Modello gerarchico

Laura Balasso

10/24/2020

Hierarchical model for italian regions

```
regions <- c('Lazio', 'Lombardia', 'Abruzzo', 'Veneto', 'Emilia-Romagna', 'Toscana', 'Campania', 'Friuli

regions
```

```
##  [1] "Lazio"                "Lombardia"            "Abruzzo"
##  [4] "Veneto"               "Emilia-Romagna"       "Toscana"
##  [7] "Campania"             "Friuli Venezia Giulia" "Marche"
## [10] "Sicilia"              "Calabria"             "Sardegna"
```

```
hier_data <- get_hier_data(data_it, regions, initial_date = as.Date('2020-08-15') )

p_delay <- get_delay_distribution()

### lockdown effect

lockdown_start <- as.Date('2020-03-09')
lockdown_end <- as.Date('2020-05-03')

lockdown <- rep(0, length(hier_data$dates))
lockdown[which(hier_data$dates> lockdown_start+10 & hier_data$dates <= lockdown_end)] <- 1
grow <- which(hier_data$dates>=lockdown_start &hier_data$dates <= lockdown_start +10)
lockdown[grow] <- (grow - which(hier_data$dates ==lockdown_start))^2 /100

## school effect

school_opening <- as.Date('2020-09-14')
school <- rep(0, length(hier_data$dates))
school[which(hier_data$dates > school_opening +10)] <- 1
grow_s <- which(hier_data$dates>=school_opening &hier_data$dates <= school_opening +10)
school[grow_s] <- (grow_s - which(hier_data$dates ==school_opening))^2 /100

stan_data_hier <- list(J = length(regions),
                       N = nrow(hier_data$exposures),
                       N_nonzero = length(hier_data$nonzero_days),
                       nonzero_days = hier_data$nonzero_days,
                       conv_gt = get_gt_convolution(nrow(hier_data$exposures)),
                       length_delay = length(p_delay),
                       p_delay = p_delay,
                       exposures = hier_data$exposures,
                       nonzero_positives = hier_data$positives[hier_data$nonzero_days ,],
                       school = school[hier_data$nonzero_days]
```

```
)


compiled_hier <- stan_model('hier_model_school.stan')

fit_hier <- sampling(compiled_hier, data = stan_data_hier, iter= 2000, cores=getOption("mc.cores", 1L))
```

```
##
## SAMPLING FOR MODEL 'hier_model_school' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.023901 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 239.01 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 2064.4 seconds (Warm-up)
## Chain 1:                1235.69 seconds (Sampling)
## Chain 1:                3300.09 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'hier_model_school' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.013171 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 131.71 seconds.
## Chain 2: Adjust your expectations accordingly!
## Chain 2:
## Chain 2:
## Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2:
```

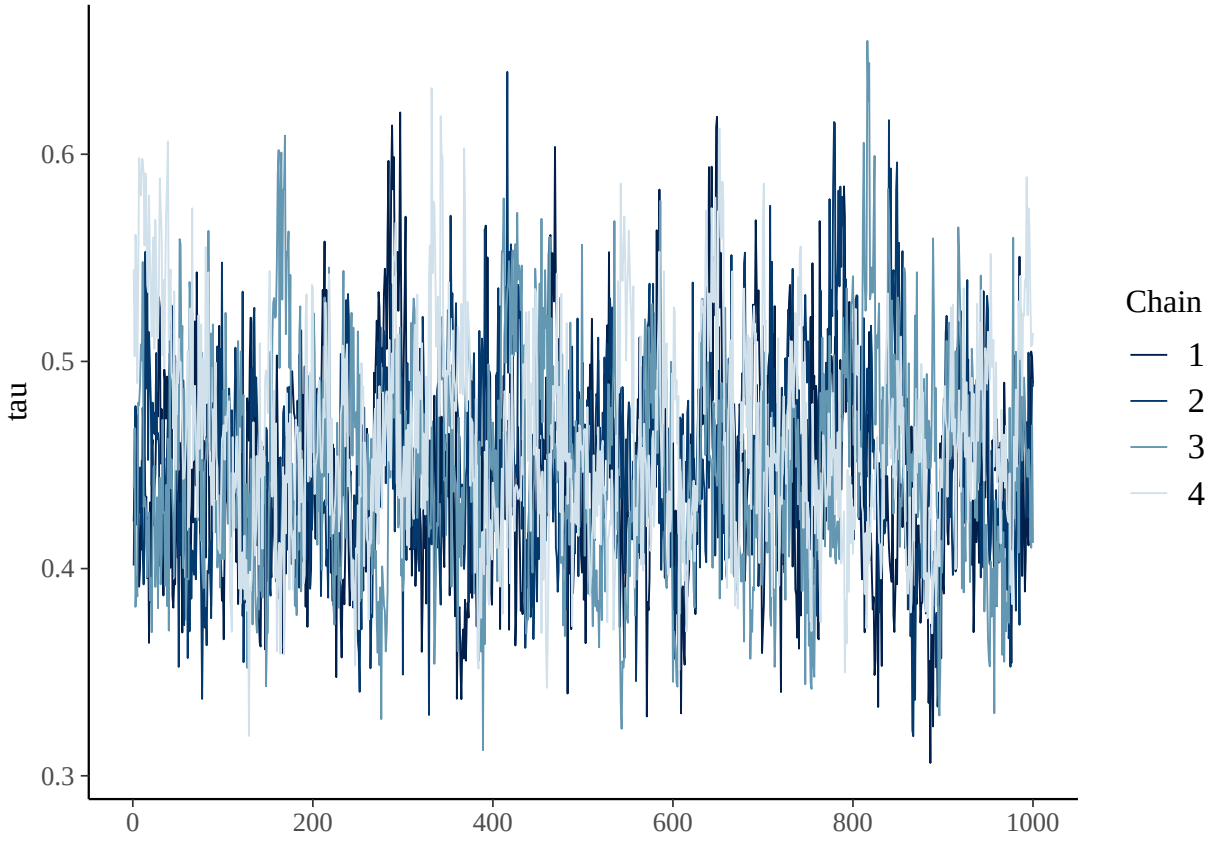
```
## Chain 2:  Elapsed Time: 1834.06 seconds (Warm-up)
## Chain 2:                1155.8 seconds (Sampling)
## Chain 2:                2989.86 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'hier_model_school' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.00903 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 90.3 seconds.
## Chain 3: Adjust your expectations accordingly!
## Chain 3:
## Chain 3:
## Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3:
## Chain 3:  Elapsed Time: 1865.41 seconds (Warm-up)
## Chain 3:                1159.63 seconds (Sampling)
## Chain 3:                3025.03 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'hier_model_school' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.010297 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 102.97 seconds.
## Chain 4: Adjust your expectations accordingly!
## Chain 4:
## Chain 4:
## Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4:
## Chain 4:  Elapsed Time: 1939.11 seconds (Warm-up)
## Chain 4:                1155.18 seconds (Sampling)
## Chain 4:                3094.29 seconds (Total)
## Chain 4:
```

```
## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be
## Running the chains for more iterations may help. See
## http://mc-stan.org/misc/warnings.html#bulk-ess
```

Trace plots

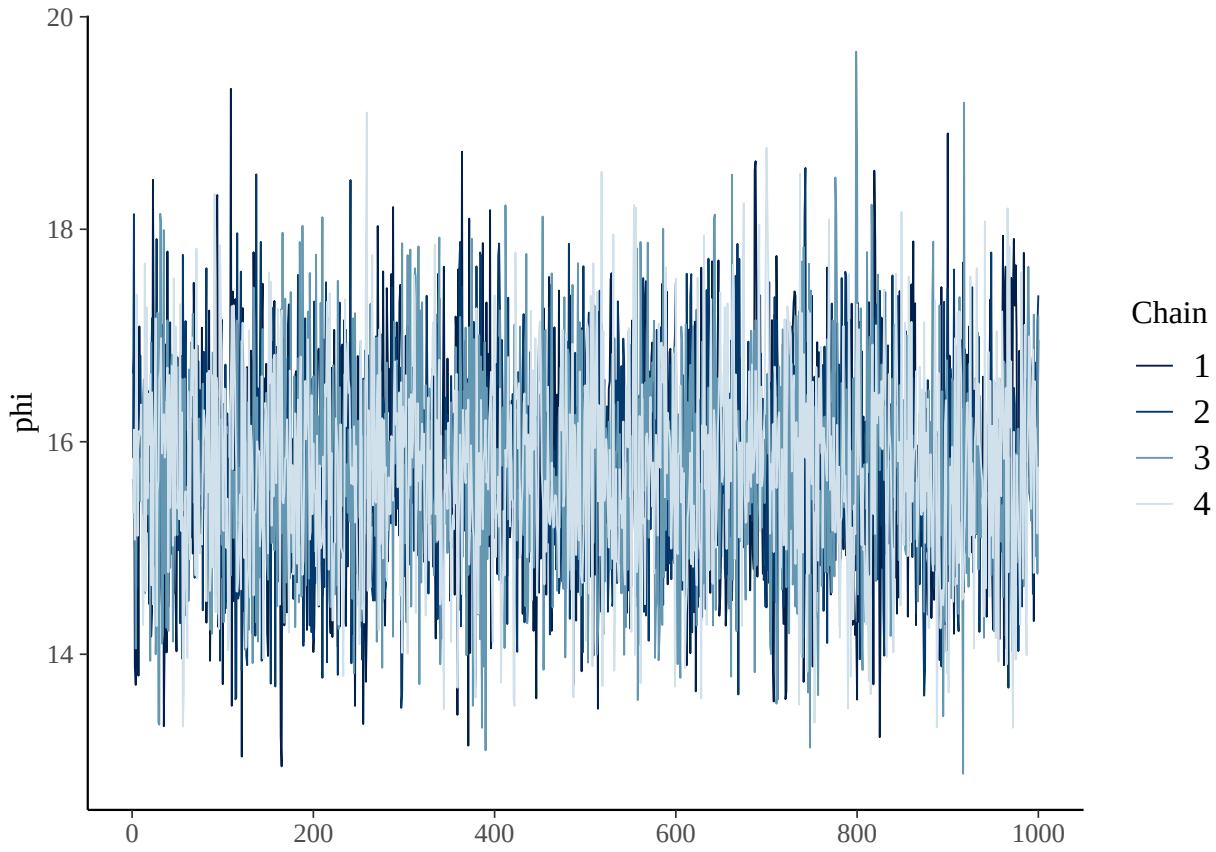
```
mcmc_trace(as.array(fit_hier, pars = c('tau')),
           np = nuts_params(fit_hier)
)
```

```
## No divergences to plot.
```



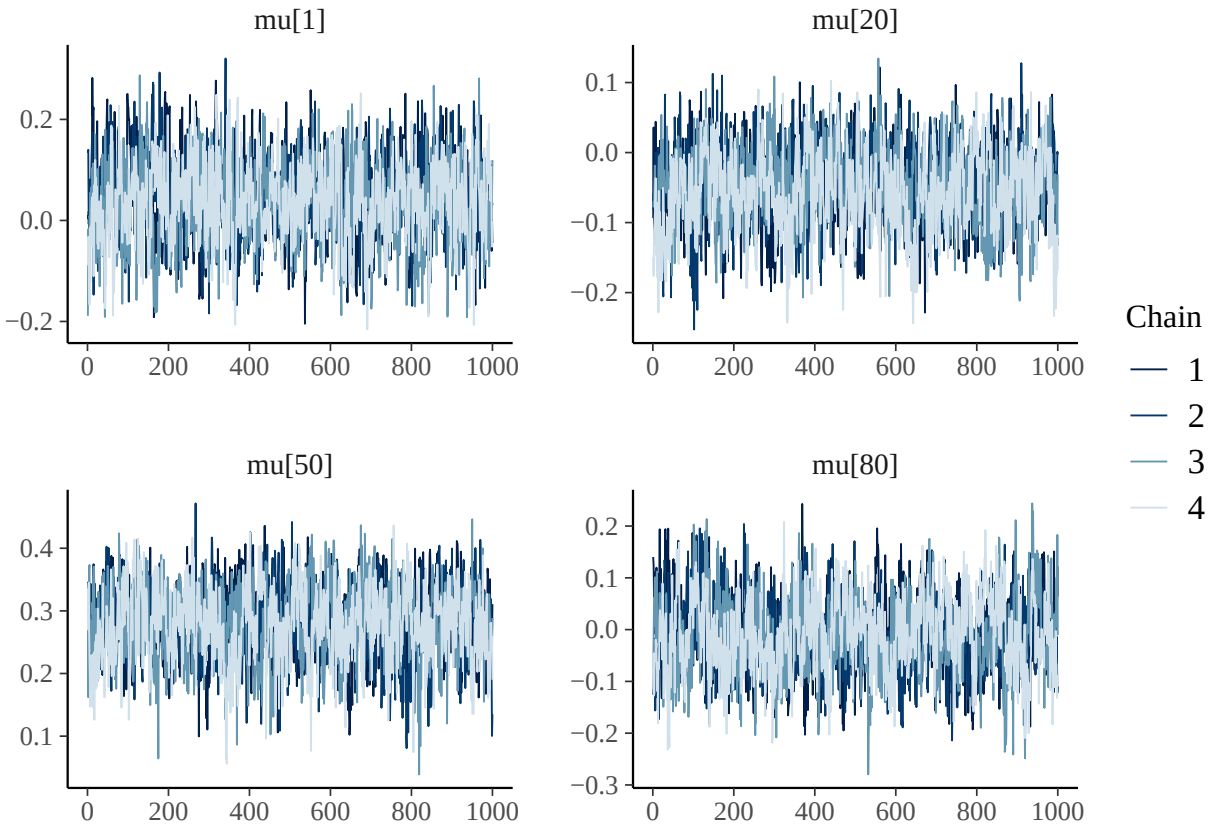

```
mcmc_trace(as.array(fit_hier, pars = c('phi')),
           np = nuts_params(fit_hier)
)
```

```
## No divergences to plot.
```




```
mcmc_trace(as.array(fit_hier, pars = c('mu[1]', 'mu[20]', 'mu[50]', 'mu[80]')),
           np = nuts_params(fit_hier)
)
```

```
## No divergences to plot.
```



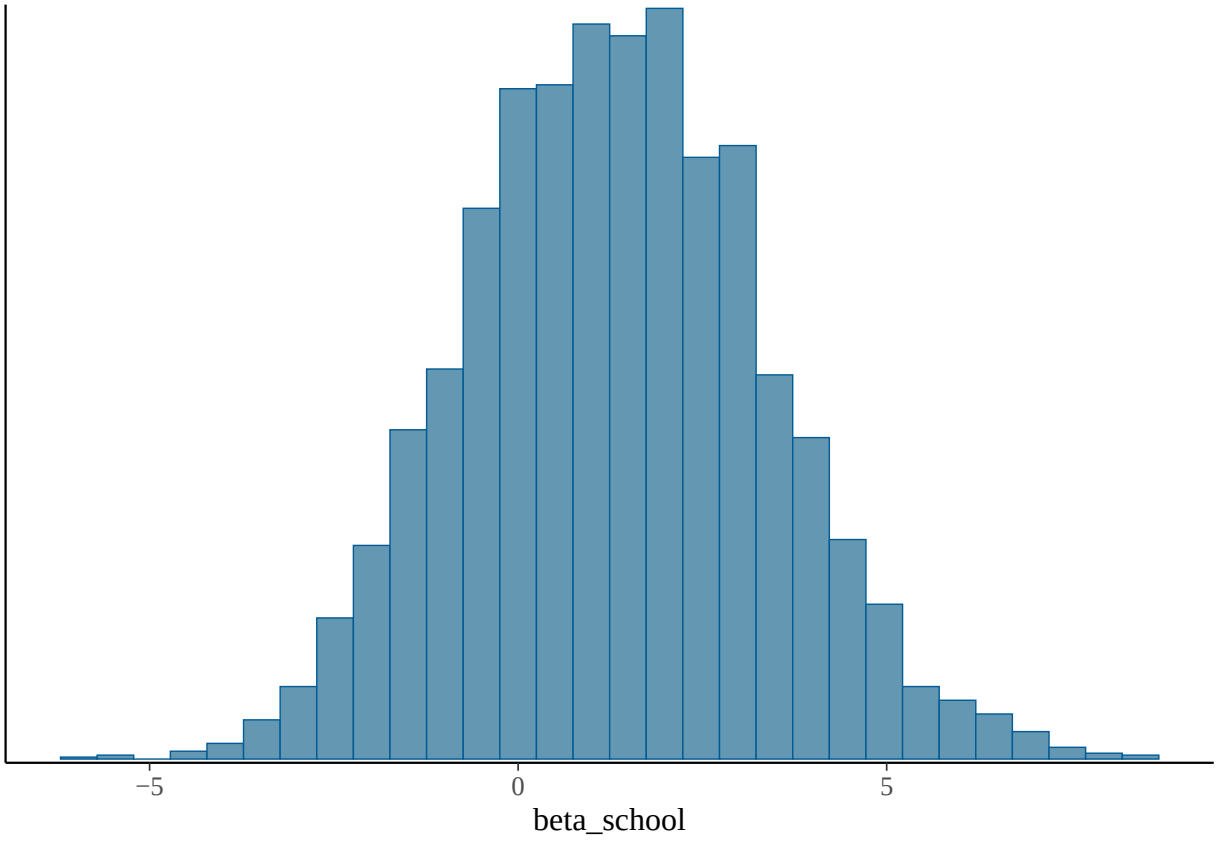
School opening effect

```
print(fit_hier, pars= 'beta_school')
```

```
## Inference for Stan model: hier_model_school.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##              mean se_mean   sd  2.5%   25% 50%  75% 97.5% n_eff Rhat
## beta_school 1.31    0.05 2.03 -2.54 -0.09 1.3 2.68  5.32  1810    1
##
## Samples were drawn using NUTS(diag_e) at Sun Nov  8 17:53:53 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).
```

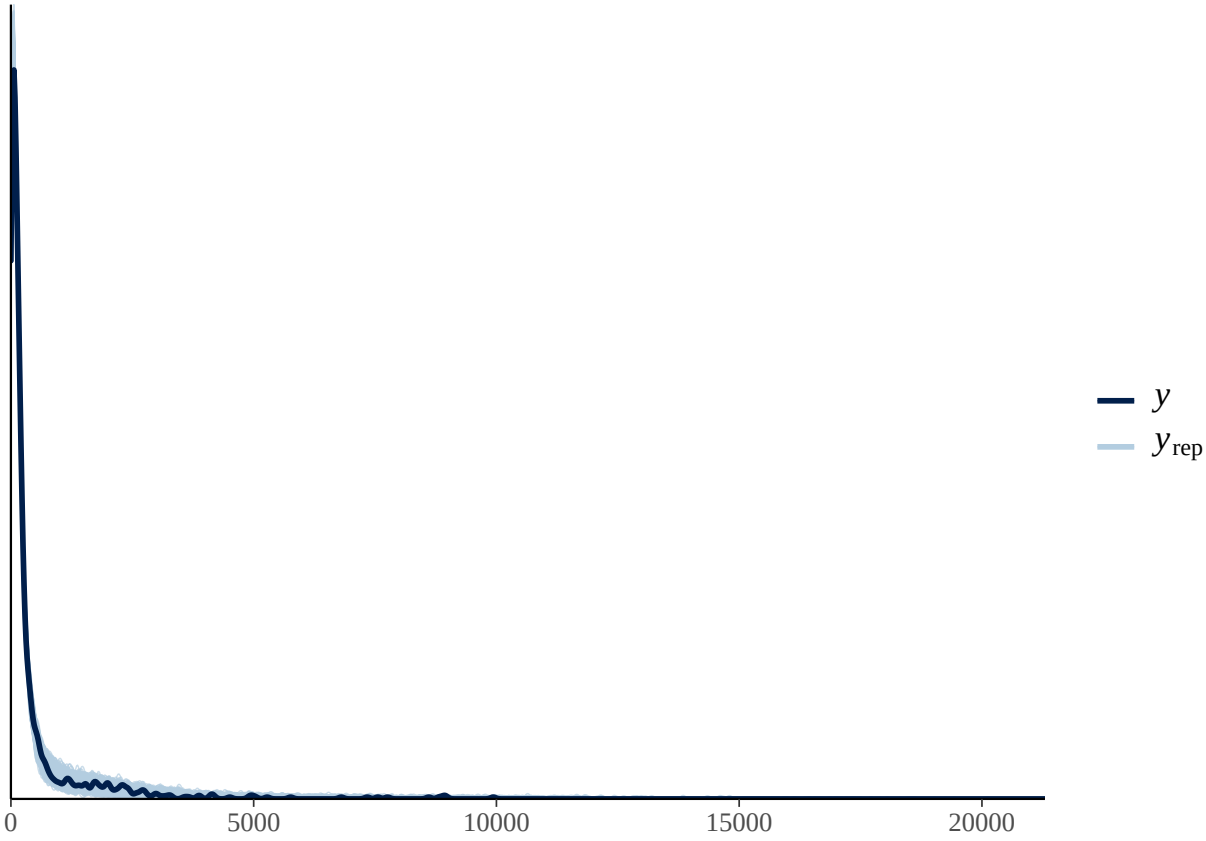
```
mcmc_hist(fit_hier, pars = 'beta_school')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

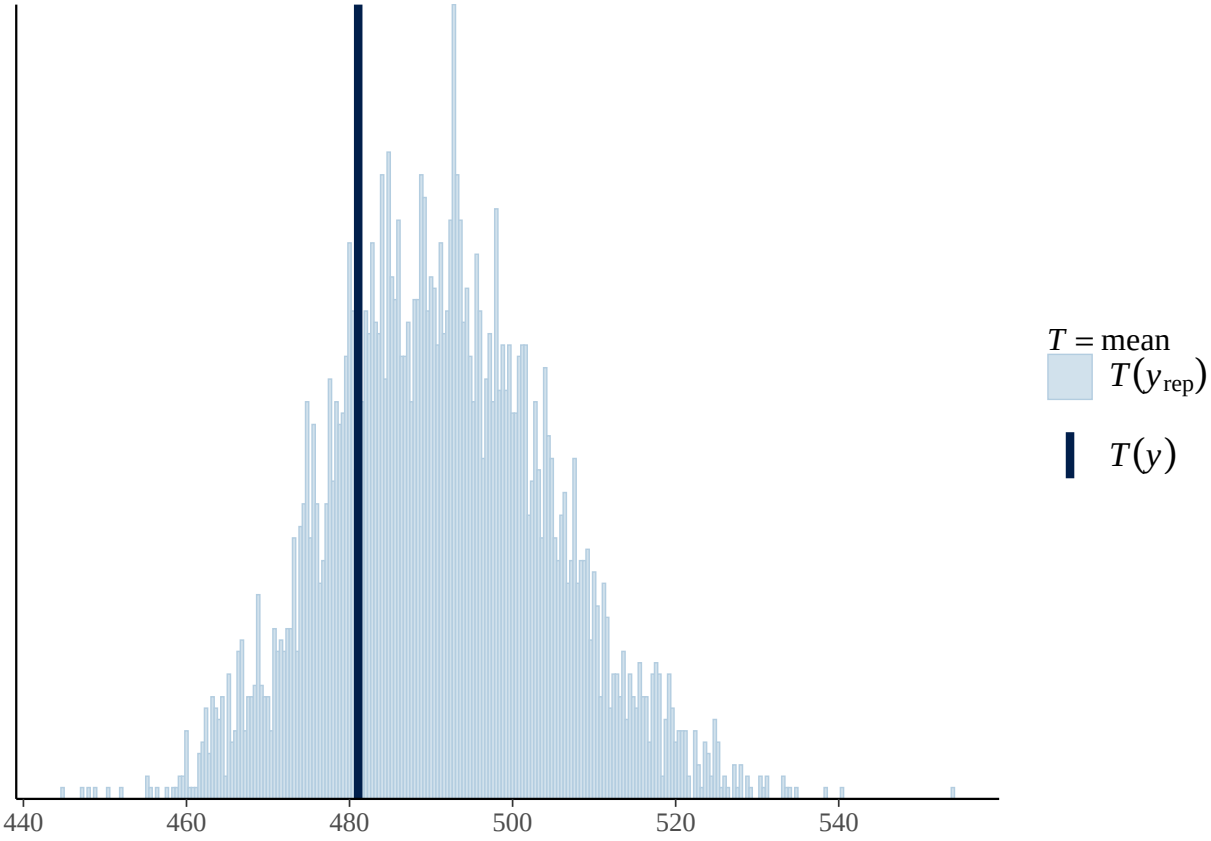


Posterior predictive check

```
y_rep <- as.matrix(fit_hier, pars = "y_rep")
ppc_dens_overlay(y = as.vector(stan_data_hier$nonzero_positives), y_rep[1:1000, ])
```




```
ppc_stat(y=as.vector(stan_data_hier$nonzero_positives), yrep =y_rep,stat="mean", binwidth = 0.4)
```



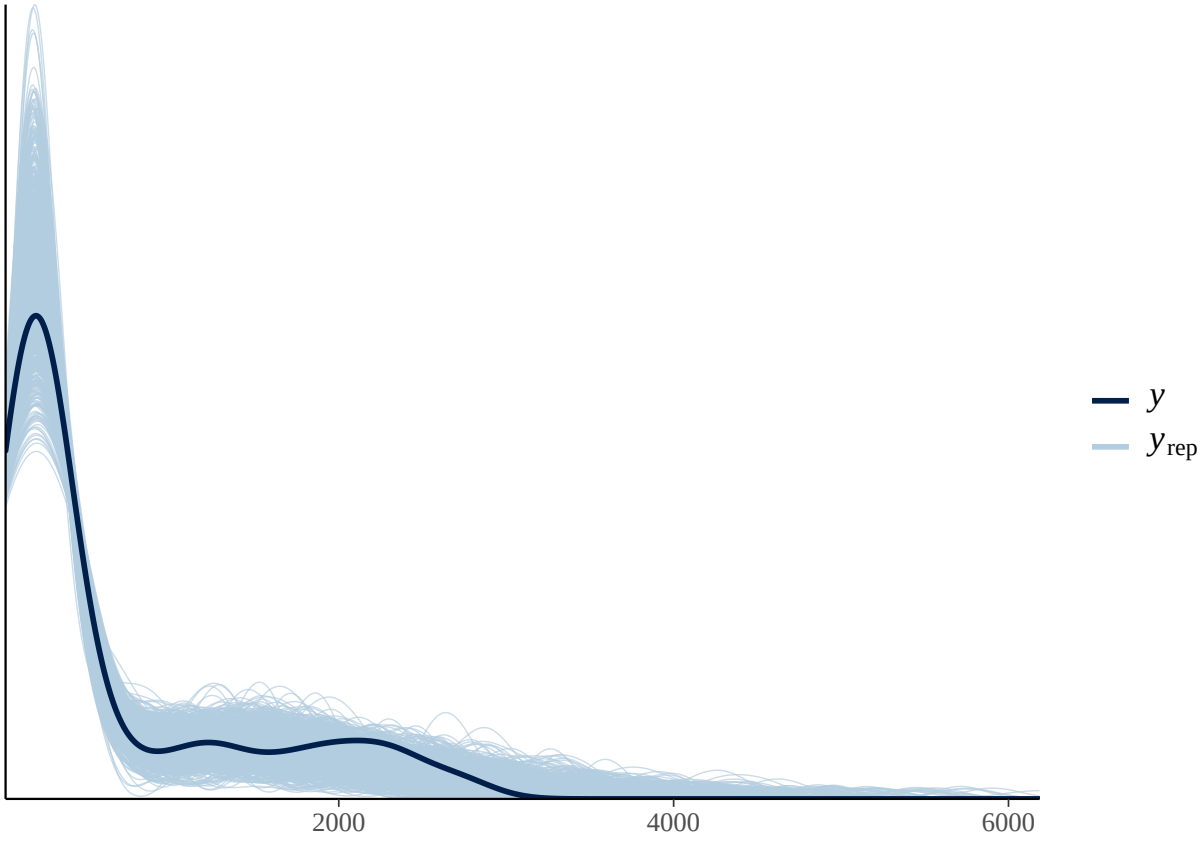
Posterior predictive check by region

```
regional_yrep_idx <- function(region, regions_vector, nonzero_days){
  region_idx <- which(regions_vector == region)
  yrep_idx <- (region_idx-1)* length(nonzero_days) + 1
  range <- yrep_idx : (yrep_idx + length(nonzero_days)-1)
  return(range)
}
```

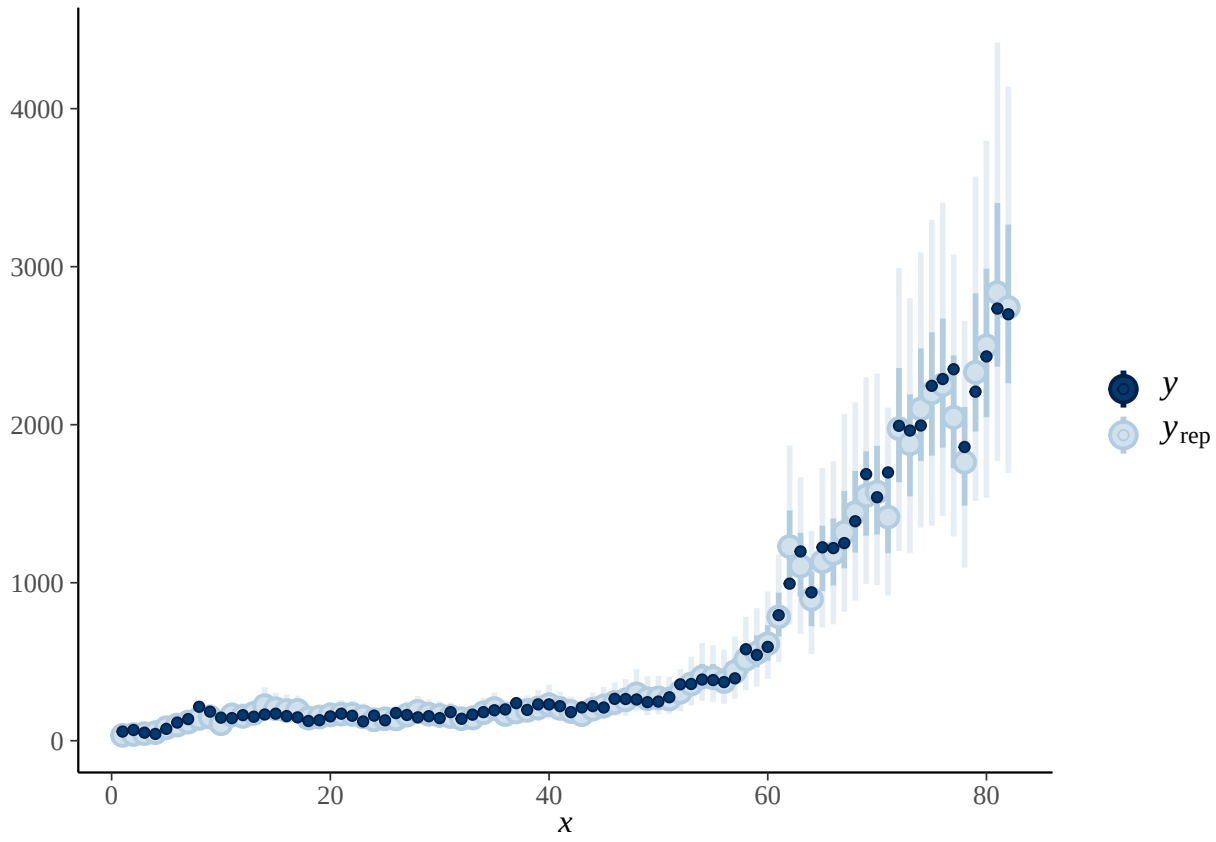
```
groups <- function(regions, nonzero_days){
  group <- rep(regions[1], length(nonzero_days))
  for(r in 2:length(regions))
    group <- c(group, rep(regions[r], length(nonzero_days)))

  return(group)
}
```

Lazio

```
ppc_dens_overlay(y = stan_data_hier$nonzero_positives[, which(regions == 'Lazio')], y_rep[1:1000,regiona
```

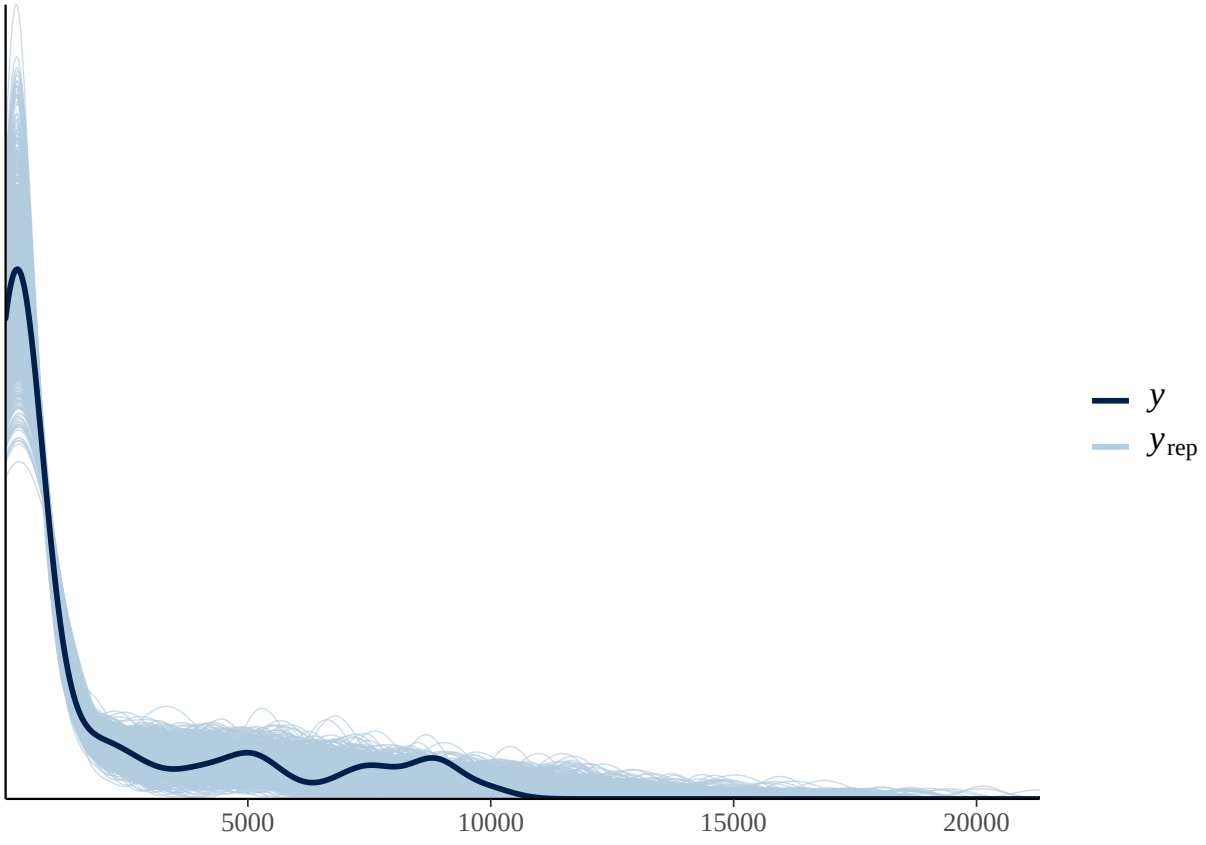



```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Lazio')], y_rep[1:1000,regional_y
```

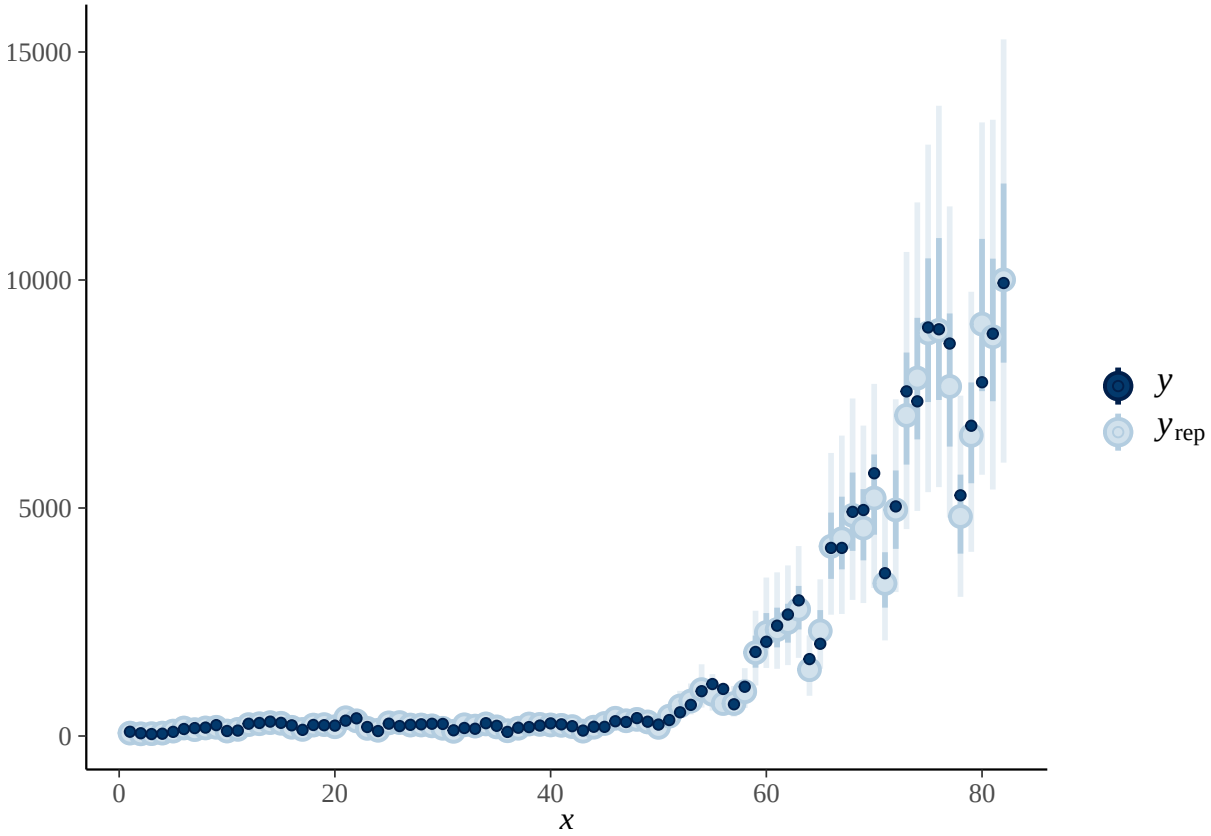


Lombardia

```
ppc_dens_overlay(y = stan_data_hier$nonzero_positives[, which(regions == 'Lombardia')], y_rep[1:1000,re
```

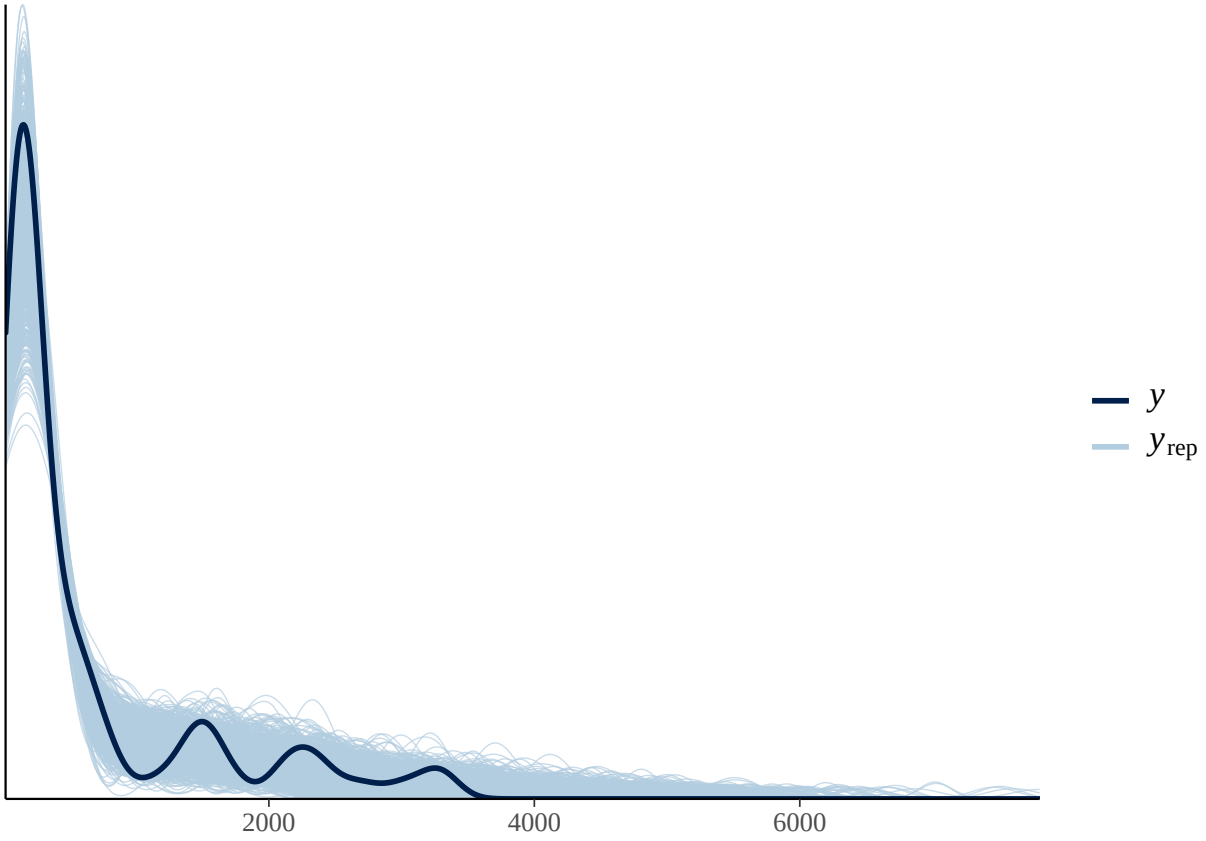



```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Lombardia')], y_rep[1:1000,regio
```

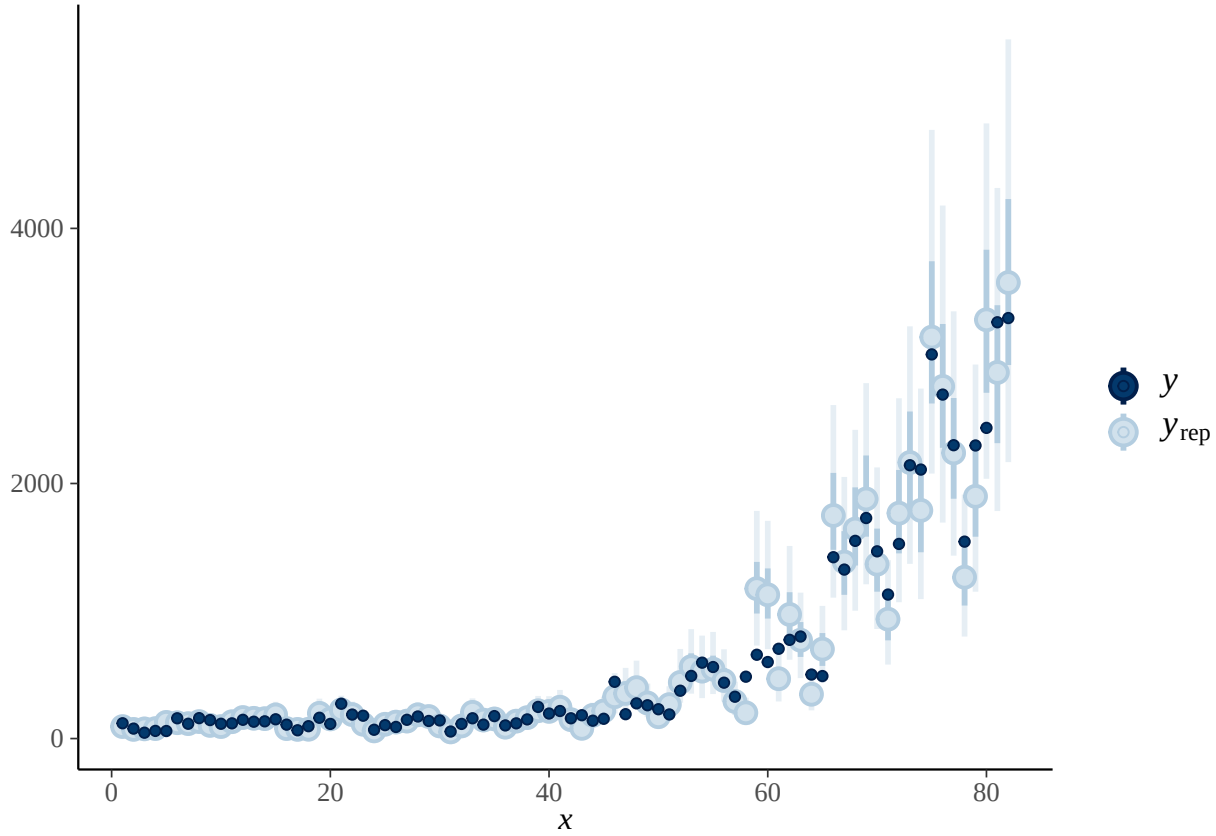


Veneto

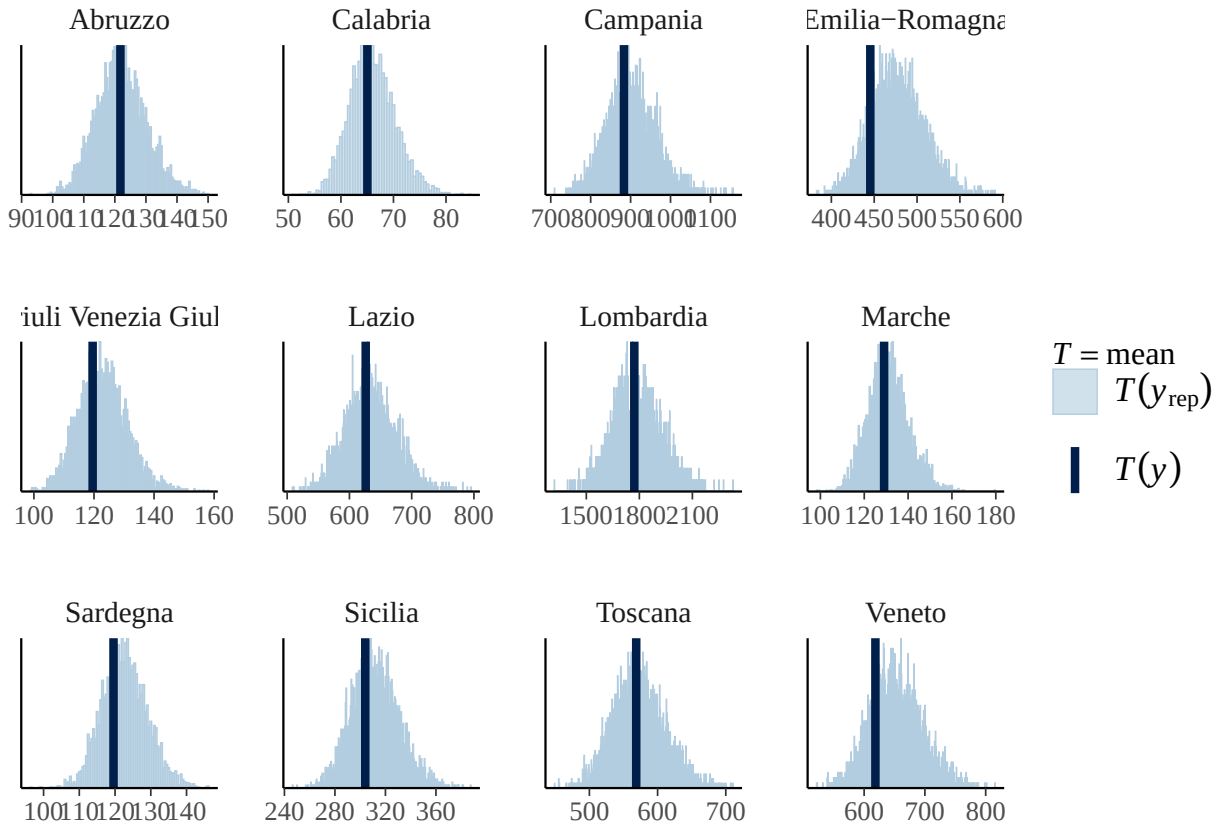
```
ppc_dens_overlay(y = stan_data_hier$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regio
```



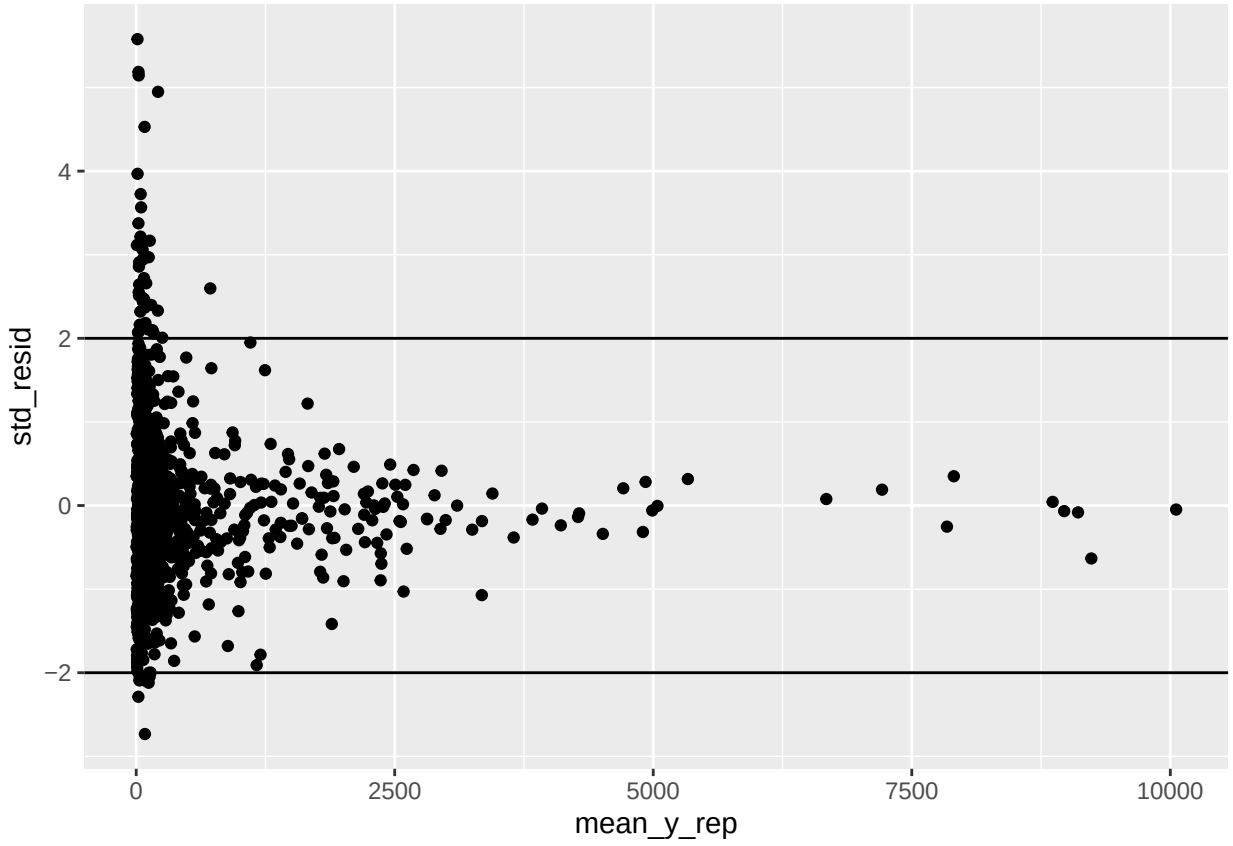

```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regional_
```




```
ppc_stat_grouped(y=as.vector(stan_data_hier$nonzero_positives), yrep =y_rep, group = groups(regions, sta
```

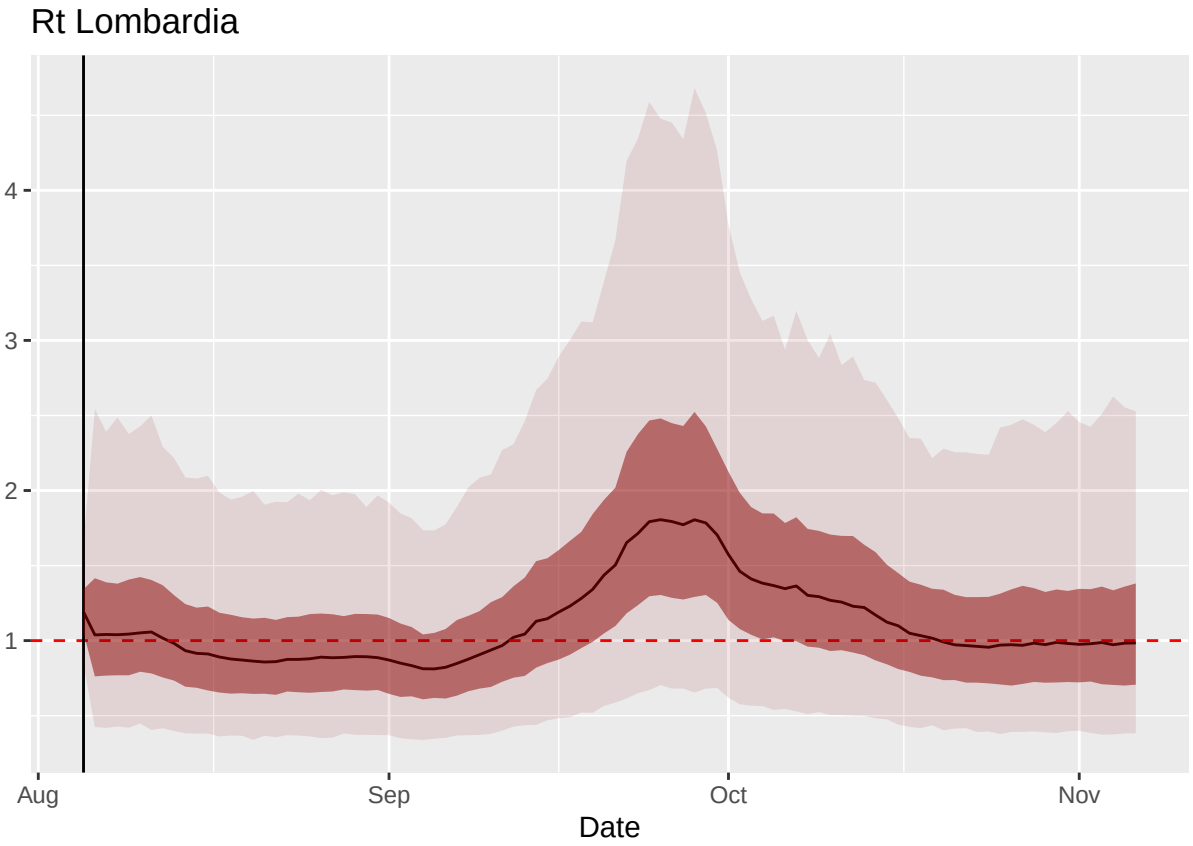


```
mean_inv_phi<-mean(rstan::extract(fit_hier)$inv_phi)
mean_y_rep<-colMeans(y_rep)
std_resid<-(as.vector(stan_data_hier$nonzero_positives)-mean_y_rep)/sqrt(mean_y_rep+mean_y_rep^2*mean_i
qplot(mean_y_rep,std_resid)+hline_at(2)+hline_at(-2)
```



Rt Lombardia

```
summary <- summary(fit_hier)
rt_1 <- which(rownames(summary$summary) == 'r_t[1,2]')
rt_index <- seq(rt_1, rt_1 + stan_data_hier$N * length(regions)  -1 , by=length(regions))
rt_median_lomb <- summary$summary[rt_index, '50%']
min_rt_50_interval <- summary$summary[rt_index, '25%']
max_rt_50_interval <- summary$summary[rt_index, '75%']
min_rt_95_interval <- summary$summary[rt_index, '2.5%']
max_rt_95_interval <- summary$summary[rt_index, '97.5%']

ggplot(data= NULL, aes(x = hier_data$dates, y=rt_median_lomb ) ) +
  geom_line()+
  xlab('Date') +
  ylab('') +
  ggtitle( 'Rt Lombardia')+
  geom_hline(yintercept=1, linetype="dashed", color = "red") +
  geom_vline(xintercept = hier_data$dates[1]) +
  geom_ribbon(aes(ymin = min_rt_50_interval, ymax = max_rt_50_interval), alpha= 0.5, fill = 'darkred') +
  geom_ribbon(aes(ymin = min_rt_95_interval, ymax = max_rt_95_interval), alpha= 0.1, fill = 'darkred')
```


Rt Lombardia
4
3
2
1
Aug
Sep
Oct
Nov
Date