Introduction to S

```
/dos/brunner > R

R : Copyright 2002, The R Development Core Team
Version 1.5.0  (2002-04-29)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type `license()' or `licence()' for distribution details.

R is a collaborative project with many contributors.
Type `contributors()' for more information.

Type `demo()' for some demos, `help()' for on-line help, or
`help.start()' for a HTML browser interface to help.
Type `q()' to quit R.

> 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
> 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
```

```
> rmat <- rbind(x,y) ; rmat
  [,1] [,2] [,3] [,4] [,5] [,6]
x    1    2    3    4    5    6
y    3    5    7    9   11   13
>
>
>
> z <- y[x>4]           # z gets y such that x > 4
> z
[1] 11 13
>
> 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
>
> z <- x/y       # Most operations are performed element by element
> cbind(x,y,z)
     x  y         z
[1,] 1  3 0.3333333
[2,] 2  5 0.4000000
[3,] 3  7 0.4285714
[4,] 4  9 0.4444444
[5,] 5 11 0.4545455
[6,] 6 13 0.4615385
> x <- seq(from=0,to=3,by=.1)   # A sequence of numbers
> y <- sqrt(x)
>
> # Plotting
>
> pdf("testor.pdf")
> plot(x,y,type='l')      # That's a lower case L
> q()
Save workspace image? [y/n/c]: y
/dos/brunner > ls
lesson2.R    mcars.dat    testor.pdf
/dos/brunner > ls -a
./           .RData       lesson2.R    testor.pdf
../          .Rhistory    mcars.dat
```

```
/dos/brunner > R

R : Copyright 2002, The R Development Core Team
Version 1.5.0  (2002-04-29)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type `license()' or `licence()' for distribution details.

R is a collaborative project with many contributors.
Type `contributors()' for more information.

Type `demo()' for some demos, `help()' for on-line help, or
`help.start()' for a HTML browser interface to help.
Type `q()' to quit R.

[Previously saved workspace restored]

> ls()
[1] "rmat" "x"    "y"    "z"
> max(x)
[1] 3
> q()
```

Random numbers and simulation

```
> rnorm(20) # 20 standard normals
 [1]  0.24570675 -0.38857202  0.47642336  0.75657595  0.71355871 -0.74630629
 [7] -0.02485569  1.93346357  0.15663167  1.16734485  0.57486449  1.32309413
[13]  0.63712982  2.00473940  0.04221730  0.70896768  0.42128470 -0.12115292
[19]  1.42043470 -1.04957255

> set.seed(12345) # Be able to reproduce the stream of pseudo-random numbers.
> rnorm(20)
 [1]  0.77795979 -0.89072813  0.05552657  0.67813726  0.80453336 -0.35613672
 [7] -1.24182991 -1.05995791 -2.67914037 -0.01247257 -1.22422266  0.88672878
[13] -1.32824804 -2.73543539  0.40487757  0.41793236 -1.47520817  1.15351981
[19] -1.24888614  1.11605686
> rnorm(20)
 [1]  0.866507371  2.369884323  0.393094088 -0.970983967 -0.292948278
 [6]  0.867358962  0.495983546  0.331635970  0.702292771  2.514734599
[11]  0.522917841 -0.194668990 -0.089222053 -0.491125596 -0.452112445
[16] -0.515548826 -0.244409517 -0.008373764 -1.459415684 -1.433710170

> set.seed(12345)
> rnorm(20)
 [1]  0.77795979 -0.89072813  0.05552657  0.67813726  0.80453336 -0.35613672
 [7] -1.24182991 -1.05995791 -2.67914037 -0.01247257 -1.22422266  0.88672878
[13] -1.32824804 -2.73543539  0.40487757  0.41793236 -1.47520817  1.15351981
[19] -1.24888614  1.11605686
> help(runif)
```

(Output deleted)

```
>
> # Illustrating the Regression Artifact by Simulation
>
> system("cat regart.R")
# regart.R    Demonstrate Regression Artifact
###################### Setup ######################
N <- 10000 ; n <- 100
truevar <- 180 ; errvar <- 45
truesd <- sqrt(truevar) ; errsd <- sqrt(errvar)
# set.seed(44444)
# Now define the function ttest, which does a matched t-test

ttest <- function(d) # Matched t-test. It operates on differences.
    {
    ttest <- numeric(4)
    names(ttest) <- c("Mean Difference"," t "," df "," p-value ")
    ave <- mean(d) ; nn <- length(d) ; sd <- sqrt(var(d)) ; df <- nn-1
    tstat <- ave*sqrt(nn)/sd
    pval <- 2*(1-pt(abs(tstat),df))
    ttest[1] <- ave ; ttest[2] <- tstat; ttest[3] <- df; ttest[4] <- pval
    ttest # Return the value of the function
    }

#########################################################

error1 <- rnorm(N,0,errsd) ; error2 <- rnorm(N,0,errsd)
truescor <- rnorm(N,100,truesd)
pre <- truescor+error1 ; rankpre <- rank(pre)

# Based on their rank on the pre-test, we take the n worst students and
# place them in a special remedial program, but it does NOTHING.

# Based on their rank on the pre-test, we take the n best students and
# place them in a special program for the gifted, but it does NOTHING.

post <- truescor+error2
diff <- post-pre # Diff represents "improvement."
                 # But of course diff = error2-error1 = noise

cat("\n") # Skip a line
cat("---------------------------------- \n")
dtest <- ttest(diff)
cat("Test on diff (all scores) \n") ; print(dtest) ; cat("\n")

remedial <- diff[rankpre<=n] ; rtest <- ttest(remedial)
cat("Test on Remedial \n") ; print(rtest) ; cat("\n")

gifted <- diff[rankpre>=(N-n+1)] ; gtest <- ttest(gifted)
cat("Test on Gifted \n") ; print(gtest) ; cat("\n")
cat("---------------------------------- \n")
```

```
>
> source("regart.R")

------------------------------------
Test on diff (all scores)
Mean Difference              t              df         p-value
    -0.07566216     -0.80423591   9999.00000000      0.42127986

Test on Remedial
Mean Difference              t              df         p-value
   6.705759e+00    7.420290e+00    9.900000e+01    4.094725e-11

Test on Gifted
Mean Difference              t              df         p-value
  -5.117714e+00   -5.456844e+00    9.900000e+01    3.586174e-07

------------------------------------
> source("regart.R")

------------------------------------
Test on diff (all scores)
Mean Difference              t              df         p-value
    -0.05532373     -0.58573175   9999.00000000      0.55806902

Test on Remedial
Mean Difference              t              df         p-value
   7.412656e+00    7.993408e+00    9.900000e+01    2.479572e-12

Test on Gifted
Mean Difference              t              df         p-value
  -7.680115e+00   -8.265243e+00    9.900000e+01    6.465939e-13

------------------------------------
> source("regart.R")

------------------------------------
Test on diff (all scores)
Mean Difference              t              df         p-value
    -0.18641863     -2.01135103   9999.00000000      0.04431518

Test on Remedial
Mean Difference              t              df         p-value
   8.598978e+00    9.018827e+00    9.900000e+01    1.509903e-14

Test on Gifted
Mean Difference              t              df         p-value
  -8.104179e+00   -8.594758e+00    9.900000e+01    1.256772e-13

------------------------------------
```

Regression with S

```
/dos/brunner >
/dos/brunner > ls
lesson2.R    mcars.dat    testor.pdf
/dos/brunner > R --vanilla < lesson2.R > lesson2.out
/dos/brunner > ls
lesson2.R     lesson2.out   mcars.dat     testor.pdf
```

Or, for long-running jobs, do this:

```
nohup nice R --vanilla < commands.R > homework2.out &

/dos/brunner > less lesson2.out

> ######################################################################
> # lesson2.R: execute with    R --vanilla < lesson2.R > lesson2.out #
> ######################################################################
>
> datalist <-  scan("mcars.dat",list(id=0,country=0,kpl=0,weight=0,length=0))
Read 100 records
> # datalist is a linked list.
> datalist
$id
  [1]   1   2   3   4   5   6   7   8   9  10  11  12  13  14  15  16  17  18
 [19]  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36
 [37]  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54
 [55]  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72
 [73]  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90
 [91]  91  92  93  94  95  96  97  98  99 100

$country
  [1] 1 2 1 1 1 1 3 1 3 1 2 1 1 3 2 1 1 1 3 2 1 1 1 3 2 1 1 1 1 1 2 1 2 1 1 1 3
 [38] 3 1 1 1 1 1 1 1 1 1 1 3 1 1 1 1 1 1 1 1 3 1 1 1 2 1 1 1 1 2 2 1 1 1 2 1 1
 [75] 1 2 2 3 1 1 1 1 3 1 1 1 1 1 1 1 1 1 1 3 3 3 1 1 1

$kpl
  [1]  5.04 10.08  9.24  7.98  7.98  7.98  9.66  7.56  5.88 10.92 12.60  8.40
 [13]  8.82 10.92  7.56 12.18  5.04  5.88  7.14 13.02  5.88 10.92  6.72 10.50
 [25]  8.82  5.88  6.72 11.76  9.24  7.56  7.56 11.76 10.50  5.88  9.24  7.98
 [37]  7.14 17.22  6.72  7.98  7.14  6.30  5.88  8.82  9.24  9.24  5.88  8.40
 [49] 10.50  9.24  7.56  7.56 12.60 12.60  7.98  7.56  8.40  9.66  7.56  6.30
 [61]  5.88  7.56 10.08  5.04  8.82 11.76 14.70 10.08  9.24 10.92 10.50  7.56
 [73]  8.82  7.56  7.14  7.56 10.08  8.82  5.88  8.82  8.82 10.08 17.22  6.72
 [85]  9.24  5.88  7.56 11.76  7.98  8.82  5.88  5.88  7.14  5.04 17.22 17.22
 [97]  7.14 10.50  6.72  7.56

$weight
  [1] 2178.0 1026.0 1188.0 1444.5 1485.0 1485.0  972.0 1665.0 1539.0 1003.5
 [11]  891.0 1273.5 1930.5  823.5 1084.5  949.5 2178.0 1755.0 1426.5  990.0
 [21] 1827.0 1134.0 1813.5 1192.5 1237.5 1858.5 1813.5 1062.0 1431.0 1651.5
 [31] 1201.5 1062.0 1008.0 1858.5 1318.5 1440.0 1273.5  918.0 1813.5 1530.0
 [41] 1683.0 1836.0 1723.5 1827.0 1449.0 1318.5 1858.5 1273.5  868.5 1318.5
```

```
 [51] 1665.0 1620.0  954.0  954.0 1516.5 1665.0 1462.5  972.0 1665.0 1674.0
 [61] 1755.0 1201.5 1237.5 2178.0 1930.5 1062.0  922.5 1026.0 1449.0 1134.0
 [71]  990.0 1084.5 1930.5 1516.5 1507.5 1084.5 1026.0  958.5 1858.5 1930.5
 [81] 1192.5 1237.5  918.0 1813.5 1449.0 1755.0 1561.5 1062.0 1489.5 1192.5
 [91] 1827.0 1755.0 1683.0 2178.0  918.0  918.0 1426.5  990.0 1660.5 1498.5

$length
  [1] 591.82 431.80 426.72 510.54 502.92 502.92 436.88 543.56 487.68 431.80
 [11] 391.16 495.30 518.16 360.68 441.96 414.02 591.82 518.16 490.22 419.10
 [21] 561.34 462.28 523.24 449.58 467.36 551.18 523.24 431.80 490.22 553.72
 [31] 444.50 431.80 436.88 551.18 472.44 505.46 480.06 393.70 523.24 508.00
 [41] 558.80 563.88 510.54 558.80 508.00 472.44 551.18 495.30 393.70 472.44
 [51] 543.56 523.24 414.02 414.02 508.00 543.56 497.84 436.88 543.56 538.48
 [61] 518.16 444.50 454.66 591.82 518.16 431.80 416.56 431.80 508.00 462.28
 [71] 419.10 441.96 518.16 502.92 439.42 441.96 431.80 408.94 551.18 518.16
 [81] 454.66 454.66 393.70 523.24 508.00 518.16 502.92 431.80 502.92 454.66
 [91] 561.34 518.16 558.80 591.82 393.70 393.70 490.22 419.10 538.48 510.54

> # There are other ways to read raw data. See help(read.table).

> weight <- datalist$weight ; length <- datalist$length ; kpl <- datalist$kpl
> country <- datalist$country
> cor(cbind(weight,length,kpl))
           weight     length        kpl
weight  1.0000000  0.9462018 -0.7704194
length  0.9462018  1.0000000 -0.7899859
kpl    -0.7704194 -0.7899859  1.0000000

> # The table command gives a bare-bones frequency distribution
> table(country)
country
 1  2  3
73 13 14

> # That was a matrix. The numbers 1 2 3 are labels.
> # You can save it, and you can get at its contents
> countrytable <- table(country)
> countrytable[2]
 2
13

> # There is an "if" function that you could use to make dummy variables,
> # but it's easier to use factor.
> countryfac <- factor(country,levels=c(1,2,3),
+                      label=c("US","Japanese","European"))
> # This makes a FACTOR corresponding to country, like declaring it
> # to be categorical. How are dummy variables being set up?
```

```
> contrasts(countryfac)
         Japanese European
US              0        0
Japanese        1        0
European        0        1

> # The first level specified is the reference category. You can get a
> # different reference category by specifying the levels in a different order.
> cntryfac <- factor(country,levels=c(2,1,3),
+                       label=c("Japanese","US","European"))

> contrasts(cntryfac)
         US European
Japanese  0        0
US        1        0
European  0        1



> # Test interaction. For comparison, with SAS we got F = 11.5127, p < .0001

> # First fit (and save!) the reduced model. lm stands for linear model.
> redmod <- lm(kpl ~ weight+cntryfac)

> # The object redmod is a linked list, including lots of stuff like all the
> # residuals. You don't want to look at the whole thing, at least not now.

> summary(redmod)

Call:
lm(formula = kpl ~ weight + cntryfac)

Residuals:
    Min      1Q  Median      3Q     Max
-3.0759 -0.9810 -0.1919  0.4725  5.0795

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      16.2263357  0.7631228  21.263   <2e-16 ***
weight           -0.0060407  0.0005708 -10.583   <2e-16 ***
cntryfacUS        1.2361472  0.5741299   2.153   0.0338 *
cntryfacEuropean  1.4595914  0.6456563   2.261   0.0260 *
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

Residual standard error: 1.676 on 96 degrees of freedom
Multiple R-Squared: 0.618,      Adjusted R-squared: 0.606
F-statistic: 51.76 on 3 and 96 DF,  p-value:     0
```

```
>
> # Full model is same stuff plus interaction. You COULD specify the whole thing.

> fullmod <- update(redmod,. ~ . + weight*cntryfac)

> anova(redmod,fullmod)

Analysis of Variance Table

Model 1: kpl ~ weight + cntryfac
Model 2: kpl ~ weight + cntryfac + weight:cntryfac
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     96 269.678
2     94 216.617  2    53.061 11.513 3.372e-05 ***
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1

> # The ANOVA summary table is a matrix. You can get at its (i,j)th element.
> aovtab <- anova(redmod,fullmod)
> aovtab[2,5] # The F statistic
[1] 11.51273

> aovtab[2,6] < .05       #    p < .05 -- True or false?
[1] TRUE

> 1>6 # Another example of an expression taking the logical value true or false.
[1] FALSE
```

Factorial ANOVA

```
> system("cat potato2.dat")
    Bact Temp Rot
   1  1    1   7
   2  1    1   7
   3  1    1   9
   4  1    1   0
   5  1    1   0
   6  1    1   0
   7  1    1   9
   8  1    1   0
   9  1    1   0
  10  1    2  10
  11  1    2   6
  12  1    2  10
  13  1    2   4
  14  1    2  10
  15  1    2   5
  16  1    2   8
  17  1    2   0
  18  1    2  10
  19  2    1   2
  20  2    1   4
  21  2    1   9
  22  2    1   4
  23  2    1   5
  24  2    1  10
  25  2    1   4
  26  2    1   5
  27  2    1   0
  28  2    2  17
  29  2    2  18
  30  2    2   8
  31  2    2   3
  32  2    2  23
  33  2    2   7
  34  2    2  15
  35  2    2  14
  36  2    2  17
  37  3    1  13
  38  3    1  11
  39  3    1   3
  40  3    1  10
  41  3    1   4
  42  3    1   7
  43  3    1  15
  44  3    1   2
  45  3    1   7
  46  3    2  26
```

```
  47  3   2  19
  48  3   2  24
  49  3   2  15
  50  3   2  22
  51  3   2  18
  52  3   2  20
  53  3   2  24
  54  3   2   8
>
> spuds <- read.table("potato2.dat")
> spuds[1:10,] # Rows 1:10, all columns
   Bact Temp Rot
1     1    1   7
2     1    1   7
3     1    1   9
4     1    1   0
5     1    1   0
6     1    1   0
7     1    1   9
8     1    1   0
9     1    1   0
10    1    2  10
>
> # Table of means
> mtable <- mtable <- tapply(spuds$Rot, list(spuds$Temp,spuds$Bact),mean)
> mtable
         [,1]      [,2]     [,3]
[1,] 3.555556  4.777778  8.00000
[2,] 7.000000 13.555556 19.55556
>
> # Note that just specifying the interaction forces the main effects in
> summary(aov(Rot~factor(Bact)*factor(Temp),data=spuds))
                         Df  Sum Sq Mean Sq F value    Pr(>F)
factor(Bact)              2  651.81  325.91 14.8390 9.608e-06 ***
factor(Temp)              1  848.07  848.07 38.6138 1.180e-07 ***
factor(Bact):factor(Temp)  2  152.93   76.46  3.4815   0.03874 *
Residuals                 48 1054.22   21.96
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> #
> # For custom tests, make a combination variable
> #
> rot <- spuds$Rot ; temp <- spuds$Temp ; bact <- spuds$Bact
> combo <- 10*temp + bact # Two levels of temp x 3 levels of bact
> table(temp,combo)
    combo
temp 11 12 13 21 22 23
   1  9  9  9  0  0  0
   2  0  0  0  9  9  9
> table(bact,combo)
    combo
bact 11 12 13 21 22 23
```

```
   1  9  0  0  9  0  0
   2  0  9  0  0  9  0
   3  0  0  9  0  0  9
> #
> # Suppose we want to test diff among bact just for low temp=1
> # We can do it with a dummy var setup like the default contrasts.
> #
> n <- length(combo)
> d1 <- numeric(n) ; d2 <- d1; d3 <- d1; d4 <- d1; d5 <- d1
> d1[combo==12] <- 1 ; d2[combo==13] <- 1 ; d3[combo==21] <- 1
> d4[combo==22] <- 1 ; d5[combo==23] <- 1
> table(d1,combo)
   combo
d1  11 12 13 21 22 23
  0  9  0  9  9  9  9
  1  0  9  0  0  0  0
> table(d2,combo)
   combo
d2  11 12 13 21 22 23
  0  9  9  0  9  9  9
  1  0  0  9  0  0  0
> #
> redmod <- lm(rot~d3+d4+d5) ; fullmod <- lm(rot~d1+d2+d3+d4+d5)
> anova(redmod,fullmod)
Analysis of Variance Table

Model 1: rot ~ d3 + d4 + d5
Model 2: rot ~ d1 + d2 + d3 + d4 + d5
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     50 1149.11
2     48 1054.22  2     94.89 2.1602 0.1264
> #
> # Suppose the null hypothesis is no temp effect for any type of bacteria
> # H0:  mu11=mu21 , mu12=mu22  mu13=mu23
> #
> # Fit a reduced model by transforming the independent variables
> # Need one more dummy variable
> d0 <- numeric(n) ; d0[combo==11] <- 1
> v1 <- d0+d3 ; v2 <- d1+d4 ; v3 <- d2+d5
> #
> # I can't make lm fit a model with no intercept! Must use lsfit. Ugh!
> #
> redfit <- lsfit(cbind(v1,v2,v3),rot,intercept=F) # Not like an lm object
> SSEr <- sum(residuals(redfit)^2)
> #
> # Now we need some stuff from the full model
> #
```

```
> fullmat <- anova(fullmod) # ANOVA summary table is a matrix
> fullmat
Analysis of Variance Table

Response: rot
          Df  Sum Sq Mean Sq F value    Pr(>F)
d1         1  231.48  231.48 10.5396  0.002133 **
d2         1   61.25   61.25  2.7888  0.101436
d3         1  184.08  184.08  8.3815  0.005689 **
d4         1   24.00   24.00  1.0927  0.301099
d5         1 1152.00 1152.00 52.4519 3.128e-09 ***
Residuals 48 1054.22   21.96
---
Signif. codes:  0 `***' 0.001 `**' 0.01 `*' 0.05 `.' 0.1 ` ' 1
> SSEf <- fullmat[6,2] ; SSEf

1054.222
> MSEf <- fullmat[6,3] ; MSEf

21.96296
> F2 <- (SSEr-SSEf)/(3*MSEf) ; pval <- 1-pf(F2,3,48)
> F2 ; pval

15.19224

4.402339e-07
>
```

It's so much easier with SAS

```
/* spud1.sas */
options linesize=79 noovp formdlim='_';
title 'Rotten potato check';

data spud;
     infile 'potato2.dat' firstobs=2; /* Skip the first line that R uses */
     input id bact temp rot;
     if temp=1 and bact=1 then mu11=1; else mu11=0;
     if temp=1 and bact=2 then mu12=1; else mu12=0;
     if temp=1 and bact=3 then mu13=1; else mu13=0;
     if temp=2 and bact=1 then mu21=1; else mu21=0;
     if temp=2 and bact=2 then mu22=1; else mu22=0;
     if temp=2 and bact=3 then mu23=1; else mu23=0;
     combo = 10*temp+bact;

proc glm;
     title2 'Standard 2-way ANOVA with proc glm';
     class bact temp;
     model rot=temp|bact;

proc reg;
     title2 'Custom tests with proc reg';
     model rot = mu11--mu23 / noint;
     ex1: test mu11=mu12=mu13;
     ex2: test mu11=mu21, mu12=mu22, mu13=mu23;

proc glm;
     class combo;
     title2 'Specify contrast matrix directly with proc glm';
     model rot = combo;
     contrast 'Bact diff for low temp'
        combo 1 -1  0  0  0  0,
        combo 0  1 -1  0  0  0;
     contrast 'Temp diff in each bact'
        combo 1  0  0 -1  0  0,
        combo 0  1  0  0 -1  0,
        combo 0  0  1  0  0 -1;
```

Part of the last proc glm output

Contrast	DF	Contrast SS	Mean Square	F Value	Pr > F
Bact diff for low temp	2	94.888889	47.444444	2.16	0.1264
Temp diff in each bact	3	1001.000000	333.666667	15.19	<.0001