R Notebook

Question 3 Recall the putts data from class. The R notebook posted from December 2 presents two models: a logistic model and a "physical" model. Suppose we wished to incorporate the following prior expert opinion into our models: . My best guess for the probability of making a 5-foot putt is 52.5%; however, there is a 95% chance that this probability is below 71.6%. . My best guess for the probability of making a 10-foot putt is 27.8%; however, there is a 95% chance that this probability is below 40.8%.

a) Re-fit both models incorporating this prior information. Plot the data with 95% posterior credible bands for the probabilities of successful putts. (You do not need to provide posterior predictive bands.)

```
setwd("C:/Users/Palma/Desktop/MS STATS/Year2 Sem1/Bayesian/FinalTakehome")
library(rstan)
```

```
## Warning: package 'rstan' was built under R version 3.2.5

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.2.5

## Loading required package: StanHeaders

## Warning: package 'StanHeaders' was built under R version 3.2.5

## rstan (Version 2.12.1, packaged: 2016-09-11 13:07:50 UTC, GitRev: 85f7a56811da)

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

```
load("putts.RData")
set.seed(1202)
attach(putts)
putts
```

```
##    distance    n    y
## 1         2 1443 1346
## 2         3  694  577
## 3         4  455  337
## 4         5  353  208
## 5         6  272  149
## 6         7  256  136
## 7         8  240  111
## 8         9  217   69
## 9        10  200   67
## 10       11  237   75
## 11       12  202   52
## 12       13  192   46
## 13       14  174   54
## 14       15  167   28
## 15       16  201   27
## 16       17  195   31
## 17       18  191   33
## 18       19  147   20
## 19       20  152   24
```

Let's try to fit a simple logistic model to these data:

$$
\begin{aligned}
&y_x \overset{ind}{\sim} Binomial(n_x, p_x) \\
&\ln\left(\frac{p_x}{1-p_x}\right) = \beta_0 + \beta_1(x-7), \quad x \in \{2, 3, \ldots, 20\}.
\end{aligned}
$$

```
getthetaprior <- function(bg,hi){
# Interpreting bg as the mode of a Beta(a,b) distribution,
# we determine that b = (a-1+bg(2-a))/bg. Furthermore, we want
# the 95th percentile of this beta distribution to be hi.
f <- function(a) { pbeta(hi,a,(a-1-bg*(a-2))/bg)-.95 }
a <- uniroot(f,c(1,3000))$root
b <- (a-1-bg*(a-2))/bg
return(c(a,b))
}
  ab5 <- getthetaprior(.525,.716)
  ab10 <- getthetaprior(.278,.408)
```

Based on expert opinion, our priors for θ_5 and θ_{10} are as follows:

- $\theta_5 \sim Beta(8.74, 8.01)$.
- $\theta_{10} \sim Beta(12.1, 29.84)$.

How would we use these priors on different θs to induce priors on β_0 and β_1? For each simulated $(\theta_{10}\theta_5)$ pair, we have:

$$
\begin{aligned}
&\begin{pmatrix} F^{-1}(\theta_5) \\ F^{-1}(\theta_{10}) \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 1 & 3 \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} \\
&\implies \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix} = \begin{pmatrix} 1 & -2 \\ 1 & 3 \end{pmatrix}^{-1} \begin{pmatrix} F^{-1}(\theta_5) \\ F^{-1}(\theta_{10}) \end{pmatrix} = \begin{pmatrix} .6 & .4 \\ -.2 & .2 \end{pmatrix} \begin{pmatrix} F^{-1}(\theta_5) \\ F^{-1}(\theta_{10}) \end{pmatrix} = \begin{pmatrix} .6\,F^{-1}(\theta_5) + .4\,F^{-1}(\theta_{10}) \\ -.2\,F^{-1}(\theta_5) + .2\,F^{-1}(\theta_{10}) \end{pmatrix}
\end{aligned}
$$

```
solve(rbind(c(1,-2),c(1,3)))
```

```
##      [,1] [,2]
## [1,]  0.6  0.4
## [2,] -0.2  0.2
```

```
### Model 1 - "Logistic model"
set.seed(505)
mod1code <- '
  data{
    int<lower=1>         n[19];
    int<lower=0>         y[19];
    real<lower=0>        x[19];
  }
  parameters{
   real<lower=0,upper=1> theta5;  //
  real<lower=0,upper=1> theta10;  //
  }
 transformed parameters{
  real<lower=0,upper=1> p[19];
  real b0;
  real b1;
```

```
  // For logistic regression, Finv is the logit function
  b0 = .6*logit(theta5)+.4*logit(theta10);
  b1 = -.2*logit(theta5)+.2*logit(theta10);

    for(i in 1:19){
      p[i] = inv_logit(b0+b1*(x[i]-7));
    }
  }
  model{
  theta5 ~ beta(8.744599,8.007018);
  theta10 ~ beta(12.10458,29.83995);

for(i in 1:19){
      y[i] ~ binomial(n[i],p[i]);
  }
  }
'

mod1dat <- with(putts,list(n=n,y=y,x=distance))
mod1 <- stan(model_code=mod1code, data=mod1dat)
```

```
## C:/Users/Palma/Documents/R/win-library/3.2/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
##
## SAMPLING FOR MODEL 'b36b0a5bf1d518e09d327419d60610fc' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.057 seconds (Warm-up)
##                0.065 seconds (Sampling)
##                0.122 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b36b0a5bf1d518e09d327419d60610fc' NOW (CHAIN 2).
##
## Chain 2, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2, Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
## Chain 2, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.063 seconds (Warm-up)
##                0.053 seconds (Sampling)
##                0.116 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b36b0a5bf1d518e09d327419d60610fc' NOW (CHAIN 3).
##
## Chain 3, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.062 seconds (Warm-up)
##                0.056 seconds (Sampling)
##                0.118 seconds (Total)
##
##
## SAMPLING FOR MODEL 'b36b0a5bf1d518e09d327419d60610fc' NOW (CHAIN 4).
##
## Chain 4, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.048 seconds (Warm-up)
##                0.065 seconds (Sampling)
##                0.113 seconds (Total)
```

```r
print(mod1,digits=5)
```

```
## Inference for Stan model: b36b0a5bf1d518e09d327419d60610fc.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##                mean se_mean      sd      2.5%       25%       50%
## theta5      0.71992 0.00014 0.00684   0.70661   0.71526   0.71981
## theta10     0.41664 0.00018 0.00857   0.39943   0.41090   0.41675
## p[1]        0.84711 0.00011 0.00605   0.83485   0.84314   0.84718
## p[2]        0.81094 0.00012 0.00641   0.79824   0.80666   0.81092
```

```
## p[3]        0.76855 0.00013 0.00666     0.75554     0.76410     0.76851
## p[4]        0.71992 0.00014 0.00684     0.70661     0.71526     0.71981
## p[5]        0.66551 0.00015 0.00699     0.65199     0.66078     0.66544
## p[6]        0.60631 0.00016 0.00722     0.59230     0.60133     0.60631
## p[7]        0.54381 0.00016 0.00758     0.52881     0.53856     0.54379
## p[8]        0.47990 0.00017 0.00806     0.46421     0.47449     0.48000
## p[9]        0.41664 0.00018 0.00857     0.39943     0.41090     0.41675
## p[10]       0.35603 0.00018 0.00897     0.33820     0.35013     0.35619
## p[11]       0.29971 0.00018 0.00917     0.28177     0.29370     0.29996
## p[12]       0.24887 0.00018 0.00912     0.23098     0.24284     0.24912
## p[13]       0.20415 0.00017 0.00881     0.18673     0.19825     0.20425
## p[14]       0.16569 0.00015 0.00828     0.14960     0.16011     0.16568
## p[15]       0.13328 0.00014 0.00760     0.11857     0.12807     0.13325
## p[16]       0.10640 0.00012 0.00683     0.09310     0.10182     0.10638
## p[17]       0.08442 0.00011 0.00603     0.07271     0.08039     0.08435
## p[18]       0.06665 0.00009 0.00524     0.05661     0.06311     0.06654
## p[19]       0.05240 0.00008 0.00450     0.04376     0.04941     0.05226
## b0          0.43191 0.00066 0.03027     0.37349     0.41102     0.43184
## b1         -0.25620 0.00011 0.00673    -0.26957    -0.26073    -0.25632
## lp__    -3060.90927 0.02121 0.95847 -3063.46403 -3061.26632 -3060.60219
##                 75%       97.5% n_eff    Rhat
## theta5      0.72434     0.73367  2356 0.99961
## theta10     0.42212     0.43381  2286 0.99924
## p[1]        0.85130     0.85861  2826 1.00007
## p[2]        0.81533     0.82318  2684 0.99996
## p[3]        0.77293     0.78160  2523 0.99980
## p[4]        0.72434     0.73367  2356 0.99961
## p[5]        0.67002     0.67980  2211 0.99939
## p[6]        0.61102     0.62116  2122 0.99921
## p[7]        0.54873     0.55905  2111 0.99912
## p[8]        0.48512     0.49612  2174 0.99915
## p[9]        0.42212     0.43381  2286 0.99924
## p[10]       0.36188     0.37399  2419 0.99937
## p[11]       0.30575     0.31801  2550 0.99950
## p[12]       0.25496     0.26695  2669 0.99962
## p[13]       0.21004     0.22167  2771 0.99971
## p[14]       0.17123     0.18219  2856 0.99979
## p[15]       0.13833     0.14844  2927 0.99986
## p[16]       0.11090     0.12016  2985 0.99992
## p[17]       0.08836     0.09668  3033 0.99996
## p[18]       0.07009     0.07739  3073 1.00000
## p[19]       0.05533     0.06170  3106 1.00004
## b0          0.45162     0.49448  2122 0.99921
## b1         -0.25163    -0.24289  3435 1.00029
## lp__    -3060.21742 -3059.96778  2043 1.00016
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 12 19:02:59 2016.
## 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).
```

```
# Print and plot the results
mod1
```

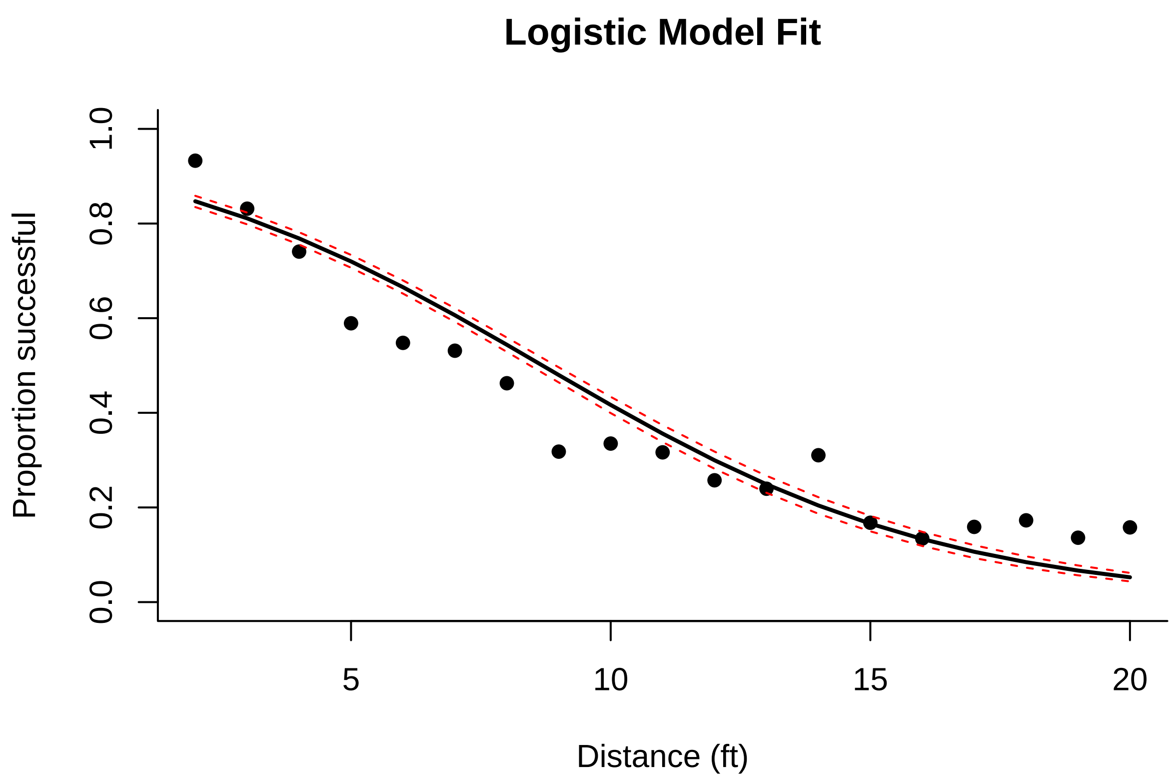
```
## Inference for Stan model: b36b0a5bf1d518e09d327419d60610fc.
```

```
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##             mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## theta5      0.72    0.00 0.01     0.71     0.72     0.72     0.72     0.73
## theta10     0.42    0.00 0.01     0.40     0.41     0.42     0.42     0.43
## p[1]        0.85    0.00 0.01     0.83     0.84     0.85     0.85     0.86
## p[2]        0.81    0.00 0.01     0.80     0.81     0.81     0.82     0.82
## p[3]        0.77    0.00 0.01     0.76     0.76     0.77     0.77     0.78
## p[4]        0.72    0.00 0.01     0.71     0.72     0.72     0.72     0.73
## p[5]        0.67    0.00 0.01     0.65     0.66     0.67     0.67     0.68
## p[6]        0.61    0.00 0.01     0.59     0.60     0.61     0.61     0.62
## p[7]        0.54    0.00 0.01     0.53     0.54     0.54     0.55     0.56
## p[8]        0.48    0.00 0.01     0.46     0.47     0.48     0.49     0.50
## p[9]        0.42    0.00 0.01     0.40     0.41     0.42     0.42     0.43
## p[10]       0.36    0.00 0.01     0.34     0.35     0.36     0.36     0.37
## p[11]       0.30    0.00 0.01     0.28     0.29     0.30     0.31     0.32
## p[12]       0.25    0.00 0.01     0.23     0.24     0.25     0.25     0.27
## p[13]       0.20    0.00 0.01     0.19     0.20     0.20     0.21     0.22
## p[14]       0.17    0.00 0.01     0.15     0.16     0.17     0.17     0.18
## p[15]       0.13    0.00 0.01     0.12     0.13     0.13     0.14     0.15
## p[16]       0.11    0.00 0.01     0.09     0.10     0.11     0.11     0.12
## p[17]       0.08    0.00 0.01     0.07     0.08     0.08     0.09     0.10
## p[18]       0.07    0.00 0.01     0.06     0.06     0.07     0.07     0.08
## p[19]       0.05    0.00 0.00     0.04     0.05     0.05     0.06     0.06
## b0          0.43    0.00 0.03     0.37     0.41     0.43     0.45     0.49
## b1         -0.26    0.00 0.01    -0.27    -0.26    -0.26    -0.25    -0.24
## lp__    -3060.91    0.02 0.96 -3063.46 -3061.27 -3060.60 -3060.22 -3059.97
##         n_eff Rhat
## theta5   2356    1
## theta10  2286    1
## p[1]     2826    1
## p[2]     2684    1
## p[3]     2523    1
## p[4]     2356    1
## p[5]     2211    1
## p[6]     2122    1
## p[7]     2111    1
## p[8]     2174    1
## p[9]     2286    1
## p[10]    2419    1
## p[11]    2550    1
## p[12]    2669    1
## p[13]    2771    1
## p[14]    2856    1
## p[15]    2927    1
## p[16]    2985    1
## p[17]    3033    1
## p[18]    3073    1
## p[19]    3106    1
## b0       2122    1
## b1       3435    1
## lp__     2043    1
##
```

```
## Samples were drawn using NUTS(diag_e) at Mon Dec 12 19:02:59 2016.
## 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).
```

```r
p <- extract(mod1)$p

with(putts,plot(y/n~distance,pch=16,
                main="Logistic Model Fit",
                ylim=c(0,1),bty="l",
                xlab="Distance (ft)",
                ylab="Proportion successful"))
lines(2:20,colMeans(p),lwd=2)
lines(2:20,apply(p,2,quantile,.025),col="red",lty=2)
lines(2:20,apply(p,2,quantile,.975),col="red",lty=2)
```



$$p_x = P\left[|\theta| \leq \sin^{-1}\left(\frac{R-r}{x}\right)\right] = P\left[|Z| \leq \frac{1}{\sigma}\sin^{-1}\left(\frac{R-r}{x}\right)\right] = 2\Phi\left(\frac{1}{\sigma}\sin^{-1}\left(\frac{R-r}{x}\right)\right) - 1,$$

where Z is a standard normal random variable and Φ is the CDF of the standard normal distribution.

Prior on σ

Our initial prior distribution on the parameter σ is $\sigma \sim Uniform\left(0, \frac{\pi}{6}\right)$. Using expert inofrmation My best guess for the probability of making a 10-foot putt is 27.8%; however, there is a 95% chance that this probability is below 40.8%. To incorporate prior information about σ we can manipulate this information and obtain the result below;

$$P\left[2\Phi\left(\frac{1}{\sigma}\sin^{-1}\left(\frac{R-r}{x}\right)\right)-1\leq .408\right]=0.95$$

We solve and obtain an expression for σ

$$P\left[\frac{1}{\sigma}\leq\frac{\Phi^{-1}(0.704)}{\sin^{-1}\left(\frac{R-r}{10}\right)}\right]=0.95$$

```
##compute sigma95

sigma95= 1/(qnorm(.5*(0.408+1))*1/(asin(0.1*(4.25/24-1.68/24))))
```

$$\implies \quad P(\sigma > 0.02) = .95.$$
$$\implies \quad 1 - F_{\sigma}(sigma95) = .95.$$

Thus since the prior of σ is a uniform distribution we can obtain apporpriate bounds on the uniform distribution, so our informative prior distirbution on σ is $Uniform(0, 0.3996)$. That is the distance of b-a of a Uniform (a,b) distribution.

```
### Model 2 - "Physical model"
set.seed(507)
mod2code <- '
  data{
    real<lower=0>         smallr;
    real<lower=smallr>    bigR;
    int<lower=1>          n[19];
    int<lower=0>          y[19];
    real<lower=0>         x[19];
  }
  parameters{
    real<lower=0> sigma;
  }
  transformed parameters{
    real<lower=0,upper=1> p[19];

    for(i in 1:19){
      p[i] = 2*Phi(asin((bigR-smallr)/x[i])/sigma) - 1;
    }
  }
  model{
    sigma ~ uniform(0,.3996);
    for(i in 1:19){
      y[i] ~ binomial(n[i],p[i]);
    }
  }
'
smallr <- 1.68/24   ## A reg golf ball has diameter 1.68 in; we convert to ft
bigR <- 4.25/24     ## A reg golf hole has diameter 4.25 in; we convert to feet

mod2dat <- with(putts,list(smallr=smallr,bigR=bigR,y=y,n=n,x=distance))
mod2 <- stan(model_code=mod2code,data=mod2dat)
```

```
## C:/Users/Palma/Documents/R/win-library/3.2/StanHeaders/include/stan/math/rev/core/set_zero_all_adjoi
##
## SAMPLING FOR MODEL '262763cb6f2fc7a501460c6793e01a54' NOW (CHAIN 1).
##
## Chain 1, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 1, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 1, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 1, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 1, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 1, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 1, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 1, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 1, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.052 seconds (Warm-up)
##                0.031 seconds (Sampling)
##                0.083 seconds (Total)
##
##
## SAMPLING FOR MODEL '262763cb6f2fc7a501460c6793e01a54' NOW (CHAIN 2).
##
## Chain 2, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 2, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 2, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 2, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 2, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 2, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 2, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 2, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 2, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.049 seconds (Warm-up)
##                0.047 seconds (Sampling)
##                0.096 seconds (Total)
##
##
## SAMPLING FOR MODEL '262763cb6f2fc7a501460c6793e01a54' NOW (CHAIN 3).
##
## Chain 3, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 3, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 3, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 3, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 3, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 3, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 3, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 3, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 3, Iteration: 2000 / 2000 [100%]  (Sampling)
```

```
##  Elapsed Time: 0.053 seconds (Warm-up)
##                0.031 seconds (Sampling)
##                0.084 seconds (Total)
##
##
## SAMPLING FOR MODEL '262763cb6f2fc7a501460c6793e01a54' NOW (CHAIN 4).
##
## Chain 4, Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4, Iteration:  200 / 2000 [ 10%]  (Warmup)
## Chain 4, Iteration:  400 / 2000 [ 20%]  (Warmup)
## Chain 4, Iteration:  600 / 2000 [ 30%]  (Warmup)
## Chain 4, Iteration:  800 / 2000 [ 40%]  (Warmup)
## Chain 4, Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4, Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4, Iteration: 1200 / 2000 [ 60%]  (Sampling)
## Chain 4, Iteration: 1400 / 2000 [ 70%]  (Sampling)
## Chain 4, Iteration: 1600 / 2000 [ 80%]  (Sampling)
## Chain 4, Iteration: 1800 / 2000 [ 90%]  (Sampling)
## Chain 4, Iteration: 2000 / 2000 [100%]  (Sampling)
##  Elapsed Time: 0.038 seconds (Warm-up)
##                0.047 seconds (Sampling)
##                0.085 seconds (Total)
```

```r
print(mod2)
```

```
## Inference for Stan model: 262763cb6f2fc7a501460c6793e01a54.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##          mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## sigma    0.03    0.00 0.00     0.03     0.03     0.03     0.03     0.03
## p[1]     0.96    0.00 0.00     0.95     0.95     0.96     0.96     0.96
## p[2]     0.82    0.00 0.01     0.81     0.82     0.82     0.82     0.83
## p[3]     0.68    0.00 0.01     0.67     0.68     0.68     0.69     0.70
## p[4]     0.58    0.00 0.01     0.57     0.57     0.58     0.58     0.59
## p[5]     0.50    0.00 0.01     0.48     0.49     0.50     0.50     0.51
## p[6]     0.43    0.00 0.01     0.42     0.43     0.43     0.44     0.45
## p[7]     0.38    0.00 0.01     0.37     0.38     0.38     0.39     0.39
## p[8]     0.34    0.00 0.00     0.34     0.34     0.34     0.35     0.35
## p[9]     0.31    0.00 0.00     0.30     0.31     0.31     0.31     0.32
## p[10]    0.29    0.00 0.00     0.28     0.28     0.29     0.29     0.29
## p[11]    0.26    0.00 0.00     0.26     0.26     0.26     0.26     0.27
## p[12]    0.24    0.00 0.00     0.24     0.24     0.24     0.24     0.25
## p[13]    0.23    0.00 0.00     0.22     0.22     0.23     0.23     0.23
## p[14]    0.21    0.00 0.00     0.21     0.21     0.21     0.21     0.22
## p[15]    0.20    0.00 0.00     0.19     0.20     0.20     0.20     0.20
## p[16]    0.19    0.00 0.00     0.18     0.18     0.19     0.19     0.19
## p[17]    0.18    0.00 0.00     0.17     0.17     0.18     0.18     0.18
## p[18]    0.17    0.00 0.00     0.16     0.17     0.17     0.17     0.17
## p[19]    0.16    0.00 0.00     0.15     0.16     0.16     0.16     0.16
## lp__  -2926.75    0.02 0.68 -2928.72 -2926.91 -2926.49 -2926.31 -2926.26
##       n_eff Rhat
## sigma  2888    1
## p[1]   2879    1
## p[2]   2888    1
```

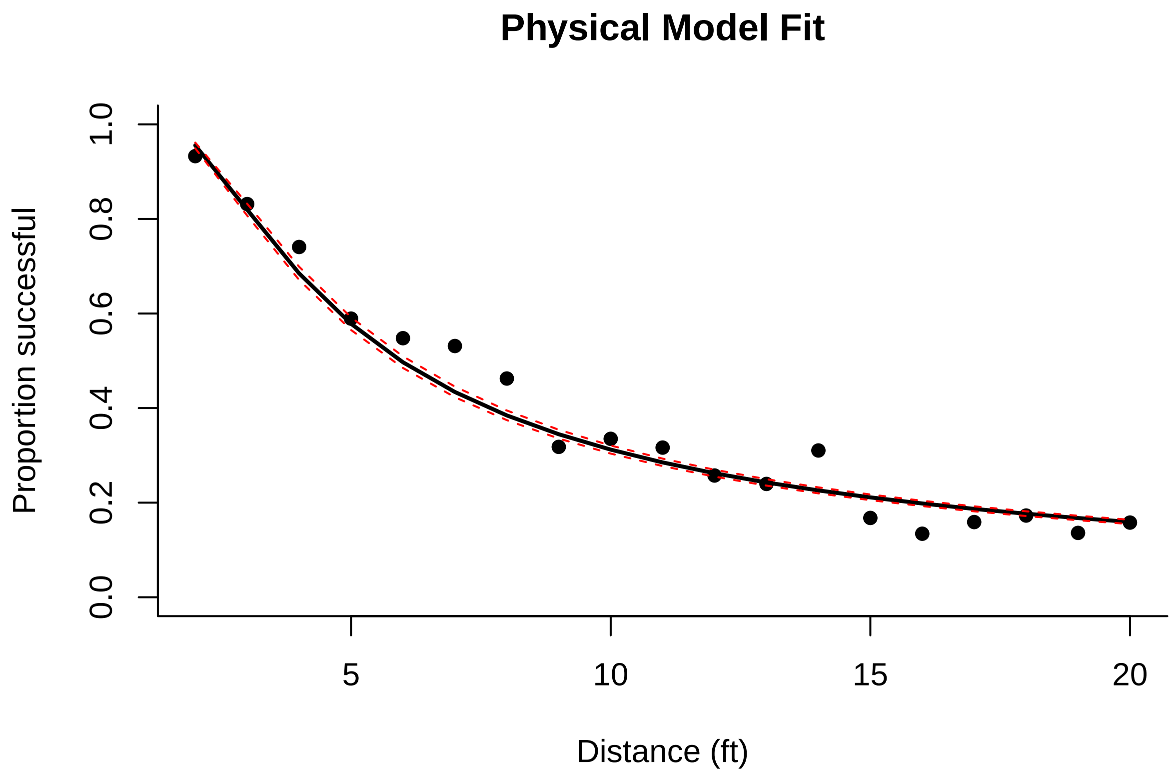
```
## p[3]   2891    1
## p[4]   2893    1
## p[5]   2894    1
## p[6]   2894    1
## p[7]   2894    1
## p[8]   2895    1
## p[9]   2895    1
## p[10]  2895    1
## p[11]  2895    1
## p[12]  2895    1
## p[13]  2895    1
## p[14]  2895    1
## p[15]  2895    1
## p[16]  2895    1
## p[17]  2895    1
## p[18]  2895    1
## p[19]  2895    1
## lp__   1815    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 12 19:03:42 2016.
## 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).
```

```
mod2
```

```
## Inference for Stan model: 262763cb6f2fc7a501460c6793e01a54.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##            mean se_mean   sd     2.5%      25%      50%      75%    97.5%
## sigma      0.03    0.00 0.00     0.03     0.03     0.03     0.03     0.03
## p[1]       0.96    0.00 0.00     0.95     0.95     0.96     0.96     0.96
## p[2]       0.82    0.00 0.01     0.81     0.82     0.82     0.82     0.83
## p[3]       0.68    0.00 0.01     0.67     0.68     0.68     0.69     0.70
## p[4]       0.58    0.00 0.01     0.57     0.57     0.58     0.58     0.59
## p[5]       0.50    0.00 0.01     0.48     0.49     0.50     0.50     0.51
## p[6]       0.43    0.00 0.01     0.42     0.43     0.43     0.44     0.45
## p[7]       0.38    0.00 0.01     0.37     0.38     0.38     0.39     0.39
## p[8]       0.34    0.00 0.00     0.34     0.34     0.34     0.35     0.35
## p[9]       0.31    0.00 0.00     0.30     0.31     0.31     0.31     0.32
## p[10]      0.29    0.00 0.00     0.28     0.28     0.29     0.29     0.29
## p[11]      0.26    0.00 0.00     0.26     0.26     0.26     0.26     0.27
## p[12]      0.24    0.00 0.00     0.24     0.24     0.24     0.24     0.25
## p[13]      0.23    0.00 0.00     0.22     0.22     0.23     0.23     0.23
## p[14]      0.21    0.00 0.00     0.21     0.21     0.21     0.21     0.22
## p[15]      0.20    0.00 0.00     0.19     0.20     0.20     0.20     0.20
## p[16]      0.19    0.00 0.00     0.18     0.18     0.19     0.19     0.19
## p[17]      0.18    0.00 0.00     0.17     0.17     0.18     0.18     0.18
## p[18]      0.17    0.00 0.00     0.16     0.17     0.17     0.17     0.17
## p[19]      0.16    0.00 0.00     0.15     0.16     0.16     0.16     0.16
## lp__   -2926.75    0.02 0.68 -2928.72 -2926.91 -2926.49 -2926.31 -2926.26
##        n_eff Rhat
## sigma   2888    1
## p[1]    2879    1
```

```
## p[2]    2888    1
## p[3]    2891    1
## p[4]    2893    1
## p[5]    2894    1
## p[6]    2894    1
## p[7]    2894    1
## p[8]    2895    1
## p[9]    2895    1
## p[10]   2895    1
## p[11]   2895    1
## p[12]   2895    1
## p[13]   2895    1
## p[14]   2895    1
## p[15]   2895    1
## p[16]   2895    1
## p[17]   2895    1
## p[18]   2895    1
## p[19]   2895    1
## lp__    1815    1
##
## Samples were drawn using NUTS(diag_e) at Mon Dec 12 19:03:42 2016.
## 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).
```

```
p2 <- extract(mod2)$p
# Re-plot the earlier plot, and add the prediction intervals
with(putts,plot(y/n~distance,pch=16,
                main="Physical Model Fit",
                ylim=c(0,1),bty="l",
                xlab="Distance (ft)",
                ylab="Proportion successful"))
lines(2:20,colMeans(p2),lwd=2)
lines(2:20,apply(p2,2,quantile,.025),col="red",lty=2)
lines(2:20,apply(p2,2,quantile,.975),col="red",lty=2)
```



b) Compare the fits of the two models using a Bayes factor. What do you conclude?

Code following this procedure for computing Bayes factors for each pair of models is given below.

```
#useful functions
logit <- function(x) { log(x/(1-x)) }
invlogit <- function(x) { exp(x)/(1+exp(x)) }


# Generate large random samples of theta5 and theta10
th5 <- rbeta(10000,ab5[1],ab5[2])
th10 <- rbeta(10000,ab10[1],ab10[2])

## For each pair of thetas, generate the beta coefficients for each model

# Logistic
b01 <- .6*logit(th5)+.4*logit(th10)
b11 <- -.2*logit(th5)+.2*logit(th10)

# Physical  #################################
smallr <- 1.68/24   ## A reg golf ball has diameter 1.68 in; we convert to ft
bigR <- 4.25/24   ## A reg golf hole has diameter 4.25 in; we convert to feet
sigma<-0.03   ### mean of the posterior distribution of sigma

## Calculate ln f(y|beta,model) for each sampled beta for each model.
## Exponentiate these to obtain f(y|beta,model) for each beta for each model,
## and take their average to obtain an estimate of f(y|model) for each model.
```

```r
# Logistic
putssuc<-putts$y/putts$n
thetas1 <- invlogit(b01 + b11 %*% t(distance-7))
logfyb1 <- log(thetas1) %*% putssuc + log(1-thetas1) %*% (1-putssuc) ###
fyb1 <- mean(exp(logfyb1))

# Physical
thetas2<-   2*pnorm(asin((bigR-smallr)/distance)/sigma)-1
logfyb2 <- log(thetas2) %*% putssuc + log(1-thetas2) %*% (1-putssuc) ##
fyb2 <- mean(exp(logfyb2))

## Estimate the Bayes factors for each pair of models:
BF12 <- fyb1/fyb2
BF12
```

```
## [1] 0.38867
```

Hence we can see that , given the expert's prior information, the two models perform differently . It appears that the physical model performs better than the Logistic model.