Exercises week 7

prof. Richard Torkar

March 31, 2021

> Ce que nous connaissons est peu de chose, ce que nous ignorons est immense.
>
> Pierre-Simon Laplace

EXERCISE 1. Now it's time to get our hands wet on a controversial dataset.[1] This dataset and the study that originally published it, stirred a lot of controversy (it's as close to all out war as it can get in academia...)[2] We're gonna have a look at it and see if what the original authors claimed is, well, correct. They wanted to study what the effect of programming languages is on software quality. They claimed the design of a language had a significant effect on software quality.

[1] B. Ray, D. Posnett, V. Filkov, and P. Devanbu. A large scale study of programming languages and code quality in GitHub. In *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 155–165, 2014. DOI: 10.1145/2635868.2635922

[2] Read more about it here: `https://www.hillelwayne.com/post/this-is-how-science-happens/`

First, we need to download the dataset (yes, it's large!) and format it. In order to do that we have an R script, which Berger et al.[3] made and which my colleague Carlo A. Furia[4] adapted. Let's download it, source it, i.e., so that R will be able to use the functionality the script provides, and then use functions from the script to download the tar.gz file, decompress it, and do some cleanup,

[3] E. D. Berger, C. Hollenbeck, P. Maj, O. Vitek, and J. Vitek. On the impact of programming languages on code quality: A reproduction study. *ACM Trans. Program. Lang. Syst.*, 41(4), 2019. DOI: 10.1145/3340571

[4] USI Università della Svizzera Italiana, Switzerland

```
library(tidyverse) #much data wrangling so install this
# the below should be on one line!
url <- "https://raw.githubusercontent.com/torkar/BDA-PL/main/
    docs/utils.R"
# change below if needed
destFile <- "~/Downloads/utils.R"
download.file(url, destFile)
# I have no idea where you put it so change below if needed
source("~/Downloads/utils.R")
setup.data()
d <- load.FSE(cleanup=TRUE)
d <- by.project.language(d)
d <- d[,-c(2:5,7:8)]
d$project <- as.integer(d$project)
dim(d)
```

by executing the last call you can check that we have 1127 rows and 3 columns. The first column is `project`, which is a unique ID for each project (a project may have several rows in the data). Next, `n_bugs` is our outcome (the lower the better). Finally, `language_id` is what we're really interested in.

So, create a number of multilevel models with language and/or project as varying intercepts. Compare all models with LOO. Finally, do we see a significant difference between the 17 languages? Can you infer something from the variability of projects and languages?

Use hyperparameters/hyperpriors.

SOLUTION 1. First, we should think about the ontological and epistemological justifications concerning the underlying data generation process. Our outcome is a count $0 \rightarrow \infty$ (recognize this?) Also the information theoretical justification (epistemological) would be to fall back on the maximum entropy distribution for counts, i.e., low probability and many trials, which is the Poisson. But, look at the mean ($\bar{x} \approx 500$) and the sample variance ($s^2 = 15031006$) and it's obvious we will face challenges down the road. I wish one would at least once get data which can be used for fitting a Poisson model...

No need to whine about it; instead we rely on the Negative-Binomial, which can model the variance separately. Let's create a set of models $M = \mathcal{M}_0, .., \mathcal{M}_n$ and compare them with LOO. Please, look carefully at each model specification below and try to figure out what I've tried to accomplish. Also, the priors might seem tight, but they are actually very broad (please do prior predictive checks and check for yourself)—remember, we use a log link and that makes funky things to the priors.

After the models, further down below, I'll explain the main differences between the models, and what I wanted to achieve. The lines you should pay special attention to end with a hash (#) and a number.

```
set.seed(061215)
m0 <- ulam(
  alist(
    n_bugs ~ dgampois(lambda, phi),
    log(lambda) <- a_l[language_id],
    a_l[language_id] ~ dnorm(a_bar, sigma), #1
    a_bar ~ dnorm(0, 1),
    sigma ~ dexp(1),
    phi ~ dexp(1)
  ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
  log_lik = TRUE, iter = 5e3
)

m1 <- ulam(
  alist(
    n_bugs ~ dgampois(lambda, phi),
    log(lambda) <- a_p[project],
    a_p[project] ~ dnorm(a_bar, sigma), #2
    a_bar ~ dnorm(0, 0.2),
    sigma ~ dexp(1),
    phi ~ dexp(1)
  ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
  log_lik = TRUE, iter = 5e3
)

m2 <- ulam(
  alist(
    n_bugs ~ dgampois(lambda, phi),
    log(lambda) <- a_l[language_id] + a_p[project],
    a_p[project] ~ dnorm(0, sigma_p), #3
    sigma_p ~ dexp(1),
    a_l[language_id] ~ dnorm(0, sigma_l), #4
    sigma_l ~ dexp(1),
    phi ~ dexp(1)
  ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
```

```
  log_lik = TRUE, iter = 5e3
)

m3 <- ulam(
  alist(
    n_bugs ~ dgampois(lambda, phi),
    log(lambda) <- a_l[language_id] + a_p[project],
    a_p[project] ~ dnorm(0, sigma_p),
    sigma_p ~ dexp(1),
    a_l[language_id] ~ dnorm(a_bar, sigma_l), #5
    a_bar ~ dnorm(0, 0.5),
    sigma_l ~ dexp(1),
    phi ~ dexp(1)
  ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
  log_lik = TRUE, iter = 5e3
)

m4 <- ulam(
  alist(
    n_bugs ~ dgampois(lambda, phi),
    log(lambda) <- a_l[language_id] + a_p[project],
    a_p[project] ~ dnorm(a_bar_p, sigma_p), #6
    sigma_p ~ dexp(1),
    a_l[language_id] ~ dnorm(a_bar_l, sigma_l), #7
    a_bar_l ~ dnorm(0, 0.5),
    a_bar_p ~ dnorm(0, 0.2),
    sigma_l ~ dexp(1),
    phi ~ dexp(1)
  ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
  log_lik = TRUE, iter = 1e4
)
```

So, $\mathcal{M}_0$ is an MLM where we model language as a varying effect. If you look at the hash sign you see that we declare a prior, which then consists of other parameters, e.g., `a_bar` is a hyperparameter and we set a hyperprior on the next line line. $\mathcal{M}_1$ is the same but instead we model projects.

Varying effects are also called random effects or group-level effects, I can't say that any of those names are particularly good, but probably exist only to confuse people.

In $\mathcal{M}_2$ we then use both project and language. Here, however, we model the variance, σ_l and σ_p, separately. In short, I have reason to believe that modeling this carefully will tell us something. For $\mathcal{M}_3$ we model the mean of language (using hyperparameters), while in $\mathcal{M}_4$ we model the mean of both language and project (with hyperparameters).

Check the chains, $\widehat{R}$, and effective sample size. Only in $\mathcal{M}_4$ we have reasons to worry. One of the parameters has $\widehat{R} = 1.01$, which is worrisome, but we still have effective sample size in the hundreds so I'll let it slide for now (you know what to do if you have $\widehat{R}$ slightly above 1.01.

```
        mean   sd  5.5% 94.5% n_eff Rhat4
sigma_p 1.08 0.06  0.99  1.17  3805  1.00
a_bar_l 1.40 0.62  0.44  2.42  7379  1.00
a_bar_p 0.22 0.20 -0.11  0.54   351  1.01
sigma_l 3.37 0.81  2.16  4.74  2531  1.00
phi     0.74 0.03  0.68  0.79  7874  1.00
```

Our models now estimate hundreds of parameters. Let's have a look at how many effective parameters we're estimating now,

```
      PSIS     SE dPSIS   dSE pPSIS weight
m3 14149.5 128.77   0.0    NA 316.9   0.62
m2 14150.5 128.26   1.0  4.20 316.1   0.37
m4 14159.0 129.44   9.5  4.25 321.5   0.01
m1 14410.0 138.18 260.5 37.17 367.5   0.00
m0 14580.0 154.15 430.5 68.70  53.0   0.00
```

so the winning model has ≈ 317 effective parameters, but in reality we model

751 parameters. Thanks to our regularizing hyperpriors we avoid overfitting!

Also, you'll see some warnings from LOO about high Pareto k values, but rerunning with pointwise comparison will lead to the same output.

In the output we clearly see the models are divided into two clusters. One where they have a lower PSIS ($\mathcal{M}_3, \mathcal{M}_2, \mathcal{M}_4$) and one which seems to not catch up, i.e., $\mathcal{M}_1$ and $\mathcal{M}_0$. What LOO is saying is that having both project and language might be a good idea (i.e., not models $\mathcal{M}_0$ and $\mathcal{M}_1$). Additionally, $\mathcal{M}_3$ and $\mathcal{M}_2$ are significantly better than $\mathcal{M}_4$.[5] What does this mean?

[5] `9.5 + c(-1,1) * 4.25 * 1.96 = [1.17,17.83]`

Well, after having accounted for modeling the variance using hyperparameters for both language and project ($\mathcal{M}_2$), focusing on modeling the mean for language using hyperparameters ($\mathcal{M}_3$) doesn't give that much more. Also, modeling mean and variance for both language and project using hyperparameters ($\mathcal{M}_4$), doesn't do that much either.

Let's have a look at the variance using `precis(m3)`,

```
        mean   sd 5.5% 94.5% n_eff Rhat4
sigma_p 1.08 0.06 0.99  1.17  2569     1
a_bar   1.29 0.59 0.36  2.24 12383     1
sigma_l 3.65 0.77 2.55  4.98 11899     1
phi     0.74 0.03 0.68  0.79  6720     1
```

so the variance for project is lower (1.08) than for language (3.65). Let's plot projects 18–34 and all our 17 languages,

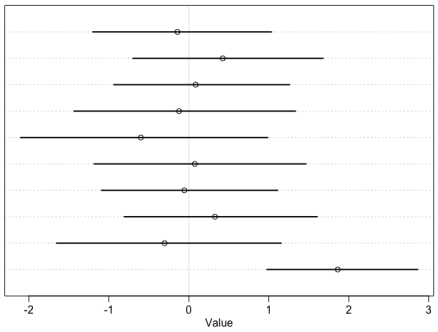

Figure 1: Project estimates

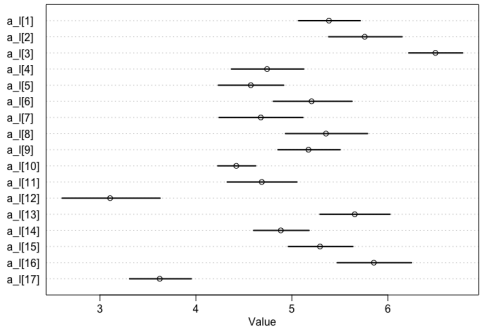

Figure 2: Language estimates

and we see that there's clearly more variability among languages. Even though we do have variability among projects, which we should take into account.

If we examine the language estimates we see that some languages have lower number of bugs than others. So, what's the conclusion, should we use Languages 12 and 17 in projects from now on? Well, perhaps...

But, first we need to think about omitted variable bias; we have more variables in the data that we could take into account. When we do that we risk adding colliders and whatnot. Also, this analysis doesn't say much in itself—we account for variance from two sources, project and language, but that's it.

If you are curious, the answer to the question is, according to me and my co-authors: it depends.[6]

[6] https://arxiv.org/abs/2101.12591