1 Crowdsourced Annotation Model

There exist several applications where the complexity and volume of the task requires crowdsourcing to a large set of human labelers. Inferring the true label of an item from the labels assigned by several labelers is facilitated by this crowdsourced annotation model[1]. The model takes into consideration the random assignment of labelers to items as well as their individual imperfections in the form of confusion matrices. The model is shown in plate notation in figure1

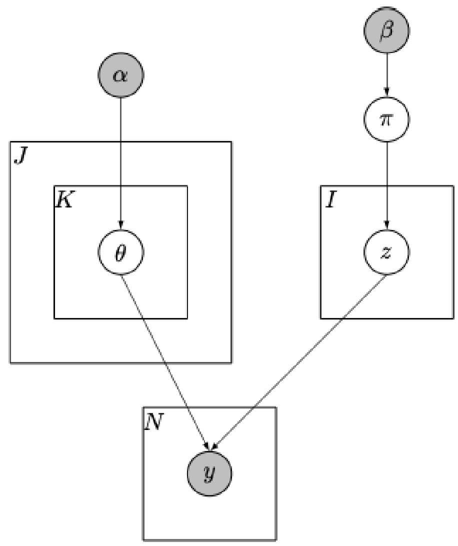

Figure 1: Plate notation of crowdsourced annotation model[1]

Let K be the number of possible labels or categories for an item, I the number of items to annotate, J the number of labelers, and N the total number of labels provided by labelers.Each item is labeled by at least one labeler.
The parameters of the model are as follows:

- $z_i \in 1 : K$ for the true category of item i,
- $\pi_k \in [0, 1]$ for the probability that an item is of category k.
- $\theta_{j,k,k'} \in [0, 1]$ for the probabilty that annotator j assigns the label k' to an item whose true category is k; Subject to $\sum_{k'=1}^{K} theta_{j,k,k'} = 1$

The model first selects the true category z_i for item i according to the prevalence π of categories where,

$$\pi \sim Dirichlet(\beta);$$

and,

$$z_i \sim Categorical(\pi)$$

Each item i is labelled by $|J_i|$ labelers, where

$$|J_i| \sim Poisson(\lambda_{labeler})$$

These labelers are choosen at random from all the labelers

$$J_i \sim Categorical(J, |J_i|)$$

For each labeler j the confusion matrix $\theta_{j,k}$,(k) is sampled from dirichlet prior α_k, where

$$\alpha_{k,k} = E_{correctness}, \quad \alpha_{k,k'} = (1 - E_{correctness})/(K - 1)$$

Here, $E_{correctness}$ denotes the expected proportion of times the labeler correctly identifies the true label. $\theta_{j,k}$ for each labeler is sampled as follows:

$$\theta_{j,k} \sim Dirichlet(\alpha_k)$$

The label y_n for item i is sampled from the labeler j's confusion matrix $\theta_{j,z[i]}$

$$y_n \sim Categorical(\theta_{J_{i[n]},z_i})$$

1.1 Data Generation

For the experiment, we simulate a model with $|J| = 100$ labelers, $|I| = 5000$ items, and $|K| = 3$ categories. The hyperprior configuration is listed in table 1. The labels of half of the items are passed to the PPL implementations for training and the other half is used to compute the posterior predictive of the obtained samples.

Table 1: Hyperpriors for Crowdsourced Annotation Model

Hyperprior	Value	Notes
$\lambda_{labeler}$	2.5	Poisson prior for number of labelers assigned to each item
β	$[\frac{1}{K}]$	Dirichlet prior for π
$E_{correctness}$	0.5	Expected accuracy of a labeler

1.2 PPL Implementations

The model was implemented in Stan and Jags PPLs, with the help of PyStan[2] and PyJAGS[3] libraries for interface. The compilation and inference times were recorded. The PPLBench is designed to accept an approximate time per PPL per iteration. The PPLBench estimates the number of posterior samples to obtain from the implementations based on this time constraint. The PPL's default warmup/burn-in and initialization parameters were left unchanged. The sampling is restricted to a single chain and any form of multi-threading is disabled. The model requires the support for discrete latent vairables, which JAGS has, but Stan does not. This means that the Stan implementation requires an additional marginaliztion step w.r.t $z[i]$, the true label of each item.

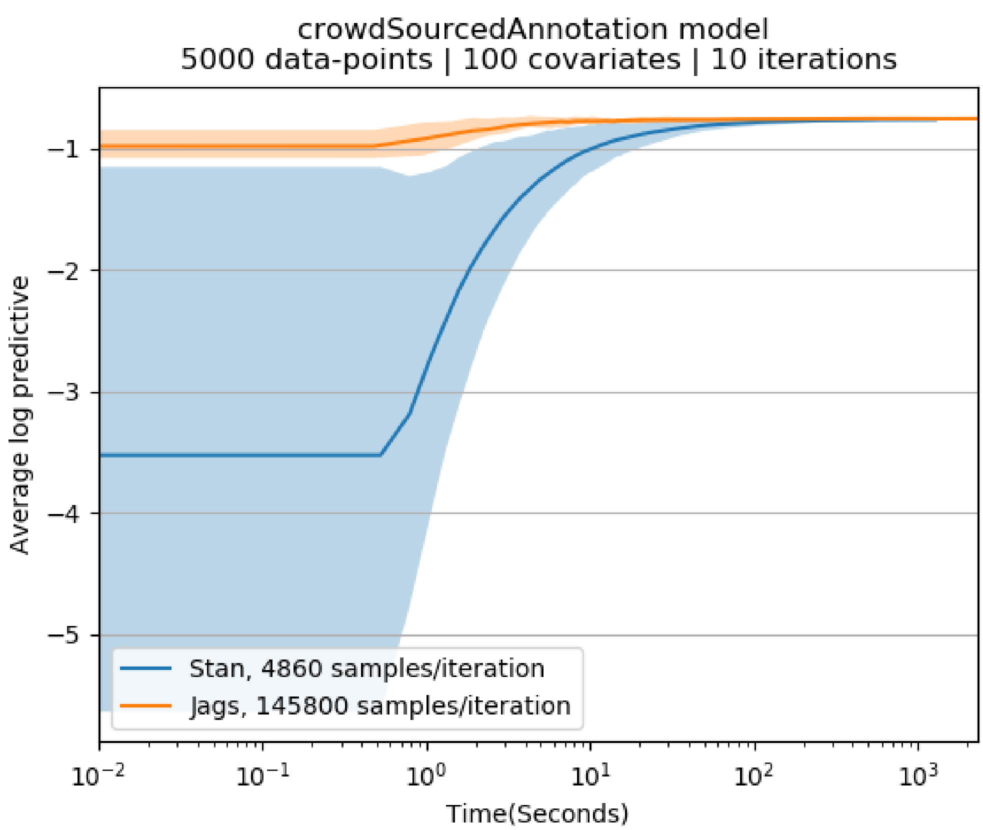

Figure 2: Posterior convergence behaviour of Jags and Stan for Crowdsourced Annotation Model

1.3 Results

Figure 2 shows the comparative performance. From the figure, we can make the following observations:

- Both implementations converge to the same posterior predictive log-likelihood given enough time.
- Jags is both faster to converge as well has a much lower time per sample.
- Jags' inference starts sampling from relatively high log-likelihood space and hence converges much quicker than STAN's

References

[1] Rebecca J Passonneau and Bob Carpenter. The benefits of a model of annotation. *Transactions of the Association for Computational Linguistics*, 2:311–326, 2014.

[2] Stan Development Team. PyStan: the Python interface to Stan, Version 2.17.1.0., 2018.

[3] Tomasz Miąsko. PyJAGS: The Python Interface to JAGS, 2019.