context, alternatives)[,3]
```

which does manually what R does under the hood.

The model with uncorrelated random effects is then:

```
alternativesMod2 <- glmer(shifted ~ context +
  (1+context1+context2||item) +
  (1+context1+context2||participant),
  data=alternatives,
  family="binomial",
  control=glmerControl(optimizer = "bobyqa"))

summary(alternativesMod2)

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##   Family: binomial ( logit )
## Formula: shifted ~ context + (1 + context1 + context2 || item) + (1 +
##       context1 + context2 || participant)
##   Data: alternatives
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC  logLik deviance df.resid
##      646.8    686.7   -314.4     628.8      613
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -2.9286 -0.5564 -0.2484  0.5676  3.5763
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   participant context2    0.00000  0.0000
##   participant.context1 0.06545  0.2558
##   participant.context2 (Intercept) 0.67379  0.8208
##   item          context2    0.29399  0.5422
##   item.context1    context1    0.00000  0.0000
##   item.context2    (Intercept) 0.56035  0.7486
##   Number of obs: 622, groups: participant, 18; item, 12
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.1533    0.3199  -3.605 0.000312 ***
## context1     -0.8716    0.1375  -6.338 2.32e-10 ***
## context2     -0.7014    0.1909  -3.673 0.000240 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

## 
## Correlation of Fixed Effects:
##          (Intr) cntxt1
## context1  0.054
## context2  0.129  0.018

Comparing the models with and without correlations:

anova(alternativesMod1, alternativesMod2)

## Data: alternatives
## Models:
## alternativesMod2: shifted ~ context + (1 + context1 + context2 || item) + (1 +
## alternativesMod2:      context1 + context2 || participant)
## alternativesMod1: prominence ~ context + (1 + context | item) + (1 + context |
## alternativesMod1:      participant)
##              Df AIC   BIC logLik deviance Chisq Chi Df
## alternativesMod2 9 646.83 686.73 -314.42   628.83
## alternativesMod1 15 654.56 721.05 -312.28   624.56 4.2702      6
##                  Pr(>Chisq)
## alternativesMod2
## alternativesMod1     0.6402

```

The difference in likelihood between the models is small and not significant ( $\Delta D = 1.94$ ,  $df = 6$ ,  $p = 0.92$ ). In addition, the fixed-effect estimates and standard errors are very similar in the two models. Thus, it is reasonable to omit the random-effect correlations.

As discussed for linear mixed-effects models in Section 7.10:

- To include a random slope for any factor (even if just 2 levels) in a model with uncorrelated random effects, you need to code as numeric variable(s).
- For a factor with 2 levels, you can just use a centered version (e.g. `arm::rescale`), which is essentially sum coding.
- For factors with more levels (or to use dummy coding for a two-level factor), you manually extract the contrasts to make numeric variables.

### 8.10.3 Appendix: What can happen if a random slope isn't included?

We show an example illustrating the potential danger from not including a random slope term, using the `givenness` data.

Let's fit a logistic mixed-effects model for this data which only contains random intercept terms (no random slopes):

```

mod1.3 <- glmer(stressshift ~ clabel.williams + npType.pron + voice.passive +
  (1|item) +
  (1|participant),
  data=givenness,
  family="binomial")

```

this model is the same as Model 1, but without random slopes.

The fixed effect estimates are:

```
summary(mod1.3)$coefficients
```

#	Estimate	Std. Error	z value	Pr(> z )
---	----------	------------	---------	----------

```
## (Intercept) -1.0110841 0.1644734 -6.147402 7.876258e-10
## clabel.williams 3.1918034 0.3288389 9.706283 2.834878e-22
## npType.pron 0.6148035 0.2767721 2.221335 2.632827e-02
## voice.passive 0.7639485 0.2781885 2.746154 6.029846e-03
```

Compare these estimates to those from Model 1 (which includes random-slope terms):

```
summary(lrMod1)$coefficients
```

	Estimate	Std. Error	z value	Pr(> z )
## (Intercept)	-1.1383258	0.2076053	-5.483124	4.178800e-08
## clabel.williams	3.7710505	0.5675843	6.644036	3.052075e-11
## npType.pron	0.7917211	0.3987366	1.985574	4.708063e-02
## voice.passive	0.6496316	0.4149409	1.565600	1.174422e-01

### Questions:

- What term(s) differ greatly in significance between the two models?
- What kind of variability does this suggest exists?
  - (Among subjects, or items?)

Let's look at the random effects for the two models:

1. Random effects for the random intercepts-only model:

```
print(VarCorr(mod1.3), comp=c("Variance", "Std.Dev.))
```

Groups	Name	Variance	Std.Dev.
## participant	(Intercept)	0.039786	0.19946
## item	(Intercept)	0.000000	0.00000

2. Random effects for the model with random slopes (Model 1):

```
print(VarCorr(lrMod1), comp=c("Variance", "Std.Dev.))
```

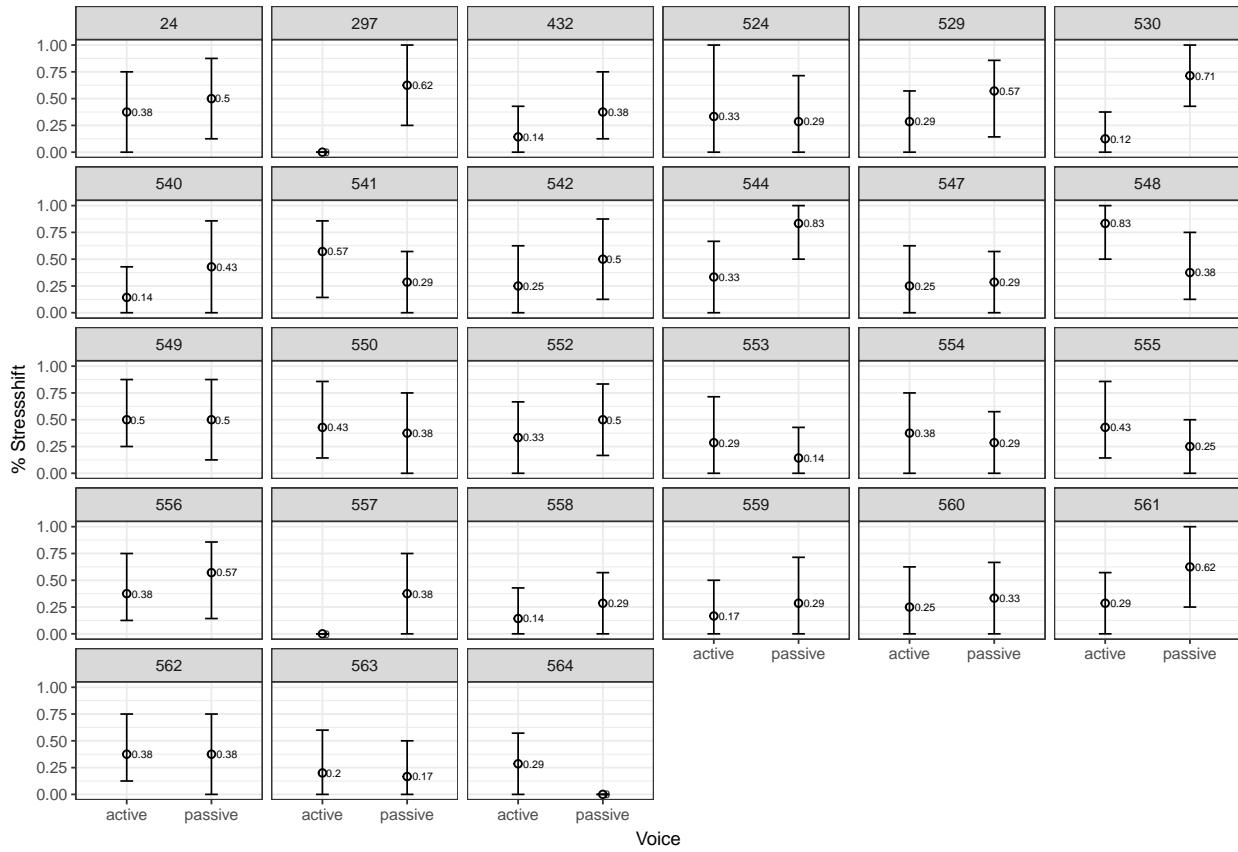
Groups	Name	Variance	Std.Dev.
## participant	voice.passive	1.6482e+00	1.2838e+00
## participant.1	npType.pron	5.8230e-01	7.6308e-01
## participant.2	clabel.williams	5.0132e-15	7.0804e-08
## participant.3	(Intercept)	1.6855e-01	4.1054e-01
## item	npType.pron	5.3942e-01	7.3445e-01
## item.1	clabel.williams	1.6193e+00	1.2725e+00
## item.2	(Intercept)	2.6237e-14	1.6198e-07

The effects of adding random slopes to this model are:

- The standard errors for all fixed-effect terms increase
- **voice.passive loses significance**

To see why this effect loses significance, we examine an (empirical) by-participant plot of the `voice.passive` effect:

```
ggplot(givenness, aes(x = voice, y = stressshift.num)) + stat_summary(fun.y=mean, geom="point", size=1, shad
```



Participants vary greatly in the size and direction of the effect, and no participant except 530 shows a robust effect.

The by-participant random slope for `voice` in the model with random slopes is  $\hat{\sigma}_{\text{voice}, \text{partic}} = 1.3$ . This is large, compared to the size of the fixed-effect for this variable ( $\hat{\beta}_{\text{voice}} = 0.65$ ).

In other words: in this data, there is too much by-participant variability in `voice` effect to say with confidence that there is an **overall** significant effect (across participants).

But without a random slope term, the model was unduly influenced by one participant who does show an effect, leading us to spuriously conclude that there is a significant ( $p < 0.05$ )  $\hat{\beta}_{\text{voice}}$  effect—a Type I error.

## 8.11 Solutions

**Question about tapping data:** Does it look like there is by-participant variability in the intercept, slope, or both? And by-item variability?

**A:** It looks like there is some by-participant variability in the intercepts, and also in the slopes; it looks like there is almost no variability in the by-item intercepts, there is quite a bit in the slopes.

**Question about givenness data:** Again, for participants and items: do you think there is variability in the intercept? Slope? Both?

**A:** There is some variability in the by-participant intercept and very little in the slope; there is almost no variability in the by-item intercept and some in the slope.

**Exercise 1: tapping:**

The model is:

```
ex1.mod <- glmer(tapped ~ speechrate.slow + syntax.trans + (1 + syntax.trans + speechrate.slow || participant + item) + (1 + syntax.trans + speechrate.slow || item), data = tapped)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: tapped ~ speechrate.slow + syntax.trans + (1 + syntax.trans +
##           speechrate.slow || participant) + (1 + syntax.trans + speechrate.slow || item)
## Data: tapped
##
##      AIC      BIC  logLik deviance df.resid
##  330.0   371.2  -156.0    312.0     706
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -2.4530 -0.2026 -0.0697 -0.0302  4.1061
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   participant (Intercept) 2.011e+00 1.418e+00
##   participant.slope syntax.trans 0.000e+00 0.000e+00
##   participant.item speechrate.slow 1.272e+01 3.566e+00
##   item          (Intercept) 3.174e-01 5.634e-01
##   item.slope     syntax.trans 4.284e-01 6.545e-01
##   item.item     speechrate.slow 5.115e-10 2.262e-05
## Number of obs: 715, groups: participant, 23; item, 8
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -4.3843    0.6155 -7.123 1.06e-12 ***
## speechrate.slow -4.0897    1.0355 -3.950 7.83e-05 ***
## syntax.trans  1.0909    0.4209  2.592  0.00955 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) spchr.
## spechrt.slw  0.419
## syntax.trns -0.156 -0.068
```

Both `syntax.trans` and `speechrate.slow` are observation level: they both vary within participants and items, as you can check with e.g. `xtabs(~syntax + participant, data=tapped)`

The model assumes that the log-odds of tapping depend on both speech rate and syntax, and that the slope of both predictors can vary between participants and items, who also differ in their “baseline” tapping rates. It does not assume that there may be an interaction between syntax and speech rate.

Participants vary more than items in the intercept (random effect  $\sigma = 0.56$  versus 1.4).

The variance component for the by-participant `syntax.trans` random slope, 0, is how much participants are predicted to vary in the slope of `syntax.trans`.

In the empirical plots, it looked like:

- with respect to **syntax**, participants vary in intercept, maybe in slope.
- items vary little in intercept (less than participant) if at all, but vary a lot in slope.
- Across participants and items, there is a positive effect of **syntax**—many more participants/items show a positive slope than a negative one.

The model results are mostly consistent with this data:

- By-participant random intercept variance > by-item random intercept variance
- There is substantial by-item variation the **syntax.trans** effect—the by-item  $\sigma$  for the random slope (0.65) is comparable to the fixed effect size (1.09). However, there is no by-participant variation predicted in the **syntax.trans** effect, which seems odd.



# Chapter 9

## Practical regression topics 2: Ordered factors, nonlinear effects, model predictions, post-hoc tests

### Preliminary code

This code is needed to make other code below work:

```
library(gridExtra) # for grid.arrange() to print plots side-by-side
library(dplyr)
library(ggplot2)
library(languageR)
library(scales)
library(rms)
library(arm)
library(lsmeans)

## loads votMcGillLing620.csv from OSF project for Sonderegger et al. (2017) data
vot <- read.csv(url("https://osf.io/qpab9/download"))

## loads halfrhymeMcGillLing620.csv from OSF project for Harder (2013) data
halfrhyme <- read.csv(url("https://osf.io/37uqt/download"))

## loads alternativesMcGillLing620.csv from OSF project for Wagner (2016) data
alternatives <- read.csv(url("https://osf.io/6qctp/download"))

## remove rows where response is NA (shouldn't be any...)
alternatives <- filter(alternatives, !is.na(prominence))

## add the 'shifted' numeric variable
alternatives <- alternatives %>% mutate(shifted = -1*as.numeric(prominence)+2)

## relevel context to be in the intuitively plausible order
alternatives <- mutate(alternatives, context=as.ordered(factor(context, levels=c("Alternative", "NoAlternative"))))

## regularity dataset:
## order levels for Auxiliary in their natural order
```

```

regularity$Auxiliary <- factor(regularity$Auxiliary, levels=c("hebben", "zijnheb", "zijn"))

## loads givennessMcGillLing620.csv from OSF project for Wagner (2012) data
givenness <- read.csv(url("https://osf.io/q9e3a/download"))

givenness <- mutate(givenness,
  conditionLabel.williams = arm::rescale(conditionLabel),
  clabel.williams = arm::rescale(conditionLabel),
  npType.pronoun = arm::rescale(npType),
  npType.pron = arm::rescale(npType),
  voice.passive = arm::rescale(voice),
  order.std = arm::rescale(order),
  stressshift.num = (as.numeric(stressshift) - 1)
)

```

**Note:** Answers to some questions/exercises not listed in text are in Solutions

## 9.1 Introduction

These notes cover several practical topics which come up in fitting regression models (mixed-effects or not):

- Ordered factors
- Nonlinear effects
- Making model predictions

We also briefly discuss:

- Post-hoc tests
- Multiple comparisons

## 9.2 Ordered factors

So far we have considered two types of variables as predictors in regression models.

First: **numeric** variables, which are continuous and *ordered*, meaning that there are “larger” and “smaller” values of the variable. When a numeric variable  $X$  is used as a predictor in a regression model, it is assumed that a unit change always has the same effect on the response  $Y$  (the “slope”): increasing  $X$  from 1 to 2 has the same effect on  $Y$  as increasing  $X$  from 3 to 4.

Second: **factors**, which are discrete and unordered: no level of a factor is “larger” than other levels.<sup>1</sup> When a factor is used as a predictor in a regression model, it is not assumed that changing from level 1 to level 2 has the same effect on the response as changing from level 2 to level 3.

**Ordered factors** lie in between these two types of variables. They are discrete, like factors, but ordered, like continuous variables. It is assumed that level 1 is conceptually “less than” level 2, and so on (like a continuous variable), but it is not assumed that level 1 → 2 has the same effect as level 2 → 3 (like a factor). Ordered factors are often used for variables where the levels can be thought of as lying on a scale, and take on few values (~3–6).

---

<sup>1</sup>This may be confusing, since levels of a factor are often discussed as if they do have an order: “level 1”, “level 2”, and so on. This ordering is used to define contrasts (e.g. which factor level is the “base level”), and is assumed by R when making plots involving the factor, etc. However, this ordering is *arbitrary*—there is no sense in which level 1 is inherently “less” than level 2.

### Example 1

Consider the Dutch verb regularity dataset, `regularity`.

- The **response** variable is `Regularity`: whether the verb has a regular or irregular past tense (1/0)
- The **predictor** of interest is `Auxiliary`: which auxiliary a verb takes (*hebben*, *zijnheb*, or *zijn*)
  - These levels have a natural order: *zijnheb* lies between the other two (because it means “this verb can take either *zijn* or *hebben* as an auxiliary.”)

Thus, we convert `Auxiliary` to an ordered factor:

```
## make ordered factor for auxiliary
regularity$Auxiliary.ord <- as.ordered(regularity$Auxiliary)
head(regularity$Auxiliary.ord)
```

```
## [1] hebben zijnheb zijn    hebben hebben hebben
## Levels: hebben < zijnheb < zijn
```

Note that the R output shows this is an ordered factor by showing the ordering of levels with `<`. Compare to the R output for an unordered factor:

```
head(regularity$Auxiliary)
```

```
## [1] hebben zijnheb zijn    hebben hebben hebben
## Levels: hebben zijnheb zijn
```

#### 9.2.1 Orthogonal polynomial contrasts

What “converting to an ordered factor” (the `as.ordered` command) actually means is using a particular contrast coding scheme (See Sec. 6.2.3 for details): *orthogonal polynomial contrasts*. For three levels, the contrasts are:

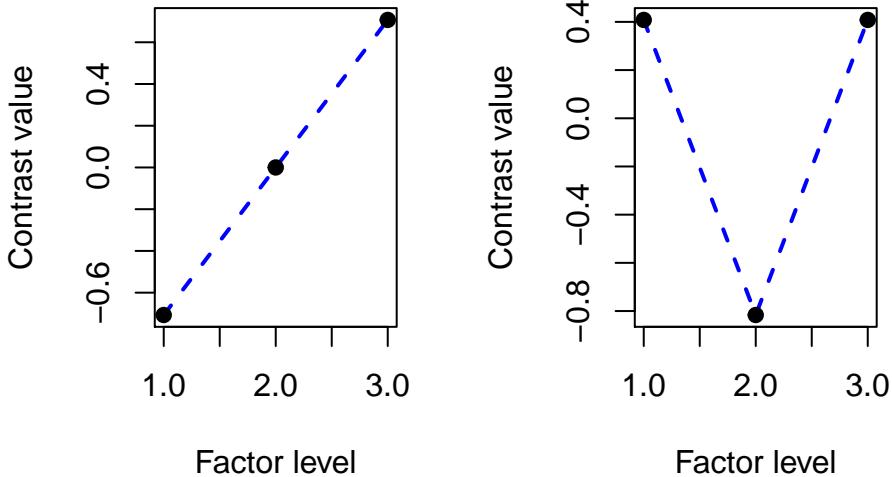
```
round(contrasts(regularity$Auxiliary.ord), 3)
```

```
##          .L      .Q
## [1,] -0.707  0.408
## [2,]  0.000 -0.816
## [3,]  0.707  0.408
```

The two contrasts correspond to “linear” and “quadratic” *trends* (.L and .Q columns), which represent different kinds of relationship between the factor and the response, **if** the levels were treated as equally-spaced (and continuous):

- Linear: how much does relationship with response look like a line?
- Quadratic: how much does relationship with the response look like a parabola?

The contrasts for  $k = 3$  can be visualized as:



For ordered factors with more levels, further contrasts correspond to a cubic trend, and so on. For a four-level ordered factor:

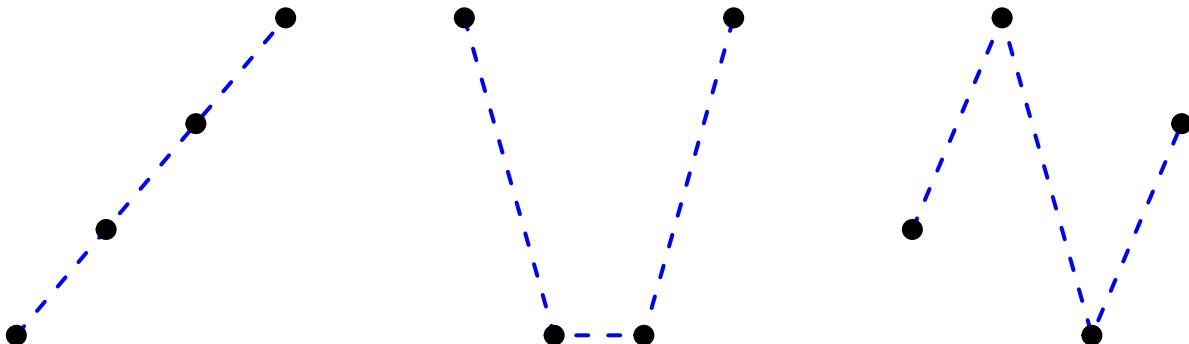
```
contr.poly(4)
```

```
##          .L     .Q      .C
## [1,] -0.6708204  0.5 -0.2236068
## [2,] -0.2236068 -0.5  0.6708204
## [3,]  0.2236068 -0.5 -0.6708204
## [4,]  0.6708204  0.5  0.2236068
```

the three contrasts capture:

- L: how much does relationship with response look like a line?
- Q: “ “ a parabola?
- C: “ “ a cubic function?

Visually, the contrasts for four levels ( $k = 4$ ) are:



where axes have been removed for clarity.

### 9.2.2 Using an ordered factor as a predictor

For our example, we predict Regularity as a function of the verb's auxiliary (Auxiliary.ord) using a logistic regression:

```
regularity.mod.1 <- glm(Regularity ~ Auxiliary.ord, data=regularity, family="binomial")
summary(regularity.mod.1)
```

```

## 
## Call:
## glm(formula = Regularity ~ Auxiliary.ord, family = "binomial",
##      data = regularity)
## 
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max 
## -1.8307  0.6438  0.6438  0.6438  1.3537 
## 
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)    
## (Intercept) 0.51944   0.17029  3.050  0.00229 **  
## Auxiliary.ord.L -1.32507   0.33145 -3.998 6.39e-05 *** 
## Auxiliary.ord.Q  0.02954   0.25324  0.117  0.90713    
## --- 
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 
## 
## (Dispersion parameter for binomial family taken to be 1)
## 
## Null deviance: 750.12  on 699  degrees of freedom 
## Residual deviance: 719.92  on 697  degrees of freedom 
## AIC: 725.92 
## 
## Number of Fisher Scoring iterations: 4

```

The linear trend has large effect size and is very significant ( $p < 0.0001$ ), while the quadratic trend has low effect size and is not significant ( $p = 0.91$ ). This means the model predicts an essentially linear relationship between `Auxiliary.ord` and `Regularity`.

This example illustrates one use for ordered factors—dimensionality reduction, by reducing the number of contrasts needed to represent a factor’s effect. If we wanted to simplify the model, it would be justified at this point to drop the quadratic trend, and just use the linear contrast to represent the effect of `Auxiliary.ord`.<sup>2</sup> We wouldn’t gain much by doing so here (just one fewer regression coefficient), but when dealing with ordered factors with 5+ levels, dimensionality reduction can make model fitting and interpretation much easier.

To visualize the predicted relationship between auxiliary and regularity probability (in log-odds and probability space):

```

## set up a dataframe which varies in Auxiliary.ord:
newdata <- with(regularity, data.frame(Auxiliary.ord = unique(Auxiliary.ord)))

## predictions at each level of Auxiliary.ord (in log-odds)
newdata$pred <- predict(regularity.mod.1, newdata=newdata)

## predictions in probability
newdata$pred.p <- invlogit(newdata$pred)

regularityPlot1 <- ggplot(aes(x=Auxiliary.ord, y=pred), data=newdata) +
  geom_point(size=2) +
  xlab("Auxiliary (log-odds)") +
  ylab("Predicted % regular (logit)")

regularityPlot2 <- ggplot(aes(x=Auxiliary.ord, y=pred.p), data=newdata) +
  geom_point(size=2) +

```

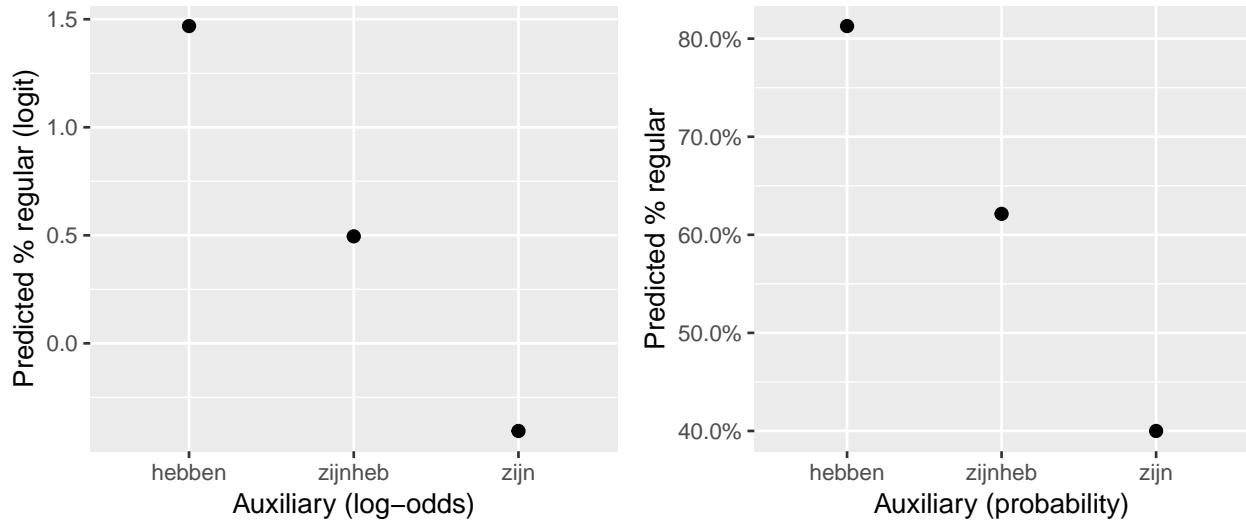
<sup>2</sup>To do this we’d have to use numeric variables for the contrast, as shown for mixed models.

```

xlab("Auxiliary (probability)") +
ylab("Predicted % regular") +
scale_y_continuous(labels=percent)

grid.arrange(regularityPlot1, regularityPlot2, ncol = 2)

```



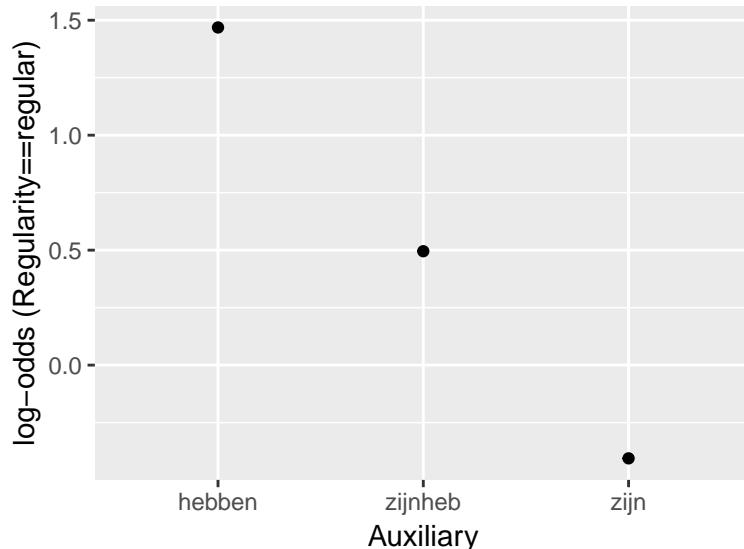
The predicted trend is “linear” in the sense that the three points lie on a line. This replicates the pattern in the empirical data (in log-odds):

```

summDf <- regularity %>% group_by(Auxiliary) %>% summarise(mean=logit(mean(Regularity=='regular')))

ggplot(aes(x=Auxiliary,y=mean), data=summDf) +
  geom_point() +
  ylab("log-odds (Regularity==regular)")

```



### 9.2.3 Further points

Some other variables that could be represented by ordered factors:

- Number of onset consonants in a syllable (Ex:  $1 < 2 < 3$ )
- Socio-economic status (SES) (Ex:  $low < low/mid < high/mid < high$ )
- Strength of prosodic boundary ( $word < small\ phrase < large\ phrase < utterance$ )
- L2 level ( $no\ knowledge < beginner < intermediate < advanced$ )

Note that ordered factors are just factors with a particular (sensible) contrast coding scheme. A model where a conceptually “ordered” variable is entered as a non-ordered factor is not incorrect, the results just may be harder to interpret.

When using ordered factors in a mixed-effects model: because ordered factors are still factors, the issues with including factors in a mixed model discussed in Section 7.10 arise. These issues (frequently overparametrized models, uncorrelated random effects) and solutions are very similar for ordered factors, and are discussed in an Appendix (Sec. 9.7).

#### Example 2

For the `alternatives` data, we model the probability of stress shifting (`shifted`) in a mixed-effects logistic regression, with:

- Fixed effect: `context`
- Random effects: by-item and by-participant intercept and slope

This predictor is conceptually ordered, with levels  $Alternative < NoAlternative < New$  (see the dataset description to understand why). So we can convert it to an ordered factor:

```
alternatives <- mutate(alternatives, context=as.ordered(factor(context, levels=c("Alternative", "NoAlternative", "New"))))
```

It turns out (see Section 9.7) that this model gives perfect random-effect correlations if the random effect structure  $(1+context|participant) + (1|context|item)$  is used, so we instead use uncorrelated random effects. To do so, we first convert `context` to two numeric contrasts, which correspond to the linear and quadratic trends:

```
alternatives <- mutate(alternatives,
                      contextL = model.matrix(~context, alternatives)[,2],
                      contextQ = model.matrix(~context, alternatives)[,3])
```

To fit and summarize the model:

```
alternativesMod2 <- glmer(shifted ~ context +
                           (1+contextL + contextQ||item) +
                           (1+contextL + contextQ||participant),
                           data=alternatives,
                           family="binomial",
                           control=glmerControl(optimizer = "bobyqa"))

summary(alternativesMod2)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula: shifted ~ context + (1 + contextL + contextQ || item) + (1 +
##       contextL + contextQ || participant)
```

```

##      Data: alternatives
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC logLik deviance df.resid
##     652.4    692.3   -317.2     634.4      613
##
## Scaled residuals:
##      Min      1Q Median      3Q      Max
## -2.8990 -0.5248 -0.2875  0.5782  3.5101
##
## Random effects:
## Groups      Name        Variance Std.Dev.
## participant contextQ    2.272e-15 4.766e-08
## participant.1 contextL    2.174e-01 4.663e-01
## participant.2 (Intercept) 6.526e-01 8.078e-01
## item         contextQ    1.723e-01 4.151e-01
## item.1       contextL    9.398e-01 9.694e-01
## item.2       (Intercept) 4.925e-01 7.018e-01
## Number of obs: 622, groups: participant, 18; item, 12
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.0609    0.3040 -3.490 0.000483 ***
## context.L    -1.9482    0.3748 -5.198 2.01e-07 ***
## context.Q     0.2936    0.2196  1.337 0.181273
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cntx.L
## context.L  0.103
## context.Q  0.048  0.108

```

### Questions:

What does the model predict about:

- The overall effect of `context`? (What kind of relationship with log-odds of stress shift?)
- By-participant and by-item variability in the `context` effect?

Comparing the predicted overall effect (for an “average” participant/item) to the empirical means (probability of stress shift for each `context` value):

```

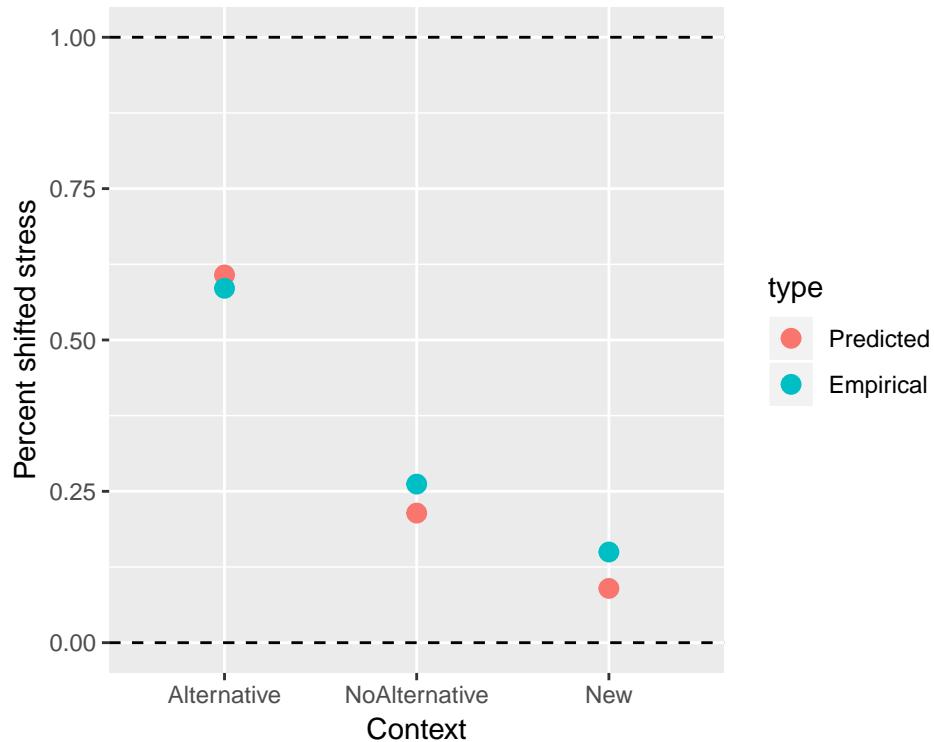
## get model predictions at each level of 'context'
newdata <- with(alternatives, data.frame(context = unique(context)))

## re.form = NA : get predictions at "average" speaker and item values
newdata$pred <- invlogit(predict(alternativesMod2, newdata=newdata, re.form=NA))
summDf2 <- alternatives %>% group_by(context) %>% summarise(pred=mean(shifted))
predEmpDf <- rbind(data.frame(newdata, type='Predicted'),
                     data.frame(summDf2, type='Empirical'))

ggplot(aes(x=context, y=pred), data=predEmpDf) +
  geom_point(aes(color=type), size=3) +
  xlab("Context") +
  ylab("Percent shifted stress") +

```

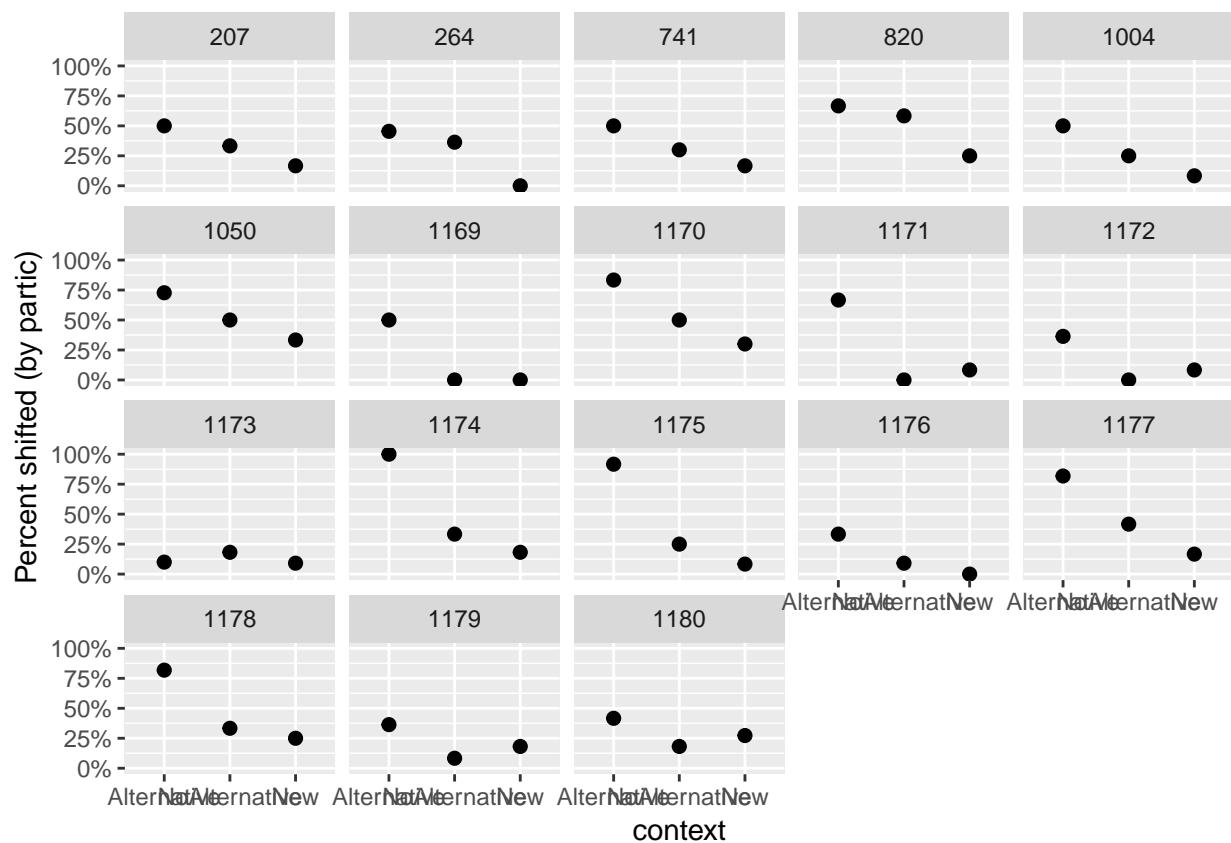
```
geom_hline(aes(yintercept=0), lty=2) +
geom_hline(aes(yintercept=1), lty=2)
```



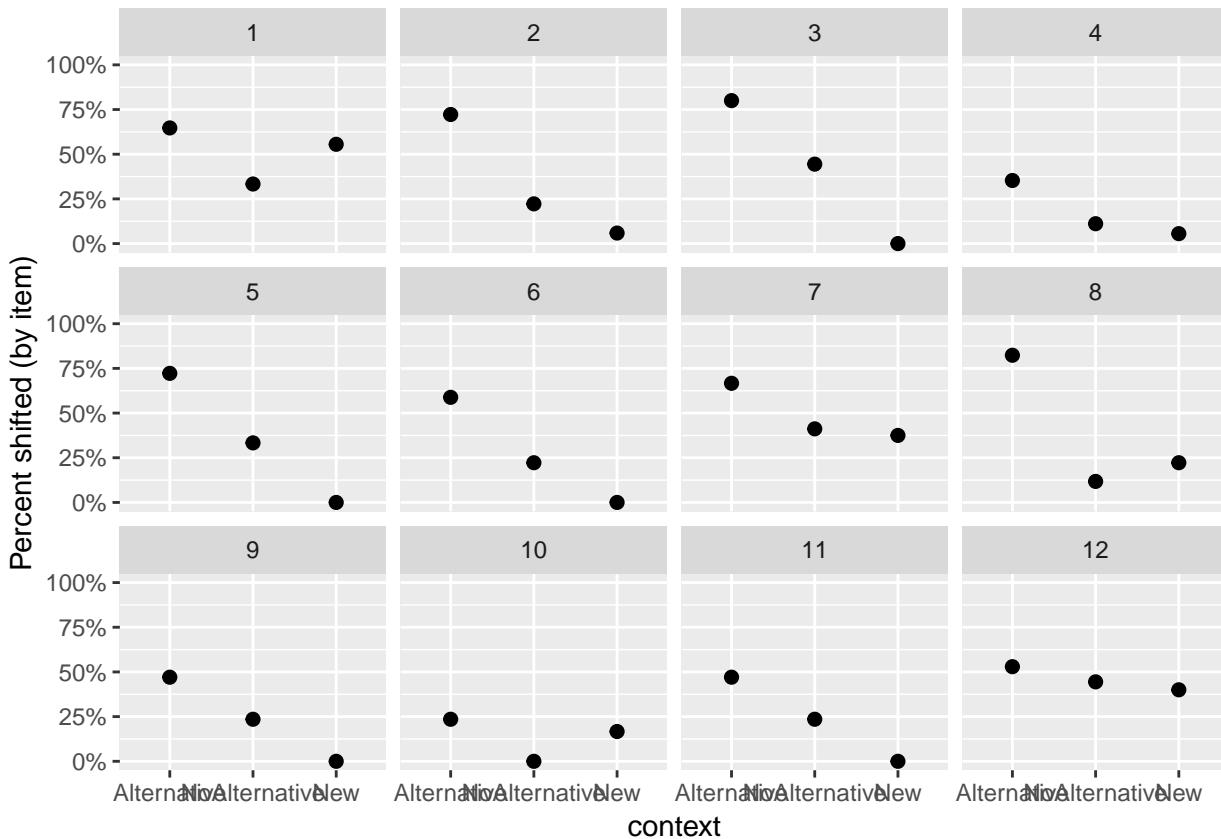
The model's prediction (significant linear trend but not quadratic trend) makes sense. The relationship is basically linear—and would look more so in log-odds space.

Interestingly, the empirical data shows substantial variability among participants and items, many of which do not show a linear trend:

```
alternatives %>% group_by(context, participant) %>%
  summarise(mean=mean(shifted)) %>%
  ggplot(aes(x=context,y=mean)) +
  geom_point(size=2) +
  ylab("Percent shifted (by partic)") +
  facet_wrap(~participant) +
  scale_y_continuous(labels=percent, limits = c(0,1))
```



```
alternatives %>% group_by(context, item) %>% summarise(mean=mean(shifted)) %>%
  ggplot(aes(x=context,y=mean)) +
  geom_point(size=2) +
  ylab("Percent shifted (by item)") +
  facet_wrap(~item) +
  scale_y_continuous(labels=percent, limits = c(0,1))
```



**Questions:**

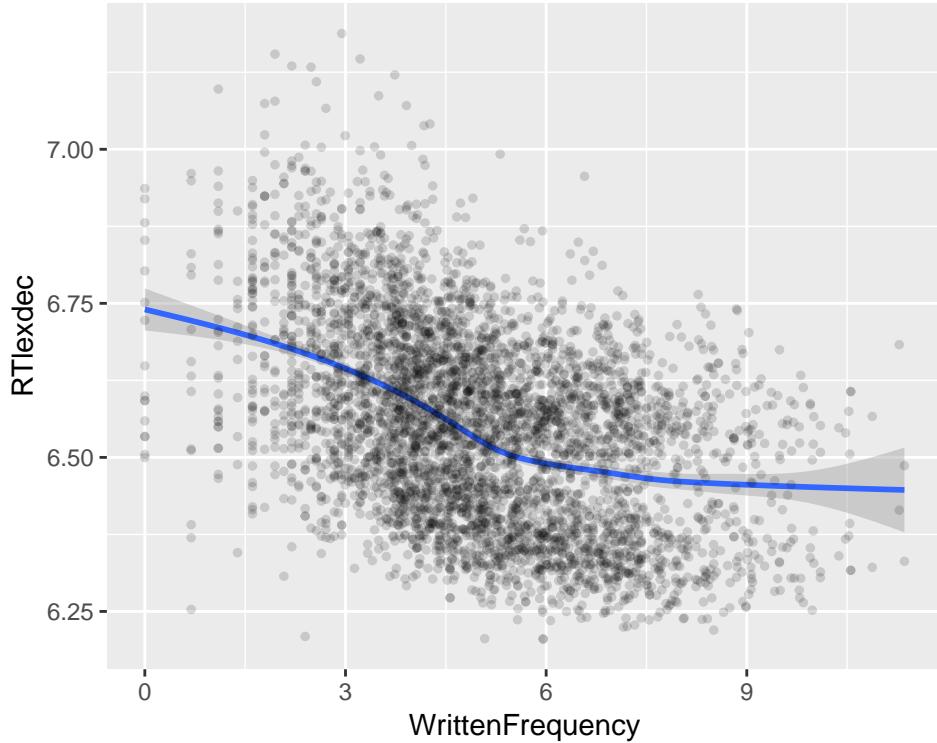
- What kind of variability is observed, among participants and among items? (In overall probability of stress shifting? In the `condition` effect?)
- How do the by-participant and by-item random effect estimates of the model reflect these patterns of variability?

### 9.3 Nonlinear effects

So far we have always assumed that continuous variables have a relationship with the response that is *linear*—well-described by a straight line. In reality this is often not the case.

For example, consider the effect of word frequency (`WrittenFrequency`) on lexical decision reaction time (`RTlexdec`) in the `english` dataset. Plotting this relationship using a nonlinear smoother:

```
ggplot(aes(x=WrittenFrequency, y=RTlexdec), data=english) +
  geom_smooth() +
  geom_point(alpha=0.15, size=1)
```



it is clear the relationship is not linear. Certainly RT decreases as a function of frequency, but the relationship is not a straight line. We would like to model this kind of relationship in a regression model.

There are several options for modeling such *nonlinear functions*, including:

- *Polynomials*, such as a parabola or cubic equation. These are fitted in R by including terms like `poly(x, 2)` in a model (for a quadratic equation).
- *Natural cubic splines*, which you fit in R using `ns()` (e.g. `ns(x, 3)` for a spline with one “bend”)

We will use a related family of functions: *restricted cubic splines* (RCS), which are fitted using the `rcs` function in the `rms` package. Some discussion of RCS is given by Baayen (2008), and more technical discussion by Harrell (2001).

### 9.3.1 Splines: Definition and benefits

While polynomials are probably familiar to you (from high school math: lines, parabolas, etc.) splines are probably not. What are splines, and why should we use them instead of polynomial functions—which are easier to understand?

Intuitively a spline is a function made up of several polynomials glued together (“piecewise-defined”), constrained to be “smooth” at the places where the polynomial pieces connect (called *knots*). The glued-together polynomials can be small pieces of parabolas, lines, and so on.

Splines are like polynomials, but better behaved: they avoid interpolation errors and Runge’s phenomenon. They are better than polynomials especially at the very high and low values of a variable being modeled: polynomials tend to “blow up” when extrapolated beyond the range of the variable in the data, while splines can be constrained to grow only linearly.<sup>3</sup>

---

<sup>3</sup>For example, think about trying to fit a cubic equation to the reaction time ~ frequency relationship above. The cubic you would need to draw will grow as frequency<sup>3</sup> for frequencies below 0, or above 10.

A polynomial is made up of *components* for different orders: you add together multiples of 1,  $x$ ,  $x^2$ , and so on to approximate a function. Splines also approximate functions, but using a different (and more complex) set of components. There are many ways of defining splines (hence `bs`, `ns`, `rcs`, and other R functions), of which we consider only RCS.

### 9.3.2 Restricted cubic splines

Intuitively, a restricted cubic spline with  $k$  *knots* describes a curve with  $k - 2$  “bends”. Thus, the simplest possible RCS has three knots, and describes a curve with one bend (analogous to a quadratic polynomial). An RCS term for a variable `x` is be added to a regression model in R using the notation `rcs(x,k)`. For example, here we fit three models of reaction time as a function of word frequency, using progressively more complex splines ( $k = 3, 5, 7$ ):

```
mod.rcs3 <- lm(RTlexdec ~ rcs(WrittenFrequency,3), data=english)
mod.rcs5 <- lm(RTlexdec ~ rcs(WrittenFrequency,5), data=english)
mod.rcs7 <- lm(RTlexdec ~ rcs(WrittenFrequency,7), data=english)
```

The predictions of these models are:

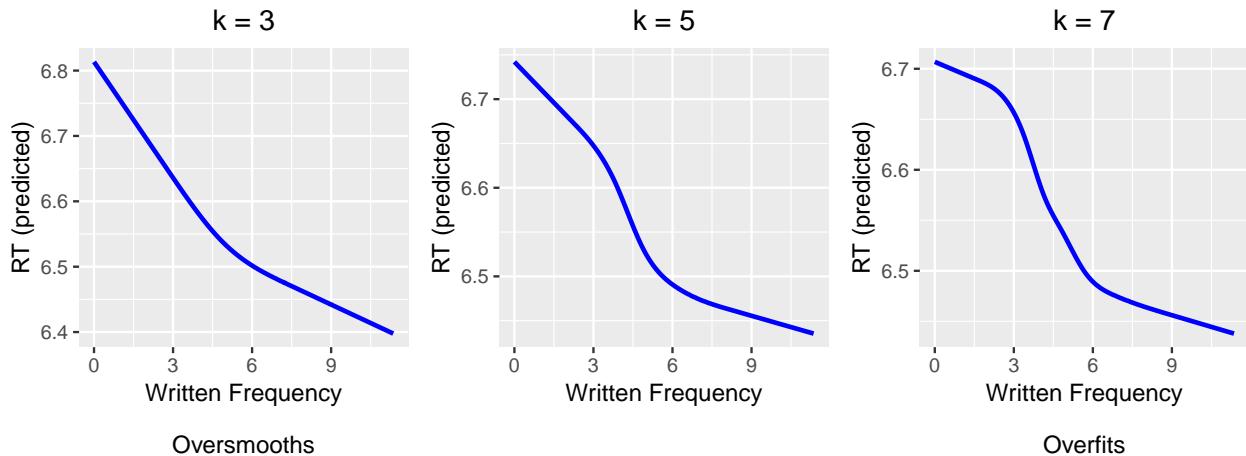
```
english$pred.rcs3 <- predict(mod.rcs3)
english$pred.rcs5 <- predict(mod.rcs5)
english$pred.rcs7 <- predict(mod.rcs7)

rcs3PredPlot <- ggplot(aes(x=WrittenFrequency, y=pred.rcs3), data=english) +
  geom_line(size=1, color='blue') +
  ylab("RT (predicted)") +
  xlab("Written Frequency\n\nOversmooths") +
  ggtitle("k = 3") +
  theme(plot.title = element_text(hjust = 0.5))

rcs5PredPlot <- ggplot(aes(x=WrittenFrequency, y=pred.rcs5), data=english) +
  geom_line(size=1, color='blue') +
  ylab("RT (predicted)") +
  xlab("Written Frequency\n\n") +
  ggtitle("k = 5") +
  theme(plot.title = element_text(hjust = 0.5))

rcs7PredPlot <- ggplot(aes(x=WrittenFrequency, y=pred.rcs7), data=english) +
  geom_line(size=1, color='blue') +
  ylab("RT (predicted)") +
  xlab("Written Frequency\n\nOverfits") +
  ggtitle("k = 7") +
  theme(plot.title = element_text(hjust = 0.5))

grid.arrange(rcs3PredPlot, rcs5PredPlot, rcs7PredPlot, ncol = 3)
```



Comparing to the plot of the empirical data above, we can see visually that  $k = 3$  is not complex enough (“underfits”) and  $k = 7$  is too complex (“overfits”).

### 9.3.3 Choosing spline complexity

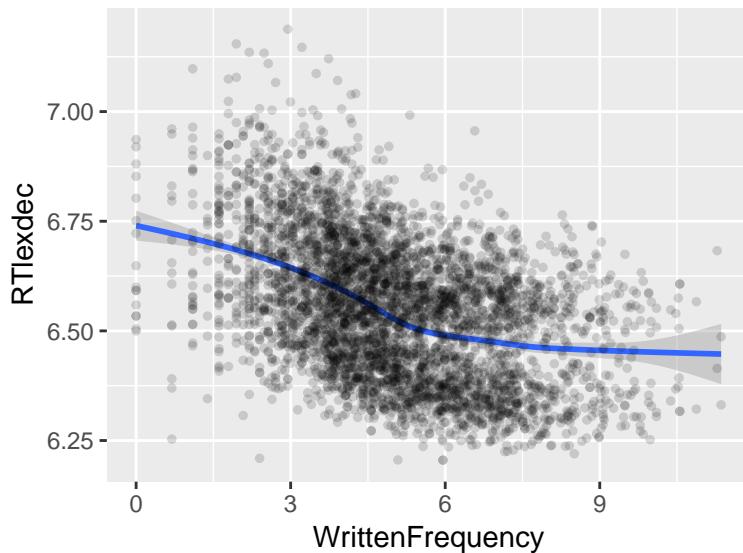
This example leads to the question: how to decide on a value of  $k$  to model a nonlinear effect, and whether a nonlinear effect is justified at all?

#### 9.3.3.1 Method 1: Visual inspection

Make a plot of the predictor versus the response, using a nonlinear smoother, and eyeball the number of bends in the relationship between the two variables.<sup>4</sup>

For example, for the RT~frequency example, it looks like there are two bends:

```
ggplot(aes(x=WrittenFrequency, y=RTlexdec), data=english) +
  geom_smooth() +
  geom_point(alpha=0.15, size=1)
```



<sup>4</sup>Ideally you should try to see the number of bends using the **empirical data**, since any nonlinear smoother you use is already making assumptions about how much to smooth the data—analogous to choosing a value of  $k$ . However, this isn’t always possible.

so we would choose  $k = 4$ .

### 9.3.3.2 Method 2: Data-driven

Fit models with different values of  $k$ , as well as a linear model:

```
mod.linear <- lm(RTlexdec ~ WrittenFrequency, data=english)
mod.rcs3 <- lm(RTlexdec ~ rcs(WrittenFrequency, 3), data=english)
mod.rcs4 <- lm(RTlexdec ~ rcs(WrittenFrequency, 4), data=english)
mod.rcs5 <- lm(RTlexdec ~ rcs(WrittenFrequency, 5), data=english)
mod.rcs6 <- lm(RTlexdec ~ rcs(WrittenFrequency, 6), data=english)
```

then use model comparison to see at what value of  $k$  adding complexity no longer gives a better fit:

```
anova(mod.linear, mod.rcs3, mod.rcs4, mod.rcs5, mod.rcs6, mod.rcs7)
```

```
## Analysis of Variance Table
##
## Model 1: RTlexdec ~ WrittenFrequency
## Model 2: RTlexdec ~ rcs(WrittenFrequency, 3)
## Model 3: RTlexdec ~ rcs(WrittenFrequency, 4)
## Model 4: RTlexdec ~ rcs(WrittenFrequency, 5)
## Model 5: RTlexdec ~ rcs(WrittenFrequency, 6)
## Model 6: RTlexdec ~ rcs(WrittenFrequency, 7)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    4566 91.194
## 2    4565 89.602  1   1.59256 81.8914 < 2.2e-16 ***
## 3    4564 89.056  1   0.54555 28.0526 1.236e-07 ***
## 4    4563 88.862  1   0.19449 10.0011  0.001575 **
## 5    4562 88.807  1   0.05496  2.8259  0.092822 .
## 6    4561 88.699  1   0.10793  5.5497  0.018526 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The significant difference between the linear model and the  $k = 3$  model means that a nonlinear effect is justified. Increasing  $k$  from 3 to 4 significantly improves the model, as does increasing from 4 to 5, but increasing from 5 to 6 doesn't ( $p = 0.09$ ). So we would choose a nonlinear relationship with  $k = 5$ . (Note that this is different from what we chose by visual inspection,  $k = 4$ , but the relationships for  $k = 4$  and  $k = 5$  turn out to be very similar.)

#### Practical note

Choosing  $k$  is a model selection problem, and like all model selection techniques the resulting model needs to be sanity-checked against the empirical data. In practice, models with very high  $k$  often are “better” than a model with lower  $k$  by model comparison via `anova()`, even when the value of  $k$  makes no sense given visual inspection. For example, a model with 9 knots significantly improves on the  $k = 5$  model:

```
mod.rcs9<- lm(RTlexdec ~ rcs(WrittenFrequency, 9), data=english)
anova(mod.rcs5, mod.rcs9)
```

```
## Analysis of Variance Table
##
## Model 1: RTlexdec ~ rcs(WrittenFrequency, 5)
## Model 2: RTlexdec ~ rcs(WrittenFrequency, 9)
##   Res.Df   RSS Df Sum of Sq      F    Pr(>F)
## 1    4563 88.862
## 2    4559 88.664  4   0.19759  2.54 0.03796 *
```

```
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

but there are obviously not 7 clear bends in the empirical relationship between `WrittenFrequency` and `RTlexdec`. In practice, you might use the lowest value of  $k$  for which model comparison shows a significant improvement (over  $k - 1$ ) and the predicted relationship is sensible.

Another option to choose a good value of  $k$  would be to use another method for model comparison, rather than the F test (what `anova` does by default for linear regressions). BIC (Sec. 3.5.2) may be a good choice, as it tends to penalize extra terms more highly than other methods we've considered (e.g. AIC, F test, LR test for logistic regressions). For the current example:

```
BIC(mod.linear, mod.rcs3, mod.rcs4, mod.rcs5, mod.rcs6, mod.rcs9)
```

```
##          df      BIC
## mod.linear 3 -4889.713
## mod.rcs3   4 -4961.764
## mod.rcs4   5 -4981.235
## mod.rcs5   6 -4982.795
## mod.rcs6   7 -4977.194
## mod.rcs9  10 -4959.256
```

choosing the model with lowest BIC would give  $k = 5$ , a sensible value.

### 9.3.4 RCS components

Examining the model with  $k = 5$ :

```
summary(mod.rcs5)
```

```
##
## Call:
## lm(formula = RTlexdec ~ rcs(WrittenFrequency, 5), data = english)
##
## Residuals:
##       Min     1Q    Median     3Q    Max 
## -0.46761 -0.11624 -0.00069  0.10278  0.53806
##
## Coefficients:
##                               Estimate Std. Error t value
## (Intercept)                 6.742126  0.016089 419.059
## rcs(WrittenFrequency, 5)WrittenFrequency -0.030765  0.005694 -5.404
## rcs(WrittenFrequency, 5)WrittenFrequency'   -0.159054  0.037966 -4.189
## rcs(WrittenFrequency, 5)WrittenFrequency''  0.761531  0.183525  4.149
## rcs(WrittenFrequency, 5)WrittenFrequency''' -0.810377  0.241481 -3.356
##                                     Pr(>|t|)    
## (Intercept)                  < 2e-16 ***
## rcs(WrittenFrequency, 5)WrittenFrequency 6.87e-08 ***
## rcs(WrittenFrequency, 5)WrittenFrequency' 2.85e-05 ***
## rcs(WrittenFrequency, 5)WrittenFrequency'' 3.39e-05 ***
## rcs(WrittenFrequency, 5)WrittenFrequency''' 0.000798 ***
##
## ---
## Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1396 on 4563 degrees of freedom
## Multiple R-squared: 0.2098, Adjusted R-squared: 0.2091
```

```
## F-statistic: 302.9 on 4 and 4563 DF, p-value: < 2.2e-16
```

there are four rows, with regression coefficients (etc.) describing the nonlinear relationship. For a linear relationship it was clear what the coefficient means: the change in  $Y$  for a unit change in  $X$ . **What do the coefficients for an RCS mean?**

### 9.3.4.1 Short answer

The short answer is: these coefficients are hard to interpret, and it's fine to mostly ignore them, increase interpreting the nonlinear effect of  $X$  as follows:

- Plotting model predictions to see the predicted effect
- To assess whether (the nonlinear effect of)  $X$  is significant, report a model comparison with a model without the nonlinear term.

For the current example: code to visualize the nonlinear relationship is above, and you could report this in a paper as:

“There was a nonlinear effect of frequency on reaction time, modeled using a restricted cubic spline with 5 knots ( $F_{4567,4} = 303$ ,  $p < 0.0001$ ), where the number of knots was chosen by picking the value which gave lowest BIC.”

If you want to also report that a *nonlinear* effect in particular was justified, you could add a sentence like

“A nonlinear relationship is clear from the empirical data (Fig. X), and significantly improves on a linear effect of frequency ( $F_{4566,3} = 40$ ,  $p < 0.0001$ ).<sup>5</sup>

The model comparisons used in these reports are:

```
mod.int <- lm(RTlexdec ~ 1, data=english)
anova(mod.linear, mod.rcs5)

## Analysis of Variance Table
##
## Model 1: RTlexdec ~ WrittenFrequency
## Model 2: RTlexdec ~ rcs(WrittenFrequency, 5)
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1    4566 91.194
## 2    4563 88.862  3    2.3326 39.926 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
anova(mod.int, mod.rcs5)

## Analysis of Variance Table
##
## Model 1: RTlexdec ~ 1
## Model 2: RTlexdec ~ rcs(WrittenFrequency, 5)
##   Res.Df   RSS Df Sum of Sq    F    Pr(>F)
## 1    4567 112.456
## 2    4563  88.862  4    23.594 302.88 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

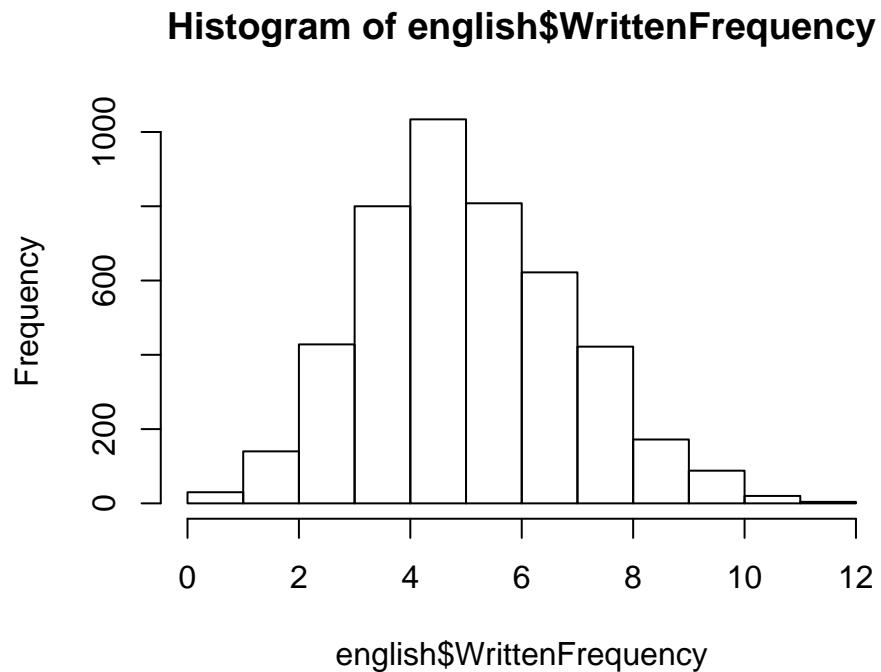
---

<sup>5</sup>This probably isn't necessary as long as there is an empirical plot where a nonlinear relationship is clear.

### 9.3.4.2 Long answer: Components of nonlinear functions

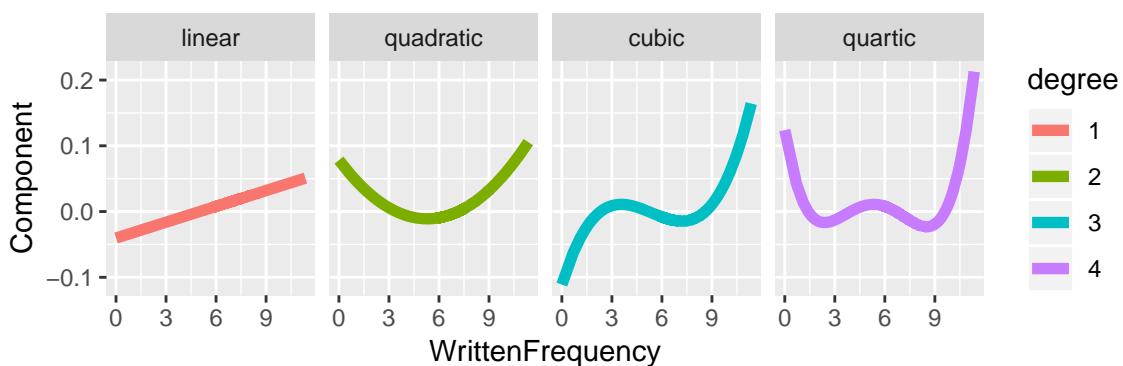
The rows of the regression table actually refer to “components” of the nonlinear function being modeled by the spline. To get a sense for what that means, let’s consider an example: the `WrittenFrequency` variable, which is roughly normally distributed with range  $\approx 0 - 12$ :

```
hist(english$WrittenFrequency)
```



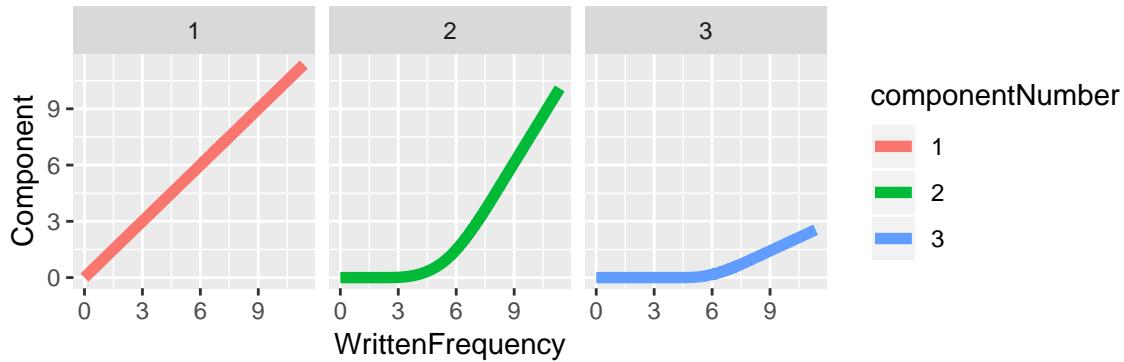
A nonlinear function of this variable could be included in a regression using a polynomial term, or an RCS term.

The first few components for a **polynomial** term would be:

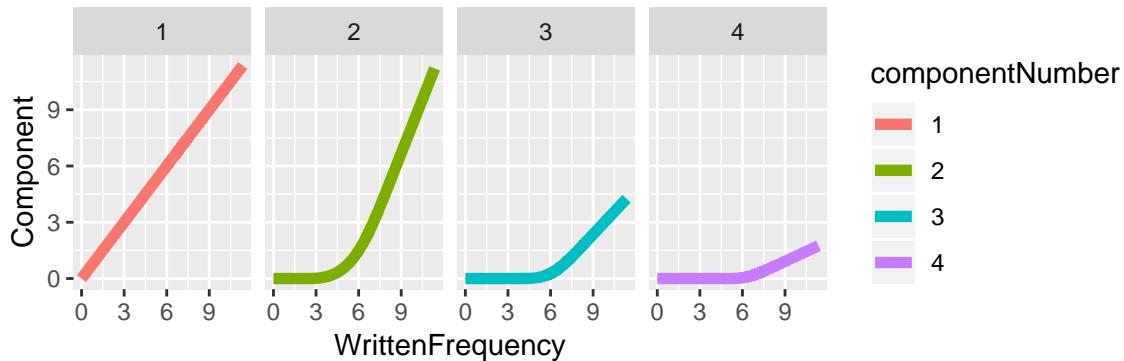


Note that for degree  $k > 2$  if `WrittenFrequency` goes below 0 or above 10, the component increases or decreases very quickly (as  $x^k$ ).

In comparison, the components for restricted cubic splines **for this data** for  $k = 4$  are:



And for  $k = 5$  the components are:



The four coefficients of the model above (with a  $k = 5$  RCS term for `WrittenFrequency`) refer to these four components. The predicted effect of `WrittenFrequency` on reaction time is:

$$-0.03 \cdot \text{component1} - 0.016 \cdot \text{component2} + 0.761 \cdot \text{component3} - 0.810 \cdot \text{component4}$$

where -0.03, etc. are the coefficient values.

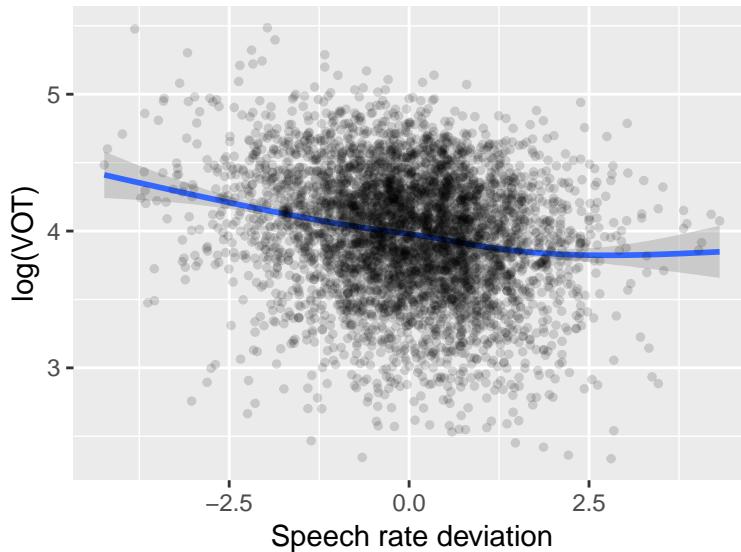
From the last two figures, we can see that each RCS component only grows **linearly** if extrapolated beyond the endpoints of the data. This is a useful property which gives better generalization on new data, when new values of the independent variable (here, `WrittenFrequency`) are observed.

### 9.3.5 Using RCS in a mixed model

As an example, we model the effect of speech rate deviation (`speakingRateDev`) on log-transformed voice onset time (`logVOT`) for the VOT dataset.

The empirical effect seems clearly non-linear:

```
ggplot(aes(x=speakingRateDev, y=logVOT), data=vot) +
  geom_smooth() +
  geom_point(alpha=0.15, size=1) +
  xlab("Speech rate deviation") +
  ylab("log(VOT)")
```



with speech rate having little effect on VOT in sufficiently fast speech.

#### Questions:

- What  $k$  would we choose by visual inspection?

#### Exercise

To choose  $k$  in a data-driven fashion, try fitting these models using just random intercepts (no slopes) of `Speaker` and `Word`, and compare them using model comparison:

- `VOT ~ speakingRateDev` (linear model)
- Similar model with RCS for  $k = 3$
- Similar model with RCS for  $k = 4$

#### 9.3.6 Random slopes for RCS terms

Fitting a “maximal” mixed-effects model with random slopes for RCS terms often results in issues that are by now familiar (Sec. 7.10):

```
votMod.corr <- lmer(logVOT ~ rcs(speakingRateDev, 3) + (1+rcs(speakingRateDev, 3) | Word) + (1+rcs(speakingRateDev, 3) | Speaker)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : unable to evaluate scaled gradient

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl =
## control$checkConv, : Model failed to converge: degenerate Hessian with 1
## negative eigenvalues

## Warning: Model failed to converge with 1 negative eigenvalue: -5.8e+03
summary(votMod.corr)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: logVOT ~ rcs(speakingRateDev, 3) + (1 + rcs(speakingRateDev,
##     3) | Word) + (1 + rcs(speakingRateDev, 3) | Speaker)
```

```

##      Data: vot
##
## REML criterion at convergence: 4595.4
##
## Scaled residuals:
##      Min     1Q Median     3Q    Max
## -4.2267 -0.6003  0.0509  0.6510  3.0740
##
## Random effects:
## Groups   Name           Variance Std.Dev. Corr
## Word     (Intercept)   0.0000000 0.00000
##          rcs(speakingRateDev, 3)speakingRateDev 0.0170030 0.13040   NaN
##          rcs(speakingRateDev, 3)speakingRateDev' 0.0235881 0.15358   NaN
## Speaker   (Intercept)   0.0253568 0.15924
##          rcs(speakingRateDev, 3)speakingRateDev 0.0022754 0.04770  0.16
##          rcs(speakingRateDev, 3)speakingRateDev' 0.0001518 0.01232  0.12
## Residual                    0.1450546 0.38086
##
## -1.00
##
## -0.96
##
## Number of obs: 4728, groups: Word, 424; Speaker, 21
##
## Fixed effects:
##                               Estimate Std. Error      df
## (Intercept)                3.99525  0.03700 21.43952
## rcs(speakingRateDev, 3)speakingRateDev -0.15343  0.01908 21.59287
## rcs(speakingRateDev, 3)speakingRateDev' 0.10948  0.01828 164.70520
## t value Pr(>|t|)
## (Intercept)                107.966 < 2e-16 ***
## rcs(speakingRateDev, 3)speakingRateDev -8.043 6.18e-08 ***
## rcs(speakingRateDev, 3)speakingRateDev'  5.989 1.29e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) rc(RD,3)RD
## rcs(RD,3)RD  0.230
## rc(RD,3)RD' -0.175 -0.837
## convergence code: 0
## unable to evaluate scaled gradient
## Model failed to converge: degenerate Hessian with 1 negative eigenvalues

```

non-convergence, and perfect correlations between random effects. There are a couple probable causes, both of which came up when we have hit these issues before:

1. The model may be overparametrized—the random-effect structure is too complex for the dataset size.
2. The predictors are not centered (see Section 9.3.4)—which results in substantial avoidable collinearity. (See the lower-right entry of **Correlation of fixed effects**: the correlation between the two RCS components is high ( $-0.83$ )).

There are two corresponding ways to deal with these issues if they come up.

### Option 1: Fit a model without random-effect correlations

To do this, we need to extract the individual components of the RCS term as numeric variables—this is analogous to extracting the contrasts as numeric variables for a factor, to use uncorrelated random effects.

This code extracts the two components for the spline with 3 knots, and adds them to the dataframe as columns:

```
vot$rateDev.comp1 <- rcspline.eval(vot$speakingRateDev, nk=3, inclx=TRUE) [,1]
vot$rateDev.comp2 <- rcspline.eval(vot$speakingRateDev, nk=3, inclx=TRUE) [,2]
```

The function `rcspline.eval` gives a matrix of components, `nk=3` selects  $k$ , `[,1]` or `[,2]` selects the matrix column number, and `inlx=TRUE` adds the linear component in addition to the nonlinear components.

We can then fit the model without random-effect correlations:

```
votMod.nocorr <- lmer(logVOT ~ rateDev.comp1 + rateDev.comp2 +
  (1+rateDev.comp1 + rateDev.comp2 || Word) +
  (1+rateDev.comp1 + rateDev.comp2 || Speaker),
  data=vot)
```

The model now converges. (This may not be kosher, though—there are conceptual issues with using uncorrelated random effects for non-centered variables (Barr et al., 2013).

The model is:

```
summary(votMod.nocorr)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
## lmerModLmerTest]
## Formula: logVOT ~ rateDev.comp1 + rateDev.comp2 + (1 + rateDev.comp1 +
##   rateDev.comp2 || Word) + (1 + rateDev.comp1 + rateDev.comp2 || 
##   Speaker)
## Data: vot
##
## REML criterion at convergence: 4475.1
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.3005 -0.5865  0.0448  0.6312  3.3378
##
## Random effects:
##   Groups      Name        Variance Std.Dev.
##   Word        (Intercept) 2.914e-02 0.170708
##   Word.1      rateDev.comp1 2.327e-04 0.015255
##   Word.2      rateDev.comp2 9.418e-06 0.003069
##   Speaker     (Intercept) 2.540e-02 0.159365
##   Speaker.1   rateDev.comp1 1.358e-03 0.036856
##   Speaker.2   rateDev.comp2 0.000e+00 0.000000
##   Residual                1.394e-01 0.373399
## Number of obs: 4728, groups: Word, 424; Speaker, 21
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)
## (Intercept)  4.07384   0.03913 29.99294 104.113 < 2e-16 ***
##
```

```

## rateDev.comp1 -0.10347    0.01391 43.27518  -7.436 2.91e-09 ***
## rateDev.comp2  0.03810    0.01237 32.65991   3.080  0.00418 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##           (Intr) rtDv.1
## rateDv.cmp1  0.176
## rateDv.cmp2 -0.259 -0.646

```

Note the significant `rateDev.comp1` and `rateDev.comp2` terms, which we can think of as the “linear” and “nonlinear” terms.

#### Questions:

- What does the fact that both are significant tell us?
- How do people differ in the speech rate effect? (Examine the random slope terms.)

#### Detour: Visualizing nonlinear effects

It would be nice to visualize the predicted nonlinear effect from a model, but this turns out to be slightly tricky and is not shown here.

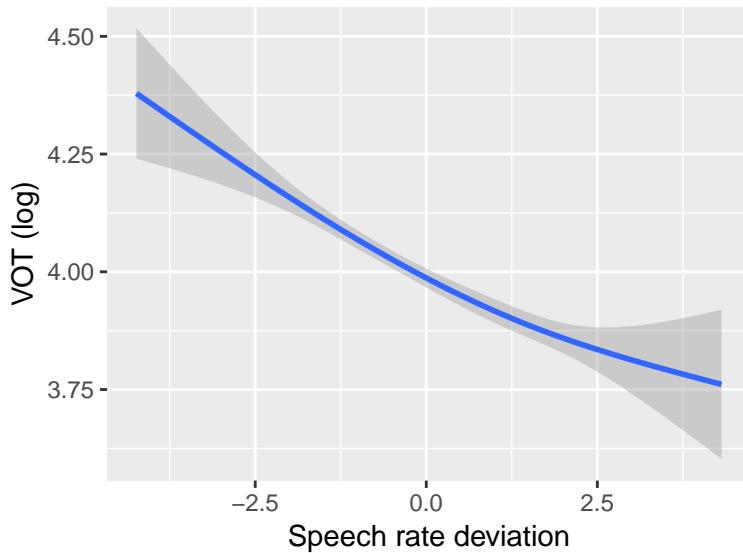
An easier approximation, which is usually a good idea anyway (as a sanity check that the empirical data shows the qualitative effect predicted by the model), is to plot a smooth over the empirical data using a spline with the same number of knots as the RCS term in the model.

For example, to visualize the effect of an RCS term with 3 knots for a `VOT ~ speakingRateDev` model:

```

ggplot(aes(x=speakingRateDev, y=logVOT), data=vot) +
  geom_smooth(method='lm', formula=y~splines::ns(x,3)) +
  xlab("Speech rate deviation") +
  ylab("VOT (log)")

```



#### Option 2: Use centered predictors

Note that RCS components are not centered, by default—as can be seen in the plots of these components for  $k = 4$  and  $k = 5$  in Sec. 9.3.4.2, where each component is positive for all values of `WrittenFrequency`.

Less problematically, they are not orthogonal—for example, components 1 and 2 are clearly correlated (for  $k = 4$  and  $k = 5$ ), as component 2 is a line for `WrittenFrequency > 6`.

We can address both issues by using the *principal components* of the RCS components: a linear transformation which makes them centered and orthogonal. You can extract the principal components in R using the `pc=TRUE` flag:

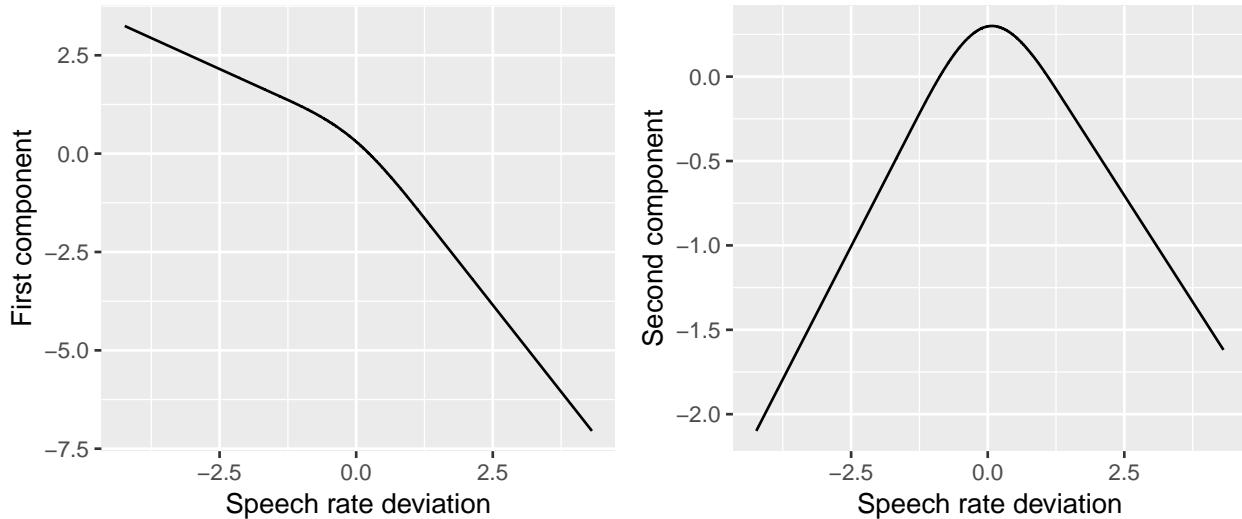
```
vot$rateDev.pcomp1 <- rcspline.eval(vot$speakingRateDev, nk=3, inclx=TRUE, pc=TRUE) [,1]
vot$rateDev.pcomp2 <- rcspline.eval(vot$speakingRateDev, nk=3, inclx=TRUE, pc=TRUE) [,2]
```

These are still restricted cubic spline components (they look like glued-together polynomials, etc.), but they look different from before. For example, the first and second components for  $k = 3$  for `speakingRateDev` for the `vot` data are:

```
rcsPlot1 <- ggplot(aes(x=speakingRateDev, y=rateDev.pcomp1), data=vot) +
  geom_line() +
  xlab("Speech rate deviation") +
  ylab("First component")

rcsPlot2 <- ggplot(aes(x=speakingRateDev, y=rateDev.pcomp2), data=vot) +
  geom_line() +
  xlab("Speech rate deviation") +
  ylab("Second component")

grid.arrange(rcsPlot1, rcsPlot2, ncol = 2)
```



(Compare to the plots for  $k = 4$  and  $k = 5$  above.) For our purposes, the most important difference is that the PCs are centered.

A model using the PC versions of the RCS components, with full maximal random-effect structure (correlations between random effects included), would be:

```
votMod.corr.2 <- lmer(logVOT ~ rateDev.pcomp1 + rateDev.pcomp2 +
  (1+rateDev.pcomp1 + rateDev.pcomp2|Word) +
  (1+rateDev.pcomp1 + rateDev.pcomp2|Speaker),
  data=vot)

summary(votMod.corr.2)

## Linear mixed model fit by REML. t-tests use Satterthwaite's method [
```

```

## lmerModLmerTest]
## Formula: logVOT ~ rateDev.pcomp1 + rateDev.pcomp2 + (1 + rateDev.pcomp1 +
##           rateDev.pcomp2 | Word) + (1 + rateDev.pcomp1 + rateDev.pcomp2 | 
##           Speaker)
## Data: vot
##
## REML criterion at convergence: 4467.3
##
## Scaled residuals:
##    Min     1Q Median     3Q    Max
## -4.2831 -0.5951  0.0451  0.6310  3.3031
##
## Random effects:
##   Groups   Name        Variance Std.Dev. Corr
##   Word     (Intercept) 0.0277041 0.16645
##          rateDev.pcomp1 0.0001375 0.01172  0.84
##          rateDev.pcomp2 0.0011219 0.03349 -0.87 -0.46
##   Speaker  (Intercept) 0.0253885 0.15934
##          rateDev.pcomp1 0.0007723 0.02779 -0.24
##          rateDev.pcomp2 0.0029008 0.05386  0.02 -0.98
##   Residual            0.1394170 0.37339
## Number of obs: 4728, groups: Word, 424; Speaker, 21
##
## Fixed effects:
##             Estimate Std. Error      df t value Pr(>|t|)    
## (Intercept) 4.103607  0.037700 25.961234 108.848 < 2e-16 ***
## rateDev.pcomp1 0.057963  0.008032 13.000512  7.217 6.78e-06 ***
## rateDev.pcomp2 -0.113209  0.021463 13.901202 -5.275 0.00012 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) rtDv.1
## ratDv.pcmp1 -0.145
## ratDv.pcmp2 -0.011 -0.427

```

Crucially, this model makes the same predictions as `votMod.corr`—the only difference is in how the nonlinear function of `speechRateDev` is coded (analogously to using different contrast coding schemes for a factor).<sup>6</sup> However, the new model converged without issue, while `votMod.corr` did not—likely because we are now representing the `speechRateDev` effect using centered and orthogonal variables.

Note that there is less collinearity in `votMod.corr.2` than in `votMod.corr`—the correlations in `Correlation of Fixed Effects` are closer to zero.

#### Questions:

- Why is this?

### 9.3.7 Nonlinear effects: Summary

To summarize:

- Nonlinear effects can be used to capture non-linear relationships between the response and (a) predictor(s). Such relationships are common, so **using nonlinear effects is often appropriate**.

---

<sup>6</sup>At least, we think this is the case.

- Modeling nonlinear relationships **somewhat** is more important than the actual method used!
- In terms of a method: we recommend using splines to code nonlinear effects. (Here we have demonstrated restricted cubic splines, but other flavors are OK.)
  - Pro: Splines are well-behaved and should lead to better generalization to new data.
  - Con: Spline components are tricky to interpret, and (in R) sometimes tricky to work with.
- Using polynomials to model nonlinear effects is OK, but dispreferred.
  - Pro: Polynomial terms are easy to interpret
  - Con: Polynomial functions are not well-behaved (interpolation issues, etc.) and can make bad predictions on new data.
  - Regardless, polynomials are commonly used, e.g. in Growth Curve Analysis (which is basically mixed-effects models with polynomials used to model a nonlinear relationship).

## 9.4 Predictions from mixed models

To visualize the predictions of mixed models, it is useful to make:

- *partial effect plots*: These show the model's prediction as one predictor is changed, holding others constant.
- predictions for different levels of grouping factors (e.g. different participants): in the overall value (“intercept”), partial effect of a given predictor (“slope”), or something more complex.

Decent software (in 2018) exists for making such predictions from mixed models, such as:

- The `effects` package (see vignette)
- The `sjPlot` package
- `plotLME.R.fnc()` in `languageR`

Nonetheless, it is also useful to know how to make such predictions yourself, as pre-existing packages make various assumptions, and don't always give exactly what you want.

### 9.4.1 Making Model Predictions

Visualizing predictions from a model in R always involves three steps:

1. Make a new data frame of values you want to predict at
2. Get predictions of model for these values
3. Make plots of the predictions

#### Example

Let's use a basic model of the `acoustics` correlate of stress shifting, for the `givenness` data:<sup>7</sup>

---

<sup>7</sup>We use uncorrelated by-item random effects in this model because the correlations do not significantly improve model likelihood (by an LR test), the by-item random effects have smaller magnitudes, and the model with correlations does not converge.

```
mod1 <- lmer(acoustics ~ conditionLabel*npType + voice +
              (1 + clabel.williams*npType.pron || item) +
              (1 + clabel.williams*npType.pron + voice.passive| participant),
              data=givenness)
```

### Predictions only (no confidence intervals)

**Step 1:** Make dataframe of new values we want to predict the response for:

The `expand.grid()` function can be used to make all possible combinations of multiple predictors:

```
newdata <- with(givenness,
                 expand.grid(conditionLabel=unique(conditionLabel),
                             npType=unique(npType),
                             voice=unique(voice)))

newdata
```

```
##   conditionLabel  npType   voice
## 1      Williams    full  passive
## 2     Contrast    full  passive
## 3      Williams  pronoun  passive
## 4     Contrast  pronoun  passive
## 5      Williams    full   active
## 6     Contrast    full   active
## 7      Williams  pronoun   active
## 8     Contrast  pronoun   active
```

**Step 2:** Make model predictions

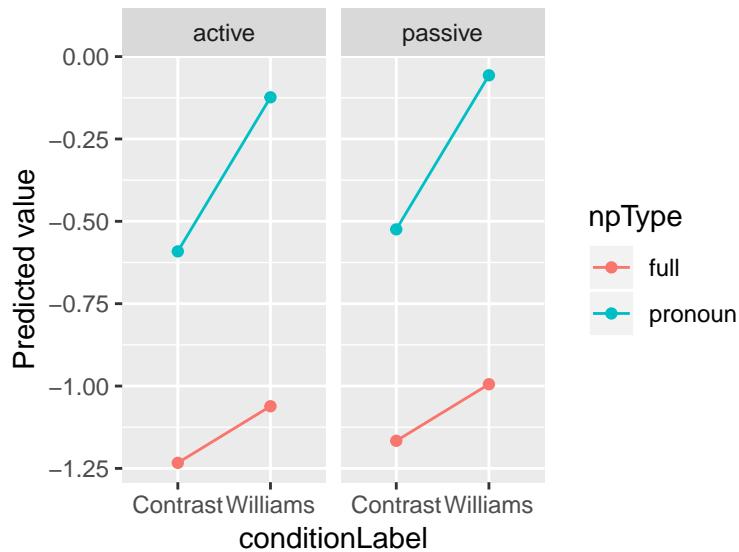
If you want predictions based on **fixed effects only** (no random effects), use the `predict()` function with the flag `re.form=NA`:

```
newdata$prediction <- predict(mod1, newdata=newdata, re.form=NA)
```

**Step 3:** Visualize

The model's predictions for `acoustics` as a function of the three predictors (condition label, NP type, voice) are:

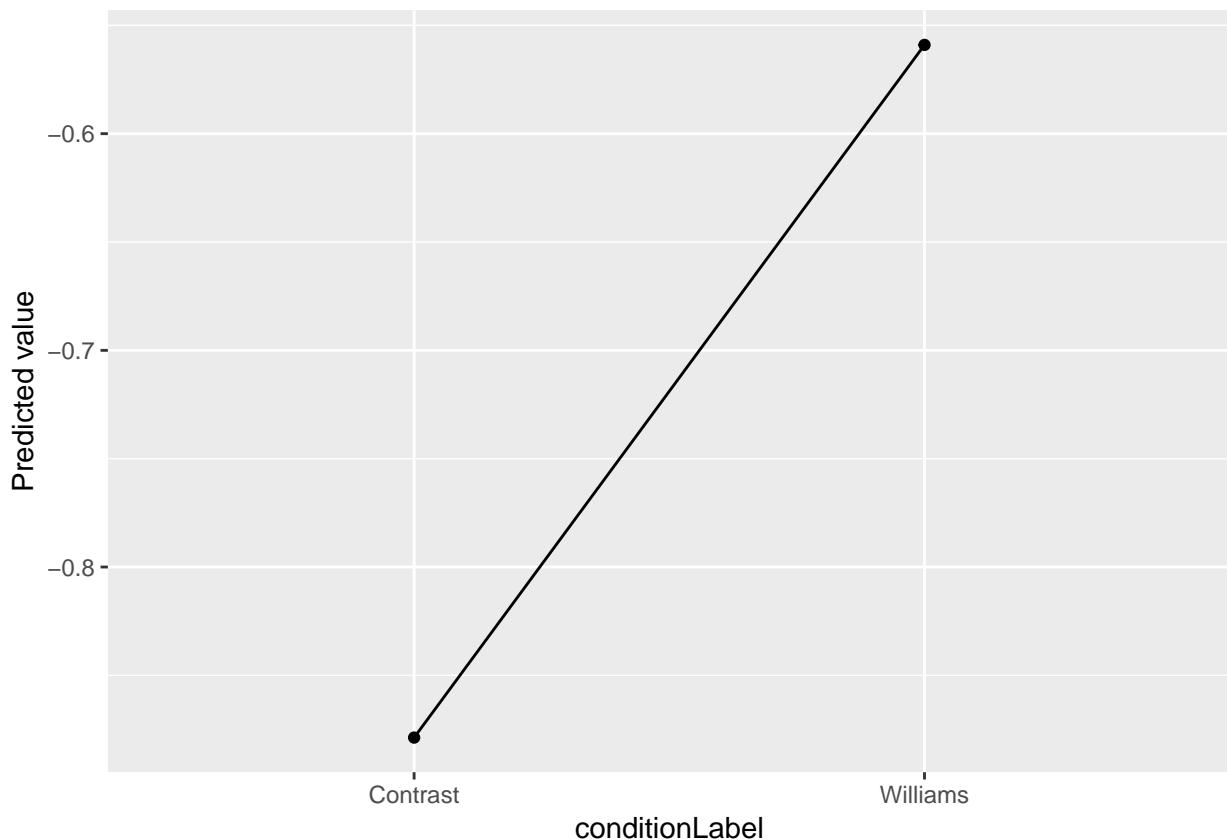
```
ggplot(aes(x=conditionLabel, y=prediction), data=newdata) +
  geom_point(aes(color=npType)) +
  geom_line(aes(color=npType, group=npType)) +
  facet_wrap(~voice) +
  ylab("Predicted value")
```



These predictions are for an average participant, and “average item”: meaning, the average over `voice=active` and `voice=passive` items.

To make a “partial effect plot” of just the effect of `conditionLabel`, say, we can simply average over the other variables:

```
newdata %>% group_by(conditionLabel) %>% summarise(pred=mean(prediction), dummy=1) %>% ggplot(aes(x=cond  
geom_line(aes(group=dummy)) +  
ylab("Predicted value"))
```



### Predictions with confidence intervals

Generally we want to show model predictions with a measure of uncertainty, such as confidence intervals (CIs)

Obtaining CIs turns out to be less straightforward than obtaining just model predictions, because there is more than one way to define “uncertainty”. For example: do we want to take into account uncertainty from just (uncertainty about the) fixed effects? What about random effects? And how? (You can google “confidence intervals prediction lme4” to get a sense of the issue.)

You can get confidence intervals relatively simply using just the uncertainty in the fixed effects, as illustrated in Ben Bolker’s FAQ. Applied to our data:

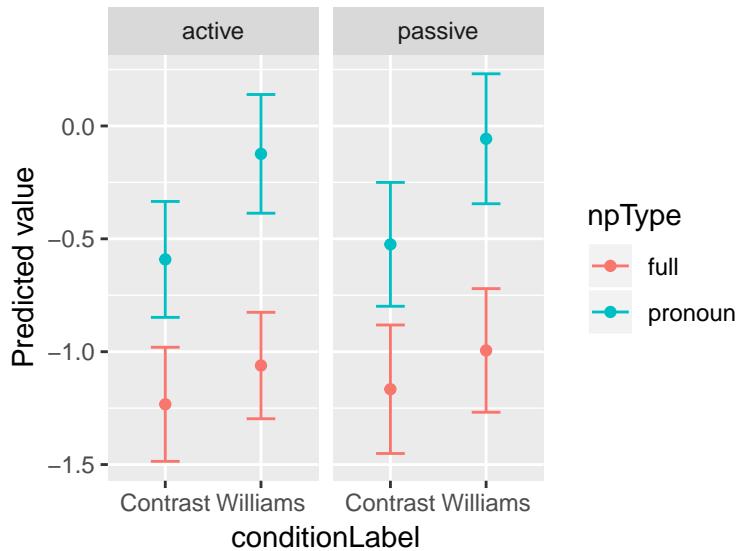
```
## "design matrix" for fitted model
mm <- model.matrix(terms(mod1), data.frame(acoustics=1, newdata))

## SE of prediction, just from uncertainty in fixed effects and residual variance:
newdata$SE <- sqrt(diag(mm %*% tcrossprod(vcov(mod1), mm)))
```

Note that the computed standard error (SE) does not take into account variation among participants, etc. It can be thought of as “the error for an average subject and item”.

Multiplying these standard errors by 1.96 can be used to construct 95% confidence intervals (again, calculated from the fixed-effect errors only), which can be visualized:

```
ggplot(aes(x=conditionLabel, y=prediction, ymin=prediction-1.96*SE, ymax=prediction+1.96*SE), data=newdata)
  geom_point(aes(color=npType)) +
  geom_errorbar(aes(color=npType), width=0.3) +
  facet_wrap(~voice) +
  ylab("Predicted value")
```



### Extracting predicted random effects

Extracting partial effects, and “overall” values (intercepts) for each participant (etc.) is also of interest. Some examples are given in previous chapters, and further examples are shown in an Appendix (Sec. 9.6).

### 9.4.2 Simulation-based predictions

The examples seen so far calculate model predictions and confidence intervals using the terms of the fitted models—fixed-effect coefficient values and SEs, and so on. This requires writing some code (and some knowledge), but the predictions/CIs are computed quickly.

The more general solution, which requires little code or knowledge—but can be very slow—is computing model predictions and confidence intervals using simulation. Parametric or semi-parametric bootstrapping, as implemented in the `bootMer` package or the `simulate` function (from the `arm` package), can be used to estimate the probability distribution over values of `any` quantity predicted by the model—including a particular combination of predictor values, as in a partial-effect plot. This distribution can be used to obtain a “prediction” and “95% confidence interval” by computing e.g. the median and the 2.5%/97.5% quantiles. The downside is that simulation can be very computationally intensive—you essentially have to re-fit the model many (1000+) times to get reasonable estimates.

Useful examples are given in:

- A vignette by J. Knowles and C. Frederick
- An R-bloggers page by grumble10.

An example of a simple simulation-based method is shown for LMMs in Section 7.12.1.

## 9.5 Post-hoc tests and multiple comparisons

The *multiple comparisons* problem refers to the issue that the more hypotheses you test, the less meaningful the  $p$ -values are. (If we test 20 hypotheses with  $\alpha$  level 0.05, we will find 1 significant effect by chance, on average.)

A solution commonly used is to correct the  $p$ -values for “multiple comparisons” before comparing to  $\alpha$  (e.g. 0.05). Many such correction methods exist, including:

- The *Bonferroni* method: multiply  $p$  by the number of tests conducted
  - This method has low power, and should never be used—see `?p.adjust.methods` in R.
- The *Holm* method
  - Less conservative than Bonferroni
- *Tukey HSD* (“Honestly Significant Difference”)
- *FDR* (“False Discovery Rate”), a.k.a. the Benjamini & Hochberg (BH) method

Not correcting for multiple comparisons when doing many statistical tests can easily lead to spurious results—this is often called “data dredging” or “p-hacking”.

Which multiple comparison method should be used? There is a trade-off between methods with higher power and lower Type I error. Reasonable and widely-used methods are Holm and Tukey HSD.

One place where the multiple comparisons problem comes up is for factors with multiple ( $>2$ ) levels. When a model includes a categorical variable  $X$  with  $k$  levels, we know how to:

1. Ask  $k - 1$  questions about how it affects the response: **contrast coding** (discussed in Sec. 6.2)
2. Ask “does  $X$  affect the response?”: **likelihood ratio test** (discussed in Sec. 6.3)

It is customary to not correct for multiple comparisons for (1)—and indeed, for any set of predictors in multiple regression models in general.<sup>8</sup>

---

<sup>8</sup>It is beyond our knowledge whether there is actually a good reason for this, but googling (e.g. here, here) suggests that “why don’t we correct for multiple comparisons in multiple regression models?” is a common question.

But often, we would like to ask more than  $k - 1$  questions, such as:

- “is level 1 diff from level 2?” “Level 2 from level 3?” “Level 1 from level 3?”
- “is there any Williams effect when `npType = pronoun?`” (for the `givenness` data)

Such questions can be answered via hypothesis tests using *post-hoc tests*.<sup>9</sup> These are also called “difference in means tests”, or other terms. It is common in ANOVA analyses to use “post-hoc tests” to refer to “testing which levels of a factor  $X$  are actually different from each other, after an ANOVA showed that  $X$  has a significant effect.”

When we ask more than  $k - 1$  questions, the questions are not independent, so we **need** to correct for multiple comparisons.

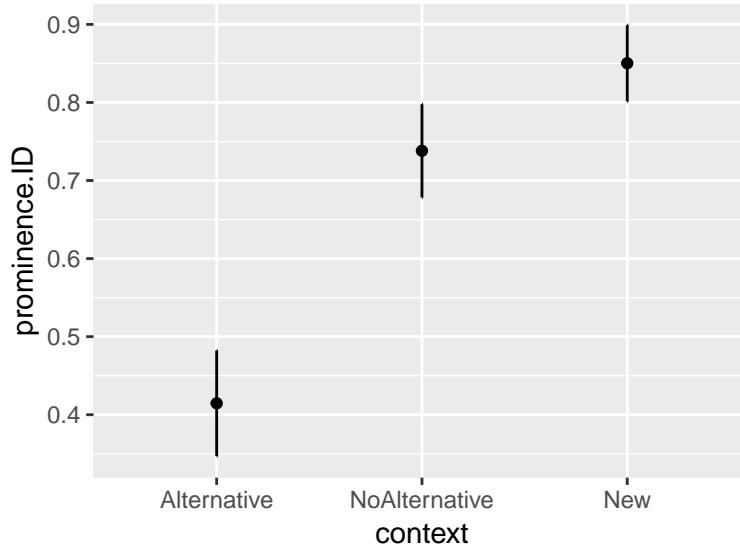
### Example

As an example, consider the `alternatives` dataset, where we will fit a mixed-effects logistic regression, with:

- Response: `prominence`
- Predictor: `context` (`NoAlternative`, `Alternative`, `None`)
- Random effects: maximal, without random-effect correlations

The empirical pattern is:

```
##           context prominence.ID          se
## 1   Alternative     0.4146341 0.03449301
## 2 NoAlternative     0.7380952 0.03041268
## 3        New       0.8502415 0.02486184
```



First, we make `context1`, `context2` numerical predictors, corresponding to the two Helmert contrasts:

```
contrasts(alternatives$context) <- contr.helmert(3)
alternatives <- mutate(alternatives,
                       context1 = model.matrix(~context, alternatives)[,2],
                       context2 = model.matrix(~context, alternatives)[,3])
```

The model is:

---

<sup>9</sup>Why “post-hoc”? The terminology comes from ANOVAs applied in a traditional experimental setup, where “post-hoc” tests are different from “planned comparisons”.

```

mod.nocorr <- glmer(prominence ~ context +
                      (1+context1 + context2||participant) +
                      (1+context1 + context2||item),
                      data=alternatives,
                      family="binomial",
                      control=glmerControl(optimizer = "bobyqa"))
summary(mod.nocorr)

## Generalized linear mixed model fit by maximum likelihood (Laplace
##   Approximation) [glmerMod]
##   Family: binomial  ( logit )
## Formula:
##   prominence ~ context + (1 + context1 + context2 || participant) +
##       (1 + context1 + context2 || item)
##   Data: alternatives
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC  logLik deviance df.resid
##      646.8    686.7   -314.4     628.8     613
##
## Scaled residuals:
##      Min      1Q  Median      3Q      Max
## -3.5763 -0.5676  0.2484  0.5564  2.9286
##
## Random effects:
##   Groups      Name      Variance Std.Dev.
##   participant (Intercept) 0.67379  0.8208
##   participant.context1 0.06545  0.2558
##   participant.context2 0.00000  0.0000
##   item         (Intercept) 0.56035  0.7486
##   item.context1 0.00000  0.0000
##   item.context2 0.29399  0.5422
##   Number of obs: 622, groups: participant, 18; item, 12
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.1533    0.3199   3.605 0.000312 ***
## context1     0.8716    0.1375   6.338 2.32e-10 ***
## context2     0.7014    0.1909   3.673 0.000240 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cntxt1
## context1  0.054
## context2  0.129  0.018

```

The two significant fixed-effect coefficients can be interpreted as:

- *NoAlternative > Alternative* (in terms of probability of prominence shift)
- *New > NoAlternative/ Alternative*

But what about *NoAlternative* versus *New*, or *Alternative* versus *New*?

The `lsmeans` package (now superceded by the `emmeans` package, with expanded functionality) is invaluable

for carrying out post-hoc tests (Lenth, 2016, 2018). In this case, we examine every difference between two levels:

```
mod.lsmeans <- lsmeans::lsmeans(mod.nocorr, ~context)
pairs(mod.lsmeans)

## contrast estimate SE df z.ratio p.value
## Alternative - NoAlternative -1.743220 0.2750211 NA -6.338 <.0001
## Alternative - New -2.975683 0.5915409 NA -5.030 <.0001
## NoAlternative - New -1.232463 0.5866488 NA -2.101 0.0896
##
## Results are given on the log odds ratio (not the response) scale.
## P value adjustment: tukey method for comparing a family of 3 estimates
```

This reports an estimated difference between each pair of levels (here, in log-odds), with an associated  $p$ -value, corrected for multiple comparisons using the Tukey HSD method.

We can conclude that:  $\text{Alternative} < \text{NoAlternative} \leq \text{New}$  (at  $\alpha = 0.05$ ).

Post-hoc tests can also be used for:

- Checking whether particular subsets of levels differ from others
- Checking whether  $X$  has an effect, for each level of  $Y$  (e.g., “is the Williams effect significant for each level of npType?”, for the `givenness` data)

For example:

```
temp <- lsmeans::lsmeans(mod1, ~conditionLabel | npType)
pairs(temp)

## npType = full:
## contrast estimate SE df t.ratio p.value
## Contrast - Williams -0.1717844 0.08115976 65.95 -2.117 0.0381
##
## npType = pronoun:
## contrast estimate SE df t.ratio p.value
## Contrast - Williams -0.4676114 0.09090331 23.77 -5.144 <.0001
##
## Results are averaged over the levels of: voice
```

You can read more in the very useful vignette for `lsmeans` (or for `emmeans`).

### Exercise

- For the `regularity` data, fit a logistic regression of `regularity` as a function of `Auxiliary`
- Use `lsmeans` to test: is there a difference between
  - *zijn/hebben*
  - *zijnheb/hebben*
  - *zijn/zijnheb*

?

## 9.6 Appendix: Model predictions for individual participants

A couple additional examples of predictions from `mod1`:

### 9.6.1 Predictions incorporating offsets for individual speakers

Get predictions for every speaker, for every combination of fixed-effect values:

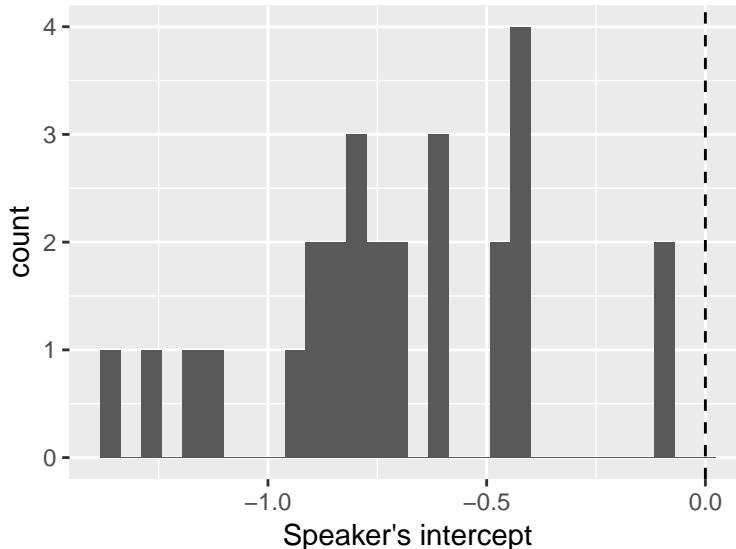
```
newdata <- with(givenness,
                 expand.grid(participant=unique(participant),
                             item=unique(item),
                             conditionLabel=unique(conditionLabel),
                             npType=unique(npType),
                             voice=unique(voice),
                             voice.passive=unique(voice.passive),
                             clabel.williams=unique(clabel.williams),
                             npType.pron=unique(npType.pron)))

newdata$prediction <- predict(mod1, newdata=newdata, re.form=~(1|participant))
```

These predictions can be used to obtain each speaker's intercept (fixed effect + by-speaker intercept), and visualize their distribution across speakers:

```
byParticInt <- newdata %>% group_by(participant) %>% summarise(intercept = mean(prediction))

ggplot(byParticInt, aes(intercept)) +
  geom_histogram(bins = 30) +
  geom_vline(xintercept=0, lty=2) +
  xlab("Speaker's intercept")
```



We can conclude that speakers differ substantially in their baseline rate of stress shifting (`acoustics`) value, but they are always more likely to *not* shift (negative value).

### 9.6.2 Predicted Williams effect for each speaker

One way to do this:

- Get overall Williams effect (across participant and items)

```
newdata2 <- with(givenness,
                  expand.grid(conditionLabel=unique(conditionLabel),
```

```

npType=unique(npType),
voice='active'))
newdata2$prediction <- predict(mod1, newdata2, re.form=NA)

overall <- mean((newdata2[1,'prediction']-newdata2[2,'prediction']),
(newdata2[3,'prediction']-newdata2[4,'prediction']))

```

- Then, add each participant's offset (random slope):

```

slopes <- ranef(mod1)$participant[['clabel.williams']]
byParticSlope <- data.frame(participant=rownames(ranef(mod1)$participant),
williamsEffect=overall + slopes)

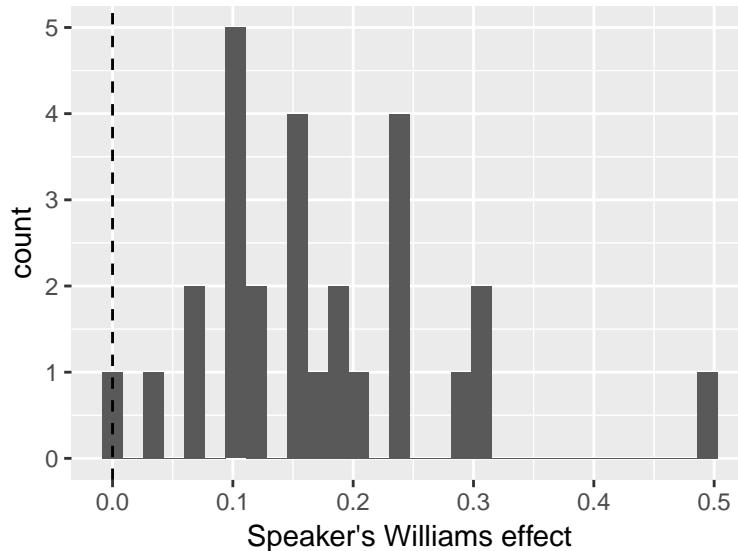
```

Visualize distribution of participant effects:

```

ggplot(byParticSlope, aes(williamsEffect)) +
  geom_histogram(bins = 30) +
  geom_vline(xintercept=0, lty=2) +
  xlab("Speaker's Williams effect")

```



## 9.7 Appendix: Random slopes for factors

Ordered factors have the same issues when fitting random slopes with uncorrelated random-effect structures that we have seen for factors with multiple levels.

As an example, using the `alternatives` data, we fit a model with a fixed effect of `context`, coded as an ordered factor (*Alternative* < *NoAlternative* < *New*)

```

## relevel context to be in the intuitively plausible order
alternatives <- mutate(alternatives, context=as.ordered(factor(context, levels=c("Alternative", "NoAlternative", "New"))

```

and with by-participant random effects only.

**Exercise:** fit this model

Solution:

```

summary(alternativesMod1)

## Generalized linear mixed model fit by maximum likelihood (Laplace
## Approximation) [glmerMod]
## Family: binomial ( logit )
## Formula:
## shifted ~ context + (1 + context | item) + (1 + context | participant)
## Data: alternatives
## Control: glmerControl(optimizer = "bobyqa")
##
##      AIC      BIC  logLik deviance df.resid
## 654.6    721.1   -312.3    624.6     607
##
## Scaled residuals:
##      Min      1Q  Median      3Q     Max
## -2.5891 -0.5648 -0.2148  0.5550  3.5691
##
## Random effects:
## Groups      Name        Variance Std.Dev. Corr
## participant (Intercept) 0.65857  0.8115
##           context.L  0.23473  0.4845   -0.11
##           context.Q  0.05022  0.2241   -0.27 -0.93
## item       (Intercept) 0.67813  0.8235
##           context.L  1.98312  1.4082   0.62
##           context.Q  0.58874  0.7673   0.61  1.00
## Number of obs: 622, groups: participant, 18; item, 12
##
## Fixed effects:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.25159   0.36521 -3.427  0.00061 ***
## context.L   -2.33728   0.58091 -4.024 5.73e-05 ***
## context.Q    0.06461   0.36381   0.178  0.85905
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##          (Intr) cntx.L
## context.L  0.567
## context.Q  0.461  0.760

```

The high random-effect correlations here suggest we should try a model with uncorrelated random effects.

As we've seen before, fitting a model with standard notation for uncorrelated random effects (||) doesn't work:

```

lmer(shifted ~ context +
  (1+context||participant) +
  (1+context||participant),
  data=alternatives)

## Warning in checkConv(attr(opt, "derivs"), opt$par, ctrl = control$checkConv, : Model is nearly uniden
## - Rescale variables?

## Warning: Model failed to converge with 2 negative eigenvalues: -9.3e-06
## -2.7e-05

```

```

## Linear mixed model fit by REML ['lmerModLmerTest']
## Formula:
## shifted ~ context + (1 + context || participant) + (1 + context || 
##   participant)
## Data: alternatives
## REML criterion at convergence: 704.8152
## Random effects:
## Groups           Name        Std.Dev. Corr
## participant    (Intercept) 0.00000
## participant.1  contextAlternative 0.17163
##                   contextNoAlternative 0.10918  0.83
##                   contextNew       0.04853  0.83  1.00
## participant.2  (Intercept) 0.00000
## participant.3  contextAlternative 0.10457
##                   contextNoAlternative 0.08913  0.67
##                   contextNew       0.03927  0.68  1.00
## Residual          0.41044
## Number of obs: 622, groups: participant, 18
## Fixed Effects:
## (Intercept)  context.L  context.Q
## 0.33134     -0.30660      0.08701
## convergence code 0; 1 optimizer warnings; 0 lme4 warnings

```

**Exercise:** carry out the same procedure to implement uncorrelated random effects as we saw for multi-level factors in Section 8.10.2

- Manually make numeric variables for each contrast, called `context1`, `context2`
- Fit a model with uncorrelated random effects using these numeric variables
- Check using a model comparison whether the models with and without correlations significantly differ

## 9.8 Solutions

**Q:** What does the model predict about:

- The overall effect of `context`? (What kind of relationship with log-odds of stress shift?)
- By-participant and by-item variability in the `context` effect?

**A:** There is a significant linear effect of the ordered predictor linear, but the quadratic component does not reach significance. There is substantial by-item and by-participant variability with respect to both linear and quadratic effect.

**Q:** What  $k$  would we choose by visual inspection?

**A:**  $k=3$  (we choose  $k$  so there are  $k - 2$  ‘bends’, and it looks like there is one bend in the curve)

**Exercise:** To choose  $k$  in a data-driven fashion, try fitting these models using just random intercepts (no slopes) of `Speaker` and `Word`, and compare them using model comparison:

- `VOT ~ speakingRateDev` (linear model)
- Similar model with RCS for  $k = 3$
- Similar model with RCS for  $k = 4$

**A:**

```

modelVOT=lmer(logVOT ~ speakingRateDev +
  (1|Speaker) + (1|Word),
  data=vot)
modelVOT.rcs3 <- lmer(logVOT ~ rcs(speakingRateDev,3) + (1|Speaker) + (1|Word), data=vot)
modelVOT.rcs4 <- lmer(logVOT ~ rcs(speakingRateDev,4) + (1|Speaker) + (1|Word), data=vot)

anova(modelVOT,modelVOT.rcs3, modelVOT.rcs4)

## refitting model(s) with ML (instead of REML)

## Data: vot
## Models:
## modelVOT: logVOT ~ speakingRateDev + (1 | Speaker) + (1 | Word)
## modelVOT.rcs3: logVOT ~ rcs(speakingRateDev, 3) + (1 | Speaker) + (1 | Word)
## modelVOT.rcs4: logVOT ~ rcs(speakingRateDev, 4) + (1 | Speaker) + (1 | Word)
##           Df     AIC     BIC logLik deviance Chisq Chi Df Pr(>Chisq)
## modelVOT      5 4493.5 4525.8 -2241.8    4483.5
## modelVOT.rcs3 6 4482.6 4521.4 -2235.3    4470.6 12.9231      1  0.0003246
## modelVOT.rcs4 7 4483.8 4529.0 -2234.9    4469.8  0.8121      1  0.3675095
##
## modelVOT
## modelVOT.rcs3 ***
## modelVOT.rcs4
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

The results are line with the visual inspection: Having splines with  $k = 4$  does not improve the model over just having splines with  $k = 3$ .

---

**Q:**

- What does the fact that both are significant tell us?
- How do people differ in the speech rate effect? (Examine the random slope terms.)

**A:** The fact that both are significant tells us that just fitting a linear predictor would not be justified—the non-linear component contributes significantly to the model. Participants show variability in the linear effect.

# Chapter 10

## Appendix: Datasets and packages

All datasets used in this book are publicly available, either as:

- Datasets in the `languageR` R package (Baayen, 2013) associated with R.H. Baayen's book *Analyzing Linguistic Data: english, regularity* datasets
- Open Science Framework (OSF) projects: all other datasets

### 10.1 `english` lexical decision and naming latencies

Description from `languageR` documentation:

“This data set gives mean visual lexical decision latencies and word naming latencies to 2284 monomorphemic English nouns and verbs, averaged for old and young subjects, with various predictor variables.”

The original source of this data is the English Lexicon Project.

To load this data and learn what different columns mean, execute in R:

```
library(languageR)  
?english
```

In this book we often use `RTlexdec`, `Word`, `AgeSubject`, `WrittenFrequency`, `LengthInLetters`.

### 10.2 Dutch regularity

This dataset, originally from the study reported by Baayen and Prado Martin (2005), is described in the `languageR` documentation:

Regular and irregular Dutch verbs and selected lexical and distributional properties.

To load this data and learn what different columns mean, execute in R:

```
library(languageR)  
?regularity
```

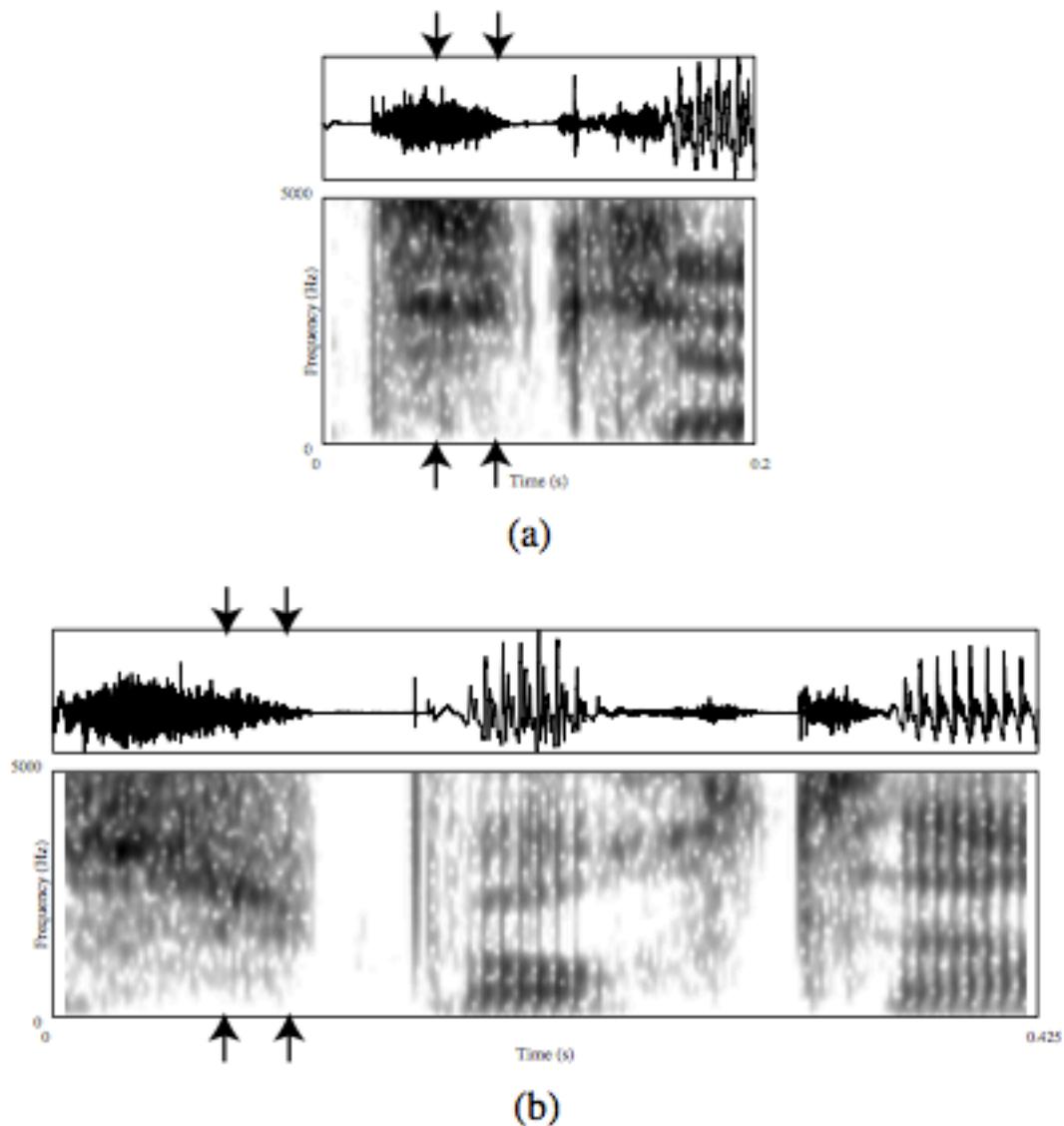
In this book we often use `WrittenFrequency`, `Verb`, `Auxiliary`, `Regularity`:

## 10.3 European French phrase-medial vowel devoicing

This dataset is from Torreira and Ernestus (2010), a study examining phrase-medial vowel devoicing in European French. The data is posted as `french_medial_vowel_devoicing.txt` in the OSF project (Torreira and Ernestus, 2018).

### 10.3.1 Background

This is an example from the paper:



**Figure 1: Examples of complete vowel devoicing in the words (a) *ticket* /tikɛ/ ‘ticket’ and (b) *supporter* /syporter/ ‘sports fan’. Arrows indicate the approximate location of the devoiced vowel.**

The `french_medial_vowel_devoicing` data (or just `devoicing`) consists of:

- 550 French syllables with voiceless obstruent onset (/pktsf/) followed by a high vowel (/iuy/),
- located in non-final position within the intonational phrase (IP)
- extracted from a corpus of spontaneous conversational speech (Torreira et al., 2010).

One way to measure to what extent “devoicing” has occurred is syllable duration.

The research question considered for this dataset (in this book) is: **are function words (e.g. “qui”, “tu”, “si”) shorter than other syllables?**

Variables:

- Response variable: **syllable duration (syldur)**
- Predictors:
  - **word type** (function/content) (**func**)
  - **speech rate** (**speechrate**)
  - **onset type** (**c1**)
  - **vowel type** (**v**)

## 10.4 North American English tapping

This data is from a speech production experiment by Kilbourn-Ceron (2017), Kilbourn-Ceron et al. (2017) examining tapping in North American English. The dataset, which contains only a subset of the data analyzed in the publications, is posted as `tappedMcGillLing620.csv` in an OSF project (Wagner et al., 2018).

### 10.4.1 Background

In North American English, the sounds [t] and [d] can sometimes be optionally pronounced as [ ] (a “tap”) if followed by a vowel:

- “For those of you who’d like to eat early, lunch will be served.”
- “For those of you who’d like to ea early, lunch will be served.”

According to earlier work, tapping interacts with syntax (e.g Scott and Cutler (1984)):

- Sounds OK: “For those of you who’d like to eat, early lunch will be served.”
- Doesn’t: “For those of you who’d like to ea , early lunch will be served.”

It seems that a syntactic juncture following [t/d] makes tapping less likely. But is this true? The effect could result either from

1. A syntactic juncture
2. A stronger prosodic boundary, which correlates with a syntactic juncture

Kilbourn-Ceron (2017), Kilbourn-Ceron et al. (2017) report a production experiment to investigate.

Participants produced sentences like:

- “If you plit, Alice will be mad”
- “If you plit Alice, John will be mad”

for nonce words like “plit” ending in [t], or [d], followed by a vowel initial word (such as “Alice”).

The two manipulations are:

1. **syntax**: the nonce word can be
  - *intransitive* (next word = following clause)
  - *transitive* (next word = complement)
2. **speakingRate**:

- Normal vs. fast

The research questions are:

1. How often did participants tap, depending on `speakingRate` and `syntax`
  - This is a categorical response variable, `tapped` (0/1).
2. Does tapping rate depend on the prosodic juncture between words?
  - The juncture strength is estimated using the duration of the preceding vowel: `vowelDuration`.

## 10.5 halfrhyme: English half-rhymes

This data is from a speech perception experiment by Harder (2013), examining what determines how “good” English speakers think imperfect rhymes are (e.g. *time/tide*). The dataset is posted as `halfrhymeMcGillLing620.csv` in an OSF project (Wagner, 2018a).

## 10.6 givenness data: the Williams Effect

This data is from a speech production experiment, reported in Wagner (2012), examining how information structure affects which words in a sentence are pronounced with more emphasis (“prominence”). The dataset is posted as `givennessMcGillLing620.csv` in an OSF project (Wagner, 2018b).

### 10.6.1 Background

When we speak, some words have more emphasis than others: this is an aspect of prosody that linguists call *prominence*. (For example, if you read the preceding sentence out loud, “prominence” is probably emphasized.) The opposite of prominence is *reduction*, where a word is produced without any emphasis.

Constituents (e.g. words) that are “given” or salient in discourse (usually because they have just been used before) and constituents whose referents are highly salient (for example the referent of a pronoun may be given, even if the pronoun itself has not been used before) are often prosodically reduced. For example, the referring expression “John” in the second sentence below is unlikely to carry an accent, and instead prominence (shown with bold) is likely to be shifted to “greeted” (shifted compared to where prominence would have fallen had the first sentence not been there):

- “John finally arrived at the function. Mary **greeted** John.”

A systematic exception is cases in which a constituent is contrastive in addition to being given. Consider the following case, where the second sentence marks a double contrast: the agent “Mary” is contrasted with the agent of the previous sentence, “John”; in addition, the patient “John” is contrasted with the patient of the previous clause, “Mary”. The fact that both “John” and “Mary” are given does not seem to be relevant if they are also contrastive:

- “John kissed Mary. Then **Mary** kissed John.”

We can conclude that contrast (marked by accenting) trumps givenness (marked by lack of accentuation). This dataset was used by Wagner (2012) to investigate a systematic class of apparent counterexamples, which were first observed by Williams (1980). According to Williams, in these cases accenting a contrastive referring expressions sounds odd, contrary to what we would expect from a semantic point of view:

- # “**John** kissed Mary. Then John was kissed by **Mary**.”

That is, even though “Mary” is contrastive in the second sentence, and thus should be accented, doing so sounds odd (indicated by “#”). Wagner’s experiment, and the analysis used in this book, tests whether this *Williams effect* is real, and how it affects the way speakers produce sentences like the one above (with “#”).

The dataset `givenness` contains data from the experiment, where participants produced sentences of four types:

1. John greeted Mary, and then John was greeted by Mary.
2. Mary was greeted by John, and then John was greeted by Mary.
3. John greeted her, and then John was greeted by her.
4. She was greeted by John, and then John was greeted by her.

These four sentences constitute one `item` in the dataset. There are many other items, each with a similar structure: two clauses, with a meaning equivalent to “noun<sub>1</sub> verbed noun<sub>2</sub>, and then noun<sub>2</sub> verbed noun<sub>1</sub>”, for different choices of the verb, noun<sub>1</sub>, and noun<sub>2</sub>.

The sentences convey the same information in four ways. In each one, the second clause is passive, and the NP referring to Mary appears at the end of the clause. The four sentences differ in whether the NP referring to Mary is a full NP (her name) or a pronoun, and whether the NP referring to Mary appears at the end of the first clause, or not:

- Full NP, at the end of clause 1
- Full NP, not at the end of clause 1
- Pronoun, at the end of clause 1
- Pronoun, not at the end of clause 1

This experiment thus has a *2x2 design*, with two within-item variables. Within an item, the sentences corresponding to #1-#4 were seen by four different `participant`s, and each participant saw exactly 1 sentence from each item. (Thus, this experiment has a *latin-square design*.)

The experiment examined what factors affect whether the final NP (henceforth the *target NP*) in sentences like #1-#4 is accented (as in (a) below), or whether the accent is shifted to an earlier word (as in (b)-(d)).

- a. **John** greeted **Mary**, and then John was greeted by **Mary**.
- b. **John** greeted **Mary**, and then **John** was greeted by Mary.
- c. **John** greeted **Mary**, and then John was **greeted** by Mary.
- d. **John** greeted **Mary**, and then John was greeted **by** Mary.

We consider four **predictors** which could affect whether stress is shifted:

- `conditionLabel` (within-item variable 1): Whether the target NP (the last word of the second clause) previously appears
  - at the beginning of clause 1 (level *Contrast*)
  - at the end of clause 1 (level *Williams*), in which case both clauses end with the same word.
- `npType` (within-item variable 2): Whether the second NP is a full NP or a pronoun
  - corresponds to the two-level factor `npType` (levels: *full, pronoun*).
- `voice`: Whether clause 2 is active or passive.
  - (Note that the values of `voice` and `conditionLabel` together determine whether clause 1 is active or passive.)
- `order`: Stimulus presentation order (within each participant).<sup>1</sup>

Why these predictors? In order:

- The main motivation of the experiment was the observation that sentences like the one marked with “#” above sound odd, at least when the final word is accented.
  - Cases like `conditionLabel=Williams`, where two consecutive sentences end with identical accented phonological chunks, are dispreferred.
  - This is called the *Williams effect* by Wagner.
- Speakers might be more or less willing to stress pronouns versus full NPs in general, or the Williams effect might differ in strength between the two types of NPs.

---

<sup>1</sup> Although `order` can only take on a discrete set of values, in applications in this book we treat it as a continuous variable (not a factor).

- Whether clause 2 is active or passive affects what word stress would be shifted to, which might also interfere with the Williams effect.
- Finally, how much participants shift stress might change over the course of the experiment as they get used to the stimuli.

Examples in this book consider two **response** variables indicating whether prominence has shifted.

1. The variable **shifted** is a binary (0/1) factor indicating whether a research assistant heard prominence as shifted or not (levels *shift*, *noshift*)
2. The variable **acoustics** is a linear combination of various acoustic cues that acts as a proxy for **shifted**.<sup>2</sup>
  - The acoustic cues are related to pitch, duration, and intensity of the target NP.
  - A higher **acoustics** value correlates with a lowering of the prominence of the target NP relative to earlier words in clause 2, and hence whether stress has been shifted from the target NP (higher **acoustics**) or not (lower **acoustics**).

## 10.7 alternatives

This data is from a speech production experiment, reported in Wagner (2016), examining how information structure affects which words in a sentence are pronounced with more emphasis (“prominence”). The dataset, which contains only a subset of the data analyzed in the paper, is posted as `alternativesMcGillLing620.csv` in an OSF project (Wagner, 2018c).

### 10.7.1 Background

In the experiment, participants read sentences like “She brought a new bicycle”, where an adjective modifies a noun. The manipulation was whether a contrast to new was mentioned in the context. There were three conditions, captured by a three-level variable **context**:

1. *New* (No previous mention of bicycle)
  - Ex: “Guess what John’s aunt, who is incredibly generous, brought for his birthday: A new bicycle!”
2. *NoAlternative* (previous mention of bicycle, but no true alternatives to “new bicycle”)
  - Ex: “Guess what John’s aunt, who produces expensive bicycles, brought for his birthday: A new bicycle!”
3. *Alternative* (previous mention of bicycle, with a true alternative to “new bicycle”):

The question of interest is: when do speakers shift prominence from “bicycle” to “new”? In condition (1), presumably they don’t, because “bicycle” has not been mentioned before. But what about (2) vs. (3)?

The **response** is the binary (0/1) variable **shifted**, which captures whether prominence was shifted to the adjective (as perceived by a research assistant).

---

<sup>2</sup>Importantly, **acoustics** was artificially constructed, and should not be used for any re-analysis or replication of the original experiment. In the original experiment, only **shifted** was annotated, and various acoustic prosodic measures (pitch, duration and intensity of different words) automatically extracted. We constructed the **acoustics** variable for teaching purposes, by running a logistic regression predicting **shifted** from the prosodic measures, and taking the resulting linear predictor from the regression to be **acoustics**. This allows us to analyze the same dataset using both linear regression (`response = acoustics`) and logistic regression (`response = shifted`).

## 10.8 VOT

This dataset contains voice onset times (VOTs) measured for speech from a corpus of speakers of different British English dialects—a subset of the VOT data analyzed by Sonderegger et al. (2017). The data is posted as `votMcGillLing620.csv` in an OSF project (Sonderegger et al., 2018).

### 10.8.1 Background

The dataset contains VOT measurements (in msec) for 4728 word-initial voiceless stops, corresponding to 424 word types and 21 speakers. Only stops beginning with /t/ and /k/ are included (/p/-initial stops have been omitted, to simplify the analysis). Like most corpus data, this dataset is very unbalanced: there are between 1 and 1505 tokens per word type, and 35-619 tokens per speaker.

The data frame contains columns corresponding to a number of variables, at the level of the speaker, the word, and individual observations:

- Speaker-level
  - `'maleSpeaker`: 1/0 for tokens from male/female speakers.
  - `speakingRateMean`: Mean speaking rate (syllables/second) for data points from this speaker.
- Word-level
  - `poaVelar`: 1/0 when place of articulation is velar/alveolar (i.e. /k/ vs. /t/)
  - `followingHigh`: 1/0 when the following vowel is high/non-high (e.g. tea vs. tap)
  - `stressedSyll`: 1/0 when the syllable containing the stop is stressed/unstressed (including function words).
- Observation-level
  - `speakingRateDev`: Speaking rate for this observation, minus `speakingRateMean` for this speaker.

The expected effects of some of these predictors on VOT can be intuitively understood as consequences of reduction. Faster speaking rate means (by definition) that speech is compressed, including the part of stop consonants corresponding to VOT. This compression might take place across speakers (those who talk faster, on average, have lower VOTs), within speakers (faster speech  $\Rightarrow$  lower VOT, for a single speaker), or both. Unstressed syllables are often realized as more reduced phonetically, corresponding to shorter segments (and hence shorter VOT). The source of the place of articulation effect on VOT is more mysterious (there are several proposed explanations), but has been observed across many languages (Cho and Ladefoged, 1999). The source of the gender effect is also not totally clear, and studies differ on whether men truly have lower VOTs than women, or whether they just speak faster than women, on average (which is true for English) (Morris et al., 2008).

The response is `logVOT`. We log-transform VOT because this brings its distribution closer to normality, and because VOT can only be positive for voiceless stops, hence a model which can predict negative VOT would not make sense.

## 10.9 Transitions

This dataset is from Roberts et al. (2015), a study examining the factors which determine the speed of turn-taking in conversation. The data is posted as `transitions.txt` in an OSF project (Roberts et al., 2018).

The dataset contains around 20,000 conversational transitions between speakers engaged in spontaneous conversation during telephone calls, from the Switchboard Corpus (Godfrey and Holliman, 1997). The dataset contains fifty variables, but only some of them are used in this book. Here is a brief explanation of some relevant variables (columns):

- `dur`: the duration (ms) of the floor transition between turn A (the turn preceding the floor transition), and turn B (the turn following the transition)
- `spkA`: the id of the speaker of the turn preceding the floor transition.
- `spkB`: the id of the speaker of the turn following the floor transition.
- `sexA`: the sex of the speaker of the turn preceding the floor transition.
- `sexB`: the sex of the speaker of the turn following the floor transition.
- `dialActA`: the dialogue act (e.g. yes-no question, wh-question, statement, answer, backchannel) of the turn preceding the floor transition.
- `dialActB`: the dialogue act of the turn following the floor transition.
- `uttNSylA`: the number of syllables in the turn preceding the floor transition.
- `uttNSylB`: the number of syllables in the turn following the floor transition.
- `uttDurA`: the duration (ms) of the turn preceding the floor transition.
- `uttDurB`: the duration (ms) of the turn following the floor transition.

Letter A is used in the name of variables referring to the turn **preceding** the floor transition, whereas letter B is used in the name of variables referring to the turn **following** a floor transition.

## 10.10 Packages

In addition to `languageR`, packages used in this book include:

- `arm` (Gelman and Su, 2018)
- `bookdown` (Xie, 2018)
- `rms` and `Hmisc` (Harrell Jr et al., 2018; Harrell Jr, 2018)
- `influence.ME` (Nieuwenhuis et al., 2012)
- `lsmeans` and successor `emmeans` (Lenth, 2016, 2018)
- “Tidyverse” packages `ggplot2`, `dplyr`, `tidyR`, etc. (Wickham, 2016; Wickham et al., 2018; Wickham and Henry, 2018)



# Bibliography

- Agresti, A. (2003). *Categorical data analysis*. Wiley.
- Agresti, A. (2007). *An Introduction to Categorical Data Analysis*. Wiley.
- Baayen, R. (2001). *Word Frequency Distributions*. Kluwer Academic Publishers.
- Baayen, R. (2008). *Analyzing linguistic data*. Cambridge University Press, Cambridge.
- Baayen, R., Davidson, D., and Bates, D. (2008). Mixed-effects modeling with crossed random effects for subjects and items. *Journal of Memory and Language*, 59(4):390–412.
- Baayen, R. H. (2013). *languageR: Data sets and functions with "Analyzing Linguistic Data: A practical introduction to statistics"*. R package version 1.4.1.
- Baayen, R. H. and Prado Martin, F. M. d. (2005). Semantic density and past-tense formation in three germanic languages. *Language*, 81(3):666–698.
- Barr, D. J., Levy, R., Scheepers, C., and Tily, H. J. (2013). Random effects structure for confirmatory hypothesis testing: Keep it maximal. *Journal of Memory and Language*, 68(3):255–278.
- Bates, D., Kliegl, R., Vasishth, S., and Baayen, H. (2015). Parsimonious mixed models. *arXiv preprint arXiv:1506.04967*.
- Bates, D., Mächler, M., Bolker, B., and Walker, S. (2014). Fitting linear mixed-effects models using lme4. *arXiv preprint arXiv:1406.5823*.
- Behrens, J. T. (1997). Principles and procedures of exploratory data analysis. *Psychological Methods*, 2(2):131–160.
- Belsley, D., Kuh, E., and Welsch, R. (1980). *Regression Diagnostics. Identifying Influential Data and Sources of Collinearity*. John Wiley & Sons, New York.
- Bland, M. (2015). *An introduction to medical statistics*. Oxford University Press.
- Chatterjee, S. and Hadi, A. (2012). *Regression Analysis by Example*. John Wiley & Sons, Hoboken, NJ, 5th edition.
- Cho, T. and Ladefoged, P. (1999). Variation and universals in VOT: evidence from 18 languages. *Journal of Phonetics*, 27(2):207–229.
- Chung, Y., Rabe-Hesketh, S., Dorie, V., Gelman, A., and Liu, J. (2013). A nondegenerate penalized likelihood estimator for variance parameters in multilevel models. *Psychometrika*, 78(4):685–709.
- Clark, H. H. (1973). The language-as-fixed-effect fallacy: A critique of language statistics in psychological research. *Journal of Verbal Learning and Verbal Behavior*, 12(4):335–359.
- Crawley, M. J. (2015). *Statistics: an introduction using R*. Wiley, second edition edition.
- Dalgaard, P. (2008). *Introductory statistics with R*. Springer, New York, NY, 2nd edition.

- Faraway, J. J. (2016). *Extending the linear model with R: generalized linear, mixed effects and nonparametric regression models*. CRC Press Taylor & Francis Group.
- Gelman, A. and Hill, J. (2007). *Data analysis using regression and multilevel/hierarchical models*. Cambridge University Press, Cambridge.
- Gelman, A. and Su, Y.-S. (2018). *arm: Data Analysis Using Regression and Multilevel/Hierarchical Models*. R package version 1.10-1.
- Godfrey, J. and Holliman, E. (1997). *Switchboard-1 Release 2*. Linguistic Data Consortium, Philadelphia.
- Harder, L. (2013). Feature mismatch in half-rhymes. *Unpublished MA Research Paper McGill University*.
- Harrell, F. (2001). *Regression modeling strategies: with applications to linear models, logistic regression, and survival analysis*. Springer Verlag, New York.
- Harrell Jr, F. E. (2018). *rms: Regression Modeling Strategies*. R package version 5.1-2.
- Harrell Jr, F. E., with contributions from Charles Dupont, and many others. (2018). *Hmisc: Harrell Miscellaneous*. R package version 4.1-1.
- Jaeger, T. (2008). Categorical data analysis: Away from ANOVAs (transformation or not) and towards logit mixed models. *Journal of Memory and Language*, 59(4):434–446.
- Johnson, K. (2008). *Quantitative methods in linguistics*. Wiley-Blackwell, Malden, MA.
- Kilbourn-Ceron, O. (2017). *Speech production planning affects variation in external sandhi*. PhD thesis, McGill University.
- Kilbourn-Ceron, O., Wagner, M., and Clayards, M. (2017). The effect of production planning locality on external sandhi: a study in /t/. In *Proceedings of the 52nd Annual Meeting of the Chicago Linguistic Society*.
- Kruschke, J. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.
- Lenth, R. (2016). Least-squares means: The R package lsmeans. *Journal of Statistical Software*, 69(1):1–33.
- Lenth, R. (2018). *emmeans: Estimated Marginal Means, aka Least-Squares Means*. R package version 1.2.3.
- Levy, R. (2012). *Probabilistic methods in the study of language*. May 17, 2012 version of draft in progress.
- Matuschek, H., Kliegl, R., Vasishth, S., Baayen, H., and Bates, D. (2017). Balancing Type I error and power in linear mixed models. *Journal of Memory and Language*, 94:305–315.
- McElreath, R. (2015). *Statistical Rethinking: A Bayesian Course with Examples in R and Stan*. Chapman & Hall/CRC.
- Morris, R. J., Mccrea, C. R., and Herring, K. D. (2008). Voice onset time differences between adult males and females: Isolated syllables. *Journal of Phonetics*, 36(2):308–317.
- Navarro, D. (2015). *Learning Statistics with R*. School of Psychology, University of Adelaide, Adelaide, Australia. Version 0.5. [Lecture notes].
- Nicenboim, B. and Vasishth, S. (2016). Statistical methods for linguistic research: Foundational Ideas—Part II. *Language and Linguistics Compass*, 10(11):591–613.
- Nieuwenhuis, R., Te Grotenhuis, M., and Pelzer, B. (2012). influence.me: Tools for detecting influential data in mixed effects models. *R Journal*, 4(2):38–47.
- Roberts, S. G., Torreira, F., and Levinson, S. C. (2015). The effects of processing and sequence organization on the timing of turn taking: a corpus study. *Frontiers in Psychology*, 6:1–16. Article 509.

- Roberts, S. G., Torreira, F., and Levinson, S. C. (2018). The effects of processing and sequence organization on the timing of turn taking: a corpus study.
- Salverda, A. P., Dahan, D., and McQueen, J. M. (2003). The role of prosodic boundaries in the resolution of lexical embedding in speech comprehension. *Cognition*, 90(1):51–89.
- Schad, D. J., Hohenstein, S., Vasishth, S., and Kliegl, R. (2018). How to capitalize on a priori contrasts in linear (mixed) models: A tutorial. *arXiv preprint arXiv:1807.10451*.
- Schielzeth, H. and Forstmeier, W. (2009). Conclusions beyond support: Over-confident estimates in mixed models. *Behavioral Ecology*, 20(2):416–420.
- Scott, D. R. and Cutler, A. (1984). Segmental phonology and the perception of syntactic structure. *Journal of Verbal Learning and Verbal Behavior*, 23(4):450–466.
- Snijders, T. and Bosker, R. (2011). *Multilevel analysis*. Sage, London, 2nd edition.
- Sonderegger, M., Bane, M., and Graff, P. (2017). The medium-term dynamics of accents on reality television. *Language*, 93(3):598–640.
- Sonderegger, M., Bane, M., and Graff, P. (2018). Medium-term dynamics of accents on reality TV.
- Sorensen, T., Hohenstein, S., and Vasishth, S. (2015). Bayesian linear mixed models using stan: A tutorial for psychologists, linguists, and cognitive scientists. *arXiv preprint arXiv:1506.06201*.
- Speelman, D., Heylen, K., and Geeraerts, D., editors (2018). *Mixed-Effects regression models in linguistics*. Springer.
- Torreira, F., Adda-Decker, M., and Ernestus, M. (2010). The Nijmegen corpus of casual French. *Speech Communication*, 52(3):201–212.
- Torreira, F. and Ernestus, M. (2010). Phrase-medial vowel devoicing in spontaneous french. In *11th Annual Conference of the International Speech Communication Association (Interspeech 2010)*, pages 2006–2009.
- Torreira, F. and Ernestus, M. (2018). French devoicing.
- Tukey, J. W. (1980). We need both exploratory and confirmatory. *The American Statistician*, 34(1):23–25.
- Vasishth, S. (2014). An introduction to statistical data analysis. *Summer 2014 version. Available at https://github.com/vasishth/Statistics-lecture-notes-Potsdam/*.
- Vasishth, S. and Broe, M. B. (2011). *The foundations of statistics: A simulation-based approach*. Springer.
- Vasishth, S. and Nicenboim, B. (2016). Statistical methods for linguistic research: Foundational ideas—part i. *Language and Linguistics Compass*, 10(8):349–369.
- Venables, W. and Ripley, B. (2002). *Modern applied statistics with S*. Springer, New York.
- Wagner, M. (2012). A givenness illusion. *Language and Cognitive Processes*, 27(10):1433–1458.
- Wagner, M. (2016). Information structure and production planning. In Féry, C. and Ishihara, S., editors, *The Oxford Handbook of Information Structure*, pages 541–561. Oxford University Press.
- Wagner, M. (2018a). Asymmetries in half-rhyme.
- Wagner, M. (2018b). A givenness illusion.
- Wagner, M. (2018c). Information structure and production planning.
- Wagner, M., Kilbourn-Ceron, O., and Clayards, M. (2018). The effect of production planning locality on external sandhi: A study in /t/.
- Wickham, H. (2016). *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York.

- Wickham, H., François, R., Henry, L., and Müller, K. (2018). *dplyr: A Grammar of Data Manipulation*. R package version 0.7.6.
- Wickham, H. and Henry, L. (2018). *tidyverse: Easily Tidy Data with 'spread()' and 'gather()' Functions*. R package version 0.8.1.
- Williams, E. (1980). Remarks on stress and anaphora. *Journal of Linguistic Research*, 1(3):1–16.
- Wurm, L. H. and Fisicaro, S. A. (2014). What residualizing predictors in regression analyses does (and what it does not do). *Journal of Memory and Language*, 72:37–48.
- Xie, Y. (2018). *bookdown: Authoring Books and Technical Documents with R Markdown*. R package version 0.7.
- Zuur, A. F. (2009). *Mixed effects models and extensions in ecology with R*. Springer.