GR5065 Homework 5

Ben Goodrich

Due March 30, 2021 at 8PM New York Time

1 Social and political correlates of Covid-19

```
library(readr)
library(dplyr)
df <- suppressWarnings(read_csv("https://wzb-ipi.github.io/corona/df_full.csv",
                                col_types = cols(X1 = col_skip(),
                                date = col_date(format = "%Y-%m-%d"),
                                infections_ebola = col_integer()))) %>%
  mutate(checks_veto = ifelse(checks_veto < 0, NA, checks_veto),
         pop_tot = round(pop_tot * 10^6))
data_date <- max(df$date_rep, na.rm = TRUE)
df_today  <- df %>% filter(as.Date(date_rep) == data_date)
```

```
measures <- read_csv("https://raw.githubusercontent.com/wzb-ipi/rep_corona/master/measures.csv") %>%
  filter(include == 1)

families <- c("state_cap_vars", "pol_account_vars", "social_vars",
              "econ_vars",  "phys_vars", "epi_vars", "health_sys_vars")

for(v in families){
  assign(v, filter(measures, family == v)$vars %>% as.character)
  assign(paste0(v, "_labels"), filter(measures, family == v)$labels %>% as.character)
 }

controls <- c("pop_tot_log", "share_older", "healthcare_qual", "health_exp_pc", "detect_index")
controls_labels <- c("Total population (logged)", "Share 65+", "Healthcare quality index (GHSI)",
                     "Healthcare spending/capita", "Health data quality")
```

1.1 Frequentist Inference

Papers that use Frequentist estimation techniques often do not state why they use Frequentist estimation techniques. The paper does not seem to mention any Frequentist goal, but the fact that they use LASSO a lot suggests that the authors are primarily interested in predictions, rather than traditional Frequentist inference. Nor does the paper test any null hypothesis.

1.2 Clustered Standard Errors

Like any Frequentist statement, these robust standard errors are estimates of the standard deviation of $\hat{\beta}$ *across datasets* of size N that have been randomly sampled from some well-defined population. As is typically the case in comparative politics and many other situations where the unit of analysis is an institution, it is nonsensical to refer to datasets in the plural because the only dataset (singular) of countries is the roughly

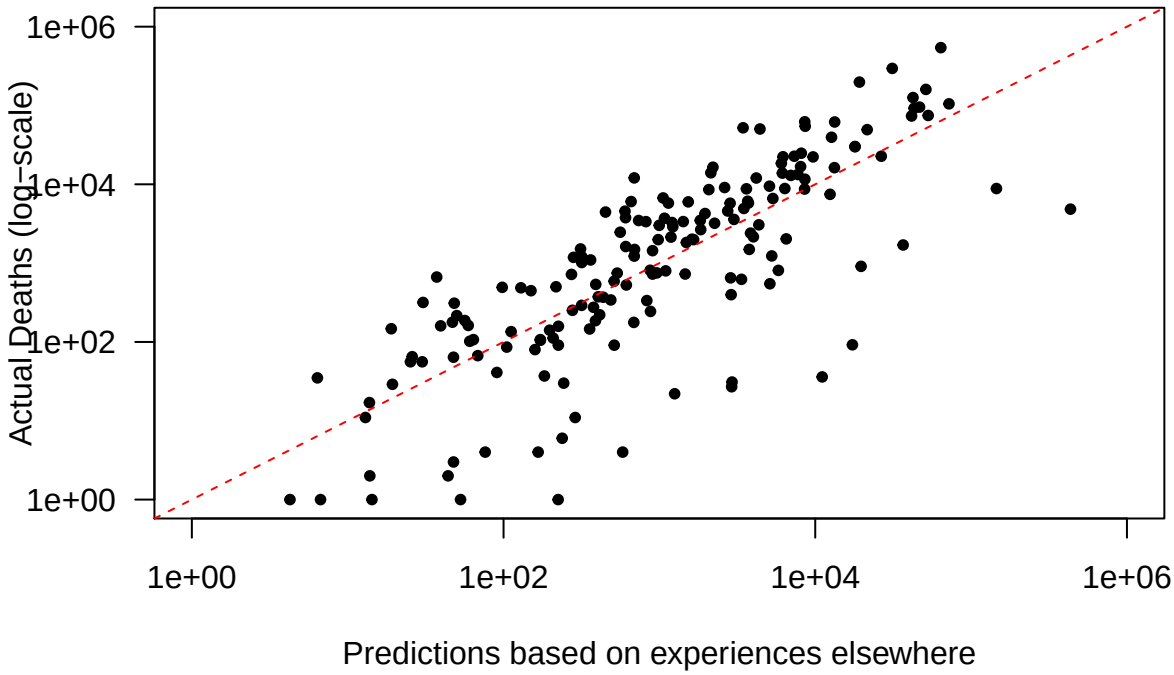
200 that currently exist in that population. But whether a country makes it into the model depends solely on whether it has observed data on all the variables, which is not a random sampling mechanism.

Robust standard errors do make it possible to consistently estimate the standard deviation of $\widehat{\beta}$ *across datasets* of size N that have been randomly sampled from some well-defined population when the true distribution of the n-th error is $\epsilon_n \sim \mathcal{N}(0, \sigma_n)$, even though OLS wrongly assumes that $\epsilon_n \sim \mathcal{N}(0, \sigma)\, \forall n$. However, obtaining a good estimate of the standard deviation of $\widehat{\beta}$ *across datasets* of size N that have been randomly sampled from some well-defined population does not magically make the estimated standard deviation of $\widehat{\beta}$ into the posterior standard deviation of $\widehat{\beta}$ *conditional on the one dataset of size N that you have*, especially not that of a posterior distribution that assumes $\epsilon_n \sim \mathcal{N}(0, \sigma_n)\, \forall n$ and estimates each σ_n rather than a common σ. The only way to obtain a posterior standard deviation of $\widehat{\beta}$ is to use an actual Bayesian estimator but even that will be conditional on the assumption about how ϵ_n was generated.

Finally, these particular robust standard errors assume each ϵ_n is independent from all of the others, although there is a similar standard error estimator that is consistent when the observations are clustered. One of the difficulties with compartive politics and pandemics is that it is fairly implausible to assume that countries are conditionally independent given the predictors (in the model). For example, if Germany has more covid deaths than is expected by the model, it is fairly likely that Austria will as well after adjusting for population differences. Thus, these particular robust standard errors presumably would not due a very good job of estimating the standard deviation of $\widehat{\beta}$ *across datasets* of size N that have been randomly sampled from some well-defined population because those N countries are not actually conditionally independent of each other. In fairness, `stan_glm` also makes the assumption that the observations are conditionally independent, but it would be fairly easy to estimate a Bayesian model that allows them to be dependent but you have to express your prior beliefs about the error dependence.

1.3 Leave-One-Out Cross-Validation

```
ols <- lm(deaths_cum_log ~ pop_tot_log + share_older + healthcare_qual +
            health_exp_pc + detect_index, data = df_today)
dataset <- model.frame(ols) # a subset of 174 observations from df_today used by lm()
predictions <- sapply(1:nrow(dataset), FUN = function(n) {
  ols_ <- update(ols, data = dataset[-n, ])
  deaths_ <- exp(predict(ols, newdata = dataset[n, ]))
  return(deaths_)
})
plot(x = predictions, y = exp(dataset$deaths_cum_log), pch = 20, log = "xy", las = 1,
     xlab = "Predictions based on experiences elsewhere",
     ylab = "Actual Deaths (log-scale)", xlim = c(1, 10^6), ylim = c(1, 10^6))
abline(a = 0, b = 1, col = 2, lty = 2)
```



The `loo` function *estimates* the predictive performance — measured by Expected Log Predictive Density (ELPD) — for what would happen if you did literal leave-one-one cross-validation of a Bayesian model by making an additional assumption that one observation could be left out without having a large effect on the posterior distribution. When that assumption is violated (as it was below) then the ELPD estimator is not valid, although the posterior distribution is (for those N observations at least). By specifying `k_threshold`, you can do literal leave-one-one cross-validation for observations whose Pareto k is above that and approximate leave-one-one cross-validation for observations whose Pareto k is below that.

1.4 Posterior Distributions

The prior on the intercept is usually the easiest because it can be interpreted as the expected value of η for a country with average values on all of its predictors. This is conceptually different from but often not that different numerically from the expected value of η marginally. But if the best you can do is express your beliefs about the expected value of η irrespective of the predictors, that is only a bit weaker than the conceptually correct conditional prior.

The overall mortality rate from covid is very low, although it is higher for older people. It is more common to hear about the "case fatality rate", which is the proportion of people diagnosed with covid who die or the "infection fatality rate", which is the proportion of of people infected with covid (regardless of whether they are diagnosed) who die If one in ten thousand people in a country die from covid, the value of α in a logit model would be

```
qlogis(10^-4)
```

```
## [1] -9.21024
```

whereas if it were one in a hundred thousand, we could calculate

```
qlogis(10^-5)
```

```
## [1] -11.51292
```

So, I will go with a normal prior on α with an expectation of -11 and a standard deviation of 3

```
library(rstanarm)
options(mc.cores = parallel::detectCores())
```

```
prior_alpha <- normal(location = -11, scale = 3)
```

```
prior_beta  <- normal(location = c("share_older" = 0.2,
                                   "healthcare_qual" = -0.2,
                                   "health_exp_pc" = -0.1,
                                   "detect_index" = 0.1,

                                   "vdem_libdem" = -0.1,
                                   "pr" = -0.1,
                                   "vdem_mecorrupt" = -0.1,
                                   "oil" = 0.1,
                                   "electoral_pop" = 0.1,
                                   "woman_leader" = -0.1,
                                   "polar_rile" = 0.1,
                                   "pos_gov_lr" = 0.1,

                                   "trust_gov" = -0.1,
                                   "al_etfra" = 0,
                                   "al_religfra" = 0,
                                   "gini" = 0.1,
                                   "trust_people" = -0.1,
                                   "migration_share" = 0.1,
                                   "share_powerless" = 0.1),
                      scale = c("share_older" = 0.25,
                                "healthcare_qual" = 0.25,
                                "health_exp_pc" = 0.25,
                                "detect_index" = 0.25,

                                "vdem_libdem" = 0.25,
                                "pr" = 0.25,
                                "vdem_mecorrupt" = 0.25,
                                "oil" = 0.5,
                                "electoral_pop" = 0.25,
                                "woman_leader" = 0.25,
                                "polar_rile" = 0.5,
                                "pos_gov_lr" = 0.25,

                                "trust_gov" = 0.25,
                                "al_etfra" = 0.25,
                                "al_religfra" = 0.25,
                                "gini" = 0.25,
                                "trust_people" = 0.25,
                                "migration_share" = 0.25,
                                "share_powerless" = 0.25))
```

In order for these priors to make sense, many of the predictors need to have their units changed into something more reasonable (percentages to proportions, dollars to thousands of dollars, etc.)

```
df_today$share_older <- df_today$share_older / 100
df_today$healthcare_qual <- df_today$healthcare_qual / 100
df_today$health_exp_pc <- df_today$health_exp_pc / 1000
df_today$detect_index <- df_today$detect_index / 100

df_today$trust_gov <- df_today$trust_gov / 100
```

```
df_today$trust_people <- df_today$trust_people / 100
df_today$migration_share <- df_today$migration_share / 100
```

Alas, `stan_glm` yields convergence problems, divergent transitions, etc. with these predictors when the priors are independent. We will talk more in future weeks about what the `QR` flag does, but in short, it estimates a model with predictors that have been transformed to be orthogonal and have a common scale, and then applies the inverse transformation to the coefficients at the end in order to yield coefficients for the original correlated and poorly-scaled predictors. That helps to overcome sampling problems and tends to yield a higher effective sample size, but it makes it impossible to specify a substantive prior on the coefficients of the original correlated and poorly-scaled predictors. Thus, we can just go with a generic, weakly-informative prior on the transformed coefficients:

```
prior_generic <- normal(location = 0, scale = 5)
```

The model that includes both political accountability and social variables has the fewest observations because more observations get dropped due to missingness. We estimate it first in order to re-use its dataset in future models.

```
both <- stan_glm(cbind(deaths_cum, pop_tot - deaths_cum) ~
                   share_older + healthcare_qual + health_exp_pc + detect_index +
                   vdem_libdem + pr + vdem_mecorrpt + oil + electoral_pop +
                   woman_leader + polar_rile + pos_gov_lr +
                   trust_gov + al_etfra + al_religfra + gini + trust_people +
                   migration_share + share_powerless,
                 data = df_today, family = binomial(link = "logit"),
                 prior_intercept = prior_alpha, prior = prior_generic, QR = TRUE, seed = 12345)
```

We can obtain the dataset used by the previous model by calling

```
dataset <- model.frame(both)
colnames(dataset)[1] <- "y"
```

and then pass `dataset` (renaming the outcome to `y`) to `stan_glm` in order to avoid later errors when comparing models:

```
controls_only <- stan_glm(y ~ share_older + healthcare_qual + health_exp_pc + detect_index,
                          data = dataset, family = binomial(link = "logit"),
                          prior_intercept = prior_alpha, prior = prior_generic,
                          QR = TRUE, seed = 12345)
```

```
pol_account <- update(controls_only, formula. = y ~
                        share_older + healthcare_qual + health_exp_pc + detect_index +
                        vdem_libdem + pr + vdem_mecorrpt + oil + electoral_pop +
                        woman_leader + polar_rile + pos_gov_lr)
```

```
social <- update(controls_only, formula. = y ~
                   share_older + healthcare_qual + health_exp_pc + detect_index +
                   trust_gov + al_etfra + al_religfra + gini + trust_people +
                   migration_share + share_powerless)
```

1.5 Model Comparison

If you imagine what would happen to the posterior distribution if you were to drop one *country* that is — in binomial models — tantamount to dropping millions of people from that country and thus it is no surprise that doing so violates the assumption of the PSISLOOCV estimator of the ELPD that any observation can be dropped without having a big effect on the posterior distribution. If we had individual level data, we could overcome this problem by using a Bernoulli likelihood rather than a binomial, but we do not in this

case. Thus, we can specify the `k_threshold` parameter so that `loo` does literal leave-one-out cross-validation in the (many) cases where this assumption is violated.

```
loo_controls_only <- loo(controls_only, k_threshold = 0.7, save_psis = TRUE)
loo_pol_account <- loo(pol_account, k_threshold = 0.7, save_psis = TRUE)
loo_social <- loo(social, k_threshold = 0.7, save_psis = TRUE)
loo_both <- loo(both, k_threshold = 0.7, save_psis = TRUE)
```

The estimated standard errors of the difference in ELPD are huge and probably not estimated very well anyway but they do suggest that the social predictors should be included.

```
loo_compare(loo_controls_only, loo_pol_account, loo_social, loo_both)
```

```
##               elpd_diff se_diff
## social              0.0       0.0
## controls_only  -88972.9  230897.5
## both          -160394.9  366641.2
## pol_account   -266549.9  440467.6
```

Alternatively, we could do K-fold cross-validation like the message suggests

```
kfolds <- list(controls_only = kfold(controls_only, K = 10),
               pol_account   = kfold(pol_account,   K = 10),
               social        = kfold(social,        K = 10),
               both          = kfold(both,          K = 10))
kfolds
```

If we calculate weights on the predictions of the models using stacking, there would be an error message saying it is unable to start the maximization process. Thus, we can simplify (and worsen) the calculation by using the "pseudo Bayesian model averaging" estimator discussed in the reading.

```
loo_model_weights(list(controls_only = loo_controls_only,
                       pol_account = loo_pol_account,
                       social = loo_social,
                       both = loo_both), method = "pseudobma")
```

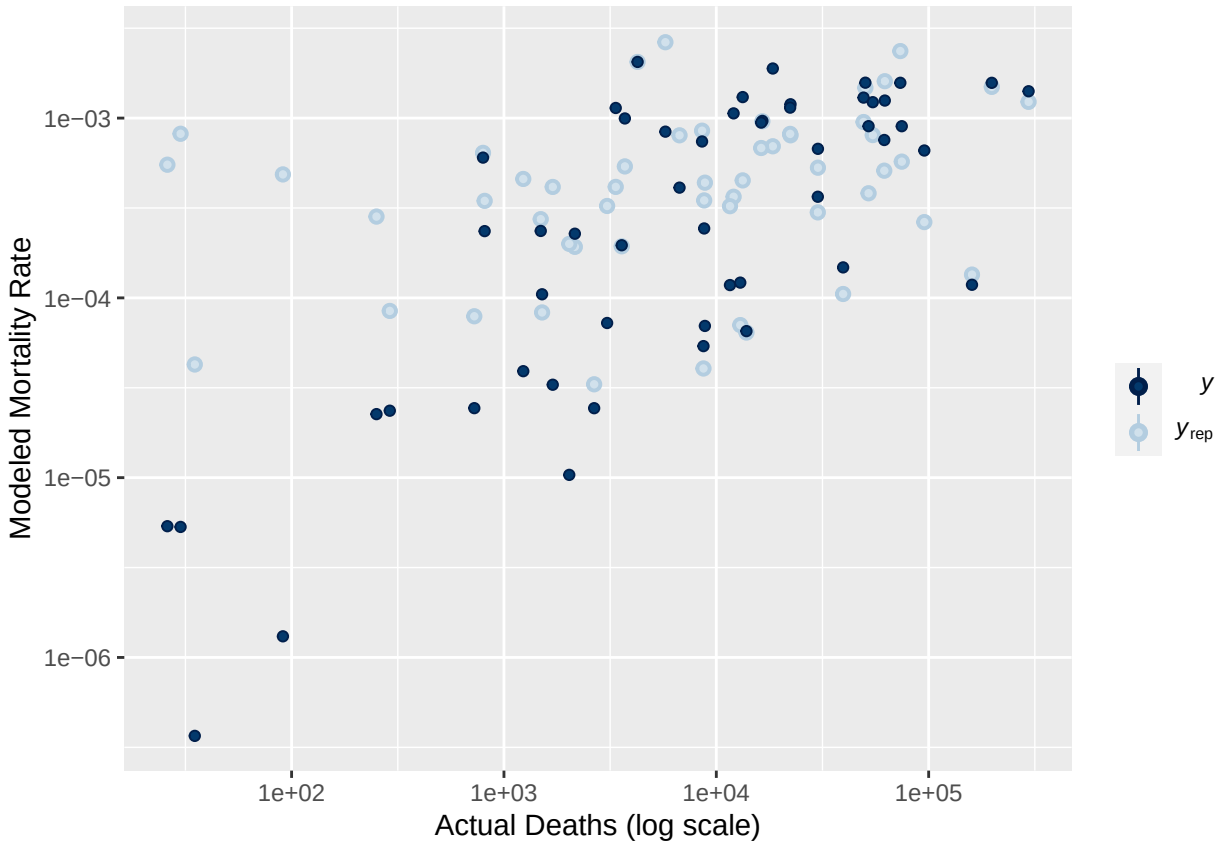
```
## Method: pseudo-BMA+ with Bayesian bootstrap
## ------
##               weight
## controls_only 0.203
## pol_account   0.085
## social        0.487
## both          0.225
```

Thus, a plurality of the weight is put on the model with social and control variables only. However, all four models get positive weight and the weighted predictions would be better than those of any individual model.

1.6 Overfitting

Although there are many predictors and not many observations, Bayesian models tend to avoid severe overfitting simply by averaging the fit measure over the entire posterior distribution, rather than evaluating the fit only at the mode. And in this case, the main problem is *underfitting*.

```
library(ggplot2)
pp_check(social, x = dataset$y[ , 1], plotfun = "intervals", size = 0.5) +
  scale_x_continuous(trans = 'log10') + scale_y_continuous(trans = 'log10') +
  xlab("Actual Deaths (log scale)") + ylab("Modeled Mortality Rate")
```



The model produces very precise predictions (the 90% intervals are less than the width of the grey dots) that are very wrong for a lot of countries, particularly those that have very few actual (reported) deaths. That could indicate under-reporting of deaths in the data or it could be that this model is not sophisticated or rich enough to capture what makes some countries (New Zealand, Vietnam, etc.) so effective at containing the covid pandemic.

2 Count Models

```
youtube <- read_csv("https://osf.io/25sz9/download")
```

2.1 Log-Likelihood

We can evaluate the log-likelihood of proposed parameters that have been drawn from the prior

```
scol <- youtube$scol - mean(youtube$scol)          # centered and in raw units
age2 <- (youtube$age2 - mean(youtube$age2)) / 365 # centered and in years

alpha_  <- rnorm(1, mean = log(10^3), sd = 2)
beta_1_ <- rnorm(1, mean = 0.05, sd = 0.10)
beta_2_ <- rnorm(1, mean = 0.01, sd = 0.05)

eta_ <- alpha_ + beta_1_ * scol + beta_2_ * age2
mu_  <- exp(eta_)

phi_ <- rexp(1, rate = 10^-4)

sum(dnbinom(youtube$views2, size = phi_, mu = mu_, log = TRUE))
```

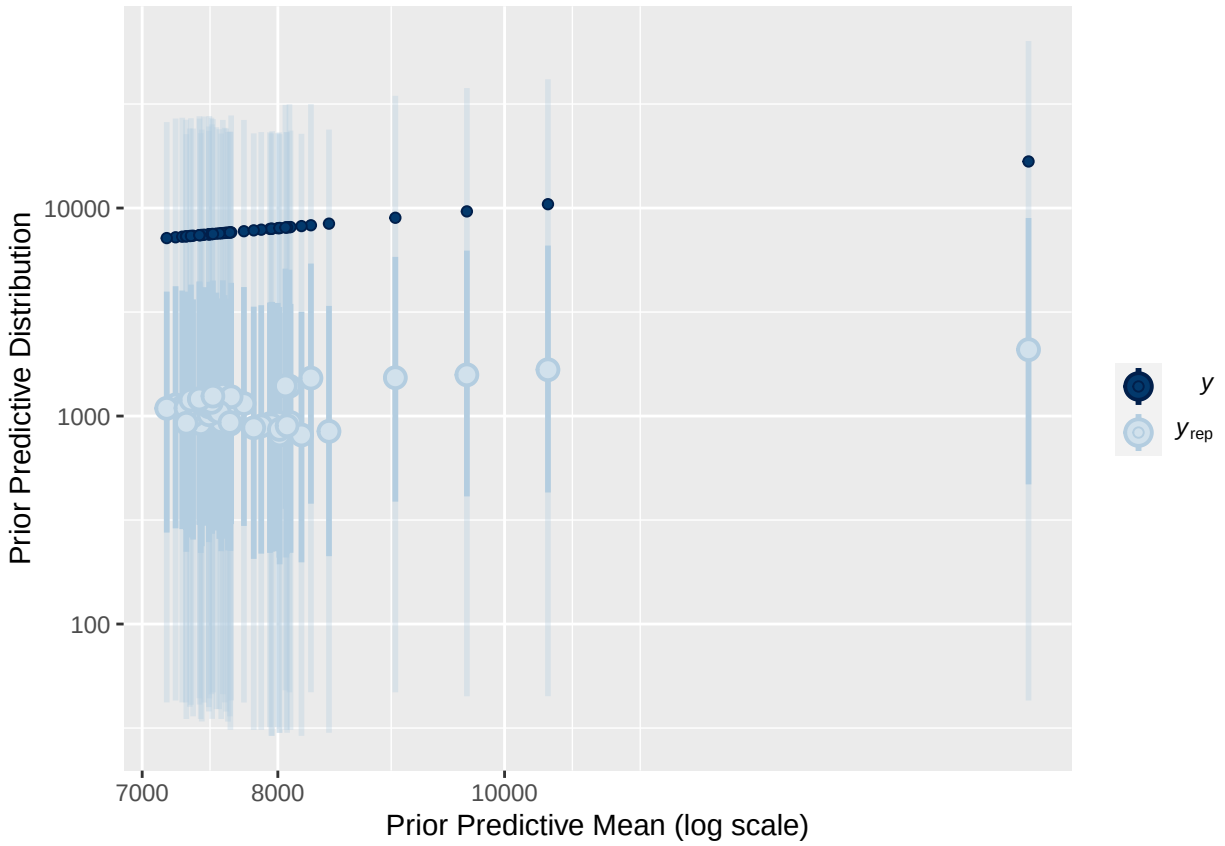
```
## [1] -9510972
```

This number has no meaning in absolute terms but would be useful to compare with the log-likelihood of other values of the parameters. Indeed, maximum likelihood chooses the parameter values in order to maximize this quantity.

2.2 Prior Predictive Distribution

We can use the same priors for the parameters and then take the extra step of drawing (multiplicative) errors from a Gamma distribution that scale the expectation of the Poisson distribution that generates the observed outcomes:

```
y_ <- t(replicate(1000, {
  alpha_  <- rnorm(1, mean = log(10^3), sd = 2)
  beta_1_ <- rnorm(1, mean = 0.05, sd = 0.10)
  beta_2_ <- rnorm(1, mean = 0.01, sd = 0.05)

  eta_ <- alpha_ + beta_1_ * scol + beta_2_ * age2
  mu_  <- exp(eta_)

  phi_ <- rexp(1, rate = 10^-4)

  epsilon_ <- rgamma(n = length(mu_), shape = phi_, rate = phi_)

  rpois(n = length(mu_), lambda = epsilon_ * mu_)
}))

bayesplot::ppc_intervals(colMeans(y_), y_, x = colMeans(y_)) +
  scale_x_continuous(trans = 'log10') + scale_y_continuous(trans = 'log10') +
  xlab("Prior Predictive Mean (log scale)") + ylab("Prior Predictive Distribution")
```



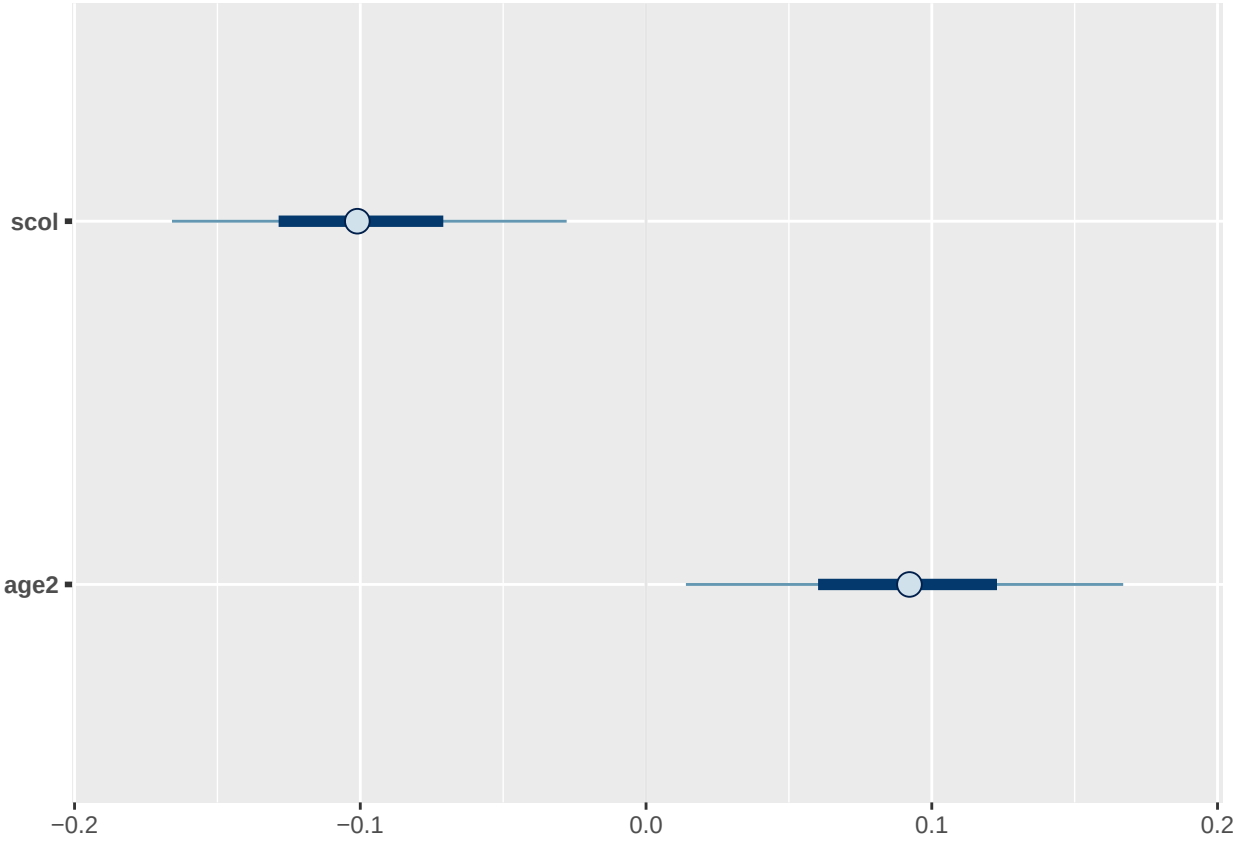
As can be seen, we are predicting about 1000 views per video, the prior predictive distribution is skewed (because the means are to the right of the medians) but not all that much (because they are about the 70-th percentile), and there is only a small probability on very large values. In particular, the proportion of predictions greater than a half million is only 0.002.

2.3 Posterior Distribution

We first need to rescale the `age2` variable to be in years and then can draw from the posterior distribution:

```
youtube$age2 <- youtube$age2 / 365
post <- stan_glm.nb(views2 ~ scol + age2, data = youtube,
                    prior_intercept = normal(location = log(10^3), scale = 2),
                    prior = normal(location = c(0.05, 0.01), scale = c(0.1, 0.05)),
                    prior_aux = exponential(rate = 10^-4))
```

```
plot(post, pars = c("scol", "age2"))
```



The posterior distribution of the coefficient on `scol` is concentrated near −0.1, in contrast to the prior and the hope that higher quality videos would accumulate more views.

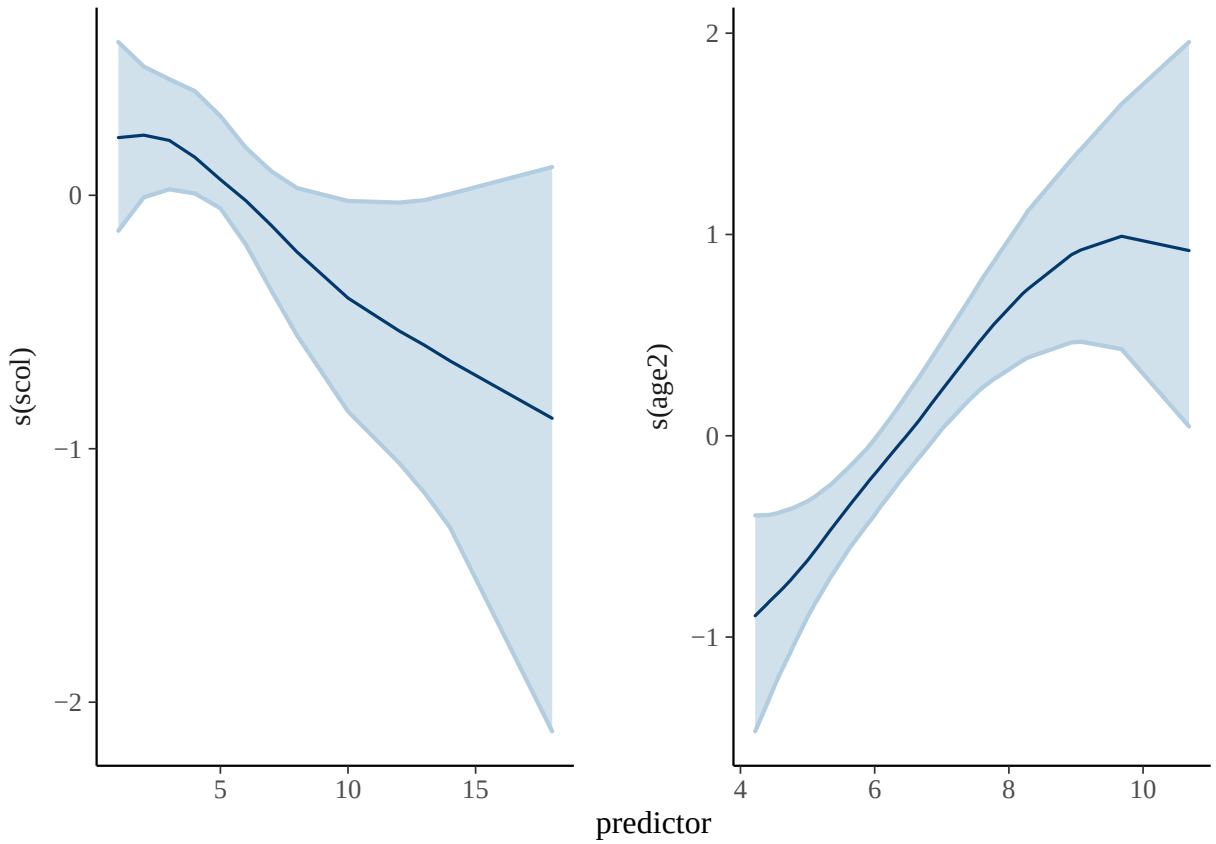
2.4 Splines

The main choice is whether to estimate univariate or bivariate splines. Bivariate splines allow for interactions between the two variables when producting the smooth non-linear function. In this case, it is hard to think of a strong reason for interactions between video quality and age, but we can try it both ways:

```
post_univariate <- stan_gamm4(views2 ~ s(scol) + s(age2),
                              data = youtube, family = neg_binomial_2,
                              prior_intercept = normal(location = log(10^3), scale = 2),
                              prior_aux = exponential(rate = 10^-4), seed = 12345)
post_bivariate <- stan_gamm4(views2 ~ s(scol, age2),
                             data = youtube, family = neg_binomial_2,
                             prior_intercept = normal(location = log(10^3), scale = 2),
                             prior_aux = exponential(rate = 10^-4), seed = 12345)
```

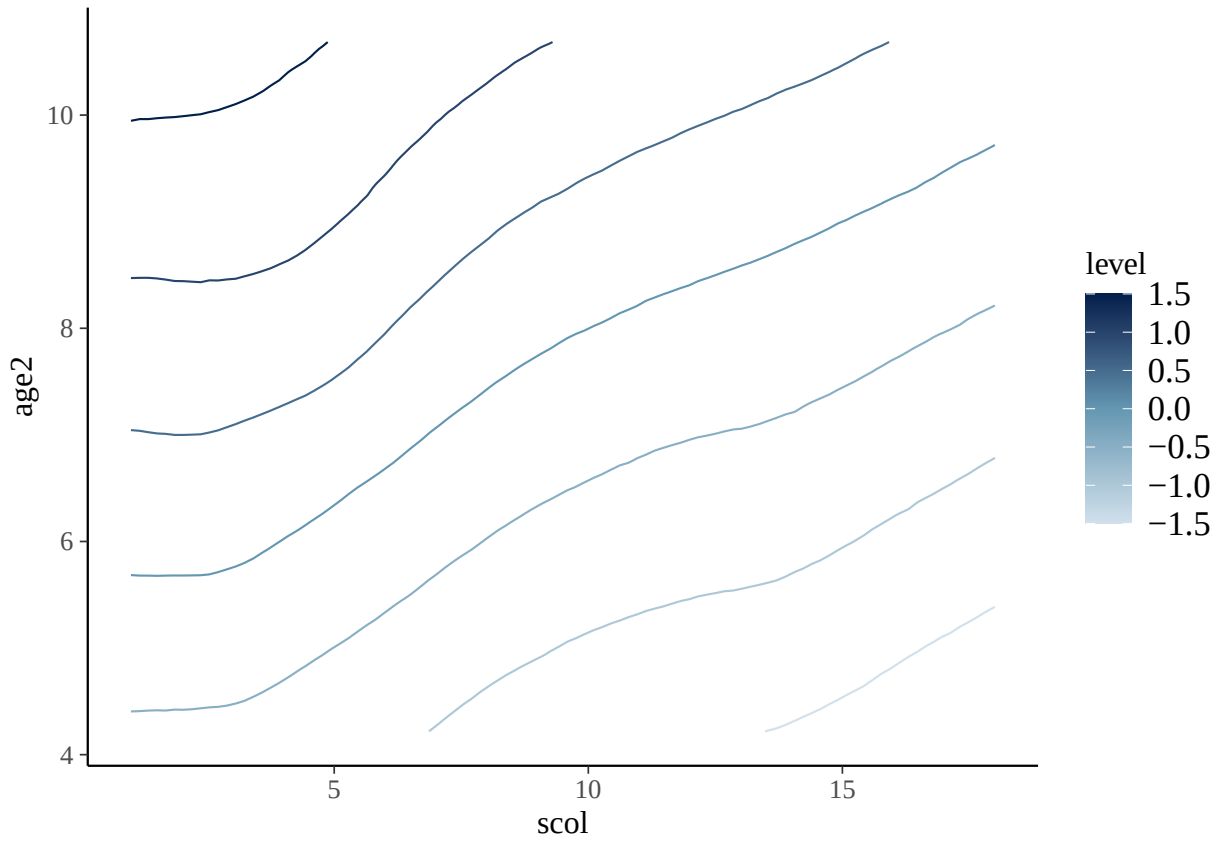
Here we see the basically negative relationship between the logarithm of the expected outcome and `scol`, as well as the basically positive relationship between the logarithm of the expected outcome and `age2`.

```
plot_nonlinear(post_univariate)
```



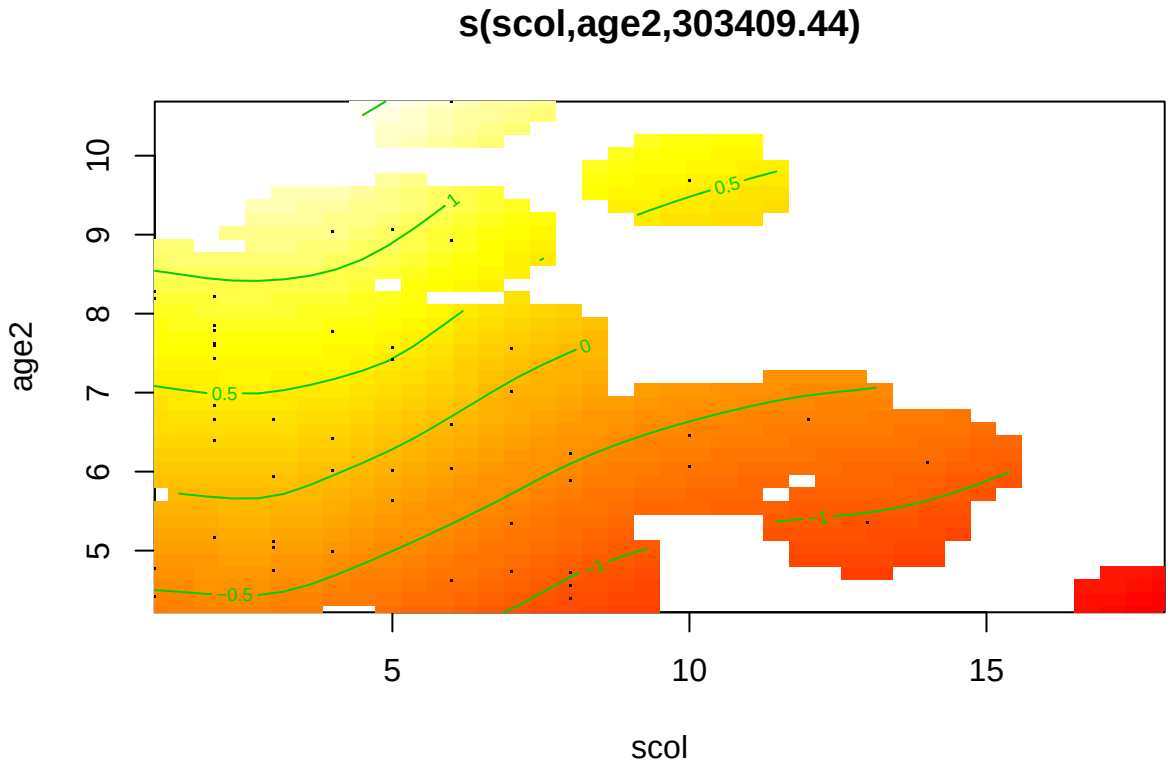
If we try to plot the bivariate function, we have to pay attention to how the *color* changes, which corresponds to the deviation in η from α.

```
plot_nonlinear(post_bivariate)
```



In some cases, you may have better luck with one of the plots from the mgcv package, like

```
plot(post_bivariate$jam, scheme = 2)
```



Here yellow / white corresponds to higher values of η and orange / red corresponds to lower values of η as the predictors vary

2.5 Model Comparison

We can compare these three models, perhaps with a little bit of literal leave-one-out cross-validation and we see that the generalized linear model (i.e. without the smooth functions) is expected to predict future data worse than either of the models with smooth terms.

```
(loo_flat <- loo(post))
```

```
##
## Computed from 4000 by 50 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -607.3  9.1
## p_loo         3.0  0.7
## looic      1214.5 18.1
## ------
## Monte Carlo SE of elpd_loo is 0.0.
##
## All Pareto k estimates are good (k < 0.5).
## See help('pareto-k-diagnostic') for details.
```

```
(loo_univariate <- loo(post_univariate, k_threshold = 0.7)) # refits without observation 2
```

```
##
## Computed from 4000 by 50 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -603.5  9.4
## p_loo         5.5  1.2
## looic      1207.0 18.9
## ------
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     44    89.8%   1250
##  (0.5, 0.7]   (ok)        5    10.2%   324
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

```
(loo_bivariate <- loo(post_bivariate,  k_threshold = 0.7))  # refits without observation 33
```

```
##
## Computed from 4000 by 50 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -603.4  9.5
## p_loo         6.3  1.3
## looic      1206.9 19.0
## ------
```

```
## Monte Carlo SE of elpd_loo is 0.1.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     43    87.8%   1295
##  (0.5, 0.7]   (ok)        6    12.2%   583
##    (0.7, 1]   (bad)       0     0.0%   <NA>
##    (1, Inf)   (very bad)  0     0.0%   <NA>
##
## All Pareto k estimates are ok (k < 0.7).
## See help('pareto-k-diagnostic') for details.
```

Comparing the three models more formally, there is little to recommend the Generalized Linear Model and little to distinguish the two smooth models (if log-predicted density is the utility function)

```
loo_compare(list(loo_flat, loo_univariate, loo_bivariate))
```

```
##                 elpd_diff se_diff
## post_bivariate   0.0       0.0
## post_univariate -0.1       1.0
## post            -3.8       2.8
```

```
loo_model_weights(list(loo_flat, loo_univariate, loo_bivariate))
```

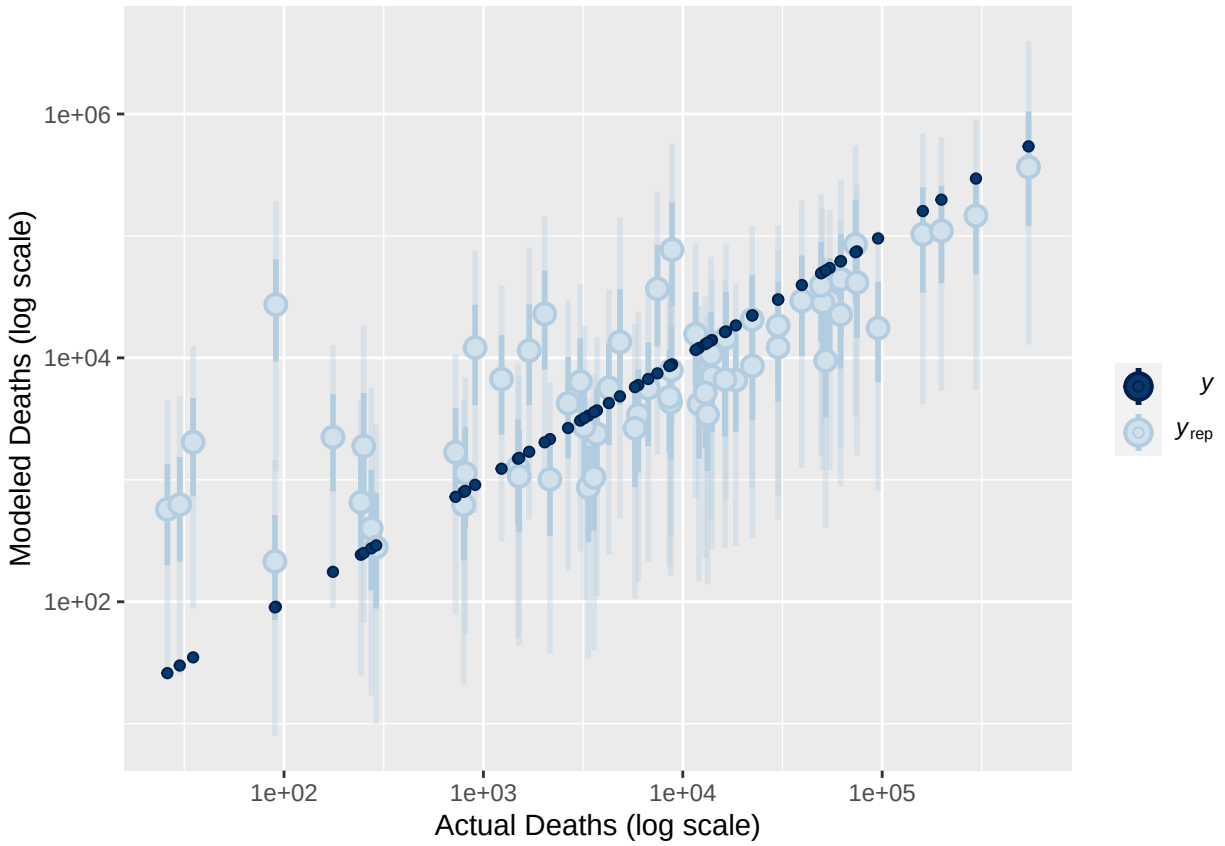
```
## Method: stacking
## ------
##        weight
## model1 0.083
## model2 0.472
## model3 0.444
```

2.6 Addendum

It would be plausible to try a negative binomial model for covid deaths but we then need to add the logarithm of population as a predictor:

```
social_nb <- stan_glm.nb(deaths_cum ~ pop_tot_log + share_older + healthcare_qual +
                           health_exp_pc + detect_index + trust_gov + al_etfra +
                           al_religfra + gini + trust_people + migration_share + share_powerless,
                         data = df_today, prior_intercept = NULL, prior = prior_generic, QR = TRUE,
                         prior_aux = exponential(rate = 1))
```

```
rstanarm::pp_check(social_nb, x = social_nb$y, plotfun = "intervals") +
  scale_x_continuous(trans = 'log10') + scale_y_continuous(trans = 'log10') +
  xlab("Actual Deaths (log scale)") + ylab("Modeled Deaths (log scale)")
```



That is a bit better than the binomial models but still badly overpredicts the number of deaths in low-death countries.