

ModellingTandF.R

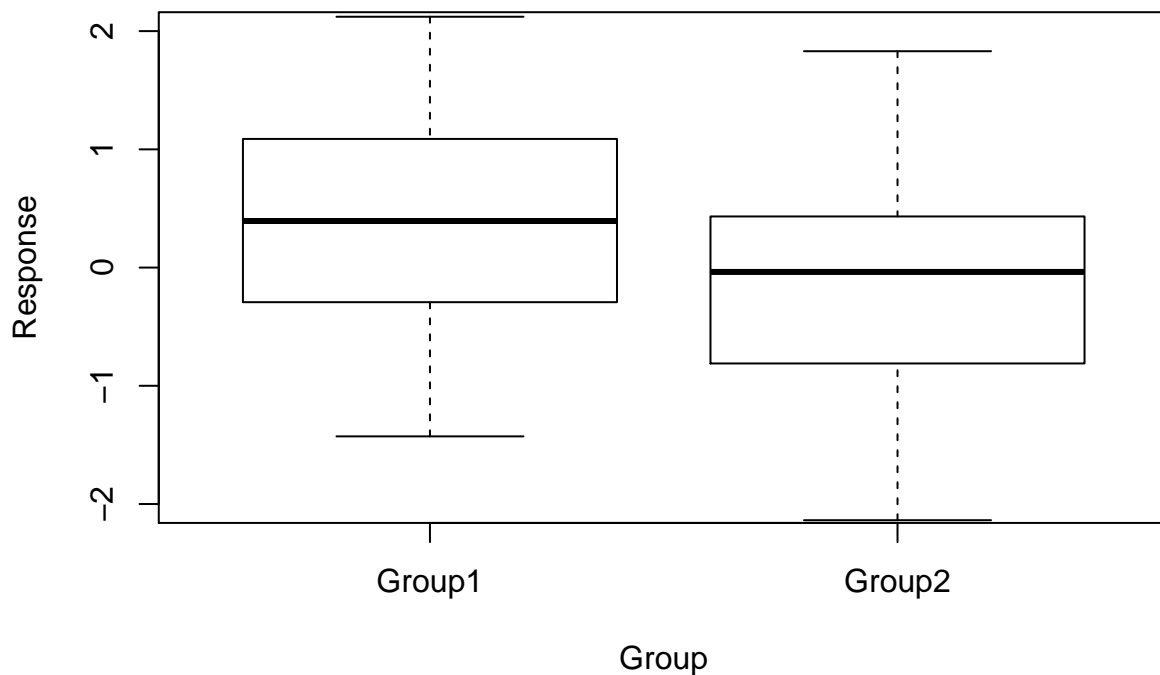
Glenn Tattersall

2019-03-25

```
# Use this script to demonstrate how to Knit output to share code and results with colleagues
# and to demonstrate the use of functions, randomised sampling, and empirically
# demonstrating how to arrive at the T and F distributions. Hopefully, this might
# demystify obscure statistics.

# Call Libraries
library(ggplot2)

# Random data can easily be generated in R
# Create two data sets, x and y that are both 30 samples, drawn from a normal distribution, mean = 0 and
set.seed(14) # try 14
n=30
x<-rnorm(n=30, mean=0, sd=1)
y<-rnorm(n, 0, 1)
z<-data.frame(Group=factor(c(rep("Group1", n), rep("Group2", n))), Response=c(x,y))
plot(Response ~ Group, z, ylim=c(-2,2))
```



```
t.test(Response ~ Group, data=z, var.equal=T)

##
## Two Sample t-test
##
## data: Response by Group
## t = 2.4084, df = 58, p-value = 0.01922
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
```

```
## 0.09754805 1.05781299
## sample estimates:
## mean in group Group1 mean in group Group2
## 0.4235508 -0.1541297

# Interesting result, we conclude that these are possibly significantly different from one another?!

# Usually we run our stats without knowing much about the theory. A t statistic is compared against a
# distribution that is a formula, see here for these formulae:
# https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/basic-statistics/probability-distribution

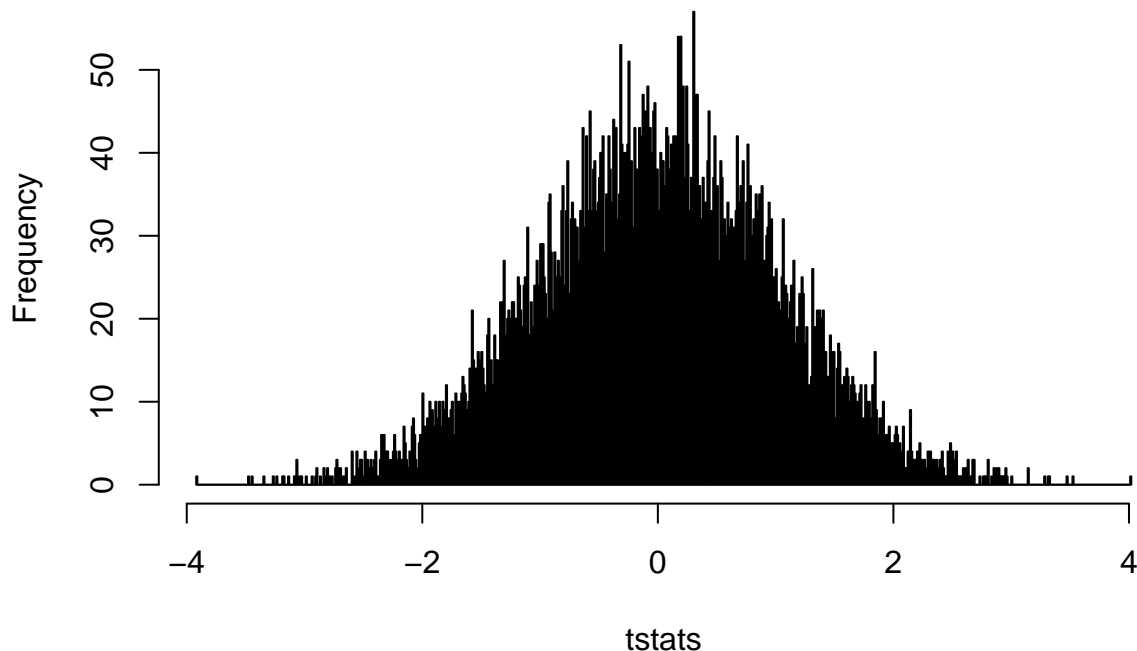
# But if we can generate random data, we should be able to generate an empirical set of data that exhibit
# distribution resulting from these formulae.

# Let's do this by using the replicate function, which performs a function we define, runs it as many times as we
# ask, and then collects the result. We'll replicate our tcalc() function 10,000 times.

# First define function to return the t statistic from a two sample t test for randomly generated data
# derived from normally distributed data
tcalc<-function(n=10){
  x<-rnorm(n, 0,1)
  y<-rnorm(n, 0,1)
  z<-data.frame(Group=factor(c(rep("Group1", n), rep("Group2", n))), Response=c(x,y))
  t<-(mean(x)-mean(y))/(sqrt((sd(x)^2)/n+(sd(y)^2)/n))
  # t<-t.test(Response ~ Group, data=z, var.equal=T)$statistic
  return(t)
}

tstats<-replicate(10000, tcalc(n=1000))
hist(tstats, breaks=1000)
```

Histogram of tstats

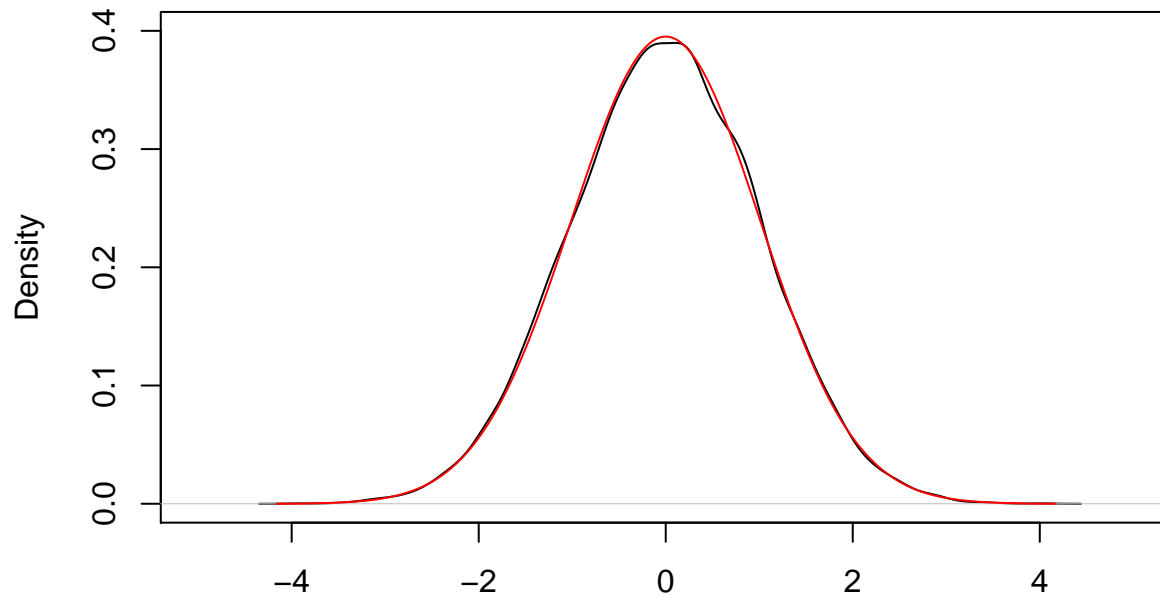


```

# Hmm....looks like t distribution data that also has noise attached to it.
# Let's make use of the density function, which
plot(density(tstats), xlim=c(-5,5), ylim=c(0,0.4))
p<-seq(0, 1, 0.0001)
lines(density(qt(p, df=998)), xlim=c(-5,5), ylim=c(0,0.4), col="red")

```

density.default(x = tstats)

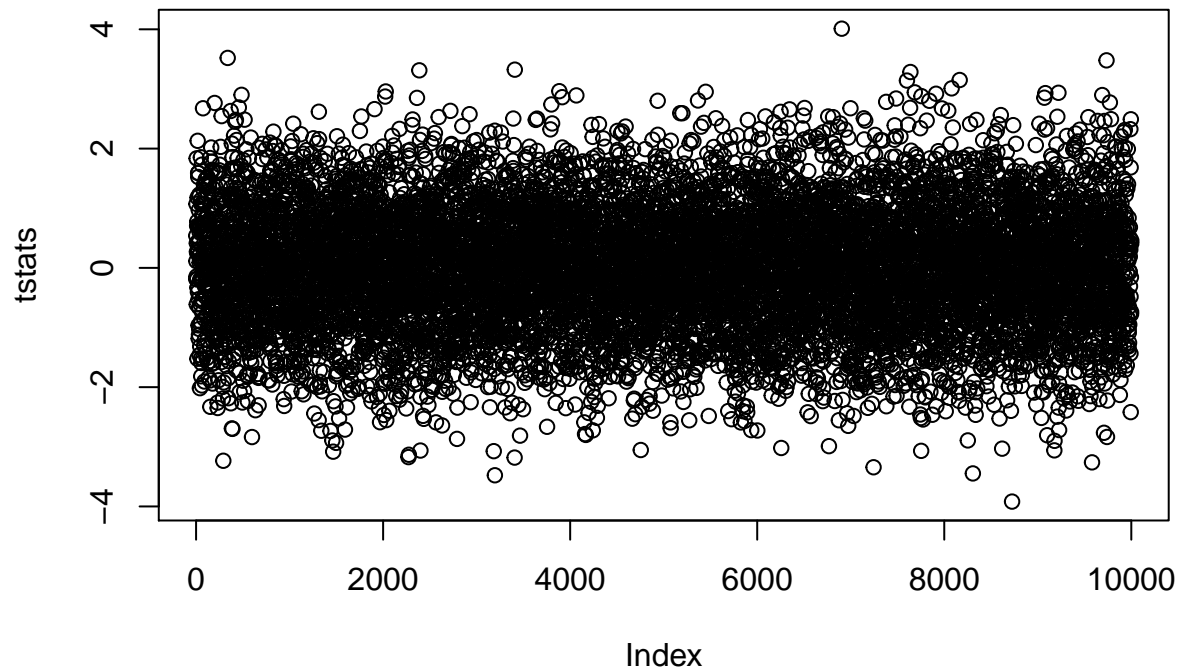


N = 10000 Bandwidth = 0.1435

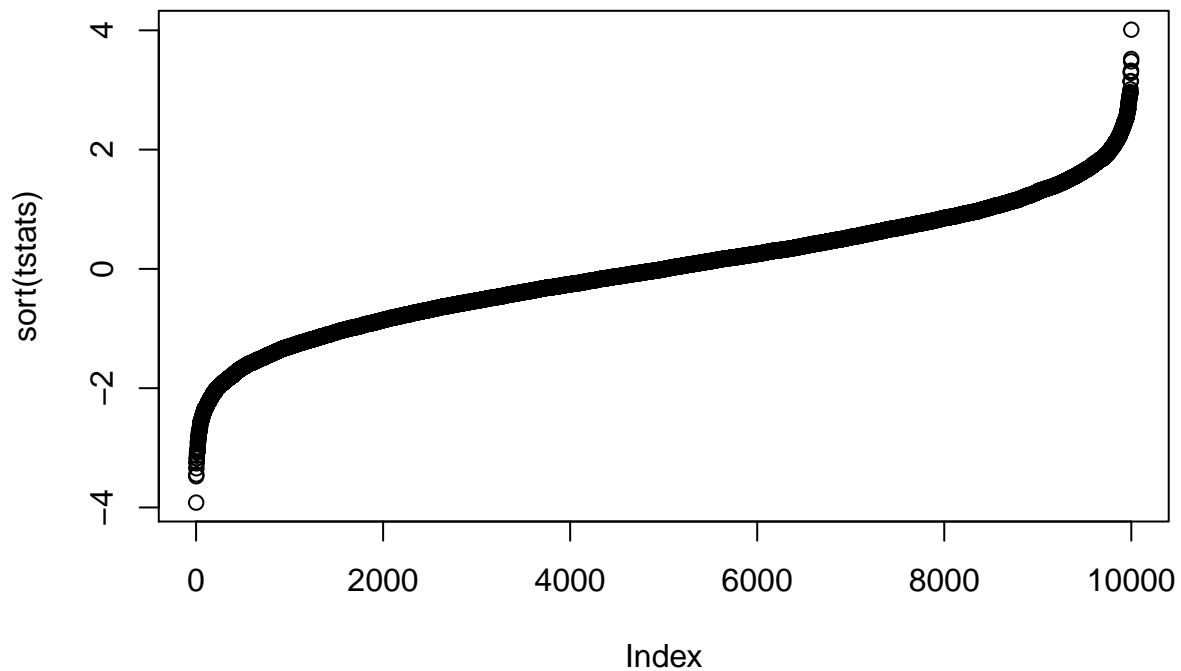
```

# our empirical distribution overlaps with that determined from the t density formula
plot(tstats)

```



```
plot(sort(tstats))
```



```
# out of 10000 randomly generated statistics, if you sort them, you can pull out  
# the ith percentile out of the 10000 available n  
sort(tstats)[9750] # 97.5th percentile from the bootstrapped tstats
```

```
## [1] 1.941606
```

```
# Or use the empirical cumulative distribution function to tell you what percentile  
# a given value would yield if it was derived from your sample distribution  
ecdf(tstats)(1.96)
```

```
## [1] 0.9758
```

```

# Compare these to using the built-in, calculated t distribution functions:
qt(0.975, df=58)

## [1] 2.001717
qt(0.975, df=100)

## [1] 1.983972
qt(0.975, df=10000)

## [1] 1.960201
# 1.96 should ring a bell as the t stat used when estimating 95% confident limits,
# assuming large sample sizes where t distributions resemble normal distributions:
qnorm(0.975)

## [1] 1.959964
t.test(Response ~ Group, data=z, var.equal=T)$statistic

##          t
## 2.408404
anova(lm(Response ~ Group, data=z))

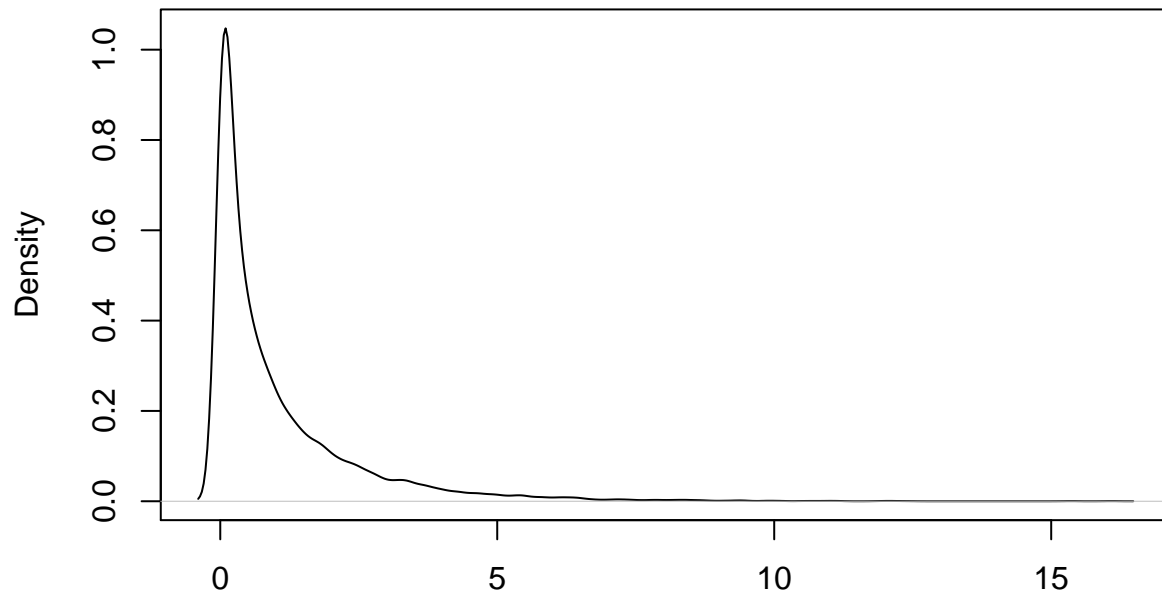
## Analysis of Variance Table
##
## Response: Response
##          Df Sum Sq Mean Sq F value    Pr(>F)
## Group      1  5.006   5.0057   5.8004 0.01922 *
## Residuals 58 50.054   0.8630
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
t.test(Response ~ Group, data=z, var.equal=T)$statistic^2

##          t
## 5.800412
# note: square of the t-statistic = the F statistic from the one way anova, just as
# squaring the tstat distribution should resemble the fstat distribution

# T distribution squared
plot(density(tstats^2))

```

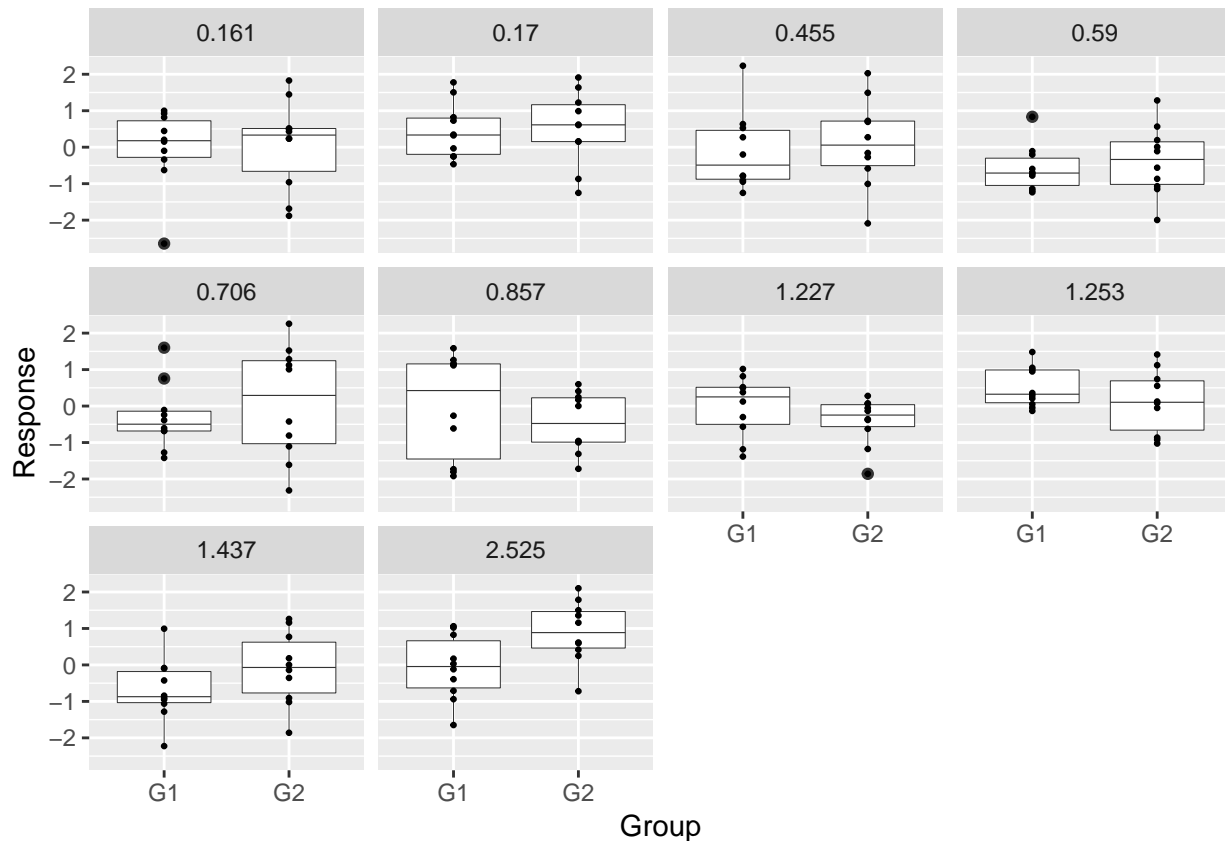
density.default(x = tstats^2)



N = 10000 Bandwidth = 0.1328

```
zz<-data.frame()
for(r in 1:10){
  n=10
  x<-rnorm(n, 0,1)
  y<-rnorm(n, 0,1)
  z<-data.frame(Replicate=r,
                Group=factor(c(rep("G1", n), rep("G2", n))),
                Response=c(x,y),
                Tstat=abs((mean(x)-mean(y))/(sqrt((sd(x)^2)/n+(sd(y)^2)/n))))
  zz<-rbind(zz,z)
}
zz$Tstat<-factor(round(zz$Tstat, 3))

# A random sample of 10 randomly created data sets, assuming 2 groups were drawn from
# a
ggplot(zz, aes(x=Group, y=Response))+
  facet_wrap(~Tstat)+
  geom_boxplot(size=0.1)+
  geom_point(size=0.5)
```



```
# Define Function to return the F statistic for randomly generated data (One Way ANOVA)
fcalc<-function(n1=2, n2=20){
  x<-rnorm(n1, 0, 1)
  y<-rnorm(n2, 0, 1)
  z<-data.frame(Group=factor(c(rep("Group1", n1), rep("Group2", n2))), Response=c(x,y))
  f<-data.frame(anova(lm(Response ~ Group, z)))[1,4]
  return(f)
}
```

```
# Look at F distribution:
```

```
n1=20
n2=20
x<-rnorm(n1, 0, 1)
y<-rnorm(n2, 0, 1)
z<-data.frame(Group=factor(c(rep("Group1", n1), rep("Group2", n2))), Response=c(x,y))
anova(lm(Response ~ Group, z))
```

```
## Analysis of Variance Table
```

```
##
```

```
## Response: Response
```

```
##      Df Sum Sq Mean Sq F value Pr(>F)
## Group  1  3.577   3.5770   3.5974 0.06549 .
```

```
## Residuals 38 37.785   0.9943
```

```
## ---
```

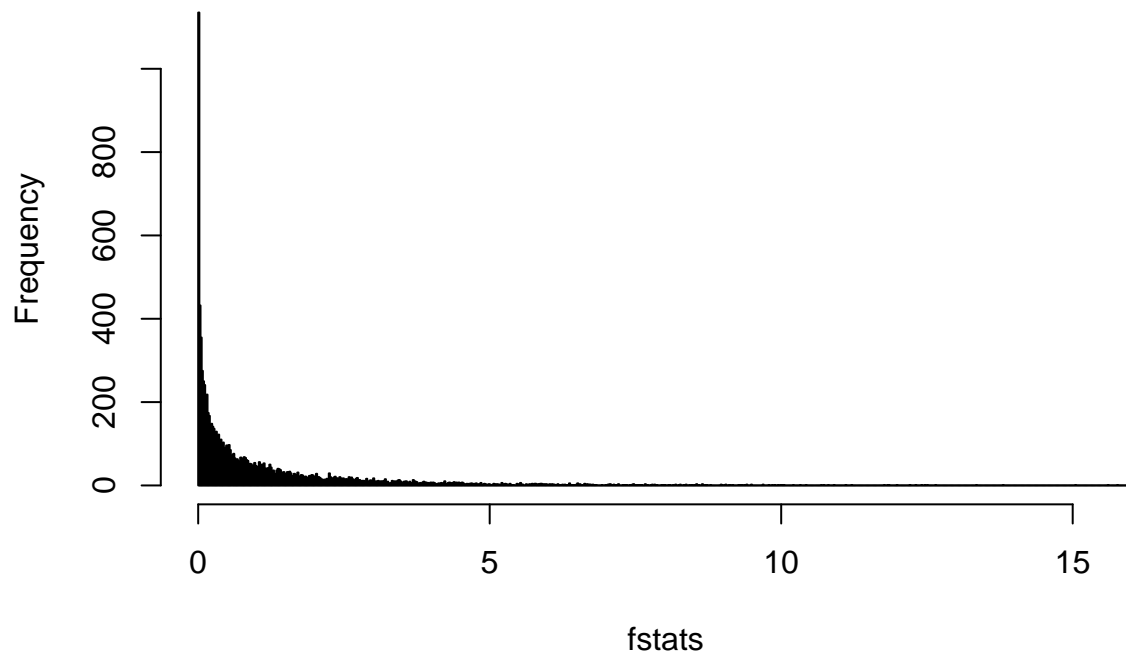
```
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```

# Replicate two groups that you want to perform an ANOVA on.
# If n1=20 and n2=20, the degrees of freedom from a one-way ANOVA will be:
# Numerator df = 1
# Denominator df = 38
fstats<-replicate(10000, fcalc(n1=20, n2=20))
hist(fstats, breaks=1000)

```

Histogram of fstats

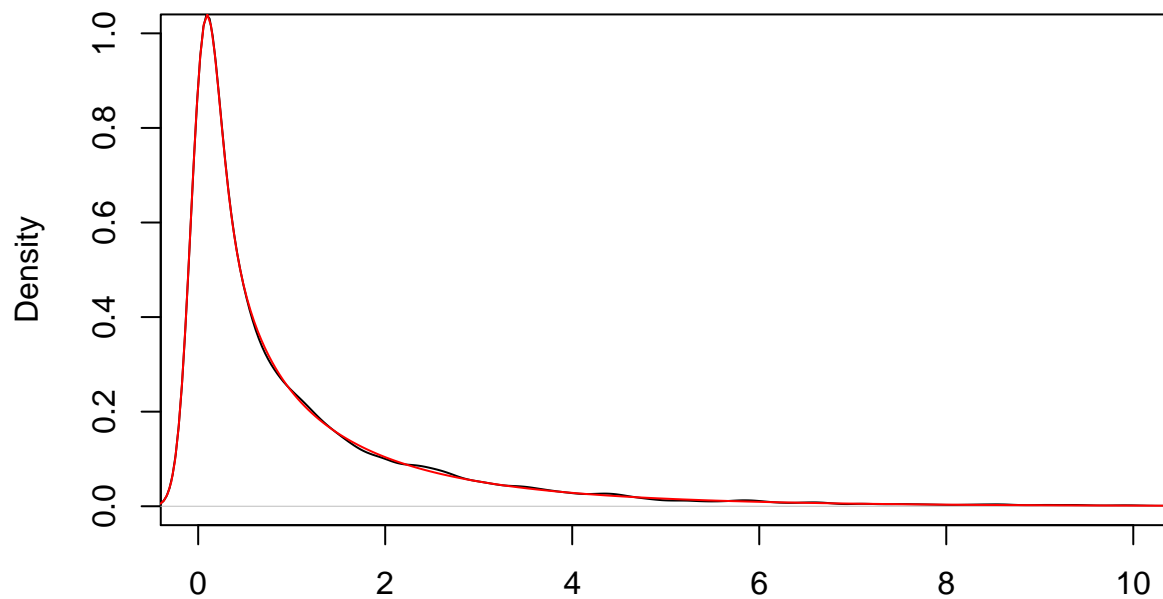


```

plot(density(fstats), xlim=c(0,10), ylim=c(0,1))
lines(density(qf(seq(0, 1, 0.0001), df1=1, df2=38, lower.tail = T)),
      xlim=c(0,10), ylim=c(0,1), col="red")

```


density.default(x = fstats)



N = 10000 Bandwidth = 0.135

```
sort(fstats)[9500] # 95% percentile from the fstats replicated values
```

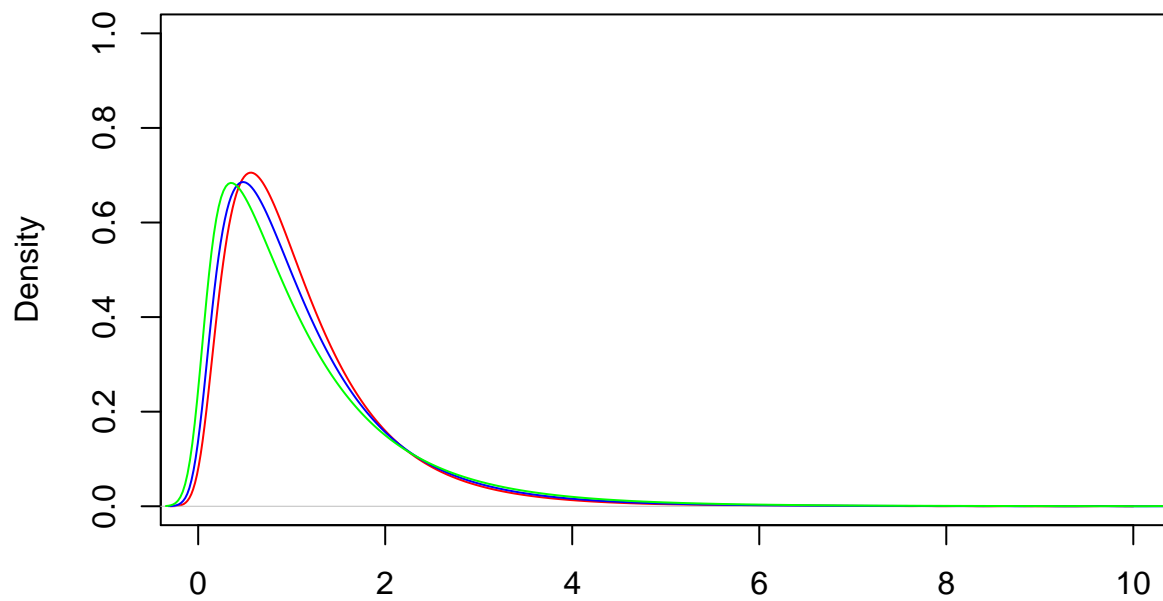
```
## [1] 4.100785
```

```
qf(0.95, df1=1, df2=38) # 95th percentile calculated using the built in function
```

```
## [1] 4.098172
```

```
plot(density(qf(seq(0, 1, 0.0001), df1=5, df2=19, lower.tail = T)),  
      xlim=c(0,10), ylim=c(0,1), col="red")  
lines(density(qf(seq(0, 1, 0.0001), df1=4, df2=19, lower.tail = T)),  
       xlim=c(0,10), ylim=c(0,1), col="blue")  
lines(density(qf(seq(0, 1, 0.0001), df1=3, df2=19, lower.tail = T)),  
       xlim=c(0,10), ylim=c(0,1), col="green")
```

density.default(x = qf(seq(0, 1, 1e-04), df1 = 5, df2 = 19, lower.tail = T



N = 10001 Bandwidth = 0.09851