# Midterm 1

*Louis Bensard*

*March 6, 2018*

## (a)

$y_i \sim Bernoulli(\pi_i(\boldsymbol{\beta}))$, so $f_{Y_i}(y_i|\pi_i(\boldsymbol{\beta})) = [\pi_i(\boldsymbol{\beta})]^{y_i}[1-\pi_i(\boldsymbol{\beta})]^{1-y_i}$ and $y_i$'s are independent, therefore the likelihood is $L(\pi(\boldsymbol{\beta})|y) = \prod_{i=1}^{n} f_{Y_i}(y_i|\pi_i(\boldsymbol{\beta}))$. Thus, we get the following log-likelihood:

$$
\begin{aligned}
l(\boldsymbol{\beta}) &= \sum_{i=1}^{n} log\{[\pi_i(\boldsymbol{\beta})]^{y_i}[1-\pi_i(\boldsymbol{\beta})]^{1-y_i}\} \\
&\boxed{= \sum_{i=1}^{n} y_i \cdot log[\pi_i(\boldsymbol{\beta})] + (1-y_i) \cdot log[1-\pi_i(\boldsymbol{\beta})]}
\end{aligned}
$$

## (b)

$\pi_i(\boldsymbol{\beta}) = \frac{exp\{x_i^T\boldsymbol{\beta}\}}{1+exp\{x_i^T\boldsymbol{\beta}\}}$, therefore we have:

$$
\begin{aligned}
d\pi_i(d\boldsymbol{\beta}) &= \frac{x_i^T d\boldsymbol{\beta} exp\{x_i^T\boldsymbol{\beta}\}(1+exp\{x_i^T\boldsymbol{\beta}\}) - exp\{x_i^T\boldsymbol{\beta}\}(x_i^T d\boldsymbol{\beta} exp\{x_i^T\boldsymbol{\beta}\})}{(1+exp\{x_i^T\boldsymbol{\beta}\})^2} \\
&= \frac{x_i^T d\boldsymbol{\beta} exp\{x_i^T\boldsymbol{\beta}\}}{(1+exp\{x_i^T\boldsymbol{\beta}\})^2} \\
&= \frac{x_i^T d\boldsymbol{\beta} \pi_i(\boldsymbol{\beta})}{1+exp\{x_i^T\boldsymbol{\beta}\}} \\
&\boxed{= \pi_i(\boldsymbol{\beta})(1-\pi_i(\boldsymbol{\beta}))x_i^T d\boldsymbol{\beta}}
\end{aligned}
$$

$$
\begin{aligned}
dd\pi_i(d\boldsymbol{\beta}, d\boldsymbol{\beta}) &= x_i^T d\boldsymbol{\beta}[d\pi_i(d\boldsymbol{\beta})(1-\pi_i(\boldsymbol{\beta})) - \pi_i(\boldsymbol{\beta})d\pi_i(d\boldsymbol{\beta})] \\
&= x_i^T d\boldsymbol{\beta}(1-2\pi_i(\boldsymbol{\beta}))d\pi_i(d\boldsymbol{\beta}) \\
&\boxed{= \pi_i(\boldsymbol{\beta})(1-\pi_i(\boldsymbol{\beta}))(1-2\pi_i(\boldsymbol{\beta}))(x_i^T d\boldsymbol{\beta})^2}
\end{aligned}
$$

## (c)

$$
\begin{aligned}
dl(\boldsymbol{\beta}) &= \sum_{i=1}^{n} y_i \cdot \frac{d\pi_i(d\boldsymbol{\beta})}{\pi_i(\boldsymbol{\beta})} - (1-y_i) \cdot \frac{d\pi_i(d\boldsymbol{\beta})}{1-\pi_i(\boldsymbol{\beta})} \\
&= \sum_{i=1}^{n} [\frac{y_i}{\pi_i(\boldsymbol{\beta})} - \frac{1-y_i}{1-\pi_i(\boldsymbol{\beta})}] d\pi_i(d\boldsymbol{\beta}) \\
&= \sum_{i=1}^{n} [\frac{y_i - \pi_i(\boldsymbol{\beta})}{\pi_i(\boldsymbol{\beta})(1-\pi_i(\boldsymbol{\beta}))}] \pi_i(\boldsymbol{\beta})(1-\pi_i(\boldsymbol{\beta})) x_i^T d\boldsymbol{\beta} \\
&\boxed{= \sum_{i=1}^{n} [y_i - \pi_i(\boldsymbol{\beta})] \cdot x_i^T d\boldsymbol{\beta} = A_{(1\times 3)} d\boldsymbol{\beta}}
\end{aligned}
$$

Thus,

$$
\boxed{\frac{\partial l(\boldsymbol{\beta})}{\partial \beta_i} = A_{1i}, \quad i = 1, 2, 3}
$$

## (d)

```
library(rgl)

data = read.table('http://mathfaculty.fullerton.edu/mori/math534/examdata/blowBF.txt', h=T)
x = data.frame(x_i1=1, x_i2=log(data[,1]), x_i3=data[,2], stringsAsFactors=FALSE)
x = data.matrix(x)

maxit = 1000

beta_0 = rep(0,3) #arbitrary relevant starting value
tolerr = 1e-4

# The booleans are: (1) Steepest(TRUE) or Fisher(FALSE)
#(2) TRUE: Nice and full (as required) output, FALSE: basic output
max_seeker(data, x, beta_0, maxit, TRUE, TRUE, tolerr)
```

```
##
## Iteration   Halving    log-likelihood       MRE
##
##     1                      -456.7840        NA
##     1          1                 NA        NA
##     1          2                 NA        NA
##     1          3                 NA        NA
##     1          4                 NA        NA
##     1          5          -3262.4913        7.6e+00
##     1          6          -1632.9088        3.8e+00
##     1          7           -845.4744        1.9e+00
##     1          8           -533.3509        9.5e-01
##     1          9           -451.1653        4.7e-01
## ------------------------------------------------------------
##     2                      -451.1653        NA
```

```
##     2          1                NA        NA
##     2          2                NA        NA
##     2          3                NA        NA
##     2          4                NA        NA
##     2          5          -3858.5437      2.9e+00
##     2          6          -1779.9833      2.2e+00
##     2          7           -826.6247      1.6e+00
##     2          8           -509.8254      8.0e-01
##     2          9           -444.1664      4.0e-01
## ------------------------------------------------------------
## ...
## ...
## ...
##   676                     -281.9510        NA
##   676          1          -282.9157      2.8e-02
##   676          2          -282.1902      1.4e-02
##   676          3          -282.0098      7.2e-03
##   676          4          -281.9652      3.6e-03
##   676          5          -281.9543      1.8e-03
##   676          6          -281.9517      9.0e-04
##   676          7          -281.9511      4.5e-04
##   676          8          -281.9510      2.2e-04
##   676          9          -281.9510      1.1e-04
## ------------------------------------------------------------
##   677                     -281.9510        NA
##   677          1          -281.9633      2.0e-02
##   677          2          -281.9539      1.0e-02
##   677          3          -281.9516      5.0e-03
##   677          4          -281.9511      2.5e-03
##   677          5          -281.9510      1.3e-03
##   677          6          -281.9509      6.3e-04
## ------------------------------------------------------------
##   678                     -281.9509        NA
##   678          1          -282.7547      2.5e-02
##   678          2          -282.1502      1.3e-02
##   678          3          -282.0000      6.3e-03
##   678          4          -281.9628      3.2e-03
##   678          5          -281.9537      1.6e-03
##   678          6          -281.9515      7.9e-04
##   678          7          -281.9510      4.0e-04
##   678          8          -281.9509      2.0e-04
##   678          9          -281.9509      9.9e-05
## ------------------------------------------------------------
## ------------------------------------------------------------
## beta_final = -9.539707 3.18947 4.49944
```

After 678 iterations, the Modified Relative Error finally gets below $10^{-4}$ with a max log-likelihood of -281.9509. The Maximum Likelihood Estimate for $\boldsymbol{\beta}$ is then $\hat{\boldsymbol{\beta}} = (-9.540, 3.189, 4.499)^T$.

# (e)

$dl(d\boldsymbol{\beta}) = Ad\boldsymbol{\beta}$, thus we have:

$$
\begin{aligned}
ddl(d\boldsymbol{\beta}, d\boldsymbol{\beta}) &= dAd\boldsymbol{\beta} \\
&= \sum_{i=1}^{n} -d\pi_i(d\boldsymbol{\beta}) x_i^T d\boldsymbol{\beta} \\
&= \sum_{i=1}^{n} \pi_i(\boldsymbol{\beta})(\pi_i(\boldsymbol{\beta}) - 1) x_i^T d\boldsymbol{\beta}) x_i^T d\boldsymbol{\beta} \\
&\boxed{= d\boldsymbol{\beta}^T \sum_{i=1}^{n} x_i \pi_i(\boldsymbol{\beta})(\pi_i(\boldsymbol{\beta}) - 1) x_i^T d\boldsymbol{\beta} = d\boldsymbol{\beta}^T B_{(3\times 3)} d\boldsymbol{\beta}}
\end{aligned}
$$

# (f)

$$
\begin{aligned}
-E(ddl(d\boldsymbol{\beta}, d\boldsymbol{\beta})) &= -d\boldsymbol{\beta}^T E(B) d\boldsymbol{\beta} \\
&\boxed{= -d\boldsymbol{\beta}^T B d\boldsymbol{\beta}} \quad (B \textit{ does not depend on } y_i)
\end{aligned}
$$

Thus, we get the following elements of the Fisher Information Matrix:

$$
\boxed{I(\boldsymbol{\beta})_{ij} = -B_{ij}}
$$

# (g)

```
beta_0 = rep(0,3)
tolerr = 1e-6

max_seeker(data, x, beta_0, maxit, FALSE, TRUE, tolerr)
```

```
##
## Iteration   Halving    log-likelihood       MRE
##
##     1                      -456.7840        NA
##     1          0           -296.5639        8.1e+00
## ------------------------------------------------------------
##     2                      -296.5639        NA
##     2          0           -282.7706        3.1e+00
## ------------------------------------------------------------
## ...
## ...
## ...
##     4                      -281.9552        NA
##     4          0           -281.9505        8.4e-02
## ------------------------------------------------------------
##     5                      -281.9505        NA
##     5          0           -281.9505        5.4e-04
## ------------------------------------------------------------
##     6                      -281.9505        NA
##     6          1           -281.9505        1.1e-08
```

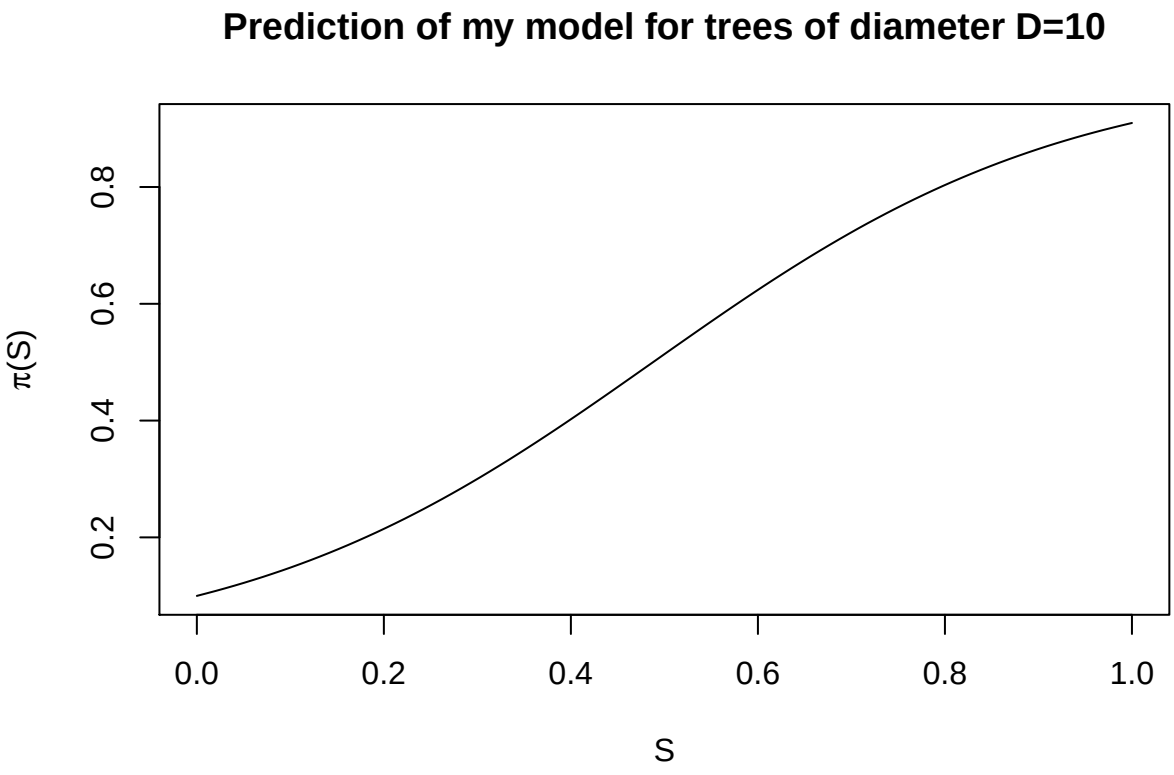
```
##      6          2          -281.9505        5.5e-09
##      6          3          -281.9505        2.8e-09
##      6          4          -281.9505        1.4e-09
##      6          5          -281.9505        6.9e-10
## ------------------------------------------------------------
## ------------------------------------------------------------
## beta_final = -9.562085 3.197563 4.508593
```

After 6 iterations, the Modified Relative Error finally gets below $10^{-6}$ with a max log-likelihood of -281.9509. The Maximum Likelihood Estimate for $\boldsymbol{\beta}$ is then $\hat{\boldsymbol{\beta}} = (-9.562, 3.198, 4.509)^T$. It is slightly different from the one obtained using the Steepest Ascent, but more accurate so I will use this one for part (h), (i) and (j).

As expected, the Fisher-Scoring Algorithm is more efficient by far than the Steepest Ascent Algorithm. It reaches a MRE ten thousand times smaller than the MRE reached by the Steepest Ascent in a number of iteration one hundred times smaller than the Steepest Ascent, with the same starting value $\beta_0$.

## (h)

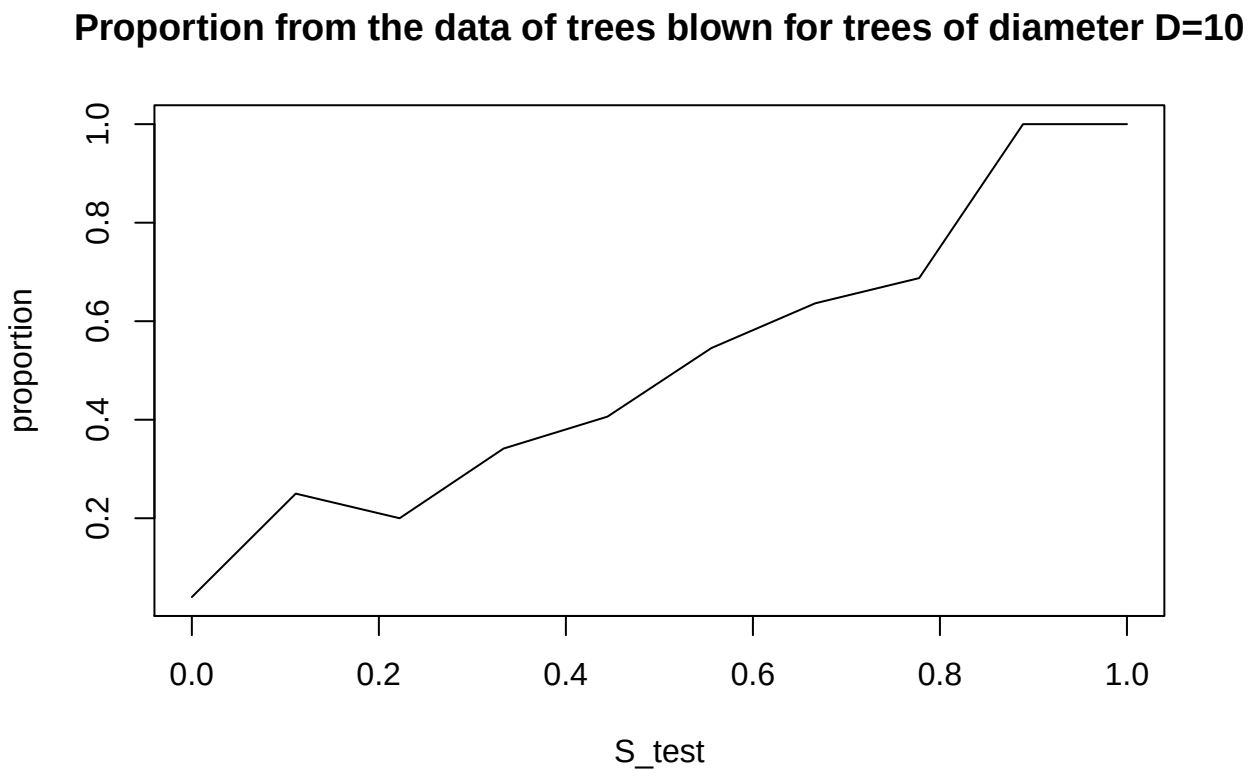
```
n = 100

S = seq(0,1, length=n)
pi_S = pi_fct_MLE(S, log(10))

plot(S, pi_S, type='l', ylab = expression(paste(pi,'(S)')))
title('Prediction of my model for trees of diameter D=10')
```



As expected, we can observe on the graph that for trees of diameter of 10, the bigger the severity of the winds in a given area, the more likely the trees in that area are to get blown down. For that kind of tree, an increase of severity will have the most impact between 0.4 and 0.6. The relationship between severity and probability to get blown down for that case is not linear but is close to it.

To make sur that my whole optimisation is correct, I decided to code a function that computes the proportion of trees blown down of diameter D in function of the severity of the winds in the area, the code will be posted in appendix 1.

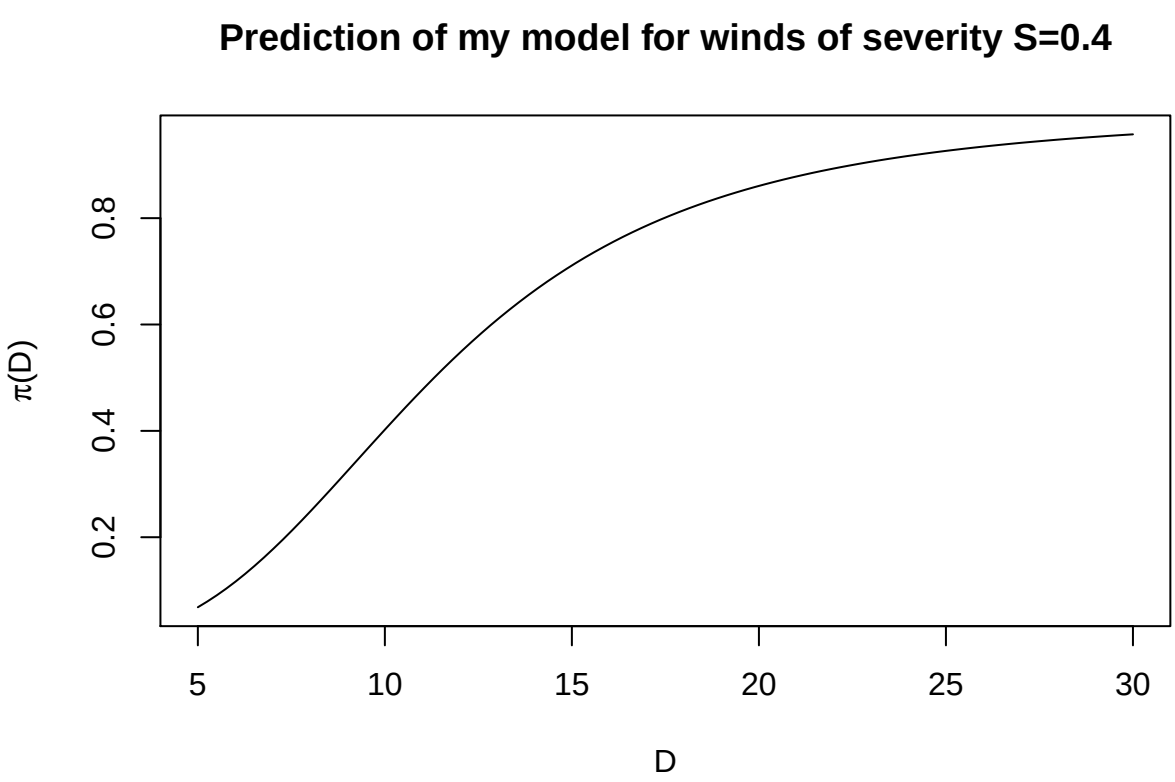
```
D = 10
pi_S_data(data, D)
```




This does not look bad, it is close enough from the graph of $\pi(\boldsymbol{\beta})$.

## (i)

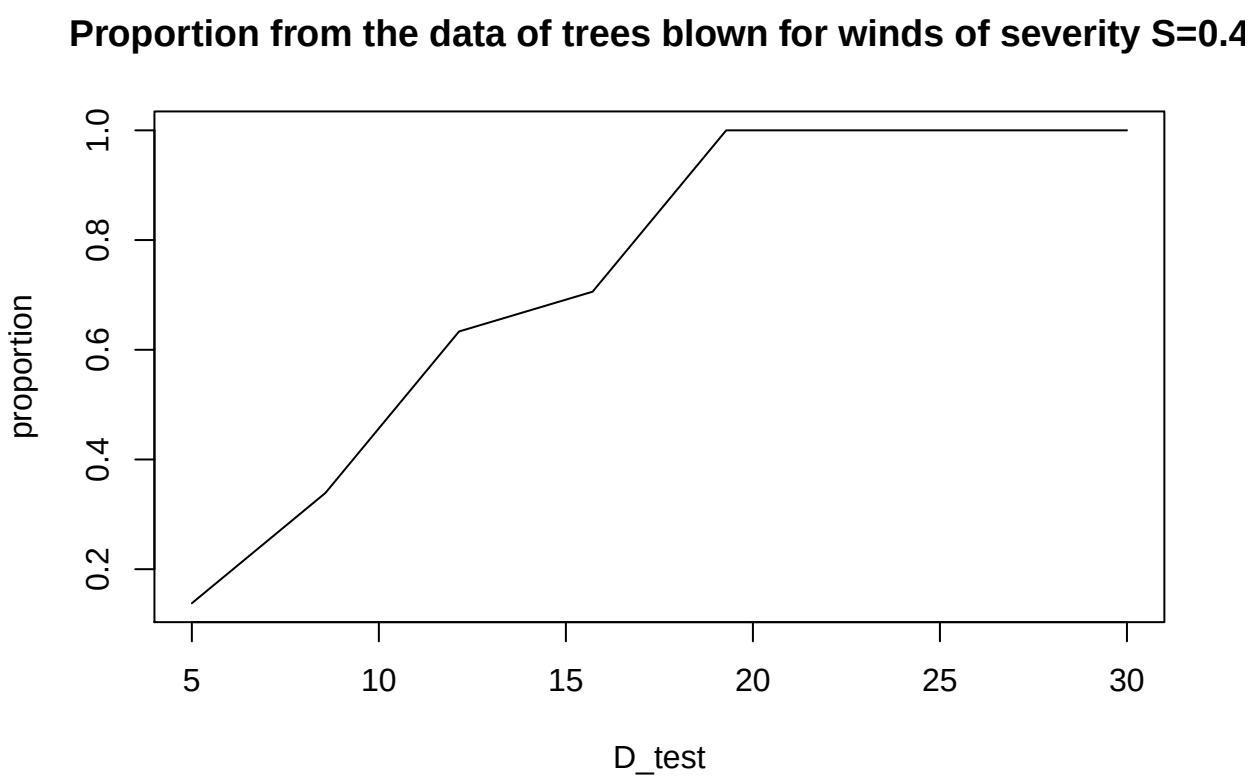
```
D = seq(5,30, length=n)
S = 0.4
pi_D = pi_fct_MLE(S, log(D))

plot(D, pi_D, type='l', ylab = expression(paste(pi,'(D)')))
title('Prediction of my model for winds of severity S=0.4')
```




This graph is interesting and somewhat unexpected. It shows that the bigger the diameter of the tree, the more likely it is to be blown down by winds of severity 0.4. Above a diameter of 20, the propability $\pi(D)$ increase very slowly. I would have expected the probability to go down as the diameter of the trees go up, but this shows the opposite.

Therefore, similarily as in part (h), I coded a function that computes the proportion of trees blown down that experienced winds of severity S in function their diameter, the code will be posted in appendix 2.

```
S=0.4
pi_D_data(data, S)
```

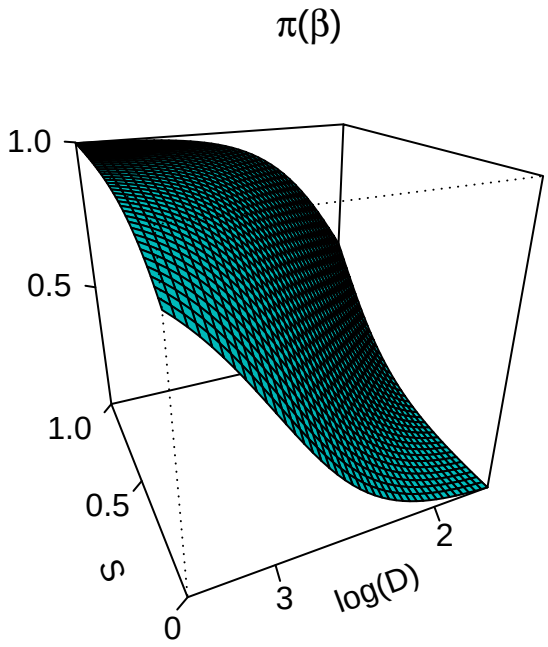



Even if the graph is very rough, it corroborates the trend of the previous graph. Therefore I am confident that our optimization above is correct.

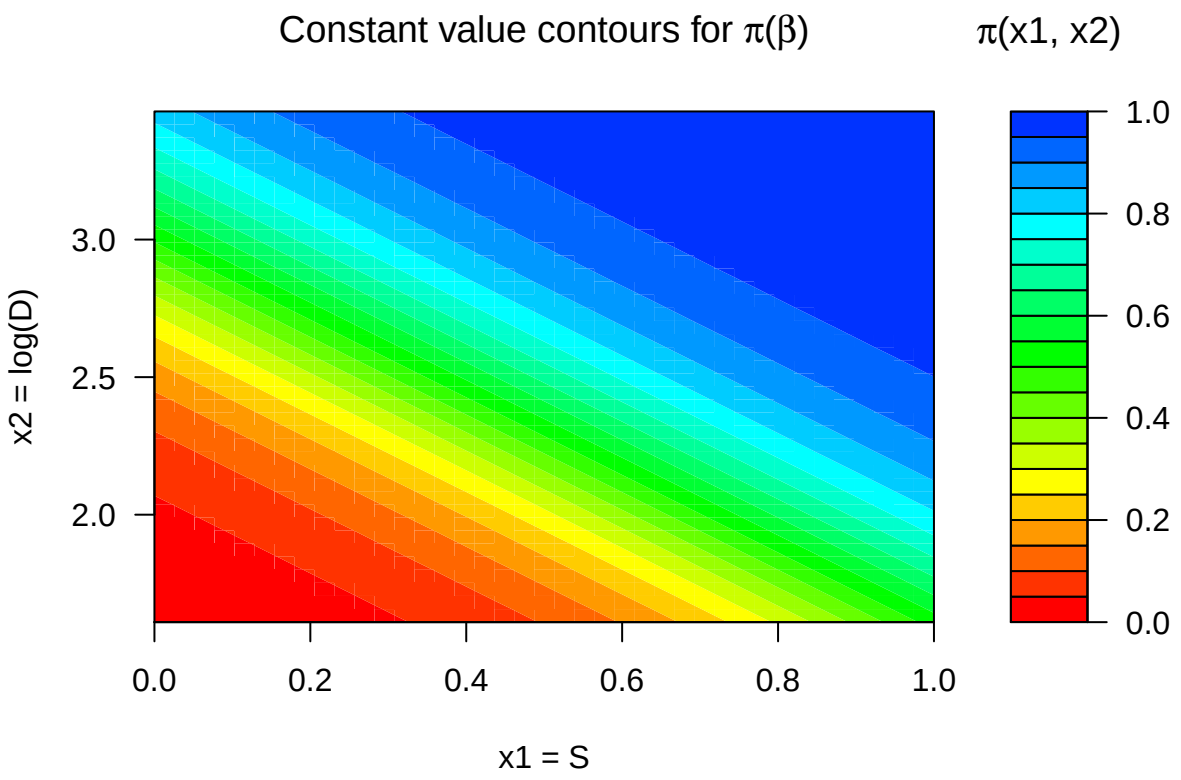
That is an interesting graph though, why are the big trees more blown down than the small one? Is that because the small ones bend instead of breaking when facing winds? Is our model limited and maybe does not hold very well for large tree diameters? More data from somewhere else for instance would probably be helpful to figure it out.

## (j)

```
n=40
x1 = seq(0,1, length=n) # = S
x2 = seq(min(x[,2]), max(x[,2]), length=n) #log (D)
x3 = outer(x1, x2, pi_fct_MLE) #pi(S, log(D))

persp(x1,x2,x3, theta=245, phi=25, r=2, shade=0.4, axes=TRUE, box=TRUE,
    ticktype="detailed" , nticks = 3, col="cyan", xlab="S", ylab="log(D)",
    zlab="", main=expression(paste(pi,'(', beta, ')')), expand = 1)
```




```
filled.contour(x1, x2, x3, col = rainbow(30), nlevels=30,
    plot.title = title(main= expression(paste('Constant value contours for ',
    pi,'(', beta, ')')), xlab="x1 = S", ylab="x2 = log(D)"),
    key.title = title(main=expression(paste(pi,'(x1, x2)'))))
```




We can see in the constant value contour plot that the Severity of the winds and the Diameter of the tree both effect significantly the probability that a tree is blown down. Basically, the more severe the winds and the bigger the trees, the more likely a tree is to be blown down. The probability for a tree to be blown down will increase the most if both parameters increase at the same time.

We can observe that small trees are unlikely to be blown down at all, whatever the winds. When the Severity get above 0.8, any size of tree is likely to be blown down. For trees of diameter above 20, any severity of winds (even small) is very likely to blow the tree down, which makes me suspicious about the model for large diameters, or about my understanding of trees, as mentionned before.

## Appendix 1:

```
pi_S_data = function(data, D){

    proportion  = c()
    breaks = 10

    for(a in 1:breaks){

        ones = c()
        for(i in 1:length(data[,1])){

            if(((a-1)/breaks <= data[i,2]) & (data[i,2] < (a/breaks))){
                if(abs(data[i,1]-D) <= 2) ones = c(ones, data[i,3])
```

```
            }
        }

        proportion = c(proportion, sum(ones) / length(ones))
    }

    S_test = seq(0,1,length=breaks)

    plot(S_test, proportion, type='l')
    title('Proportion from the data of trees blown for trees of diameter D=10')

}
```

## Appendix 2:

```
pi_D_data = function(data, S){

    proportion  = c()
    breaks = 10

    for(a in 1:breaks){

        ones = c()
        for(i in 1:length(data[,1])){

            if((((a-1)*30/breaks) + 5 <= data[i,1]) & (data[i,1] < (a*30/breaks)+5)){
                if(abs(data[i,2]-S) <= 0.1) ones = c(ones, data[i,3])
            }
        }

        if(length(ones) > 0) proportion = c(proportion, sum(ones) / length(ones))

    }

    D_test = seq(5,30,length=length(proportion))

    plot(D_test, proportion, type='l')
    title('Proportion from the data of trees blown for winds of severity S=0.4')

}
```