R Notebook Lab-2

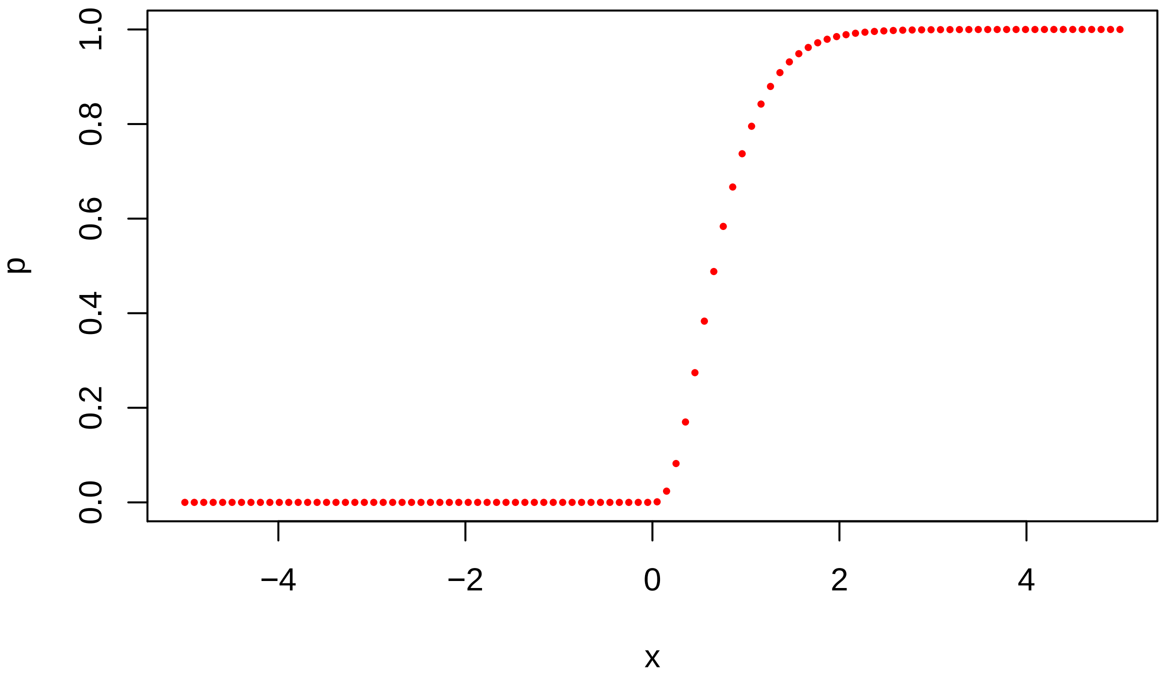
Xuanpei Ouyang

Basic simulation

General rule for simulation —-

p – p is short for probability, 'p' is used for getting distribution function value Pr(X<= p)

```
x<- seq(-5,5,length.out = 100);
p<- pgamma(x,shape = 3,scale = 1/4);
plot(x,p,pch=16,cex=.5,col="red");
```



q – q is short for quantile, 'q' is used for getting quantiles.

```
q1<- qnorm(.975)
q1
```

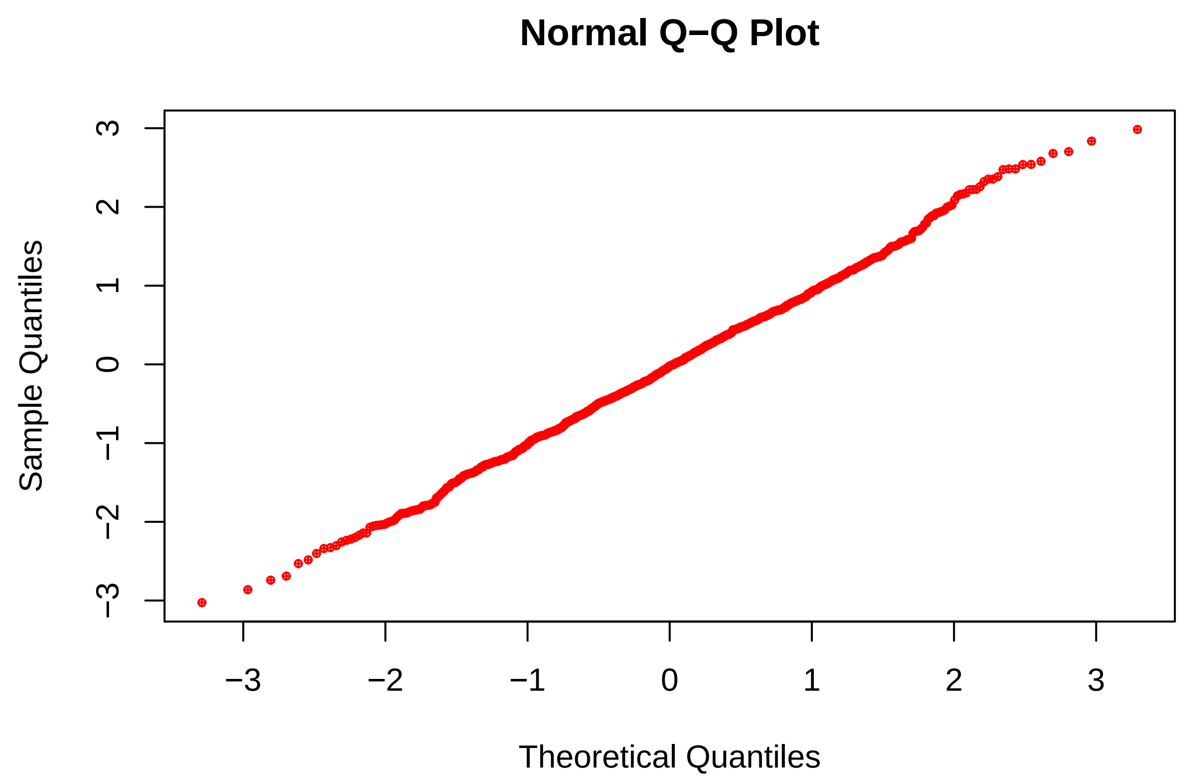
```
## [1] 1.959964
```

```
q2<- qnorm(.025)
q2
```

```
## [1] -1.959964
```

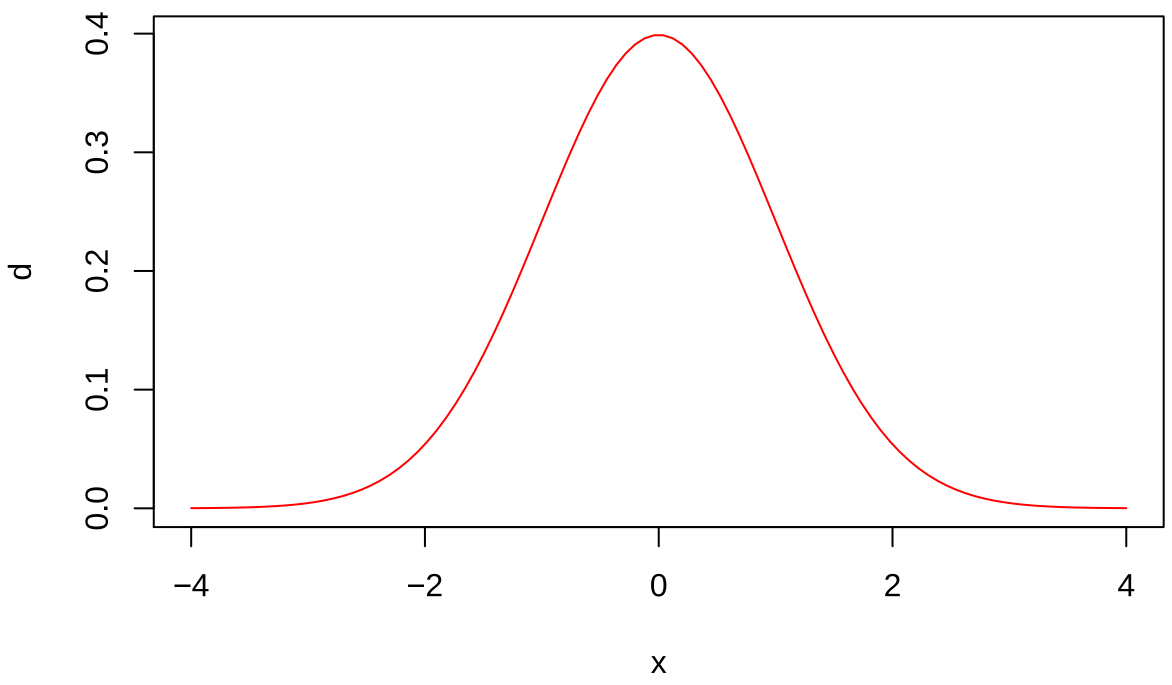
r – r is short for random, 'r' is used for simulating random sample from distribution.

```
x<- rnorm(1000);
qqnorm(x,col="red",pch=10,cex=.5)
```



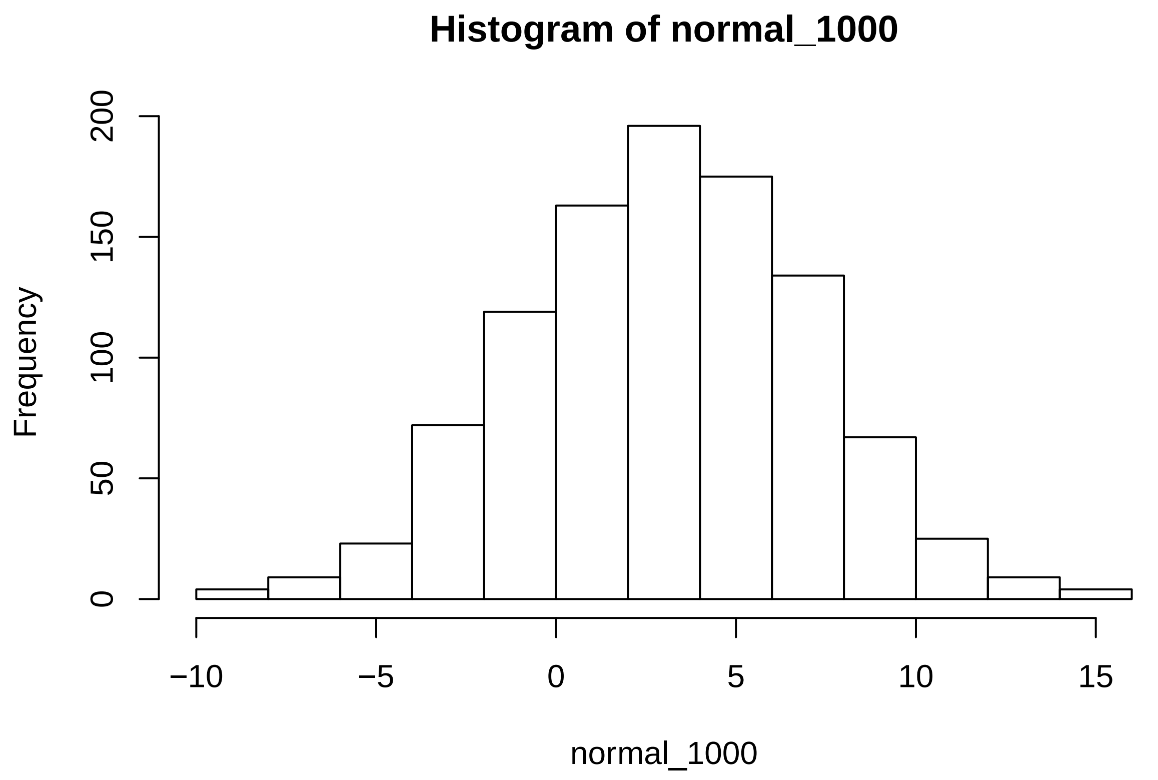
d – d is short for density, ‘d’ is used for getting density of a probability distribution

```r
x<- seq(-4,4,length.out = 100)
d<- dnorm(x)
plot(x,d,col="red",pch=16, ty='l')
```

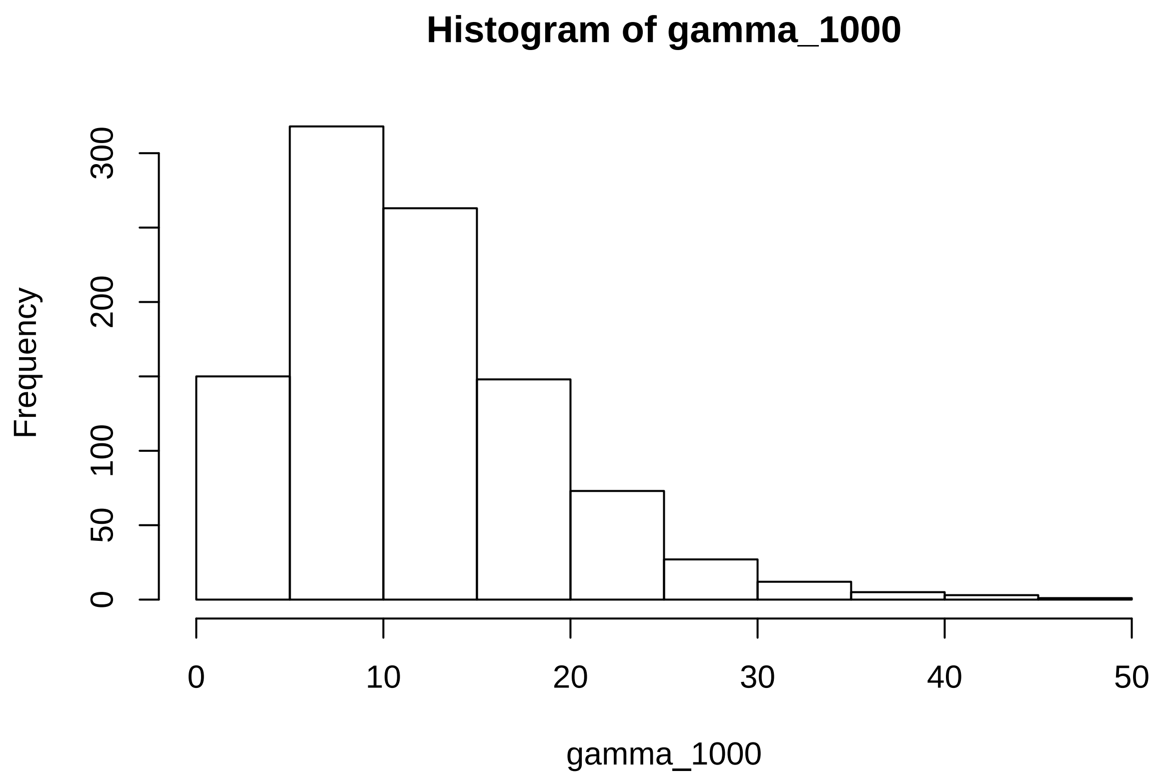


1. Simulate 1000 observation from Normal(3,4), Gamma(3,1/4), Exp(.3), Cauchy(3,5), t(df=5) , chi-sq(df = 5), plot their histograms.

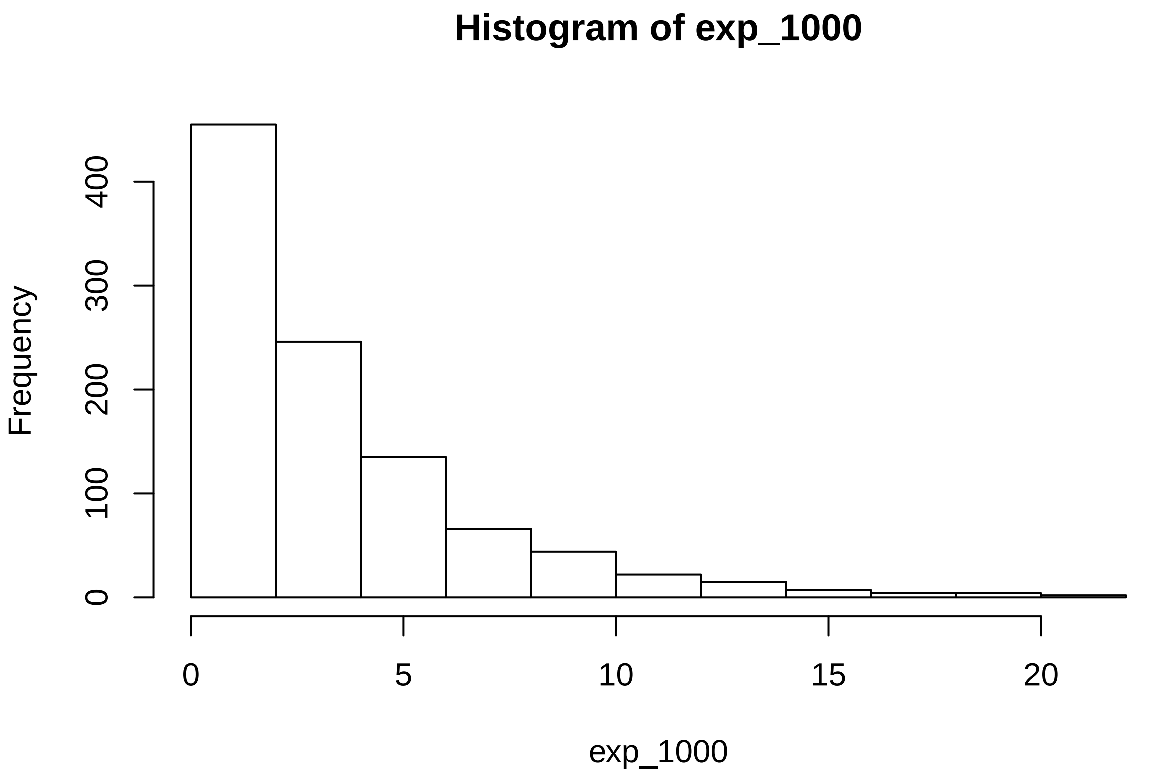
```r
normal_1000 = rnorm(1000, 3, 4)
hist(normal_1000)
```



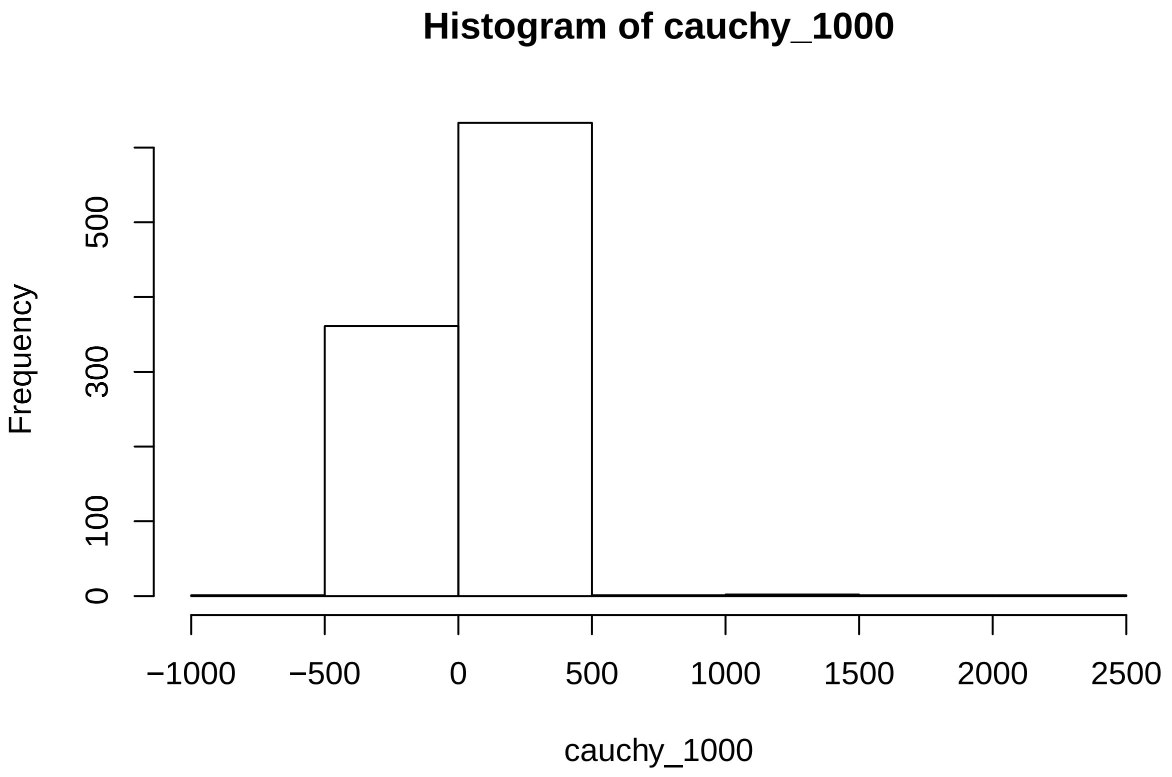
```r
gamma_1000 = rgamma(1000, 3, 1/4)
hist(gamma_1000)
```



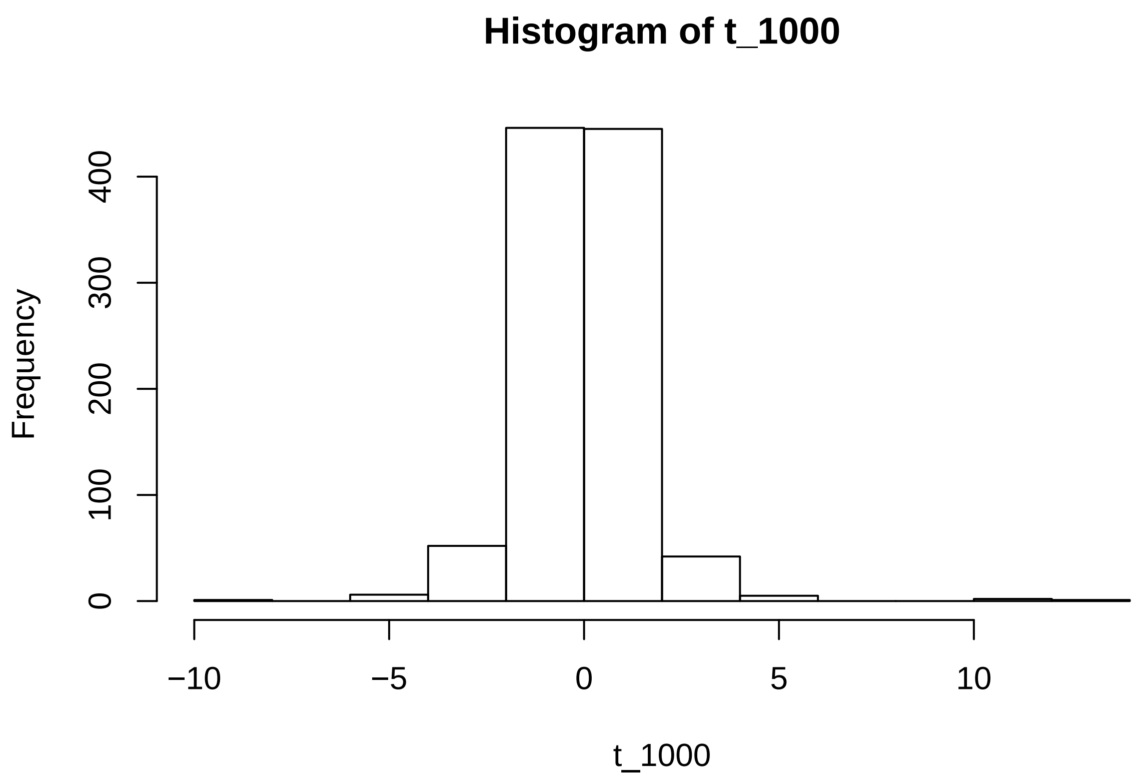
```r
exp_1000 = rexp(1000, 0.3)
hist(exp_1000)
```



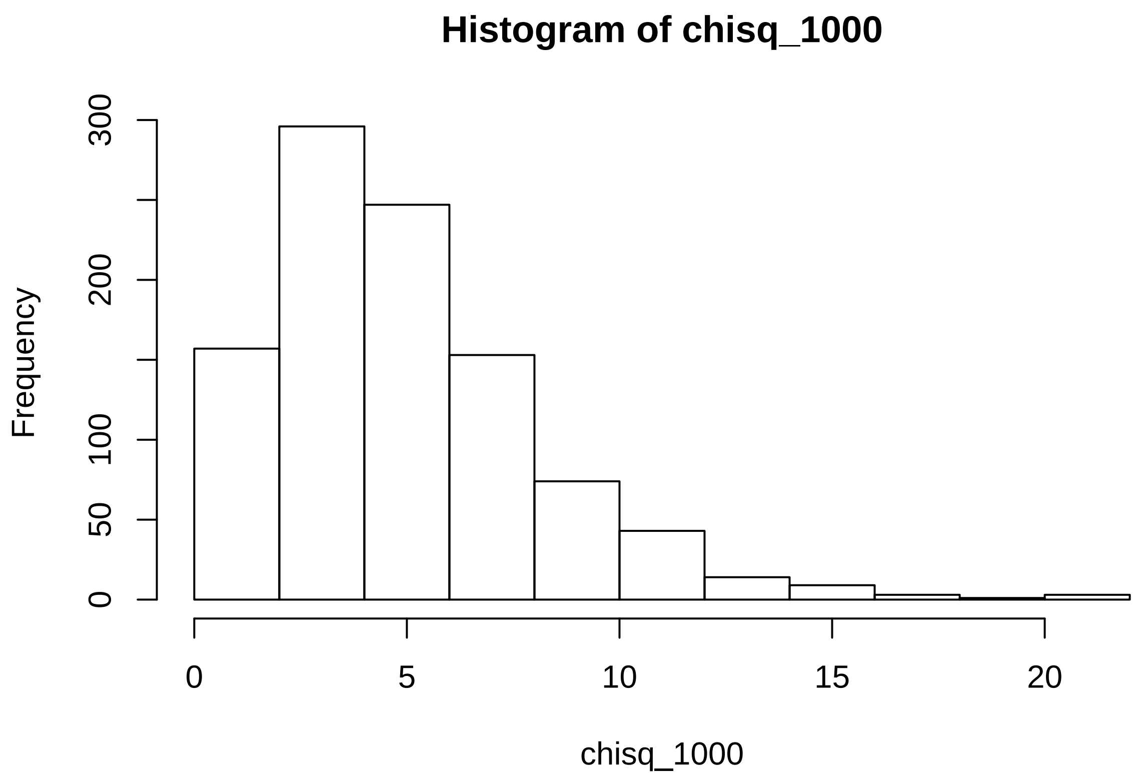
```r
cauchy_1000 = rcauchy(1000, 3, 5)
hist(cauchy_1000)
```



```
t_1000 = rt(1000, df = 5)
hist(t_1000)
```



```
chisq_1000 = rchisq(1000, df = 5)
hist(chisq_1000)
```



2. Find 2.5% and 97.5% quantiles for the above distributions.

```r
norm_quantile_2.5 = qnorm(0.025, 3, 4)
norm_quantile_2.5
```

```
## [1] -4.839856
```

```r
norm_quantile_97.5 = qnorm(0.975, 3, 4)
norm_quantile_97.5
```

```
## [1] 10.83986
```

```r
gamma_quantile_2.5 = qgamma(0.025, 3, 1/4)
gamma_quantile_2.5
```

```
## [1] 2.474688
```

```r
gamma_quantile_97.5 = qgamma(0.975, 3, 1/4)
gamma_quantile_97.5
```

```
## [1] 28.89875
```

```r
exp_quantile_2.5 = qexp(0.025, 0.3)
exp_quantile_2.5
```

```
## [1] 0.08439269
```

```r
exp_quantile_97.5 = qexp(0.975, 0.3)
exp_quantile_97.5
```

```
## [1] 12.29626
```

```r
cauchy_quantile_2.5 = qcauchy(0.025, 3, 5)
cauchy_quantile_2.5
```

```
## [1] -60.53102
```

```r
cauchy_quantile_97.5 = qcauchy(0.975, 3, 5)
cauchy_quantile_97.5
```

```
## [1] 66.53102
```

```r
t_quantile_2.5 = qt(0.025, df = 5)
t_quantile_2.5
```

```
## [1] -2.570582
```

```r
t_quantile_97.5 = qt(0.975, df = 5)
t_quantile_97.5
```

```
## [1] 2.570582
```

```r
chisq_quantile_2.5 = qchisq(0.025, df = 5)
chisq_quantile_2.5
```

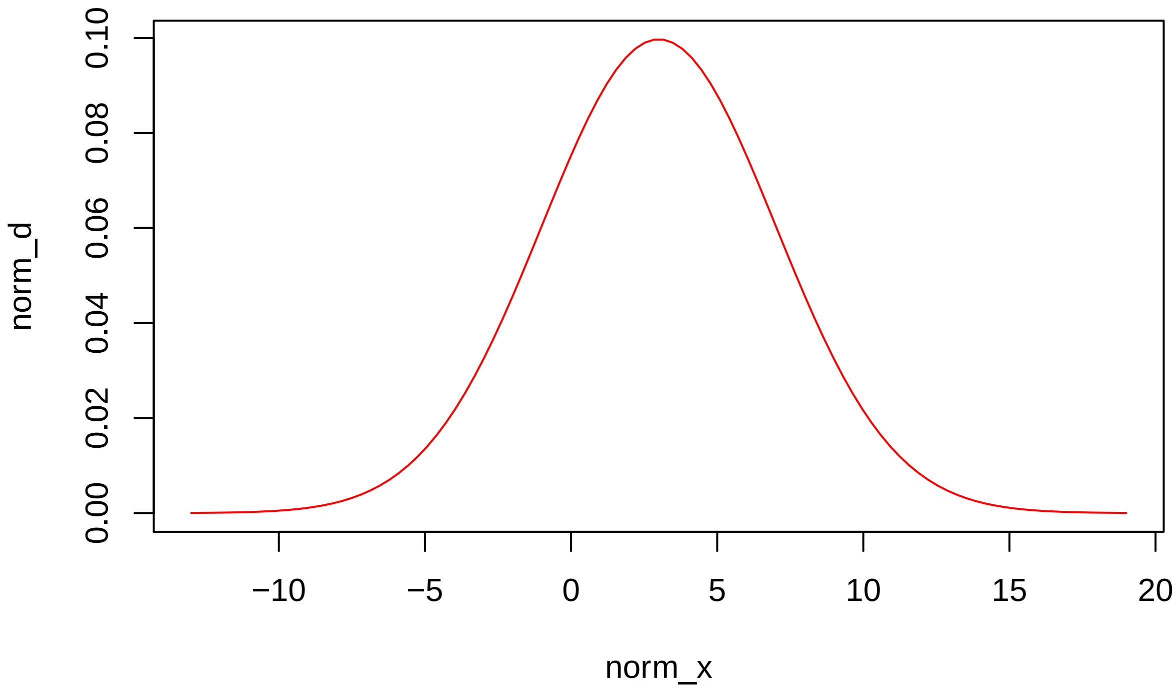
```
## [1] 0.8312116
```

```r
chisq_quantile_97.5 = qchisq(0.975, df = 5)
chisq_quantile_97.5
```

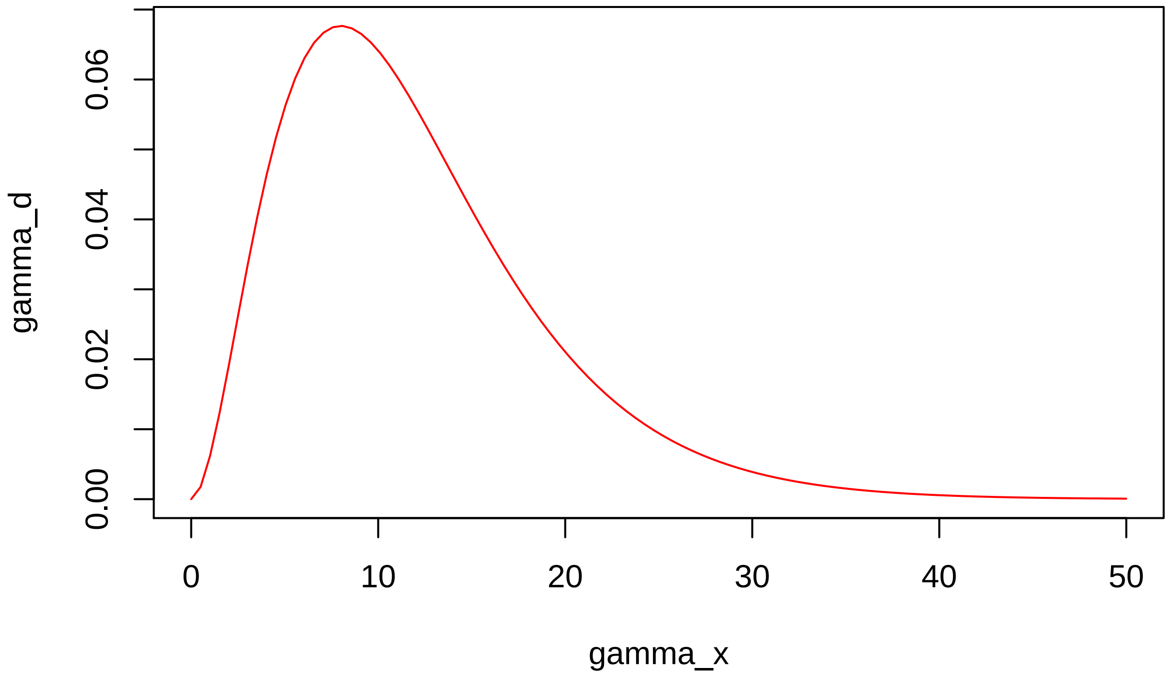
```
## [1] 12.8325
```

3. Plot density for Normal(3,4), Gamma(3,1/4), Exp(.3), use - ‘d’

```
norm_x = seq(-13, 19, length.out = 100)
norm_d = dnorm(norm_x, 3, 4)
plot(norm_x, norm_d, col="red", pch=16, ty="l")
```

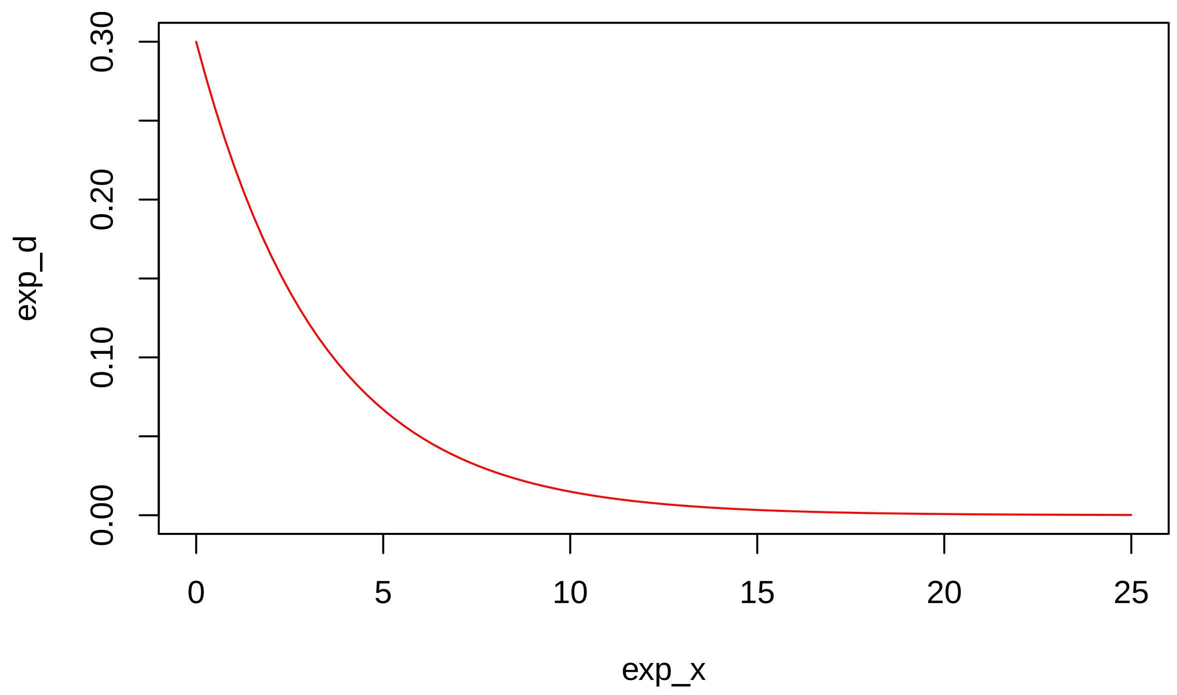


```
gamma_x = seq(0, 50, length.out = 100)
gamma_d = dgamma(gamma_x, 3, 1/4)
plot(gamma_x, gamma_d, col="red", pch=16, ty="l")
```



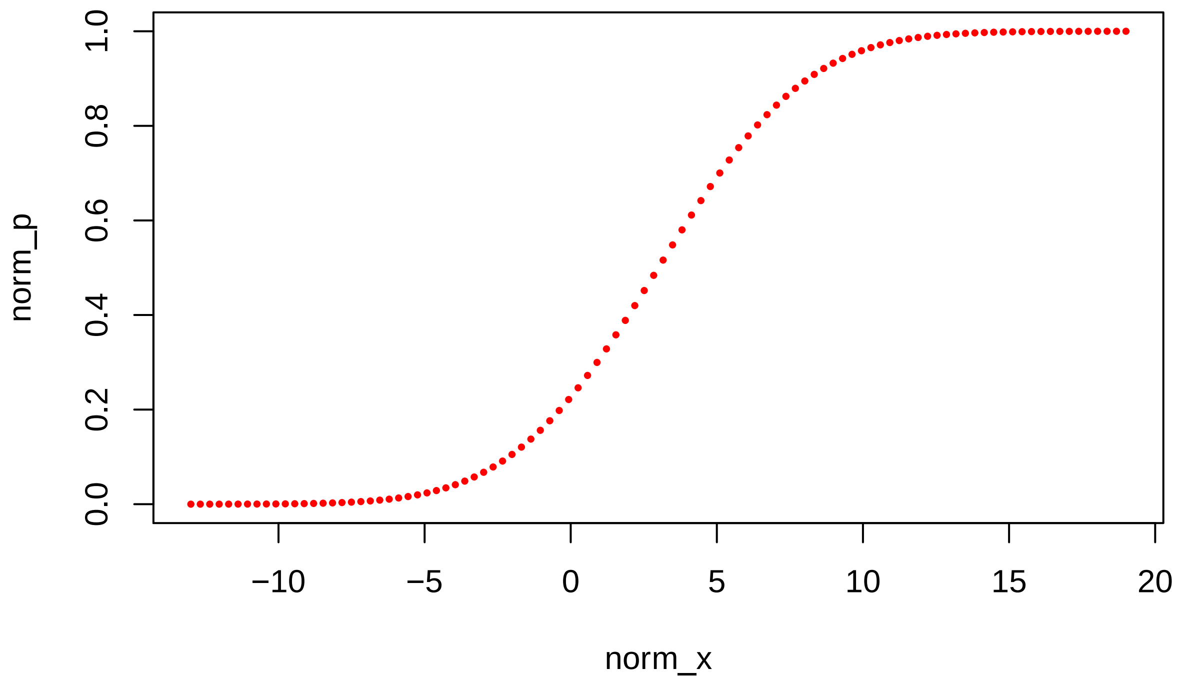
```
exp_x = seq(0, 25, length.out = 100)
exp_d = dexp(exp_x, 0.3)
```

```r
plot(exp_x, exp_d, col="red", pch=16, ty="l")
```

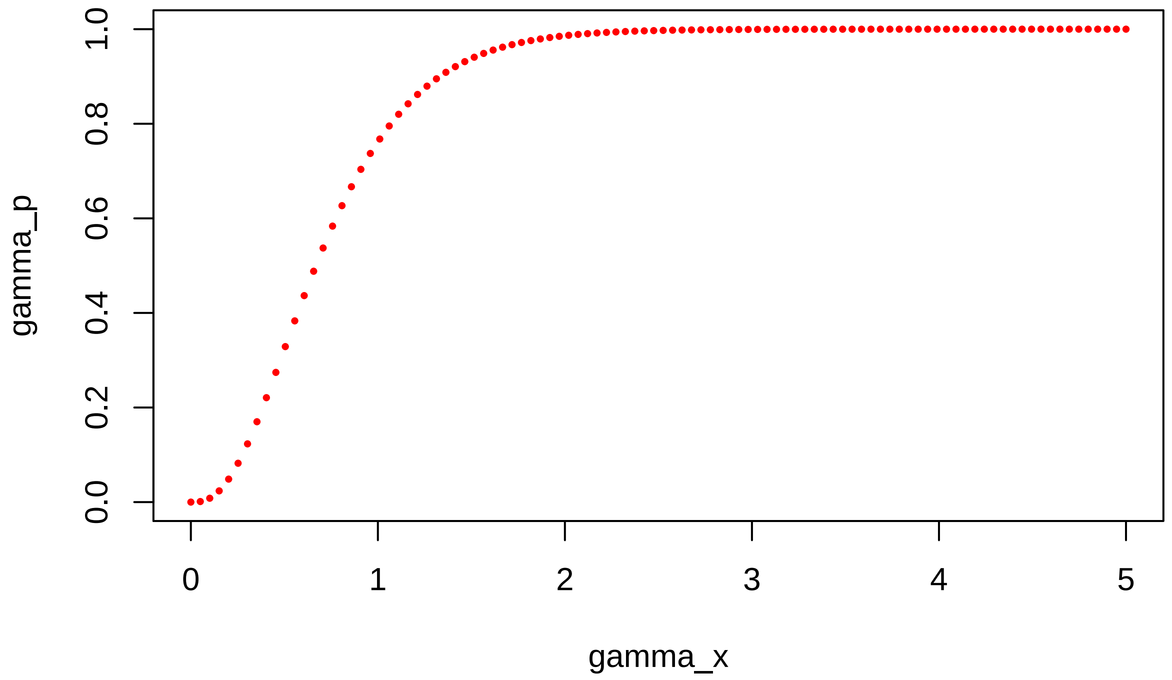


4. Plot Distribution function for Normal(3,4), Gamma(3,1/4), Exp(.3) , use – ‘p’

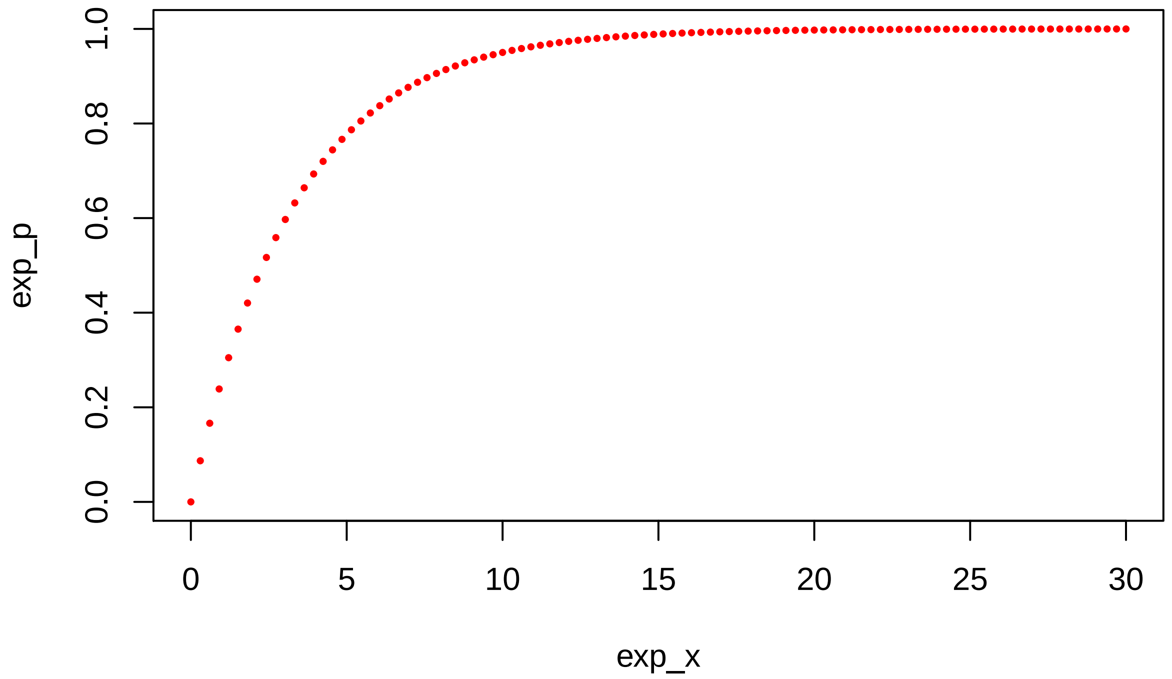
```r
norm_x = seq(-13, 19, length.out = 100)
norm_p = pnorm(norm_x, mean = 3, sd = 4)
plot(norm_x, norm_p, pch = 16, cex = 0.5, col = "red")
```



```r
gamma_x = seq(0, 5, length.out = 100)
gamma_p = pgamma(gamma_x, shape = 3,scale = 1/4)
plot(gamma_x, gamma_p, pch=16, cex=.5, col="red")
```



```r
exp_x = seq(0, 30, length.out = 100)
exp_p = pexp(exp_x, 0.3)
plot(exp_x, exp_p, pch=16, cex=.5, col="red")
```



5) —————— Simulation data from an actual linear model —————-

– generate x by seq(0,15,length.out = 30);

```r
x = seq(0, 15, length.out = 30)
x
```

```
##  [1]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
##  [7]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
## [13]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
## [19]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
## [25] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
```

– replicate x 6 times and store it in new_x, – use rep(x,6)

```r
new_x = rep(x, 6)
new_x
```

```
##   [1]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
##   [7]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
##  [13]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
##  [19]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
##  [25] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
##  [31]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
##  [37]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
##  [43]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
##  [49]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
##  [55] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
##  [61]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
##  [67]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
##  [73]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
##  [79]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
##  [85] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
##  [91]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
##  [97]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
## [103]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
## [109]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
## [115] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
## [121]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
## [127]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
## [133]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
## [139]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
## [145] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
## [151]  0.0000000  0.5172414  1.0344828  1.5517241  2.0689655  2.5862069
## [157]  3.1034483  3.6206897  4.1379310  4.6551724  5.1724138  5.6896552
## [163]  6.2068966  6.7241379  7.2413793  7.7586207  8.2758621  8.7931034
## [169]  9.3103448  9.8275862 10.3448276 10.8620690 11.3793103 11.8965517
## [175] 12.4137931 12.9310345 13.4482759 13.9655172 14.4827586 15.0000000
```

– Generate y by y = 2+3*x

```r
y = 2 + 3*new_x
y
```

```
##   [1]  2.000000  3.551724  5.103448  6.655172  8.206897  9.758621 11.310345
##   [8] 12.862069 14.413793 15.965517 17.517241 19.068966 20.620690 22.172414
##  [15] 23.724138 25.275862 26.827586 28.379310 29.931034 31.482759 33.034483
##  [22] 34.586207 36.137931 37.689655 39.241379 40.793103 42.344828 43.896552
##  [29] 45.448276 47.000000  2.000000  3.551724  5.103448  6.655172  8.206897
##  [36]  9.758621 11.310345 12.862069 14.413793 15.965517 17.517241 19.068966
##  [43] 20.620690 22.172414 23.724138 25.275862 26.827586 28.379310 29.931034
##  [50] 31.482759 33.034483 34.586207 36.137931 37.689655 39.241379 40.793103
##  [57] 42.344828 43.896552 45.448276 47.000000  2.000000  3.551724  5.103448
##  [64]  6.655172  8.206897  9.758621 11.310345 12.862069 14.413793 15.965517
##  [71] 17.517241 19.068966 20.620690 22.172414 23.724138 25.275862 26.827586
##  [78] 28.379310 29.931034 31.482759 33.034483 34.586207 36.137931 37.689655
##  [85] 39.241379 40.793103 42.344828 43.896552 45.448276 47.000000  2.000000
##  [92]  3.551724  5.103448  6.655172  8.206897  9.758621 11.310345 12.862069
##  [99] 14.413793 15.965517 17.517241 19.068966 20.620690 22.172414 23.724138
## [106] 25.275862 26.827586 28.379310 29.931034 31.482759 33.034483 34.586207
## [113] 36.137931 37.689655 39.241379 40.793103 42.344828 43.896552 45.448276
## [120] 47.000000  2.000000  3.551724  5.103448  6.655172  8.206897  9.758621
## [127] 11.310345 12.862069 14.413793 15.965517 17.517241 19.068966 20.620690
## [134] 22.172414 23.724138 25.275862 26.827586 28.379310 29.931034 31.482759
## [141] 33.034483 34.586207 36.137931 37.689655 39.241379 40.793103 42.344828
## [148] 43.896552 45.448276 47.000000  2.000000  3.551724  5.103448  6.655172
## [155]  8.206897  9.758621 11.310345 12.862069 14.413793 15.965517 17.517241
## [162] 19.068966 20.620690 22.172414 23.724138 25.275862 26.827586 28.379310
## [169] 29.931034 31.482759 33.034483 34.586207 36.137931 37.689655 39.241379
## [176] 40.793103 42.344828 43.896552 45.448276 47.000000
```

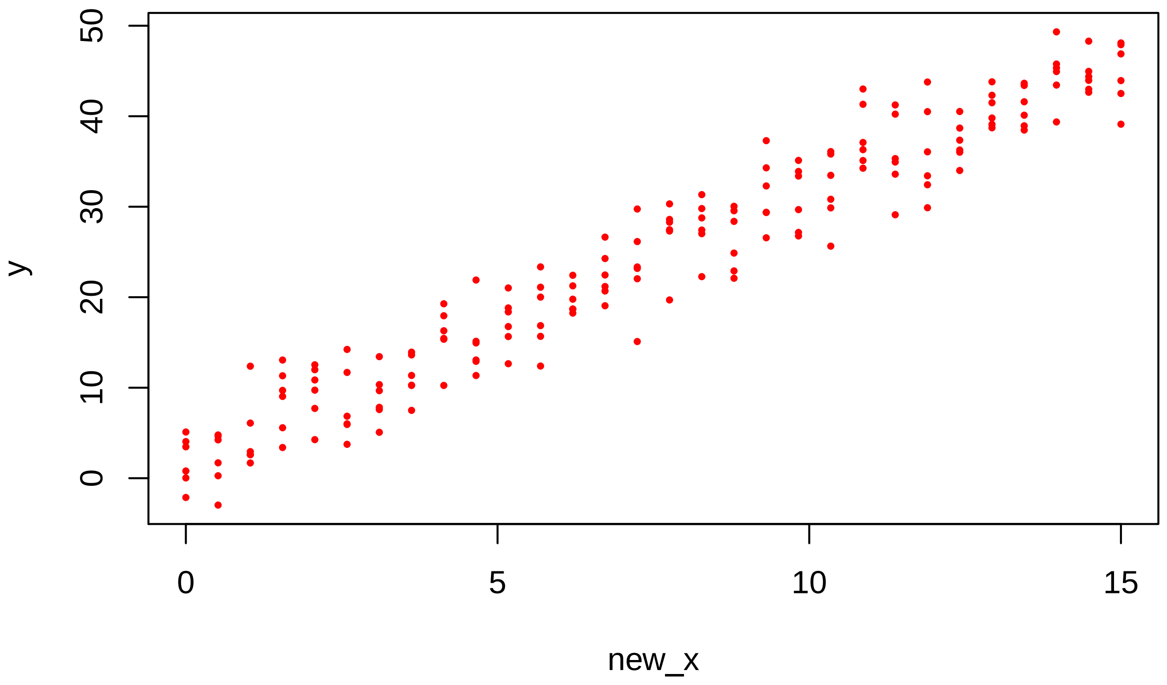
– add Gaussian noise to y (sd = 3.5)

```
y = y + rnorm(length(y), mean = 0, sd = 3.5)
y
```

```
##   [1] -2.13019487  0.27024063  2.61409731  9.03765468  7.71115709
##   [6]  5.93956447 10.34072340  7.49562314 19.27408396 11.34592848
##  [11] 18.37774155 21.10358339 18.68844687 26.64114578 15.10178214
##  [16] 30.31204833 31.33415493 30.04524544 29.38285272 26.77121099
##  [21] 33.46569213 41.31854714 40.22831652 29.88673379 36.26485954
##  [26] 39.10213280 40.11185638 43.44435066 44.95291479 43.93702026
##  [31]  4.04564239  4.65735104 12.37686392  5.57635599  4.26055179
##  [36] 11.68957960  7.83163311 11.35271865 15.44550590 15.12718301
##  [41] 18.81698253 12.39403708 19.77669186 19.05635317 22.04083467
##  [46] 28.28587643 28.76412693 29.55518825 34.29748622 33.38927190
##  [51] 35.82141480 35.10296402 33.59729824 32.42609369 38.68926284
##  [56] 39.80226415 43.64213214 45.76831312 43.96202036 42.51212323
##  [61]  5.10371695 -2.97264612  2.59133019 13.05788503 10.85411388
##  [66]  6.84644221 13.43075909 13.62091943 16.29346130 14.94796117
##  [71] 15.65226433 16.85889795 18.24490678 21.18263761 23.34198669
##  [76] 27.31368941 27.42446070 22.90676791 37.30082477 35.11661382
##  [81] 30.82279317 43.00762304 29.11116975 40.50624594 37.35205015
##  [86] 41.49074929 41.59825752 49.32275320 42.64617256 48.09995185
##  [91]  0.79384890  1.69727545  1.68344772 11.31826814 12.52901350
##  [96]  6.00489124  9.67308051 10.24787439 10.24927769 21.89953247
## [101] 16.75384465 20.01058128 18.69096907 24.27646712 26.14997433
## [106] 27.45838797 22.27261626 24.87617587 32.29681191 27.15790585
## [111] 25.64233133 36.30486106 41.24700763 36.06423227 40.52055250
## [116] 38.72015692 38.47054720 39.36682738 44.37969548 46.88792987
## [121]  3.46720672  4.23322304  2.93502882  9.69967703 11.98764928
## [126] 14.22645926  5.07321112 13.92838919 15.35317194 13.06226762
## [131] 12.64969132 23.35034509 21.25520422 20.68707145 23.18060432
## [136] 28.59583675 27.02062155 28.37909398 26.56758867 33.89834699
## [141] 36.09227719 34.25304308 34.93836561 33.41689958 36.02088786
## [146] 43.80444014 43.40036034 45.34271140 42.97608766 39.11632701
## [151]  0.02856075  4.77720450  6.08727985  3.39104753  9.72751475
## [156]  3.74368805  7.58678025 10.27176434 17.95007144 12.91258049
## [161] 21.02220991 15.67454019 22.42360361 22.45500021 29.74570801
## [166] 19.69790030 29.79565902 22.09337946 29.36099277 29.67650845
## [171] 29.87308836 37.09882469 35.32027258 43.77946507 34.00374557
## [176] 42.31446247 38.93927511 44.93456679 48.29230530 47.90587831
```

– plot x vs y

```
plot(new_x, y, pch = 16, cex = 0.5, col = "red")
```



Transformation with real data .

—– Example 1: Emission data

The data consists of the emissions of three different pollutants from 46 different engines.

We will be analyzing the carbon-monoxide data here. Lets see if the data is skewed or not ?

6) Load the data, use summary() to get a summary of the variables, Calculate the (q3-median)/(median-q1).

```
engine = read.csv("~/Desktop/STAT 151A/STAT-151A/lab/lab2/table_7_3.csv", sep=",", head = TRUE)
summary(engine)
```

```
##        en              hc               co              nox
##  Min.   : 1.00   Min.   :0.3400   Min.   : 1.850   Min.   :0.490
##  1st Qu.:12.75   1st Qu.:0.4375   1st Qu.: 4.388   1st Qu.:1.110
##  Median :24.50   Median :0.5100   Median : 5.905   Median :1.315
##  Mean   :24.00   Mean   :0.5502   Mean   : 7.879   Mean   :1.340
##  3rd Qu.:35.25   3rd Qu.:0.6025   3rd Qu.:10.015   3rd Qu.:1.495
##  Max.   :46.00   Max.   :1.1000   Max.   :23.530   Max.   :2.940
```

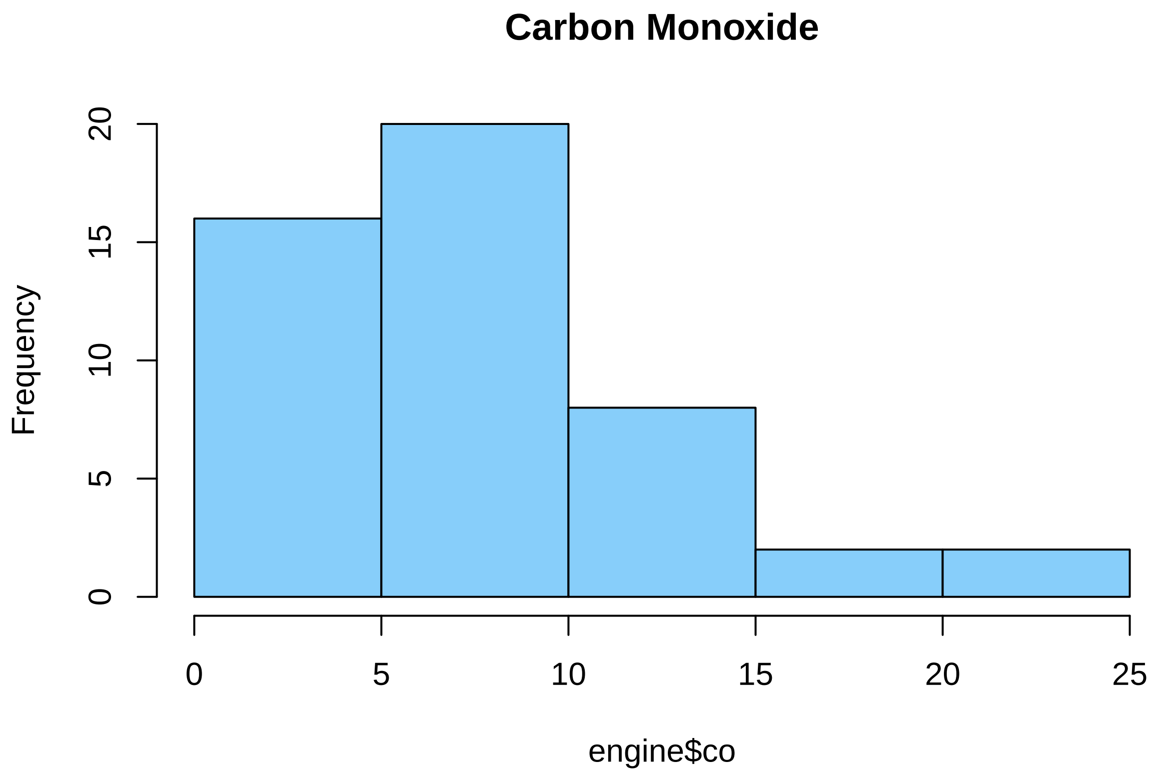
```
co_coeff = (10.015 - 5.905)/(5.905 - 4.388)
co_coeff
```

```
## [1] 2.709295
```

At first glance the carbon monoxide data appears to be skewed, because the spread between the third quantile and the max is five times the spread between the min and the first quantile.

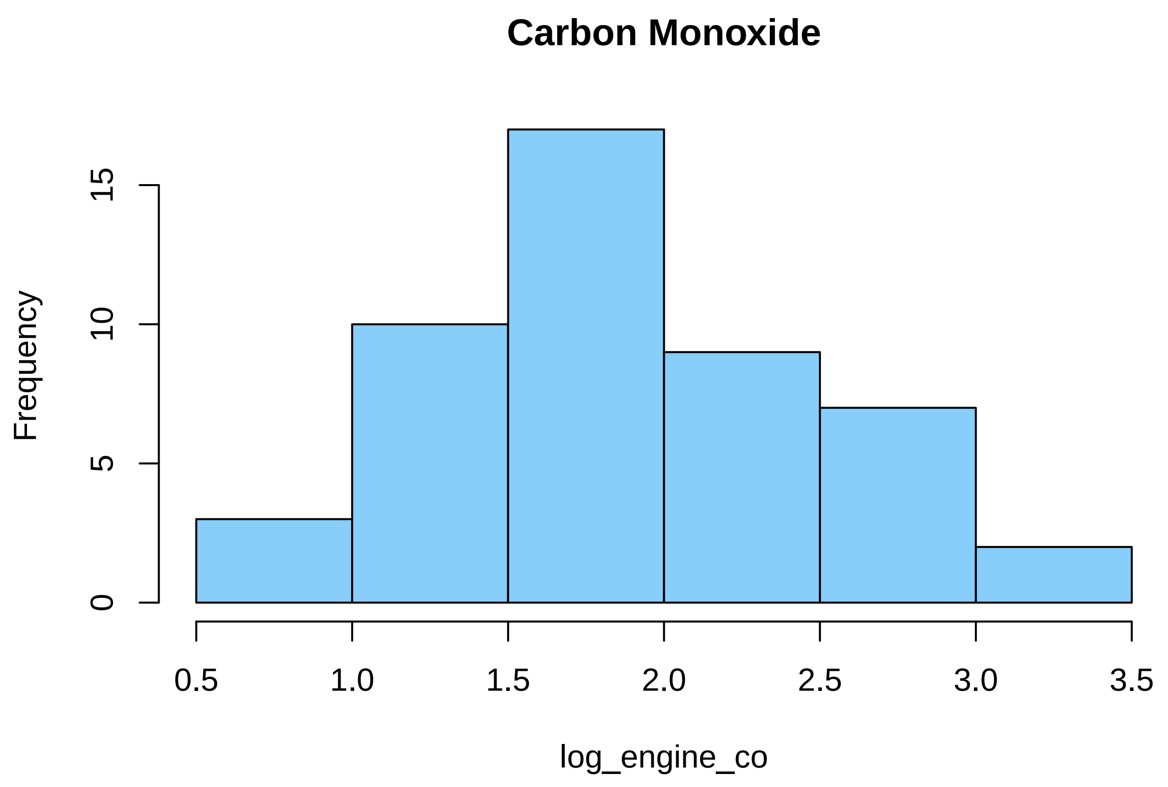
7) Plot the histogram for co .

```r
hist(engine$co, main = "Carbon Monoxide", col = "lightskyblue", breaks = 6)
```



8) Use log-transformation on "co" and plot histogram of transformed variable.

```r
log_engine_co = log(engine$co)
hist(log_engine_co, main = "Carbon Monoxide", col = "lightskyblue")
```



Example 2:

Import and GDP

9. Load data

```
trade = read.csv("~/Desktop/STAT 151A/STAT-151A/lab/lab2/imports.csv")
```

10. See summary data.

```
summary(trade)
```

```
##       Country      IMPORTS              GDP         
##  Argentina: 1   Min.   :   0.17   Min.   :    0.618  
##  Australia: 1   1st Qu.:   4.80   1st Qu.:   21.400  
##  Bolivia  : 1   Median :  31.80   Median :  200.000  
##  Brazil   : 1   Mean   : 124.82   Mean   :  969.993  
##  Canada   : 1   3rd Qu.: 164.00   3rd Qu.:  923.000  
##  Cuba     : 1   Max.   :1148.00   Max.   :10082.000  
##  (Other)  :19
```

11. Calculate (q3-q2)/(q2-q1) for IMPORT and GDP

```
import_coeff = (164 - 31.8)/(31.8-4.8)
import_coeff
```

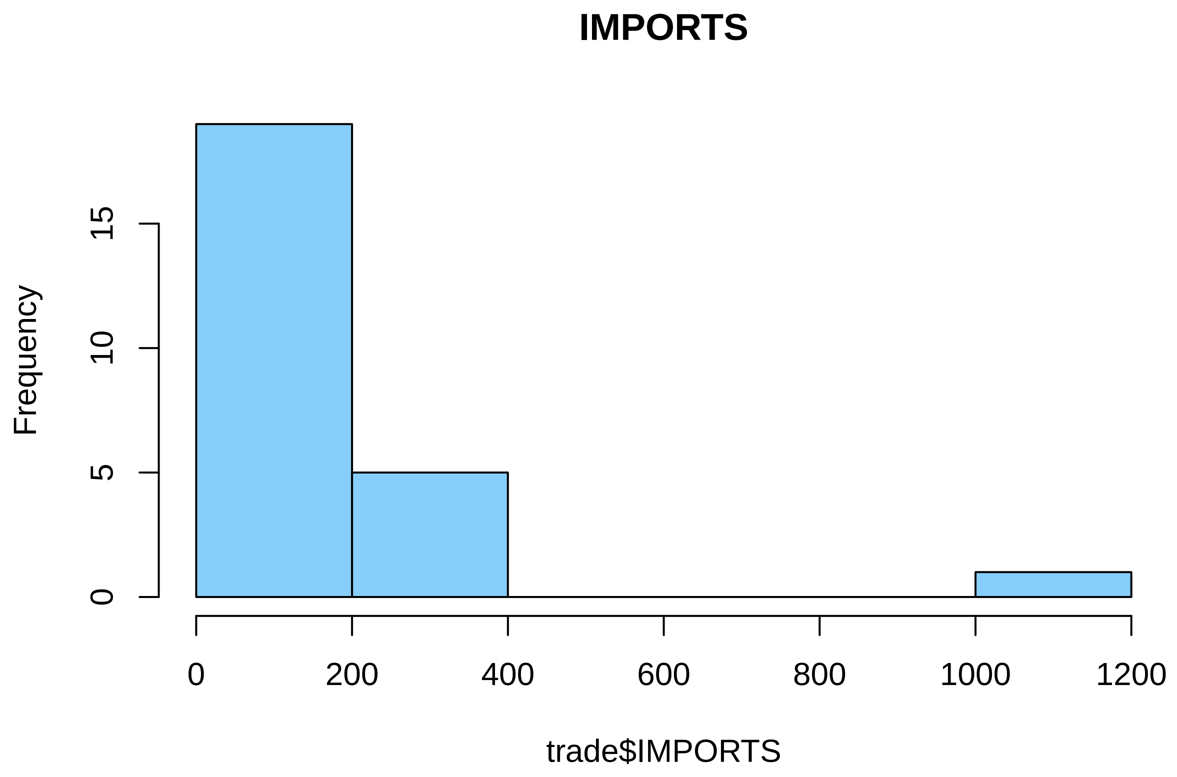
```
## [1] 4.896296
```

```
GDP_coeff = (923 - 200)/(200-21.4)
GDP_coeff
```

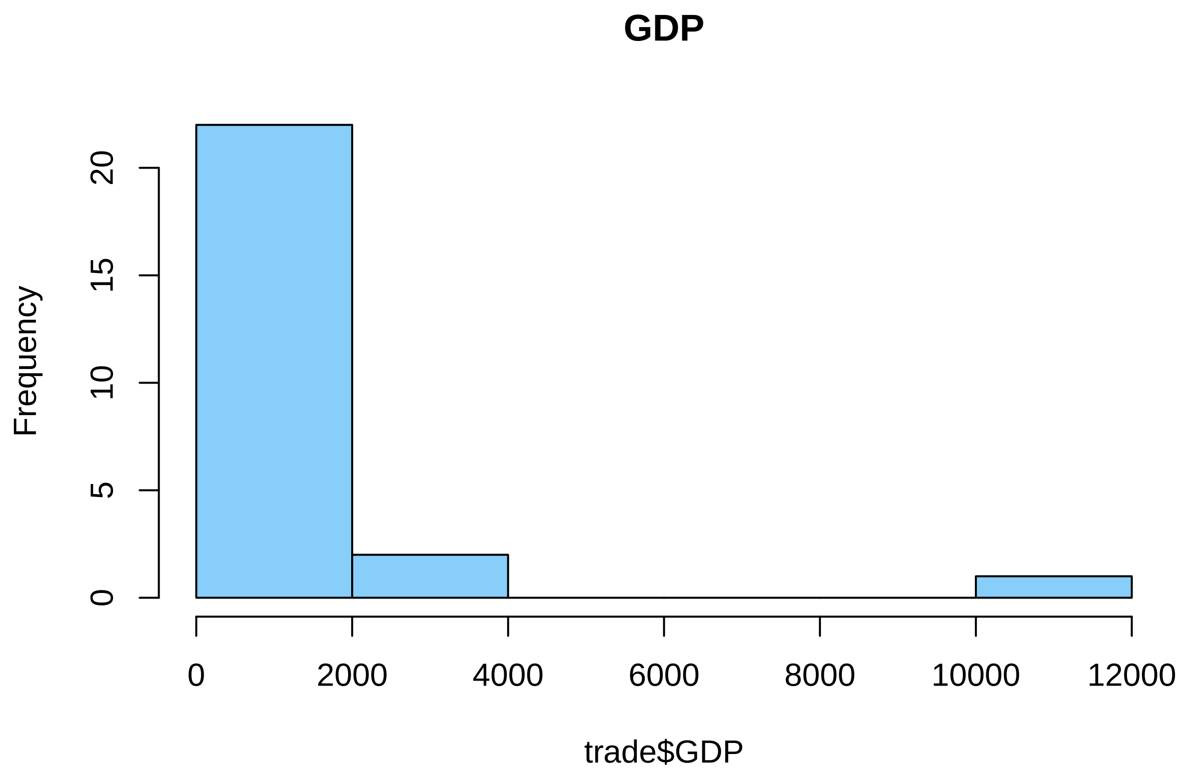
```
## [1] 4.048152
```

12. Lets look at the histograms .

```
hist(trade$IMPORTS, main="IMPORTS", col = "lightskyblue")
```



```
hist(trade$GDP, main="GDP", col = "lightskyblue")
```



13. What do we see in these histograms ?

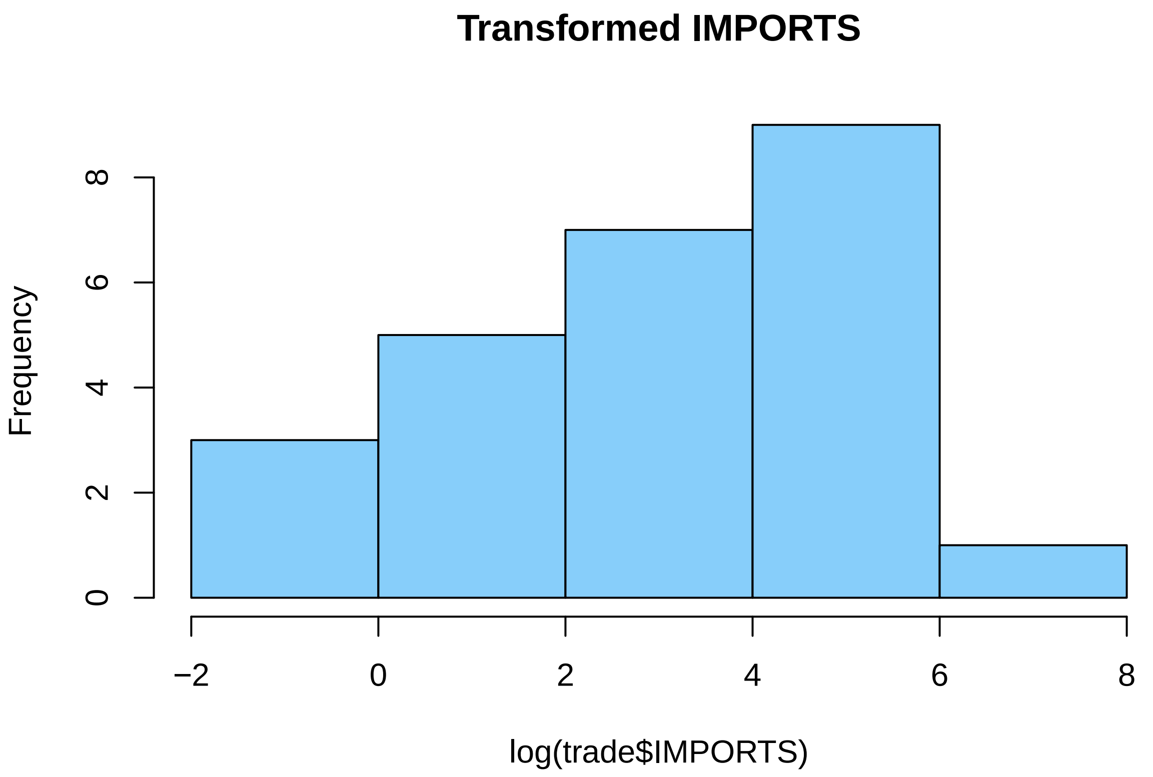
They are both highly right skewed distribution.

14. What transform to use – Any suggestions ?

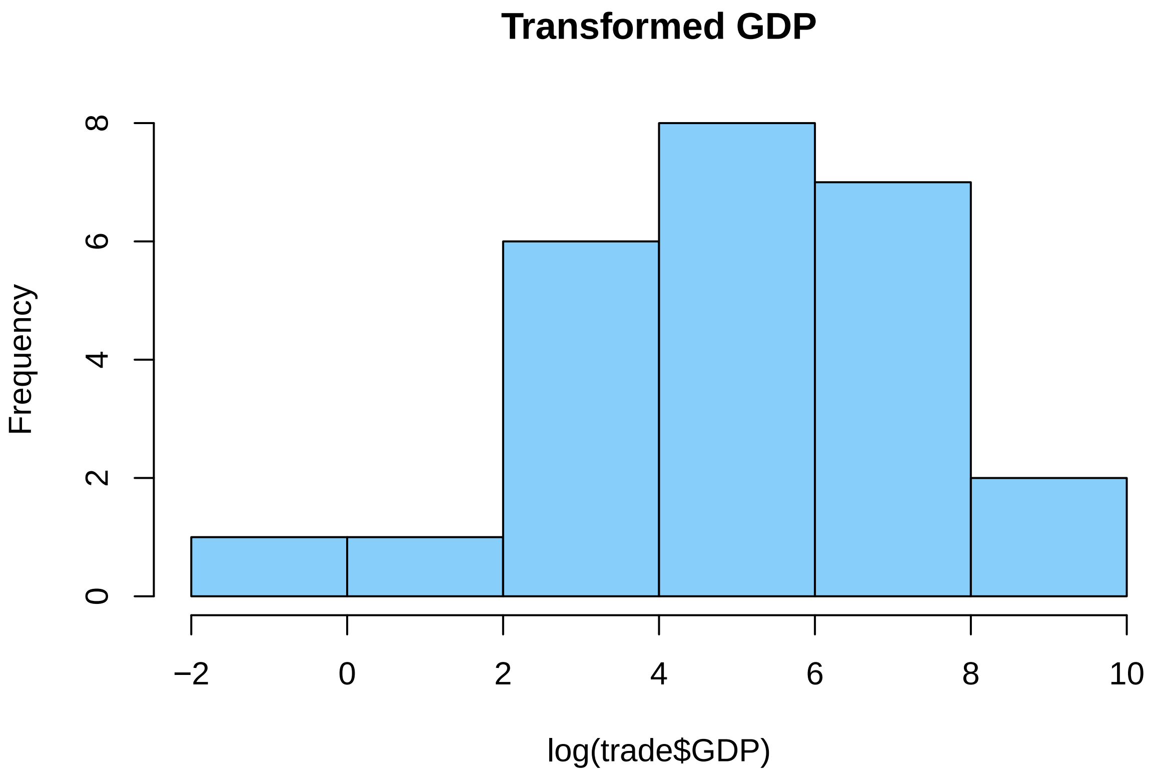
Since they are highly right skewed distribution, we can use logrithm transformation.

15. Plot the histogram for the transformed data ? How does it look ?

```r
hist(log(trade$IMPORTS), main="Transformed IMPORTS", col = "lightskyblue")
```



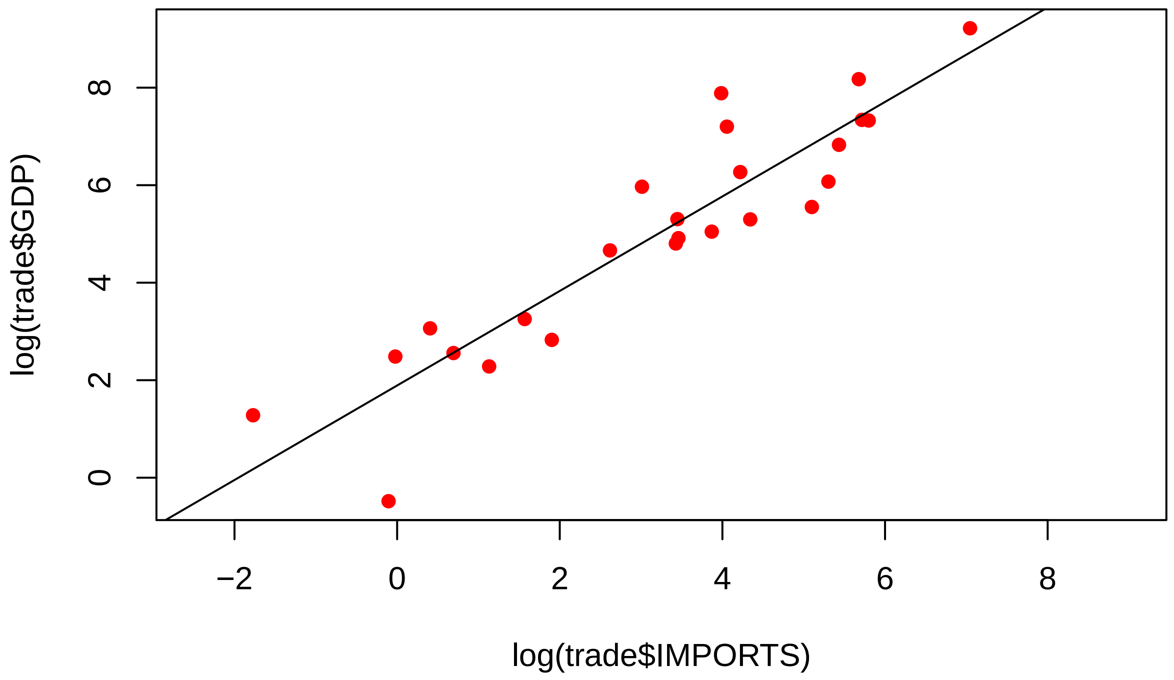
```r
hist(log(trade$GDP), main="Transformed GDP", col = "lightskyblue")
```



After transformation, both data look less right skewed and become more normally distributed.

16. Plot Import vs GDP (after transformation).

```
plot(log(trade$IMPORTS), log(trade$GDP), col = "red", xlim = c(-2.5, 9), pch = 16)
model = lm( log(trade$GDP) ~ log(trade$IMPORTS))
abline(model)
```


log(trade$GDP)
log(trade$IMPORTS)