07Compare_lp_and_log

Miao Cai miao.cai@slu.edu

2019-07-18

Contents

1 _lp form

```
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;
  }
  real nhppnoevent_lp(real tau, real beta, real theta){
    real loglikelihood = - (tau/theta)^beta;
    return(loglikelihood);
  }
}
data {
  int<lower=1> N; //total # of failures
  int<lower=1> K; //number of predictors
  int<lower=1> S; //total # of shifts
  int<lower=1> D; //total # of drivers
  int<lower=1> id[S];//driver index, must be an array
  vector<lower=0>[S] tau;//truncated time
  vector<lower=0>[N] event_time; //failure time
  int group_size[S]; //group sizes
  matrix[S, K] X_predictors;//predictor variable matrix
}
transformed data{
  matrix[S, K] X_centered;
  vector[K] X_means;
  for(k0 in 1:K){
    X_means[k0] = mean(X_predictors[, k0]);
    X_centered[,k0] = X_predictors[, k0] - X_means[k0];
  }
}
parameters{
  real mu0; // hyperparameter
  real<lower=0> sigma0;// hyperparameter
  real<lower=0> beta;
  vector[K] R1_K; // fixed parameters
  vector[D] R0; // random intercept
}
transformed parameters{
  //vector[S] r_1_1 = ()
}
model{
  int position = 1;
  vector[S] theta_temp;

  for (s0 in 1:S){
    theta_temp[s0] = exp(R0[id[s0]] + X_centered[s0,]*R1_K);
  }

  for (s1 in 1:S){
    if(group_size[s1] == 0) {
      target += nhppnoevent_lp(tau[s1], beta, theta_temp[s1]);
    }else{
      segment(event_time, position, group_size[s1]) ~ nhpp(beta, theta_temp[s1], tau[s1]);
      position += group_size[s1];
    }
  }
  beta ~ gamma(1, 1);
  R0 ~ normal(mu0, sigma0);
  R1_K  ~ normal(0, 10);
  mu0 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
```

```
  //theta_temp ~ gamma(1, 0.01);
}
generated quantities{
  real mu0_true = mu0 - dot_product(X_means, R1_K);
  vector[D] R0_true = R0 - dot_product(X_means, R1_K);
  //real theta_correct = theta_temp - dot_product(X_centered, R1_K);
}
```

2 _log form

```
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;
  }
  real nhppnoevent_log(real tau, real beta, real theta){
    real loglikelihood = - (tau/theta)^beta;
    return(loglikelihood);
  }
}
data {
  int<lower=1> N; //total # of failures
  int<lower=1> K; //number of predictors
  int<lower=1> S; //total # of shifts
  int<lower=1> D; //total # of drivers
  int<lower=1> id[S];//driver index, must be an array
  vector<lower=0>[S] tau;//truncated time
  vector<lower=0>[N] event_time; //failure time
  int group_size[S]; //group sizes
  matrix[S, K] X_predictors;//predictor variable matrix
}
transformed data{
  matrix[S, K] X_centered;
  vector[K] X_means;
  for(k0 in 1:K){
    X_means[k0] = mean(X_predictors[, k0]);
    X_centered[,k0] = X_predictors[, k0] - X_means[k0];
  }
}
parameters{
  real mu0; // hyperparameter
  real<lower=0> sigma0;// hyperparameter
  real<lower=0> beta;
  vector[K] R1_K; // fixed parameters
  vector[D] R0; // random intercept
}
transformed parameters{
  //vector[S] r_1_1 = ()
}
model{
  int position = 1;
  vector[S] theta_temp;

  for (s0 in 1:S){
    theta_temp[s0] = exp(R0[id[s0]] + X_centered[s0,]*R1_K);
  }

  for (s1 in 1:S){
    if(group_size[s1] == 0) {
      tau[s1] ~ nhppnoevent(beta, theta_temp[s1]);
    }else{
      segment(event_time, position, group_size[s1]) ~ nhpp(beta, theta_temp[s1], tau[s1]);
      position += group_size[s1];
    }
  }
  beta ~ gamma(1, 1);
  R0 ~ normal(mu0, sigma0);
  R1_K  ~ normal(0, 10);
  mu0 ~ normal(0, 10);
  sigma0 ~ gamma(1, 1);
```

```
  //theta_temp ~ gamma(1, 0.01);
}
generated quantities{
  real mu0_true = mu0 - dot_product(X_means, R1_K);
  vector[D] R0_true = R0 - dot_product(X_means, R1_K);
  //real theta_correct = theta_temp - dot_product(X_centered, R1_K);
}
```

```
##   [1] -7.221453e-03  6.176758e-04 -8.903879e-04 -6.255816e-04  7.190347e-04
##   [6] -6.014367e-04 -1.645071e-03 -9.581580e-03 -1.171470e-02 -3.415722e-03
##  [11] -3.671379e-03 -6.407417e-03  1.001959e-02 -2.821101e-04 -3.864305e-03
##  [16] -4.750421e-03 -9.610046e-03 -3.009567e-03 -4.247267e-03 -1.815514e-03
##  [21] -4.273637e-03 -7.678390e-03 -9.116996e-04 -1.059586e-03 -7.336148e-03
##  [26]  1.700370e-04 -4.732822e-03  4.018976e-04 -2.760521e-03  4.824642e-03
##  [31]  5.953360e-03  5.178390e-04 -2.158975e-03 -3.097760e-03 -3.439021e-03
##  [36]  8.404735e-03  5.059903e-04 -3.547078e-03 -4.720049e-03  4.809879e-03
##  [41]  1.831566e-03 -1.078650e-02 -9.474132e-03 -2.473275e-03 -1.658102e-03
##  [46] -4.242975e-03  2.156887e-04  6.172791e-03 -1.707048e-03 -1.910010e-03
##  [51] -4.039101e-03 -1.470030e-02 -1.997182e-03 -4.920321e-03  7.524045e-04
##  [56]  8.979552e-03 -6.156459e-03 -5.800770e-04 -8.516586e-03 -1.064970e-02
##  [61] -2.350728e-03 -2.606386e-03 -5.342424e-03  1.108459e-02  7.828835e-04
##  [66] -2.799311e-03 -3.685427e-03 -8.545053e-03 -1.944573e-03 -3.182274e-03
##  [71] -7.505208e-04 -3.208644e-03 -6.613396e-03  1.532939e-04  5.407893e-06
##  [76] -6.271154e-03  1.235031e-03 -3.667828e-03  1.466891e-03 -1.695528e-03
##  [81]  5.889636e-03  7.018354e-03  1.582833e-03 -1.093982e-03 -2.032767e-03
##  [86] -2.374028e-03  9.469729e-03  1.570984e-03 -2.482084e-03 -3.655055e-03
##  [91]  5.874873e-03  2.896560e-03 -9.721502e-03 -8.409139e-03 -1.408282e-03
##  [96] -5.931087e-04 -3.177982e-03  1.280682e-03  7.237784e-03 -6.420543e-04
## [101] -8.450168e-04 -2.974108e-03 -1.363531e-02 -9.321888e-04 -3.855328e-03
## [106]  1.817398e-03  1.004455e-02
```

Nominal difference. I am suspecting there is no difference between using `_lp` or `_log` in this case.