Bayesian Survival Analysis in STAN for Improved Measuring of Uncertainty in Parameter Estimates

Riko Kelter

MEASUREMENT: INTERDISCIPLINARY RESEARCH AND PERSPECTIVES
2020, VOL. 18, NO. 2, 101–109
https://doi.org/10.1080/15366367.2019.1689761

SOFTWARE REVIEW

Bayesian Survival Analysis in STAN for Improved Measuring of Uncertainty in Parameter Estimates

Riko Kelter

Department of Mathematics, University of Siegen

ABSTRACT
Survival analysis is an important analytic method in the social and medical sciences. Also known under the name time-to-event analysis, this method provides parameter estimation and model fitting commonly conducted via maximum-likelihood. Bayesian survival analysis offers multiple advantages over the frequentist approach for measurement practitioners, however, computational difficulties have mitigated interest in Bayesian survival models. This paper shows that Bayesian survival models can be fitted in a straightforward manner via the probabilistic programming language Stan, which offers full Bayesian inference through Hamiltonian Monte Carlo algorithms. Illustrations show the benefits for measurement practitioners in the social and medical sciences.

KEYWORDS
Software review; Bayesian survival modeling; Stan; Hamiltonian Monte Carlo

Introduction

Survival analysis or time-to-event analysis deals with censored data. This type of data is most often observed in clinical trials where the event often equals death, or in social science, where the event could be divorce or job change of a person. Censored data usually consists of the time $x_i \in \mathbb{R}_+$ and the censoring status v_i. If $v_i = 1$, x_i is observed without censoring (e.g. death, divorce), and if $v_i = 0$, x_i is censored so it is unknown what happens after x_i (e.g. because the study time ends or a patient is lost to follow-up).

The usual approach for survival data analysis is based on maximum likelihood estimation (MLE), the most prominent approach being the Cox proportional hazards model (Klein, van Houwelingen, Ibrahim, & Scheike, 2014). Bayesian analysis on the other hand uses posterior distributions of model parameters to draw inference about them. These posterior distributions are obtained via Markov-Chain-Monte-Carlo (MCMC) algorithms in realistic settings (Kelter, 2020; Ibrahim, Chen, & Sinha, 2001). In practice, algorithms like Gibbs sampling are necessary to provide posterior inference. A knowledge of probability theory is a must for researchers wanting to apply the Bayesian methodology to survival analysis.

In the last two decades, flexible modeling languages for Bayesian inference have grown in popularity. The BUGS language (Lunn, Spiegelhalter, Thomas, & Best, 2009) was the first widely used language (Monnahan, Thorson, & Branch, 2017), and was then made platform-independent in the OpenBUGS language (Lunn et al., 2009). Also, JAGS (Plummer, 2003) was a popular alternative making Bayesian inference more accessible to practitioners. Stan (Carpenter et al., 2017) can be seen as a relatively new successor to these modeling languages, implementing a new and more efficient Hamiltonian Monte Carlo (HMC) algorithm. Instead of Gibbs sampling, most often utilized by JAGS or OpenBUGS, Stan uses the No-U-Turn sampler introduced by Hoffman and Gelman (2014). This software review discusses Stan and demonstrates that survival analysis can be carried out in Stan following the Bayesian paradigm, making it an attractive option to researchers in the medical and social sciences.

CONTACT Riko Kelter riko.kelter@uni-siegen.de Department of Mathematics, University of Siegen, Siegen, Germany.

Program description

Stan is a language used in statistical modeling and high-performance statistical computation (Carpenter et al., 2017). Stan has currently been used for statistical modeling, prediction and data analysis in the social, medical and physical sciences as well as engineering and business (Gordon et al., 2018; Kucukelbir, Ranganath, Gelman, & Blei, 2015; Natanegara et al., 2014).

Stan requires the user to first specify a log density function using its probabilistic programming language. Next, parameter estimation can be achieved via full Bayesian inference with HMC sampling. Parameter estimation can also be done by approximate Bayesian inference via variational inference or by penalized maximum likelihood estimation with optimization (Carpenter et al., 2017; Gelman, Lee, & Guo, 2015).

Program interface

Stan currently does not offer a graphical user interface. Instead, Stan interfaces with other popular programming languages including R (rstan), Python (pystan), MATLAB, Julia and shell-access (Cmdstan). Stan runs on all major operating systems where installation is straightforward. While the missing graphical user interface may seem like a drawback, users acquainted with one of the other languages can easily use the Stan interface and incorporate Stan into their programming.

Stan requires the user to code their own models in Stan's probabilistic programming language, so a little programming experience is required. For users not acquainted to one of the supported languages it is recommended they start with R, as there are plenty of good ressources for R and Stan (Kruschke, 2014; McElreath, 2016). While there is no graphical user interface, Stan offers intuitive visualization of results as well as a web browser based dashboard which can be used to visualize the results of a conducted analysis, bridging the gap for users acquainted with using a graphical user interface.

Performance

The key advantage of Stan over its competitors is the algorithm. Stan uses the No-U-Turn sampler as introduced by Hoffman and Gelman (2014). The algorithm basically exploits the geometry of the posterior distribution to be sampled through Hamiltonian dynamics. Accessible introductions to the idea of Hamiltonian Monte Carlo can be found in Betancourt (2017) and Monnahan et al. (2017). By exploiting the Hamiltonian dynamics, the No-U-Turn sampler automatically scales the proposal steps, in contrast to traditional algorithms like Metropolis-Hastings, in which the proposal step size needs to be selected carefully. Stan therefore eases the burden of tuning parameter selection in MCMC for measurement practitioners and provides easy access to Bayesian modeling.

Flexibility and ease-of-use

Stan offers a highly flexible built-in probabilistic programming language which makes it possible to code complex models for inference. Although users can adapt a given complex model to their specific analytic needs, they are also required to have good theoretical and programming knowledge. Stan's selection of algorithms does include multiple MCMC algorithms like NUTS or plain HMC. The additionally available selection of algorithms which conduct optimization and variational inference makes it possible to apply Stan in a wide range of application contexts. However, the application of these flexible algorithms requires good knowledge of their underlying theory and limitations.

Regarding measurement modeling, with a particular focus on survival models, Stan offers methods to create a multitude of models. Researchers can modify or extend existing models and thereby foster discussion about Bayesian measurement modeling. This alleviates algorithmic aspects for measurement practitioners and allows them to focus on modeling perspectives.

Stan can be viewed as a highly flexible but equally complex software, but once mastered it has a variety of benefits for the measurement practictioner. Also, Stan can be successfully incorporated into a data analysis workflow based on one of the other supported programming languages.

Comprehensive output

Full Bayesian inference in Stan yields samples of the posterior distribution in the computer output. These samples are then processed in the next step, which can be achieved easily via Stan's built-in plotting and summary functions. Also, this is the point where the browser-based dashboard can be invoked if desired. Because there exist plenty of different functions to process the result of Stan's sampling algorithms, it is difficult to interpret the output, as it depends on the functions and libraries utilized. There are also plenty of open-source packages at least in R and Python so that there are practically no limitations in processing the posterior samples. Gelman et al. (2013) provide recommendations for a fully Bayesian workflow. Statistical summaries of the posterior as well as density plots and posterior predictive checks can easily be obtained via the standard Stan interfaces in the respective language.

Regarding errors, Stan gives compiler-like error messages when either the model coded includes a mistake or the programming language interface encounters a problem. These error messages are in most cases good enough to solve the problem, and help can be found also at the official Stan community forum (https://discourse.mc-stan.org) or via the documentation.

Manual and documentation

Stan's official documentation consists of a reference manual and a user guide, both of which can be found at the official website. The user guide provides example models as well as programming techniques for writing efficient Stan models which are successfully handled by the MCMC algorithms. While intended as an example-driven introduction to Bayesian modeling and inference, users with little prior knowledge and programming experience will most probably have trouble following the user guide. It is therefore recommended to use one of the excellent textbooks on Stan for the psychological and medical sciences (Kruschke, 2014) or for the social sciences (McElreath, 2016). The reference manual is a programming API which is not very helpful for practical purposes. Stan's documentation can be viewed as rudimentary, especially lacking an easy-to-follow introduction for beginners. This may be attributed to Stan's high flexibility, which in turn offers estimation and fitting of many complex models. Researchers interested in full Bayesian inference for survival models should concentrate one of the two textbooks recommended above as a starting point.

Bayesian measurement modeling in Stan

Most quantities used in statistical models arise from the variable measurements taken. In realistic applications, most of these measurements have measurement error. If the measurement error is small relative to the quantity being measured, its impact on a model can be gauged as small. If on the other hand measurement error is large relative to the measured quantity, it is reasonable to introduce an explicit measurement error model. Stan allows for multiple models with measurement error, and a whole chapter is dedicated to this topic in the Stan user guide.

Meta-analysis also plays an important role in applied research, especially when inferences drawn from multiple data sets need to be combined to make statements about all of them together. Inferences for each data set in Stan can be treated as providing a kind of measurement error regarding the true parameter values.

These two examples highlight where Stan's modeling language allows for flexible Bayesian measurement modeling. Stan's flexibility also offers other methods, which can be easily implemented. For a general overview of Bayesian measurement modeling and Bayesian error measurement see Richardson and Gilks (1993) and Buzas, Stefanski, and Tosteson (2005).

A detailed example

To illustrate the use of Stan for survival modeling, a simple parametric survival model example is discussed. Parametric survival models are important as most Bayesian survival analyses in clinical research are carried out using parametric survival models (Brard, Le Teuff, Le Deley, & Hampson, 2017). Therefore, the example presented below uses the parametric exponential model. The exponential model is the most basic model for Bayesian survival analysis and assumes that the survival times $y = (y_1, y_2, \ldots, y_n)$ are each distributed exponentially with parameter λ, that is

$$f(y_i|\lambda) = \lambda \exp(-\lambda y_i) \qquad \text{for } i = 1, \ldots, n \tag{1}$$

Denoting the censoring indicators as $v = (v_1, v_2, \ldots, v_n)$ where $v_i = 0$ if y_i is right censored (lost to follow-up) and $v_i = 1$ if y_i is a failure time (death, divorce, job change), the survival function, which is simply defined as the probability of surviving past the time point y_i is given by

$$S(y_i|\lambda) = \mathrm{P}(T \geq y_i|T \geq 0) = 1 - F(y_i|\lambda) = 1 - [1 - \exp(-\lambda y_i)] = \exp(-\lambda y_i) \tag{2}$$

where $F(\cdot|\lambda)$ is the cumulative distribution function of the exponential distribution with parameter λ, and T is a random variable modeling the survival time. The observed data D are composed of the number of observations n, the observations y themselves and the censoring status v, and we can write the likelihood as

$$L(\lambda|D) = \prod\nolimits_{i=1}^{n} f(y_i|\lambda)^{v_i} \; S(y_i|\lambda)^{1-v_i} \tag{3}$$

The likelihood is simply a product of $f(y_i|\lambda)$ for all observations with censoring status $v_i = 1$ (death observed) and the survival function $S(y_i|\lambda)$ for all observations with censoring status $v_i = 0$. It is possible to use conjugate priors to reach a closed-form posterior, but if one does not want to limit modeling to the Gamma conjugate family of prior distributions for λ, Stan can be used for more flexible modeling.

Covariates need to be incorporated into the model. One could for example set $\lambda = x_i^T\beta$ for a $p \times 1$ covariate vector x_i and a $p \times 1$ regression coefficients vector β. The reason that usually $\lambda = \exp(x_i^T\beta)$ is used as a predictor instead of $\lambda = x_i^T\beta$ is simple: One wants to interpret increasing coefficients β as increasing risk, that is as a decreasing survival function. If $\lambda = \exp(x_i^T\beta)$, first of all λ is larger than zero for all coefficients β. Second, if β increases, λ does, too. Lastly, if β and subsequently λ increases, the survival function $S(t|\lambda) = \exp(-\lambda \cdot t)$ decreases, leading to the desired behavior. In summary, the exponential survival model can be written as

$$y_i|v_i \sim f(y_i|\lambda)^{v_i} + S(y_i|\lambda)^{1-v_i} = [\lambda \exp(-\lambda y_i)]^{v_i} + [\exp(-\lambda y_i)]^{1-v_i} \tag{4}$$

$$\lambda \sim p(\lambda) \tag{5}$$

$$\lambda = \exp(x_i^T\beta) \tag{6}$$

where $p(\lambda)$ is the prior on λ.

Modeling the parametric survival model in Stan is straightforward. Listing 1 shows the Stan model code for the exponential survival model where the model is specified directly as a string in R.

```
Stan_exponential_survival_model<-"
data{
  int<lower=1> N_uncensored;
  int<lower=1> N_censored;
  int<lower=0> numCovariates;
  matrix[N_censored,numCovariates] X_censored;
  matrix[N_uncensored,numCovariates] X_uncensored;
  vector<lower=0>[N_censored] times_censored;
  vector<lower=0>[N_uncensored] times_uncensored;
}
parameters{
  vector[numCovariates] beta; // regression coefficients
  real alpha; // intercept
}
model{
  beta ~ normal(0,10); // prior on regression coefficients
  alpha ~ normal(0,10); // prior on intercept

  target += exponential_lpdf(times_uncensored | exp(alpha+X_
   uncensored*beta)); // log-likelihood part for uncensored
   times
  target += exponential_lccdf(times_censored | exp(alpha+X_
   censored*beta)); // log-likelihood for censored times
}
generated quantities{
  vector[N_uncensored] times_uncensored_sampled; //
   prediction of death
  for(i in 1:N_uncensored) {
    times_uncensored_sampled[i] = exponential_rng(exp(alpha+X
   _uncensored[i,]*beta));
  }
}
"
```

Listing 1. Exponential Survival Model in Stan.

The Stan model code consists of three core blocks: The data block, the parameters and the model block. Also, the generated quantities block is added here, which is optional. The data block contains all data variables handed to the Stan model. Here, the number of uncensored and censored observations are defined by the variables `N_uncensored` and `N_censored`. `numCovariates` is the number of covariates used, which will be set to one in the example below. Then, the design matrices `X_censored` and `X_uncensored` for the censored and uncensored observations are defined. The last data handed to Stan as input are the observations y_i, split into the censored and uncensored observations in form of the vectors `times_censored` and `times_uncensored`. The parameters block includes all parameters for which posterior draws are desired. Interest lies in $\lambda = \exp(x_i^T\beta)$, and more specific in the coefficients β, denoted as beta. Also, the intercept term is modeled directly via the parameter `alpha` instead of assuming that a design matrix with the first column being only ones is handed to Stan. The model block then proceeds by computing the likelihood for all observations. The first line beginning with `target + =` uses the log-exponential probability density function for the uncensored observations. Note that $\lambda = \exp(x_i^T\beta)$ so that `exp(alpha+X_uncensored*beta)` is the parameter of the log-exponential density. The following line proceeds by adding the log-complement cumulative distribution function for the censored times. The log-complement cumulative density function is defined as $1{-}F(x)$, where $F(x)$ is the cumulative distribution function. So, this is exactly the survival function $S(t)$. Finally, in the generated quantities block failure (death) times for the uncensored observations of the input data are generated. This way, failure times for each uncensored observation y_i can be predicted. The priors for β and α are defined as N(0, 10) here, which is a vague prior, staying uninformative.

Listing 2 shows the R-Code to fit the Stan model to the ovarian dataset via the interface *rstan* (Goodrich et al., 2020), which is the official interface for R to Stan. The ovarian dataset contains survival times in a randomized clinical trial comparing two treatments for ovarian cancer and can be accessed by installing the survival package in R from CRAN.[1]. The predictor used in the example is the treatment used, where the first treatment is coded as 1 and the second treatment as 2. First, the data is prepared into the formats defined in the data block of the Stan model, and after that Stan is run.

```
# Prepare data
set.seed(42); require(tidyverse); N <- nrow(ovarian); X <-
  as.matrix(pull(ovarian, rx)); is_censored <- pull(ovarian,
  fustat)==0; times <- pull(ovarian,futime); msk_censored <-
   is_censored == 1; N_censored <- sum(msk_censored);

# Put data into a list for Stan
Stan_data <- list(N_uncensored=N-N_censored, N_censored=N_
  censored, numCovariates=ncol(X), X_censored=as.matrix(X[
  msk_censored,]), X_uncensored=as.matrix(X[!msk_censored,])
  , times_censored=times[msk_censored], times_uncensored =
  times[!msk_censored])
Stan_data

# Fit Stan model
require(rstan)
exp_surv_model_fit <- stan(model_code = Stan_exponential_
  survival_model, data=Stan_data)

# Print model fit
exp_surv_model_fit
             mean  se_mean  sd    2.5%   25%     50%     75%   97.5%
beta[1]     -0.65  0.02    0.60  -1.89  -1.04   -0.63   -0.24  0.48
alpha       -6.28  0.03    0.89  -8.00  -6.86   -6.26   -5.68 -4.56
```

Listing 2. Exponential Survival Model fit in R via Stan.

The model fit shows that the treatment coefficient β_1 has a posterior mean of –0.67, indicating that the second treatment (coded as two) may increase the survival probability of patients. Still, as the 97.5% quantile is 0.48 and larger than zero, the estimate is highly uncertain. A likelihood based exponential model would yield a MLE for the treatment coefficient of –0.596 with a standard error of 0.587, indicating that there is a slightly beneficial effect in the second treatment on the survival time. The p-value would be $p = .3$, indicating no significance. The sample size of only 26 patients makes the approximations used for computation of the standard error and p-value highly unreliable, and the obtained results unreliable. The advantage of the Bayesian model is the embracing of uncertainty and increased flexibility via the prior modeling. Few researchers would accept a vague prior if previous studies indicated evidence of a positive effect for one or both treatments. A second advantage for measurement practitioners is given by the fact that it is easy to construct survival functions $S(t|\lambda, x_i = j)$ for given covariate values $x_i = j$. In the above example, there is only one covariate x_1 which is either 1 if the first treatment is used, or 2 if the second treatment is used. Thus, one can compare the estimated posterior survival functions $S(t|\lambda, x_i) = \exp(-\lambda y_i) = \exp[-\exp(x_i^T\beta)y_i]$ for different treatments, where $S(t|\lambda, x_i = 1) = \exp[-\exp(\beta)y_i]$ and $S(t|\lambda, x_i = 2) = \exp[-\exp(2\beta)y_i]$. Figure 1 shows the posterior survival functions for the first and second treatment using the posterior mean of β_1 (middle solid line), as well as the 2.5% and 97.5% quantiles (lower and upper dotted lines). Overlayed are various survival functions using a range of credible posterior values of β_1. It is clear that while the survival function of the first treatment group decreases much faster, the credible ranges of survival functions for both groups overlap widely. This was already reflected in the 97.5% quantile of β_1 crossing zero. Thus, while a traditional survival analysis using the Cox proportional hazards model would yield a single p-value and at best a point-estimator with confidence intervals, the Bayesian parametric exponential model embraces the uncertainty in the very small dataset of just 26 patients by providing a whole posterior distribution for the treatment coefficient β_1 which in turn leads to a range of credible survival curves, given the data.

Conclusion

In this software review the probabilistic programming language Stan was discussed with a simple example to show how to fit a survival model in a fully Bayesian workflow using R via Stan. Stan offers some excellent features for Bayesian inference, which includes a high performance algorithm, interfaces to a wide range of programming languages and customizable program output. However, these advantages have their price: The learning curve of Stan is steep, and the missing graphical user interface and the highly technical documentations are clear limitations in practical use. Overall, the measurement practitioner should have both programming experience and solid theoretical knowledge of Bayesian inference to (1) code the correct model in Stan's probabilistic programming language and (2) run the model via a programming language interface (like the *rstan* package used in the example above).

The time spent to learn Stan however is worth the effort. The survival analysis example illustrated that not only is Bayesian inference simplified by using Stan, but that the uncertainty of parameter estimates is embraced, gauging the reliability of the results better than via traditional maximum

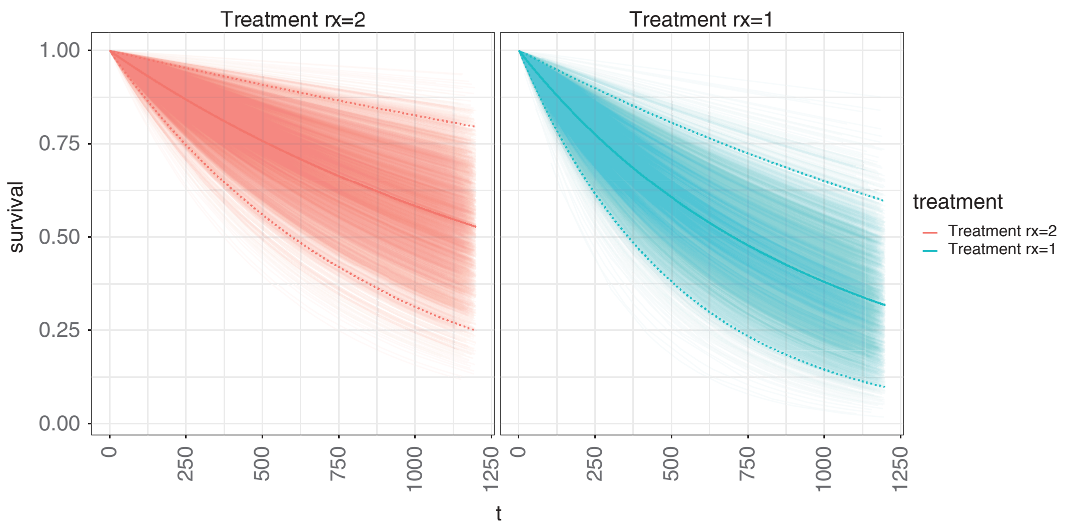

Figure 1. Posterior survival functions per treatment group for the ovarian dataset.

likelihood based point-estimates. Also, measurement can be seen as associated with the probability model generating the quantities measured, which is a strength of Stan. The probabilistic programming language offers coding of complex and highly customizable models of a real phenomenon, therefore enabling researchers to build and subsequently analyze otherwise untreatable probability models in a reasonable amount of time. Only by this added layer of complexity could the processing of the results in the form of a credible range of survival curves for each treatment in the ovarian example be achieved. This in turn allows for a precise measuring of the predicted survival time for each treatment condition, and the model could also be extended to predict survival time for each patient, each gender or combinations of both.

Another benefit of full Bayesian inference in Stan for survival models is that the specific model must be reported to understand the analysis. This fosters critical thinking about modeling aspects, as the model should reflect the reality underlying the measurement process. Also, measurement modeling can be done in a fully Bayesian way with little effort in Stan. The main advantage of Stan thus can be seen as a model-based access to full Bayesian inference with a highly customizable modeling language and efficient algorithms. This should be a strong benefit for Bayesian measurement modeling, in particular for survival models in the social, psychological or medical sciences.

Note

1. See https://cran.r-project.org/web/packages/survival/index.html.

ORCID

Riko Kelter http://orcid.org/0000-0001-9068-5696

References

Betancourt, M. (2017). A conceptual introduction to Hamiltonian Monte Carlo. arXiv preprint.

Brard, C., Le Teuff, G., Le Deley, M.-C., & Hampson, L. V. (2017, feb). Bayesian survival analysis in clinical trials: What methods are used in practice? *Clinical Trials: Journal of the Society for Clinical Trials*, *14*(1), 78–87. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/27729499http://journals.sagepub.com/

Buzas, J. S., Stefanski, L. A., & Tosteson, T. D. (2005). Measurement error. In W. Ahrens & I. Pigeot (Eds.), *Handbook of epidemiology* (pp. 729–765). Berlin, Heidelberg: Springer Berlin Heidelberg.

Carpenter, B., Guo, J., Hoffman, M. D., Brubaker, M., Gelman, A., Lee, D., … Betancourt, M. (2017). Stan: A probabilistic programming language. *Journal of Statistical Software*, *76*, 1. doi:10.18637/jss.v076.i01

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2013). *Bayesian data analysis* (3rd ed.). CRC Press Taylor & Francis.

Gelman, A., Lee, D., & Guo, J. (2015, oct). Stan: A probabilistic programming language for Bayesian inference. *Journal of Educational and Behavioral Statistics*, *40*(5), 530–543. Retrieved from doi: 10.3102/1076998615606113.

Goodrich, B., Gabry, J., Ali, I., & Brilleman S. (2020). rstanarm: Bayesian applied regression modeling via Stan. R package version 2.19.3.

Gordon, G. S. D., Joseph, J., Alcolea, M. P., Sawyer, T., Macfaden, A. J., Williams, C., Fitzpatrick, C. R. M., Jones, P. H., di Pietro, M., Fitzgerald, R. C., Wilkinson, T. D., Bohndiek, S. E. (2018). Quantitative phase and polarisation endoscopy applied to detection of early oesophageal tumourigenesis. arXiv preprint. Retrieved from: http://arxiv.org/abs/1811.03977

Hoffman, M. D., & Gelman, A. (2014). The No-U-turn sampler: Adaptively setting path lengths in Hamiltonian Monte Carlo. *Journal of Machine Learning Research*, *15*, 1351–1381. Retrieved from http://arxiv.org/abs/1111.4246

Ibrahim, J. G., Chen, M.-H., & Sinha, D. (2001). *Bayesian survival analysis*. Boston, MA: Springer.

Kelter, R. (2020). Analysis of Bayesian posterior significance and effect size indices for the two-sample t-test to support reproducible medical research. *BMC Medical Research Methodology 20*(88). doi: https://doi.org/10.1186/s12874-020-00968-2.

Klein, J. P., van Houwelingen, H. C., Ibrahim, J. G., & Scheike, T. H. (2014). *Handbook of survival analysis*. Boston, MA: Chapman & Hall/CRC. doi:10.1201/b16248.

Kruschke, J. K. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan* (2nd ed.). Academic Press. doi:10.1016/B978-0-12-405888-0.09999-2.

Kucukelbir, A., Ranganath, R., Gelman, A., & Blei, D. M. (2015). *Automatic variational inference in Stan*. NIPS'15 Proceedings of the 28th International Conference on Neural Information Processing Systems. (2015), Vol. I, p. 568-576, Montreal, Canada. MIT Press Cambridge.

Lunn, D., Spiegelhalter, D., Thomas, A., & Best, N. (2009, November). The BUGS project: Evolution, critique and future directions. *Statistics in Medicine*, *28*(25), 3049–3067. Retrieved from http://www.ncbi.nlm.nih.gov/pubmed/19630097

McElreath, R. (2016). *Statistical rethinking: A Bayesian course with examples in R and Stan*. Leipzig, Germany: Chapman & Hall, CRC Press. doi:10.3102/1076998616659752.

Monnahan, C. C., Thorson, J. T., & Branch, T. A. (2017, March). Faster estimation of Bayesian models in ecology using Hamiltonian Monte Carlo. *Methods in Ecology and Evolution*, *8*(3), 339–348. doi: 10.1111/2041-210X.12681.

Natanegara, F., Neuenschwander, B., Seaman, J. W., Kinnersley, N., Heil- Mann, C. R., Ohlssen, D., & Rochester, G. (2014, jan). The current state of Bayesian methods in medical product development: Survey results and recommendations from the DIA Bayesian scientific working group. *Pharmaceutical Statistics*, *13*(1), 3–12. doi:10.1002/pst.1595.

Plummer, M. (2003). *JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling*. Proceedings of the 3rd international workshop on distributed statistical computing (dsc 2003), Vienna, Austria. doi: 10.1111/j.2517-6161.1996.tb02070.x.

Richardson, S., & Gilks, W. R. (1993). A Bayesian approach to measurement error problems in epidemiology using conditional independence models. *American Journal of Epidemiology*, *138*(6), 430–442. doi:10.1093/oxfordjournals.aje.a116875