Model_comparisons_longData.R

gioia

2020-05-21

```
################################
### Model comparison
################################

library(rstan)
```

```
## Loading required package: StanHeaders
```

```
## Loading required package: ggplot2
```

```
## rstan (Version 2.19.3, GitRev: 2e1f913d3ca3)
```

```
## For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores()).
## To avoid recompilation of unchanged Stan programs, we recommend calling
## rstan_options(auto_write = TRUE)
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'
```

```
## The following objects are masked from 'package:dplyr':
##
##     intersect, setdiff, union
```

```
## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
library(bayesplot)
```

```
## This is bayesplot version 1.7.1
```

```
## - Online documentation and vignettes at mc-stan.org/bayesplot
```

```
## - bayesplot theme set to bayesplot::theme_default()
##    * Does _not_ affect other ggplot2 plots
##    * See ?bayesplot_theme_set for details on theme setting
```

```
library(loo)
```

```
## This is loo version 2.2.0
## - Online documentation and vignettes at mc-stan.org/loo
## - As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use the 'cores' ar
##
## Attaching package: 'loo'
## The following object is masked from 'package:rstan':
##
##     loo
```

```
theme_set(bayesplot::theme_default())

# data
pest_data <- readRDS('data/pest_data_longer_stan_dat.RDS')
str(pest_data)
```

```
## List of 13
##  $ complaints      : num [1:360] 1 0 1 1 0 1 2 0 4 4 ...
##  $ traps           : num [1:360] 7 7 7 7 7 7 6 7 8 8 ...
##  $ pred_mat        : num [1:360, 1:4] 0 0 0 0 0 0 0 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr [1:4] "live_in_super" "" "" ""
##  $ N               : int 360
##  $ K               : num 4
##  $ log_sq_foot     : num [1:360] 1.42 1.42 1.42 1.42 1.42 ...
##  $ mo_idx          : int [1:360] 1 2 3 4 5 6 7 8 9 10 ...
##  $ M               : num 36
##  $ J               : int 10
##  $ building_idx    : int [1:360] 1 1 1 1 1 1 1 1 1 1 ...
##  $ log_sq_foot_pred: num [1:10] 1.42 1.54 1.96 1.56 1.75 ...
##  $ M_forward       : num 12
##  $ building_data   : num [1:10, 1:4] -0.3 -0.3 -0.3 -0.3 0.7 -0.3 -0.3 0.7 -0.3 0.7 ...
##   ..- attr(*, "scaled:center")= Named num [1:4] 0.3 49.4 3687.3 49.9
##   .. ..- attr(*, "names")= chr [1:4] "live_in_super" "age_of_building" "monthly_average_rent" "averag
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:10] "1" "2" "3" "4" ...
##   .. ..$ : chr [1:4] "live_in_super" "age_of_building" "monthly_average_rent" "average_tenant_age"
```

```
N_buildings <- length(unique(pest_data$building_idx))
N_buildings
```

```
## [1] 10
```

```
## Data acquisition

stan_dat_simple <- list(
  N = length(pest_data$complaints),
  complaints = pest_data$complaints,
```

```
  traps = pest_data$traps
)


## Models fit

# simple poisson
comp_model_P <- stan_model('To_compare_models/simple_poisson_regression.stan')
fit_P_real_data <- sampling(comp_model_P, data = stan_dat_simple)
```

```
##
## SAMPLING FOR MODEL 'simple_poisson_regression' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.5 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: 0.211448 seconds (Warm-up)
## Chain 1:                0.183546 seconds (Sampling)
## Chain 1:                0.394994 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'simple_poisson_regression' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 2.1e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.21 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: 0.184326 seconds (Warm-up)
## Chain 2:                0.185845 seconds (Sampling)
## Chain 2:                0.370171 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'simple_poisson_regression' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 2.8e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.28 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: 0.205297 seconds (Warm-up)
## Chain 3:                0.18469 seconds (Sampling)
## Chain 3:                0.389987 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'simple_poisson_regression' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 2.5e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.25 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: 0.182597 seconds (Warm-up)
## Chain 4:                0.182051 seconds (Sampling)
```

```
## Chain 4:                0.364648 seconds (Total)
## Chain 4:
```

```
# multiple poisson
stan_dat_simple$log_sq_foot <- pest_data$log_sq_foot
stan_dat_simple$live_in_super <- pest_data$pred_mat[,"live_in_super"]
comp_model_P_mult <- stan_model('To_compare_models/multiple_poisson_regression.stan')
fit_model_P_mult_real <- sampling(comp_model_P_mult, data = stan_dat_simple)
```

```
##
## SAMPLING FOR MODEL 'multiple_poisson_regression' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 7.3e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.73 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: 0.358808 seconds (Warm-up)
## Chain 1:                0.351982 seconds (Sampling)
## Chain 1:                0.71079 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'multiple_poisson_regression' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 3e-05 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.3 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: 0.398161 seconds (Warm-up)
## Chain 2:                0.34907 seconds (Sampling)
## Chain 2:                0.747231 seconds (Total)
## Chain 2:
##
## SAMPLING FOR MODEL 'multiple_poisson_regression' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 3e-05 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.3 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: 0.411953 seconds (Warm-up)
## Chain 3:                0.355761 seconds (Sampling)
## Chain 3:                0.767714 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'multiple_poisson_regression' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 3e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.3 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: 0.410548 seconds (Warm-up)
## Chain 4:                0.368655 seconds (Sampling)
## Chain 4:                0.779203 seconds (Total)
## Chain 4:
```

```r
# negative binomial
comp_model_NB <- stan_model('To_compare_models/multiple_NB_regression.stan')
fitted_model_NB <- sampling(comp_model_NB, data = stan_dat_simple)
```

```
##
## SAMPLING FOR MODEL 'multiple_NB_regression' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000132 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.32 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: 1.31266 seconds (Warm-up)
## Chain 1:                1.1441 seconds (Sampling)
## Chain 1:                2.45676 seconds (Total)
## Chain 1:
##
## SAMPLING FOR MODEL 'multiple_NB_regression' NOW (CHAIN 2).
## Chain 2:
## Chain 2: Gradient evaluation took 0.00014 seconds
## Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 1.4 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: 1.3163 seconds (Warm-up)
## Chain 2:                1.28008 seconds (Sampling)
## Chain 2:                2.59638 seconds (Total)
## Chain 2:
```

```
##
## SAMPLING FOR MODEL 'multiple_NB_regression' NOW (CHAIN 3).
## Chain 3:
## Chain 3: Gradient evaluation took 0.000236 seconds
## Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.36 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: 1.30571 seconds (Warm-up)
## Chain 3:                1.06747 seconds (Sampling)
## Chain 3:                2.37318 seconds (Total)
## Chain 3:
##
## SAMPLING FOR MODEL 'multiple_NB_regression' NOW (CHAIN 4).
## Chain 4:
## Chain 4: Gradient evaluation took 9e-05 seconds
## Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.9 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: 1.40379 seconds (Warm-up)
## Chain 4:                1.16407 seconds (Sampling)
## Chain 4:                2.56786 seconds (Total)
## Chain 4:
```

```
stan_dat_hier <- pest_data
# hier NB regression
comp_model_NB_hier <- stan_model('To_compare_models/hier_NB_regression.stan')
```

```
fitted_model_NB_hier <-
  sampling(
    comp_model_NB_hier,
    data = stan_dat_hier,
    chains = 4,
    cores = 4,
    iter = 4000
  )

# hier NCP NB regression
comp_model_NB_hier_ncp <- stan_model('To_compare_models/hier_NB_regression_ncp.stan')
fitted_model_NB_hier_ncp <- sampling(comp_model_NB_hier_ncp,
                                     data = stan_dat_hier,
                                     chains = 4,
                                     cores = 4,
                                     iter=4000)

# hier NB slopes
comp_model_NB_hier_slopes <- stan_model('To_compare_models/hier_NB_regression_ncp_slopes_mod.stan')
fitted_model_NB_hier_slopes <-
  sampling(
    comp_model_NB_hier_slopes,
    data = stan_dat_hier,
    chains = 4, cores = 4,
    control = list(adapt_delta = 0.95)
  )
```

```
## Warning: There were 4 divergent transitions after warmup. Increasing adapt_delta above 0.95 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
```

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

```
# hier NB slopes
comp_model_NB_hier_slopes_mos <- stan_model('To_compare_models/hier_NB_regression_ncp_slopes_mod_mos.sta
fitted_model_NB_hier_slopes_mos <-
  sampling(
    comp_model_NB_hier_slopes_mos,
    data = stan_dat_hier,
    chains = 4, cores = 4,
    control = list(adapt_delta = 0.95)
  )
```

```
## Warning: There were 3 divergent transitions after warmup. Increasing adapt_delta above 0.95 may help
## http://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup

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

```
##

library(shinystan)
```

```
## Loading required package: shiny
```

```
##
## This is shinystan version 2.5.0
```

```
y <- pest_data$complaints
launch_shinystan(fitted_model_NB_hier_slopes)
```

```
##
## Launching ShinyStan interface... for large models this  may take some time.
```

```
##
## Listening on http://127.0.0.1:4148
```

```
## Extract pointwise log-likelihood and
## compute loo and waic

# loo
log_lik_pois <- extract_log_lik(fit_P_real_data)
loo_pois <- loo(log_lik_pois)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
waic_pois <- waic(log_lik_pois)
```

```
## Warning:
## 11 (3.1%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_mult_pois <- extract_log_lik(fit_model_P_mult_real)
loo_mult_pois <- loo(log_lik_mult_pois)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic') for details.
```

```
waic_mult_pois <- waic(log_lik_mult_pois)
```

```
## Warning:
## 19 (5.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_mult_NB <- extract_log_lik(fitted_model_NB)
loo_mult_NB <- loo(log_lik_mult_NB)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
waic_mult_NB <- waic(log_lik_mult_NB)
```

```
## Warning:
## 1 (0.3%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_hier_NB <- extract_log_lik(fitted_model_NB_hier)
loo_hier_NB <- loo(log_lik_hier_NB)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
waic_hier_NB <- waic(log_lik_hier_NB)
```

```
## Warning:
## 2 (0.6%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_hier_NB_ncp <- extract_log_lik(fitted_model_NB_hier_ncp)
loo_hier_NB_ncp <- loo(log_lik_hier_NB_ncp)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
waic_hier_NB_ncp <- waic(log_lik_hier_NB_ncp)
```

```
## Warning:
## 2 (0.6%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_slopes <- extract_log_lik(fitted_model_NB_hier_slopes)
loo_slopes <- loo(log_lik_slopes)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
waic_slopes <- waic(log_lik_slopes)
```

```
## Warning:
## 4 (1.1%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

```
log_lik_mos <- extract_log_lik(fitted_model_NB_hier_slopes_mos)
loo_mos <- loo(log_lik_mos)
```

```
## Warning: Relative effective sample sizes ('r_eff' argument) not specified.
## For models fit with MCMC, the reported PSIS effective sample sizes and
## MCSE estimates will be over-optimistic.
```

```
## Warning: Some Pareto k diagnostic values are slightly high. See help('pareto-k-diagnostic') for deta
```

```
waic_mos <- waic(log_lik_mos)
```

```
## Warning:
## 25 (6.9%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

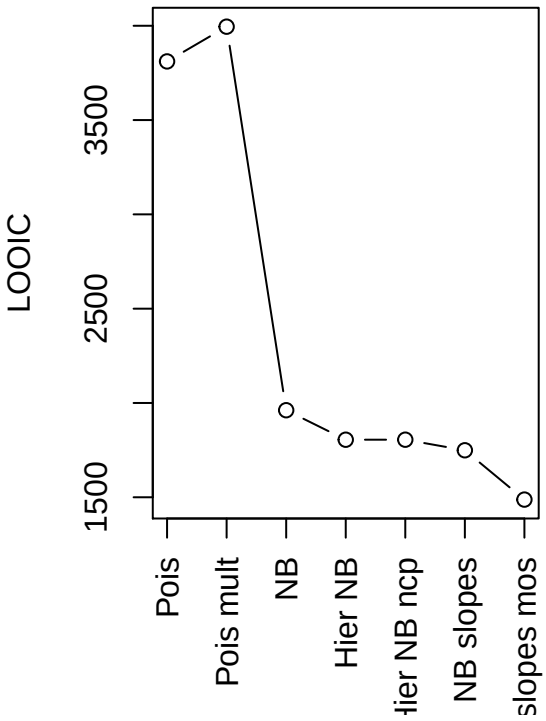
```
compare(loo_pois, loo_mult_pois,
        loo_mult_NB, loo_hier_NB,
        loo_hier_NB_ncp, loo_slopes,
        loo_mos)
```

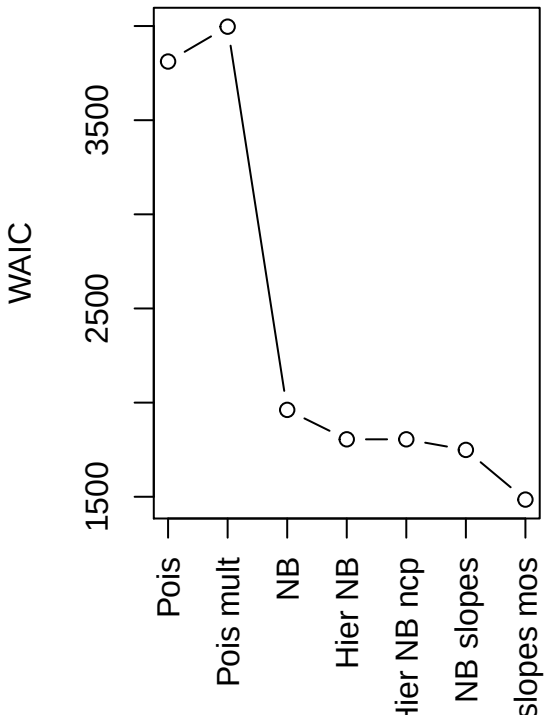
```
## Warning: 'compare' is deprecated.
## Use 'loo_compare' instead.
## See help("Deprecated")
```

```
##                 elpd_diff se_diff elpd_loo p_loo   looic
## loo_mos             0.0       0.0  -743.9     42.6  1487.8
## loo_slopes       -130.7      12.6  -874.6     16.8  1749.2
## loo_hier_NB      -158.7      13.2  -902.6     10.5  1805.2
## loo_hier_NB_ncp  -158.7      13.2  -902.6     10.5  1805.3
## loo_mult_NB      -236.9      16.2  -980.8      4.7  1961.5
```

```
## loo_pois        -1161.6     129.6 -1905.5     27.4  3811.0
## loo_mult_pois   -1253.7     152.7 -1997.6     39.1  3995.1
```

```
looic1<-loo_pois$estimates[3,1]
looic2<-loo_mult_pois$estimates[3,1]
looic3<-loo_mult_NB$estimates[3,1]
looic4<-loo_hier_NB$estimates[3,1]
looic5<-loo_hier_NB_ncp$estimates[3,1]
looic6<-loo_slopes$estimates[3,1]
looic7<-loo_mos$estimates[3,1]

looics <-c(looic1, looic2, looic3, looic4, looic5,
           looic6, looic7)

waic1<-waic_pois$estimates[3,1]
waic2<-waic_mult_pois$estimates[3,1]
waic3<-waic_mult_NB$estimates[3,1]
waic4<-waic_hier_NB$estimates[3,1]
waic5<-waic_hier_NB_ncp$estimates[3,1]
waic6<-waic_slopes$estimates[3,1]
waic7<-waic_mos$estimates[3,1]

waics <-c(waic1, waic2, waic3, waic4, waic5, waic6,
           waic7)

par(xaxt="n", mfrow=c(1,2))
plot(looics, type="b", xlab="", ylab="LOOIC")
par(xaxt="s")
axis(1, c(1:7), c("Pois", "Pois mult", "NB",
                  "Hier NB", "Hier NB ncp", "NB slopes","NB slopes mos"), las=2)
par(xaxt="n")
plot(waics, type="b", xlab="", ylab="WAIC")
par(xaxt="s")
axis(1, c(1:7), c("Pois", "Pois mult", "NB",
                  "Hier NB", "Hier NB ncp", "NB slopes","NB slopes mos"), las=2)
```


LOOIC
1500
2500
3500
Pois
Pois mult
NB
Hier NB
lier NB ncp
NB slopes
slopes mos


WAIC
1500
2500
3500
Pois
Pois mult
NB
Hier NB
lier NB ncp
NB slopes
slopes mos