A Brief Introduction to R*

Background and reference: *An Introduction to R* by Venables, Smith and others

```
> 1+1
[1] 2
> 2^3 # Two to the power 3
[1] 8

> 1:30
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25
[26] 26 27 28 29 30

> gamma(.5)^2      # Gamma(1/2) = Sqrt(Pi)
[1] 3.141593

> x = 1           # Assigns the value 1 to x
> y = 2
> x+y
[1] 3
> z = x+y
> z
[1] 3
> x = c(1,2,3,4,5,6)    #  Collect these numbers; x is now a vector

> z # No dynamic updating; it's not a spreadsheet
[1] 3
> x+y
[1] 3 4 5 6 7 8

> y = 1 + 2*x
> cbind(x,y)
     x  y
[1,] 1  3
[2,] 2  5
[3,] 3  7
[4,] 4  9
[5,] 5 11
[6,] 6 13

> z = y[x>4]           # z gets y such that x > 4
> z
[1] 11 13

> # If you put an array of integers inside the brackets, you get those
> # elements, in the order indicated.

> y[c(6,5,4,3,2,1)] # y in opposite order
[1] 13 11  9  7  5  3
> y[c(2,2,2,3,4)] # Repeats are okay
[1] 5 5 5 7 9
> y[7] # There is no seventh element.  NA is the missing value code.
[1] NA
```

---

* This document is free and open source. See last page for copyright information.

```
> # Computing probabilities, etc.
>
> pnorm(0) # Area less than zero for a standard normal
[1] 0.5
>
> pnorm(160,mean=100,sd=15)  # IQ of 160
[1] 0.9999683
>
> pcauchy(4)
[1] 0.9220209
>
> dnorm(0) # height of the curve
[1] 0.3989423
>
> dpois(0,lambda=3) # P(Y=0) for Y ~ Poisson(3)
[1] 0.04978707
>
> qnorm(0.975) # z value with P(Z<z) = 0.975
[1] 1.959964
>
> qf(0.975,df1=6,df2=122) # Critical value for F, not in any table
[1] 2.513606
>
> CriticalValue = qchisq(0.95,df=1:8)
> df=1:8; cbind(df,CriticalValue)
     df CriticalValue
[1,]  1      3.841459
[2,]  2      5.991465
[3,]  3      7.814728
[4,]  4      9.487729
[5,]  5     11.070498
[6,]  6     12.591587
[7,]  7     14.067140
[8,]  8     15.507313
```

```
> # Random number generation
> # Maybe transforming a uniform by inverse CDF
> help(Exponential) # Could also use  help(rexp)
```

Exponential {stats} R Documentation

The Exponential Distribution

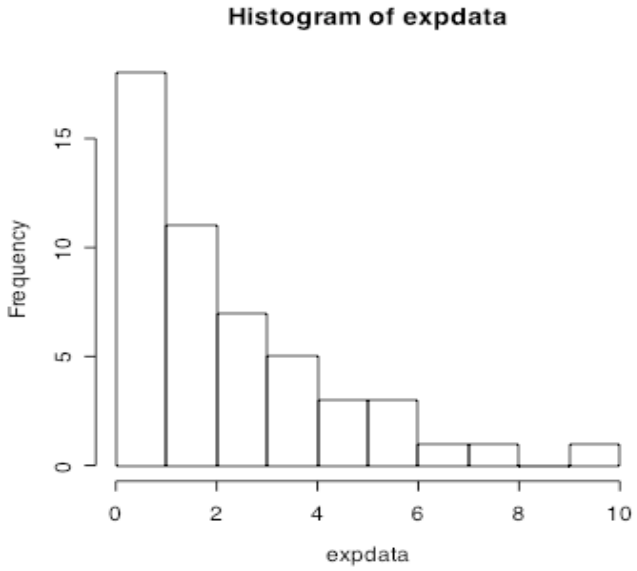
Description

Density, distribution function, quantile function and random generation for the exponential distribution with rate `rate` (i.e., mean `1/rate`).

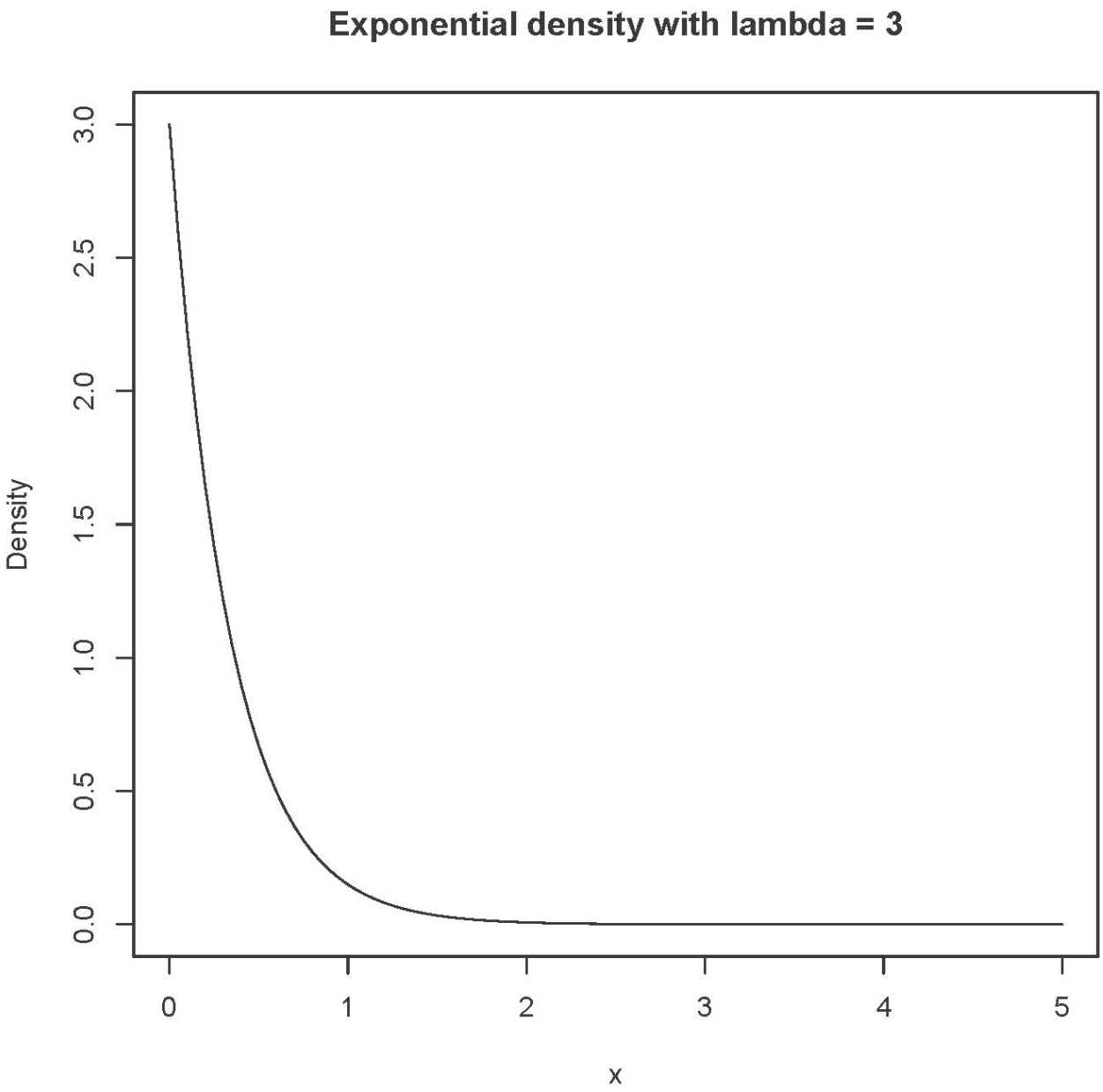
Usage

```
dexp(x, rate = 1, log = FALSE)
pexp(q, rate = 1, lower.tail = TRUE, log.p = FALSE)
qexp(p, rate = 1, lower.tail = TRUE, log.p = FALSE)
rexp(n, rate = 1)
```

```
> expdata = rexp(50,rate=1/2) # Random sample from exponential distribution, mean=2
> expdata
 [1] 0.4330015 5.7893762 0.9803759 0.7172530 2.2696433 4.0045302 3.3989651 0.3104736
 [9] 1.2026790 0.8543951 1.0438012 5.5095891 0.7587579 1.9263300 6.0660176 9.3017992
[17] 1.0910204 0.6551285 1.5747176 5.9417700 0.8464761 7.6684436 0.1107589 1.6787699
[25] 2.4744338 3.3470232 0.3209082 4.4307811 4.5510434 1.4316870 0.3457547 0.1302476
[33] 0.5777305 1.0898631 1.4467458 3.2472808 1.8113195 0.5090032 2.4633656 0.8972205
[41] 0.7562905 2.4623634 0.3413955 2.3122374 0.4166320 2.6279765 1.5072294 3.5732947
[49] 3.5449348 2.6472542
> mean(expdata)
[1] 2.267962
> hist(expdata)
```



```
> expfun = function(x) dexp(x,rate=3) # Exponential density with lambda=3
> curve(expfun, from=0, to=5, xlab='x', ylab='Density')
> titlestring = expression(paste('Exponential Density with Parameter ',lambda,' = 3'))
> title('Exponential density with lambda = 3')
```



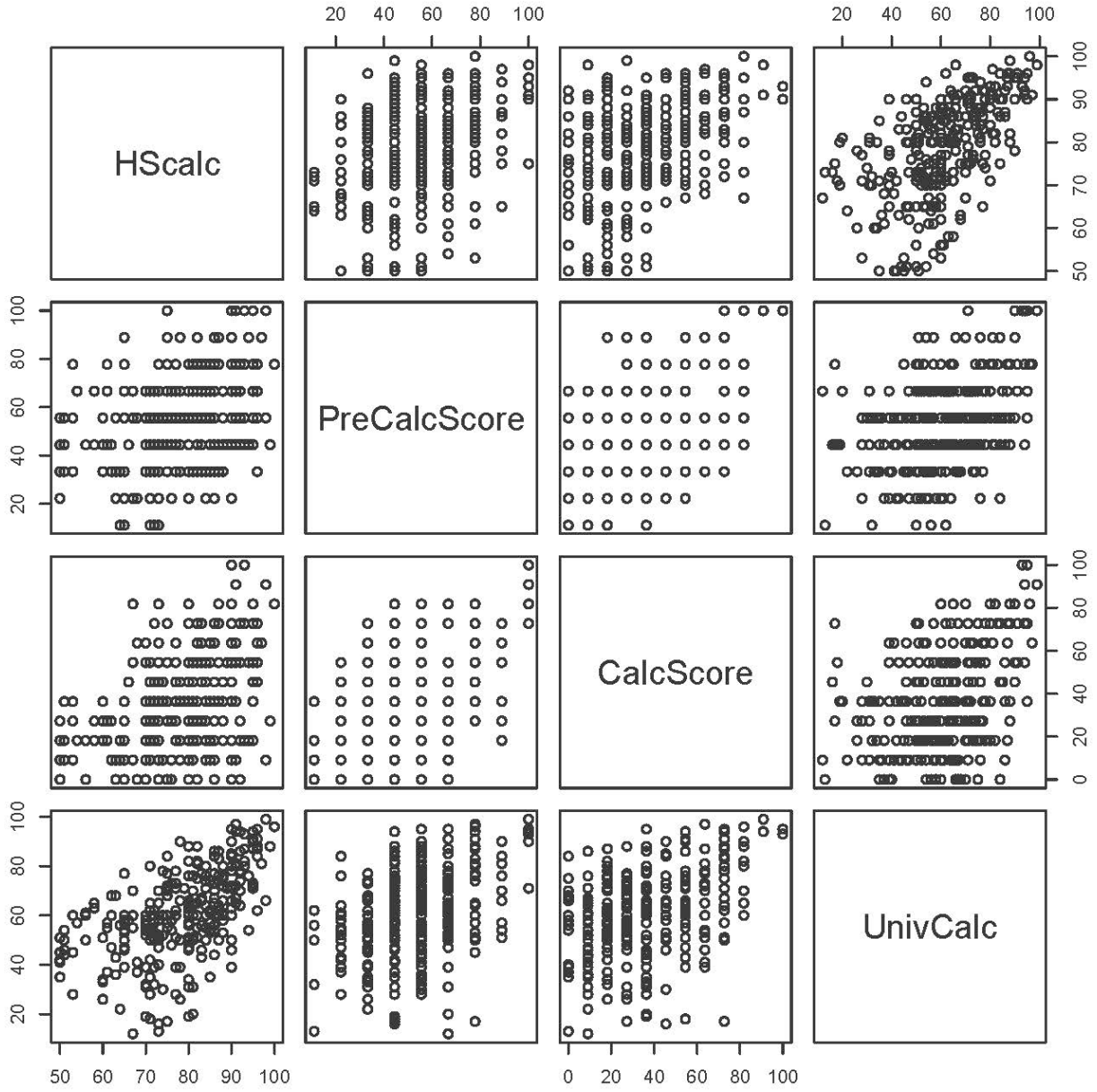
Tests and Confidence Intervals

Before the beginning of the Fall term, students in a first-year Calculus class took a diagnostic test with two parts: Pre-calculus and Calculus. Their High School Calculus marks and their marks in University Calculus were also available. In order, the variables in the data file are: Identification code, Mark in High School Calculus, Score on the Pre-calculus portion of the diagnostic test, Score on the Calculus portion of the diagnostic test, and mark in University Calculus. Thanks to Dr. Cleo Boyd for permission to use these data.

```
  1        65        2        0       39
  2        54        6        2       57
  3        77        4        4       62
  4        80        5        2       76
  5        87        4        4       86
  6        53        3        1       60

  .         .        .        .        .
  .         .        .        .        .
  .         .        .        .        .

290        83        4        3       56
291        81        6        3       70
292        73        5        9       60
293        80        5        2       50
294        56        4        2       50
295        80        6        1       61
```

```
> math = read.table("http://www.utstat.toronto.edu/brunner/data/legal/mathtest.txt")
> head(math)
  V1 V2 V3 V4 V5
1  1 65  2  0 39
2  2 54  6  2 57
3  3 77  4  4 62
4  4 80  5  2 76
5  5 87  4  4 86
6  6 53  3  1 60
> colnames(math) = c("ID","HScalc","PreCalcScore","CalcScore","UnivCalc")
> head(math)
  ID HScalc PreCalcScore CalcScore UnivCalc
1  1     65            2         0       39
2  2     54            6         2       57
3  3     77            4         4       62
4  4     80            5         2       76
5  5     87            4         4       86
6  6     53            3         1       60
> summary(math)
       ID             HScalc        PreCalcScore     CalcScore        UnivCalc
 Min.   :  1.0   Min.   : 50.00   Min.   :1.000   Min.   : 0.000   Min.   :12.00
 1st Qu.: 74.5   1st Qu.: 71.50   1st Qu.:4.000   1st Qu.: 2.000   1st Qu.:51.00
 Median :148.0   Median : 80.00   Median :5.000   Median : 4.000   Median :60.00
 Mean   :148.0   Mean   : 78.51   Mean   :4.837   Mean   : 3.963   Mean   :60.91
 3rd Qu.:221.5   3rd Qu.: 87.00   3rd Qu.:6.000   3rd Qu.: 6.000   3rd Qu.:73.00
 Max.   :295.0   Max.   :100.00   Max.   :9.000   Max.   :11.000   Max.   :99.00
> attach(math) # Make variable names available
> # PreCalc score is out of 9 and Calc score is out of 11. Convert to percentages.
> PreCalcScore = 100 * PreCalcScore/9
> CalcScore = 100 * CalcScore/11
```

```
> cor(datamat); pairs(datamat)

                HScalc PreCalcScore CalcScore  UnivCalc
HScalc       1.0000000    0.3271197 0.4132186 0.5548571
PreCalcScore 0.3271197    1.0000000 0.4573164 0.3736003
CalcScore    0.4132186    0.4573164 1.0000000 0.3793398
UnivCalc     0.5548571    0.3736003 0.3793398 1.0000000
```



```
> ###########  Fit the full regression model ##########
> fullmodel = lm(UnivCalc ~ HScalc+PreCalcScore+CalcScore)
> sumfull = summary(fullmodel); sumfull

Call:
lm(formula = UnivCalc ~ HScalc + PreCalcScore + CalcScore)

Residuals:
    Min      1Q  Median      3Q     Max
-48.699  -7.954   1.603   9.242  30.260

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  -6.32155    6.01019  -1.052  0.29376
HScalc        0.70097    0.08133   8.619  4.4e-16 ***
PreCalcScore  0.16849    0.05182   3.252  0.00128 **
CalcScore     0.08722    0.04282   2.037  0.04257 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 14.16 on 291 degrees of freedom
Multiple R-squared:  0.3583,  Adjusted R-squared:  0.3517
F-statistic: 54.17 on 3 and 291 DF,  p-value: < 2.2e-16

> # Confidence interval: betahat1 +or- tcrit*se
> fullmodel$coefficients
 (Intercept)       HScalc PreCalcScore    CalcScore
  -6.3215544    0.7009720    1.8720834    0.7928857
> betahat1 = fullmodel$coefficients[2]; betahat1
  HScalc
0.700972
> tcrit = qt(0.975,291)
> sumfull$coefficients
              Estimate Std. Error   t value     Pr(>|t|)
(Intercept)  -6.3215544 6.01018690 -1.051807 2.937609e-01
HScalc        0.7009720 0.08133028  8.618832 4.400339e-16
PreCalcScore  1.8720834 0.57572372  3.251704 1.282019e-03
CalcScore     0.7928857 0.38927156  2.036845 4.257051e-02
> se = sumfull$coefficients[2,2]; se
[1] 0.08133028
> lower95 = betahat1 - tcrit*se; upper95 = betahat1 + tcrit*se
> c(lower95,upper95)
   HScalc    HScalc
0.5409019 0.8610422

> # Now 2 ways to test PreCalcScore and CalcScore simultaneously
> # H0: beta2 = beta3 = 0
> # First the Full versus restricted model approach (extra sums of squares)
> redmodel = lm(UnivCalc ~ HScalc) # Reduced model, without terms being tested
> anova(redmodel,fullmodel)
Analysis of Variance Table

Model 1: UnivCalc ~ HScalc
Model 2: UnivCalc ~ HScalc + PreCalcScore + CalcScore
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    293 62967
2    291 58375  2    4591.5 11.444 1.643e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

```
> # The car (Companion to Applied Regression) packages has the
> # linearHypothesis function for testing H0: L beta = 0
> # install.packages("car",dependencies=TRUE) # Only need to do this once
> library(car) # Load the package
Loading required package: carData
> help(linearHypothesis)

> # Now test H0: L beta = 0
> # First compose the L matrix
> L = rbind(c(0,0,1,0),
+           c(0,0,0,1) )

> linearHypothesis(fullmodel,L,test="F")

Linear hypothesis test

Hypothesis:
PreCalcScore = 0
CalcScore = 0

Model 1: restricted model
Model 2: UnivCalc ~ HScalc + PreCalcScore + CalcScore

  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    293 62967
2    291 58375  2    4591.5 11.444 1.643e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

> # For comparison,
> anova(redmodel,fullmodel)
Analysis of Variance Table

Model 1: UnivCalc ~ HScalc
Model 2: UnivCalc ~ HScalc + PreCalcScore + CalcScore
  Res.Df   RSS Df Sum of Sq      F    Pr(>F)
1    293 62967
2    291 58375  2    4591.5 11.444 1.643e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```
