Dataset #2: Movie Ratings

Load the Data

In [21]:
```
import numpy as np
import pandas as pd

import warnings
warnings.simplefilter('ignore', DeprecationWarning)

# dataset
datatrain = pd.read_csv('/Users/WeikangFan/Desktop/Movies_train.csv')
datatest = pd.read_csv('/Users/WeikangFan/Desktop/Movies_test.csv')
```

Pre-Process the Movies Dataset

In [22]:
```
%load_ext rmagic
import rpy2 as Rpy
```

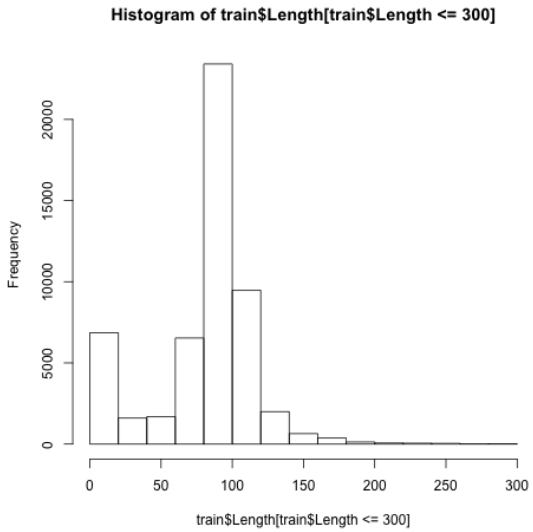
```
The rmagic extension is already loaded. To reload it, use:
  %reload_ext rmagic
```

In [23]:
```
%%R
train <- read.csv("Movies_train.csv")
sort <- sort(train$Length, decreasing = TRUE)
sort[1:50]
```

```
 [1] 5220 2880 1100  873  647  566  555  501  485  480  480  418  417  410  407
[16]  402  399  390  390  360  358  358  321  320  320  316  316  315  312  311
[31]  306  302  301  300  300  300  300  299  298  294  293  288  288  285  285
[46]  285  285  279  278  278
```

We could find that only a very small proportion of the training set observations (33 observations in total) have Length values larger than 300.

In [24]:
```
%%R
hist(train$Length[train$Length <= 300])
```



From the histogram above we could find that Length values between 200 and 300 are rare too.

In [25]:
```
%%R
length(train$Length[train$Length <= 200])/length(train$Length)
```

```
[1] 0.9958428
```

Length values no more than 200 take more than 99% of the whole sample. I decide to use the observaions with Length no more than 200 to train the classifier, so we can avoid the effects of some huge outliers.

Moreover, I use the mean value to substitute the NAs in Budget, and use the logs of Year, Length, Budget, and Votes intead of their original values.

In [26]:
```
datatrain = datatrain.loc[datatrain['Length'] <= 200]

from numpy import log1p
datatrain['Year'] = [log1p(max(datatrain['Year']) - i) for i in datatrain['Year'] ]
datatrain['Budget'] = datatrain['Budget'].fillna(datatrain['Budget'].dropna().mean())
datatrain['Length'] = [log1p(i) for i in datatrain['Length'] ]
datatrain['Budget'] = [log1p(i) for i in datatrain['Budget'] ]
datatrain['Votes'] = [log1p(i) for i in datatrain['Votes'] ]

datatest['Year'] = [log1p(max(datatest['Year']) - i) for i in datatest['Year'] ]
datatest['Budget'] = datatest['Budget'].fillna(datatest['Budget'].dropna().mean())
datatest['Length'] = [log1p(i) for i in datatest['Length'] ]
datatest['Budget'] = [log1p(i) for i in datatest['Budget'] ]
datatest['Votes'] = [log1p(i) for i in datatest['Votes'] ]
```

Variable Selection

The dataset has the following variables:

Title	movie title
Year	year released
Length	length in minutes
Budget	production budget in US dollars (usually NA)
Votes	number of voting IMDB users
R1	approximate percentage of users voting for rating: 1
R2	approximate percentage of users voting for rating: 2
R3	approximate percentage of users voting for rating: 3
R4	approximate percentage of users voting for rating: 4
R5	approximate percentage of users voting for rating: 5
R6	approximate percentage of users voting for rating: 6
R7	approximate percentage of users voting for rating: 7
R8	approximate percentage of users voting for rating: 8
R9	approximate percentage of users voting for rating: 9
R10	approximate percentage of users voting for rating: 10
MPAA	MPAA parental guidance rating (blank, NC-17, PG, PG-13, R)
Action	1 if Action, 0 otherwise
Animation	1 if Animation, 0 otherwise
Comedy	1 if Comedy, 0 otherwise
Drama	1 if Drama, 0 otherwise
Documentary	1 if Documentary, 0 otherwise
Romance	1 if Romance, 0 otherwise
Short	1 if Short Film, 0 otherwise

Among those variables, Length is negatively correlated with Short: movies of higher Length has Short = 0, and movies of lower Length has Short = 1. The following correlation table computed by R agrees with my assumption.

In [28]:
```
%%R
train <- read.csv("/Users/WeikangFan/Desktop/Movies_train.csv")
cor(train[,c('Length','Short')])
```

```
           Length      Short
Length  1.0000000 -0.6655172
Short  -0.6655172  1.0000000
```

And compared with the predictor set including MPAA, the predictor set without MPAA gives better prediction results. Title has no correlation with Rating, hence I finally decide to use all the variables except for Title, MPAA, and Short.

Prediction: Use Random Forest

I have tried various methods and random forest definitely beats all the others.

Use random forest to train the classifier and make predictions

In [29]:
```
x_train = datatrain.drop(["Rating", "Title", "Short", "MPAA"],1)
y_train = datatrain.Rating
x_test = datatest.drop(["Rating", "Title", "Short", "MPAA"],1)

# use Random Forest to predict
from sklearn.ensemble import RandomForestClassifier
np.random.seed(1234)
rf = RandomForestClassifier(n_estimators=500, max_features='auto').fit(x_train,y_train)
pred = rf.predict(x_test)
```

Write the result to a text file

In [31]:
```
# Write results to a txt file
def two_digit_value(x):
    return ("%3.1f" % x)
    # return a value of the form X.Y, where X and Y are digits
output_str = "\n".join(map(two_digit_value, pred))
f = open('/Users/WeikangFan/Desktop/output-movie.txt', 'w')
f.write(output_str);
f.close()
```

The correction rate is 51.41% according to the mooshak.