Bayesian Statistics: Laboratory 6

Vincenzo Gioia

DEAMS

University of Trieste

vincenzo.gioia@units.it

Building D, room 2.13

Office hour: Friday, 15 - 17

19/05/2023

Outline

1. Cockroaches' example: Recap from the previous Labs
2. Negative Binomial hierarchical regression model with varying intercept, slope and time varying-effect
3. Model Comparison

Section 1

Cockroaches' example: Recap from the previous Labs

Recap from the previous Labs

- During the last labs we implemented several regression models for analysing the relationship between the number of coackroaches complaints and the number of traps considering
  - Poisson distribution
    - Model 1: Including only the covariate `traps`
    - Model 2: Adding to Model 1 the covariate `super` and the offset `sqfoot`
  - Negative Binomial (accounting for the overdispersion)
    - Model 3: Same linear predictor of Model 2
    - Model 4/5: Hierarchical model with varying intercept (CP/NCP)
    - Model 6: Hierarchical model with varying intercept and slope (NCP and on an extended version of the data)

Load packages and data

- Here, we directly load the longer version of the dataset

```
library(rstan)
library(loo)
library(bayesplot)
theme_set(bayesplot::theme_default())
set.seed(123)
stan_dat_hier_long <- readRDS('pest_data_long.RDS')
```

Exercise

- While thereafter we will use the extended version of the dataset, a good exercise would be to consider the smaller dataset and repeat the analysis addressing the model comparison. Maybe, you will discover some differences comparing the models

NB hier. model with varying intercept and slope

- Recall the specification of the last model we implemented

$$
\begin{aligned}
\textit{complaints}_{b,t} &\sim \text{Neg} - \text{Binomial}(\lambda_{b,t}, \phi) \\
\lambda_{b,t} &= \exp\left(\eta_{b,t}\right) \\
\eta_{b,t} &= \mu_b + \kappa_b \, \textit{traps}_{b,t} + \textit{log}(\textit{sqfoot})_i \\
\mu_b &\sim \mathcal{N}(\alpha + \text{building\_data}_b \zeta, \sigma_\mu) \\
\kappa_b &\sim \mathcal{N}(\beta + \text{building\_data}_b \gamma, \sigma_\kappa) \\
\alpha &\sim \mathcal{N}(\log(4), 1) \\
\zeta_k &\sim \mathcal{N}(0, 1), \quad k = 1, \ldots, 4 \\
\sigma_\mu &\sim \mathcal{TN}(0, 1, 0, +\infty) \\
\beta &\sim \mathcal{N}(-0.25, 1) \\
\gamma_k &\sim \mathcal{N}(0, 1), \quad k = 1, \ldots, 4 \\
\sigma_\kappa &\sim \mathcal{TN}(0, 1, 0, +\infty) \\
\phi^{-1} &\sim \mathcal{TN}(0, 1, 0, +\infty)
\end{aligned}
$$

NB hier. model with varying intercept and slope

- Compile

```
comp_model_NB_hier_slopes <- stan_model('hier_NB_regression_ncp_slopes.stan')
```

- Sampling

```
fit_NB_hier_slopes <- sampling(
    comp_model_NB_hier_slopes,
    data = stan_dat_hier_long,
    refresh = 0,
    control = list(adapt_delta = 0.95))
```

```
## Warning: There were 4 divergent transitions after warmup. See
## https://mc-stan.org/misc/warnings.html#divergent-transitions-after-warmup
## to find out why this is a problem and how to eliminate them.

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

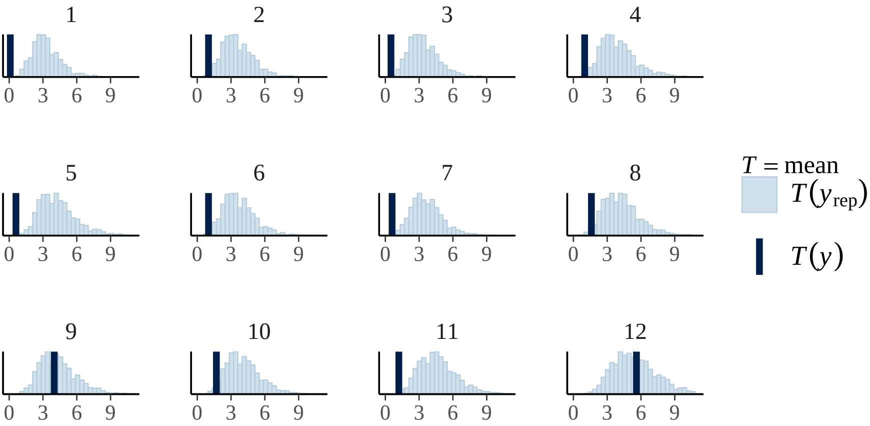
NB hier. model with varying intercept and slope - PPCs

- We have not yet explored how the number complaints vary as function of the time
- Take a look at whether there is any pattern over time by analysing the PPCs by month
- Here, we restrict the check to only the first year of data

```
y_rep <- as.matrix(fit_NB_hier_slopes, pars = "y_rep")
sel_1year <- which(stan_dat_hier_long$mo_idx %in% 1:12)
with(stan_dat_hier_long, ppc_stat_grouped(
  y = complaints[sel_1year],
  yrep = y_rep[, sel_1year],
  group = mo_idx[sel_1year],
  stat = 'mean'
) + xlim(0, 11))
```

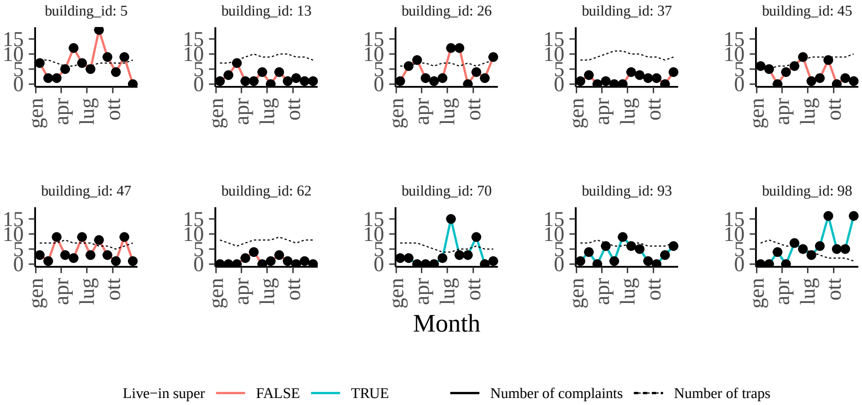
NB hier. model with varying intercept and slope - PPCs

- We are missing the variation over time for the number of complaints



Time series of complaints and traps for each building

- Several competing factors can be related to the change over time of the number of complaints. There might be more roaches in the environment during the summer, but also more roach control in the summer as well



Section 2

Negative Binomial hierarchical regression model with varying intercept, slope and time varying-effect

NBH: varying intercept/slope and time varying-effect

- In addition, maybe after a first sighting of roaches in a building, residents are more vigilant and the number complaints could increase

- We can expand the model including a (log-) additive monthly effect, mo_t, that is

$$\eta_{b,t} = \mu_b + \kappa_b \, traps_{b,t} + mo_t + log(sqfoot)_b$$

- We specify the following autoregressive (of order 1) prior structure for our monthly effects

$$mo_t \sim \mathcal{N}(\rho \, mo_{t-1}, \sigma_{mo})$$

Equivalently

$$mo_t = \rho \, mo_{t-1} + \epsilon_t, \quad \epsilon_t \sim \mathcal{N}(0, \sigma_{mo}) \quad \rho \in [-1, 1]$$

NBH: varying intercept/slope and time varying-effect

- Using the stationary assumption of the AR models, we get the marginal distribution for $\texttt{mo}_1$

- Marginal variance: by independence of ϵ_{t-1} and ϵ_t and by stationarity

$$\mathrm{Var}(mo_t) = \mathrm{Var}(\rho mo_{t-1}) + Var(\epsilon_t) = \rho^2 \mathrm{Var}(mo_t) + \sigma^2_{mo} \implies \mathrm{Var}(mo_t) = \frac{\sigma^2_{mo}}{1-\rho^2}$$

- Marginal mean (for $\rho \neq 1$)

$$\mathbb{E}(mo_t) = \mathbb{E}(\rho mo_{t-1}) + \mathbb{E}(\epsilon_t) = \frac{0}{1-\rho} = 0$$

- Finally,

$$mo_1 \sim \mathcal{N}\left(0, \frac{\sigma_{mo}}{\sqrt{1-\rho^2}}\right)$$
$$mo_t \sim \mathcal{N}(\rho\, mo_{t-1}, \sigma_{mo})$$

NBH: varying intercept/slope and time varying-effect

- There is a problem in Stan for specifying the prior for the autoregressive parameter, ρ, because Stan does not implement densities on $[-1, 1]$

- Thus, we overcome the problem by using a variable transformation

- That is, we define a variable `rho_raw` ($\tilde{\rho}$) defined in $[0, 1]$ and we transform it to get a density on $[-1, 1]$

$$\tilde{\rho} \in [0, 1] \qquad \rho = 2 \times \tilde{\rho} - 1$$

- There could be positive or negative association between months, but there should be a bit more weight placed on positive ρ. Thus, we could consider $\tilde{\rho} \sim \mathrm{Beta}(10, 5)$, that is an informative prior pushing the parameter towards positive estimate of ρ

NBH: varying intercept/slope and time varying-effect

- Take a look to the Stan file `hier_NB_regression_ncp_slopes_mos.stan`
- Compile

```
comp_model_NB_hier_mos <- stan_model('hier_NB_regression_ncp_slopes_mos.stan')
```

- Sampling

```
fit_NB_hier_mos <- sampling(comp_model_NB_hier_mos,
                            data = stan_dat_hier_long,
                            refresh = 0,
                            control = list(adapt_delta = 0.95))
```

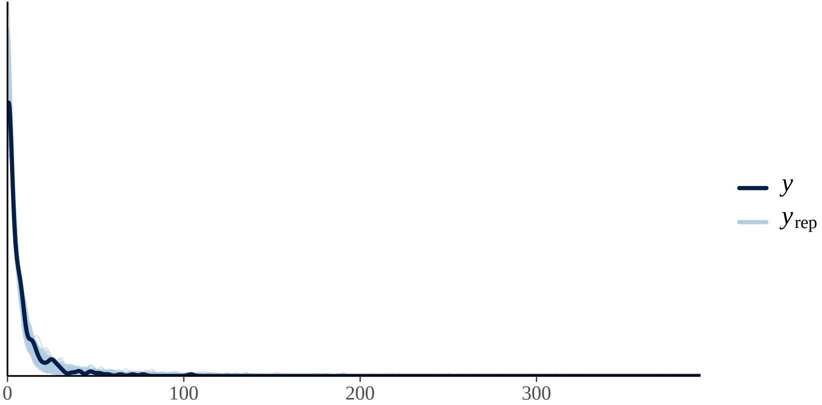
NBH: varying intercept/slope and time varying-effect

- We do not expand further the model, although we could
- Rather, we run through our PPCs

```
y_rep2 <- as.matrix(fit_NB_hier_mos, pars = "y_rep")
ppc_dens_overlay(
  y = stan_dat_hier_long$complaints,
  yrep = y_rep[1 : 200,]
)
```

NBH: varying intercept/slope + time varying-effect - PPCs

- It looks OK, with no large difference w.r.t. the previous model

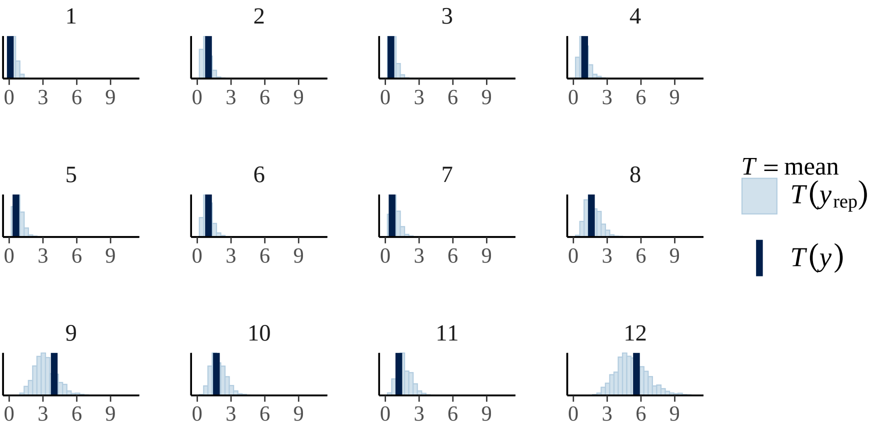


- Again, PPCs by month

```r
sel_1year <- which(stan_dat_hier_long$mo_idx %in% 1 : 12)
with(stan_dat_hier_long,
     ppc_stat_grouped(
     y = complaints[sel_1year],
     yrep = y_rep2[, sel_1year],
     group = mo_idx[sel_1year],
     stat = 'mean') + xlim(0, 11))
```

NBH: varying intercept/slope + time varying-effect - PPCs

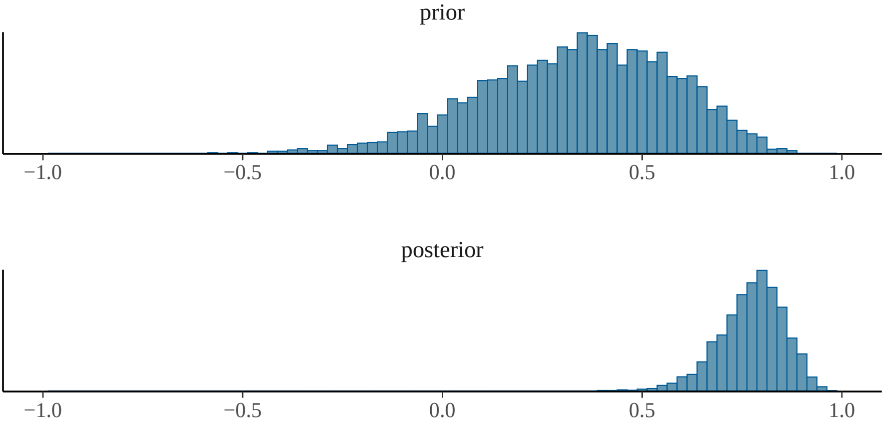
- As we can see, our monthly random intercept has captured a monthly pattern across all the buildings



- We can also compare the prior and posterior for the autoregressive parameter to see how much we have learned. Here, we compare draws from prior and draws from posterior

```
rho_draws <- cbind(
  2 * rbeta(4000, 10, 5) - 1, # draw from prior
  as.matrix(fit_NB_hier_mos, pars = "rho")
)
colnames(rho_draws) <- c("prior", "posterior")
mcmc_hist(rho_draws, freq = FALSE, binwidth = 0.025,
          facet_args = list(nrow = 2)) + xlim(-1, 1)
```

NBH: varying intercept/slope + time varying-effect - PPCs



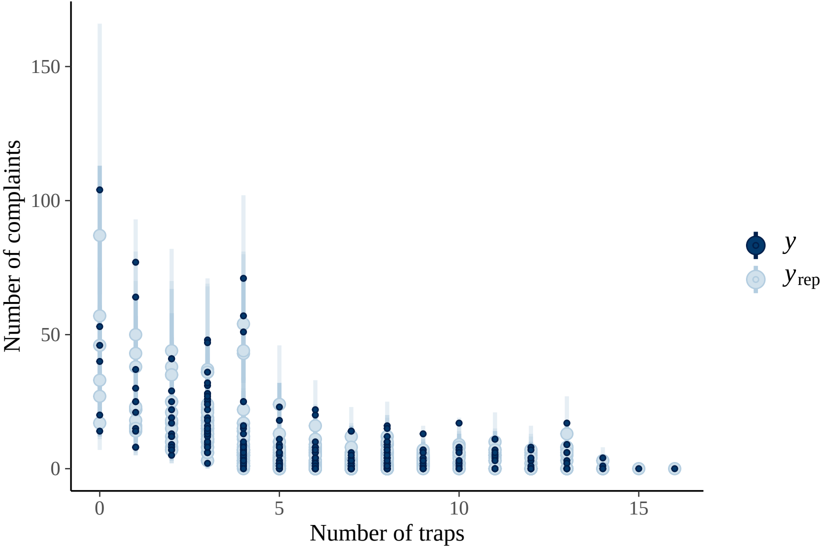
NBH: varying intercept/slope + time varying-effect - PPCs

- Plot of predictions by number of bait stations, with uncertainty intervals

```
with(stan_dat_hier_long, ppc_intervals(
  y = complaints,
  yrep = y_rep2,
  x = traps) +
  labs(x = "Number of traps", y = "Number of complaints"))
```

NBH: varying intercept/slope + time varying-effect - PPCs

- Overall, the model seems to capture the data



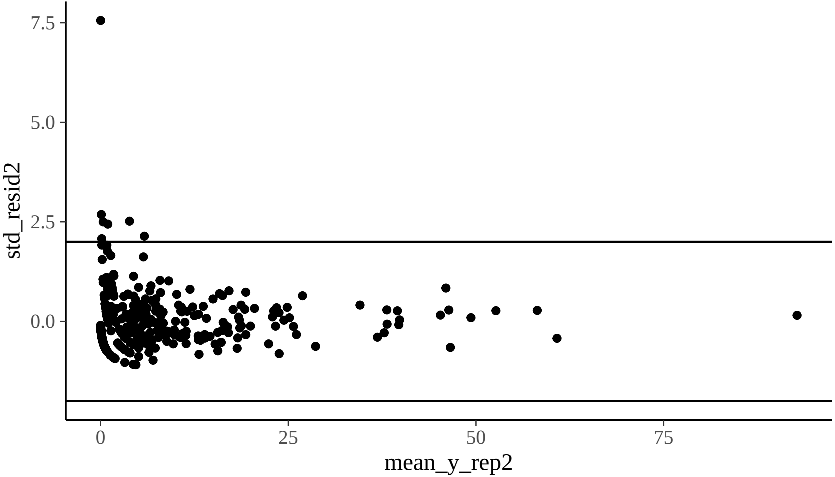
NBH: varying intercept/slope + time varying-effect - PPCs

- Standardised residuals:

```
mean_y_rep2 <- colMeans(y_rep2)
mean_inv_phi2 <- mean(as.matrix(fit_NB_hier_slopes,
                                pars = "inv_phi"))
std_resid2 <- (stan_dat_hier_long$complaints - mean_y_rep2) /
  sqrt(mean_y_rep2 + mean_y_rep2^2*mean_inv_phi2)
ggplot() +
  geom_point(mapping = aes(x = mean_y_rep2, y = std_resid2)) +
  geom_hline(yintercept = c(-2,2))
```

NBH: varying intercept/slope + time varying-effect - PPCs

- Only one observation seems not to be well captured: try to discover which is and investigate the source
  for this large residual. Do we miss a source of information in the model?



Section 3

Model Comparison

Model Comparison

- We built several models of increasing complexity: it is the turn to compare them

- We compare the models leveraging predictive information criteria (IC)

$$\text{crit} = -2\widehat{\text{elpd}} = -2(\widehat{\text{lpd}} - \text{parameters penalty})$$

  - $\widehat{\text{lpd}}$ is a measure of the log predictive density of the fitted model: **computed log pointwise predictive density**

$$\widehat{\text{lppd}} = \sum_{i=1}^{n} \log \left(\frac{1}{S} \sum_{s=1}^{S} p(y_i|\theta^{(s)}) \right)$$

    with $\theta^{(s)},\ s = 1, \dots, S$ are the draws from $\pi(\theta|y)$

  - **parameters penalty** is a penalization accounting for the effective number of parameters of the fitted model

- Lower is the value for an IC, and the better is the model fit

Model comparison

- Several well known IC belongs to this class of predictive IC, such as AIC, BIC and DIC
- Here, we will see the Watanabe AIC (WAIC) and the leave-one-out (LOO) cross-validation
- The purpose of using LOO or WAIC is to estimate the **pointwise out-of-sample prediction accuracy** from a fitted Bayesian model using the log-likelihood evaluated at the posterior simulations of the parameter values
- For more details, see https://link.springer.com/article/10.1007/s11222-013-9416-2/ and https://link.springer.com/article/10.1007/s11222-016-9696-4

Model comparison: WAIC

- The WAIC takes the form

$$\mathrm{WAIC} = -2(\widehat{\mathsf{lppd}} + 2\mathrm{p}_{\mathrm{WAIC}})$$

  with

$$p_{\mathrm{WAIC}} = \sum_{i=1}^{n} \mathrm{Var}_{\theta|\mathrm{y}}(\mathsf{log}(\mathrm{p}(\mathrm{y_i}|\theta)),$$

  which computes the variance separately for each data point. We can practically compute this quantity by using the sample variance

- Compared to AIC and DIC, WAIC has the desirable property of **averaging over the posterior distribution** rather than conditioning on a point estimate

Model comparison: LOO

- Exact CV requires re-fitting the model with different training sets, while approximate leave-one-out CV (LOO) can be computed easily using importance sampling
- Drawback: resulting estimate is noisy, as the variance of the importance weights can be large or even infinite
- The LOO can be improved using Pareto smoothed importance sampling (PSIS), which applies a smoothing procedure to the importance weights, obtaining $\omega_i^{(s)}$
- The PSIS estimate of the LOO elpd determines the LOOIC criteria:

$$\mathrm{LOOIC} = -2 \sum_{i=1}^{n} \log \left\{ \left(\sum_{s=1}^{S} \omega_i^{(s)} p(y_i|\theta^{(s)}) \right) / \sum_{s=1}^{S} \omega_i^{(s)} \right\}$$

- Here p_{LOO} is not need for computing the LOOIC, but has diagnostic value; it can be computed as the difference between elpd_{LOO} and the non-cross-validated lpd

Model Comparison

- At first, we need to recover the `super` variable and include in the extended dataset

```
pest_data <- readRDS('pest_data.RDS')
lis <- pest_data$live_in_super[seq(1, 120, by  = 12)]
stan_dat_hier_long$super <- rep(lis, each = 36)
```

- Otherwise, we have problems fitting the multiple Poisson an Negative Binomial regression models

Model Comparison

- Compile and sampling

```
stan_model <- c("simple_poisson_regression.stan",
                "multiple_poisson_regression.stan",
                "multiple_NB_regression.stan",
                "hier_NB_regression.stan",
                "hier_NB_regression_ncp.stan",
                "hier_NB_regression_ncp_slopes.stan",
                "hier_NB_regression_ncp_slopes_mos.stan")
no_model <- length(stan_model)

comp_model <- fit <- list()
set.seed(2)
for(j in 1 : no_model){
  comp_model[[j]] <- stan_model(stan_model[[j]])
  fit[[j]] <- sampling(comp_model[[j]],
                       data = stan_dat_hier_long,
                       control = list(adapt_delta = 0.95))
  print(j)
}
```

Model comparison

- The **loo** R package provides the functions `loo()` and `waic()`, which allow computing PSIS-LOO and WAIC for fitted Bayesian models

- These functions take in argument the fitted model, extracting the $S \times N$ matrix that include the log-likelihood contribution, with S the number of retained draws and N the number of data points

- The `loo()` function returns PSIS-LOOIC and p_{LOO}, while the `waic()` function returns the WAIC and p_{WAIC}

Model comparison

- We need to compute and store the pointwise log-likelihood in Stan
- The model does not change, the only part to change is the **generated quantities** block, where we include a `log_lik` vector (of size N) and in the for loop we store the log-likelihood contributions. For the last model, it corresponds to

```
generated quantities {
  int y_rep[N];
  vector[N] log_lik;
  real eta_n;
  for (n in 1:N) {
    eta_n = mu[building_idx[n]] +
            kappa[building_idx[n]] * traps[n] +
            mo[mo_idx[n]] + log_sqfoot[n];
    y_rep[n] = neg_binomial_2_log_safe_rng(eta_n, phi);
    log_lik[n] = neg_binomial_2_log_lpmf(complaints[n] | eta_n, phi);
  }
}
```

Model comparison

- Extract pointwise log-likelihood, compute the LOO and WAIC and compare the models

```
log_lik <- loo_mod <- waic_mod <- list()
for(j in 1 : no_model){
  log_lik[[j]] <- extract_log_lik(fit[[j]])
  loo_mod[[j]] <- loo(log_lik[[j]])
  waic_mod[[j]] <- waic(log_lik[[j]])
}
loo_compare(loo_mod)
loo_compare(waic_mod)
```

Model comparison

```
loo_compare(loo_mod)
```

```
##        elpd_diff se_diff
## model7     0.0       0.0
## model6  -130.3      12.6
## model5  -158.3      13.3
## model4  -158.4      13.2
## model3  -236.3      16.3
## model1 -1160.9     129.6
## model2 -1254.0     153.0
```

```
loo_compare(waic_mod)
```

```
##        elpd_diff se_diff
## model7     0.0       0.0
## model6  -131.4      12.5
## model5  -159.5      13.2
## model4  -159.6      13.2
## model3  -237.6      16.3
## model1 -1162.2     129.6
## model2 -1255.8     153.3
```

Model comparison

- The estimated shape parameter $\hat{k}$ of the generalized Pareto distribution can be used to assess the reliability of the estimate:

- $\hat{k} \leq 1/2$: the variance of the raw importance ratios is finite, the central limit theorem holds, and the estimate converges quickly

- $\hat{k} > 1/2$: the variance of the PSIS estimate is finite but may be large

```
loo_mod[[7]]
```

```
##
## Computed from 4000 by 360 log-likelihood matrix
##
##          Estimate   SE
## elpd_loo   -744.3 26.8
## p_loo        43.0  3.9
## looic      1488.6 53.7
## ------
## Monte Carlo SE of elpd_loo is NA.
##
## Pareto k diagnostic values:
##                          Count Pct.    Min. n_eff
## (-Inf, 0.5]   (good)     351   97.5%   468
##  (0.5, 0.7]   (ok)         8    2.2%   221
##    (0.7, 1]   (bad)        1    0.3%   674
##    (1, Inf)   (very bad)   0    0.0%   <NA>
## See help('pareto-k-diagnostic') for details.
```

Model comparison

```
waic_mod[[7]]
```

```
##
## Computed from 4000 by 360 log-likelihood matrix
##
##           Estimate   SE
## elpd_waic   -743.0 26.7
## p_waic        41.7  3.8
## waic        1486.0 53.5
##
## 23 (6.4%) p_waic estimates greater than 0.4. We recommend trying loo instead.
```

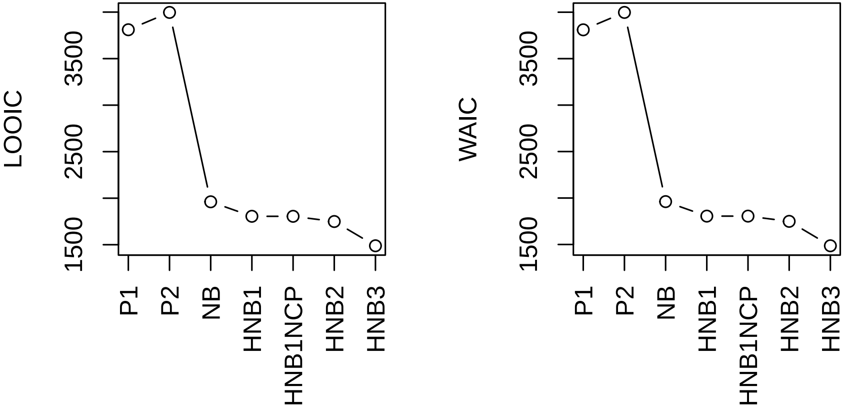
Model comparison

```
looic <- waic <- list()
for(j in 1 : no_model){
  looic[[j]] <- loo_mod[[j]]$estimates[3,1]
  waic[[j]] <- waic_mod[[j]]$estimates[3,1]
}


looics <- unlist(looic)
waics <- unlist(waic)
mod_names <- c("P1", "P2", "NB", "HNB1", "HNB1NCP", "HNB2","HNB3" )
par(xaxt="n", mfrow=c(1,2))
plot(looics, type="b", xlab = "", ylab = "LOOIC")
par(xaxt="s")
axis(1, 1:7, mod_names, las=2)

par(xaxt="n")
plot(waics, type="b", xlab="", ylab="WAIC")
par(xaxt="s")
axis(1, 1:7, mod_names, las = 2)
```

Model comparison



A shiny app

- A useful way to explore several aspects of your fitting by means of a shiny app

```
library(shinystan)
launch_shinystan(fit[[4]])
```