# Lab 5 - Simulation from chisquared, t and F-distribution

---

## Lab Goals

1. Create random draws that should follow various distributions, according to the theoretical formulas;confirm that they have the right distributions by comparing histograms to density curves of the desired distributions
2. Practice the use of the distributions to derive test statistics and confidence intervals for linear regression.
3. Construct F statistics for groups of coefficients.
4. Calculate the power of tests.

*Stats concepts*: - Chi-squared, t, and F distributions - Relationship to normal distribution - Noncentral distributions - QQ plots - Power

*R concepts*: - Generating random draws from a distribution - Getting quantiles of a distribution - qqplot() - Adding points to a plot - Adding a legend to a plot

## Some Simulations

Lets start out by generating some normals. *Note: you can reproduce the same simulation using set.seed(); e.g. set.seed(100)*

```
N=1000                          # number of simulations
s=(1:N)/(N+1)
df1=5; df2=15                   # degrees of freedom
z1=rnorm(N)                     # N vector of normals
z2=rnorm(N)                     #
Z1=matrix(rnorm(N*df1),N,df1)   # N by df1 matrix of normals
Z2=matrix(rnorm(N*df2),N,df2)   #
```

What is the relationship between chi-squared and normal distributions? Check the theory, in class notes from last week!

```
x1=apply(Z1^2,1,sum)            # rchisq(N,df1)
x2=apply(Z2^2,1,sum)            # rchisq(N,df2)
```

What is the relationship between t and chi-squared distributions?

```
t1=z1/sqrt(x1/df1)              # rt(N,df1)
t2=z2/sqrt(x2/df2)              # rt(N,df2)
```

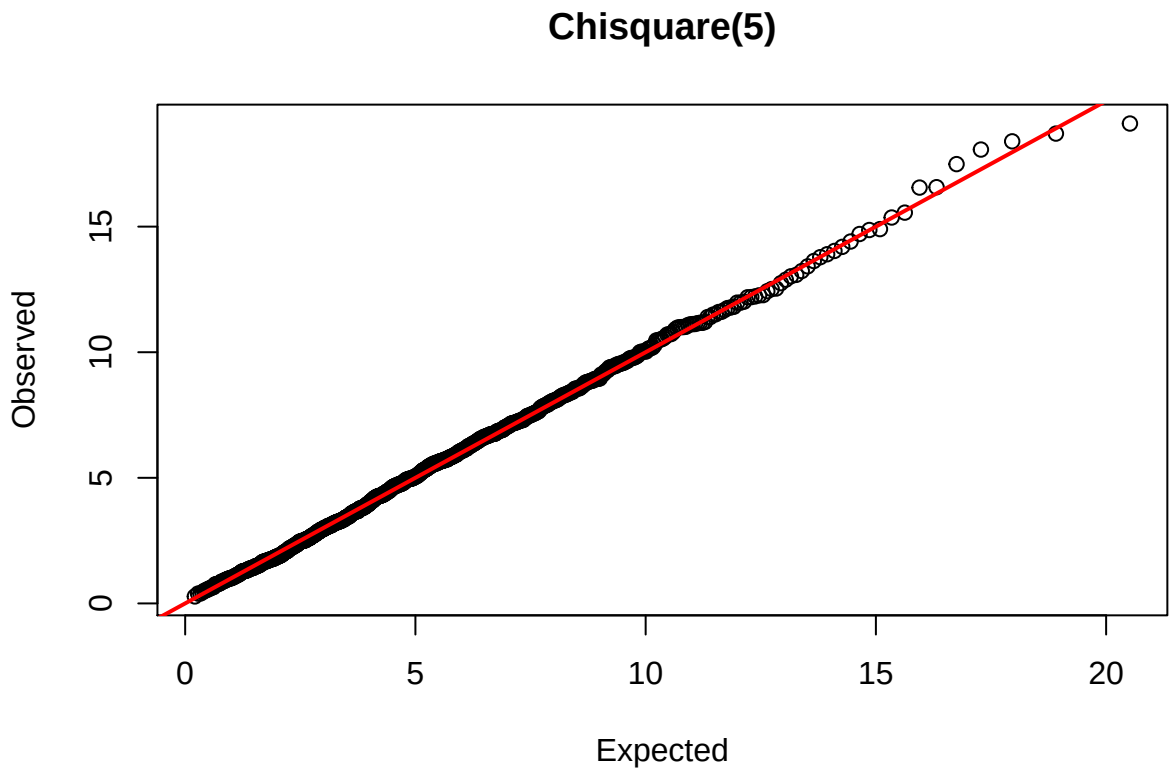
What is the relationship between F and chi-squared distributions?

```
f12=(x1/df1)/(x2/df2)           # rf(N,df1,df2)
f21=(x2/df2)/(x1/df1)           # rf(N,df2,df1)
```

Check the chisquared(5) sample using a qqplot

```
q=qchisq(s,df1) ##What is the vector q giving you?
qqplot(q,x1,xlab="Expected",ylab="Observed",pch=1,
```

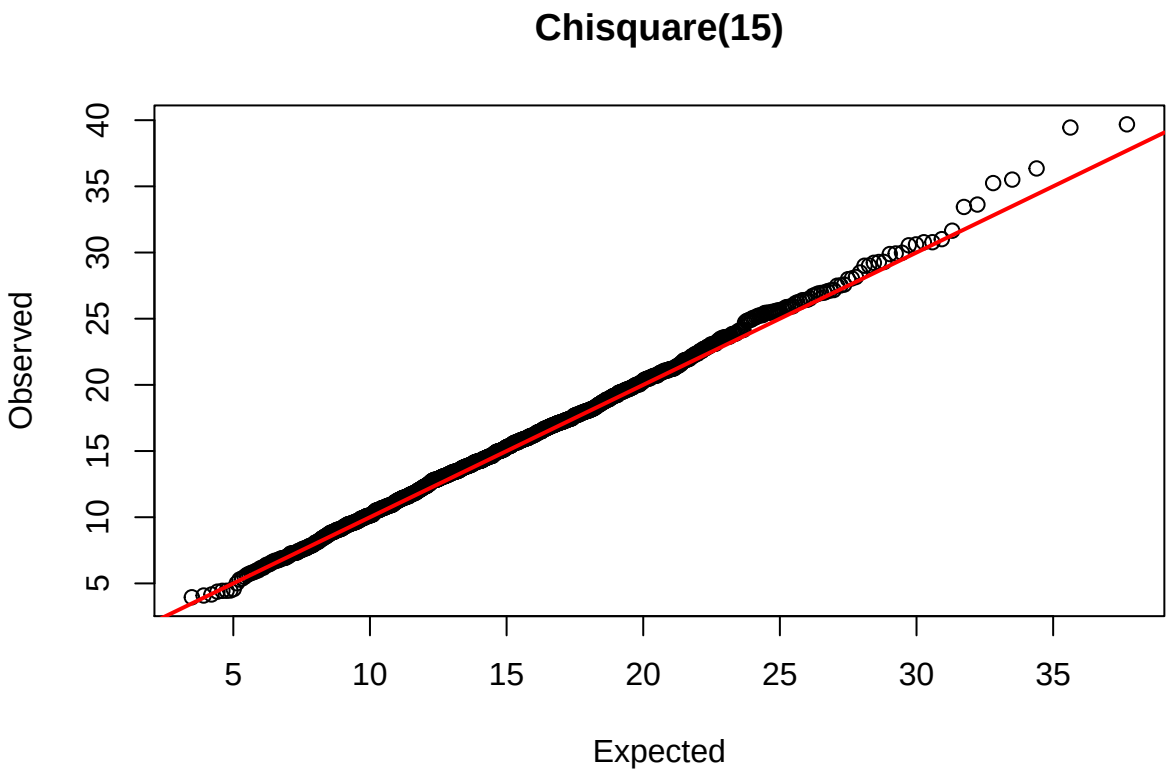
```
       main="Chisquare(5)")
abline(0,1,col="red",lwd=2)
```




- What is the relationship between the points and the line? - What does that mean? - What would you think if they didn't have that relationship?

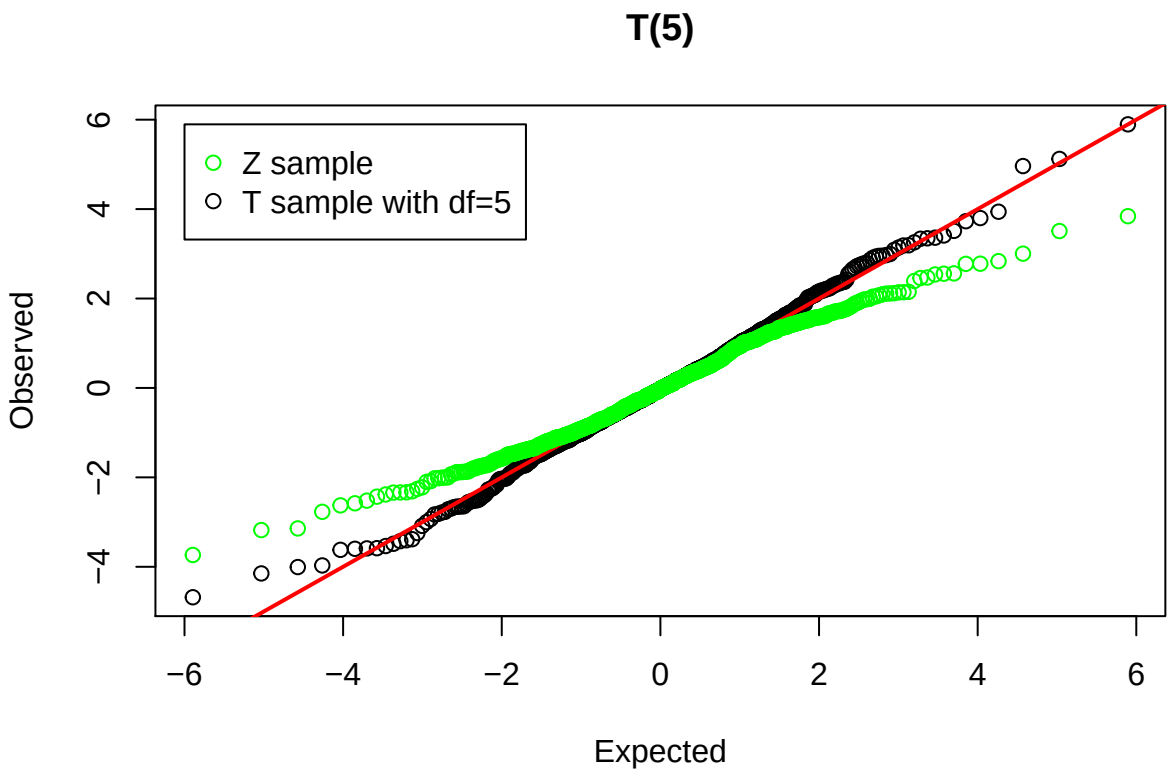
chisquared(15) qqplot

```
q=qchisq(s,df2)
qqplot(q,x2,xlab="Expected",ylab="Observed",pch=1,
       main="Chisquare(15)")
abline(0,1,col="red",lwd=2)
```




Check the t(5) samples using a qqplot

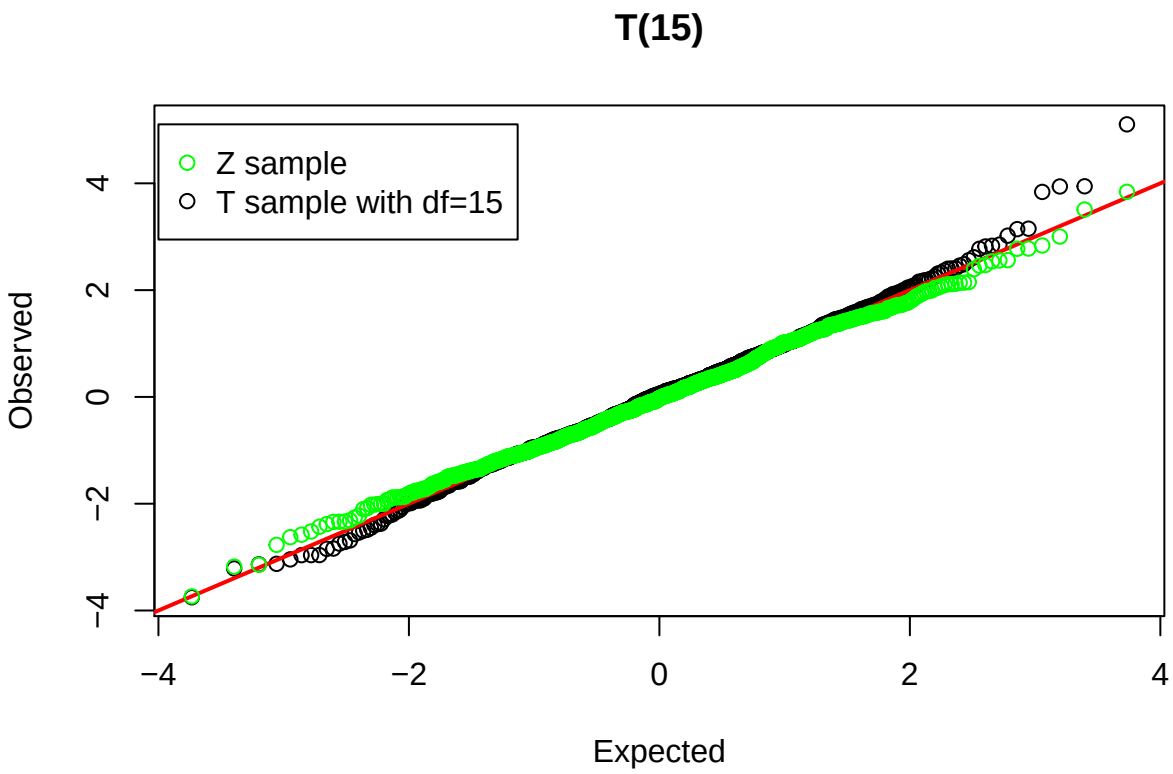
```
q=qt(s,df1)
qqplot(q,t1,xlab="Expected",ylab="Observed",pch=1,
       main="T(5)")
abline(0,1,col="red",lwd=2)

# Add standard normal order statistics versus t-quantiles

points(q,sort(z1),col="green")

##The points in z1 come from what distribution?
legend(-6,max(t1),pch=1,col=c("green","black"),
       legend=c("Z sample","T sample with df=5"))
```




What do the green points look like relative to the black points? Why?

t(15) qqplot

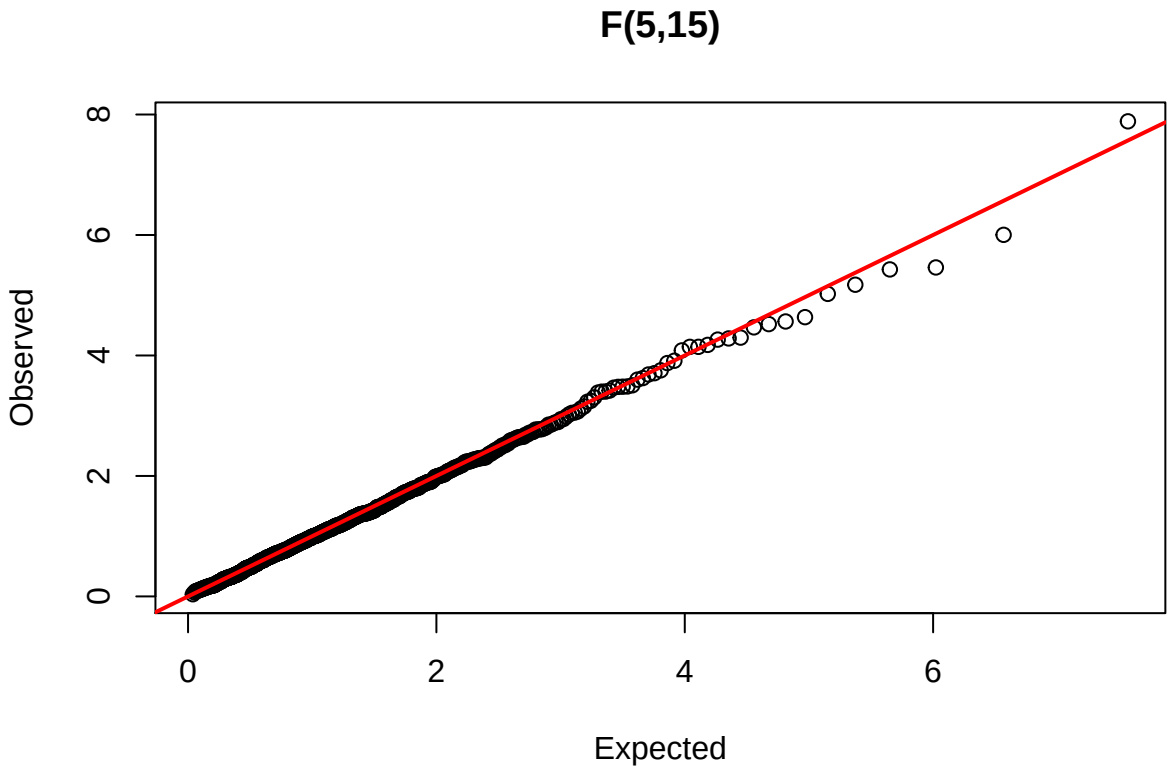
```
q=qt(s,df2)
qqplot(q,t2,xlab="Expected",ylab="Observed",pch=1,
       main="T(15)")
abline(0,1,col="red",lwd=2)
points(q,sort(z1),col="green")
legend(-4,max(t2),pch=1,col=c("green","black"),
       legend=c("Z sample","T sample with df=15"))
```




Compare this plot to the previous one. How do they look different? Why?

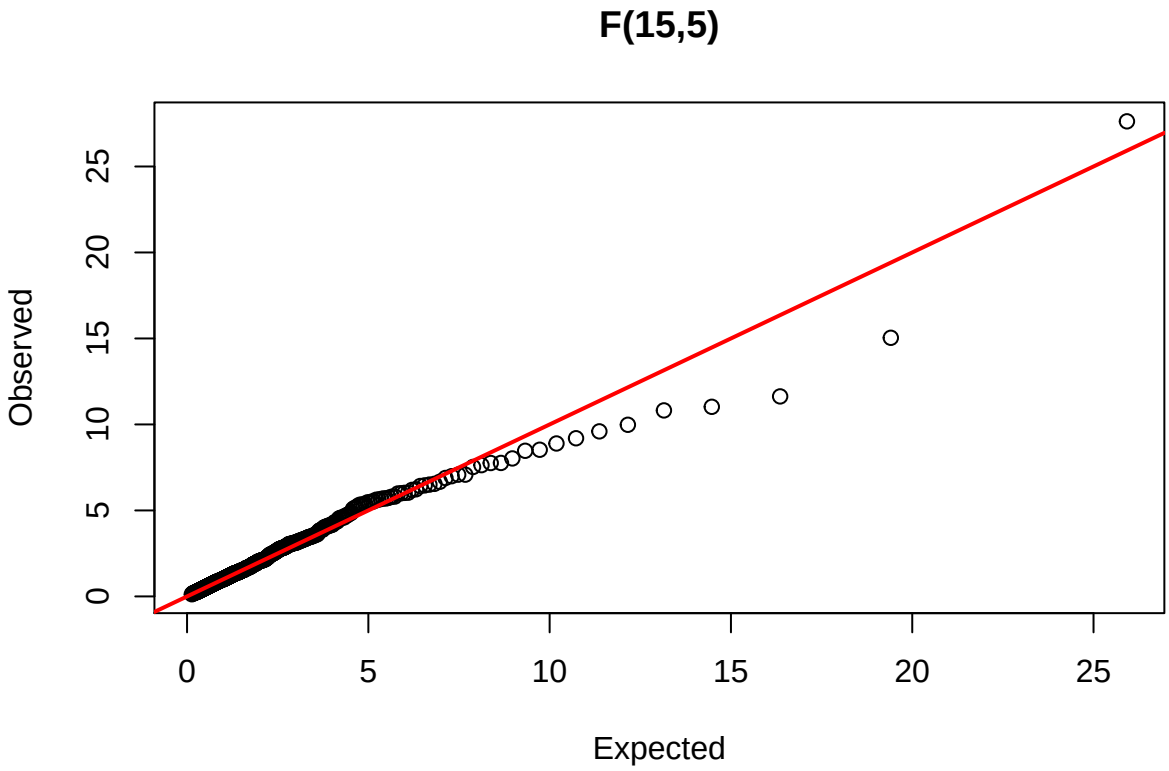
Check the F(5,15) sample using a qqplot

```
q=qf(s,df1,df2)
qqplot(q,f12,xlab="Expected",ylab="Observed",pch=1,
       main="F(5,15)")
abline(0,1,col="red",lwd=2)
```



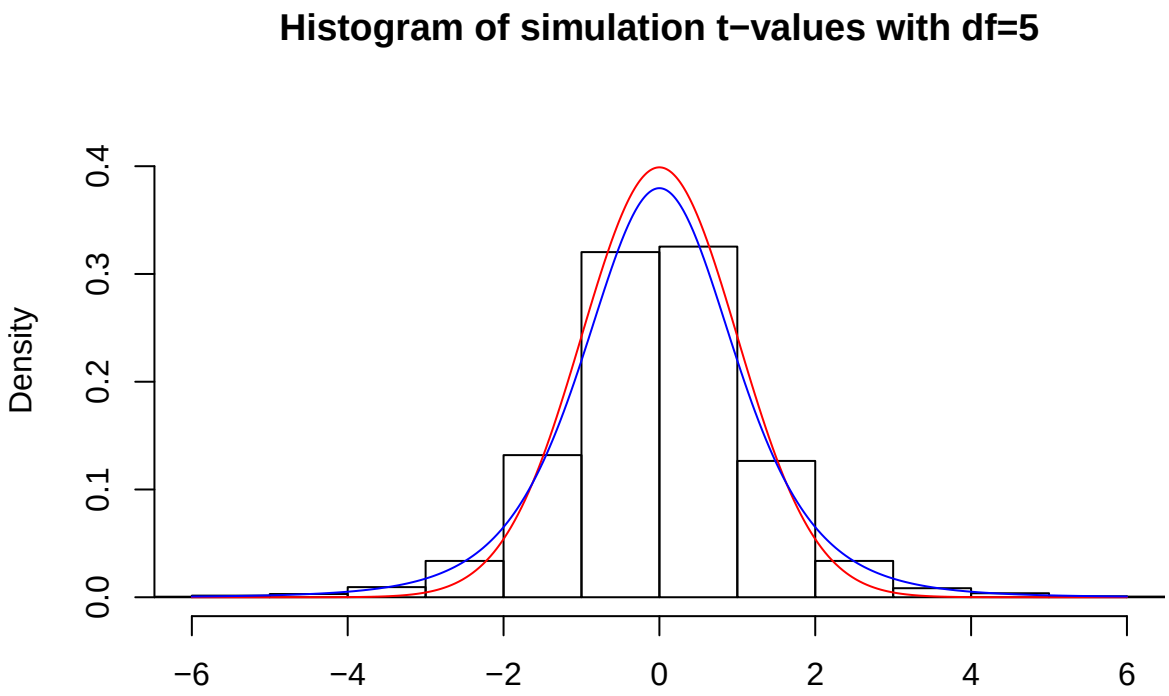

F(15,5) qqplot

```
q=qf(s,df2,df1) ##Note that the order of df1 & df2 matters!
qqplot(q,f21,xlab="Expected",ylab="Observed",pch=1,
       main="F(15,5)")
abline(0,1,col="red",lwd=2)
```




Histogram of sample from t-distribution comparision with standard normal density

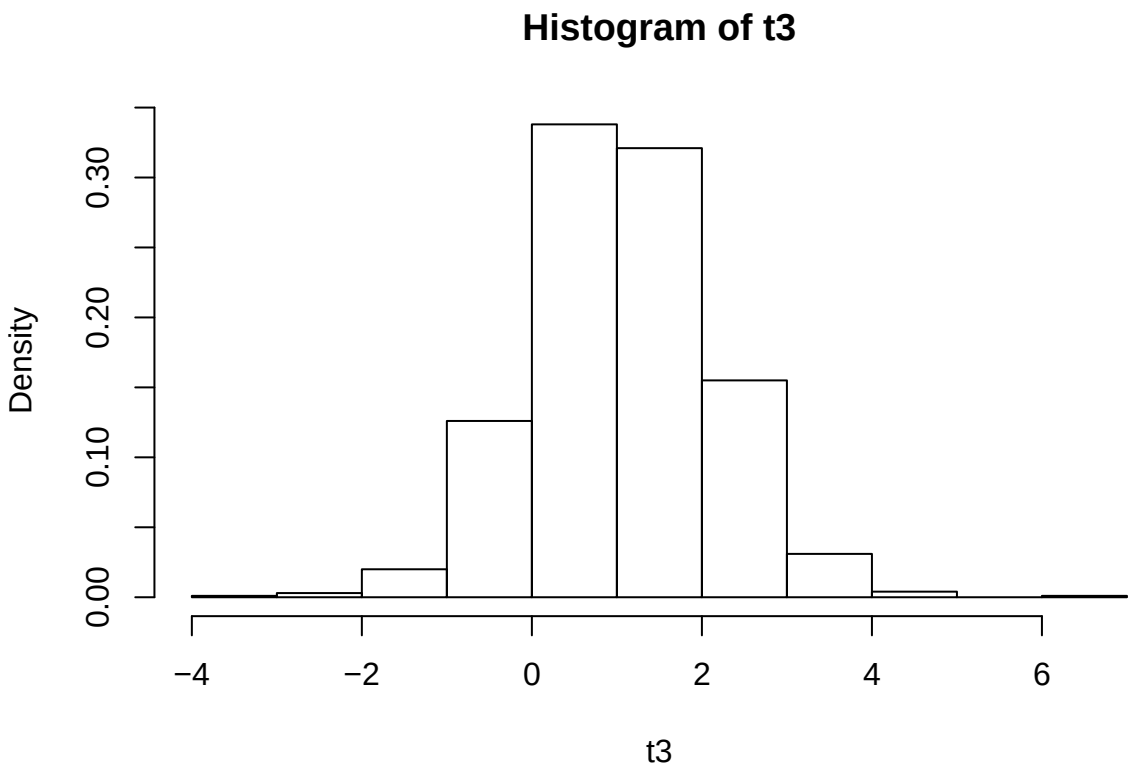
```
df=5
t=rt(10000,df) ##The points in t come from what distribution?
x=seq(-6,6,.01)
dz=dnorm(x)
dt=dt(x,df)
dmax=max(dz) ##Why do we care what dmax is? (Where do we use it below?)
hist(t,prob=TRUE,ylim=c(0,dmax*1.1),xlim=c(-6,6),
     main="Histogram of simulation t-values with df=5",xlab="")
##If you got an error here, run it again and see if it works
##...then figure out why you got an error one time but not the other time
lines(x,dz,col="red")
lines(x,dt,col="blue") # better fit especially in the tails
```



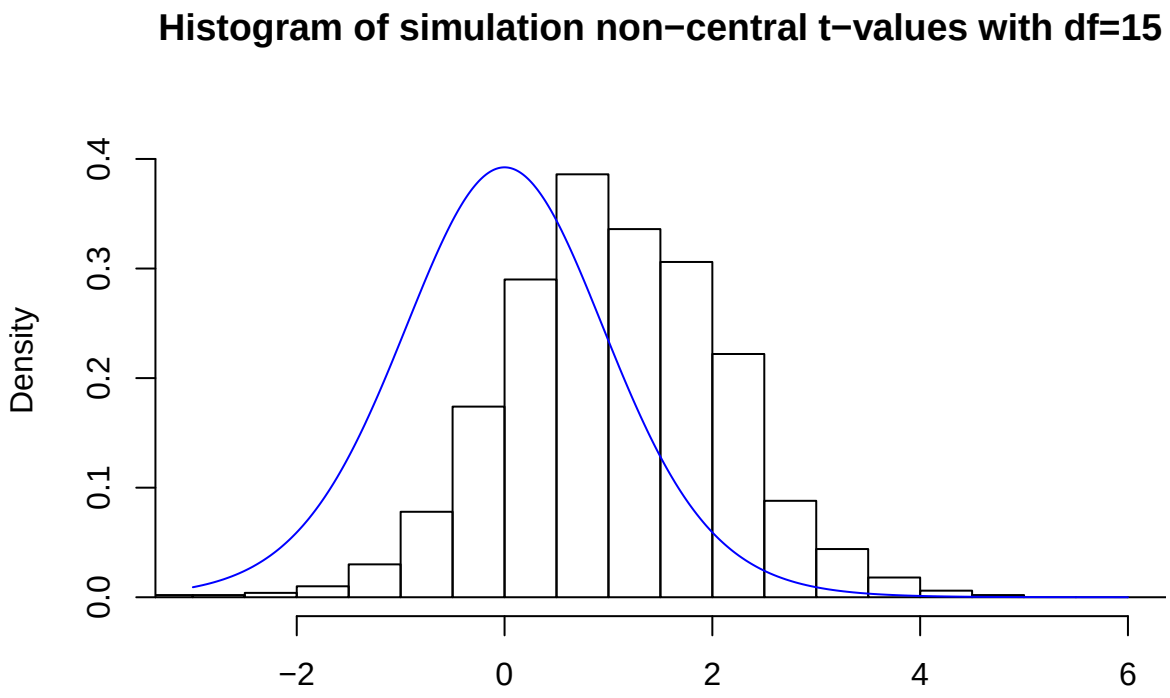

Use "Zoom" to get a better look! Now repeat this block with df=30. What do you see now?

## Non-central distributions

```
mu=1
z3=rnorm(N,mean=mu)
t3=z3/sqrt(x2/df2)
hist(t3,prob=TRUE)
```



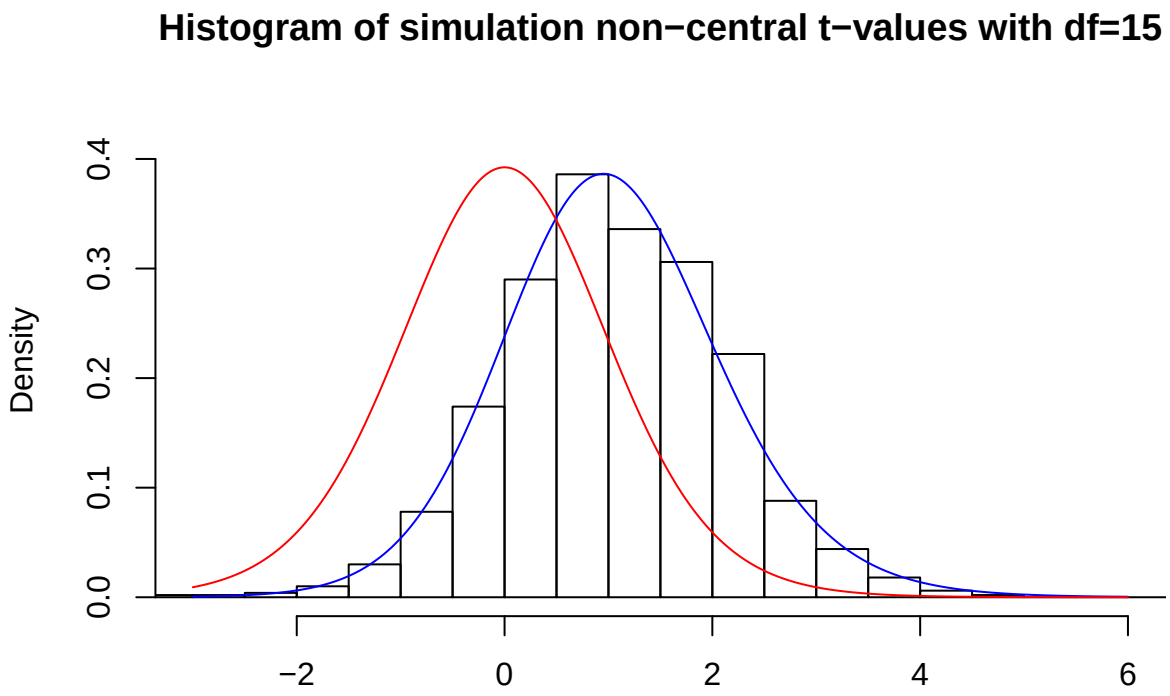

```
x=seq(-3,6,.01)
dt=dt(x,df2)
dmax=max(dt)
hist(t3,prob=TRUE,ylim=c(0,dmax*1.1),xlim=c(-3,6),
     breaks=seq(-20,20,0.5),
     main="Histogram of simulation non-central t-values with df=15",xlab="")
lines(x,dt,col="blue")
```




Does the line fit the histogram? Why?

Here we will try re-drawing with a histogram specifying a noncentrality parameter

```
hist(t3,prob=TRUE,ylim=c(0,dmax*1.1),xlim=c(-3,6),
     breaks=seq(-20,20,0.5),
     main="Histogram of simulation non-central t-values with df=15",xlab="")

dt3 = dt(x,df2,ncp=mu)
lines(x,dt3,col="blue")
lines(x,dt,col='red')
```




which accounts for the normal random variable being centered away from zero.

We can also add noncentrality parameters to an $F$ distribution. Recall that $F$ is defined by a ratio of $\chi^2$. If we replace Z1 above with

```
Z1.2 = matrix(rnorm(N*df1, mean = mu),N,df1)
```

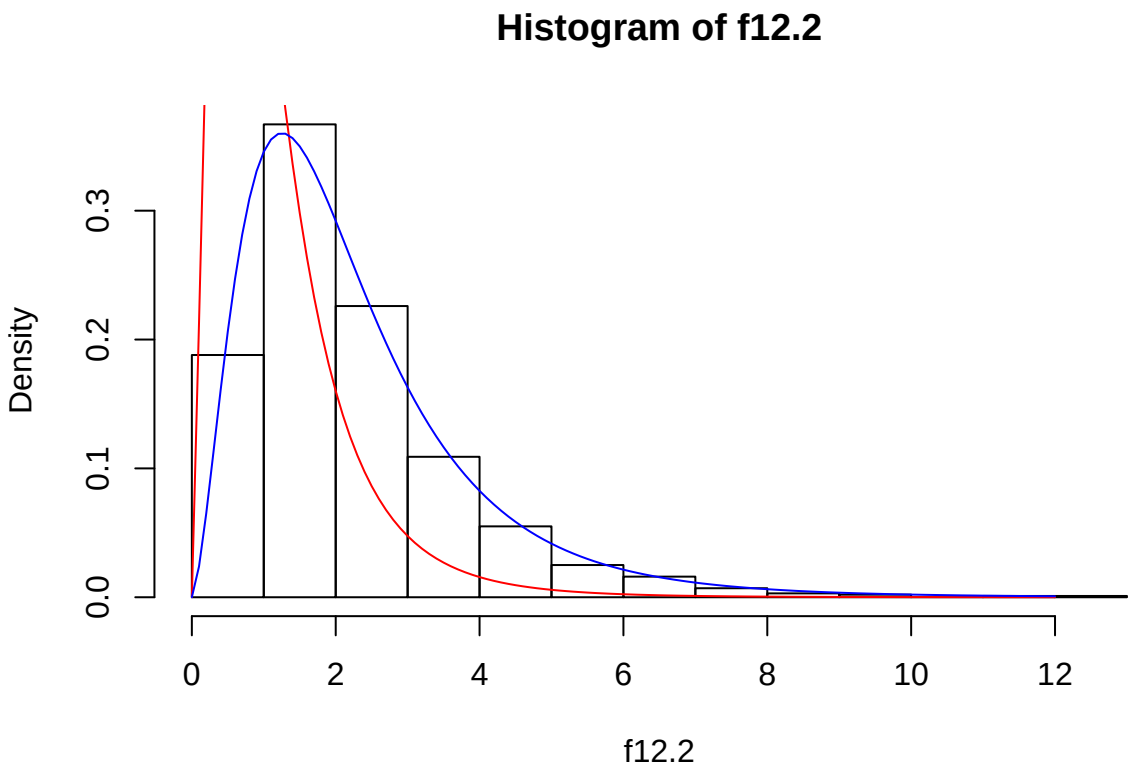
Then it's sum of squares is a non-central chi-square

```
x1.2=apply(Z1.2^2,1,sum)
```

and we can form a non-central F (note that the denominator $\chi^2$ still has to be central)

```
f12.2=(x1.2/df1)/(x2/df2)
```

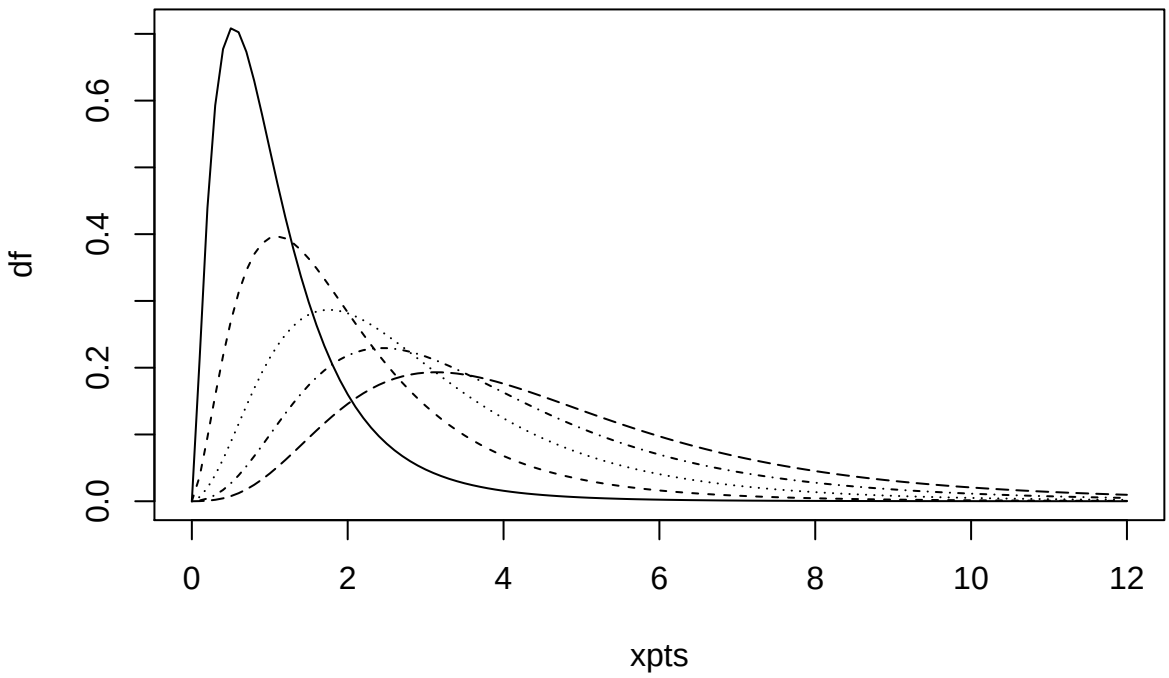
```
hist(f12.2,prob=TRUE)

xpts = seq(0,12,by=0.1)
df = df(xpts,df1,df2)
lines(xpts,df,col="red")

df.2 =df(xpts,df1,df2,ncp=df1*mu^2)
lines(xpts,df.2,col="blue")
```




Note that R uses $\sum \mu_i^2$ as a non-centrality parameter, rather than dividing it by 2.

Notice that as the ncp increases the density shifts further to the right:

```
plot(xpts,df,type='l')
lines(xpts,df(xpts,df1,df2,ncp=4),type='l',lty=2)
lines(xpts,df(xpts,df1,df2,ncp=8),type='l',lty=3)
lines(xpts,df(xpts,df1,df2,ncp=12),type='l',lty=4)
lines(xpts,df(xpts,df1,df2,ncp=16),type='l',lty=5)
```




## Constructing t and F Tests

Let's use some of these statistically. For this, we will refer to data from 2018 midterm. Here we have the Speed at which each of 3 Cultivars of grass grows on golf greens. These are also affected by which of 8 regions the grass is grown in, and Humidity.

```
Grass = read.table('Grass.csv',head=TRUE,sep=',')
Grass$Region = as.factor(Grass$Region)
```

In these data, we have coded Cultivar with two indicator functions, but left Region as a factor in R. Our basic model is

```
mod = lm(Speed~.,data=Grass)
summary(mod)
```

```
##
## Call:
## lm(formula = Speed ~ ., data = Grass)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.23817 -0.08706 -0.01128  0.04992  0.29325
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  8.421762   0.169847  49.584 3.34e-16 ***
```

```
## Humidity    -0.022765   0.002453  -9.281 4.25e-07 ***
## Region2     -0.072989   0.155935  -0.468   0.6475
## Region3     -0.084832   0.155846  -0.544   0.5954
## Region4     -0.186642   0.168924  -1.105   0.2892
## Region5      0.434006   0.164553   2.637   0.0205 *
## Region6      0.340397   0.158506   2.148   0.0512 .
## Region7      0.433041   0.164995   2.625   0.0210 *
## Region8      0.252458   0.155974   1.619   0.1295
## C2           0.917971   0.095581   9.604 2.87e-07 ***
## C3           1.885567   0.095644  19.714 4.55e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1904 on 13 degrees of freedom
## Multiple R-squared:  0.975,  Adjusted R-squared:  0.9559
## F-statistic:  50.8 on 10 and 13 DF,  p-value: 9.257e-09
```

and we can look at

```
library(car)
```

```
## Warning: package 'car' was built under R version 3.5.1
```

```
## Loading required package: carData
```

```
Anova(mod)
```

```
## Anova Table (Type II tests)
##
## Response: Speed
##             Sum Sq Df  F value    Pr(>F)
## Humidity    3.1225  1  86.1339 4.247e-07 ***
## Region      1.3045  7   5.1406  0.005481 **
## C2          3.3438  1  92.2396 2.869e-07 ***
## C3         14.0893  1 388.6571 4.554e-11 ***
## Residuals   0.4713 13
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

but our primary interest is in testing for the difference between cultivars.

Let's reconstruct the t-test for humidity. We first obtain model matrices

```
X  = model.matrix(mod)
p = ncol(X); n = nrow(X)  # Note that p includes the intercept here
```

and set up estimates and hat matrices

```
iXX = solve(t(X)%*%X)

# Estimate
betahat = iXX%*%t(X)%*%Grass$Speed

# Hat matrix
H = X%*%iXX%*%t(X)

# MSE
mse = sum( (Grass$Speed - H%*%Grass$Speed)^2 )/(n - p)
```

```
# Beta Standard Error
beta.var =  mse*iXX
```

and calculate a t-tstatistic for beta[2]

```
t.obs = betahat[2]/sqrt(beta.var[2,2])
t.obs
```

```
## [1] -9.280833
```

and we can calculate a p-value from

```
2*(1-pt(abs(t.obs),n-p))
```

```
## [1] 4.246771e-07
```

*Verify this with the summary table above. Why do we use `1-2*pt()`?*

Now we'll construct a sum of squares between a model with the Cultivars and without and test the hypothesis that the joint effect of C2 and C3 is 0.

That is we'll construct a hat matrix that doesn't use them

```
X2 = X[,1:9]  # Drop Cultivars
H2 = X2%*%solve(t(X2)%*%X2, t(X2))
```

Now the sum of squares is

```
SS.C = t(Grass$Speed)%*%(H-H2)%*%Grass$Speed
SS.C
```

```
##          [,1]
## [1,] 14.09484
```

**Test that this is the same as the sum of squared differences between mod$fit and the fitted values for a model without C2 and C3.**

We will use this to construct an F statistic

```
F.obs =  (SS.C/2)/mse
1-pf(F.obs,2,n-p)
```

```
##              [,1]
## [1,] 2.06299e-10
```

**Problem** construct a test for Region.

# Power Analysis

Finally, we'd like to look at power. For a t-test this is the probability that a $t_k(\lambda)$ random variable is larger than $t_k^{1-\alpha/2}$.

Formally, with df2 = 15 degrees of freedom, the critical value is

```
t.crit = qt(0.975,df2)
t.crit
```

```
## [1] 2.13145
```

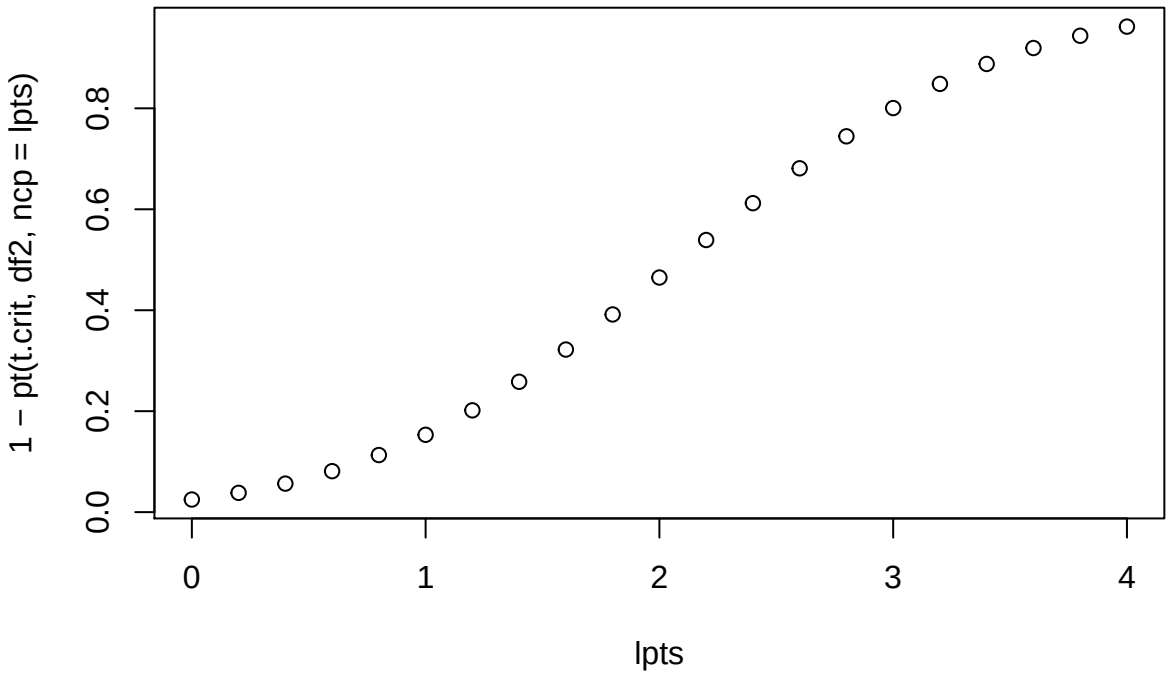
then if we take *lambda* = 1, we have that the power is given by

```
1-pt(t.crit,df2,ncp=1)
```

```
## [1] 0.1531911
```

and we can look at various values of lambda

```
lpts = seq(0,4,0.2)  # possible lambda points
plot(lpts, 1-pt(t.crit,df2,ncp=lpts))
```

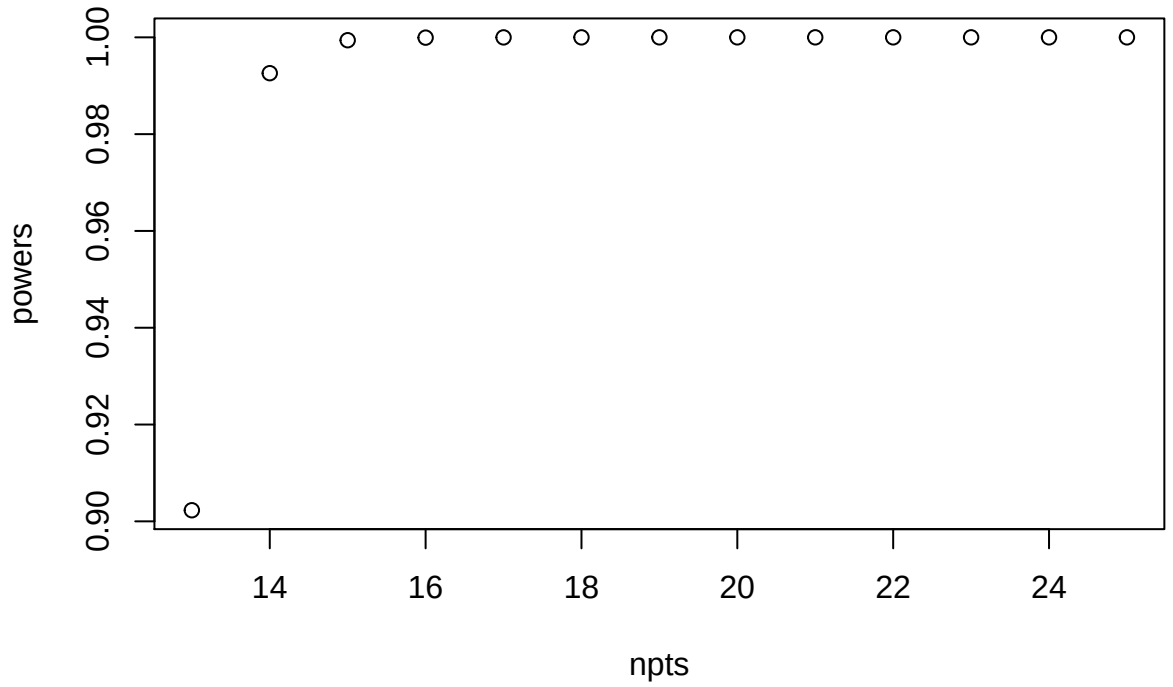



One way we can look at this is in experimental design. Suppose we look at changing sample size, with $\lambda = \sqrt{n} * T$ where

```
T = abs(t.obs/sqrt(n))
```

(this comes from assuming that $X^T X/n \approx C$ for any $n$), then one thing to notice is that the degrees of freedom also change, and therefore so does the critical value. We can set this up from

```
npts = 13:25
t.crits = qt(0.975,df=npts-11)
powers = 1-pt(t.crits, df=npts-11, ncp = sqrt(npts)*T)
plot(npts,powers)
```




where we can see that 5 data points would be enough to achieve the default requirement of 80% power.

**Problem** perform an experiment to calculate the number of data points necessary to achieve 80% power to find the effect of C2 and C3. To do this a) assume that for a given $n$, SS.C is given by $nK$ where you estimate $K$ from the data by $SS.C/n$ using our SS.C above (note that we should use $SS.C/n\sigma^2$, but here we will set $\sigma^2 = 1$ to produce a power curve that looks interesting). b) remember that the denominator degrees of freedom change with n, but the numerator remains the same at 2.

Plot the power as n increases (what is the minimum n you could have?) and find where it crosses 80%.