Hierarchical model with area subdivision

Laura Balasso

11/13/2020

Hierarchical model for italian regions

```
regions <- c('Lombardia', 'Veneto', 'Lazio', 'Abruzzo', 'Liguria', 'Puglia', 'Basilicata')

regions
```

```
## [1] "Lombardia"  "Veneto"     "Lazio"      "Abruzzo"    "Liguria"
## [6] "Puglia"     "Basilicata"
```

```
hier_data <- get_hier_data(data_it, regions, Y, O, R, initial_date = as.Date('2020-09-20') )

p_delay <- get_delay_distribution()


stan_data_hier <- list(J = length(regions),
                       N = nrow(hier_data$exposures),
                       N_nonzero = length(hier_data$nonzero_days),
                       nonzero_days = hier_data$nonzero_days,
                       conv_gt = get_gt_convolution_ln2(nrow(hier_data$exposures)),
                       length_delay = length(p_delay),
                       p_delay = p_delay,
                       exposures = hier_data$exposures,
                       nonzero_positives = hier_data$positives[hier_data$nonzero_days , ],
                       yellow = hier_data$yellow_dummies,
                       orange = hier_data$orange_dummies,
                       red = hier_data$red_dummies
)


compiled_hier <- stan_model('../stan/hier_model_area.stan')
```

```
## Trying to compile a simple C file
```

```
## Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
## clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
## ^
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/src/Core/util/
## namespace Eigen {
```

```
##                ^
##                ;
## In file included from <built-in>:1:
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/StanHeaders/incl
## In file included from /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/inclu
## /Library/Frameworks/R.framework/Versions/4.0/Resources/library/RcppEigen/include/Eigen/Core:96:10: fa
## #include <complex>
##          ^~~~~~~~
## 3 errors generated.
## make: *** [foo.o] Error 1
```

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

```
##
## SAMPLING FOR MODEL 'hier_model_area' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00979 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 97.9 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: 1080.61 seconds (Warm-up)
## Chain 1:                1229.16 seconds (Sampling)
## Chain 1:                2309.76 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'hier_model_area' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.005202 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 52.02 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: 1091.62 seconds (Warm-up)
## Chain 2:                795.682 seconds (Sampling)
## Chain 2:                1887.3 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'hier_model_area' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.005042 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 50.42 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: 1007.19 seconds (Warm-up)
## Chain 3:                1001.94 seconds (Sampling)
## Chain 3:                2009.12 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'hier_model_area' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 0.005087 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 50.87 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: 1163.74 seconds (Warm-up)
## Chain 4:                1325.96 seconds (Sampling)
## Chain 4:                2489.71 seconds (Total)
## Chain 4:
```

```
## Warning: There were 3 divergent transitions after warmup. Increasing adapt_delta above 0.8 may help.
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

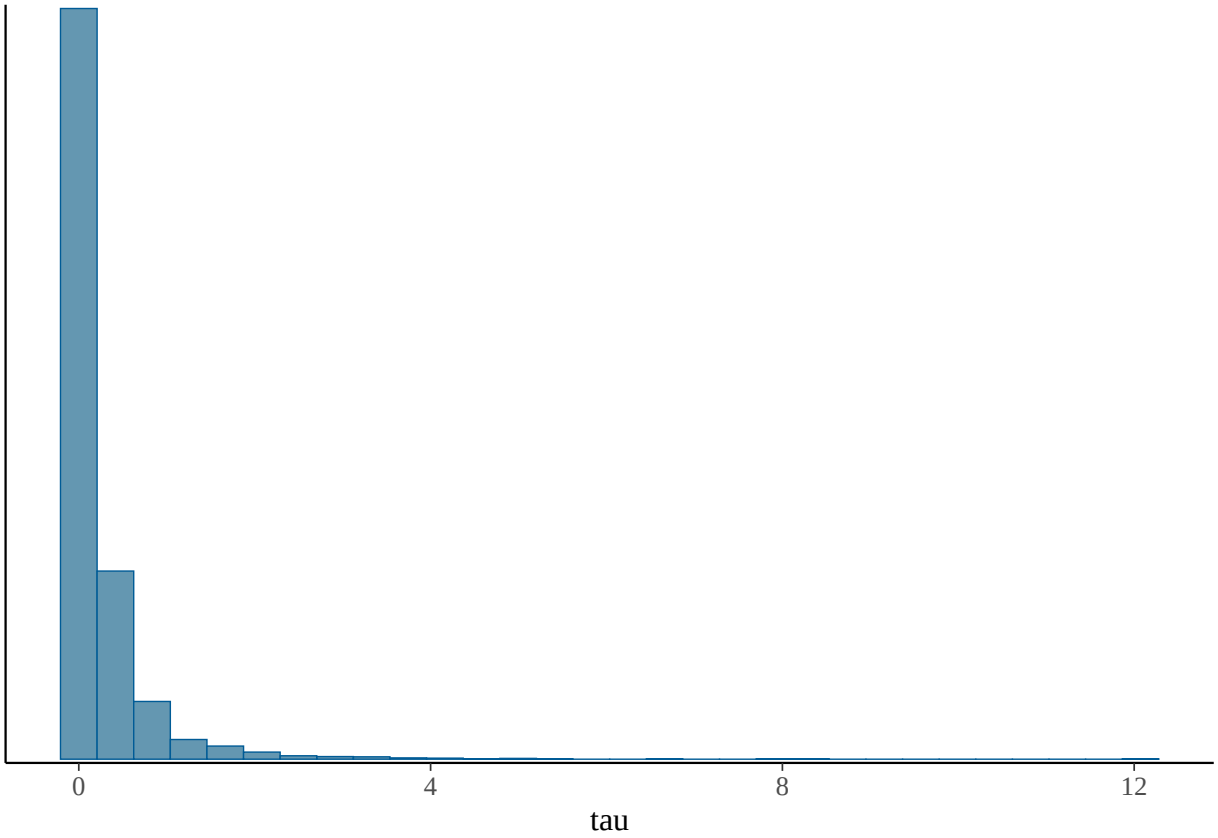
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

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

```
## Inference for Stan model: hier_model_area.
## 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
## tau 0.25    0.01 0.56    0 0.02 0.06 0.25   1.6  3493    1
##
## Samples were drawn using NUTS(diag_e) at Sat Dec 12 17:36:53 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='tau')
```

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

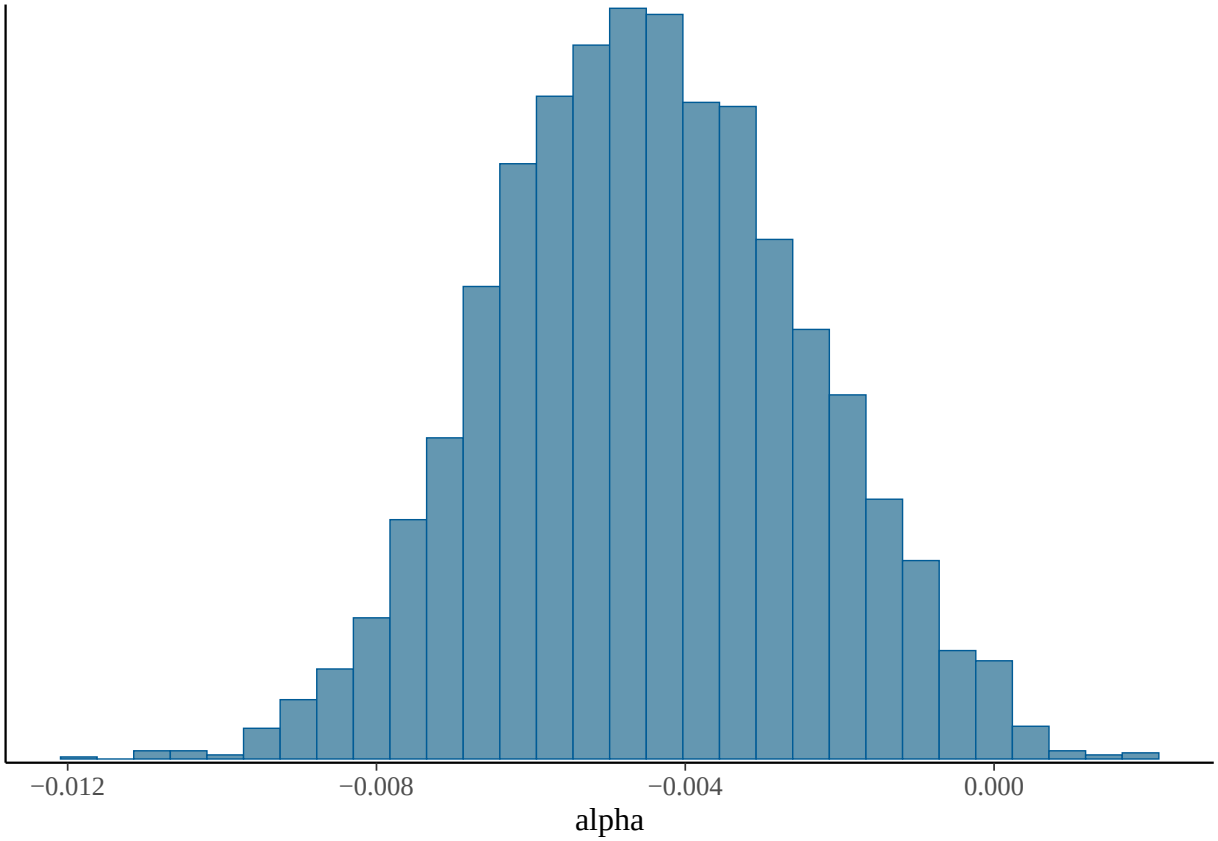



```
print(fit_hier, pars = 'alpha')
```

```
## Inference for Stan model: hier_model_area.
## 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
## alpha    0       0  0 -0.01 -0.01   0   0     0  3310    1
##
## Samples were drawn using NUTS(diag_e) at Sat Dec 12 17:36:53 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 = 'alpha')
```

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



Areas parameters

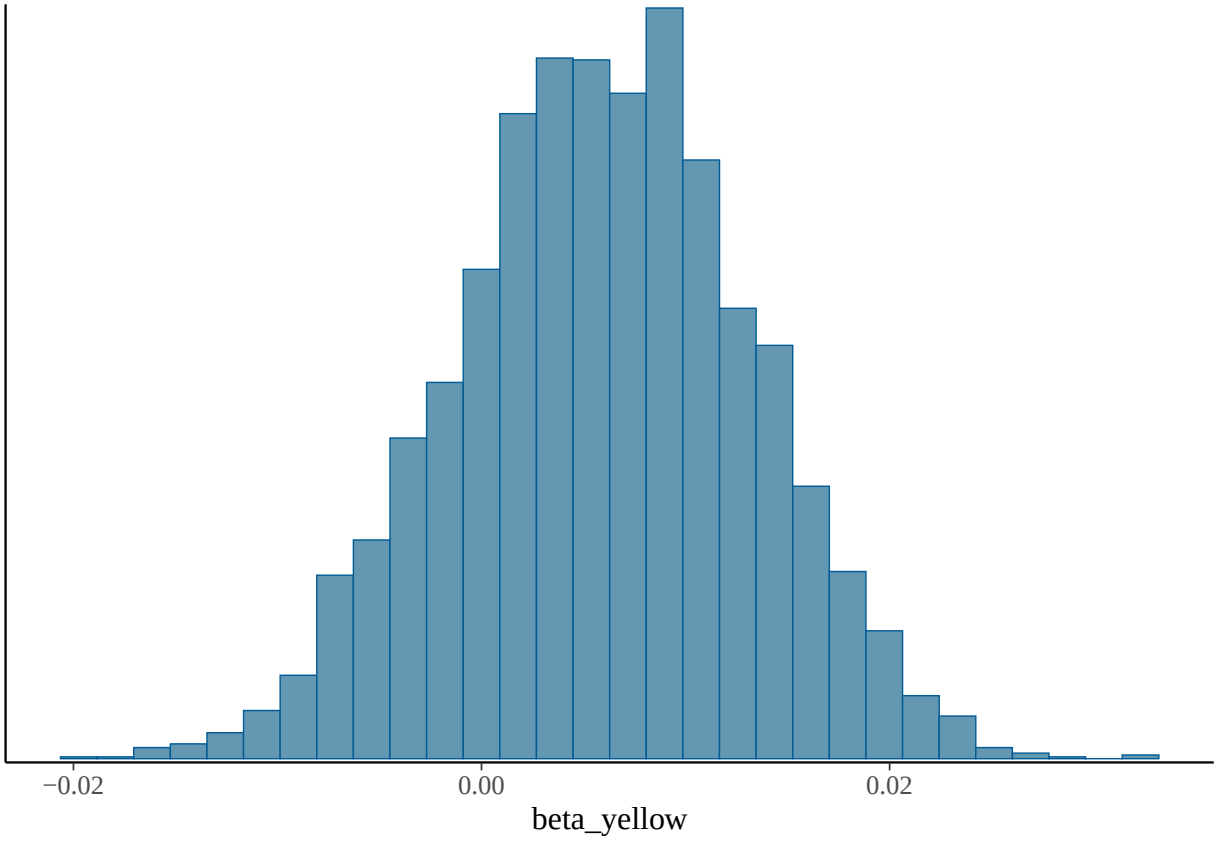
```
print(fit_hier, pars=c('beta_yellow', 'beta_orange', 'beta_red'))
```

```
## Inference for Stan model: hier_model_area.
## 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_yellow 0.01       0 0.01 -0.01  0.00 0.01 0.01  0.02  1801    1
## beta_orange 0.00       0 0.01 -0.01  0.00 0.00 0.01  0.02  1698    1
## beta_red    0.00       0 0.01 -0.02 -0.01 0.00 0.01  0.02  1831    1
```

```
##
## Samples were drawn using NUTS(diag_e) at Sat Dec 12 17:36:53 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).
```

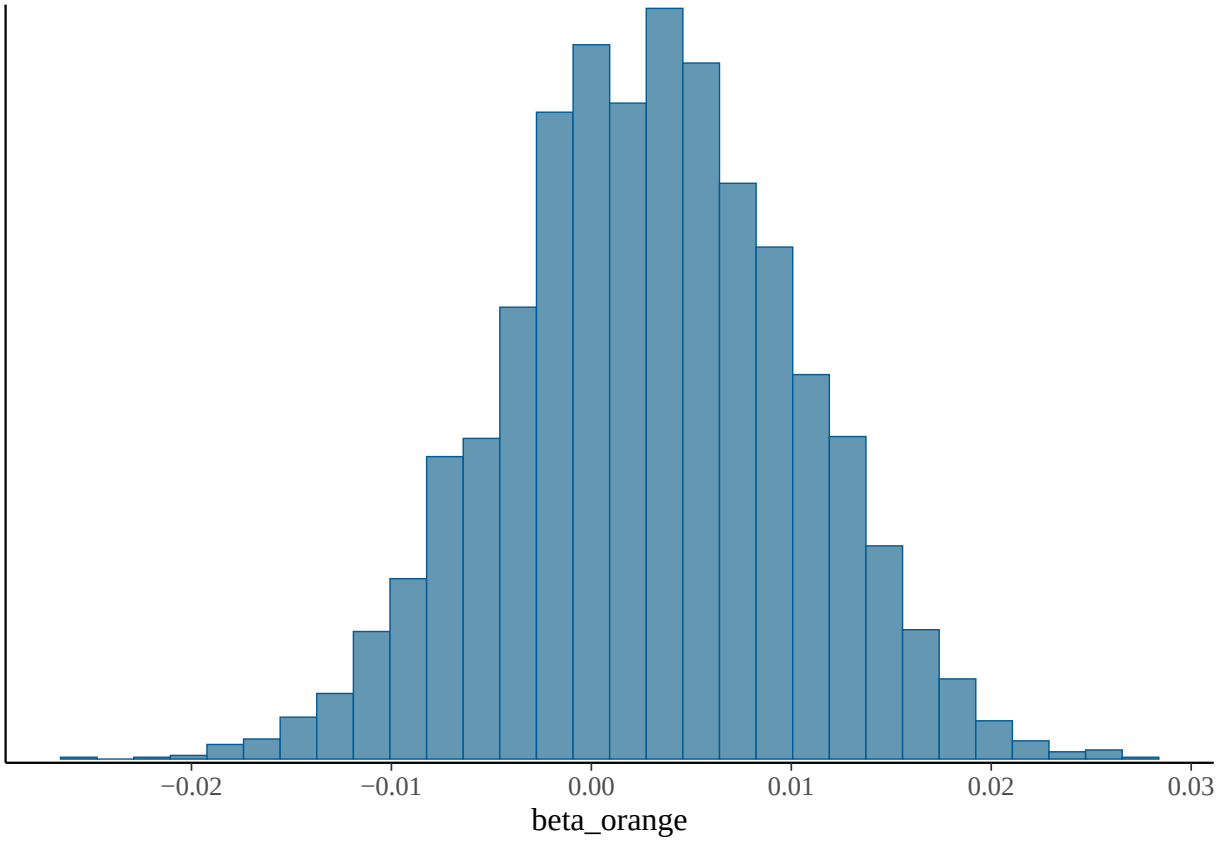
```
mcmc_hist(fit_hier, pars = c('beta_yellow'))
```

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



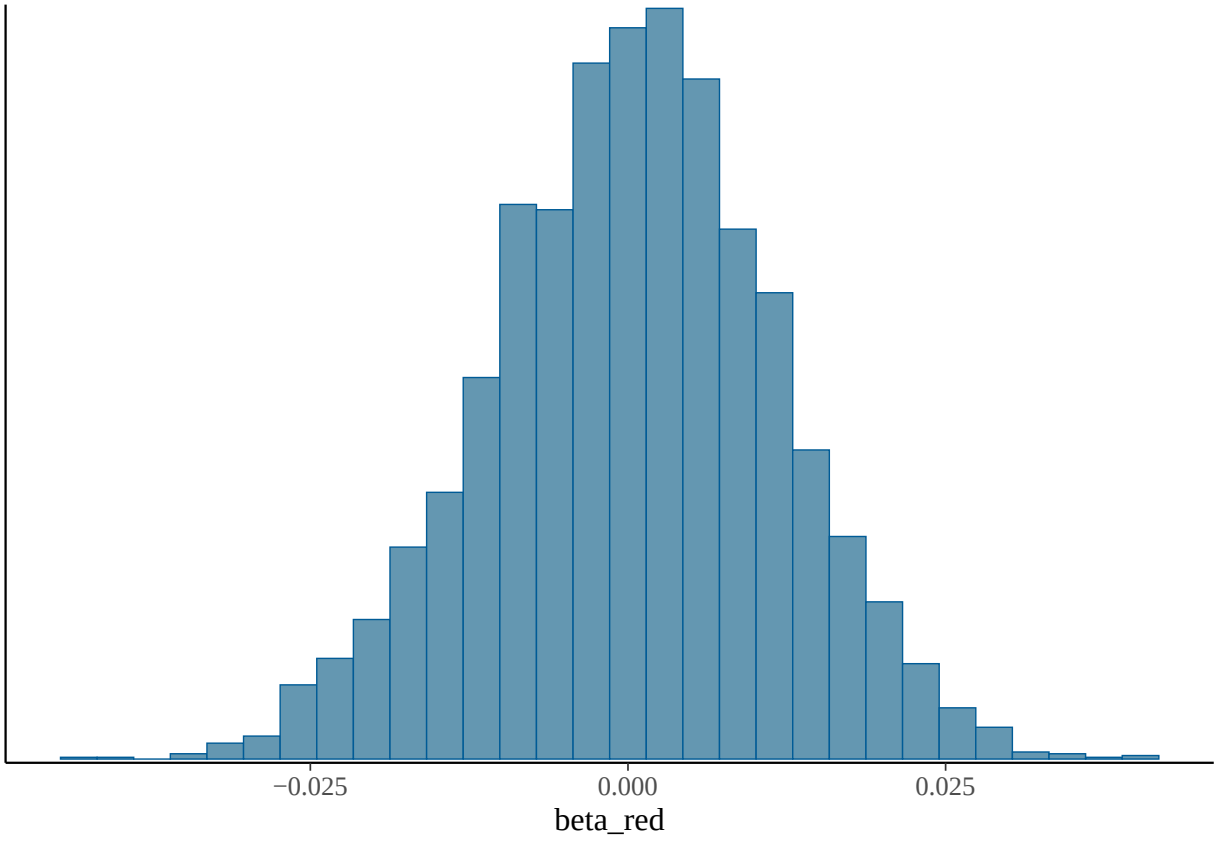

```
mcmc_hist(fit_hier, pars = c('beta_orange'))
```

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



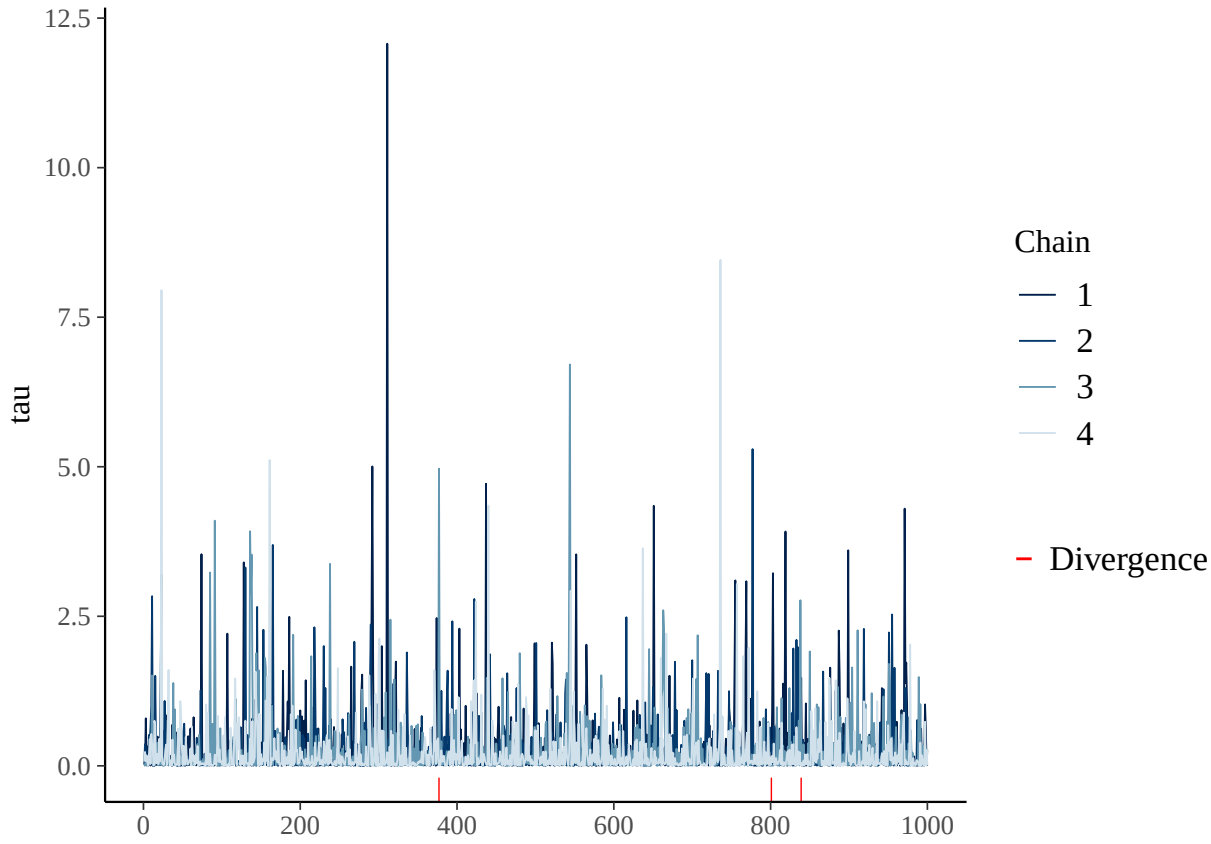

```
mcmc_hist(fit_hier, pars = c('beta_red'))
```

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

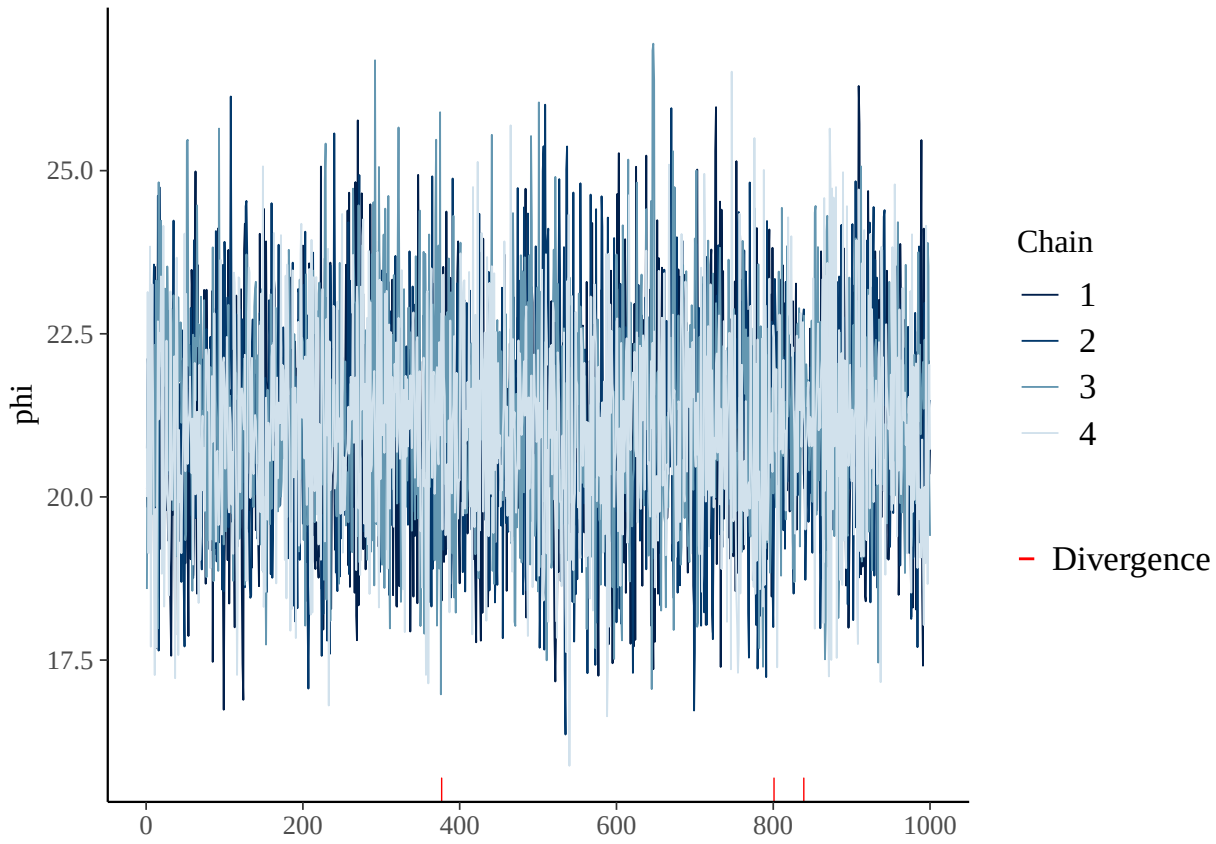


Trace plots

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

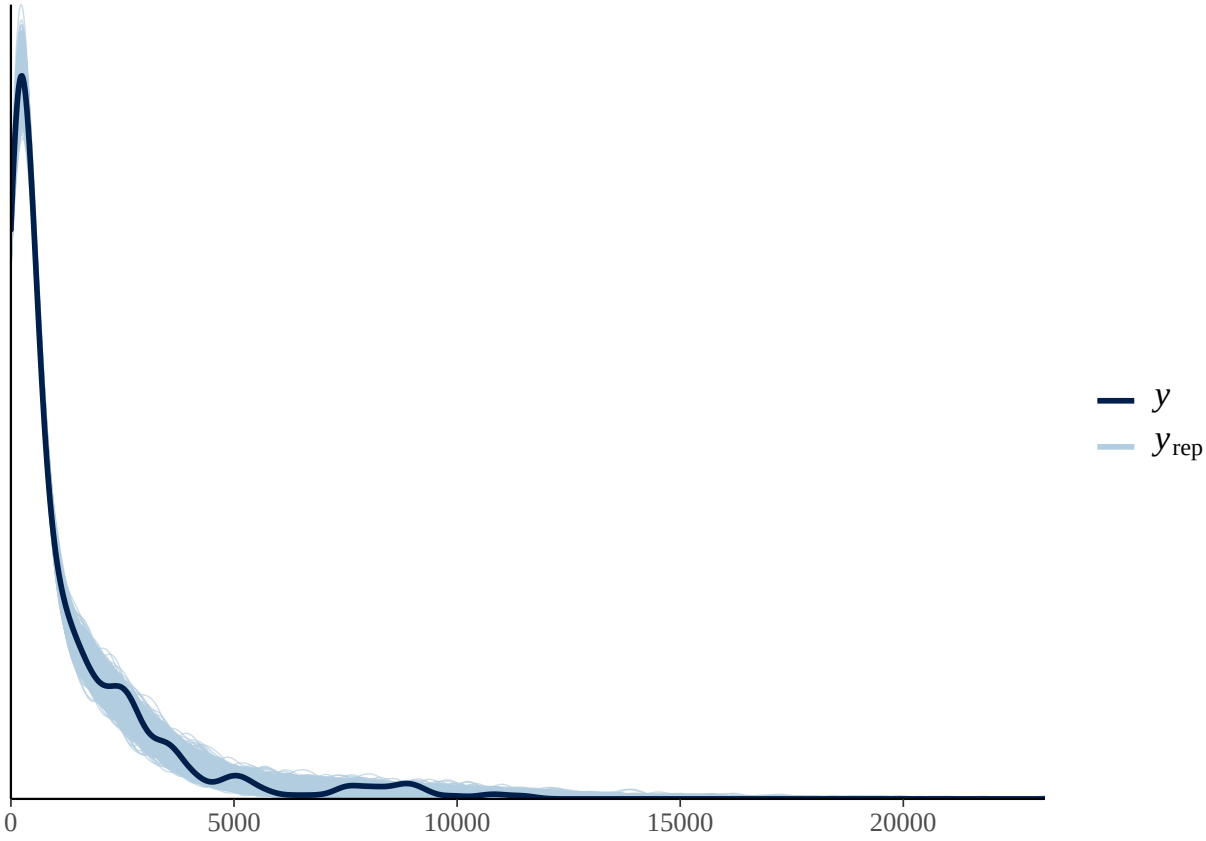



```
mcmc_trace(as.array(fit_hier, pars = c('phi')),
           np = nuts_params(fit_hier)
)
```



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, ])
```



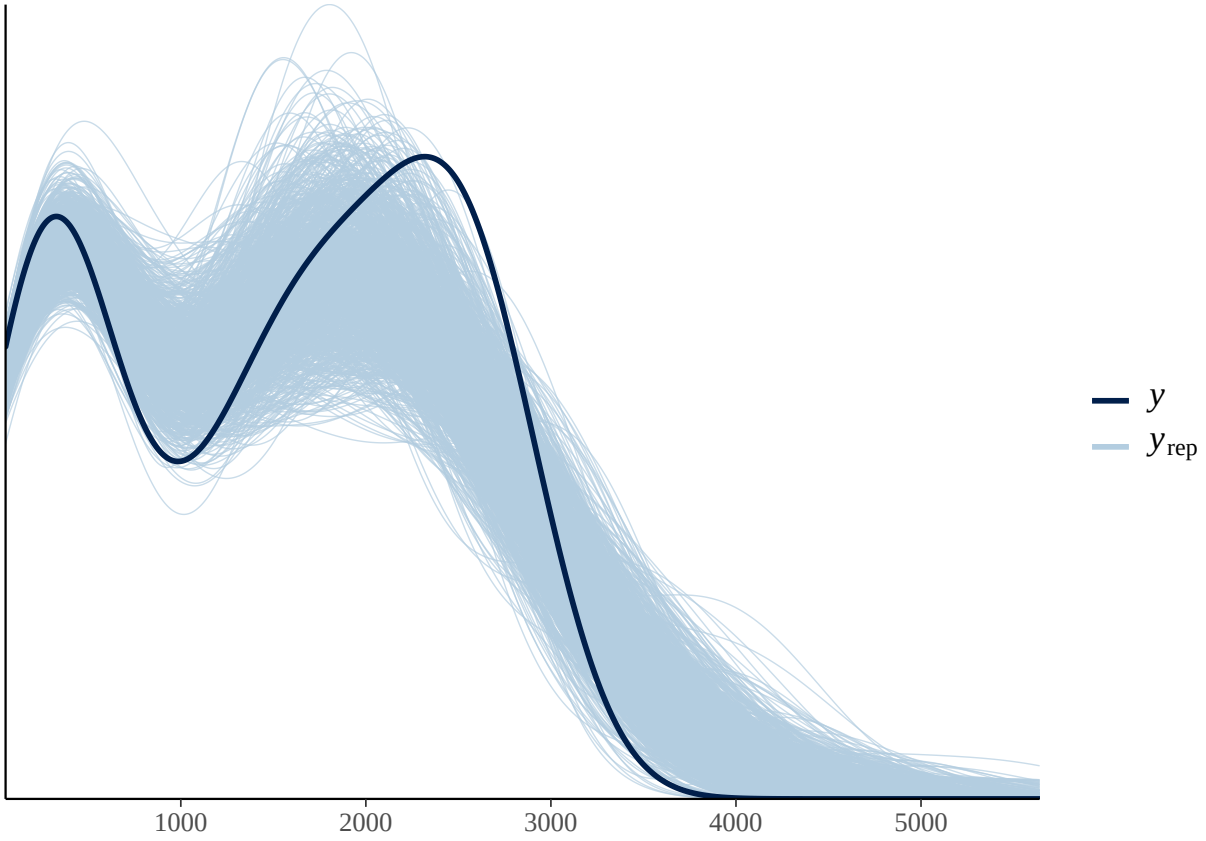
Posterior predictive check by region

```
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)
}
```

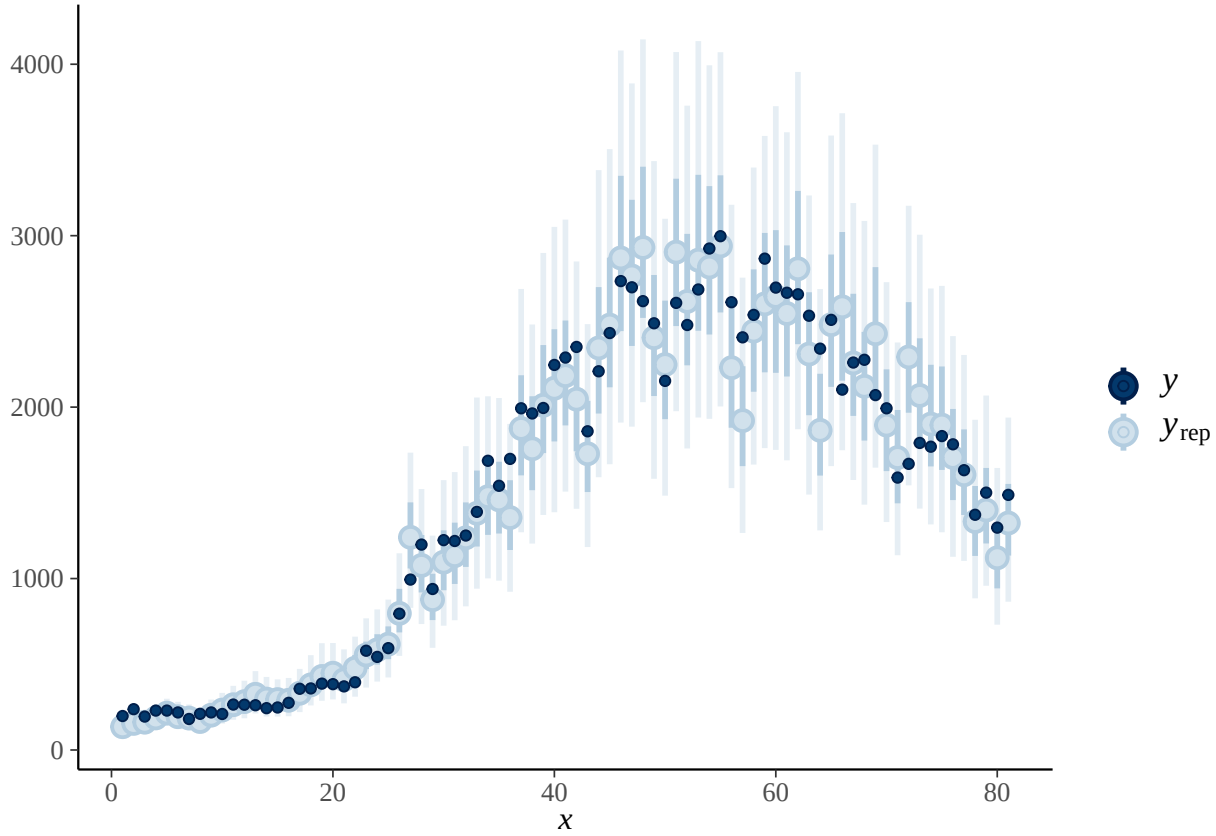
```
groups <- function(regions, nonzero_days){
  group <- rep(regions[1], length(nonzero_days))
  for(r in 2:length(regions))
    group <- c(group, rep(regions[r], length(nonzero_days)))

  return(group)
}
```

Lazio

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

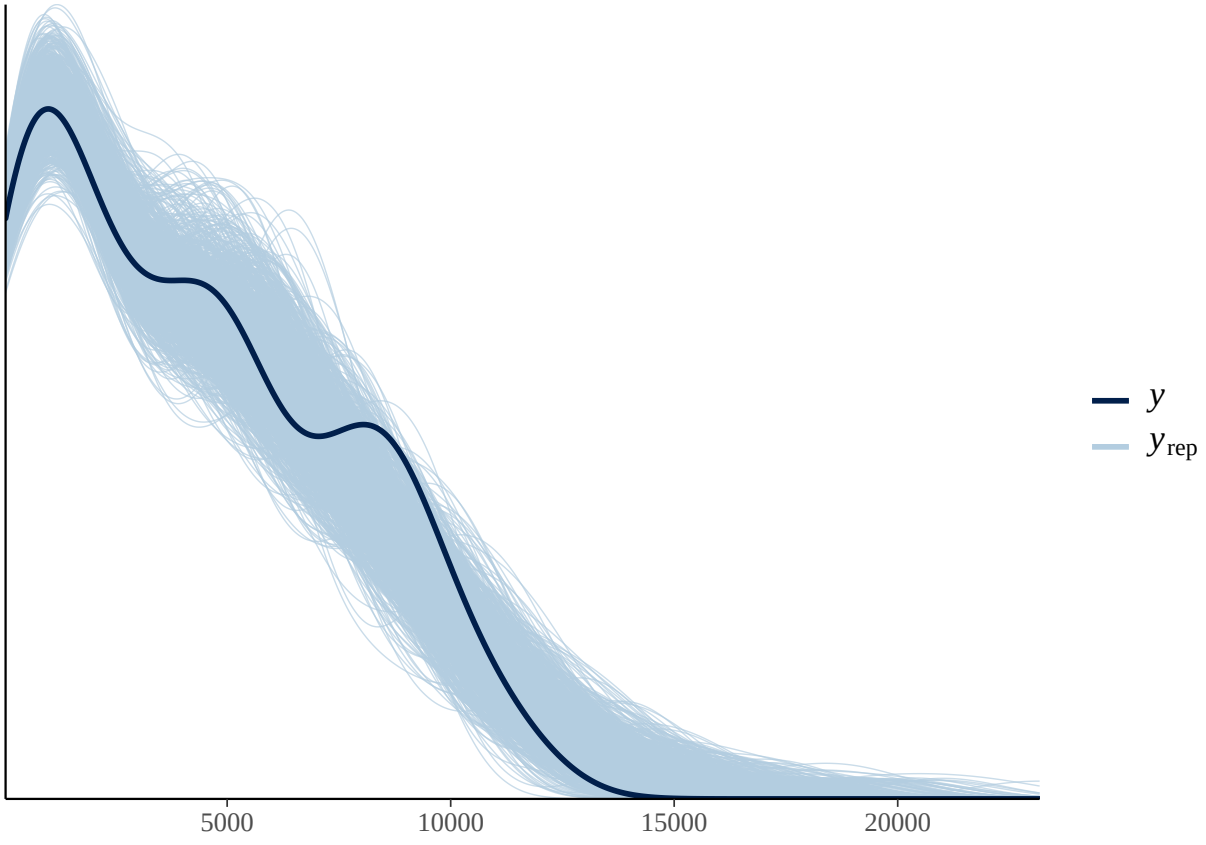



```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Lazio')], y_rep[1:1000,regional_
```

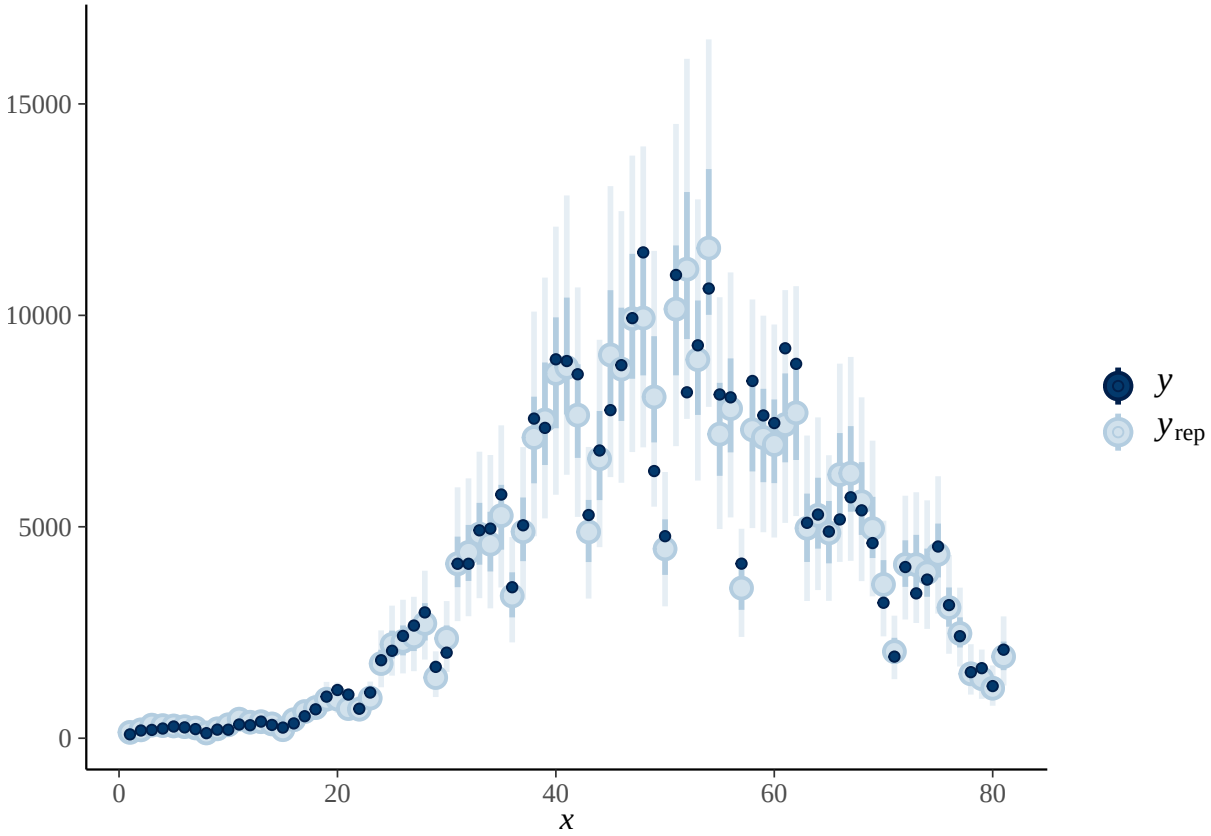


Lombardia

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

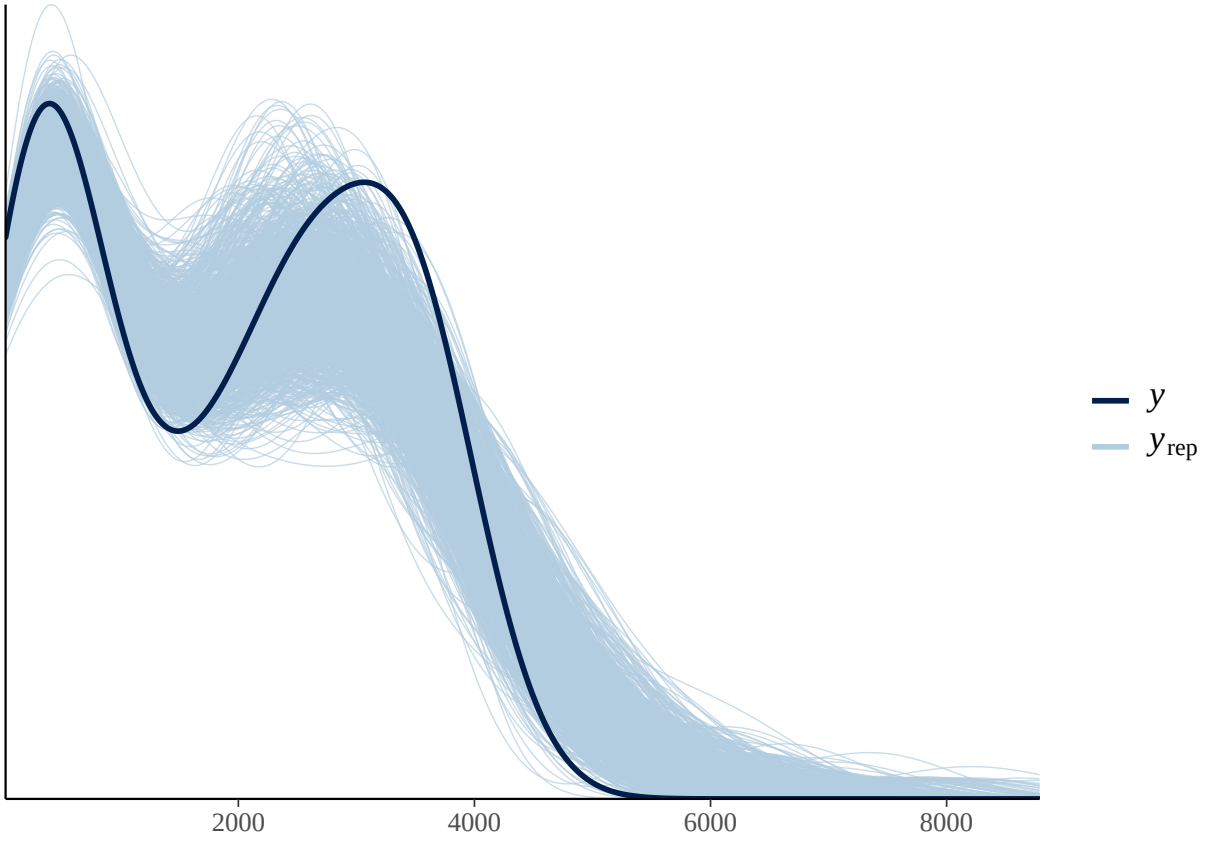



```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Lombardia')], y_rep[1:1000,regio
```

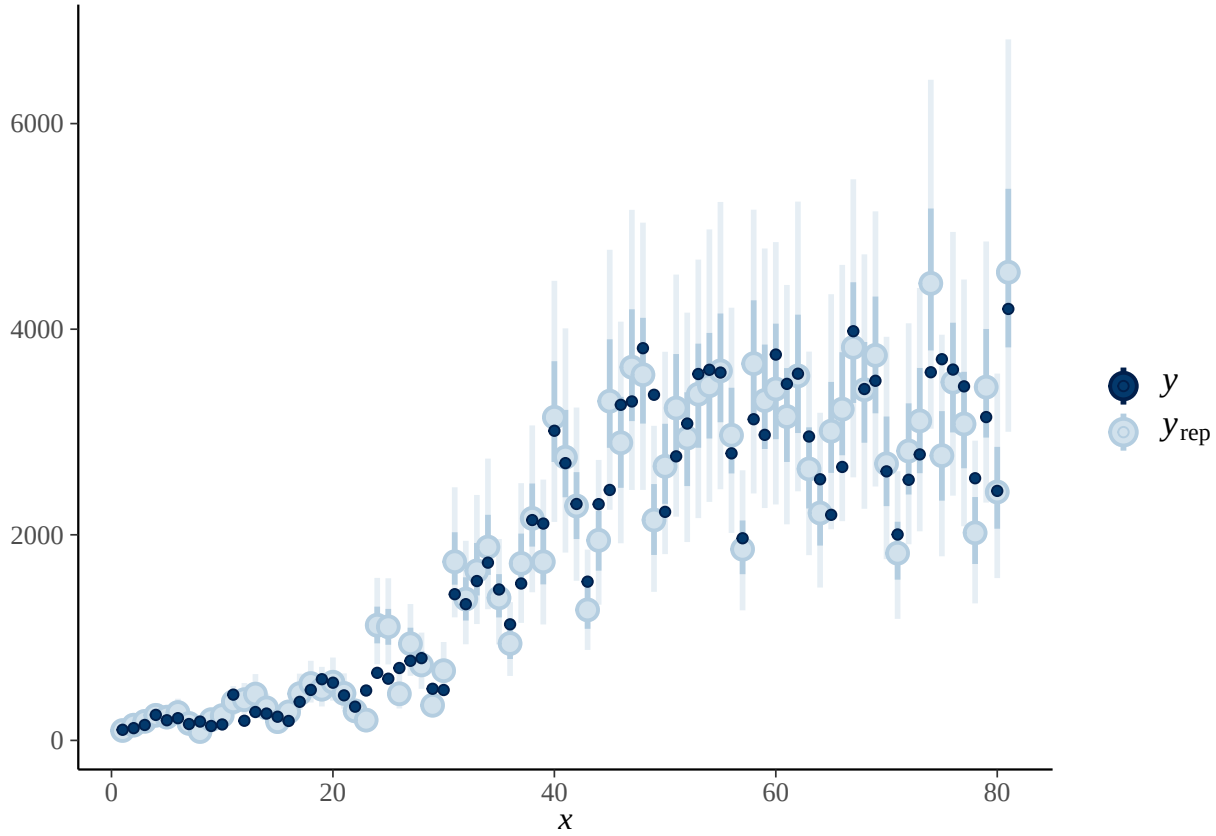


Veneto

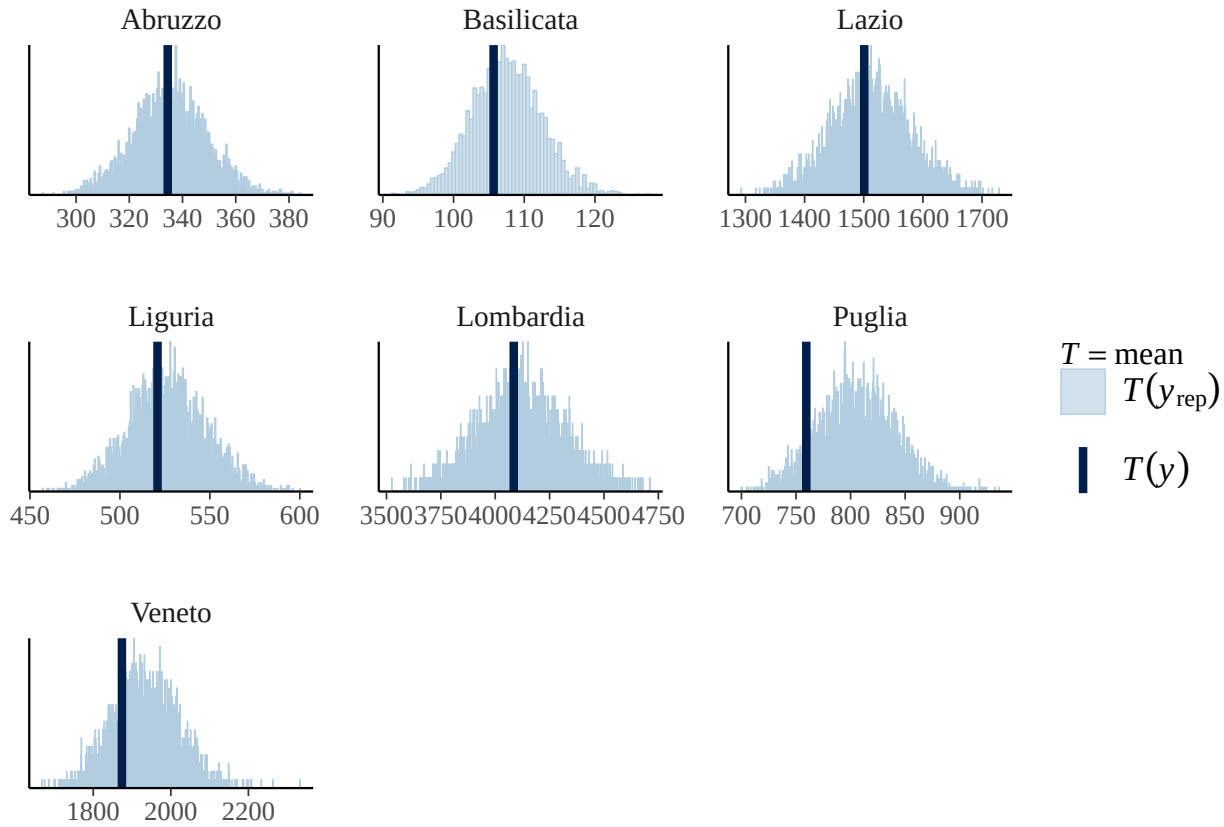
```
ppc_dens_overlay(y = stan_data_hier$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regio
```



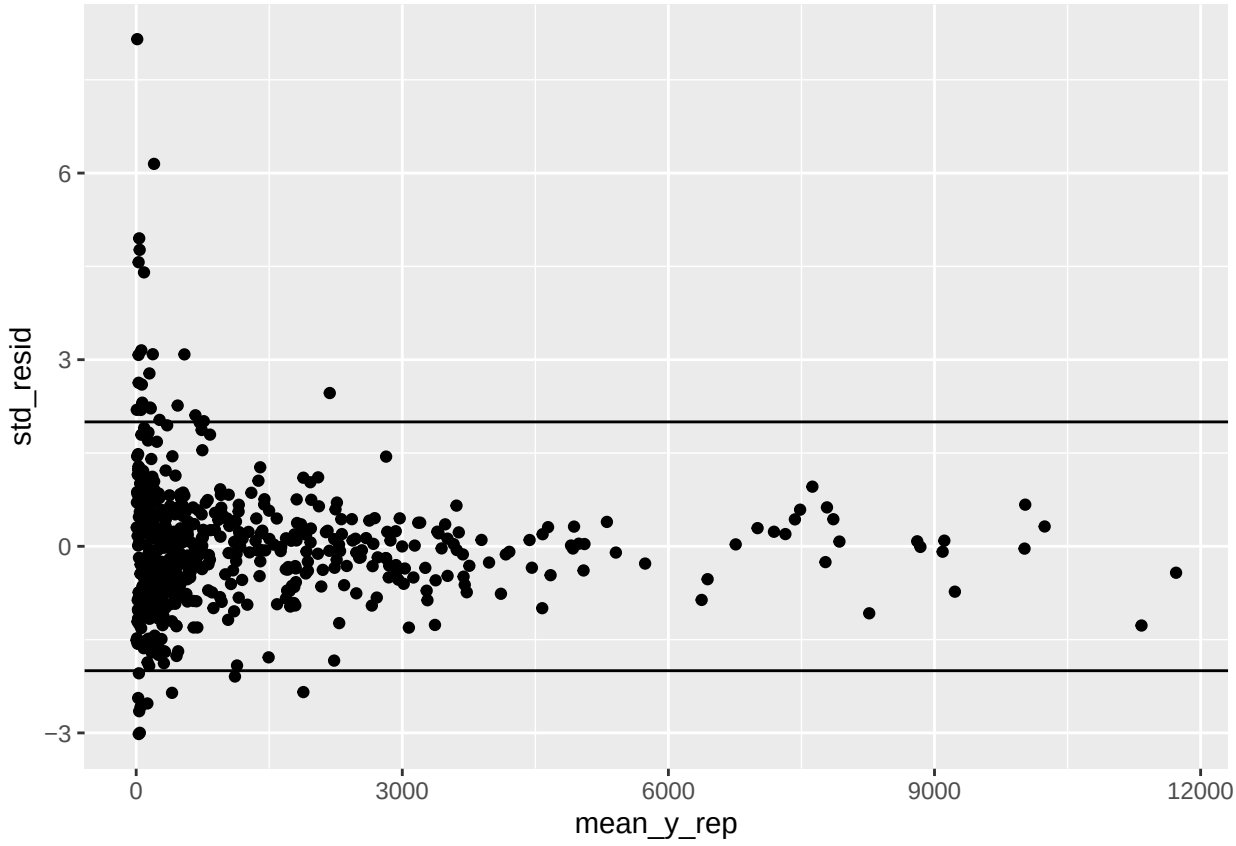

```
ppc_intervals(y = stan_data_hier$nonzero_positives[, which(regions == 'Veneto')], y_rep[1:1000,regional_
```




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

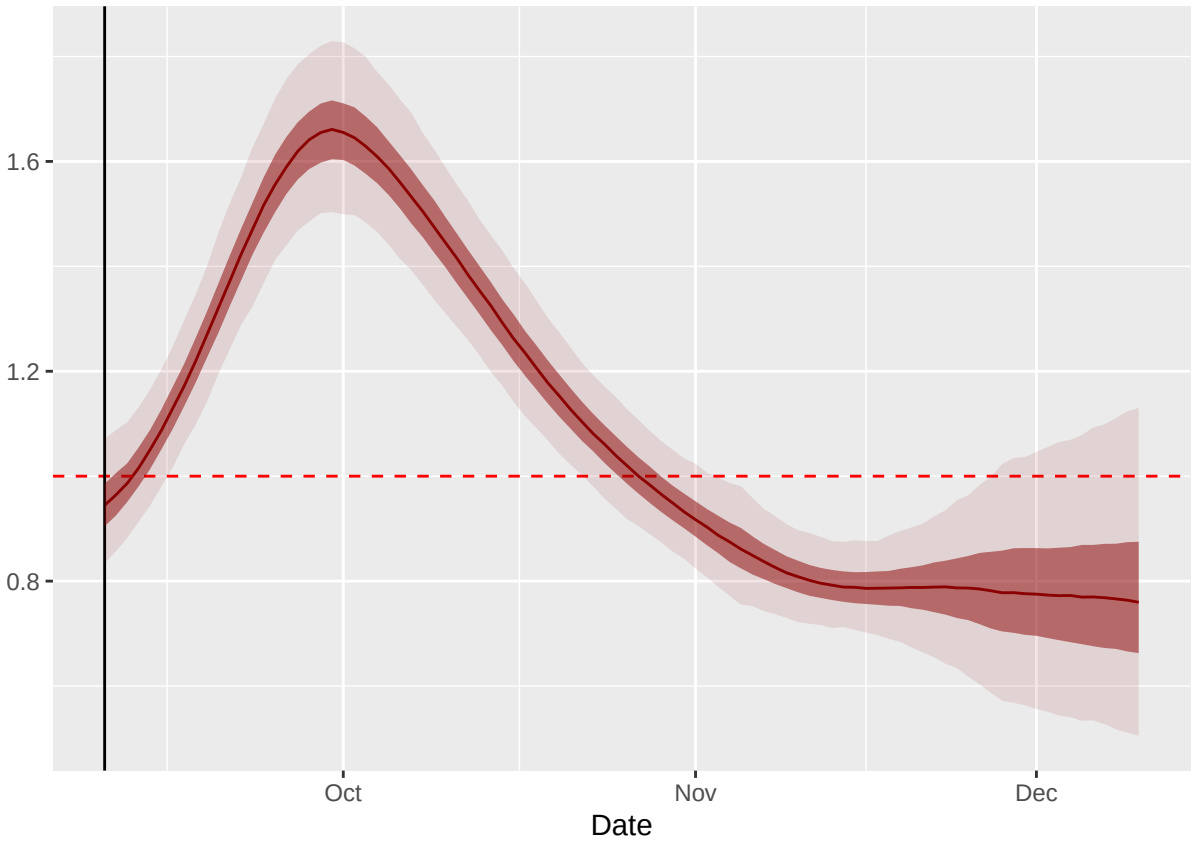


```
mean_inv_phi<-mean(rstan::extract(fit_hier)$inv_phi)
mean_y_rep<-colMeans(y_rep)
std_resid<-(as.vector(stan_data_hier$nonzero_positives)-mean_y_rep)/sqrt(mean_y_rep+mean_y_rep^2*mean_i
qplot(mean_y_rep,std_resid)+hline_at(2)+hline_at(-2)
```



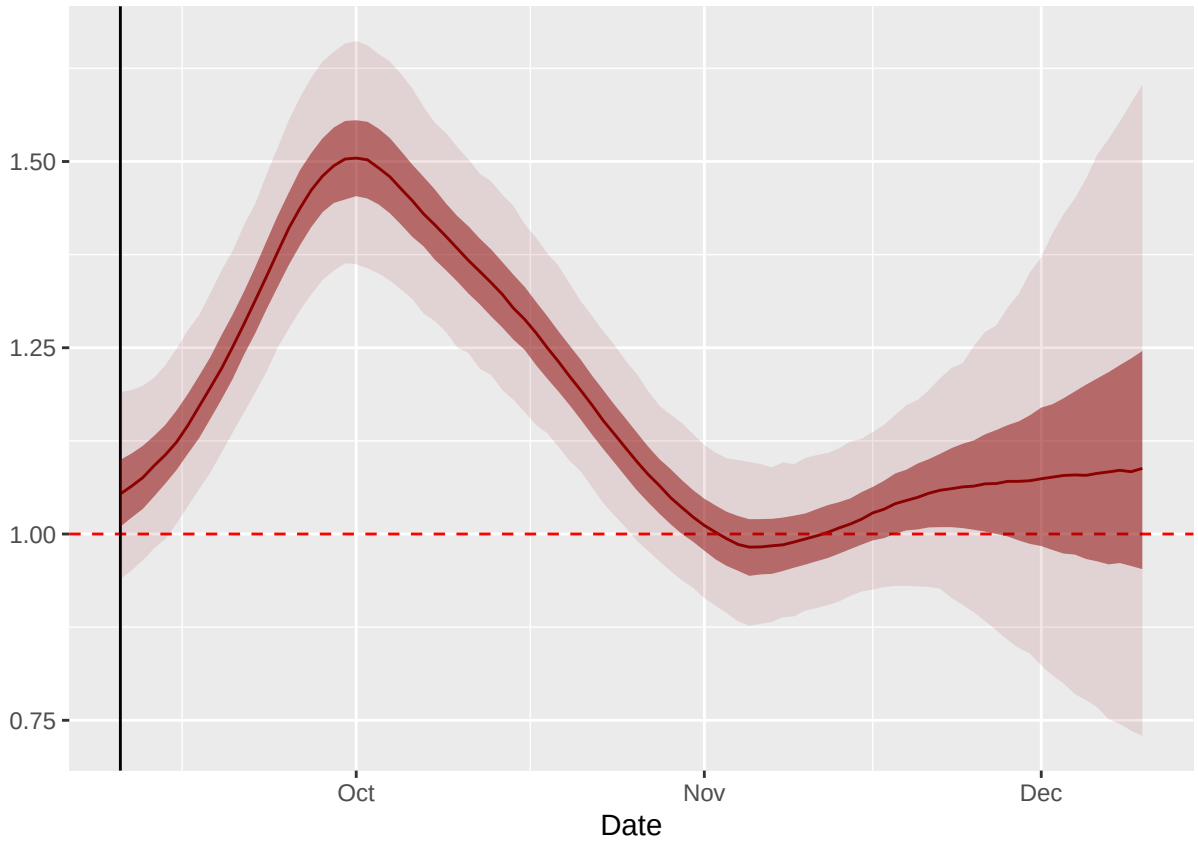
Rt Lombardia

```
plot_rt_hier(hier_data, fit_hier, regions, 'Lombardia')
```



Rt Veneto

```
plot_rt_hier(hier_data, fit_hier, regions, 'Veneto')
```


1.50
1.25
1.00
0.75
Oct
Nov
Dec
Date