

Bayesian regression modeling for (factorial designs): A tutorial

Michael Franke & Timo Roettger

Generalized linear mixed models are very well-rounded and handy tools for statistical inference. Bayesian approaches to applying these models have recently become increasingly popular. This tutorial provides an accessible, non-technical introduction to the use and feel of Bayesian mixed effects regression models. The focus is on data from a factorial-design experiment.

This tutorial should take you about 1 hour.

Motivation & intended audience

This tutorial provides a very basic introduction to Bayesian regression modeling using R (R Core Team, 2017). We wrote this tutorial with a particular reader in mind. If you have used R before and if you have a basic understanding of linear regression, and now you want to find out what a Bayesian approach has to offer, this tutorial is for you. In comparison to other introductions (e.g. Sorensen, Hohensteinb, and Vasishth, 2016), this tutorial remains very conceptual. We don't want to "sell Bayes" to you, and we do not want to scare you away with mathematical details. We just want to give you an impression of how a Bayesian regression analysis looks and feels. So no reason to be afraid! But also: no reason to be bored, because we *will* cover all the essential concepts and we *will* explain how to run and interpret the output of a Bayesian regression analysis using the wonderful R package `brms` written by Paul Buerkner (2016).

If you don't have any experience with regression modeling, you will probably still be able to follow, but you might also want to consider doing a crash course. To bring you up to speed, we recommend the excellent two-part tutorial by Bodo Winter (2013) on mixed effects regression in a non-Bayesian—a.k.a. classical or frequentist—paradigm. In a sense, this tutorial could be considered part three of Bodo's nice and lofty introduction. We will for example use the same data set.

This tutorial contains text boxes (with a gray background) which contain additional background information on some topics. The information is sometimes a bit technical but never absolutely necessary for understanding the main ideas. So feel free to read or skip any of the text boxes to suit your needs.

To follow this tutorial, you should have R installed on your computer (<https://www.r-project.org>). Unless you already have a favorite editor for tinkering with R scripts, we recommend to try out RStudio (<https://www.rstudio.com>). You will also need some packages,

Remember that you can install a package called XYZ with the command `install.packages("XYZ")`.

which you can import with the following code:

```
1 # package for convenience functions (e.g. plotting)
2 library(tidyverse)
3
4 # package for Bayesian regression modeling
5 library(brms)
6
7 # option for Bayesian regression models:
8 # use all available cores for parallel computing
9 options(mc.cores = parallel::detectCores())
10
11 # package for credible interval computation
12 library(HDIInterval)
13
14 # set the random seed in order to make sure
15 # you can reproduce the same results
16 set.seed(1702)
```

All code and data for this tutorial is also available for download here: https://github.com/michael-franke/bayes_mixed_regression_tutorial

Data, research questions & hypotheses

As experimental researchers we collect data to answer questions of interest about how nature works. This tutorial looks at a data set relevant for investigating whether voice pitch differs across female and male speakers, and whether it differs across social contexts (say: informal and polite contexts). To load the data into your R environment, run the following code:

```
1 # load the data into variable 'politedata'
2 politedata = read_csv("https://raw.githubusercontent.com/michael-
  franke/bayes_mixed_regression_tutorial/master/code/
  politeness_data.csv")
```

The data is originally from research presented by Winter and Grawunder (2012). It is used in the tutorials by Winter (2013) as well, but here we ‘massaged’ the data a bit, i.e., we renamed variables and removed a line with missing data.

Type `head(politedata)` and you should see the first lines of the data:

```
1 > head(politedata)
2   subject gender sentence context pitch
3   <chr>   <chr>  <chr>   <chr>  <dbl>
4 1 F1      F      S1      pol    213.
5 2 F1      F      S1      inf    204.
6 3 F1      F      S2      pol    285.
7 4 F1      F      S2      inf    260.
8 5 F1      F      S3      pol    204.
```

Here, we show only part of the output that you can see when executing this command.

This data set contains information about different subjects, with an anonymous identifier stored in variable `subject`. Because voice pitch is highly dependent on gender (i.e. there are anatomical differences between women and men that affect voice pitch), we stored whether our subjects are F(emale) or M(ale) in variable `gender`. Subjects produced different sentences (stored

in variable `sentence`), and the experiment manipulated whether the sentence was produced in a polite or an informal context, indicated by the variable `context`. Crucially, each row contains a measurement of pitch in Hz stored in variable `pitch`.

Often, we are interested in comparing a **dependent variable** (here `pitch`) across different conditions or groups, i.e. **independent variables** (here `gender` and `context`). Before our data collection, we might have formulated concrete predictions about the relationship between the dependent variable and the independent variables. For example, we might have formulated the following three hypotheses:

- H1: Female speakers have a lower average pitch in polite than in informal contexts.
- H2: Male speakers have a lower average pitch in polite than in informal contexts.
- H3: Male speakers have a lower average pitch in informal than female speakers have in polite contexts.

Exploring the data visually

To get a first idea of possible relationships in our data, let's plot them. Figure 1 displays the mean pitch values for each sentence (semi-transparent points) across gender and context. The solid points indicate the average pitch values across all sentences and speakers. Looking at the plot, we can see that pitch values from female speakers are generally higher than those from male speakers (points in the left column are higher than in the right column). We also see that pitch values in the informal context are slightly higher than those in the polite context (orange points are slightly higher than purple points).

Looking at the plot, we might want to shout: "Eureka! The data confirm all of our hypotheses!" But, of course, we need to be more careful. As Bayesians, we would like to translate the data into an expression of **evidence**: does the data provide evidence for our research hypotheses? Or are the observable differences too meager? – Also, notice that there is quite a lot of variability between different sentences (the semi-transparent points in Figure 1). For example, some values from the informal condition for female speakers (orange points in left column), are lower than their corresponding polite counterparts. Similarly, there could be differences between individual speakers. What we want are precise estimates of potential differences between conditions. We also want a measure of how confident we can be in these estimates.

Extensive plotting is always recommended to start data analysis. You need to know your data inside out. Pictures often reveal relationships much better than numbers can.

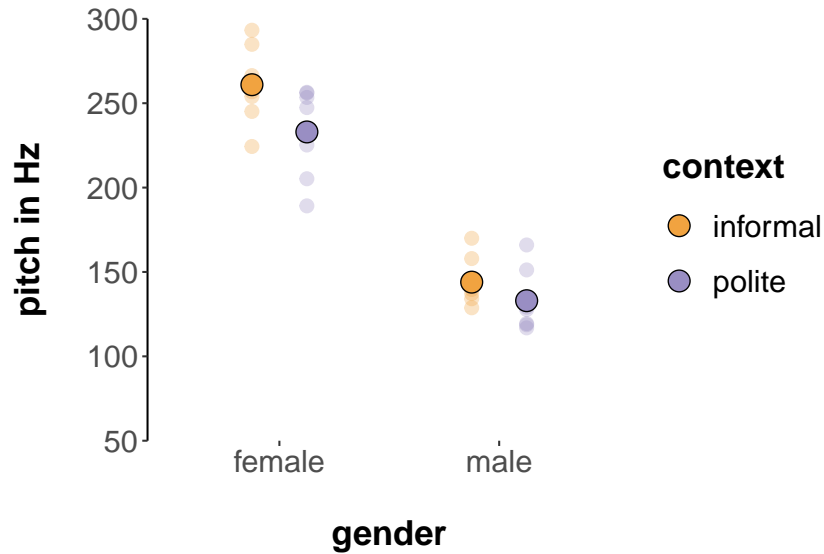


Figure 1: Basic plot of the data displaying overall averages (thick points) and averages for individual sentences (smaller semitransparent points).

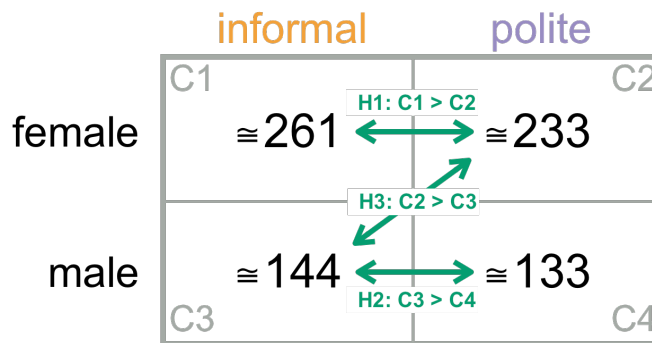


Figure 2: Means of each design cell, together with research hypotheses as statements about ordinal relations between cell means.

A regression model for our data

Another way of looking at the data in connection with our research hypotheses is displayed in Figure 2. Each cell in this **design matrix** represents one unique combination of the gender and the context factor, and the table shows the mean of observed pitch values for each cell. Our hypotheses can be related to the comparison between some of these cell-based means. H1 makes a statement about the comparison between C(ells) 1 and 2 (the context effect for female speakers); H2 makes a statement about C3 and C4 (the context effect for male speakers); and H3 makes a statement about C2 and C3 (the difference between informal male speakers and polite female speakers).

One way of testing our hypotheses using a Bayesian approach to data

	informal	polite
female	C1 β_0 reference level	C2 $\beta_0 + \beta_{\text{pol}}$
male	C3 $\beta_0 + \beta_{\text{male}}$	C4 $\beta_0 + \beta_{\text{male}} + \beta_{\text{pol}} + \beta_{\text{pol\&male}}$

Figure 3: Coefficients of a dummy-coded regression model for the factorial 2×2 design.

analysis, is to ask whether the relevant differences between cell means are *credibly different from zero*. This is jargon for asking whether, given the observed data, we should believe that the relevant cell means are different from each other. The fact that we are now in Bayesian land should hit us when we hear the phrases “credibly different” and “should believe”. Bayesian thinking is about updating beliefs based on observations. And we express our beliefs as as probability distributions. At first, this may appear scary or technically involved. But at the end of the day, the intuitions captured by this approach are arguably very natural, and perhaps even easier to understand than the reasoning underlying other approaches to statistical inference. Let’s walk through our Bayesian approach step-by-step.

First, let’s look at the **regression model** we want to use. Our regression model assumes that pitch values observed in each cell are sampled from a population that is normally distribution, where each cell c_i has its own mean μ_i . We are interested in the probability of one cell mean being larger than another cell mean, i.e., the probability that $\mu_i > \mu_j$. Put differently, we are interested in the probability that the difference between μ_i and μ_j is larger than zero: $\mu_i - \mu_j > 0$. Figure 7 illustrates the encoding scheme of our cell means in terms of a regression analysis. It assumes that there is a **reference level** for each factor. Here it is the level `female` for the factor `gender` and the level `informal` for the factor `context`.

All cell means can then be expressed in terms of differences between the **intercept** β_0 which is the cell mean of the reference level (here, the cell mean of female subjects in informal contexts), deviations from this **reference cell** for each individual factor (β_{male} , and β_{polite}), and a so-called **interaction term** $\beta_{\text{pol\&male}}$. In other words, our regression estimates the mean of the reference level and estimates how much we need to adjust this mean when we change either the context level (C2), the gender level (C3), or both (C4). The β terms are also called **coefficients**. They are **free parameters** of the model.

This is so-called **dummy coding** of the regression coefficients. Other coding schemes exist, but are not discussed here.

Bayesian inference: priors, likelihoods and posteriors

Jones is a rational scientist. She has recently inherited her grandma's lucky coin. Grandma used this coin many times during Jones' childhood to determine whether Jones was allowed a sweet or not. Jones suspects that grandma's coin might be a trick coin, but she is not sure. She is determined to find out. How? Well, naturally, by rationally updating her *prior beliefs* about the coin's bias to obtain a new *posterior belief* based on empirical observation (outcomes of coin flips). Central to this updating is Jones' *likelihood function*, which encodes how likely each relevant coin bias may have generated the observed data. — Sounds fancifully abstract? It's actually fairly intuitive. Consider this example.

Prior beliefs. Jones knows that there are two factories who produce coins. One produces fair coins, the other produces coins which give heads three times more often than tails. She believes that it's equally likely that the coin is from either factory. Numerically, Jones' *prior beliefs* can be written as, where $\theta \in [0; 1]$ is the coin's bias: $P(\theta = 1/2) = 1/2$, and $P(\theta = 3/4) = 1/2$.

Likelihood. The bias θ is, by definition, the probability of the coin landing heads on the next trial. Let's assume that Jones tosses the coin only once (hm, maybe not so rational a scientist after all? or just too busy?). If $D = \{\text{heads}, \text{tails}\}$ is the set of potential outcomes of this experiment, the *likelihood function* determines the likelihood of observing each observation $d \in D$ for each θ , which in our case is just rather trivial: $P(D = \text{heads} | \theta) = \theta$ and $P(D = \text{tails} | \theta) = 1 - \theta$.

Posterior beliefs. Jones observes that the coin landed heads. What should she believe now? By *Bayes rule* her posterior beliefs are defined as:

$$P(\theta | D = \text{heads}) = \frac{P(\theta)P(D = \text{heads} | \theta)}{\sum_{\theta'} P(\theta')P(D = \text{heads} | \theta')}$$

Jones' posterior belief that the coin is twice as likely to land heads is therefore:

$$\begin{aligned} P(\theta = 3/4 | D = \text{heads}) &= \\ &= \frac{P(\theta = 3/4) P(D = \text{heads} | \theta = 3/4)}{P(\theta = 3/4) P(D = \text{heads} | \theta = 3/4) + P(\theta = 1/2) P(D = \text{heads} | \theta = 1/2)} = \\ &= \frac{1/2 \cdot 3/4}{1/2 \cdot 3/4 + 1/2 \cdot 1/2} = \frac{3/8}{7/8} = \frac{3}{5} \end{aligned}$$

After making her observation, rational Jones believes that the bias towards heads has a probability of about 0.6. So she has updated her prior belief of 0.5.

Info Box 1: Priors, likelihood and posteriors in Bayesian inference.

A Bayesian analysis of a (fixed effects) regression model

Having spelled out a model like the above, a Bayesian analysis asks: What are credible values for the model's parameters, i.e., what should a rational reasoner believe about the values of, in particular, the coefficients β_0 , β_{pol} , β_{male} and $\beta_{\text{pol}\&\text{male}}$ given the reasoner's initial beliefs and the data, on the assumption that the model is true? Formally, if θ is a vector of parameter values of the model, we are interested in the **posterior distribution** $P(\theta | D)$, which assigns a non-negative number to each tuple of parameters in proportion to how likely that tuple is. From a Bayesian point of view, the model consists of a **likelihood function** $P(D | \theta)$, which specifies how likely an observation of data D is for each value of parameters θ , and a **prior distribution** $P(\theta)$, which specifies how likely we (the rational reasoners) believe each tuple of parameter values is in the first place. With these ingredients, we can compute the posterior distribution by Bayes rule, as follows:

$$P(\theta | D) = \frac{P(\theta) P(D | \theta)}{\int P(\theta') P(D | \theta') d\theta'}$$

The R package `brms` (Buerkner, 2016) makes it easy to run Bayesian regression models. It uses a very similar formula syntax as related packages for regression analysis. In our case, we want to regress the dependent variable `pitch` against the independent variables `gender` and `context` and their two-way interaction. This model is expressed by the formula:

```
1 # formula for (fixed effects) regression model
2 formulaFE = pitch ~ gender * context
```

The Bayesian model can then be fitted with the function `brm` from the `brms` package. We only need to specify the formula and supply the data:

```
1 # run regression model in brms
2 modelFE = brm(
3   formula = formulaFE,
4   data = politedata)
```

The `brms` packages uses the probabilistic programming language `Stan` in the background. `brms` basically translates our syntax into `Stan` code and executes it. The `Stan` code is then translated to C++ (hence the message about “compiling C++” when you run this code). Conceptually, `Stan` obtains samples from the posterior distribution, based on an algorithm called *Hamiltonian Monte Carlo*. This is an instance of a more general class of algorithms, called *Markov Chain Monte Carlo* methods. The purpose of these methods is to return representative samples from the posterior distribution. If

Info Box 1 provides some background on Bayesian reasoning

Supplying a **prior**, really just *any* prior, is important to get a Bayesian analysis off the ground; a circumstance which is discussed controversially. For many practical purposes, however, the precise choice of prior is not decisive and tools like the `brms` package which we will use here will default to generically reasonable choices of priors for your model (more on this below).

you are interested in finding out more about this 'sampling stuff', check out Info Box 2?

You can type in `modelFE` in order to get a summary of the model fit. It should look like the following output.

```

1 > modelFE
2 Family: gaussian
3 Links: mu = identity; sigma = identity
4 Formula: pitch ~ gender * context
5 Data: politedata (Number of observations: 83)
6 Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
7         total post-warmup samples = 4000
8
9 Population-Level Effects:
10      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
11 Intercept      260.68      8.07   244.99   276.78      2409 1.00
12 genderM        -116.09     11.44  -138.37   -93.80      2094 1.00
13 contextpol      -27.38     11.33   -50.01    -5.98      2092 1.00
14 genderM:contextpol  15.74     16.38   -17.03    49.13      1831 1.00
15
16 Family Specific Parameters:
17      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
18 sigma      36.15       2.87    30.93    42.40      3652 1.00
19
20 Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
21 is a crude measure of effective sample size, and Rhat is the potential
22 scale reduction factor on split chains (at convergence, Rhat = 1).

```

Lines 2–5 give us information about the model and the data used. Lines 6 and 7 tell us about the sampling procedure (see Info Box 2 for more information). Lines 9–14 contain information about our parameters of interest. We will discuss them in detail below. Lines 16–18 contain information that look similar to those in lines 9–14. This is the estimation of the standard deviation σ , describing the variance of the assumed normal distributions (which describe the distribution of measures in each design cell). Finally, lines 20–22 contain general information about the model fit and the information presented in this summary.

Let us look at lines 9–14 in detail now. What these lines give us is a table with four rows, each of which corresponds to a parameter in the model, namely the coefficients shown in Figure 7. The variable `Intercept` refers to our β_0 , which represents the mean of our reference level in cell 1 (female speakers in polite contexts). The variable `genderM` corresponds to our β_{male} , `contextpol` corresponds to our β_{pol} , and `genderM:contextpol` is the interaction term $\beta_{\text{pol}\&\text{male}}$. For each of these parameters, the table contains very useful summary statistics based on the samples returned from the model fit. More details about the information given in the other columns can be found in Info Box 3.

For our current purposes, the information in columns `l-95% CI` and `u-95% CI` is most important. These numbers give the lower and upper

If the model failed to converge or other problems occurred, you would see an informative message in the last part of this summary (see Info Box 3).

Intuitively, the 95% credible interval is the range of values that we can often practically consider credible enough to care about.

MCMC sampling

Bayesian ideas are old. Why have they come to such popularity only recently? – Part of it is due to our ever growing computational resources. Bayesian inference is computationally very expensive. To understand this, consider Bayes rule for data analysis. We have a prior $P(\theta)$ over parameter vector θ , a likelihood function $P(D | \theta)$ and we want to compute the posterior distribution, using Bayes rule like so:

$$P(\theta | D) = \frac{P(\theta) P(D | \theta)}{\int P(\theta') P(D | \theta') d\theta'}$$

The problem is the integral in the denominator. If θ is a large vector of parameters (e.g., in a hierarchical regression model), it might not be all that easy (euphemistic for “quite impossible”) to crack that integral. Fortunately, clever algorithms like *Markov Chain Monte Carlo* allow us to approximate the posterior distribution without having to calculate the integral-of-doom. They do that by giving us **samples**.

- We have a total of 4000 samples from the posterior distribution, obtained from 4 chains all of which had 2000 iterations but discarded the first 1000 as warm up.

Good vs. bad samples? For this tutorial, it is not important to understand how MCMC algorithms work precisely. Let’s just accept that they give us samples from the posterior distribution, which we can use to plot or reason with, e.g., like in Figure 4. That’s nice, but it would be foolish to just trust whatever samples some piece of software gives us without some sanity checking. That’s why it is common to inspect our samples either by visual inspection using plots, and/or by certain numeric indexes of how “good” our samples really are.

To understand the most basic diagnostics (which is really all we need for common applications), we need to understand one thing about how MCMC methods generate samples. They do that just like a rather dumb frog would hop from one lily pad to another to catch flies. Call the frog “Franz”. Franz sits on a lily pad and observes the number of flies there. He then considers exactly one other lily pad and wonders: “Should I jump there?” If there are more flies, he jumps; if not, he sometimes stays, sometimes jumps (with a probability proportional to the ratio of flies at the current lily and the next one). If he does that over and over again, the probability of visiting a lily pad is proportional to its number of flies (relative to that of all other lily pads Franz might visit). Enough of frogs? Okay, the lily pads are parameter values for θ and the number of flies is the quantity $P(\theta) P(D | \theta)$. So, Franz’s visits to a lily pad are visits to a vector of parameter values, and the history of these visits gives us information about the relative posterior probability of parameter values θ . — Roughly, modulo frogs and lilies, that’s how basic MCMC works.
[tr: its not that easy to understand, I am afraid (and I know how MCMC works), is there any way to make this easier, shorter, split it off or sth?]

The problem is that we can only be sure that the visits are “good” samples from the posterior distribution if we take infinitely many. Usually, Franz is too lazy to jump around the lily pads that (and we are too busy). If we only have a small-ish number of samples, it might be that Franz was sitting on the

Information displayed in the summary of a brm model fit

The first column *Estimate* gives the mean of the obtained samples, thereby approximating the mean of the posterior distribution (beliefs we should hold) about each parameter. For example, the parameter `Intercept` is estimated to have a mean of about 261, which (here) coincides with the mean of the data points in cell 1, as shown in Figure 2. The other columns give further useful information. *Est.Error* is the estimation error, an indication of the certainty we should have about the whole inference procedure. The columns `l-95% CI` and `u-95% CI` give the lower and upper bound of the **95% credible interval** for each parameter, estimated from the posterior samples. The column `Eff. Sample`, for efficient samples, gives a rough measure of how many of all the samples we took (4000 in our case) are contributing non-redundant information to our estimation (see Info Box 2). The higher this number, the better. Finally, `Rhat` is a measure of whether the samples obtained are likely representative of the true distribution (again, see Info Box 2). Concretely, it indicates whether the four chains we ran in parallel all ended up with the same results, so to speak. If this column contains values bigger than 1.1 this is an indication that our model fit has not converged. (If your model output indicates non-convergence, you may want to increase the number of samples, but you should also consider the possibility that you are trying to fit a model which cannot be “trained” based on the (perhaps too little) data and the particular method of posterior sampling. For common regression analyses, this will usually entail considering a simpler model (e.g., with fewer explanatory factors, less (correlated) random effects, etc.))

Info Box 3: Information in summaries of `brm` model fits.

bounds of a **95% credible interval**. Take the parameter `contextpol`, corresponding to our coefficient β_{pol} . This parameter corresponds to the estimated adjustment of the mean of the reference level when we change the context level to polite. The 95% CI is roughly [-50;-6]. We could take values outside of this interval to be sufficiently unlikely to be ignorable for most purposes. Consequently, a very special value for this parameter has to be considered implausible, namely 0. In other words, this analysis suggests that 0 is a very unlikely value for the coefficient β_{pol} ; rather we should believe that β_{pol} is most likely negative. — Hurray! This directly addresses our first research hypothesis. In a research paper we could now write: “Based on the regression model, the data suggests that H1 is likely true.”

How likely is it that β_{pol} is smaller than 0? —Instead of simply making a binary thumbs-up / thumbs-down decision, it would be even more elegant, if we could put a number to it. As Bayesians, we fortunately can. To see how this works, let us have a more intimate look at the samples that the `brm` function returns. We can access the samples of a model fitted with `brm` with

the function `posterior_samples`:

```
1 # extract posterior samples
2 post_samples_FE = posterior_samples(modelFE)
3 head(post_samples_FE)
```

The output of this could look like this:

```
1 > head(post_samples_FE)
2   b_Intercept b_genderM b_contextpol b_genderM:contextpol sigma  lp__
3 1      270.6    -123.5      -22.1           16.1  36.0 -422.4
4 2      268.7    -131.0      -36.8           36.1  39.6 -421.1
5 3      255.8    -108.9      -20.2           11.7  39.2 -420.5
6 4      259.4    -115.7      -29.4           12.8  31.2 -420.8
7 5      256.5    -106.0      -15.2          -11.7  36.2 -420.8
8 6      245.8    -112.2       -9.6           17.6  39.5 -423.1
```

What you see here is the top 6 rows of a data frame with columns for each parameter and 4000 rows, corresponding to each sample of that parameter (so our sampling method has generated 4000 values for each parameter, where each row is a sample from the posterior distribution $P(\theta \mid D)$). We can use these samples to produce density plots reflecting out posterior distribution. The plot in Figure 4 shows, for each of the four main model parameters an estimate of the posterior density. Each curve shows how much credence we should put on particular parameter values. For example, we see that our beliefs concerning plausible values for the mean of cell 1 (female speakers in polite contexts, the reference cell) should hover around 261, roughly spreading from about 240 to 280. We also see that all values that receive substantial probability density for `context:pol` (our β_{pol}) are negative (as captured in the 95% CI discussed above). Zero is estimated to be a comparatively unlikely value for this parameter.

The column `lp__` contains the log-probability of the data for the parameterization in each row. This is useful for model comparison and model criticism but not important for our current adventures.

Now, here comes a nice gadget. Based on the samples obtained for `contextpol` (β_{pol}), it is very easy to estimate our belief that β_{pol} is indeed negative. We simply have to calculate the proportion of samples that were negative. That's all. For instance, with the code below, which reveals that given the data and the model, the posterior probability of the proposition that $\beta_{\text{pol}} < 0$ is about 0.99275. This is very close to 1!

```
1 # proportion of negative samples for parameter b_contextpol
2 # this number approximates P(b_contextpol < 0 | model, data)
3 mean(post_samples_FE$b_contextpol < 0)
```

As an interim summary, we have seen how to run a Bayesian regression analysis with the `brms` package and deal with its output. We have also seen

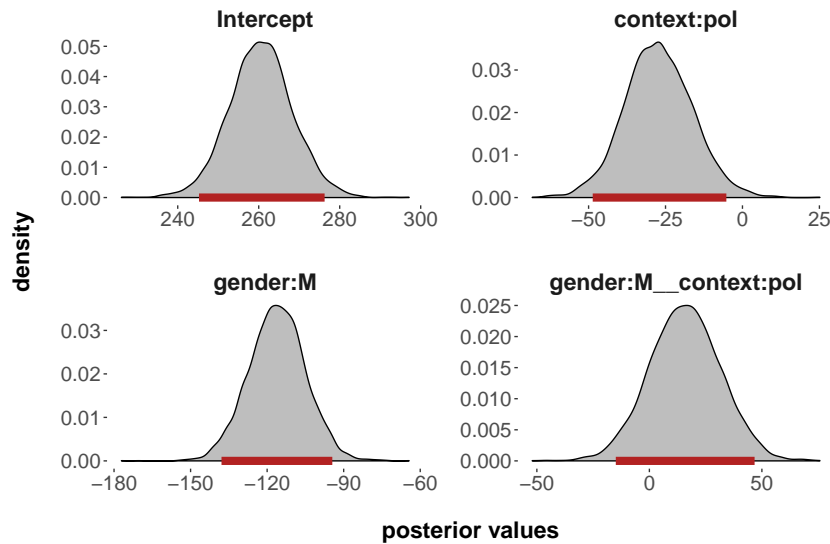


Figure 4: Posterior density of parameter values in the fixed-effects regression model. The thick red lines indicate the 95% credible intervals, i.e., the range of parameter values that it is reasonable to believe in.

that the output can be interpreted in very intuitive ways (e.g., “The probability of H1, given our model, priors, and data, is more than .99”).

Unfortunately, what we have not seen yet is what our model and data say about hypotheses 2 or 3. This is because there is no single parameter in the (dummy-coded) regression model that corresponds to the differences between cells 3 and 4 (for hypothesis 2) and cells 2 and 3 (for hypothesis 3). Notice that this problem is not specific to Bayesian analyses, but inherent in the way the regression coefficients were set up. Fortunately, we can recover information about any derived measure from the obtained samples. Here’s how:

Take hypothesis 3 which requires us to compare cells 2 and 3. The hypothesis states that $\beta_0 + \beta_{\text{pol}} > \beta_0 + \beta_{\text{male}}$, which reduces to $\beta_{\text{pol}} > \beta_{\text{male}}$. We can approximate the posterior probability that this is true based on the samples that we obtained for our model in the same general way as before, namely:

```
1 # proportion of samples where the mean for cell 2 was larger
2 # than that of cell 3
3 # this number approximates the quantity:
4 #   P(b_contextpol > b_genderM | model, data)
5 mean(post_samples_FE$b_contextpol > post_samples_FE$b_genderM)
```

Based on the posterior samples we obtained, the estimated probability is 1. That’s a strong result. If the model was true, then, given the data, our certainty that hypothesis 3 is true should be pretty much almost at ceiling.

To conclude this section, the Bayesian approach to regression modeling allows us to retrieve all direct comparisons between cells in a factorial

A potential way of testing different hypotheses of the kind we have set out here, is to run different regression analyses, each with a different reference cell. This is a rather unhandy work flow. It wouldn’t help with hypothesis 3 either, which compares the cell means in our design matrix “diagonally”: there is no way of changing the reference level of either factor such that dummy coding gives us a single coefficient as the difference between cells 2 and 3.

design (all possible comparisons). It also allows us to retrieve quantitative information about our hypotheses in a way which is easy to communicate and understand. We can calculate the (estimated) posterior probability that a particular hypothesis holds.

The `faintr` package

To make the comparison of pairs of cells even easier and applicable for even bigger factorial designs, this tutorial comes with a little R package, the `faintr` package. You can install the package from GitHub with the `devtools` package, as follows:

```
1 # load package to allow installation from GitHub
2 library(devtools)
3 # install package with convenience function for Bayesian regression
4 # models for factorial designs from GitHub
5 install_github(
6   repo = 'michael-franke/bayes_mixed_regression_tutorial',
7   subdir = 'faintr',
8   build_vignettes = TRUE)
9 # load the just installed package
10 library(faintr)
```

The `faintr` package provides two main helper functions. The function `extract_posterior_cell_means()` takes as input the fitted regression model and outputs samples for all design cell means, and a comparison of all design cells against each other. For example, although the model fitted β -coefficients, we can reconstruct posterior estimates of each cell's means by typing:

```
1 extract_posterior_cell_means(modelFE)$predictor_values
```

Using these reconstructed samples, we can plot approximate posterior estimates for each cell, as in Figure 5

Another helpful function is `compare_groups()`, which takes as input a fitted regression model together with a specification of which two (subsets of) cells to compare against each other. For example, we can compare (diagonally) the cells for female speakers in polite contexts with male speakers in informal contexts with this call:

```
1 compare_groups(
2   model = modelFE,
3   lower = list(gender = "M", context = "inf"),
4   higher = list(gender = "F", context = "pol")
5 )
```

The output looks like this:

You can read a vignette with more information on the `faintr` package if you type `vignette("faintr_basics")` after installing the package.

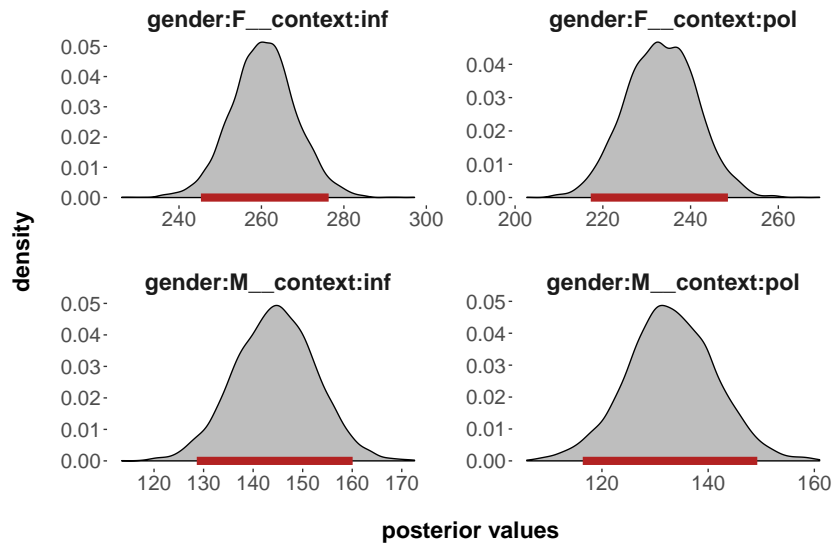


Figure 5: Posterior density of cell means in the fixed-effects regression model. The thick red lines indicate the 95% credible intervals, i.e., the range of parameter values that it is reasonable to believe in.

```

1 Outcome of comparing groups:
2 * higher: gender:F context:pol
3 * lower: gender:M context:inf
4 Mean 'higher - lower': 88.97
5 95% CI: [ 67.33 ; 110.2 ]
6 P('higher - lower' > 0): 1

```

Using the `compare_groups()` function, the source code provides a convenient function to produce the posterior probability of the three hypothesis relevant for this tutorial:

```

1 > get_posterior_beliefs_about_hypotheses_new(model_MaxRE)
2 # A tibble: 3 x 2
3   hypothesis                                probability
4   <chr>                                     <dbl>
5 1 Female-polite < Female-informal          0.973
6 2 Male-polite < Male-informal              0.850
7 3 Male-informal < Female-polite            0.922

```

Priors

One important difference between traditional frequentist' and Bayesian inference are priors. Priors are pieces of information about our data that we assume before actually looking at them. Specifying priors has several advantages. First, we can constrain the sampling procedure to realistic numbers and that reduces the computational resources needed to estimate the model parameters. For example, pitch values (and many other things we measure

in nature) cannot be smaller than 0. Human pitch values are also limited to a certain range defined by physiological and bio mechanical constraints on our laryngeal system. In adults, values larger than let's say 1000 Hz are very unlikely. We can use this prior knowledge about nature to **regularize** the possible parameter space of our model. If we don't specify this information, the model will assume that 1000 Hz is as likely of a value as 150 Hz. This lack of information can slow down our parameter estimation.

Second, priors can also express our subjective beliefs about our data. For example, knowing what we know about the physiological differences between women and men and the relationship between these physiological differences and pitch, we might be already very certain that female speakers have higher pitch values than male speakers.

But wait a minute. Subjective beliefs? This is science. We are supposed to be objective, right? You are right. We should have a heart of stone and be skeptical about possible relationships in the first place. Practically, this means the following for us: If our research hypothesis is that male voice pitch is lower than female voice pitch, we obviously don't want to specify a prior that assumes the hypothesized relationship before having seen the data. Instead, we should feed our model a **skeptical** prior. A possible skeptical prior could be the assumption that there is *no* relationship between speaker gender and voice pitch (i.e. the average difference is 0). Of course, there might be some variability around this average value (nature is messy after all), so we might want to assume that pitch differences between male and female speakers are normally distributed with an average of 0 and a standard deviation of, say, 50 Hz. Importantly, such a skeptical prior is conservative with regard to our research hypothesis. Our point of departure (before we observed new data) is to assume that there is no effect of gender on pitch. The data need to convince us otherwise.

How does that look in practice? Let's define some priors for our hierarchical model from above. We keep it simple and define a regularizing prior for the intercept (i.e. our reference level) that defines a normal distribution in plausible-pitch-value-land (for convenience we simply pick the mean and sd for our data set). We also define a skeptical prior for all our fixed effects in question (class = b) as normal distributions centered around zero (again, our prior assumption is that there is no difference between conditions).

Defining regularizing priors is essential for more complex models which have to estimate many parameters. Regularizing priors can help our model to converge more quickly.

You can specify every single parameter of your model individually. To see all parameters for which you can specify priors, use the `get_prior()` function of `brms`

```
1 priorFE <- c(
2   # define a regularizing prior for the intercept within
3   # the range of possible pitch values: a normal distribution
4   # with the mean of 194 and a standard deviation of 66
5   prior(normal(194, 66, class = Intercept),
6   # define a skeptical prior for all relevant fixed effects
7   prior(normal(0, 50), class = b)
8 )
```

We add this prior to the model as an argument and calculate the probability for our hypotheses again. We can see that our priors did not affect the posteriors very much.

```
1 modelFE_prior = brm(formula = pitch ~ gender * context,
2   # add prior
3   prior = priorFE,
4   data = politedata,
5   control = list(adapt_delta = 0.99)
6   )

1 > get_posterior_beliefs_about_hypotheses_new(modelFE_prior)
2 # A tibble: 3 x 2
3   hypothesis                                probability
4   <chr>                                     <dbl>
5 1 Female-polite < Female-informal          0.983
6 2 Male-polite < Male-informal              0.917
7 3 Male-informal < Female-polite            1
```

Model criticism

So we know now how to extract relevant comparisons to quantify the evidence surrounding our hypothesis. We have also learned how to specify prior information. The next important step is being able to check if our model really reflects the observed data? A common and easy way to answer this question is to use so-called **posterior predictive checks**. These checks generate new data similar to our posterior distributions and compare them to the observed data. `brms` offers a neat function called `pp_check()`.

To illustrate this point, let's run our model from above without the gender predictor.

```
1 modelFE_noGender= brm(formula = pitch ~ context,
2   prior = priorFE,
3   data = politedata,
4   control = list(adapt_delta = 0.99)
5   )
```

We know that a big chunk of variation is accounted for by gender, so this model should be a poor fit. If we run `pp_check()` on our model (we also specify how many samples we want to compare the observations to), we can see that a model without the `gender` predictor fails to capture an important property of our data: having pitch values from men and women leads to a bimodal distribution of pitch values (i.e. two bumps).

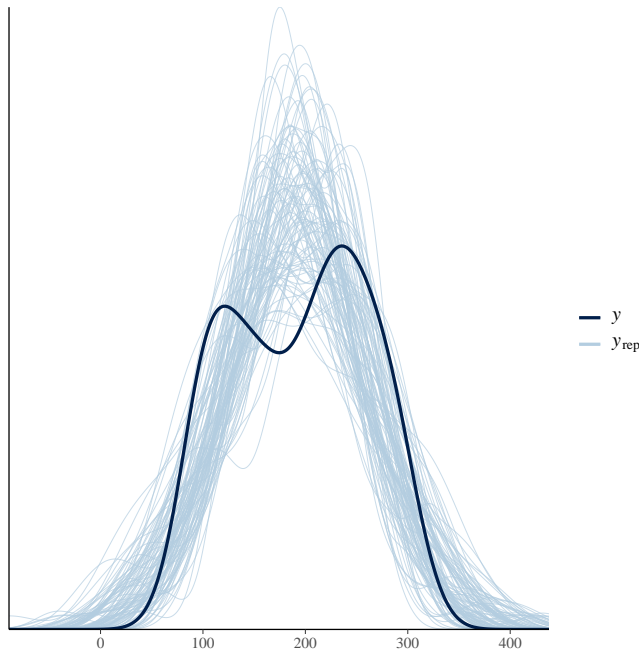


Figure 6: Output of the posterior predictive check for a model **without** the predictor `gender`

```
1 pp_check(modelFE_noGender, nsample = 100)
```

The model overestimates the probability of very low and high pitch values, underestimates the probability of values that surround our two bumps and heavily overestimates the probability of values around 200 Hz. Our earlier model which takes `gender` into account looks quite a bit better. While still showing much uncertainty surrounding the two bumps in the observed distribution, it clearly captures the evidence better.

Adding random effects

In our experiment, we measured pitch multiple times for each subject (since they produced multiple sentences). We also have multiple measures for each sentence (as each sentence was produced by multiple speakers). A crucial assumption of linear regression models is the independence assumptions. Many before us have covered this aspect of linear models (e.g. Winter, 2013; Clark, 1973) so we won't discuss this important issue here in detail. In a nutshell, we need to inform our model about these dependencies between observations. The way we're going to handle this is to add random effects to our model, just as we do in frequentists' frameworks. Random effects are additional parameters that our Bayesian model estimates.

Although the results look quite different, running hierarchical random effect models with `brms` is very similar to the look and feel of non-Bayesian

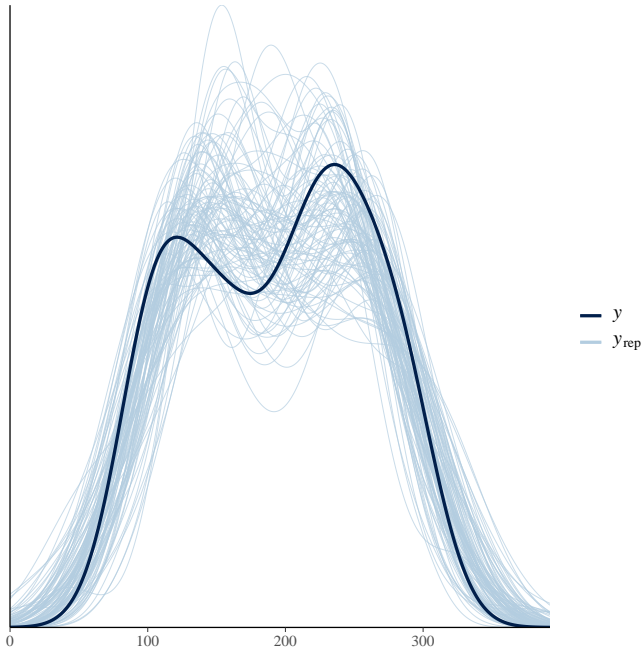


Figure 7: Output of the posterior predictive check for a model with the predictor gender

approaches. Here is the function call to a model with the maximal random effect structure licensed by the design. We also defined regularizing priors for the intercept and skeptical priors for the predictor coefficients. The outcome of this model fit is shown below:

```

1 # define priors
2 prior_MaxRE <- c(
3   # define a regularizing prior for the intercept within the
4     range of possible pitch values
5   prior(normal(194, 66), class = Intercept),
6   # define a skeptical prior for the relevant coefficients
7   prior(normal(0, 50), class = b)
8 )
9 # model
10 model_MaxRE = brm(formula = pitch ~ gender * context +
11   (1 + gender * context | sentence) +
12   (1 + context | subject),
13   prior = prior_MaxRE,
14   data = politedata,
15   control = list(adapt_delta = 0.99)
16 )

```

```

1 Family: gaussian
2 Links: mu = identity; sigma = identity
3 Formula: pitch ~ gender * context + (1 + gender * context | sentence) + (1 + context | subject)
4 Data: politedata (Number of observations: 83)
5 Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
6         total post-warmup samples = 4000
7
8 Group-Level Effects:
9 ~sentence (Number of levels: 7)
10
11      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
12 sd(Intercept)      21.89    10.35    7.07   46.85      1701 1.00
13 sd(genderM)        10.66     8.59    0.39   31.99      2163 1.00
14 sd(contextpol)     15.23    11.51    0.67   42.78      1495 1.00
15 sd(genderM:contextpol) 16.52    12.69    0.72   47.94      1906 1.00
16 cor(Intercept,genderM) -0.23     0.44   -0.89    0.69      3745 1.00
17 cor(Intercept,contextpol) 0.00     0.41   -0.75    0.78      3625 1.00
18 cor(genderM,contextpol) -0.07     0.44   -0.83    0.75      2620 1.00
19 cor(Intercept,genderM:contextpol) -0.10     0.43   -0.84    0.73      4391 1.00
20 cor(genderM,genderM:contextpol) -0.04     0.45   -0.82    0.79      2741 1.00
21 cor(contextpol,genderM:contextpol) -0.14     0.45   -0.87    0.74      2562 1.00
22
23 ~subject (Number of levels: 6)
24
25      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
26 sd(Intercept)      41.86    21.82    16.02  100.10      1295 1.00
27 sd(contextpol)       9.15     9.11     0.30   32.92      2009 1.00
28 cor(Intercept,contextpol) -0.00     0.58   -0.94    0.95      3461 1.00
29
30 Population-Level Effects:
31
32      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
33 Intercept      238.25    26.09   180.41  282.88      1255 1.00
34 genderM        -79.10    34.17  -134.89   -0.44      1188 1.01
35 contextpol     -24.30    12.02   -47.03    0.39      2229 1.00
36 genderM:contextpol  11.13    16.28   -22.13   43.42      1820 1.00
37
38 Family Specific Parameters:
39
40      Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
41 sigma      25.00     2.36    20.83   30.02      3124 1.00
42
43 Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
44 is a crude measure of effective sample size, and Rhat is the potential
45 scale reduction factor on split chains (at convergence, Rhat = 1).

```

The lines 29–34 again give the estimates of the fixed-effects coefficients. The mean estimates look very similar to the ones that we obtained in the fixed-effect only model above. However, not surprisingly, our uncertainty surrounding these estimates is larger. We now also get information about the parameters implied by the specified random effect structure. Lines 9–21 cover the by-sentence random effects, lines 23–27 cover the by-subject random effects.

To check the probability of our hypotheses of interest, we can use the `faintr` package again:

```

1 > get_posterior_beliefs_about_hypotheses_new(model_MaxRE)
2 # A tibble: 3 x 2
3   hypothesis                                probability
4   <chr>                                     <dbl>
5 1 Female-polite < Female-informal          0.974
6 2 Male-polite < Male-informal              0.853
7 3 Male-informal < Female-polite            0.918

```

When we compare these values to the simpler model above, we can see that our evidence for our hypotheses is a little bit weaker for the maximal random effect model. This is not surprising because the simpler model assumed independence where there was none and therefore underestimated the variance. This is particularly noteworthy for H3. Before, we were certain that H3 was true (the posteriors suggested a probability of 1).

Reporting the results

How do we report our analysis and our results in a Bayesian framework? There are as many ways to write up your analysis as there are authors. The following suggestion is one way to do it.

Description of analysis We fitted Bayesian hierarchical linear models to pitch values as a function of `gender`, and `context` and their two-way interaction, using the Stan modeling language (Carpenter et al., 2016) and the package *brms* (Buerkner, 2016). The models included maximal random-effect structures justified by the design, allowing the predictors of interest and their interactions to vary by participants (`context`) and sentences (`GENDER`, `CONTEXT`). We used a regularizing Gaussian prior for the intercept defined centered around the mean of the observation with $\sigma = 66$. We used weakly informative Gaussian priors centered around zero with $\sigma = 50$ for all population-level regression coefficients. Four sampling chains with 2000 iterations each were run for each model, with a warm-up period of 1000 iterations. For all relevant predictor levels and differences between them, we report 95% credible intervals (CIs) and the posterior probability that a parameter β is smaller than zero $P(\beta < 0)$. A 95% credible interval demarcates the range of values that comprise 95% of probability density or mass of our posterior beliefs. We judge there to be *compelling evidence* for an effect if zero is (by a reasonably clear margin) not included in the 95% CI and $P(|\beta| < 0)$ is close to one.

Description of results [tr: here we need the estimates of each level for our convenient data report, right? we havent done that above. Ideas?]

Further reading

e.g. Bayes factor

References

- Buerkner, Paul-Christian (2016). “brms: An R package for Bayesian multi-level models using Stan”. In: *Journal of Statistical Software* 80.1, pp. 1–28.
- Carpenter, Bob et al. (2016). “Stan: A probabilistic programming language”. In: *Journal of Statistical Software* 20, pp. 1–37.
- Clark, Herbert H (1973). “The language-as-fixed-effect fallacy: A critique of language statistics in psychological research”. In: *Journal of verbal learning and verbal behavior* 12.4, pp. 335–359.
- R Core Team (2017). *R: A Language and Environment for Statistical Computing*. Vienna, Austria: R Foundation for Statistical Computing. URL: <https://www.R-project.org/>.
- Sorensen, Tanner, Sven Hohensteinb, and Shravan Vasishth (2016). “Bayesian linear mixed models using Stan: A tutorial for psychologists, linguists, and cognitive scientists”. In: *The Quantitative Methods for Psychology*.
- Winter, Bodo (2013). *Linear models and linear mixed effects models in R with linguistic applications*. URL: <https://arxiv.org/abs/1308.5499>.
- Winter, Bodo and S. Grawunder (2012). “The Phonetic Profile of Korean Formality”. In: *Journal of Phonetics* 40, pp. 808–815.