Question 1

a) Stemming the Federalists

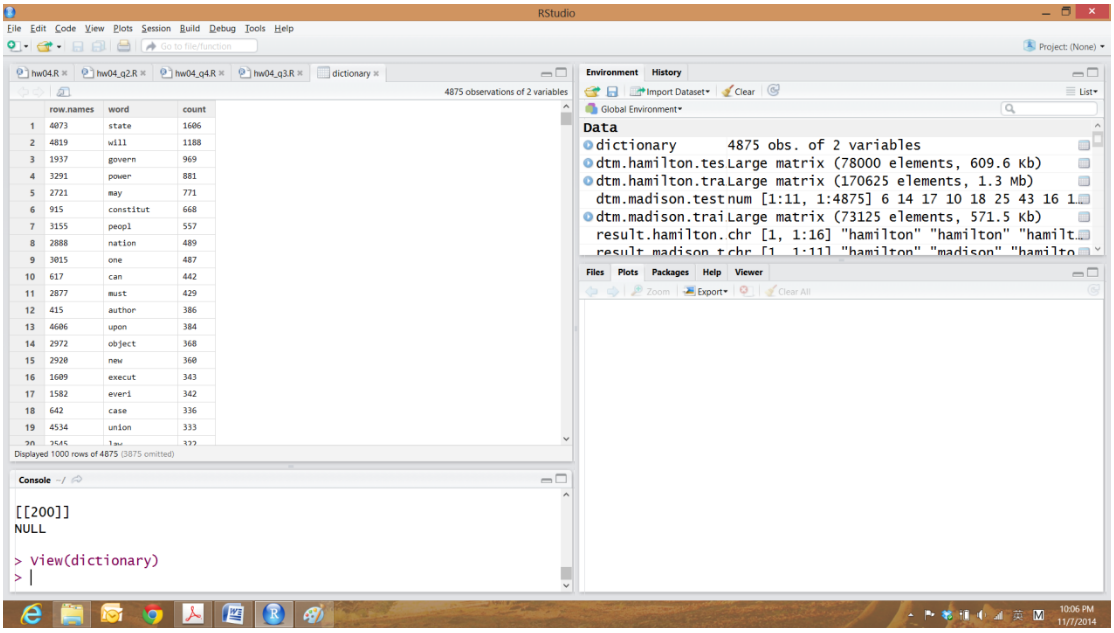
Such is a segment from file Federalist01.txt in folder fp_hamilton_test_clean:

```
general introduct
independ journal
peopl state new york
unequivoc experi ineffici
subsist feder govern call upon deliber
new constitut unit state america subject
speak import comprehend consequ
noth less exist union safeti welfar
part compos fate empir mani
respect interest world frequent
remark seem reserv peopl
countri conduct exampl decid import
question whether societi men realli capabl
establish good govern reflect choic whether
forev destin depend polit
constitut accid forc truth
remark crisi arriv may proprieti
regard era decis made
wrong elect part shall act may view deserv
consid general misfortun mankind
idea will add induc philanthropi
patriot heighten solicitud consider
good men must feel event happi will choic
```

b) Read Federalists into Workspace. This is segment of infile No.15 in hamilton.test:

```
[[15]]
  [1] "subject"      "continu"      "concern"      "power"        "congress"
  [6] "regul"        "elect"        "member"       "new"          "york"
 [11] "packet"       "tuesday"      "februari"     "26"           "1788"
 [16] "peopl"        "state"        "new"          "york"         "seen"
 [21] "uncontrol"    "power"        "elect"        "feder"        "govern"
 [26] "without"      "hazard"       "commit"       "state"        "legislatur"
 [31] "let"          "us"           "now"          "see"          "danger"
 [36] "side"         "confid"       "ultim"        "right"        "regul"
 [41] "elect"        "union"        "pretend"      "right"        "ever"
```

c) Make a dictionary.



4875 observations of 2 variables

	row.names	word	count
1	4073	state	1606
2	4819	will	1188
3	1937	govern	969
4	3291	power	881
5	2721	may	771
6	915	constitut	668
7	3155	peopl	557
8	2888	nation	489
9	3015	one	487
10	617	can	442
11	2877	must	429
12	415	author	386
13	4606	upon	384
14	2972	object	368
15	2920	new	360
16	1609	execut	343
17	1582	everi	342
18	642	case	336
19	4534	union	333
20	2545	law	322

Displayed 1000 rows of 4875 (3875 omitted)

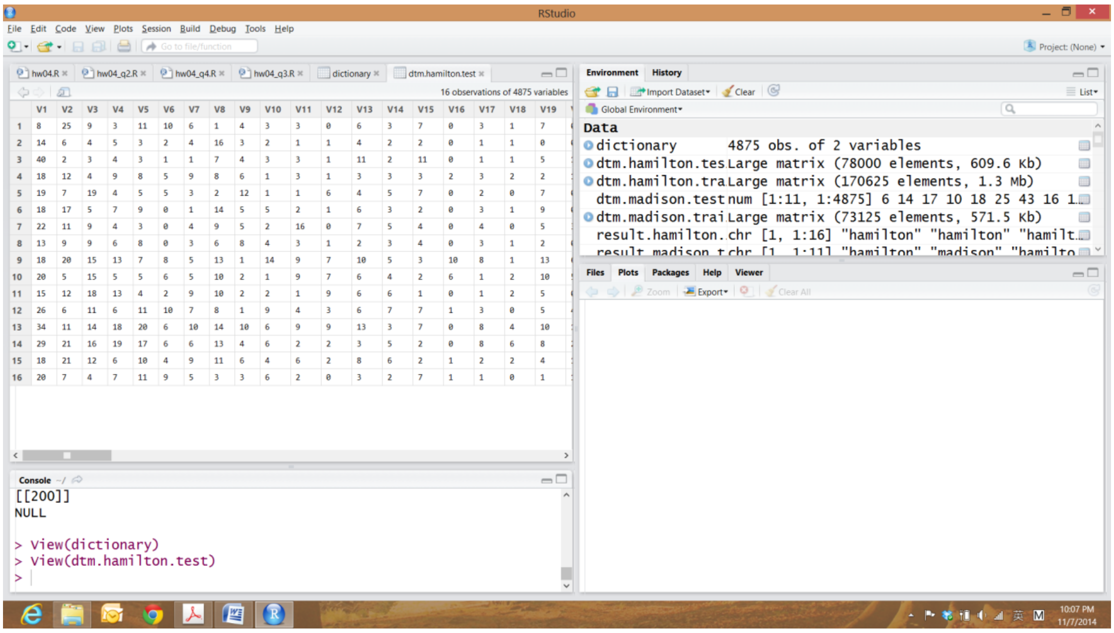
```
Data
dictionary         4875 obs. of 2 variables
dtm.hamilton.tes.Large matrix (78000 elements, 609.6 Kb)
dtm.hamilton.tra.Large matrix (170625 elements, 1.3 Mb)
dtm.madison.test num [1:11, 1:4875] 6 14 17 10 18 25 43 16 1...
dtm.madison.trai.Large matrix (73125 elements, 571.5 Kb)
result.hamilton..chr [1, 1:16] "hamilton" "hamilton" "hamilt...
result.madison.t.chr [1, 1:11] "hamilton" "madison" "hamilto
```

```
[[200]]
NULL

> View(dictionary)
> 
```

d) Calculate document-terms matrix.

This is a segment from dtm.hamilton.test



16 observations of 4875 variables

	V1	V2	V3	V4	V5	V6	V7	V8	V9	V10	V11	V12	V13	V14	V15	V16	V17	V18	V19
1	8	25	9	3	11	10	6	1	4	3	3	0	6	3	7	0	3	1	7
2	14	6	4	5	3	2	4	16	3	2	1	1	4	2	2	0	1	1	0
3	40	2	3	4	3	1	1	7	4	3	3	1	11	2	11	0	1	1	5
4	18	12	4	9	8	5	9	8	6	1	3	1	3	3	3	2	3	2	2
5	19	7	19	4	5	5	3	2	12	1	1	6	4	5	7	0	2	0	7
6	18	17	5	7	9	0	1	14	5	5	2	1	6	3	2	0	3	1	9
7	22	11	9	4	3	0	4	9	5	2	16	0	7	5	4	0	4	0	5
8	13	9	9	6	8	0	3	6	8	4	3	1	2	3	4	0	3	1	2
9	18	20	15	13	7	8	5	13	1	14	9	7	10	5	3	10	8	1	13
10	20	5	15	5	5	6	5	10	2	1	9	7	6	4	2	6	1	2	10
11	15	12	18	13	4	2	9	10	2	2	1	9	6	6	1	0	1	2	5
12	26	6	11	6	11	10	7	8	1	9	4	3	6	7	7	1	3	0	5
13	34	11	14	18	20	6	10	14	10	6	9	9	13	3	7	0	8	4	10
14	29	21	16	19	17	6	6	13	4	6	2	2	3	5	2	0	8	6	8
15	18	21	12	6	10	4	9	11	6	4	6	2	8	6	2	1	2	2	4
16	20	7	4	7	11	9	5	3	3	6	2	0	3	2	7	1	1	0	1

```
Data
dictionary         4875 obs. of 2 variables
dtm.hamilton.tes.Large matrix (78000 elements, 609.6 Kb)
dtm.hamilton.tra.Large matrix (170625 elements, 1.3 Mb)
dtm.madison.test num [1:11, 1:4875] 6 14 17 10 18 25 43 16 1...
dtm.madison.trai.Large matrix (73125 elements, 571.5 Kb)
result.hamilton..chr [1, 1:16] "hamilton" "hamilton" "hamilt...
result.madison.t.chr [1, 1:11] "hamilton" "madison" "hamilto
```

```
[[200]]
NULL

> View(dictionary)
> View(dtm.hamilton.test)
> 
```

e) Calculate vectors of log probability with given penalty μ

Such as a segment from logp.hamilton.test

```
> logp.hamilton.test
 [1]  -3.867745  -4.415339  -4.554825  -4.812970  -4.767515  -5.368599
 [7]  -5.206798  -4.696066  -5.341938  -5.438538  -5.368599  -5.760508
[13]  -5.087765  -5.513738  -5.409973  -6.627443  -5.721304  -6.494034
[19]  -5.140122  -6.145978  -5.721304  -6.008854  -5.467943  -5.409973
[25]  -5.780703  -5.438538  -5.315970  -6.206565  -6.580967  -6.117008
[31]  -5.865823  -5.665236  -7.273183  -6.304939  -5.934784  -6.175813
[37]  -6.088853  -6.145978  -5.780703  -5.888286  -6.781431  -5.665236
[43]  -5.888286  -6.206565  -6.627443  -5.721304  -6.453246  -6.238293
[49]  -6.088853  -6.376345  -5.843854  -5.801313  -6.453246  -5.934784
[55]  -7.590939  -6.206565  -6.034816  -5.482976  -7.368307  -6.145978
[61]  -6.580967  -5.843854  -6.414056  -5.958869  -6.899072  -6.340005
```

Question 2

We use the basic formula Naive Bayes Classifier calculate the likelihood,

$$P\{Y = y_i \mid X = Words\} \propto P\{Y = y_i\}\prod_{k=1}^{n}\left[\#Word_k \times P\{X_k = Word_k\} \mid Y = y_i\right]$$

Take log() on both side, we have,

$$\log(P\{Y = y_i \mid X = Words\}) \propto \log(P\{Y = y_i\}) + \sum_{k=1}^{n}\left[\#Word_k \times \log(P\{X_k = Word_k\} \mid Y = y_i)\right]$$

If $\log(P\{Y = Hamilton \mid X = Words\}) > \log(P\{Y = Madison \mid X = Words\})$, we assign this Federalist to Hamilton; otherwise, assign it to Madison.

```
## define a function
# Naive Bayes Classification Procedure
naiveBayes <- function(logp.hamilton.train, logp.madison.train,
                    log.prior.hamilton, log.prior.madison , dtm.test){
  # initial result vector
  result = c();
  # calculate the words probability vector of each Federalist in dtm.test
  for(i in 1: nrow(dtm.test)){
    # probability = frenquency
    temp= dtm.test[i,];
    # directly calculate log likelihood of each Federalist i in dtm.test
    # the number of word * prob that its belongs to class k(k = 1, 2)
    lh.hamilton = t(temp) %*% logp.hamilton.train + log.prior.hamilton;
    # print(lh.hamilton);
    lh.madison = t(temp) %*% logp.madison.train + log.prior.madison;
    # print(lh.madison)
    if(lh.hamilton >= lh.madison){
      class = "hamilton";
    }else{
      class = "madison";
    }
    result = cbind(result, class);
  }
  return(result);
}
```

The we test this model with dtm.hamilton.test and dtm.madison.test. The results shows below:

```
> result.hamilton.test
     class      class      class      class      class      class     
[1,] "hamilton" "hamilton" "hamilton" "hamilton" "hamilton" "hamilton"
     class      class      class      class      class      class     
[1,] "hamilton" "hamilton" "hamilton" "hamilton" "hamilton" "hamilton"
     class      class      class      class     
[1,] "hamilton" "hamilton" "hamilton" "hamilton"

> result.madison.test
     class      class     class      class     class     class    
[1,] "hamilton" "madison" "hamilton" "madison" "madison" "madison"
     class     class     class      class     class     
[1,] "madison" "madison" "hamilton" "madison" "hamilton"
```

Then we see that all Federalists in Hamilton testing set are classified correctly. However, Federalist No.10, No.41, No.57, No.62 from Madison are wrong classified as from Hamilton.

Question 3

The result is as below,

		Predicted Class		Total
		Null	Non-null	
True Class	Null	7(0.636)	4(0.364)	11
	Non-null	0(0.000)	16(1.000)	16
Tatal		7	20	

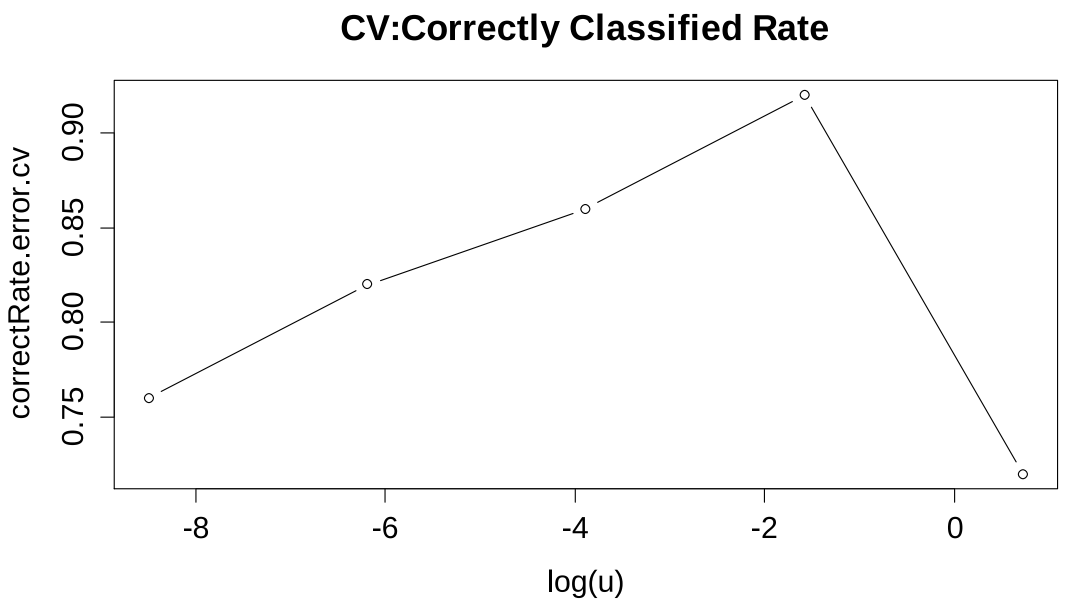
```
> print(sprintf('The percentage of correctly classified paper is: %s', prec.correct(res
ult.hamilton.test, result.madison.test)));
[1] "The percentage of correctly classified paper is: 0.851851851851852"
```

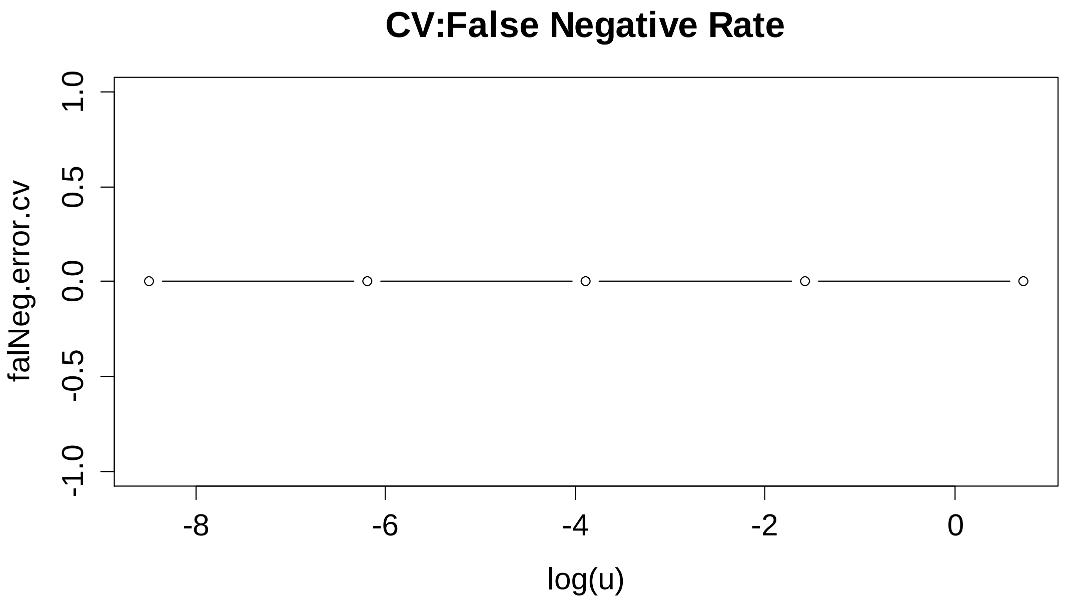
```
> # true positive
> print(sprintf("True Positive:%s",truePos(result.hamilton.test)));
[1] "True Positive:1"
> # true nagetive
> print(sprintf("True Negative:%s",trueNeg(result.madison.test)));
[1] "True Negative:0.636363636363636"
> # false_positive
> print(sprintf("False Positive:%s",falsePos(result.madison.test)));
[1] "False Positive:0.363636363636364"
> # false_negative
> print(sprintf("False Negaitive:%s",falseNeg(result.hamilton.test)));
[1] "False Negaitive:0"
```

Question 4

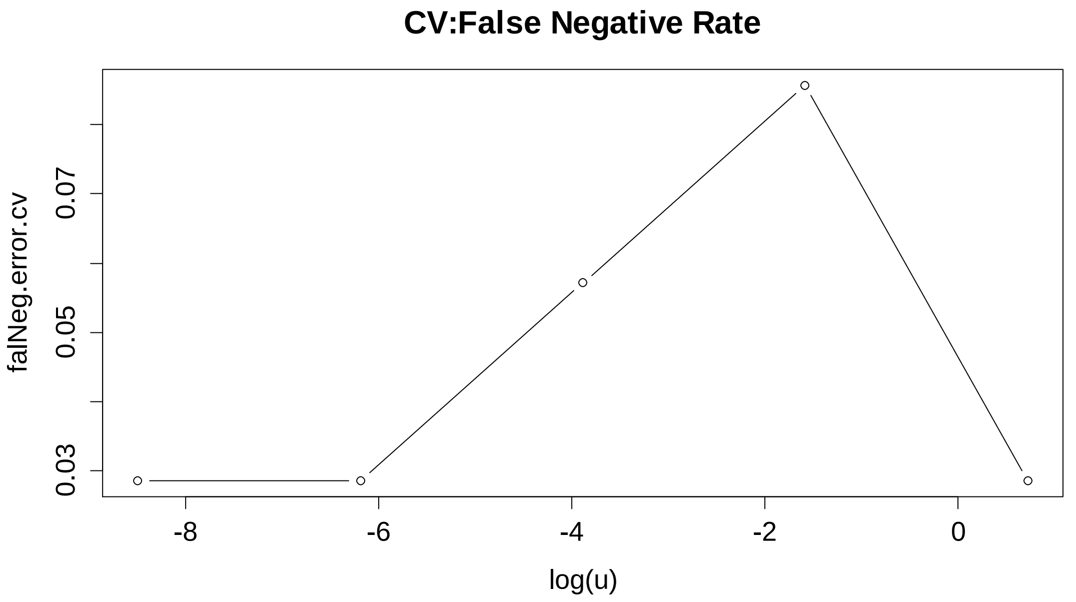
a) Using 5-fold cross-validation on the training set.

	1/|D|	10/|D|	100/|D|	1000/|D|	10000/|D|
Correct rate	0.76	0.82	0.86	0.92	0.72
False Neg	0.00	0.00	0.00	0.00	0.00
False Pos	0.80	0.60	0.47	0.27	0.93

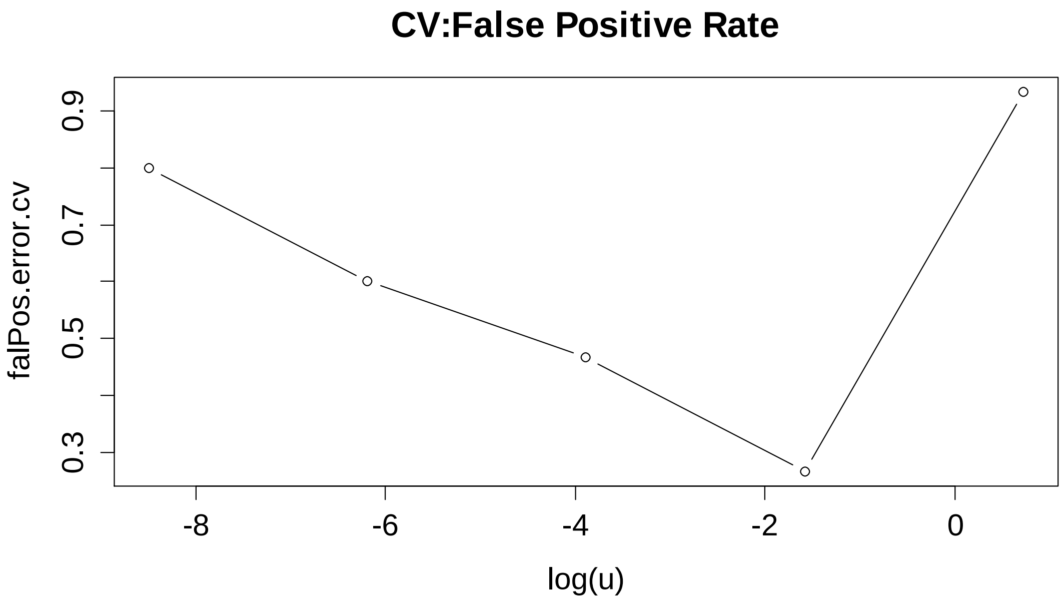




Notice: This rate could be different if we use "randomly" Cross Validation and "consecutively" Cross Validation since there is slightly difference near 0 in false Negative rates. The "consecutively" CV test result is



This "consecutively" selected cross folds would cause overestimation on false negative rates. On the other hand, the "randomly" selected cross folds would underestimate this rate at the best μ. This situation also happens on correctly classification rate and false positive rate.

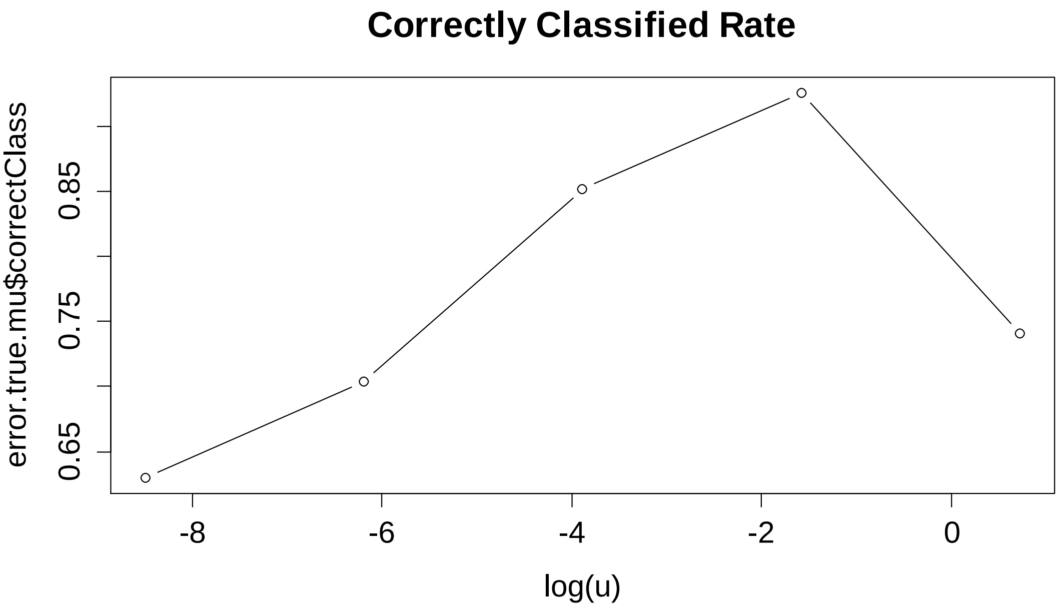


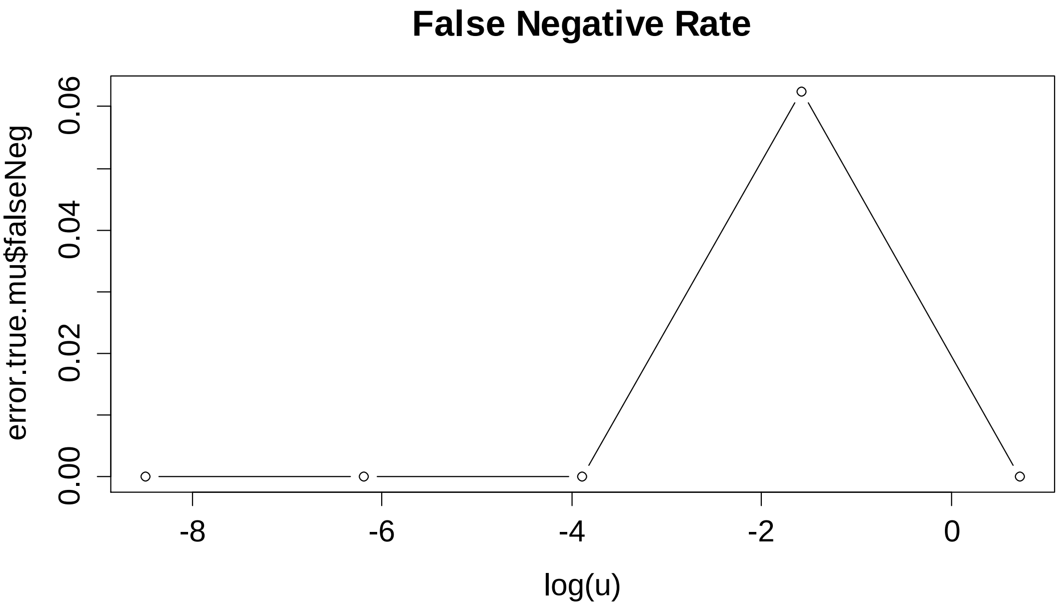
b) Best μ

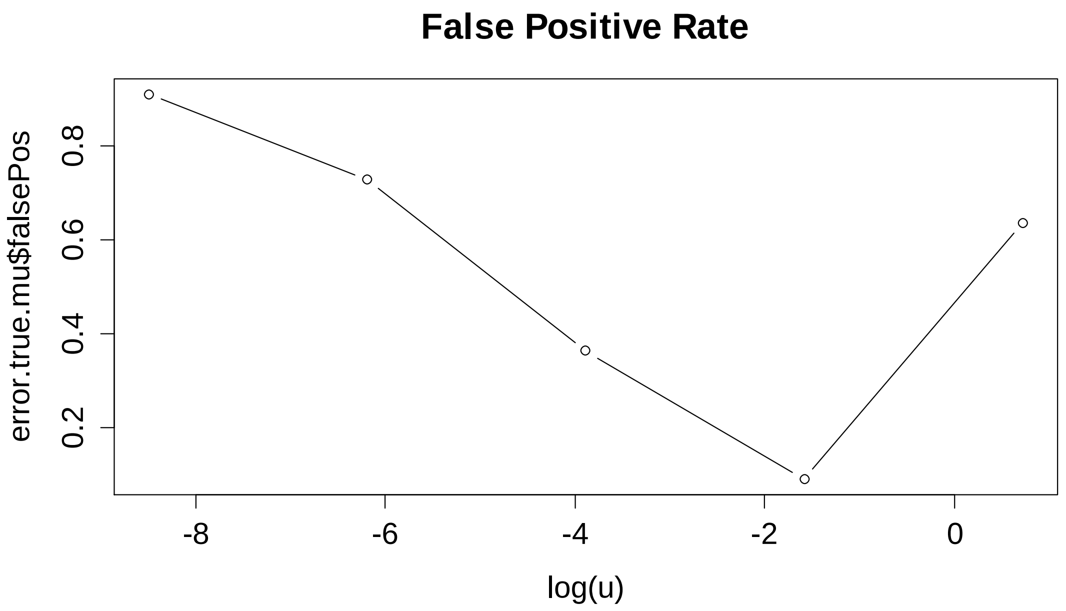
We could conclude from the graphs that $\mu = 1000\frac{1}{|D|} = 0.2051$ seems to be the best penalty in this Naive Bayes classifier. On one hand, it produces a highest correctly classification rate, at about 92%; On the other hand, it produces a lowest false positive rate at about 0.27, given that their false negative rates are the same.

c) True rates

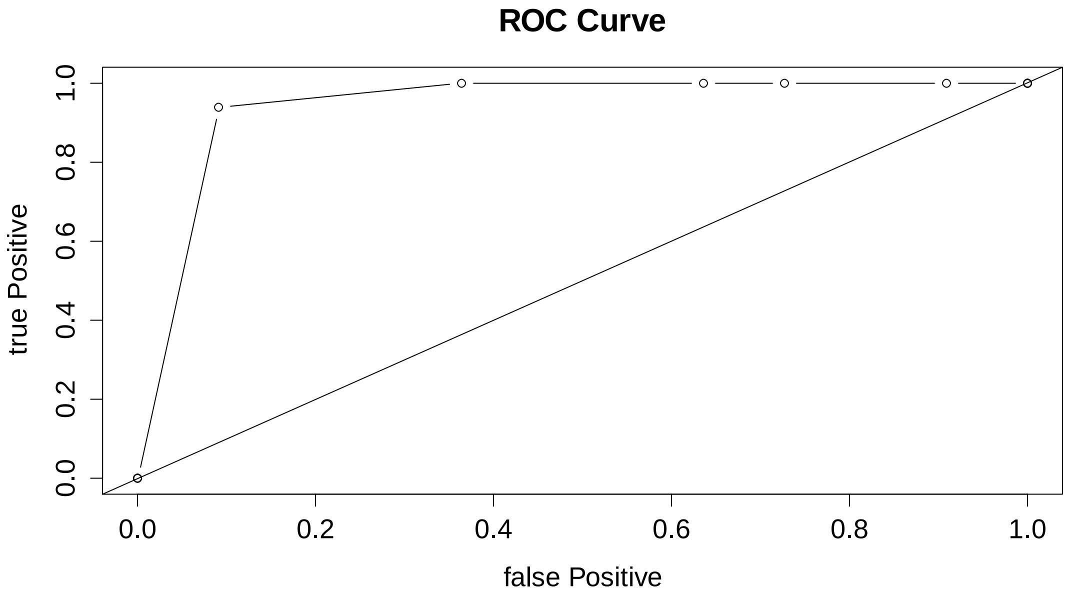
	1/|D|	10/|D|	100/|D|	1000/|D|	10000/|D|
Correct rate	0.630	0.704	0.852	0.926	0.741
False Neg	0.000	0.000	0.000	0.063	0.000
False Pos	0.909	0.727	0.364	0.091	0.636


Correctly Classified Rate


False Negative Rate



We could plot its ROC curve to test which thresholds generated by different parameter is a good classifier.



It is shown that $\mu = 1000\frac{1}{|D|} = 0.2051$is the best parameter because the threshold generated by this parameter is huge to the top left of the curve(high true positive rate and low false positive rate). This is consistent to our conclusion in b).

d) Difference between true error rate and cross validation error rates.

We calculate percentage error under the best value of μ.

Rate	CorrectClassified	FalseNeg	FalsePos
Percentage	0.64%	-100%	193%

We could see that the correct-classified rate and false positive rate are overestimated but the false negative rate is underestimated by the "randomly" selected cross-validation sampling method.

As discuss on the false negative rate, the flexibility of cross-validation sampling (randomly, consecutively, interleaved, etc) could be one of the reasons that leads to difference on such rates. In addition, one reason could be the numbers of training samples. The numbers of training samples in both hamilton.train and madison.train accounts for the prior significantly, so that the classification result. Also, the number of folds could leads to bias in estimation of such rates, but this effect should be so slight that can be ignored, like LOOCV and 10-folds CV are almost same.

In general, the flexibility (the variability in how the observations are divided into 5-folds) is related to the estimation of classification error in Cross-validation sampling. This significantly depends on the value of μ. So choosing a best penalty in Naive Bayes Classification is the way to control the flexibility, hence the best way to minimum misclassification rate.