Exercises week 6

prof. Richard Torkar

Oct. 11, 2021

> If fallacious reasoning always led to absurd conclusions, it would be found out at once and corrected. But once an easy, shortcut mode of reasoning has led to a few correct results, almost everybody accepts it; those who try to warn against it are not listened to.
>
> Edwin Thompson Jaynes

EXERCISE 1. For this example we'll make use of a dataset found in the PROMISE repository donated by Prof. Martin Shepperd in 2005, and originally from J. M. Desharnais' master thesis.[1]

[1] http://promise.site.uottawa.ca/SERepository/datasets/desharnais.arff

You can download the file like this,

```
# where is the file?
url <- "https://torkar.github.io/desharnais.csv"
# where should we place the file?
destFile <- "~/Downloads/desharnais.csv"
# d/l file
download.file(url, destFile)
# read the file into d
d <- read.csv("~/Downloads/desharnais.csv")
# remove columns we don't need
d <- d[-c(1)]
str(d) # check format
```

which provides the output below,

```
'data.frame':   77 obs. of  4 variables:
 $ Effort    : int  5152 5635 805 3829 2149 2821 2569 3913 ...
 $ TeamExp   : int  2 1 5 1 1 1 3 2 4 4 ...
 $ ManagerExp: int  5 1 5 1 1 1 2 3 2 5 ...
 $ Language  : int  1 1 1 1 1 1 2 1 1 1 ...
```

We would like to predict `Effort` (our outcome) for implementing a software artifact, given programming language (`Language`), team experience (`TeamExp`), and manager experience (`ManagerExp`), which are our predictors.

What is your implied DAG? Estimate the causal influence (assuming the DAG is correct) of language, by designing one or more models.

EXERCISE 2. Analyzing ordered categorical data (e.g., Likert scale data) is very common. Each year we see numerous master theses at our division that use this type of analysis for analyzing surveys etc. Here we'll use data that was collected by a researcher at our division (Dr. Richard Berntsson Svensson).[2]

[2] https://github.com/torkar/feature-selection-RBS

Start by downloading and cleaning the data,

```
url <- "https://github.com/torkar/feature-selection-RBS/blob/
    master/data/data.rds?raw=true"
```

```
destFile <- "~/Downloads/data.rds"
download.file(url, destFile)
f <- readRDS(destFile)
f <- f[-c(1,3:14,16:18)] # rm columns not needed

d <- list(
  State = as.integer(f$State),
  b_val = as.integer(f$b_val),
  alpha = rep(2,3) # 3 twos since b_val are four categories
)
```

and now `d` contains three variables.

`State`, our outcome, is an ordered categorical variable that indicates how far a requirement has come in the requirements prioritization process. There are six states a requirement can be in and they are numbered accordingly,

1. Elicited, Dropped
2. Elicited, Prio, Dropped
3. Elicited, Prio, Planned, Dropped
4. Elicited, Prio, Planned, Implemented, Dropped
5. Elicited, Prio, Planned, Implemented, Tested, Dropped
6. Elicited, Prio, Planned, Implemented, Tested, Released

Next, we have `b_val` (business value), a predictor, which is also ordered categorical. This is a value the teams set on a requirement indicating how valuable the requirement is from a business perspective. `b_val` is $1, .., 4$, where higher is more valuable. Finally, we set a convenience variable called `alpha`. This is the value we set for our Dirichlet prior. Remember, we had $K = 4$ categories in `b_val`, thus we have $K - 1$ priors we need to set, i.e., 3 of them.[3]

[3] Check pp. 392–393 in the book.

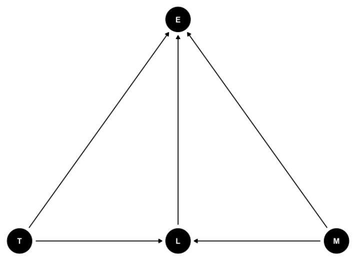
First, design a model, $\mathcal{M}_0$, where we only predict the grand mean for our outcome `State` (we'll actually predict the borders between the six categories in our outcome).

Next, design a model, $\mathcal{M}_1$, where you also add the ordered categorical predictor `b_val`. Is it significant?

Chapters 12.3 and 12.4 are your friends...

Finally, compare the two models using LOO. Does having the predictor help significantly in improving out of sample predictions?

SOLUTION 1. Below we see our implied DAG,



which we've plotted by doing this,

```
dag <- dagitty("dag{
    E <- L
    E <- T -> L
    E <- M -> L
}")
coordinates(dag) <- list( x=c(E=1,L=1,T=0,M=2) ,
    y=c(E=0,L=-1,T=-1,M=-1) )
drawdag(dag)
```

We assume that there is a direct effect of `Language` on `Effort`. We also believe that there's a direct effect of `TeamExp` and `ManagerExp` on `Effort`, but there are also direct effects going to `Language`, from these two entities. What to do?

Well, `impliedConditionalIndependencies(dag)` tell us that $M \perp\!\!\!\perp T$, i.e., `ManagerExp` is conditionally independent from `TeamExp` (which is something one can see directly from the plot).

Using `adjustmentSets(dag, exposure="L", outcome="E")` gives this output: `{ M, T }`. In short, conditioning on M **or** T would suffice.

Assuming our DAG is correct!

So, let's create two models, one where we condition on L and one where we condition on L and T. Our outcome, `Effort`, is a count going from $0 \to \infty$. Hence, a max_ent distribution in this case would be the Poisson. But, recall, a Poisson has one parameter to estimate, i.e., λ, and it implies that the mean and the variance should be equal. In our case (check yourself!) it is definitely not equal.

Please, fit a model using Poisson and you will see the sampler will get into problems after you start adding predictors.

We need to fall back on modeling the variance as a separate component. For this we have Negative-Binomial, or as McElreath calls it Gamma-Poisson (see pp. 373).

```
m0 <- ulam(
    alist(
        Effort ~ dgampois(lambda, phi),
        log(lambda) <- a[Language],
        a[Language] ~ dnorm(0,0.5),
        phi ~ dexp(1)
    ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
        iter = 5e3
)

m1 <- ulam(
    alist(
        Effort ~ dgampois(lambda,phi),
        log(lambda) <- a_l[Language] + a_t[TeamExp],
        a_l[Language] ~ dnorm(0,0.5),
        a_t[TeamExp] ~ dnorm(0,0.5),
        phi ~ dexp(1)
    ), data = d, cores = 4, chains = 4, cmdstan = TRUE,
        iter = 5e3
)
```

As you see I've started using `cmdstan=TRUE` to use the new and improved sampler. Also, I've asked the sampler for 5000 iterations, in order for the chains to converge. This is not uncommon when using Negative-Binomial.

Now check the output using `precis()` and you will see how the effects of language changes depending on if and which predictors we add. According to our DAG, if it is 'true', we should rely on $\mathcal{M}_1$. With $\mathcal{M}_0$ we will presumably wrongly estimate the causal effect.

While we're at it, let's see how the differences between the three languages differ between the two models,

Please check `precis()` to see how they differ!
Thanks to Fredrik Ullman (MSc SE, 2021) for pointing out an error in the original code below

```
post0 <- extract.samples(m0)
post0$diff12 <- post0$a[,1] - post0$a[,2]
post0$diff13 <- post0$a[,1] - post0$a[,3]
post0$diff23 <- post0$a[,2] - post0$a[,3]

post1 <- extract.samples(m1)
post1$diff12 <- post1$a_l[,1] - post1$a_l[,2]
post1$diff13 <- post1$a_l[,1] - post1$a_l[,3]
post1$diff23 <- post1$a_l[,2] - post1$a_l[,3]

plot(precis(post0),pars = c("diff12","diff13","diff23"))
plot(precis(post1),pars = c("diff12","diff13","diff23"))
```

Note the different horizontal axes.

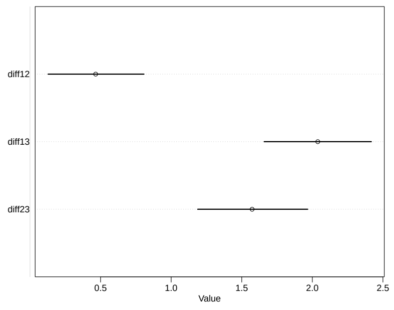


Figure 1: $\mathcal{M}_0$: From top to bottom, the three languages are compared: L1 v. L2, L1 v. L3, and L2 v. L3.

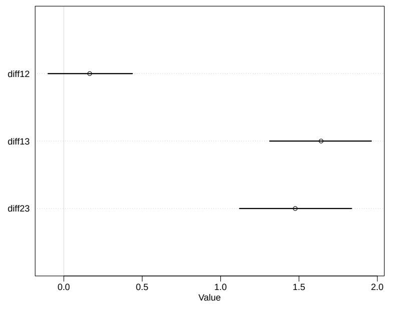


Figure 2: $\mathcal{M}_1$: From top to bottom, the three languages are compared: L1 v. L2, L1 v. L3, and L2 v. L3.

SOLUTION 2. First, let's sample and check the estimates for the first model where we only model the intercepts, i.e., κ. Notice that I use `cmdstan = TRUE` and `log_lik = TRUE`.[4]

κ is the Greek letter kappa.

[4] `log_lik = TRUE` ensures that we also have the log likelihood of each observation stored. It's needed for model comparisons.

These models will give some initial warnings during warmup but there's no need to worry.

```
m0 <- ulam(
  alist(
    State ~ dordlogit(0, kappa),
    kappa ~ dnorm(0, 1.5)
  ), data = d, chains = 4, cores = 4, cmdstan = TRUE,
  log_lik = TRUE
)
precis(m0, depth=2)
```

which gives us the following output,

```
          mean   sd  5.5% 94.5% n_eff Rhat4
kappa[1] -0.97 0.02 -1.01 -0.94  1740     1
kappa[2] -0.09 0.02 -0.12 -0.06  2696     1
kappa[3]  0.61 0.02  0.58  0.65  3082     1
kappa[4]  0.85 0.02  0.82  0.88  3134     1
kappa[5]  0.89 0.02  0.86  0.92  2908     1
```

indicating that our chains have converged and that we have good sample sizes (check the traceplots while you're at it :)

But what is `kappa[i]`? Well, `kappa[1]` is the first border, between Category 1 and 2. We can convert them to probability (remember they are on the logit scale!) like this,

```
inv_logit(coef(m0))
```

which spits out this,

```
 kappa[1]  kappa[2]  kappa[3]  kappa[4]  kappa[5]
0.2742288 0.4777283 0.6486877 0.7005881 0.7085920
```

so $\approx$ 27% of the probability mass is below the first border, and $>$ 70% is below the fifth border (incl. Category 1, .., 5). If we want to calculate how much probability mass Category 6 has then we simply do $1 - 0.709 = 0.291$.

Next, let's specify a model where we add the ordered categorical predictor,

```
m1 <- ulam(
  alist(
    State ~ dordlogit(phi, kappa),
    phi <- bB * sum( delta_j[1:b_val]),
    kappa ~ dnorm(0, 1.5),
    bB ~ dnorm(0,1),
    vector[4]: delta_j <<- append_row( 0, delta),
    simplex[3]: delta ~ dirichlet(alpha)
  ), data = d, chains = 4, cores = 4, cmdstan = TRUE,
  log_lik = TRUE
)
```

Please see pp. 394 for an explanation of the notation. I know it looks fishy but it's important you understand it.

and if we look at the estimates of interest we see this,

`precis(m1, depth=2, omit="kappa")`

```
         mean   sd 5.5% 94.5% n_eff Rhat4
bB       0.12 0.05 0.04  0.20  1753     1
delta[1] 0.39 0.18 0.11  0.70  1575     1
delta[2] 0.37 0.18 0.10  0.68  1816     1
delta[3] 0.24 0.14 0.05  0.51  2437     1
```

The overall association of business level β_B is positive. But what about our four categories (i.e. three borders)? Well, as McElreath writes (pp. 395), the easiest way is to use `pairs()`, but already by looking at the output above we see something. We have three borders and if they would have exactly the same probability mass then they each would have $\frac{1}{3}$, we clearly see that is not the case, Category 1: $> \frac{1}{3}$, Category 2: $> \frac{1}{3}$, and finally Category 3: $\approx \frac{1}{4}$.

Next, let's see what LOO has to say,

```
compare(m0,m1,func=LOO)
```

```
      PSIS     SE dPSIS  dSE pPSIS weight
m1 34045.4 104.13   0.0   NA   6.3   0.78
m0 34048.0 104.16   2.6 4.51   4.9   0.22
```

hmm... so, $\mathcal{M}_1$ takes the lead (78% of the weight). However, it's not a clear lead. If we calculate a 95% confidence interval,

```
# dPSIS + c(-1,1) * dSE * 95% z value
2.6 + c(-1,1) * 4.51 * 1.96
```

it's clearly crossing zero, i.e., $\text{CI}_{95\%} = [-6.2396, 11.4396]$.

To conclude, adding our *predictor* business value to the model had an effect, but it is not statistically significant on the 95%-level according to LOO. Does it matter? No, in this case we wanted to know if the *parameter* was significant, which it actually is. Both conclusions are correct :)