Rigdon Basu (1989) Time truncated data example

replication in Stan

Miao Cai *miao.cai@slu.edu*

1/29/2019

1 Time truncated data - 115 kV Transmission line example

This data is presented by Martz (1975). It gives the failure times, or interrpution times, of the 115 kV transmission circuit from Cunningham Generating Station, located near Hobbs, New Mexico, to Eddy County Interchange, located near Artesia, New Mexico. We assume that data collection was terminated on December 31, 1971.

Data collected in this manner, with testing terminated at a predetermined time, are called time truncated. It is important to distinguish between these approches to data collection because statistical inference precedures are different for the two situations.

```
library(rstan)
```

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

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

```
## rstan (Version 2.18.2, 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)
```

```
## For improved execution time, we recommend calling
## Sys.setenv(LOCAL_CPPFLAGS = '-march=native')
## although this causes Stan to throw an error on a few processors.
```

```
t = c(0.129, 0.151, 0.762, 0.869, 2.937, 3.077, 3.841, 3.964, 4.802, 4.898, 7.868, 8.430)
trunc_time = 8.463

datstan = list(
  n = length(t),
  tau = trunc_time,
  t = t)
```

MLE estimates of β and *theta* are provided by Rigdon and Basu (1989):

$$\hat{\beta} = \frac{12}{\sum_{i=1}^{12} \ln(8.463/t_i)} = 0.678$$

$$\hat{\theta} = \frac{8.463}{12^{1/0.678}} = 0.217$$

1.1 Diffuse priors for beta and theta - GAMMMA(1, 1)

The log likelihood function l for time truncated non-homogeneous Poisson process is:

$$
\begin{aligned}
l &= \log\Big(\prod_{i=1}^{n}\frac{\beta}{\theta}(\frac{t_i}{\theta})^{\beta-1}\Big)e^{-(\tau/\theta)^{\beta}} \\
&= \sum_{i=1}^{n}\log\Big(\frac{\beta}{\theta}(\frac{t_i}{\theta})^{\beta-1}\Big) - (\frac{\tau}{\theta})^{\beta} \\
&= n\log\beta - n\beta\log\theta + (\beta-1)\sum_{i=1}^{n}\log t_i - (\frac{\tau}{\theta})^{\beta}
\end{aligned}
$$

```
plpstan11 = '
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
}
data {
  int<lower=0> n; //total # of obs
  real<lower=0> tau;//truncated time
  vector<lower=0>[n] t; //failure time
}
parameters{
  real<lower=0> beta;
  real<lower=0> theta;
}
model{
  t ~ nhpp(beta, theta, tau);
//PRIORS
  beta ~ gamma(1, 1);
  theta ~ gamma(1, 1);
}
'

fitplp <- stan(
  model_code=plpstan11, model_name="NHPP", data=datstan,
  iter=5000,warmup = 2000, chains=1, seed = 123
)
```

```
##
## SAMPLING FOR MODEL 'NHPP' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
```

```
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.092 seconds (Warm-up)
## Chain 1:                0.127 seconds (Sampling)
## Chain 1:                0.219 seconds (Total)
## Chain 1:
```

```
fitplp
```

```
## Inference for Stan model: NHPP.
## 1 chains, each with iter=5000; warmup=2000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=3000.
##
##         mean se_mean   sd   2.5%    25%    50%   75% 97.5% n_eff Rhat
## beta    0.75    0.01 0.17   0.46   0.63   0.73  0.86  1.13   805    1
## theta   0.41    0.01 0.30   0.05   0.19   0.34  0.56  1.17   916    1
## lp__  -10.37    0.04 1.06 -13.27 -10.77 -10.05 -9.62 -9.32   737    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jan 30 10:19:27 2019.
## 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).
```

Compared to MLE estimate of $\hat{\beta} = 0.678, \hat{\theta} = 0.217$.

1.2 Weakly informative priors for beta and theta - GAMMMA(1, 5)

```
plpstan15 = '
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
```

```
}
data {
  int<lower=0> n; //total # of obs
  real<lower=0> tau;//truncated time
  vector<lower=0>[n] t; //failure time
}
parameters{
  real<lower=0> beta;
  real<lower=0> theta;
}
model{
  t ~ nhpp(beta, theta, tau);
//PRIORS
beta ~ gamma(1, 5);
theta ~ gamma(1, 5);
}
'

fitplp15 <- stan(
  model_code=plpstan15, model_name="NHPP15", data=datstan,
  iter=5000,warmup = 2000, chains=1, seed = 123
)
```

```
##
## SAMPLING FOR MODEL 'NHPP15' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.088 seconds (Warm-up)
## Chain 1:                0.128 seconds (Sampling)
## Chain 1:                0.216 seconds (Total)
## Chain 1:
```

```
fitplp15
```

```
## Inference for Stan model: NHPP15.
## 1 chains, each with iter=5000; warmup=2000; thin=1;
```

```
## post-warmup draws per chain=3000, total post-warmup draws=3000.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta    0.57    0.00 0.11   0.38   0.49   0.56   0.65   0.81   777    1
## theta   0.16    0.00 0.12   0.02   0.07   0.12   0.21   0.46   882    1
## lp__  -14.03    0.03 0.98 -16.77 -14.42 -13.76 -13.32 -13.06   904    1
##
## Samples were drawn using NUTS(diag_e) at Wed Jan 30 10:08:29 2019.
## 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).
```

Compared to MLE estimate of $\hat{\beta} = 0.678, \hat{\theta} = 0.217$.

1.3 Another example

This software failure times in seconds example was presented in Bar-Lev, Lavi, and Reiser (1992).

```
library(rstan)

software_failure = c(
  115, 115, 198, 376, 570, 706, 1783, 1798, 1813, 1905, 1955, 2026, 2632,
  3821, 3861, 4649, 4871, 4943, 5558, 6147, 6162, 6552, 8415, 9752, 14260,
  15094, 18494, 18500, 23061, 26229, 36800, 37363, 40133, 40785, 46378,
  58074, 64798, 67344)
software_fTrun = software_failure[length(software_failure)]

datsoftware = list(
  n = length(software_failure),
  tau = software_fTrun,
  t = software_failure)
```

```
plpsoftware <- stan(
  model_code=plpstan11, model_name="NHPPfailureT", data=datsoftware,
  iter=5000,warmup = 2000, chains=1, seed = 123
)
```

```
##
## SAMPLING FOR MODEL 'NHPP' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
```

```
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.174 seconds (Warm-up)
## Chain 1:                0.272 seconds (Sampling)
## Chain 1:                0.446 seconds (Total)
## Chain 1:
```

```
plpsoftware
```

```
## Inference for Stan model: NHPP.
## 1 chains, each with iter=5000; warmup=2000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=3000.
##
##          mean se_mean   sd    2.5%     25%     50%     75%   97.5% n_eff
## beta     0.33    0.00 0.03    0.28    0.32    0.33    0.35    0.38   635
## theta    1.46    0.04 1.08    0.15    0.65    1.20    2.00    4.15   824
## lp__  -303.73    0.04 1.05 -306.45 -304.08 -303.40 -302.98 -302.72   656
##       Rhat
## beta     1
## theta    1
## lp__     1
##
## Samples were drawn using NUTS(diag_e) at Wed Jan 30 10:08:30 2019.
## 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).
```

2 Failure truncation data (INCORRECT)

This failure truncation example is also from Rigdon and Basu (1989).

MLE estimates for β and θ are:

$$\hat{\beta} = \frac{13}{\sum_{i=1}^{13-1} \ln(4596/t_i)} = 0.569$$

$$\hat{\theta} = \frac{4596}{13^{1/0.596}} = 50.7$$

```
library(rstan)

t1 = c(55, 166, 205, 341, 488, 567, 731, 1308, 2050, 2453, 3115, 4017, 4596)
trunc_timet1 = t1[length(t1)]

datfailure = list(
  n = length(t1),
  tau = trunc_timet1,
  t = t1)
```

2.1 Gamma(100, 1) as the prior

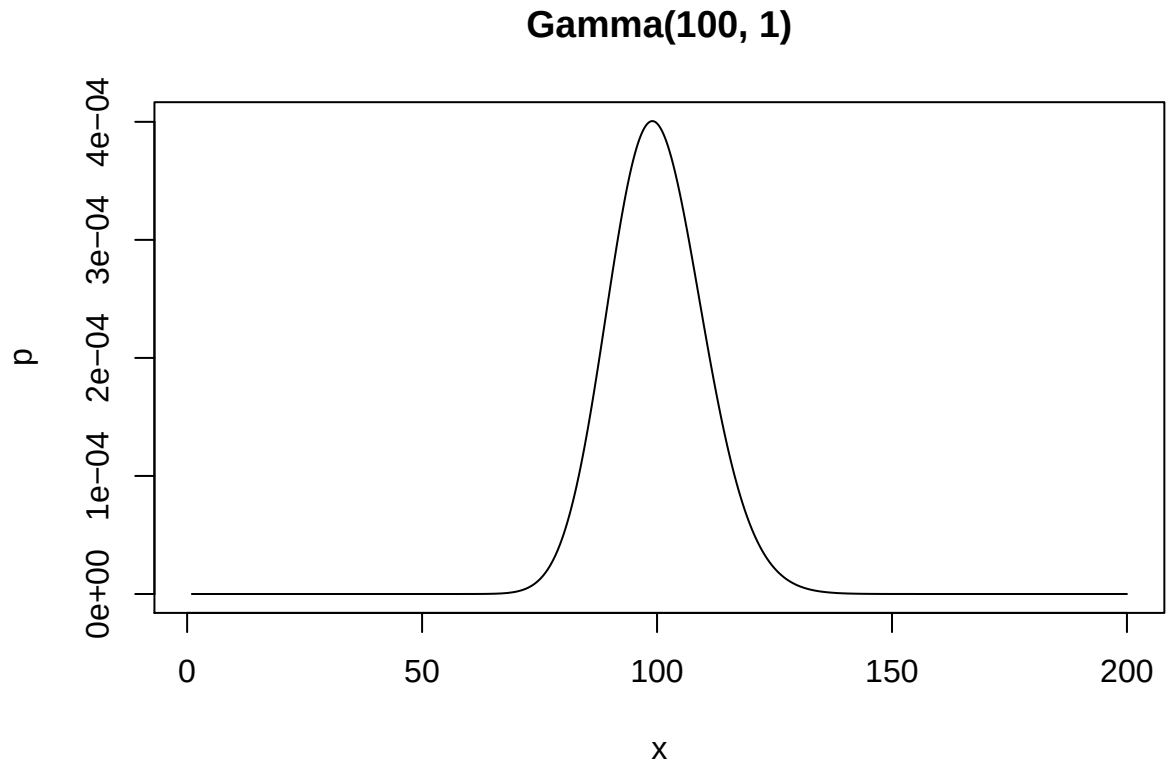
Gamma distribution parameterization in Stan:

$$\text{Gamma}(y|\alpha,\beta) = \frac{\beta^\alpha}{\Gamma(\alpha)} y^{\alpha-1}\exp(-\beta y)$$

Here we assume that $\beta \sim gamma(1,1), \theta \sim gamma(100,1)$. So essentially $\beta \sim EXP(1)$, while the probability density function of θ can be re-written as:

$$f(y|\alpha,\beta) = \frac{100}{99!} y^{99}\exp(-y)$$

```
PGAM = function(x, alpha, beta) return(beta^alpha/factorial(alpha)*x^(alpha - 1)*exp(-beta*x))
x = seq(1, 200, 0.1)
p = PGAM(x, 100, 1)
plot(x, p, type = "l", main = "Gamma(100, 1)")
```




```
plpstan11 = '
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
```

```
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
}
data {
  int<lower=0> n; //total # of obs
  real<lower=0> tau;//truncated time
  vector<lower=0>[n] t; //failure time
}
parameters{
  real<lower=0> beta;
  real<lower=0> theta;
}
model{
  t ~ nhpp(beta, theta, tau);
//PRIORS
  beta ~ gamma(1, 1);
  theta ~ gamma(100, 1);
}
'


plpfailure <- stan(
  model_code=plpstan11, model_name="NHPPfailureT", data=datfailure,
  iter=5000,warmup = 2000, chains=1, seed = 123, init = list(chain_1 = list(beta = 0.5, theta = 100))
)
```

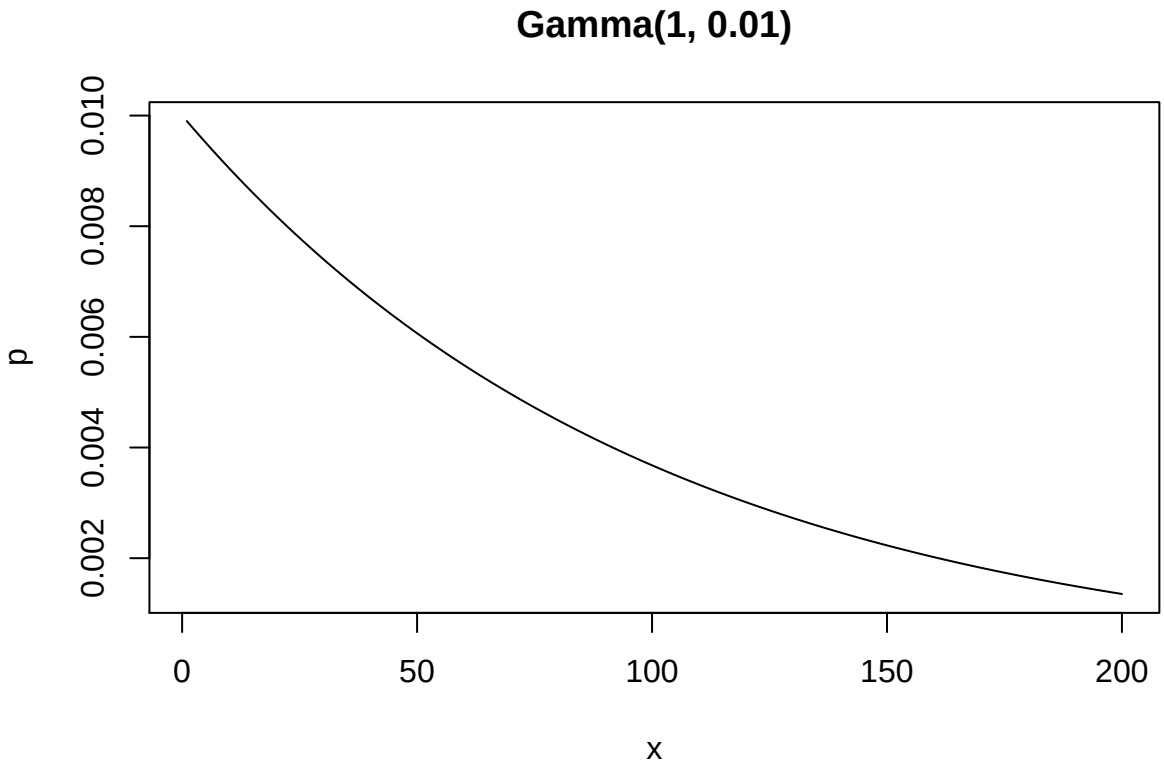
```
##
## SAMPLING FOR MODEL 'NHPPfailureT' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.9e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.0316 seconds (Warm-up)
## Chain 1:                0.058022 seconds (Sampling)
## Chain 1:                0.089622 seconds (Total)
## Chain 1:
```

```
plpfailure
```

```
## Inference for Stan model: NHPPfailureT.
## 1 chains, each with iter=5000; warmup=2000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=3000.
##
##         mean se_mean   sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta    0.64    0.00 0.07   0.51   0.60   0.64   0.69   0.77  1779    1
## theta  99.34    0.22 9.69  80.96  92.71  99.01 105.80 118.98  1980    1
## lp__  271.55    0.03 0.96 268.97 271.18 271.84 272.24 272.49  1283    1
##
## Samples were drawn using NUTS(diag_e) at Thu Jan 31 14:16:50 2019.
## 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).
```

2.2 Gamma(1, 100) as the prior

```
PGAM = function(x, alpha, beta) return(beta^alpha/factorial(alpha)*x^(alpha - 1)*exp(-beta*x))
x = seq(1, 200, 0.1)
p = PGAM(x, 1, 0.01)
plot(x, p, type = "l", main = "Gamma(1, 0.01)")
```

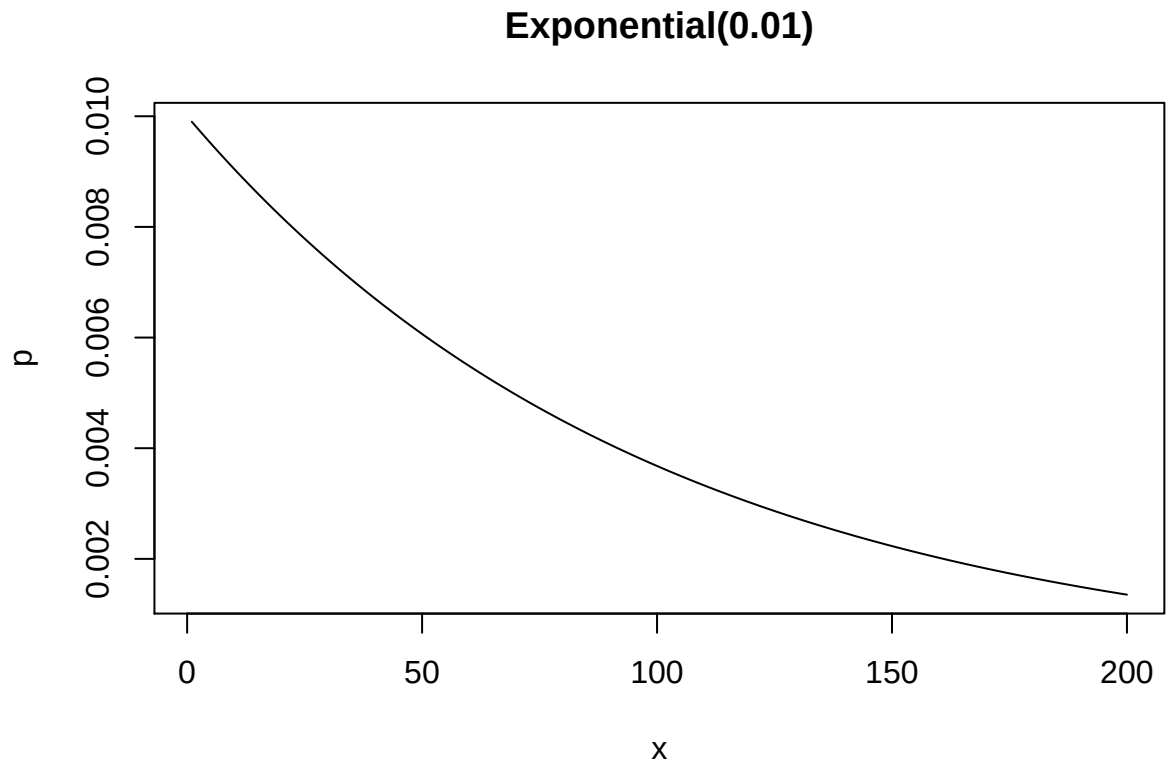


2.3 Rerun the model with exponential(0.01)/gamma(1, 0.01) prior for beta

Exponential distribution parameterization in Stan:

$$\text{Exponential}(y|\beta) = \beta \exp(-\beta y)$$

```
PEXP = function(x, beta) return(beta*exp(-beta*x))
x = seq(1, 200, 0.1)
p = PEXP(x, 1/100)
plot(x, p, type = "l", main = "Exponential(0.01)")
```




```
plpstanexp = '
functions{
  real nhpp_log(vector t, real beta, real theta, real tau){
    vector[num_elements(t)] loglik_part;
    real loglikelihood;
    for (i in 1:num_elements(t)){
      loglik_part[i] = log(beta) - beta*log(theta) + (beta - 1)*log(t[i]);
    }
    loglikelihood = sum(loglik_part) - (tau/theta)^beta;
    return loglikelihood;
  }
}
data {
  int<lower=0> n; //total # of obs
```

```
  real<lower=0> tau;//truncated time
  vector<lower=0>[n] t; //failure time
}
parameters{
  real<lower=0> beta;
  real<lower=0> theta;
}
model{
  t ~ nhpp(beta, theta, tau);
//PRIORS
  beta ~ gamma(1, 1);
  theta ~ exponential(0.01);
}
'


plpfailureexp <- stan(
  model_code=plpstanexp, model_name="NHPPfailureT", data=datfailure,
  iter=5000,warmup = 2000, chains=1, seed = 123
  #, init = list(chain_1 = list(beta = 0.5, theta = 100))
)
```

```
##
## SAMPLING FOR MODEL 'NHPPfailureT' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 1.5e-05 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.15 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Warmup)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Warmup)
## Chain 1: Iteration: 2001 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 0.05538 seconds (Warm-up)
## Chain 1:                0.071996 seconds (Sampling)
## Chain 1:                0.127376 seconds (Total)
## Chain 1:
```

```
plpfailureexp
```

```
## Inference for Stan model: NHPPfailureT.
## 1 chains, each with iter=5000; warmup=2000; thin=1;
## post-warmup draws per chain=3000, total post-warmup draws=3000.
```

```
##
##         mean se_mean    sd   2.5%    25%    50%    75%  97.5% n_eff Rhat
## beta    0.59    0.00  0.11   0.39   0.51   0.58   0.66   0.83   659    1
## theta  80.18    2.13 60.74   8.80  35.31  64.73 109.51 233.44   813    1
## lp__  -85.35    0.03  0.98 -87.97 -85.73 -85.04 -84.64 -84.37   820    1
##
## Samples were drawn using NUTS(diag_e) at Thu Jan 31 14:17:27 2019.
## 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).
```

```
plpdraws = extract(plpfailureexp)
mean(plpdraws$beta)
```

```
## [1] 0.591951
```

```
mean(plpdraws$theta)
```

```
## [1] 80.17507
```

References

Bar-Lev, Shaul K, Idit Lavi, and Benjamin Reiser. 1992. "Bayesian Inference for the Power Law Process." *Annals of the Institute of Statistical Mathematics* 44 (4). Springer: 623–39.

Martz, HF. 1975. "Pooling Life-Test Data by Means of the Empirical Bayes Method." *IEEE Transactions on Reliability* 24 (1). IEEE: 27–30.

Rigdon, Steven E, and Asit P Basu. 1989. "The Power Law Process: A Model for the Reliability of Repairable Systems." *Journal of Quality Technology* 21 (4). Taylor & Francis: 251–60.