GR5065 Assignment 6

Juan Lopez Martin

1 Coronavirus

1.1 Prior predictive distribution

We start loading the data (code taken from the blog post).

```
data(us_state_populations)

state_pop <- filter(us_state_populations,year==2010) %>%
  select(state,population)

merge_names <- tibble(state.abb,
                      state=state.name)

nyt_data <- read_csv("corona_tscs/retrospective_model_paper/us-states.csv") %>%
  complete(date,state,fill=list(cases=0,deaths=0,fips=0)) %>%
  mutate(month_day=ymd(date)) %>%
  group_by(state) %>%
    arrange(state,date) %>%
  mutate(Difference=cases - dplyr::lag(cases),
         Difference=coalesce(Difference,0,0)) %>%
  left_join(merge_names,by="state")

tests <- read_csv("corona_tscs/retrospective_model_paper/states_daily_4pm_et.csv") %>%
  mutate(month_day=ymd(date)) %>%
  arrange(state,month_day) %>%
  group_by(state) %>%
  mutate(tests_diff=total-dplyr::lag(total),
         cases_diff=positive-dplyr::lag(positive),
         cases_diff=coalesce(cases_diff,positive),
         cases_diff=ifelse(cases_diff<0,0,cases_diff),
         tests_diff=coalesce(tests_diff,total),
         tests_diff=ifelse(tests_diff<0,0,tests_diff)) %>%
  select(month_day,tests="tests_diff",total,state.abb="state")

# merge cases and tests

combined <- left_join(nyt_data,tests,by=c("state.abb","month_day")) %>%
  left_join(state_pop,by="state") %>%
  filter(!is.na(population))

# add suppression data

emergency <- read_csv("corona_tscs/retrospective_model_paper/state_emergency_wikipedia.csv") %>%
```

```
  mutate(day_emergency=dmy(paste0(`State of emergency declared`,"-2020")),
         mean_day=mean(as.numeric(day_emergency),na.rm=T),
         sd_day=sd(as.numeric(day_emergency),na.rm=T),
         day_emergency=((as.numeric(day_emergency) - mean_day)/sd_day)) %>%
  select(state="State/territory",day_emergency,mean_day,sd_day) %>%
  mutate(state=substr(state,2,nchar(state))) %>%
  filter(!is.na(day_emergency))

combined <- left_join(combined,emergency,by="state")

# impute data

combined <- group_by(combined,state) %>%
  mutate(test_case_ratio=sum(tests,na.rm=T)/sum(Difference,na.rm=T)) %>%
  ungroup %>%
  mutate(test_case_ratio=ifelse(test_case_ratio<1 | is.na(test_case_ratio),
                                mean(test_case_ratio[test_case_ratio>1],na.rm=T),test_case_ratio)) %>%
  group_by(state) %>%
    mutate(tests=case_when(Difference>0 & is.na(tests)~Difference*test_case_ratio,
                     Difference==0~0,
                     Difference>tests~Difference*test_case_ratio,
                     TRUE~tests)) %>%
  arrange(state)

# create case dataset

cases_matrix <- select(combined,Difference,month_day,state) %>%
  group_by(month_day,state) %>%
  summarize(Difference=as.integer(mean(Difference))) %>%
  spread(key = "month_day",value="Difference")

cases_matrix_num <- as.matrix(select(cases_matrix,-state))

# create tests dataset

tests_matrix <- select(combined,tests,month_day,state) %>%
  group_by(month_day,state) %>%
  summarize(tests=as.integer(mean(tests))) %>%
  spread(key = "month_day",value="tests")

tests_matrix_num <- as.matrix(select(tests_matrix,-state))

# need the outbreak matrix

outbreak_matrix <- as.matrix(lapply(1:ncol(cases_matrix_num), function(c) {
  if(c==1) {
    outbreak <- as.numeric(cases_matrix_num[,c]>0)
  } else {
    outbreak <- as.numeric(apply(cases_matrix_num[,1:c],1,function(col) any(col>0)))
  }
  tibble(outbreak)
}) %>% bind_cols)
```

```r
colnames(outbreak_matrix) <- colnames(cases_matrix_num)

time_outbreak_matrix <- t(apply(outbreak_matrix,1,cumsum))

just_data <- distinct(combined,state,day_emergency,population) %>% arrange(state)

# now give to Stan

ortho_time <- poly(scale(1:ncol(cases_matrix_num)),degree=3)

real_data <- list(time_all=ncol(cases_matrix_num),
                  num_country=nrow(cases_matrix_num),
                  country_pop=floor(just_data$population/100),
                  cases=cases_matrix_num,
                  ortho_time=ortho_time,
                  phi_scale=.1,
                  count_outbreak=as.numeric(scale(apply(outbreak_matrix,2,sum))),
                  tests=tests_matrix_num,
                  time_outbreak=time_outbreak_matrix,
                  suppress=just_data$day_emergency)
```

For clarity, a summary of the model is shown below. All the parameters in the STAN file correspond to the coefficients shown in this formula. We used tighter priors that the ones proposed in the blog for the coefficients and intercepts. This is particularly relevant for the intercepts, for which the original idea was using a $N(0, 10)$ but for which we expected really low values.

Note that the model displayed in the paper is lacking the $g(.)$ function in the first formula It is necessary to include it because if not the expected value of the beta distribution would not be restricted between 0 and 1. It also shows the formulas for $q_c t$ and $a_c t$ in a somewhat confusing way, something we correct here to clarify they are proportions over the total population (i.e. what the paper really shows is $q_c t_a$ and $q_c t_a$ which correspond to the total number and not the proportion) and using $I_c t$ as a reference instead of I_{ct_a} as it is the quantity we estimate in the first formula.

$$Pr(I_{ct}|T=t) \sim Beta(g(\alpha_1 + \alpha_c + \beta_{O1} \sum_{c=1}^{C} \mathbf{1}(a_{ct'} > 0) \forall t' \in t' < t + \mathbf{1}\beta_{S1} + \beta_{I1} t_o + \beta_{I2} t_o^2 + \beta_{I3} t_o^3 + \mathbf{1}\beta_{S2}), \phi)$$

$$q_{ct} \sim \frac{BB(c_p, g(\alpha_2 + \beta_q I_{ct}), \phi_q)}{c_p}$$

$$a_{ct} \sim \frac{BB(q_{ct}, g(\alpha_3 + \beta_a I_{ct})), \phi_a)}{c_p}$$

We add the prior $q_{ct}/I_{ct} \sim N_{>0.1}(0.15, 0.05)$. That is, we expect the value will be close to 0.15+-0.05 with a lower bound of 0.1.

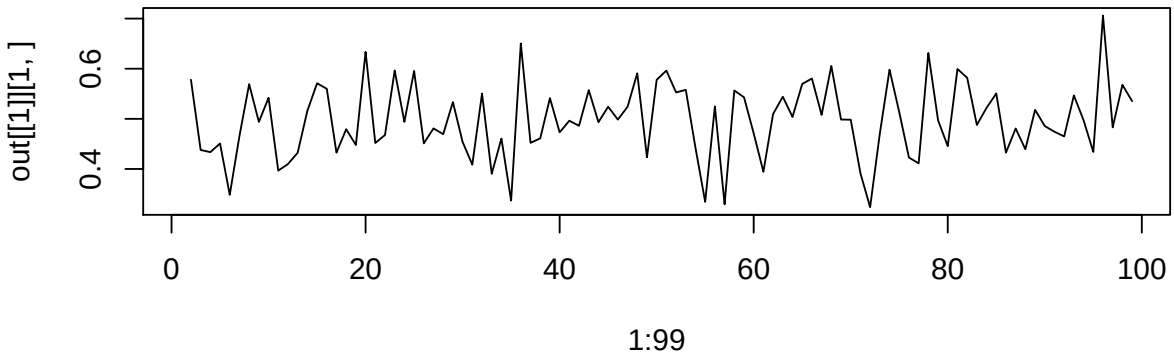
```r
expose_stan_functions("part1.stan")
```

```r
out <- coronavirus_PPD_rng(time_all=ncol(cases_matrix_num),
                           num_country=nrow(cases_matrix_num),
                           country_pop=floor(just_data$population/100),
                           cases=cases_matrix_num,
                           ortho_time=ortho_time,
                           phi_scale=.1,
```

```
                        count_outbreak=as.numeric(scale(apply(outbreak_matrix,2,sum))),
                        tests=tests_matrix_num,
                        # fixed time_outbreak so it works with expose_stan_functions
                        time_outbreak=lapply(seq_len(nrow(time_outbreak_matrix)), function(i) time_ou
                        suppress=just_data$day_emergency)
```

The function actually returns three matrices: I_{ct}, q_{ct}, and a_{ct}. We can visualize, for instance, the progression of the first county I_{ct}. As the parameters are initialized at random the plot does not gives us much information.

```
plot(1:99, out[[1]][1,], 'l')
```



1.2 Mistakes

I noticed two minor discrepancies: the county-specific intercept α_c is not used when calculating $\mu_{I_{ct}}$ and the jacobian adjustment is added while in the blog he argues this is not necessary. However, this potentially reflects the work-in-progress nature of the model.

However, there is a major contradiction that I do not understand. In the stan code, the $\mu_{I_{ct}}$ is calculated as expected. Then the $logit^{-1}$ function is used for the $logq_{ct} - logI_{ct}$ comparison. However, this logit-transormed parameter is never used as the mean parameter in a beta distribution with variance ϕ. Instead, it is inputed directly to calculate $\mu_{q_{ct}}$ and $\mu_{a_{ct}}$, without even being in the model part of the script. Again, I do not find any reason for skipping this step but I may be failing to understand the logic of the model. For me, this should be a quantity to estimate and it should be in the model section, with the ϕ being taken into account as it serves to quantify the difference between the $\mu_{I_{ct}}$ and the empirical proportion of infected. This also generates a chaotic notation that can be seen when in the blog plost the I_{ct} (i.e. proportion of infected in a given state at a given time) suddenly becomes I_{ct_a} (i.e. number of people infected in a given state at a given time). It is probably easier to only talk about the proportions I_{ct}, q_{ct}, a_{ct} (as I did in my summary of the model) or the number of people I_{ct_a}, q_{ct_a}, a_{ct_a}, but mixing both can create a lot of confusion.

2 CES Models

2.1 Stan program

We consider the outcome Y conditional on the variables K, E, and A and the parameters $\gamma, \delta, \delta_1, \rho, \rho_1, \nu$.

$$E(y|K, E, A, \gamma, \delta, \delta_1, \rho, \rho_1, \nu) \sim N(z, \sigma)$$

$$z = \gamma \left[\delta \Big(\delta_1 K^{-\rho_1} + (1-\delta_1) E^{-\rho_1} \Big)^{\rho/\rho_1} + (1-\delta) A^{-\rho} \right]^{-\nu/\rho}$$

The stan program is attached as file part2.stan. We have used weakly informative priors based on the requirements stated in the CES paper.

2.2 Posterior Distribution

```
data(GermanIndustry, package = "micEconCES")
GermanIndustry$time <- GermanIndustry$year - 1960
GermanIndustry <- subset( GermanIndustry, year < 1973 | year > 1975)
GermanIndustry <- GermanIndustry[ , c("Y", "K", "E", "A")]
summary(GermanIndustry)
```

```
##        Y                K                E               A
##  Min.   : 453.5   Min.   : 24.25   Min.   :561.0   Min.   :10.79
##  1st Qu.: 612.5   1st Qu.: 32.76   1st Qu.:609.8   1st Qu.:11.09
##  Median : 816.5   Median : 61.71   Median :628.5   Median :11.53
##  Mean   : 763.0   Mean   : 69.23   Mean   :645.7   Mean   :11.86
##  3rd Qu.: 869.9   3rd Qu.: 85.39   3rd Qu.:702.8   3rd Qu.:12.78
##  Max.   :1002.2   Max.   :145.54   Max.   :748.8   Max.   :13.16
```

```
germany_data <- list(N = nrow(GermanIndustry),
                     y_n = GermanIndustry$Y,
                     k_n = GermanIndustry$K,
                     e_n = GermanIndustry$E,
                     a_n = GermanIndustry$A)
```

```
fit <- stan(file = 'part2.stan', data = germany_data, iter = 5000, chains = 3, control = list(adapt_del
```

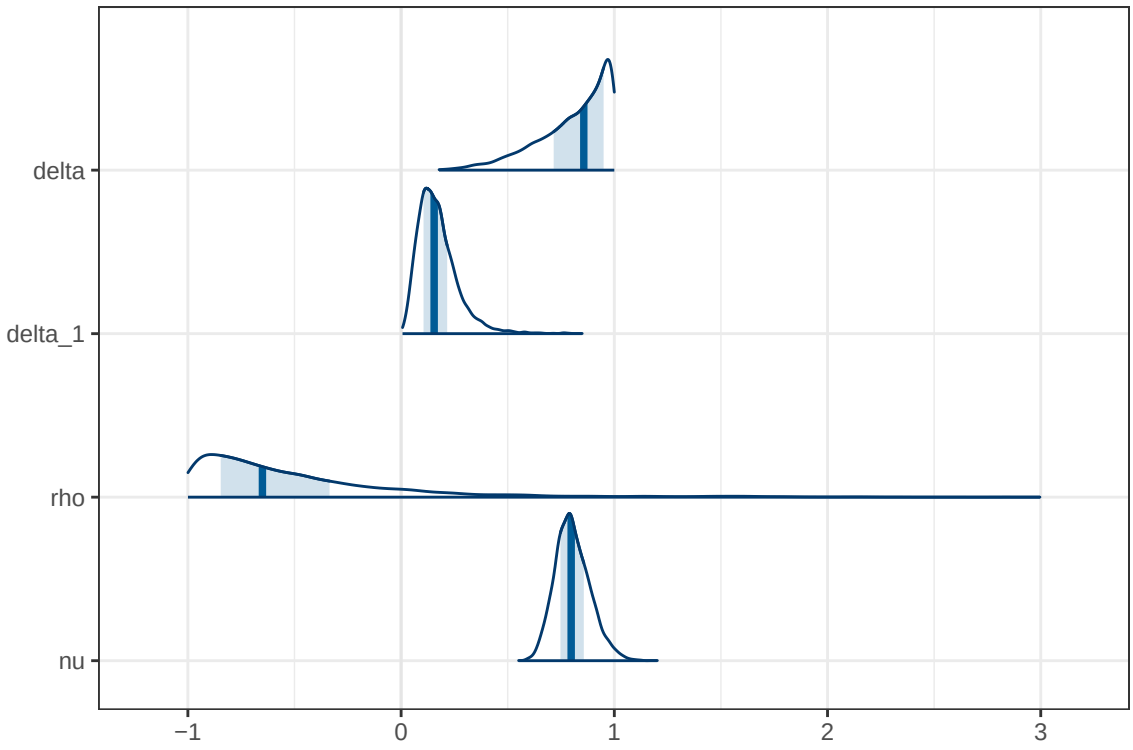
Note the fit does not produce any warnings when increasing the number of iterations, adapting delta and also increasing max_treedepth.

```
fit
```

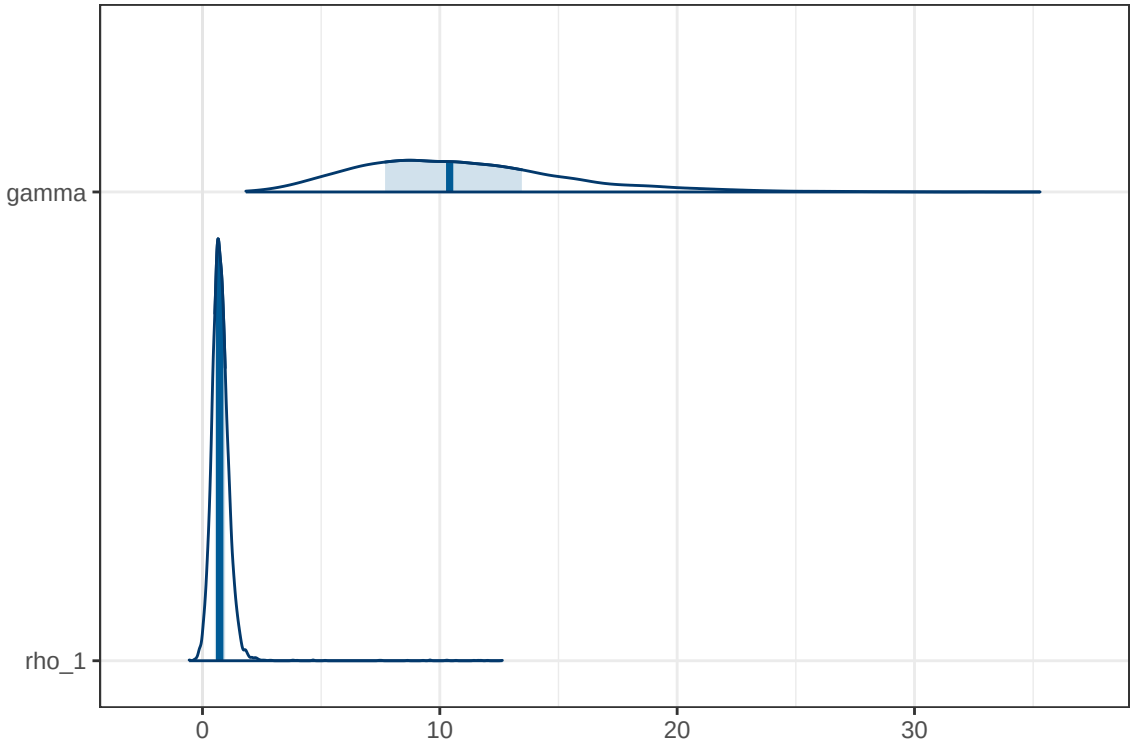
```
## Inference for Stan model: part2.
## 3 chains, each with iter=5000; warmup=2500; thin=1;
## post-warmup draws per chain=2500, total post-warmup draws=7500.
##
##               mean se_mean     sd    2.5%     25%     50%     75%   97.5% n_eff
## p_gamma       0.75    0.00   0.15    0.51    0.62    0.75    0.88    0.99  8018
## p_delta       0.50    0.00   0.20    0.17    0.33    0.50    0.68    0.83  8930
## p_delta_1     0.50    0.00   0.20    0.17    0.32    0.50    0.67    0.83  8513
## p_rho         0.72    0.00   0.16    0.46    0.59    0.72    0.86    0.99  7938
## p_rho_1       0.72    0.00   0.16    0.47    0.59    0.72    0.86    0.98  7857
## p_nu          0.75    0.00   0.15    0.51    0.62    0.75    0.87    0.99  9111
## gamma        10.95    0.09   4.46    4.11    7.69   10.41   13.46   21.34  2737
## delta         0.81    0.00   0.17    0.40    0.72    0.86    0.95    1.00  6117
## delta_1       0.17    0.00   0.10    0.04    0.11    0.15    0.22    0.40  2106
## rho          -0.50    0.01   0.51   -0.99   -0.85   -0.65   -0.34    1.00  1568
## rho_1         0.77    0.02   0.55    0.12    0.52    0.73    0.96    1.53   674
## nu            0.80    0.00   0.08    0.66    0.75    0.80    0.86    0.98  2951
## sigma        26.08    0.04   3.08   20.73   23.88   25.82   28.03   32.74  5996
## b_1           0.08    0.01   0.40    0.00    0.00    0.01    0.05    0.53  3354
## theta_star    0.94    0.14   4.08    0.20    0.37    0.48    0.67    2.52   811
```

```
## theta         0.40    0.18   4.36    0.00    0.00    0.01    0.03    0.65   558
## theta_1       0.60    0.14   3.53    0.09    0.17    0.23    0.33    1.44   685
## theta_2       0.35    0.02   0.85    0.10    0.19    0.25    0.34    0.96  2464
## theta_3       0.40    0.18   4.36    0.00    0.00    0.01    0.03    0.65   558
## au_12        -2.00    3.14 272.34  -28.68    1.06    2.11    4.03   33.57  7517
## au_13         0.66    0.04   2.97   -0.03    0.54    0.60    0.73    1.90  7036
## au_23         0.66    0.04   2.97   -0.03    0.54    0.60    0.73    1.90  7036
## hm_12        -3.57    3.99 345.80  -28.80    1.21    2.25    4.13   31.90  7508
## hm_13        -0.90    0.13  11.28   -3.67   -0.88   -0.67   -0.58   -0.51  7334
## hm_23        -0.72    0.05   4.58   -2.73   -0.87   -0.66   -0.57   -0.50  7404
## lp__       -139.25    0.06   2.96 -145.93 -141.06 -138.89 -137.07 -134.50  2657
##            Rhat
## p_gamma       1
## p_delta       1
## p_delta_1     1
## p_rho         1
## p_rho_1       1
## p_nu          1
## gamma         1
## delta         1
## delta_1       1
## rho           1
## rho_1         1
## nu            1
## sigma         1
## b_1           1
## theta_star    1
## theta         1
## theta_1       1
## theta_2       1
## theta_3       1
## au_12         1
## au_13         1
## au_23         1
## hm_12         1
## hm_13         1
## hm_23         1
## lp__          1
##
## Samples were drawn using NUTS(diag_e) at Thu Apr 30 15:22:56 2020.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
fit_mat <- as.matrix(fit)
mcmc_areas(fit_mat, pars = c("delta", "delta_1", "rho", "nu")) + theme_bw()
```




```
mcmc_areas(fit_mat, pars = c("gamma", "rho_1")) + theme_bw()
```



These values seem plausible and consistent with the tables 1, 2, and 3 of section 5.3. Note there is a strong uncertainty around γ, ρ, and to some extent ρ_1.

2.3 Comparison to frequentist estimation

The frequentist approach tries to find the parameter values that are most likely to have originated the data according to the model. That is, the approach is based on maximizing the likelihood function. This is relatively easy in some cases, but in others, such as this one, the optimization task becomes really hard as "very different parameter vectors result in very similar values of the objective function". As explained in the quote, this may even imply the optimization to be stuck in local optima instead of the global maximum. Additionally, the complicated geometry of the objective function can, as stated in the homework, invalidate the assumptions that are required for NHST.

The bayesian approach does not have this problem simply because it does not tries to solve an optimization problem. Instead of trying to get a set of point estimates of the parameters, it provides a joint probability distribution for the parameters given the data. That is, it results in a range of possible values for all the parameters in the model. Note that, although a complicated geometry can be troublesome in some cases, HMC is in general well-suited to navigate the posterior distribution. Therefore, the Bayesian approach does not only simplify the identification, but it also provides a useful range of values for the parameters that can help us make better inferences.

Another potential advantage of Bayesian methods comes in the use of priors. In the original CES paper there seems to be a discussion on whether some of the values of the parameters are economically meaningful or not. I'm not familiar with this field of study, but it may be possible that including more informative priors than I did could partially solve this problem. That is, if economic theory proposes some parameter values are more likely than others, this information can be included in the model. Furthermore, relationships between variables (i.e. quantities of interest) can also be informed by priors. Thus, partially pooling towards what we know a priori are reasonable values for the parameters could also be a useful idea in this example.