Bayesian Logistic Regression

Maike Holthuijzen

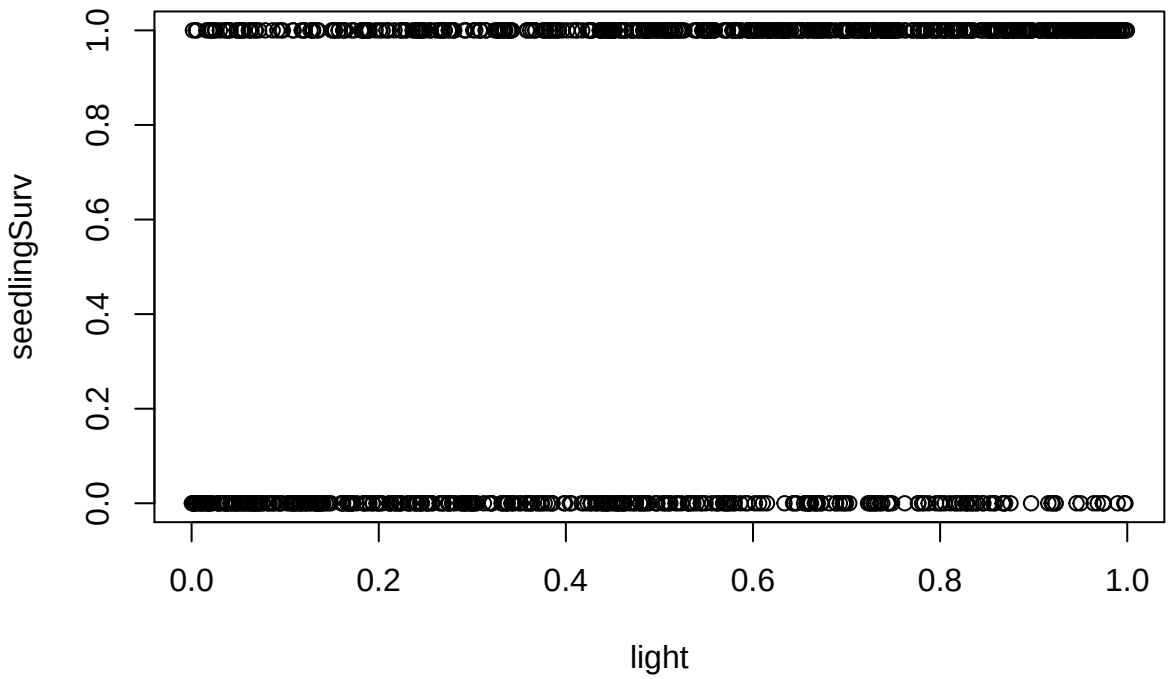
November 30, 2017

First, I used the `optim` function in R to estimate parameters for the logistic regression model. The function `mle.logreg` below estimates parameters using maximum likelihood.

```
#simulate data
set.seed(57)
beta0<- -1.0; beta1<-3.0
light<-runif(1000)
pVect<-exp(beta0+beta1*light)/(1+exp(beta0+beta1*light))
range(pVect)
```

```
## [1] 0.2690835 0.8806948
```

```
seedlingSurv<-rbinom(n=length(light),size=1,prob=pVect)

seedlingSurvData<-data.frame(light,seedlingSurv) # if you want a data frame
plot(light,seedlingSurv)
```




```
head(seedlingSurvData)
```

```
##        light seedlingSurv
## 1 0.24391435            1
```

```
## 2 0.51294954            1
## 3 0.03862843            1
## 4 0.16617658            0
## 5 0.73320525            0
## 6 0.66280162            1
```

```
#solve using maximum liklihood
mle.logreg = function(fmla, data)
{
  # negative log likelihood function
  logl <- function(theta,x,y){
    y <- y
    x <- as.matrix(x)
    beta <- theta[1:ncol(x)]

    # Use the log-likelihood of the Bernoulli distribution, where p is
    # defined as the logistic transformation of a linear combination
    loglik <- sum(-y*log(1 + exp(-(x%*%beta))) - (1-y)*log(1 + exp(x%*%beta)))
    return(-loglik)
  }

  # Prepare the data
  outcome = rownames(attr(terms(fmla),"factors"))[1]
  dfrTmp = model.frame(data)
  x = as.matrix(model.matrix(fmla, data=dfrTmp))
  y = as.numeric(as.matrix(data[,match(outcome,colnames(data))]))

  # Define initial values for the parameters
  theta.start = rep(0,(dim(x)[2]))
  names(theta.start) = colnames(x)

  # Calculate the maximum likelihood
  mle = optim(theta.start,logl,x=x,y=y,hessian=T)
  out = list(beta=mle$par,vcov=solve(mle$hessian),ll=2*mle$value)
}

fmla = as.formula("seedlingSurv~light")
mylogit = mle.logreg(fmla, seedlingSurvData) #Estimate coefficients
```

The parameter estimates for the intercept (`beta0`) and coefficient for β_1 are

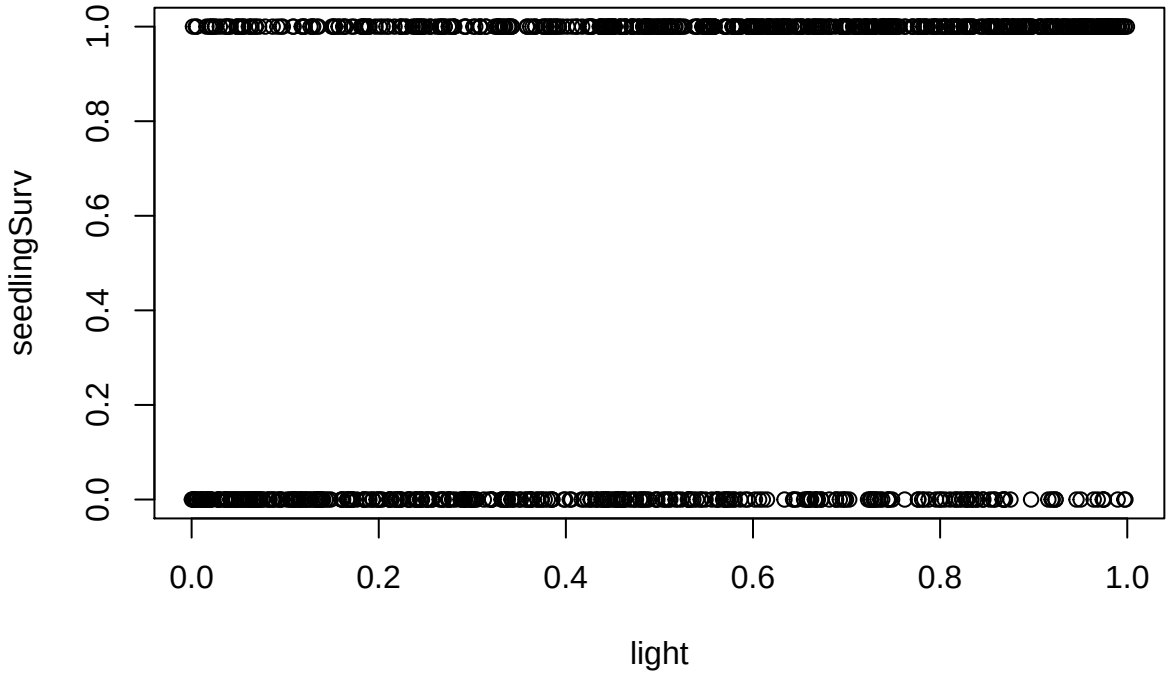
```
mylogit
```

```
## $beta
## (Intercept)       light
##  -0.9653514   2.8151933
##
## $vcov
##             (Intercept)       light
## (Intercept)  0.01841652 -0.02908678
## light       -0.02908678  0.06236447
##
## $ll
## [1] 1199.66
```

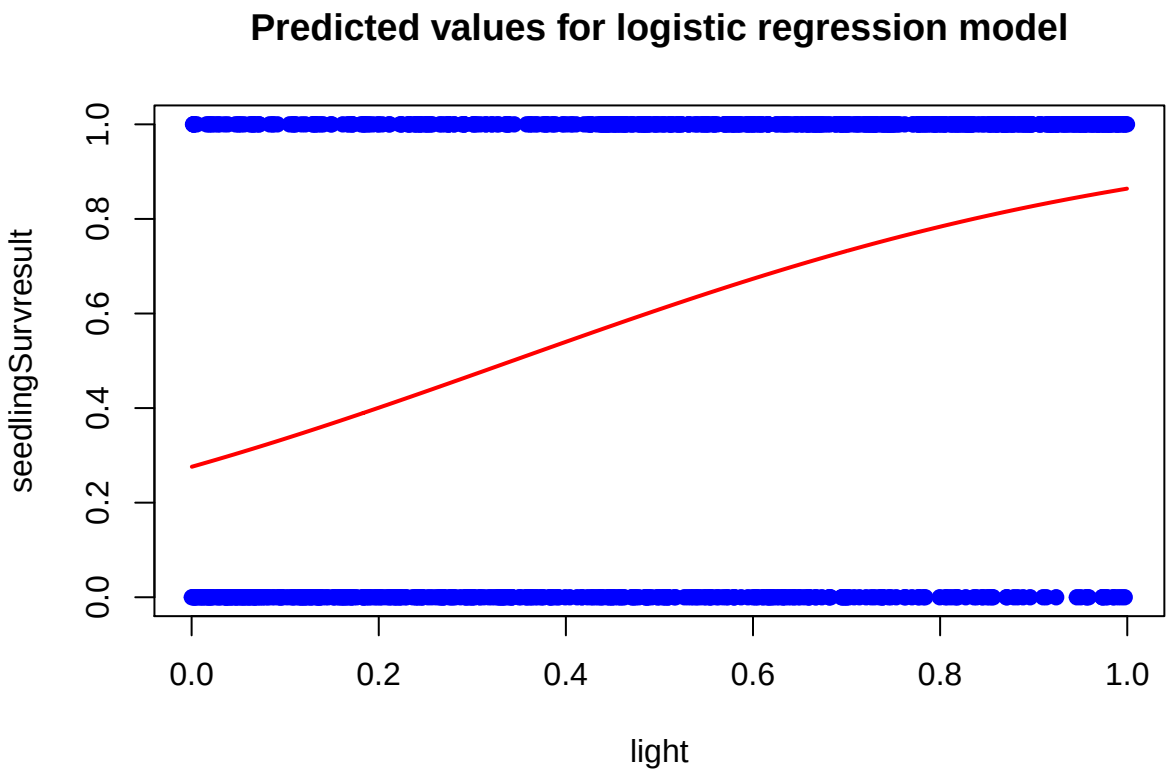
The estimates are fairly close to the actual values (-1 and 3 for the slope and intercept, respectively.). The

plot below shows a plot of the fitted data.

```
lightPreds<-light[order(light)]
plot(light,seedlingSurv)
```




```
outresult<-exp(mylogit[[1]][1]+mylogit[[1]][2]*light)/(1+exp(mylogit[[1]][1]+mylogit[[1]][2]*light))

seedlingSurvresult<-rbinom(n=length(light),size=1,prob=outresult)

plot(light,seedlingSurvresult, pch = 19, col = "blue", main = "Predicted values for logistic regression
f = function (x) 1 / -(1 + exp(mylogit[[1]][1] + mylogit[[1]][2] * x))
f2 = function (x) exp(mylogit[[1]][1]+mylogit[[1]][2]*x)/(1+exp(mylogit[[1]][1]+mylogit[[1]][2]*x))

lines(lightPreds, f2(lightPreds), col = "red", lwd= 2)
```



We can also compare these results to the output from R's `glm` function.

```
#compare to glm output
outGlmLogReg = glm(seedlingSurv~light, family=binomial,data=seedlingSurvData)
summary(outGlmLogReg)
```

```
##
## Call:
## glm(formula = seedlingSurv ~ light, family = binomial, data = seedlingSurvData)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.9956  -1.0364   0.6186   0.9209   1.6033
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.9656     0.1357  -7.115 1.12e-12 ***
## light         2.8156     0.2497  11.274  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1346.0  on 999  degrees of freedom
## Residual deviance: 1199.7  on 998  degrees of freedom
## AIC: 1203.7
##
```

```
## Number of Fisher Scoring iterations: 4
```

Next, I used the `brms` library for the logistic regression model. Again, the results are about the same as those obtained from the maximum likelihood estimates.

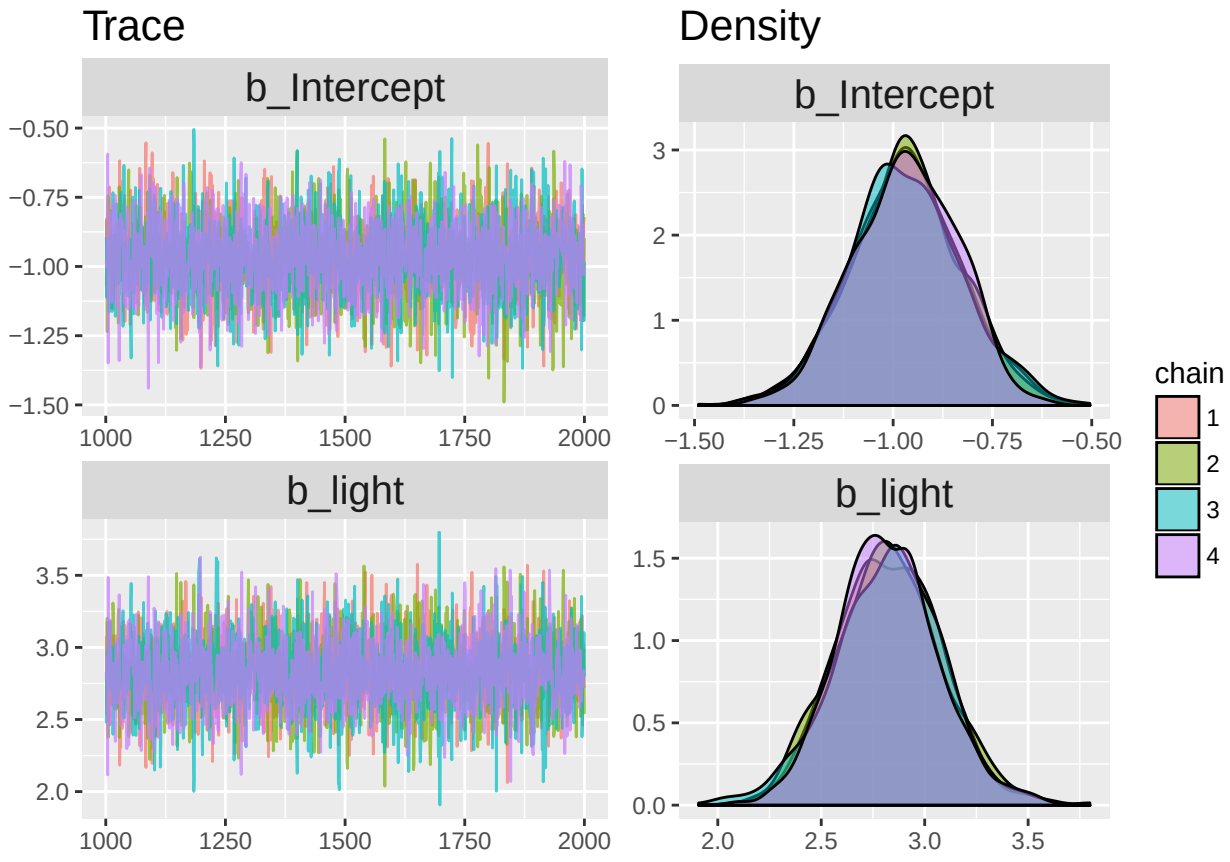
```
## Loading required package: ggplot2

## Loading required package: StanHeaders

## rstan (Version 2.16.2, packaged: 2017-07-03 09:24:58 UTC, GitRev: 2e1f913d3ca3)

## For execution on a local, multicore CPU with excess RAM we recommend calling
## rstan_options(auto_write = TRUE)
## options(mc.cores = parallel::detectCores())

## Loading 'brms' package (version 1.1.0). Useful instructions
## can be found by typing help('brms'). A more detailed introduction
## to the package is available through vignette('brms').

##
## Attaching package: 'brms'

## The following object is masked from 'package:MicrosoftML':
##
##     categorical

## Compiling the C++ model

## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config.hpp:39:0,
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/math/tools/config.
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/m
##                  from file3d1428eb7d67.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config/compiler/gcc.hpp:186:0: warning:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/base.hp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array.hpp:21
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint/util
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint.hpp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/m
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/m
##                  from file3d1428eb7d67.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In static
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:42:43: war
##        typedef typename Array::index_range index_range;
##                                           ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:43:37: war
##        typedef typename Array::index index;
##                                     ^
```

```
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In stati
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:53:43: wa
##        typedef typename Array::index_range index_range;
##                                           ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:54:37: wa
##        typedef typename Array::index index;
##                                     ^
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/
##                  from file3d1428eb7d67.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
##      static void set_zero_all_adjoints() {
##                  ^
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.627 seconds (Warm-up)
##                0.568 seconds (Sampling)
##                1.195 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 2).
##
## Gradient evaluation took 0.001 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.645 seconds (Warm-up)
##                0.75 seconds (Sampling)
##                1.395 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 3).
##
## Gradient evaluation took 0.001 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 10 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.664 seconds (Warm-up)
##                0.6 seconds (Sampling)
##                1.264 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.651 seconds (Warm-up)
##                0.816 seconds (Sampling)
##                1.467 seconds (Total)
```

```
##  Family: bernoulli (logit)
## Formula: seedlingSurv ~ light
##    Data: seedlingSurvData (Number of observations: 1000)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##    WAIC: Not computed
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.97      0.14    -1.24    -0.69       3964    1
## light         2.82      0.25     2.34     3.32       3471    1
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```




```
## Compiling the C++ model
```

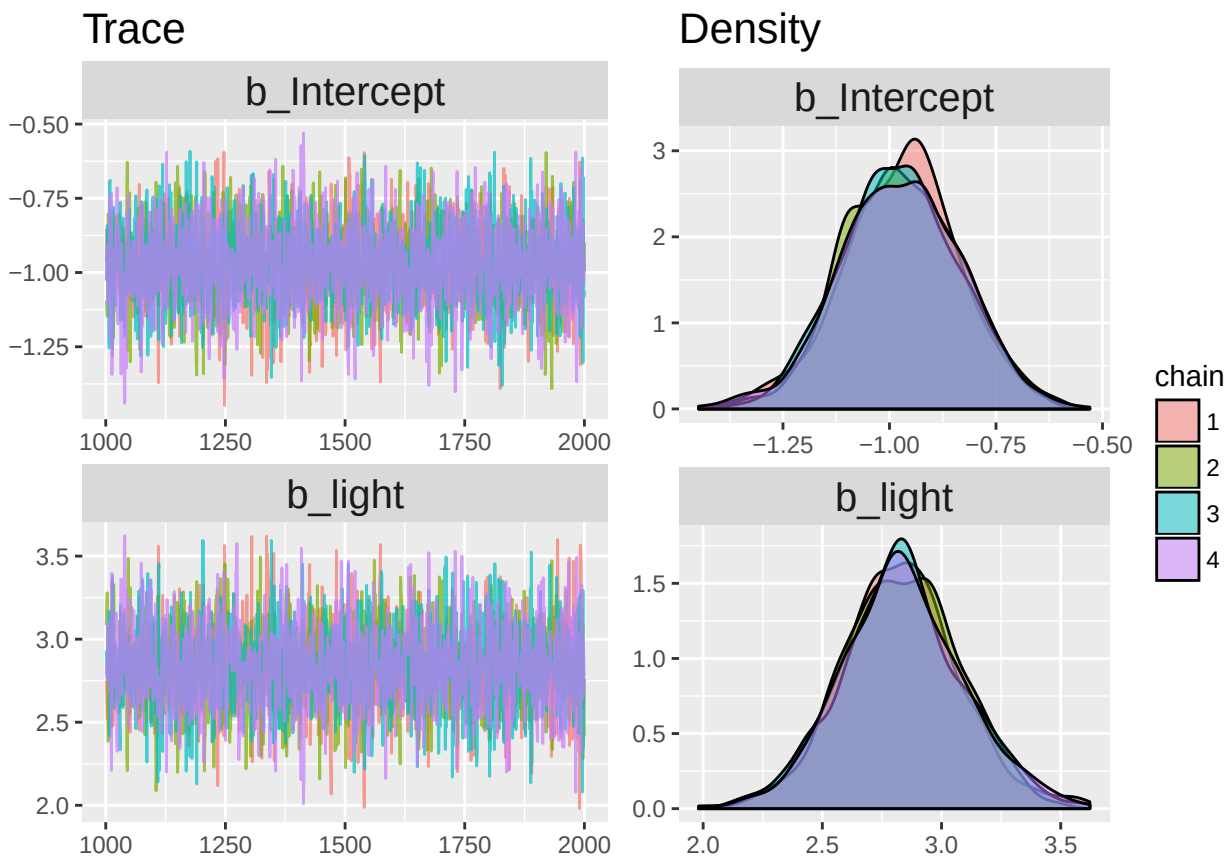
```
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config.hpp:39:0,
```

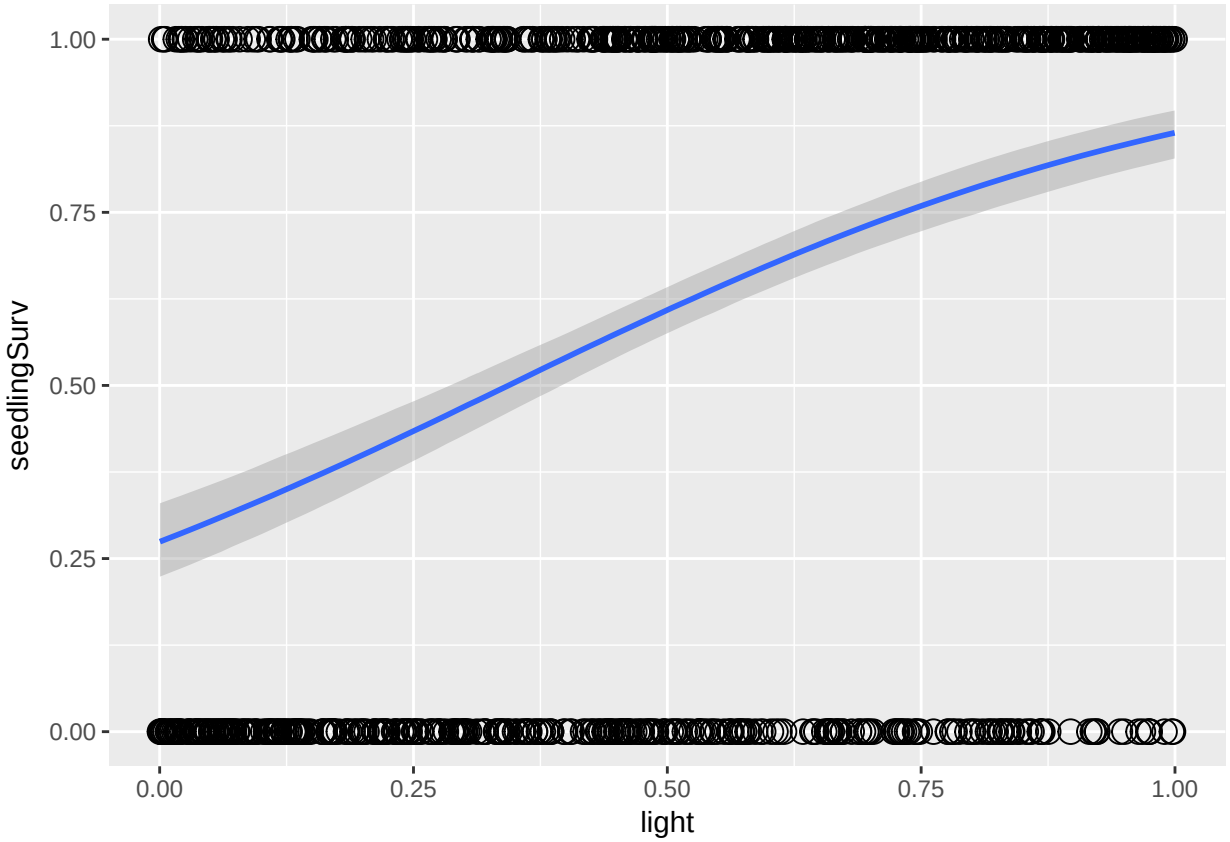
```
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/math/tools/config.
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/m
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/
##                  from file3d143fd4394a.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config/compiler/gcc.hpp:186:0: warning:
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/base.hp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array.hpp:21
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint/util
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint.hpp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/m
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/
##                  from file3d143fd4394a.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In stati
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:42:43: wa
##        typedef typename Array::index_range index_range;
##                                           ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:43:37: wa
##        typedef typename Array::index index;
##                                     ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In stati
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:53:43: wa
##        typedef typename Array::index_range index_range;
##                                           ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:54:37: wa
##        typedef typename Array::index index;
##                                     ^
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/
##                  from file3d143fd4394a.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
##      static void set_zero_all_adjoints() {
##                  ^
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
```

```
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.684 seconds (Warm-up)
##                0.733 seconds (Sampling)
##                1.417 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.633 seconds (Warm-up)
##                0.734 seconds (Sampling)
##                1.367 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
```

```
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.626 seconds (Warm-up)
##                0.517 seconds (Sampling)
##                1.143 seconds (Total)
##
##
## SAMPLING FOR MODEL 'bernoulli(logit) brms-model' NOW (CHAIN 4).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 2000 [  0%]  (Warmup)
## Iteration:  200 / 2000 [ 10%]  (Warmup)
## Iteration:  400 / 2000 [ 20%]  (Warmup)
## Iteration:  600 / 2000 [ 30%]  (Warmup)
## Iteration:  800 / 2000 [ 40%]  (Warmup)
## Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Iteration: 2000 / 2000 [100%]  (Sampling)
##
##  Elapsed Time: 0.659 seconds (Warm-up)
##                0.815 seconds (Sampling)
##                1.474 seconds (Total)
```

```
##  Family: bernoulli (logit)
## Formula: seedlingSurv ~ light
##    Data: seedlingSurvData (Number of observations: 1000)
## Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
##          total post-warmup samples = 4000
##    WAIC: Not computed
##
## Population-Level Effects:
##           Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
## Intercept    -0.97      0.14    -1.24    -0.71       3712    1
## light         2.83      0.25     2.34     3.34       3181    1
##
## Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
## is a crude measure of effective sample size, and Rhat is the potential
## scale reduction factor on split chains (at convergence, Rhat = 1).
```


Trace
Density
b_Intercept
b_light
−0.50
−0.75
−1.00
−1.25
1000
1250
1500
1750
2000
3.5
3.0
2.5
2.0
3
2
1
0
−1.25
−1.00
−0.75
−0.50
1.5
1.0
0.5
0.0
2.0
2.5
3.0
3.5
chain
1
2
3
4



Then, I used the `stanarm` library. Again, results were similar to the previous approaches.

```
library(rstanarm)
```

```
## Loading required package: Rcpp
## rstanarm (Version 2.12.1, packaged: 2016-09-12 13:08:24 UTC)
## - Do not expect the default priors to remain the same in future rstanarm versions.
## Thus, R scripts should specify priors explicitly, even if they are just the defaults.
## - For execution on a local, multicore CPU with excess RAM we recommend calling
## options(mc.cores = parallel::detectCores())
##
## Attaching package: 'rstanarm'
## The following object is masked from 'package:brms':
##
##     pp_check
```

```
outrstanarmLogReg<-stan_glm(seedlingSurv ~ light, data = seedlingSurvData,
                      family = binomial, iter=2000, warmup=1000, cores=4)
```

```
## trying deprecated constructor; please alert package maintainer
```

```
summary(outrstanarmLogReg)
```

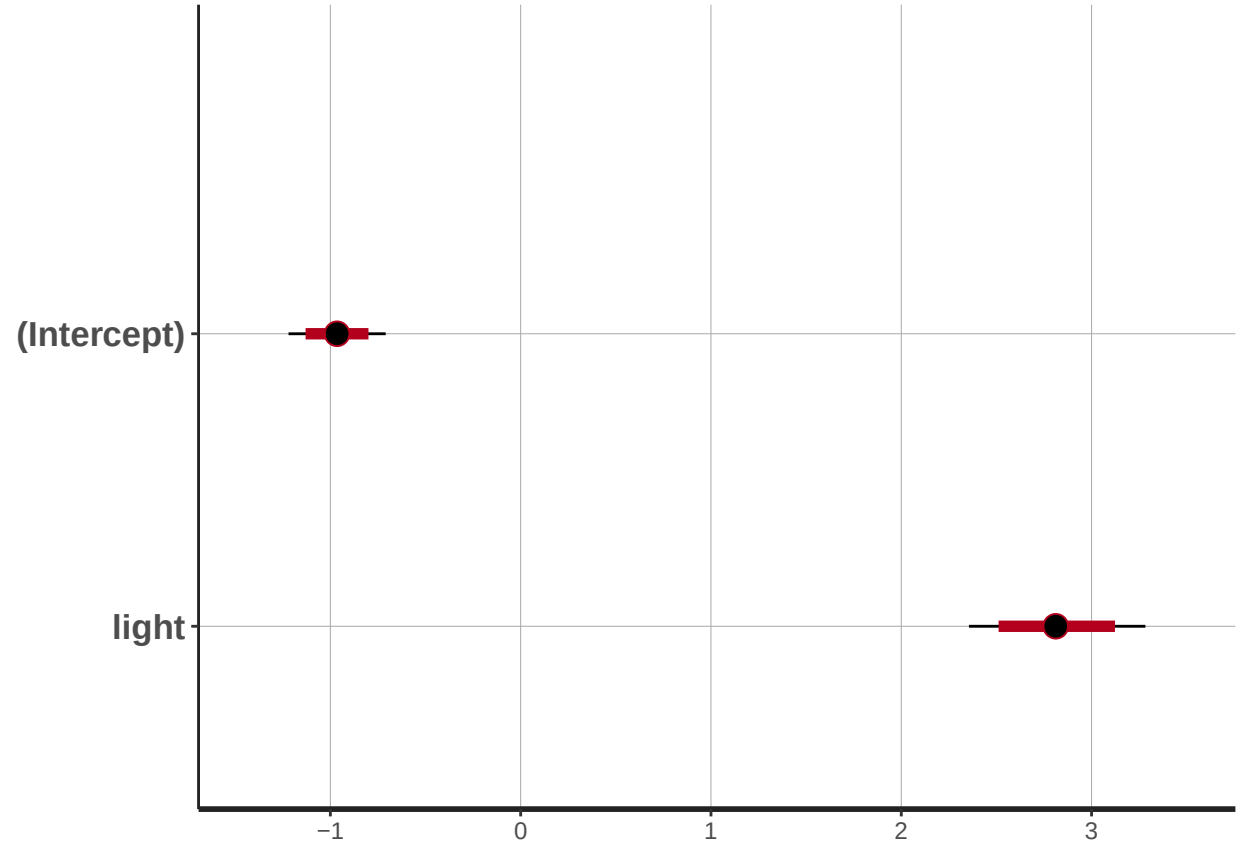
```
## stan_glm(formula = seedlingSurv ~ light, family = binomial, data = seedlingSurvData,
##     iter = 2000, warmup = 1000, cores = 4)
```

```
##
## Family: binomial (logit)
## Algorithm: sampling
## Posterior sample size: 4000
## Observations: 1000
##
## Estimates:
##                 mean   sd     2.5%   25%    50%    75%    97.5%
## (Intercept)     -1.0    0.1   -1.2   -1.1   -1.0   -0.9   -0.7
## light            2.8    0.2    2.4    2.7    2.8    3.0    3.3
## mean_PPD         0.6    0.0    0.6    0.6    0.6    0.6    0.6
## log-posterior -605.1    0.9 -607.7 -605.4 -604.8 -604.5 -604.2
##
## Diagnostics:
##               mcse Rhat n_eff
## (Intercept)   0.0  1.0  3852
## light         0.0  1.0  3369
## mean_PPD      0.0  1.0  3554
## log-posterior 0.0  1.0  2197
##
## For each parameter, mcse is Monte Carlo standard error, n_eff is a crude measure of effective sample
```

```
plot(outrstanarmLogReg)
```

```
## ci_level: 0.8 (80% intervals)
```

```
## outer_level: 0.95 (95% intervals)
```



Next, I used actual Stan code within the `rstan` library.

```
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config.hpp:39:0,
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/math/tools/config.
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/c
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/m
##                  from file3d1412be627d.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/config/compiler/gcc.hpp:186:0: warning: "
##  #  define BOOST_NO_CXX11_RVALUE_REFERENCES
##  ^
## <command-line>:0:0: note: this is the location of the previous definition
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/base.hp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array.hpp:21
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint/util
##                  from C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/numeric/odeint.hpp
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/a
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/prim/m
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/m
##                  from file3d1412be627d.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In static
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:42:43: war
##        typedef typename Array::index_range index_range;
##                                            ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:43:37: war
##        typedef typename Array::index index;
##                                      ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp: In static
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:53:43: war
##        typedef typename Array::index_range index_range;
##                                            ^
## C:/Users/Maike/Documents/R/win-library/3.3/BH/include/boost/multi_array/concept_checks.hpp:54:37: war
##        typedef typename Array::index index;
##                                      ^
## In file included from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/co
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/ma
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math.hpp:4
##                  from C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/src/stan/model/m
##                  from file3d1412be627d.cpp:8:
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoin
## C:/Users/Maike/Documents/R/win-library/3.3/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoin
##      static void set_zero_all_adjoints() {
##                  ^
##
## SAMPLING FOR MODEL 'fe004f478f8ad74110963f3663c16e41' NOW (CHAIN 1).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
```

```
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 3000 [  0%]  (Warmup)
## Iteration:  300 / 3000 [ 10%]  (Warmup)
## Iteration:  501 / 3000 [ 16%]  (Sampling)
## Iteration:  800 / 3000 [ 26%]  (Sampling)
## Iteration: 1100 / 3000 [ 36%]  (Sampling)
## Iteration: 1400 / 3000 [ 46%]  (Sampling)
## Iteration: 1700 / 3000 [ 56%]  (Sampling)
## Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Iteration: 3000 / 3000 [100%]  (Sampling)
##
##  Elapsed Time: 0.668 seconds (Warm-up)
##                2.751 seconds (Sampling)
##                3.419 seconds (Total)
##
##
## SAMPLING FOR MODEL 'fe004f478f8ad74110963f3663c16e41' NOW (CHAIN 2).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 3000 [  0%]  (Warmup)
## Iteration:  300 / 3000 [ 10%]  (Warmup)
## Iteration:  501 / 3000 [ 16%]  (Sampling)
## Iteration:  800 / 3000 [ 26%]  (Sampling)
## Iteration: 1100 / 3000 [ 36%]  (Sampling)
## Iteration: 1400 / 3000 [ 46%]  (Sampling)
## Iteration: 1700 / 3000 [ 56%]  (Sampling)
## Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Iteration: 3000 / 3000 [100%]  (Sampling)
##
##  Elapsed Time: 0.597 seconds (Warm-up)
##                2.485 seconds (Sampling)
##                3.082 seconds (Total)
##
##
## SAMPLING FOR MODEL 'fe004f478f8ad74110963f3663c16e41' NOW (CHAIN 3).
##
## Gradient evaluation took 0 seconds
## 1000 transitions using 10 leapfrog steps per transition would take 0 seconds.
## Adjust your expectations accordingly!
##
##
## Iteration:    1 / 3000 [  0%]  (Warmup)
```

```
## Iteration:  300 / 3000 [ 10%]  (Warmup)
## Iteration:  501 / 3000 [ 16%]  (Sampling)
## Iteration:  800 / 3000 [ 26%]  (Sampling)
## Iteration: 1100 / 3000 [ 36%]  (Sampling)
## Iteration: 1400 / 3000 [ 46%]  (Sampling)
## Iteration: 1700 / 3000 [ 56%]  (Sampling)
## Iteration: 2000 / 3000 [ 66%]  (Sampling)
## Iteration: 2300 / 3000 [ 76%]  (Sampling)
## Iteration: 2600 / 3000 [ 86%]  (Sampling)
## Iteration: 2900 / 3000 [ 96%]  (Sampling)
## Iteration: 3000 / 3000 [100%]  (Sampling)
##
##  Elapsed Time: 0.583 seconds (Warm-up)
##                2.767 seconds (Sampling)
##                3.35 seconds (Total)
```

```
## $summary
##             mean     se_mean        sd      2.5%       25%        50%
## beta0 -0.9722504 0.004824192 0.1321159 -1.224641 -1.064865 -0.9703983
## beta1  2.8291668 0.008657310 0.2370902  2.384171  2.662065  2.8343223
##              75%     97.5% n_eff      Rhat
## beta0 -0.8821296 -0.722219   750 0.9980000
## beta1  2.9876787  3.267216   750 0.9986143
##
## $c_summary
## , , chains = chain:1
##
##          stats
## parameter       mean        sd      2.5%       25%        50%        75%
##     beta0 -0.9691986 0.1327635 -1.215911 -1.067099 -0.9699606 -0.8671854
##     beta1  2.8275144 0.2411156  2.367019  2.656022  2.8317811  2.9974880
##          stats
## parameter     97.5%
##     beta0 -0.750219
##     beta1  3.283041
##
## , , chains = chain:2
##
##          stats
## parameter       mean        sd      2.5%       25%        50%        75%
##     beta0 -0.9753559 0.1332142 -1.213840 -1.070007 -0.9757403 -0.8862542
##     beta1  2.8367076 0.2375048  2.379688  2.669312  2.8282827  2.9955967
##          stats
## parameter      97.5%
##     beta0 -0.6999382
##     beta1  3.2781272
##
## , , chains = chain:3
##
##          stats
## parameter       mean        sd      2.5%       25%        50%       75%
##     beta0 -0.9721967 0.1308162 -1.256152 -1.050601 -0.9640484 -0.886754
##     beta1  2.8232784 0.2333386  2.411116  2.659894  2.8413074  2.968733
##          stats
```

```
## parameter      97.5%
##     beta0 -0.7258352
##     beta1  3.2409186
```

```
## Warning: package 'coda' was built under R version 3.3.3
```

```
##
## Attaching package: 'coda'
```

```
## The following object is masked from 'package:rstan':
##
##     traceplot
```

Lastly, I used the `rjags` library to fit the logistic regression model. The code is similar to the `rstan` model, but it is a little simpler. Again, the posterior distributions for the slope and intercept parameters show a high density around the actual parameters, and the trace plot (obtained via the `coda` library) shows that mixing within the MCMC chain was also sufficient

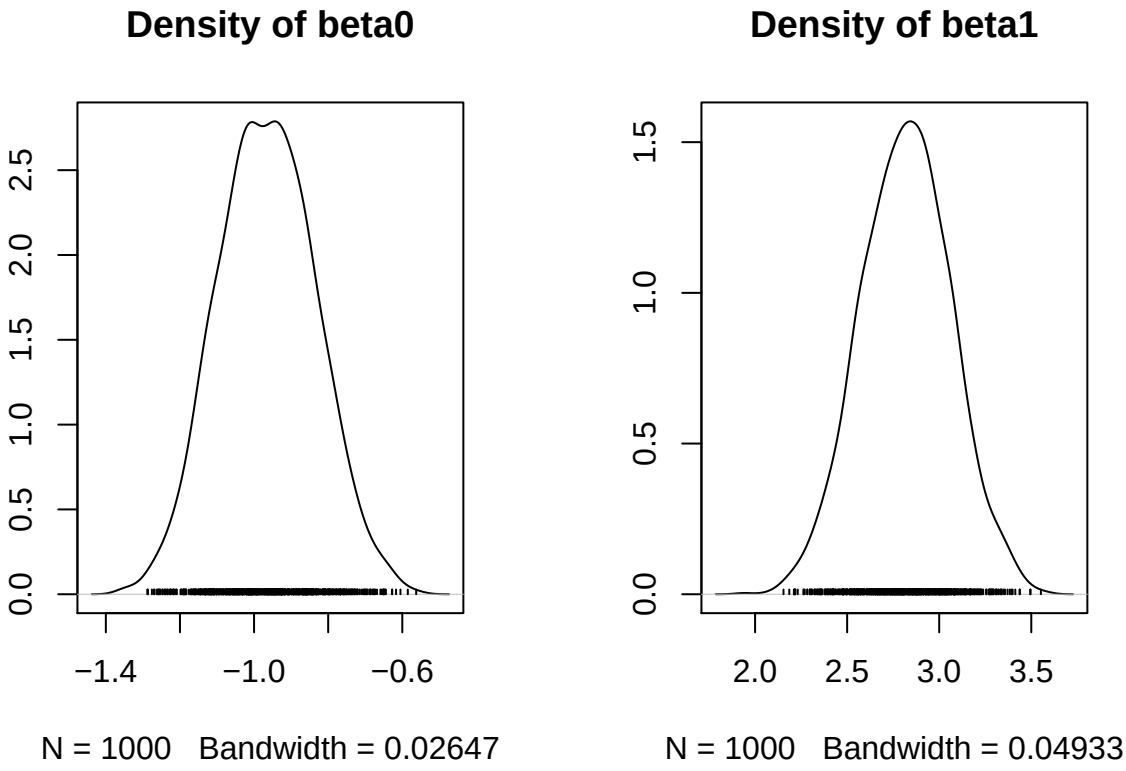
```
## Warning: package 'rjags' was built under R version 3.3.3
```

```
## Linked to JAGS 4.3.0
```

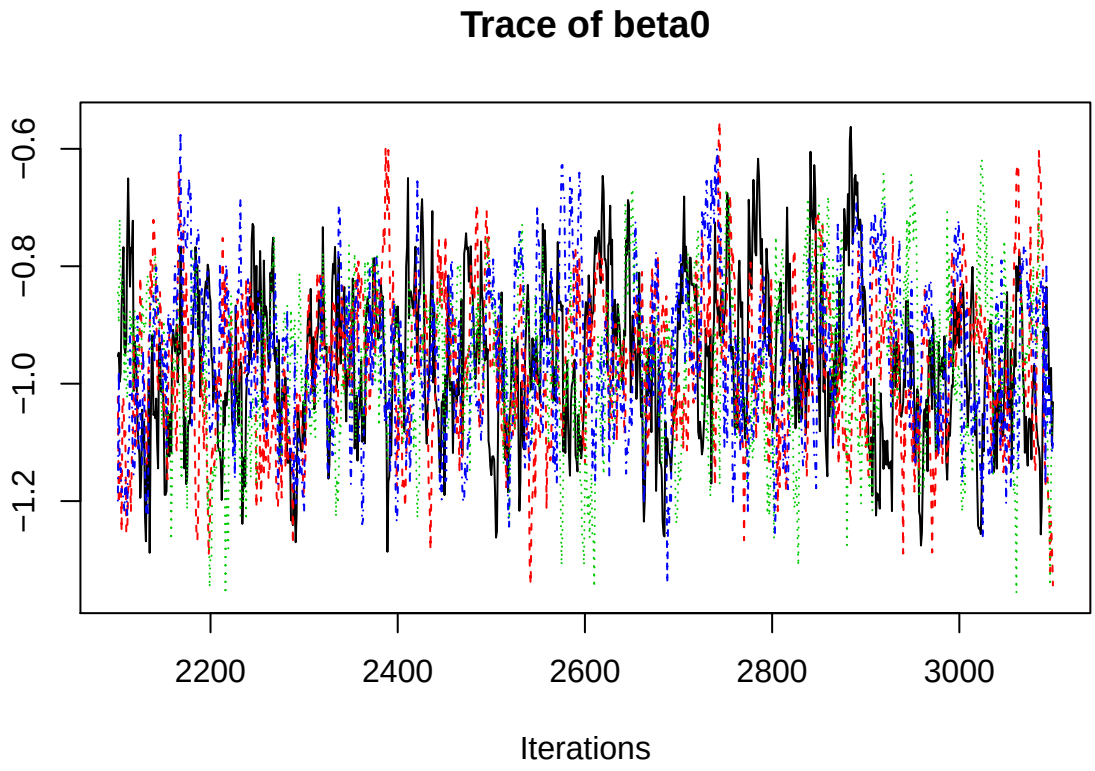
```
## Loaded modules: basemod,bugs
```

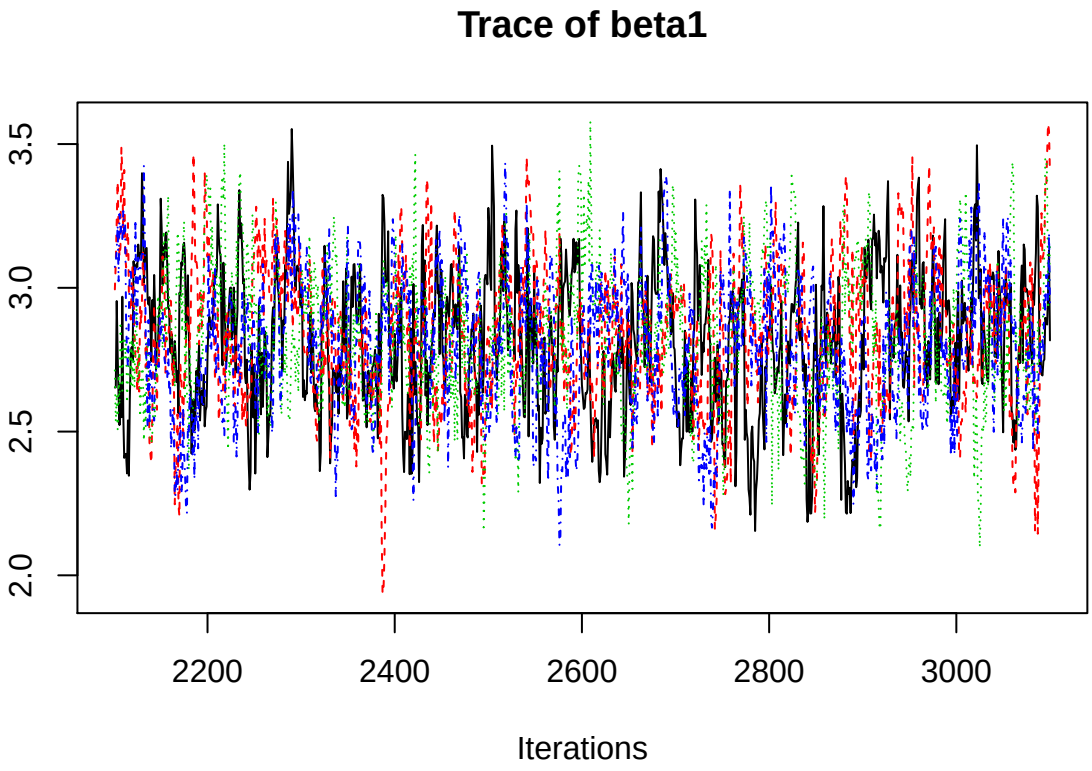
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 1000
##    Unobserved stochastic nodes: 2
##    Total graph size: 5005
##
## Initializing model
```

```
## $beta0
## mcarray:
## [1] -0.9746154
##
## Marginalizing over: iteration(1000),chain(4)
##
## $beta1
## mcarray:
## [1] 2.835911
##
## Marginalizing over: iteration(1000),chain(4)
```




```
##
## Iterations = 2101:3100
## Thinning interval = 1
## Number of chains = 4
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##          Mean     SD Naive SE Time-series SE
## beta0 -0.9651 0.1312 0.002074       0.006165
## beta1  2.8195 0.2445 0.003866       0.011588
##
## 2. Quantiles for each variable:
##
##         2.5%    25%     50%     75%   97.5%
## beta0 -1.216 -1.057 -0.9657 -0.8732 -0.7055
## beta1  2.338  2.651  2.8214  2.9864  3.3054
```


Trace of beta0
−0.6
−0.8
−1.0
−1.2
2200
2400
2600
2800
3000
Iterations


Trace of beta1
3.5
3.0
2.5
2.0
2200
2400
2600
2800
3000
Iterations