HW0_Fitting_Distributions

CS249 — Spring 2016 — D.S. Parker

1 HW0: Fitting Distributions with Maximum Likelihood

The basic problem here is to determine, when given an input sequence of real values, which distribution it follows. More specifically, for this assignment you are to develop a notebook that reads in a numeric table, and – for each dataset (i.e., each column in the table) – determines the distribution and parameters that gives the closest match to it.

For example, your notebook could read an input table like this:

D1	D2	D3	D4	D5	D6
7.1378018	6.8581740	0.2379494	0.1476523	5.3174948	3.1291521
3.3713903	6.2437282	0.2138276	0.1699299	1.5583491	0.6543210
⋮	⋮	⋮	⋮	⋮	⋮
8.7408465	2.7770890	2.7271062	0.5956197	2.3983975	8.8628316

The columns of this table define six datasets. Your notebook should produce a CSV file HW0_output.csv giving distributions that (it thinks) best fit the data. A correct output file could then look like this:

```
gamma,9,2,
normal,4,2,
lognormal,0,1
exponential,1,,
chi-squared,5,,
logistic,3,2,
```

For simplicity, the parameters used in this assignment will always be integers, so the printed output should always have integer parameter values.

Your notebook can determine the distribution that fits best in any way you like. However, the fitdistr() function in the MASS library gives a simple way to do this. The notebook describes the assignment in much more detail, and gives orientation about how to do things in R.

In other words: yes, this is a simple assignment. It is intended as a warmup.

After running your notebook on the test input file HW0_test.csv, to complete this assignment please upload two files to CCLE:

- your output CSV file HW0_output.csv
- your notebook file HW0_Fitting_Distributions.ipynb

We will not execute your uploaded notebook. It should have the commands you used to produce the output file — in order to show your work. As announced, all assignment grading in this course will be automated, and the notebook is needed in order to check results of the grading program.

Summary: the basic problem is to determine, given a dataset of random real values, which distribution it follows. Your notebook should read in a numeric table, and – where each column in the table is a "dataset" – identify the distribution and parameters that gives the closest match to it.

Important Notes:

- For simplicity, **the parameters in this assignment will always be integers.** Your printed output should always have integer parameter values.
- **Some distribution descriptions are equivalent.** For example,

```
exponential,1,
gamma,1,1
weibull,1,1
```

  are equivalent. These pdfs all are equal to e^{-x}. Your output file can use ANY equivalent form. The grading program will treat equivalent forms for distributions as correct.
- **We will use Paul Eggert's Late Policy:** The number of days late is $N = 0$ for the first 24 hrs, $N = 1$ for the next 24 hrs, etc., and if you submit an assignment H hours late, $2^{\lfloor H/24 \rfloor}$ points are deducted.

1.1 fitdistr

You can use the "fitdistr" function in the MASS library to fit distributions to data.

In [47]:
```
not.installed <- function(pkg) !is.element(pkg, installed.packages()[,1])

if (not.installed("MASS"))  install.packages("MASS")  # we need the MASS package

library(MASS)  #  load the MASS package

#  ?fitdistr      #  look at the help for the fitdistr function
```

1.2 Generate a sample table with 6 columns (datasets)

The table is of size (N x 6), where N=10000. Each column in this dataset is a random sample from a different distribution.

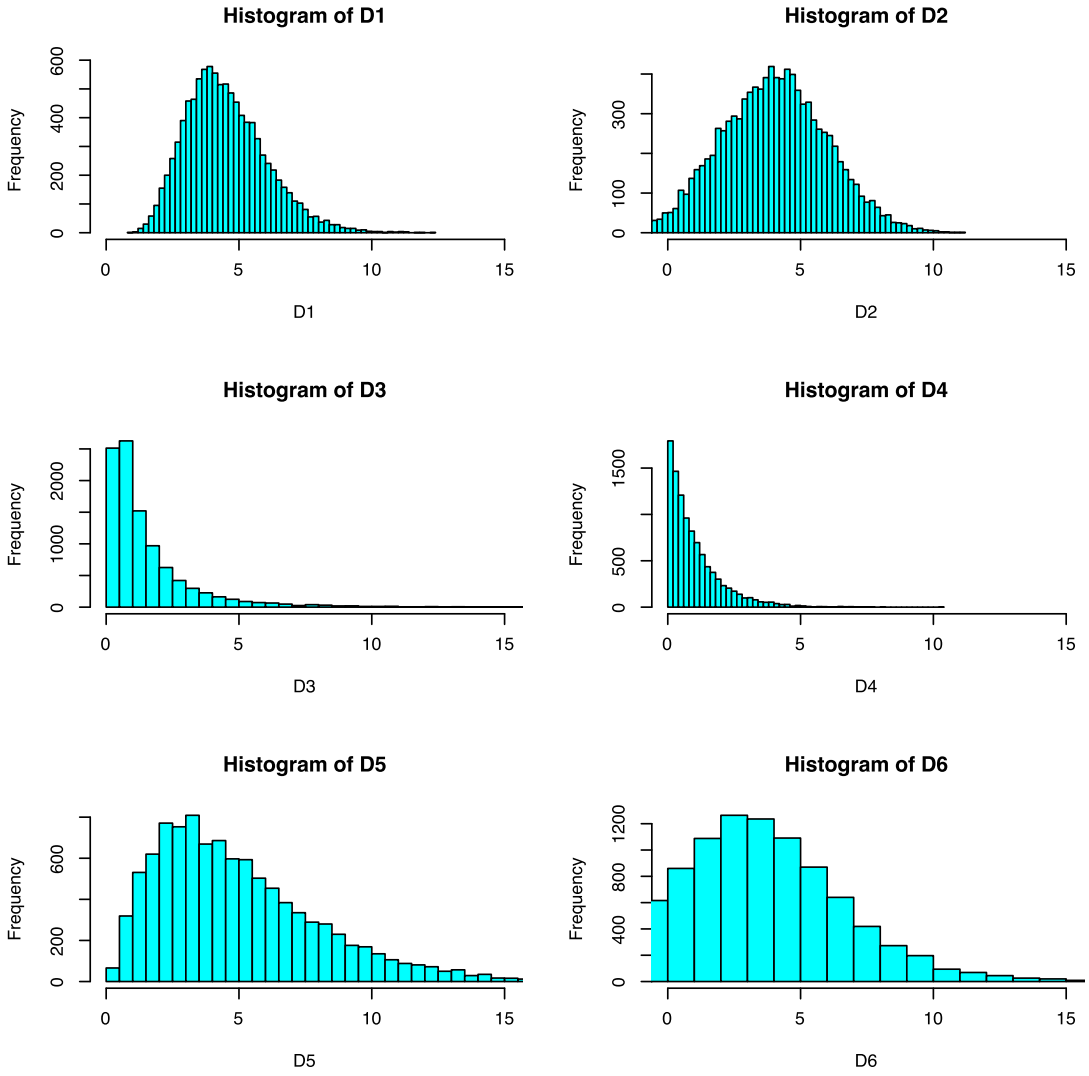
In [48]:
```
# We generate a table whose _columns_ are random samples from different distributions.

# Each sample is of size N:

N = 10000

D1 = rgamma( N, 9, 2 )
D2 = rnorm( N, 4, 2 )
D3 = rlnorm( N, 0, 1 )
D4 = rexp( N, 1 )
D5 = rchisq( N, 5 )
D6 = rlogis( N, 3, 2 )

# All parameter values in this assignment will be integers !

Table = round(cbind( D1,D2,D3,D4,D5,D6 ),8)

colnames(Table) = c("D1","D2","D3","D4","D5","D6")

# print the first few lines of the (N x 6) table:

head(Table)
```

Out[48]:

D1	D2	D3	D4	D5	D6
4.53256780	4.12387032	1.46654007	0.07384185	4.14580352	6.21513567
7.1378018	6.8581740	0.2379494	0.1476523	5.3174948	3.1291521
3.3713903	6.2437282	0.2138276	0.1699299	1.5583491	0.6543210
2.7725880	5.5875745	0.4583172	0.3767378	2.8429449	1.9559299
7.2775941	5.2902876	0.9740191	2.6121070	5.9899608	6.7003783
8.7408465	2.7770890	2.7271062	0.5956197	2.3983975	8.8628316

1.3 Histograms permit visualization of each column/sample in the Table

In [52]:
```
opar = par(mfrow = c(3,2))  # make a 3x2 grid of plots

# make 3 rows of plots, with 2 plots per row

B=60
hist( D1, col="cyan", xlim=c(0,15), breaks=B )
hist( D2, col="cyan", xlim=c(0,15), breaks=B )
hist( D3, col="cyan", xlim=c(0,15), breaks=B )
hist( D4, col="cyan", xlim=c(0,15), breaks=B )
hist( D5, col="cyan", xlim=c(0,15), breaks=B )
hist( D6, col="cyan", xlim=c(0,15), breaks=B )

par(opar)  # restore previous values of plotting parameters
```



1.4 You should handle these kinds of distributions:

```
In [50]: Distribution_name = c(
             "normal",
             "t",
             "chi-squared",
             "lognormal",
             "exponential",
             "gamma",
             "logistic"
         )

         Distribution_can_have_negative_values = c(
             TRUE,
             TRUE,
```

```
    FALSE,
    FALSE,
    FALSE,
    FALSE,
    TRUE
)

Distribution_function = c(
    dnorm,
    dt,
    dchisq,
    dlnorm,
    dexp,
    dgamma,
    dlogis
)

Distribution_color = c(
    "blue",
    "cyan",
    "green",
    "gold",
    "magenta",
    "red",
    "purple"
)

add_curve = function( dist_name, p ) {
    if (dist_name == "normal")      curve( dnorm(x, p[1], p[2] ),    col="blue",   lwd=2, add="
    if (dist_name == "t")           curve( dt(x, p[1], p[2], p[3] ), col="cyan",   lwd=2, add="
    if (dist_name == "chi-squared") curve( dnorm(x, p[1] ),          col="green",  lwd=2, add="
    if (dist_name == "lognormal")   curve( dlnorm(x, p[1], p[2] ),   col="gold",   lwd=2, add="
    if (dist_name == "exponential") curve( dexp(x, p[1] ),           col="magenta",lwd=2, add="
    if (dist_name == "gamma")       curve( dgamma(x, p[1], p[2] ),   col="red",    lwd=2, add="
    if (dist_name == "logistic" )   curve( dlogis(x, p[1], p[2] ),   col="purple", lwd=2, add="
}
```

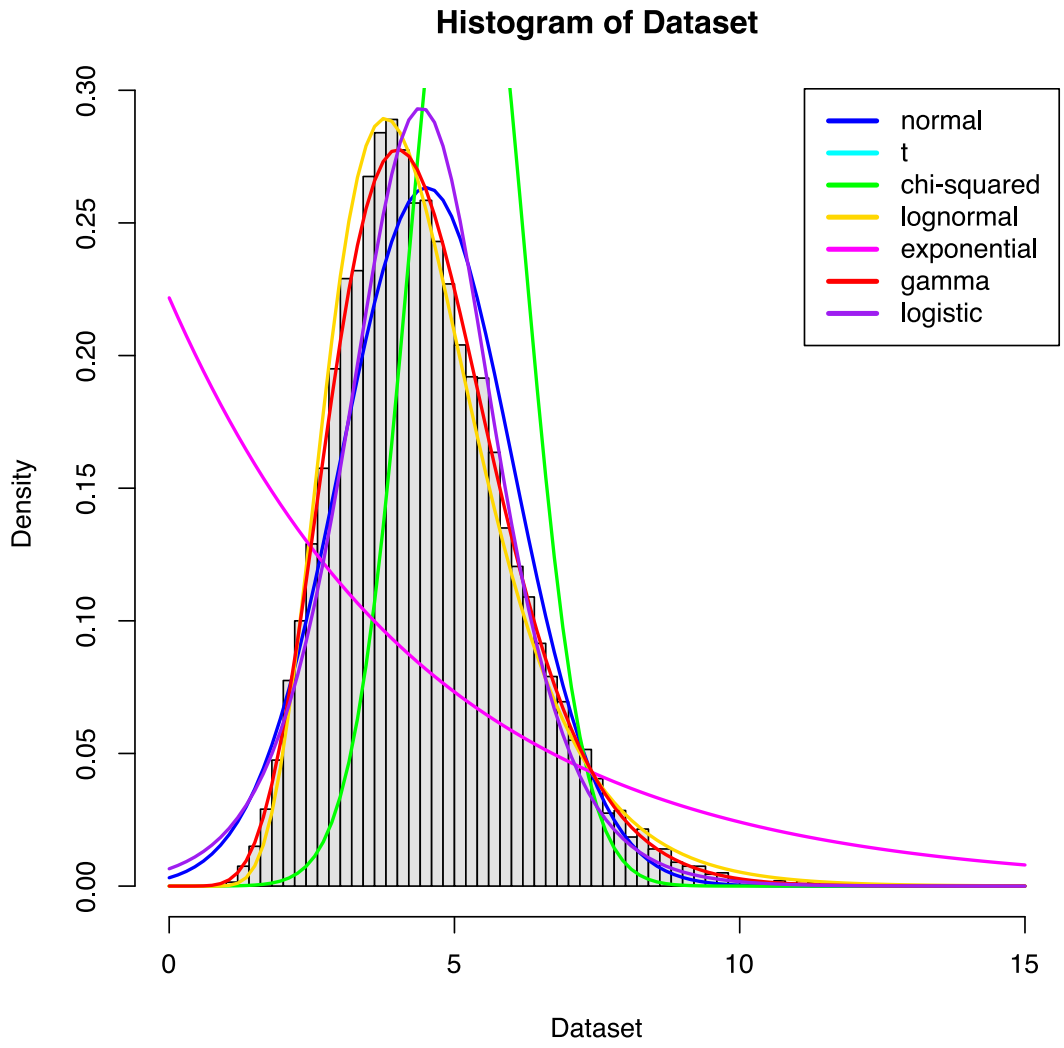
1.5 Sample analysis of the data in R, with fitdistr:

In [51]:
```
n = nrow(Table)
p = ncol(Table)

for (j in 1:p) {
    Dataset = Table[,j]    #  j-th dataset = j-th column of the data table
    cat(sprintf("\ntrying Dataset %d:\n", j))

    Dataset_is_nonnegative = !any( Dataset < 0 )
    if (Dataset_is_nonnegative) {
        cat("Dataset is nonnegative\n")
    } else {
        cat("Dataset has some negative values, so it cannot follow nonnegative distributions\n"
    }

    hist( Dataset, col="gray90", xlim=c(0,15), breaks=50, probability=TRUE )
```

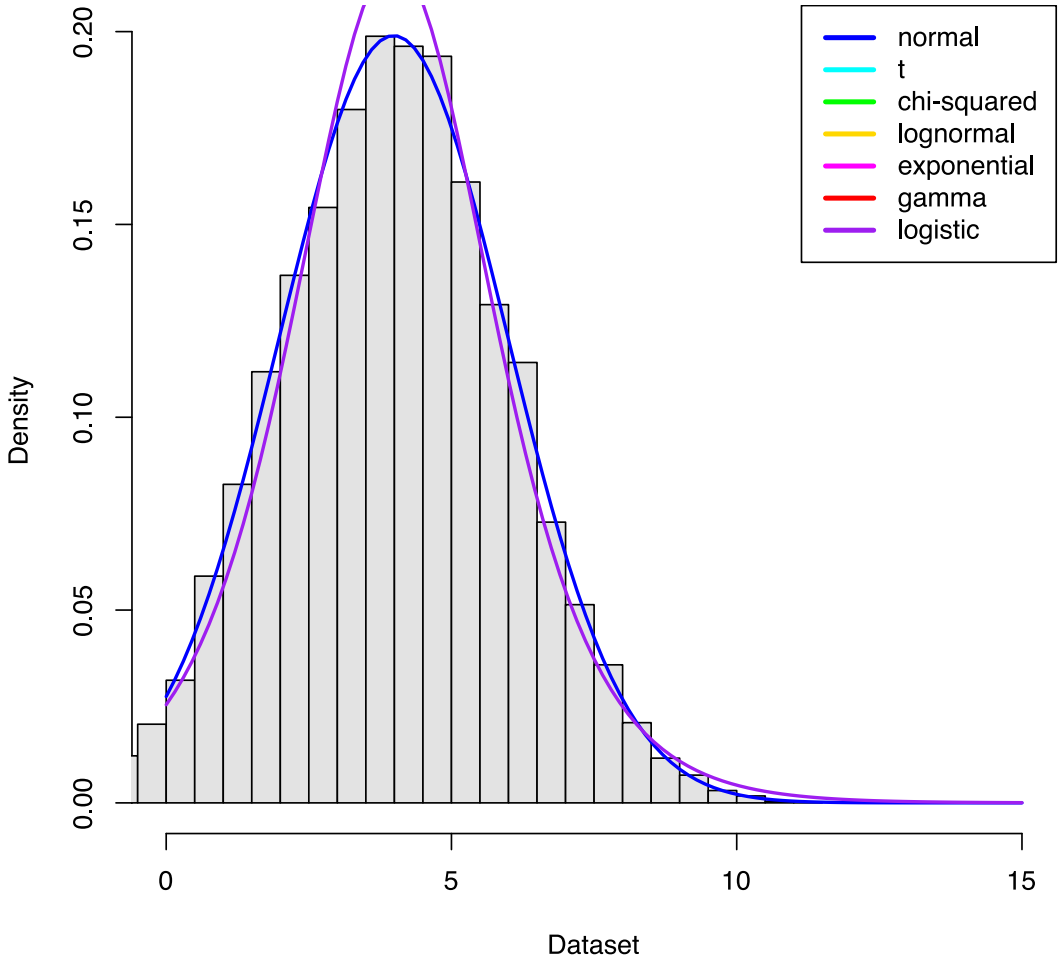
```
            # display a histogram for each column Dataset

            legend( "topright", Distribution_name, col=Distribution_color, lwd=3 )

            for (i in 1:length(Distribution_name)) {
                dist_name = Distribution_name[i]
                if (Distribution_can_have_negative_values[i] || Dataset_is_nonnegative) {
                    # don't fit a nonnegative distribution to data that is negative

                    if (dist_name == "chi-squared") {  # fitdistr requires special handling of chi-squ
                        fit = suppressWarnings( fitdistr( Dataset, dist_name,
                                                list(df=round(mean(Dataset))), method="BFGS" )
                    } else {
                        fit = suppressWarnings( fitdistr( Dataset, dist_name ) )
                    }

                    # "fit" is the object returned by fitdistr, describing the fit

                    fitted_parameters = fit$estimate
                    log_likelihood = fit$loglik

                    parameter_value_string = paste(round(fitted_parameters), collapse=" ")
                    # we round the parameter values so that they are integers.

                    # This is what the output is supposed to look like:
                    cat(sprintf("%s %s\n", dist_name, parameter_value_string))

                    # To show how good the fit is, we also print the log-likelihood here
                    cat(sprintf("               log-likelihood = %f\n", log_likelihood))

                    add_curve( dist_name, fitted_parameters )  # show the fit on the histogram

                    # The optimal distribution is the one with maximum-likelihood
                    #  (and:  maximum-likelihood == maximum-log-likelihood).
                    # Your program needs to determine which distribution maximizes this.
                }
            }
        }
```

```
trying Dataset 1:
Dataset is nonnegative
normal 5 2
               log-likelihood = -18341.937944
t 4 1 13
               log-likelihood = -18287.940947
chi-squared 5
               log-likelihood = -20955.815656
lognormal 1 0
               log-likelihood = -17986.331880
exponential 0
               log-likelihood = -25061.397949
gamma 9 2
               log-likelihood = -17916.670936
```

```
logistic 4 1
               log-likelihood = -18318.684765

trying Dataset 2:
Dataset has some negative values, so it cannot follow nonnegative distributions
```

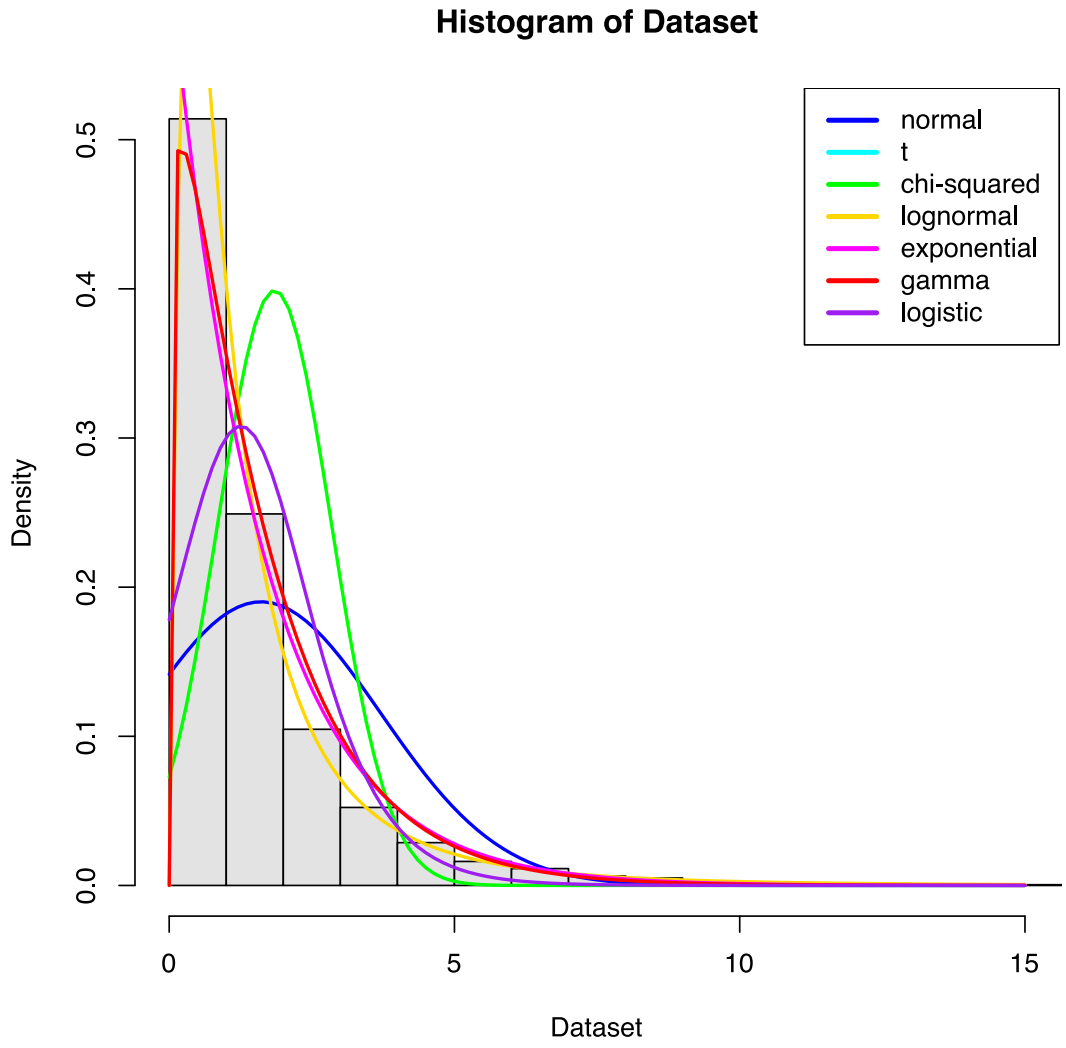



```
normal 4 2
               log-likelihood = -21145.908478
t 4 2 73
               log-likelihood = -21149.296562
logistic 4 1
               log-likelihood = -21253.423182

trying Dataset 3:
Dataset is nonnegative
```



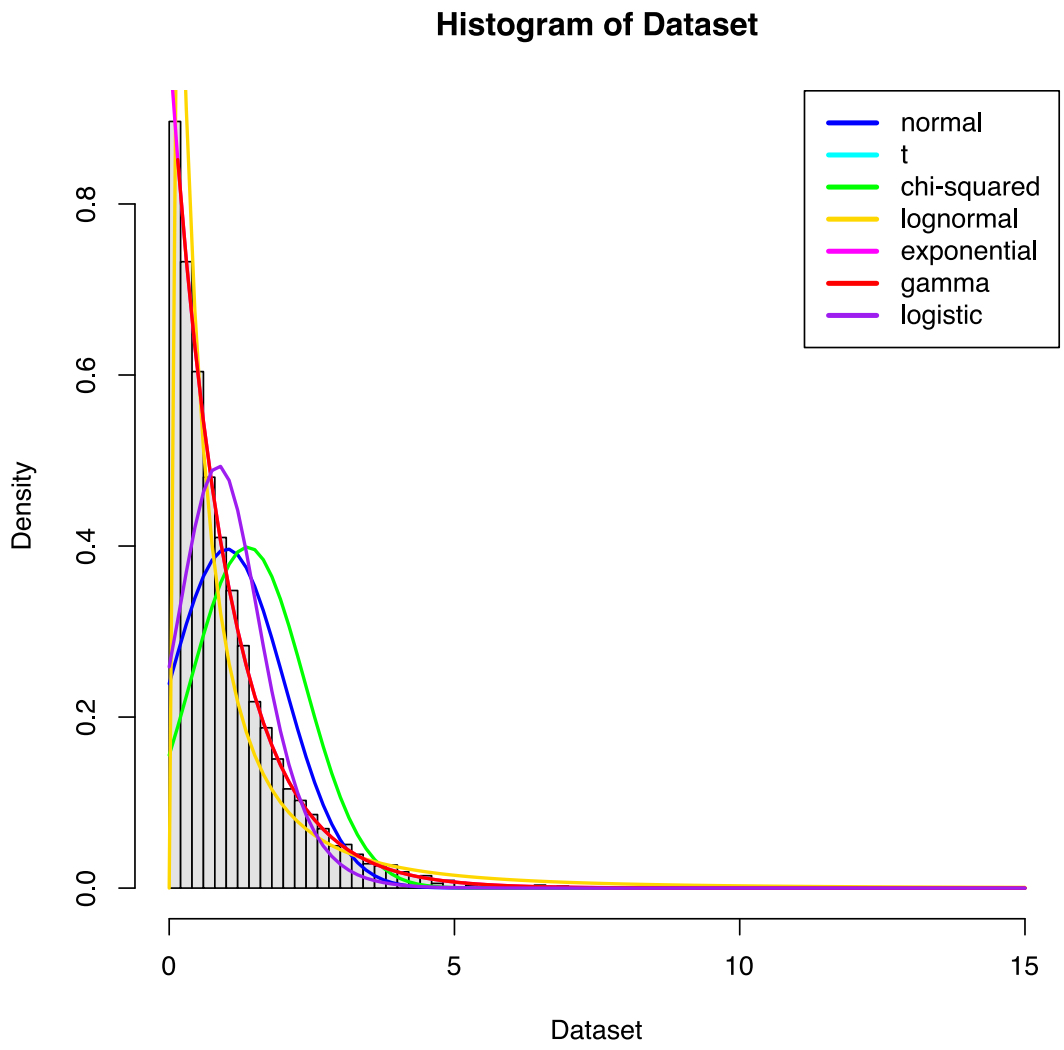

```
normal 2 2
              log-likelihood = -21597.612168
t 1 1 1
              log-likelihood = -16698.716736
chi-squared 2
              log-likelihood = -14942.206866
lognormal 0 1
              log-likelihood = -13934.857839
exponential 1
              log-likelihood = -14778.392531
gamma 1 1
              log-likelihood = -14721.341610
logistic 1 1
              log-likelihood = -18728.957856
```

```
trying Dataset 4:
Dataset is nonnegative
```



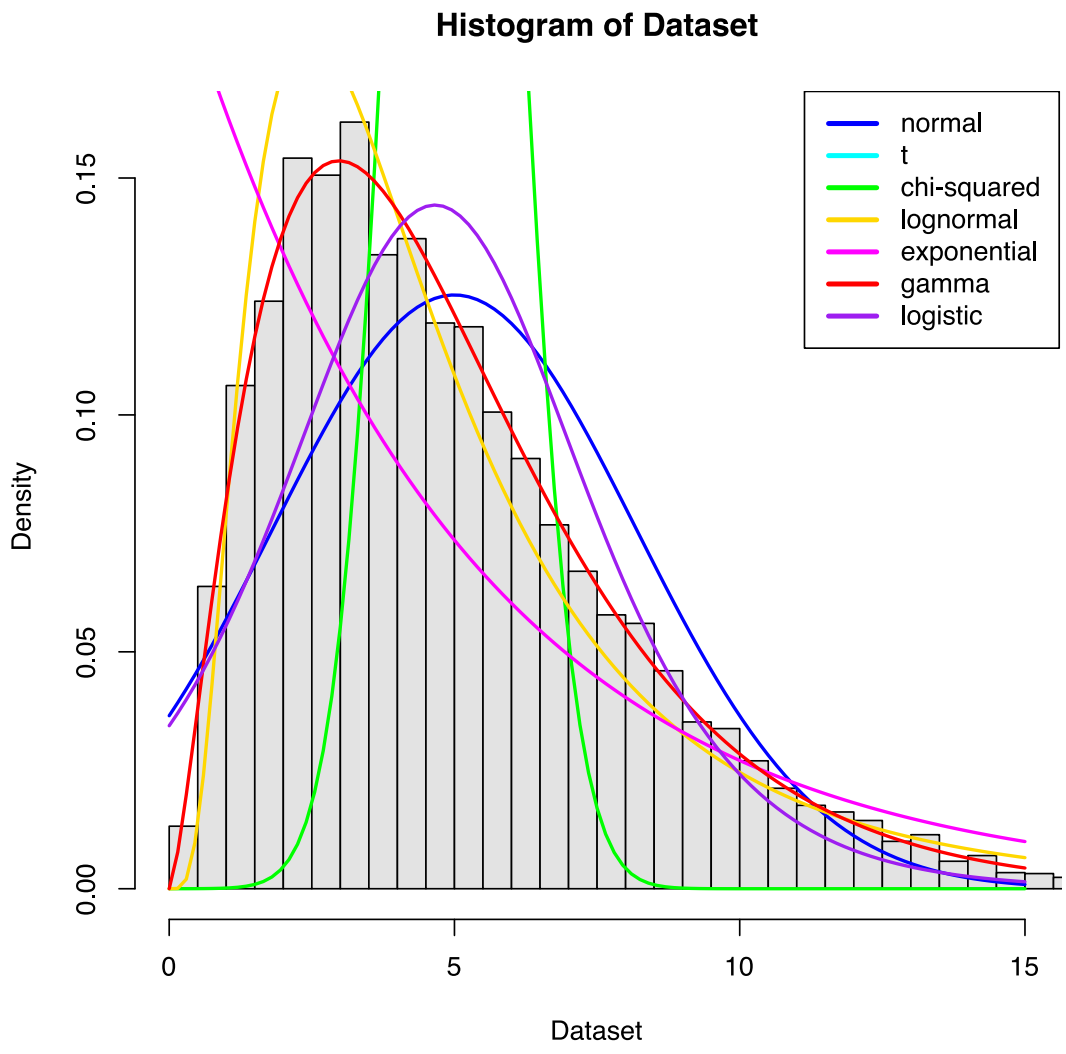

```
normal 1 1
                log-likelihood = -14248.606034
t 1 1 3
                log-likelihood = -13103.567454
chi-squared 1
                log-likelihood = -10824.495582
lognormal -1 1
                log-likelihood = -11099.145141
exponential 1
                log-likelihood = -10123.494203
gamma 1 1
                log-likelihood = -10123.493032
```

```
logistic 1 1
               log-likelihood = -13465.695558

trying Dataset 5:
Dataset is nonnegative
```

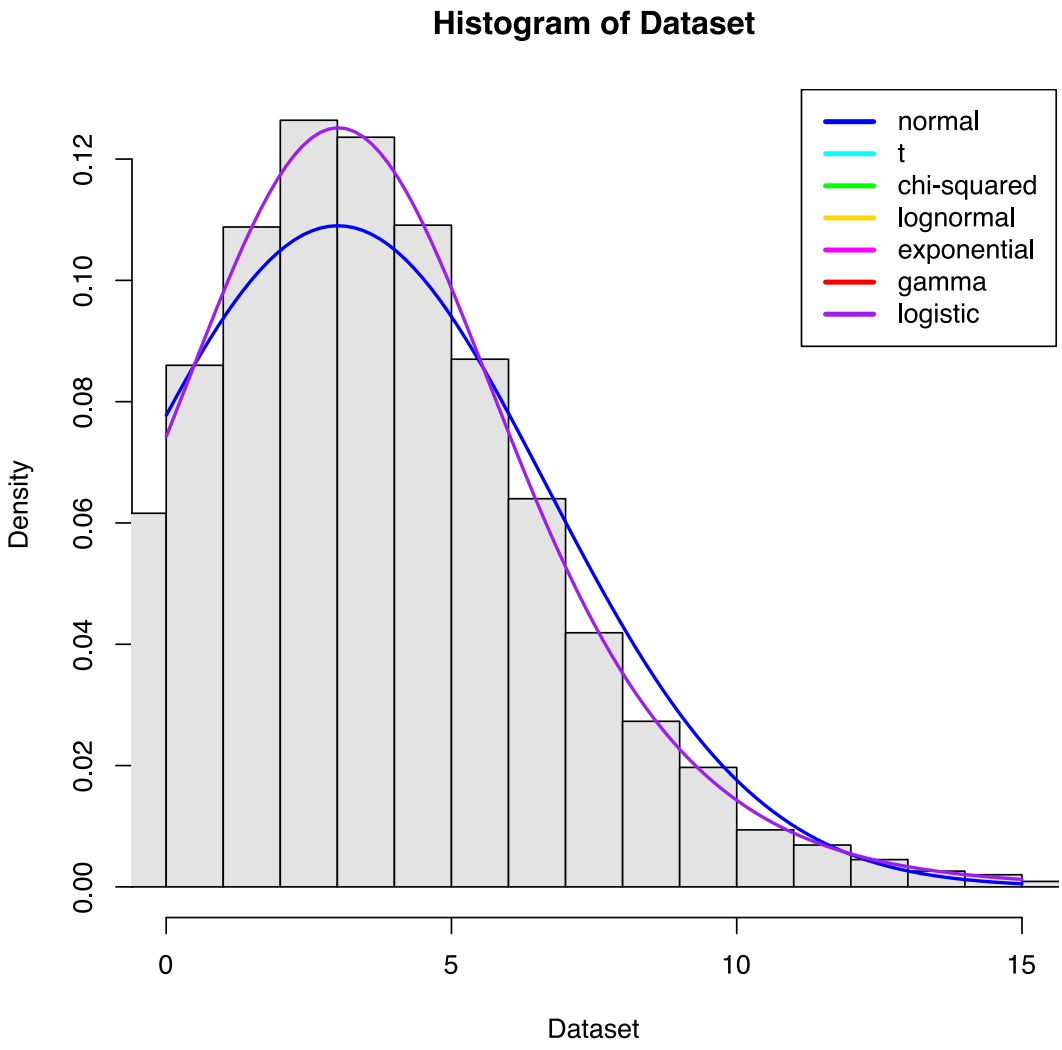



```
normal 5 3
               log-likelihood = -25768.940694
t 5 3 6
               log-likelihood = -25487.619038
chi-squared 5
               log-likelihood = -24274.332114
lognormal 1 1
               log-likelihood = -24585.026963
exponential 0
```

```
                log-likelihood = -26093.799784
gamma 2 0
                log-likelihood = -24273.951478
logistic 5 2
                log-likelihood = -25516.785583

trying Dataset 6:
Dataset has some negative values, so it cannot follow nonnegative distributions
```




```
normal 3 4
                log-likelihood = -27163.454578
t 3 3 6
                log-likelihood = -26947.991305
logistic 3 2
                log-likelihood = -26950.184779
```



2 Problem: Fit a distribution to each input dataset.

Specifically:

Step 1. Extend the program above to handle ANY input table ..... and also handle the Beta and Weibull distributions.

These distributions are supported by fitdistr. Note: values from the Beta distribution are always in the interval [0,1], so any input data that goes outside this range cannot be from the Beta distribution. Warning: The optim optimizer in R does not handle the dbeta function well. Error messages like initial value in 'vmmin' is not finite are a result. Remember: any input data that goes outside the interval [0,1] cannot be from the Beta distribution. Your program does not have to handle the F distribution, Negative Binomial distribution, Poisson distribution, etc.

Your R program might be an extension of the outline in the notebook.

Step 2. The output of your notebook should be a CSV file "HW0_output.csv"

Your output CSV file "HW0_output.csv" should look like this:

If your program had been given the Table above as input, it should print the following CSV file, a table with six rows, and FOUR columns:

There should be NO header line in this file. Each row has FOUR fields: distribution name, and at most three parameter values. (The t distribution takes 3 parameter values, for example.) If the input table has p columns (i.e., p random samples), the output file should have p rows. The parameters in this assignment will always be integers, so the printed output should always have integer parameter values.

Step 3. Run your notebook using the file "HW0_test.csv" as input.

Step 4. Submit your output CSV file and revised notebook on CCLE.

Upload your .ipynb and your .csv file for Assignment "HW0". Both are required.

We will use Paul Eggert's Late Policy: The number of days late is $N = 0$ for the first 24 hrs, $N = 1$ for the next 24 hrs, etc., and if you submit an assignment H hours late, $2^{\lfloor H/24 \rfloor}$ points are deducted.