WebersLaw

(Have to run R code snippets in order, and make sure plots are going to the plot window.)

Sean's problem is a good chance to think through the math for the probability distribution of a random variable that has undergone a transformation. In this case, the transformation will be using Weber's Law to translate a desired perceived line length (p) to an actual length (a). Andrew has already done the work of coding up this transformation in a format that works for Sean's experiment.

Here are the functions

```
#Weber's Law (perception is the log of actual stimulus magnitude)
phy2psy=function(a) log(a/150,1.05)

#inversion (Andrew's function)
psy2phy <- function(p) {
   return(150*1.05^p)
}
```

Following Andrew, I will use the following distributions of *perceived* line length:

Non-member: uniform ranging from 0 to 22.52.

The values in this distribution are samples of p measured as number of just-noticeable-differences (JNDs) above the minimum line length stimulus.

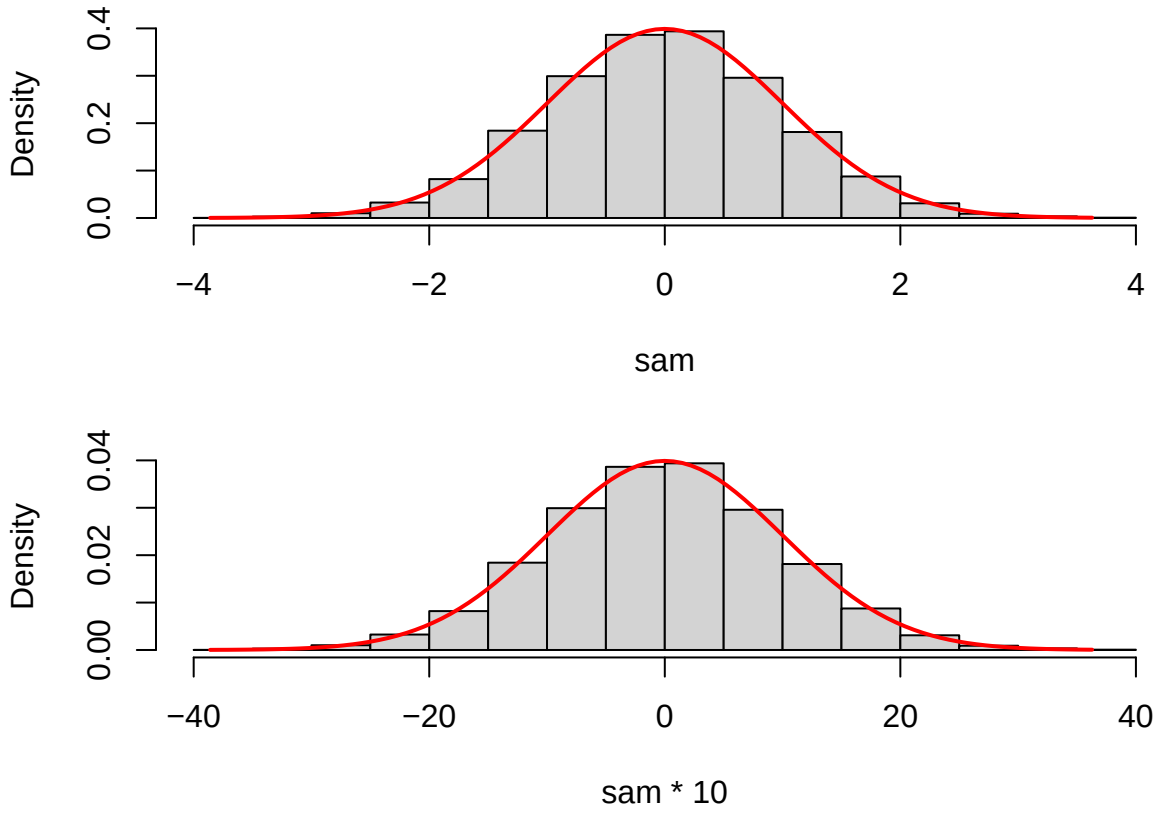
Member: normal with mean 11.26 and SD 2.252 (see Andrew's files)

```
#non-member samples of perceived length
pNon=runif(100000,0,22.52)

#member samples of perceived length
pMem=rnorm(100000,11.26,2.252)

#as in Andrew's file, here are corresponding samples for the actual line lengths
aNon=psy2phy(pNon)

aMem=psy2phy(pMem)
```

OK, we know the probability density of the original distribution of perceived lengths, and we want to work out the probability density for the transformed lengths (transformed to the actual-length scale). Part of this is easy: just go find the probability density for the untransformed value that corresponds to a particular transformed value with phy2psy(a).

That doesn't work by itself if the transformation involves stretching the scale, though. Probability density makes area probability, so if you stretch out the scale you need to change the density to get the right area. For example, if you take a random variable distributed as a standard normal and multiply every score by 10, you need to divide all densities by 10 to get an area of 1 across the much bigger span of the converted variable:

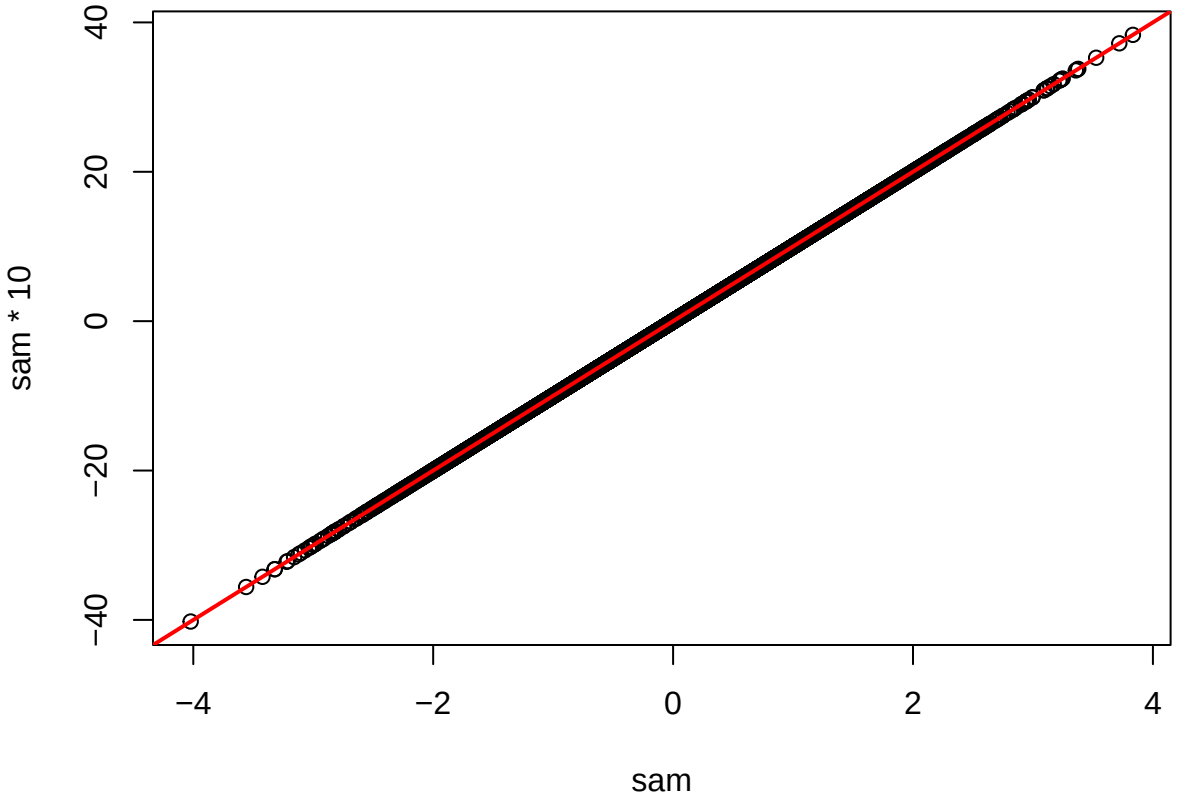
```
sam=rnorm(10000)
par(mfcol=c(2,1),mar=c(5,5,1,1))
hist(sam,freq=F,
       main="")
xSeq=seq(min(sam),max(sam),length.out=100)
points(xSeq,dnorm(xSeq),type='l',col='red',lwd=2)

transDens=function(x) dnorm(xSeq/10) / 10
hist(sam*10,freq=F,
       main="")
xSeq=seq(min(sam*10),max(sam*10),length.out=100)
points(xSeq,transDens(xSeq),type='l',col='red',lwd=2)
```



More generally, to get the density of the transformed variable, you need to divide the density for the corresponding value on the original scale by how much the transformation stretches the scale (as a multiple).

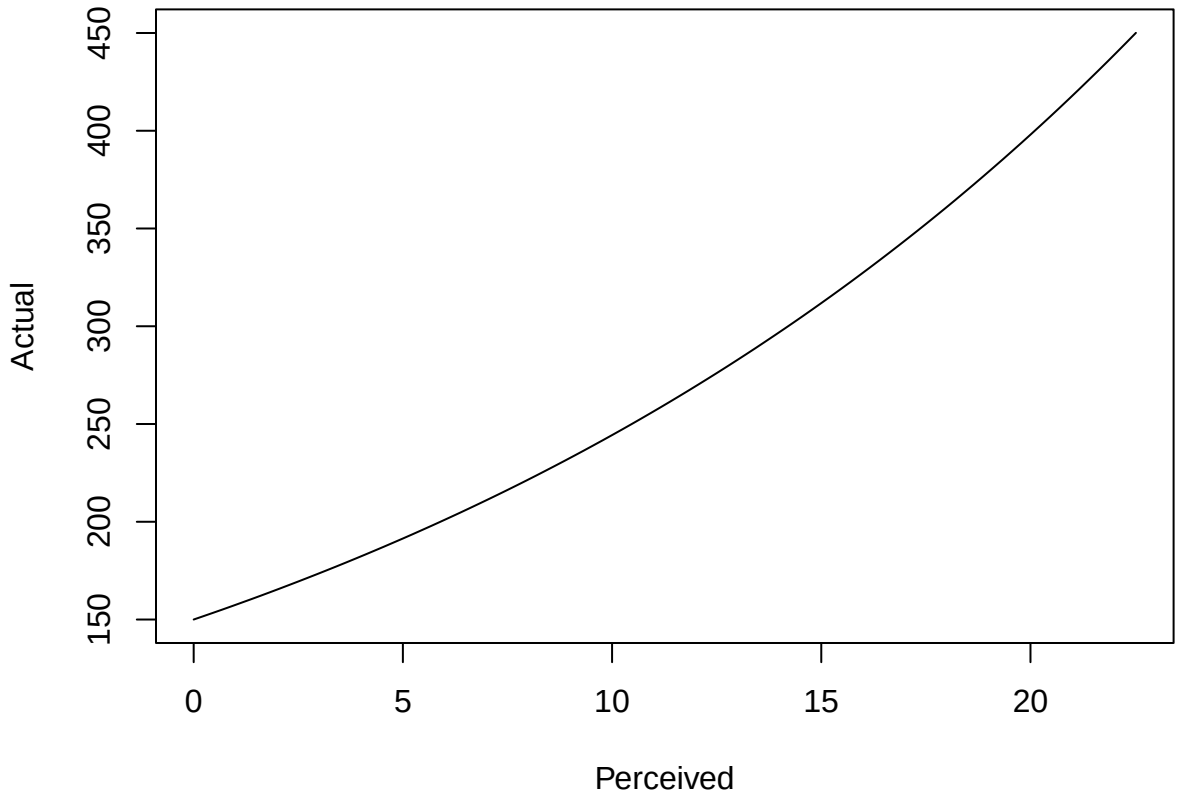
The extent to which the scale has been stretched (or squished) by the transformation is the slope of the transformation function, 10 for the example above:

```
sam=rnorm(10000)
par(mfcol=c(1,1),mar=c(5,5,1,1))
plot(sam,sam*10,main="")
abline(0,10,col="red",lwd=2)
```



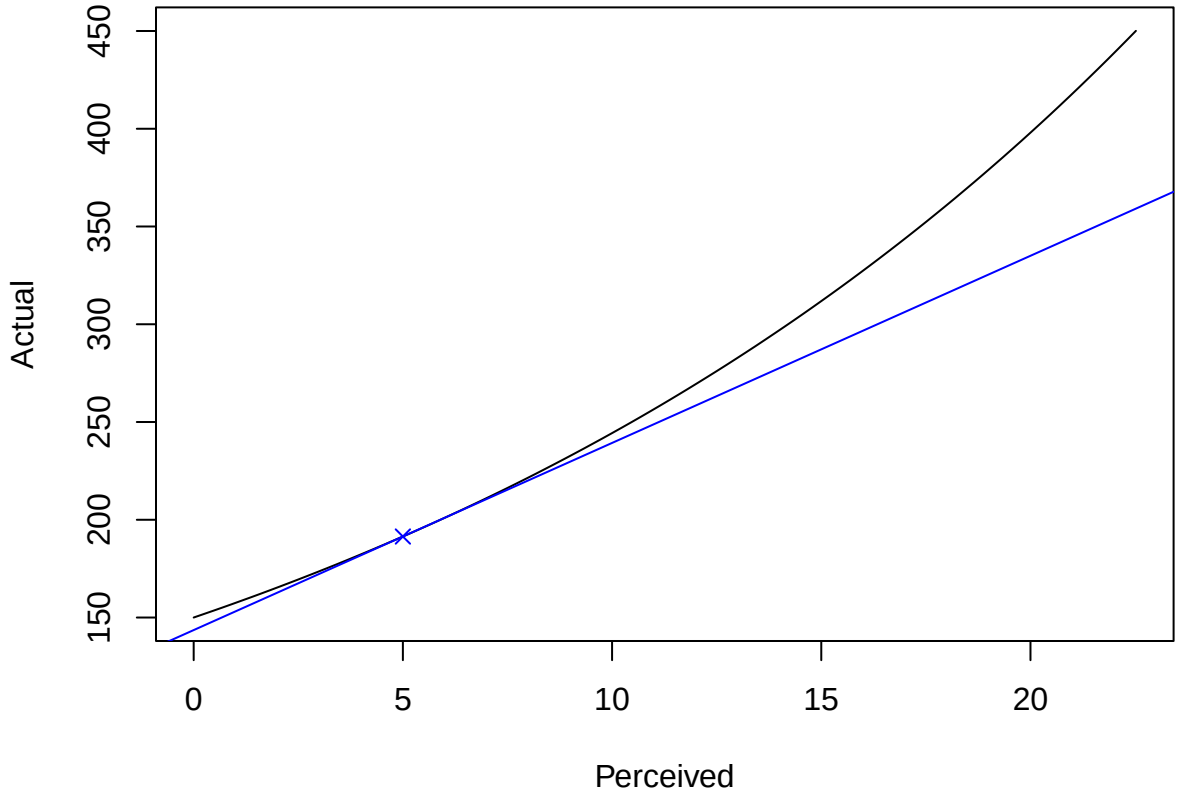
Here's the transformation function we are applying to the perceived lengths (as in Andrew's file):

```
par(mfcol=c(1,1),mar=c(5,5,1,1))
xSeq=seq(0,22.52,length.out = 100)
plot(xSeq,psy2phy(xSeq),main="",
     type='l',xlab="Perceived",
     ylab="Actual")
```



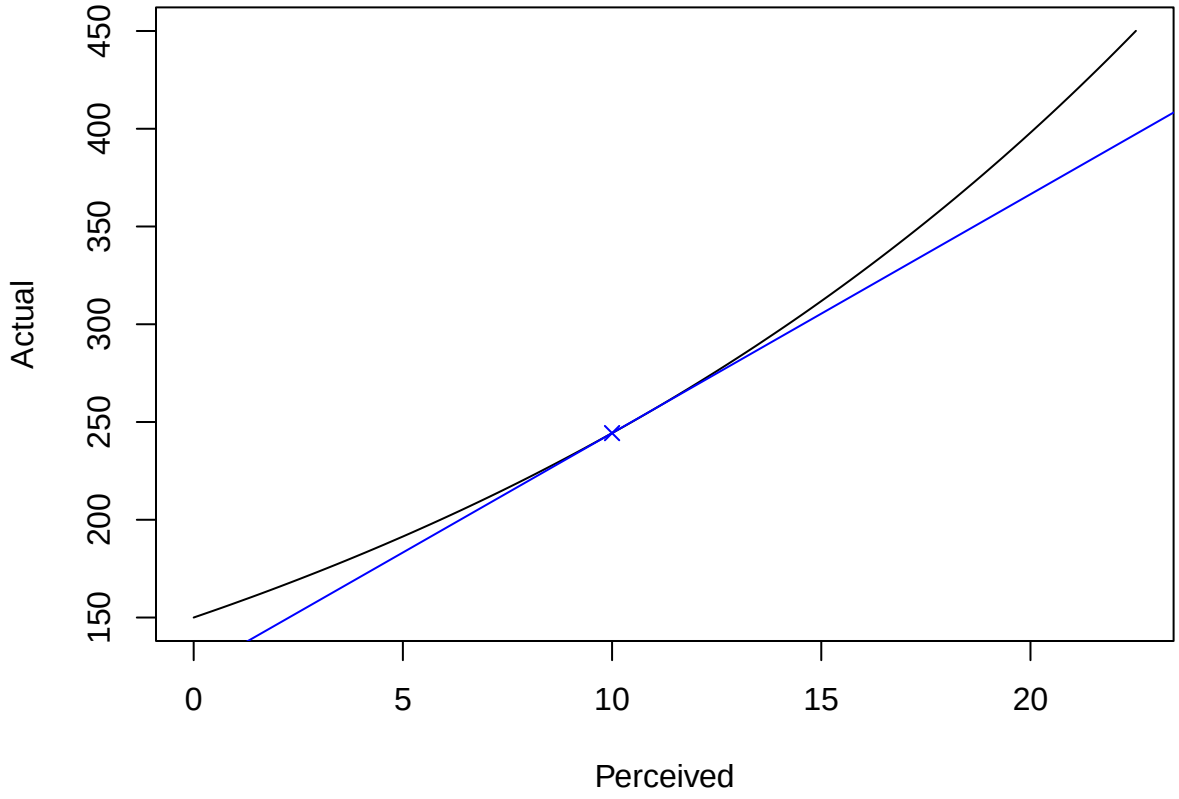
Our conversion function has an exponent, so it stretches the scale to a different extent for every value that you pass it. The amount of stretching for a given value is the derivative of the transformation function. For a given actual length (a), the derivative of the function is .05a (I think). Trying some examples, at 5 JNDs above the shortest line, the actual value is psy2phy(5) = 191.44 for a stretch factor of psy2phy(5)*.05 = 9.57.

```
par(mfcol=c(1,1),mar=c(5,5,1,1))
xSeq=seq(0,22.52,length.out = 100)
plot(xSeq,psy2phy(xSeq),main="",
     type='l',xlab="Perceived",
     ylab="Actual")
slp=psy2phy(5)*.05
int=psy2phy(5)-5*slp
abline(int,slp,col="blue")
points(5,psy2phy(5),pch=4,col="blue")
```



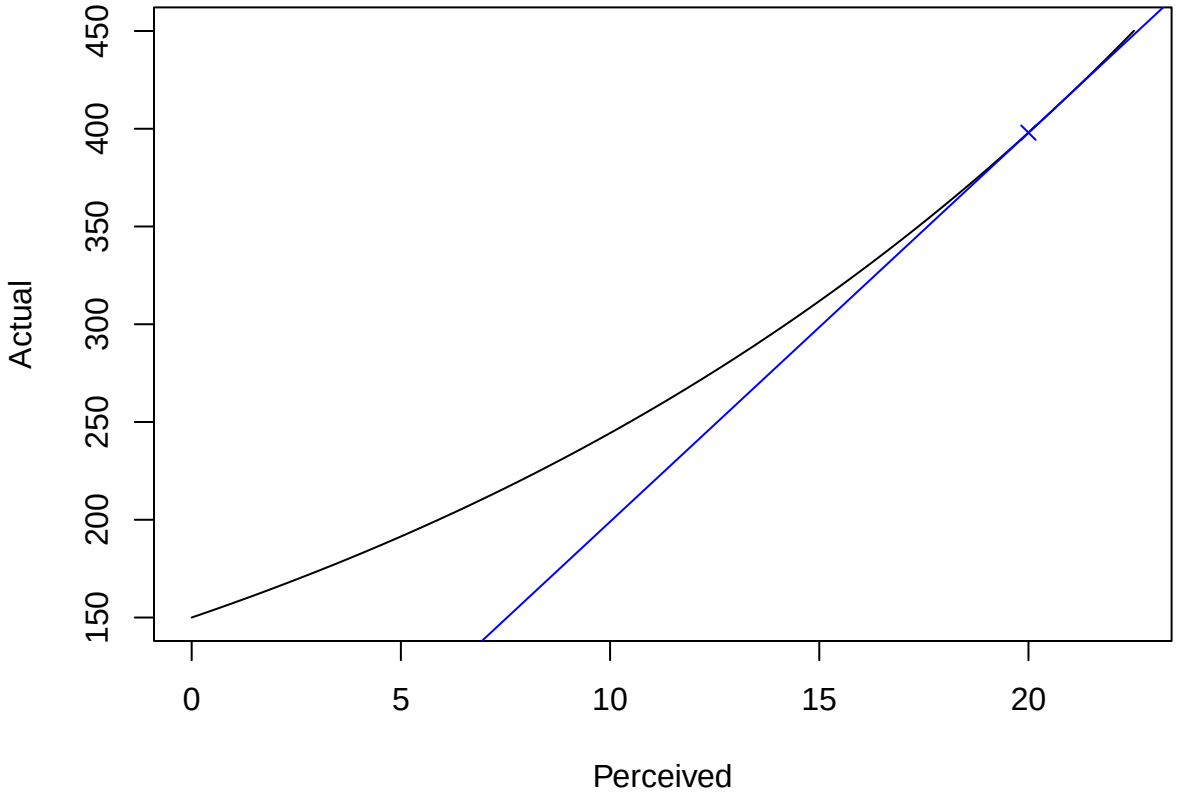
At 10 JNDs above the shortest line, the actual value is psy2phy(10) = 244.33 for a stretch factor of psy2phy(10)*.05 = 12.22.

```
par(mfcol=c(1,1),mar=c(5,5,1,1))
xSeq=seq(0,22.52,length.out = 100)
plot(xSeq,psy2phy(xSeq),main="",
     type='l',xlab="Perceived",
     ylab="Actual")
slp=psy2phy(10)*.05
int=psy2phy(10)-10*slp
abline(int,slp,col="blue")
points(10,psy2phy(10),pch=4,col="blue")
```



At 20 JNDs above the shortest line, the actual value is psy2phy(20) = 397.99 for a stretch factor of psy2phy(20)*.05 = 19.90.

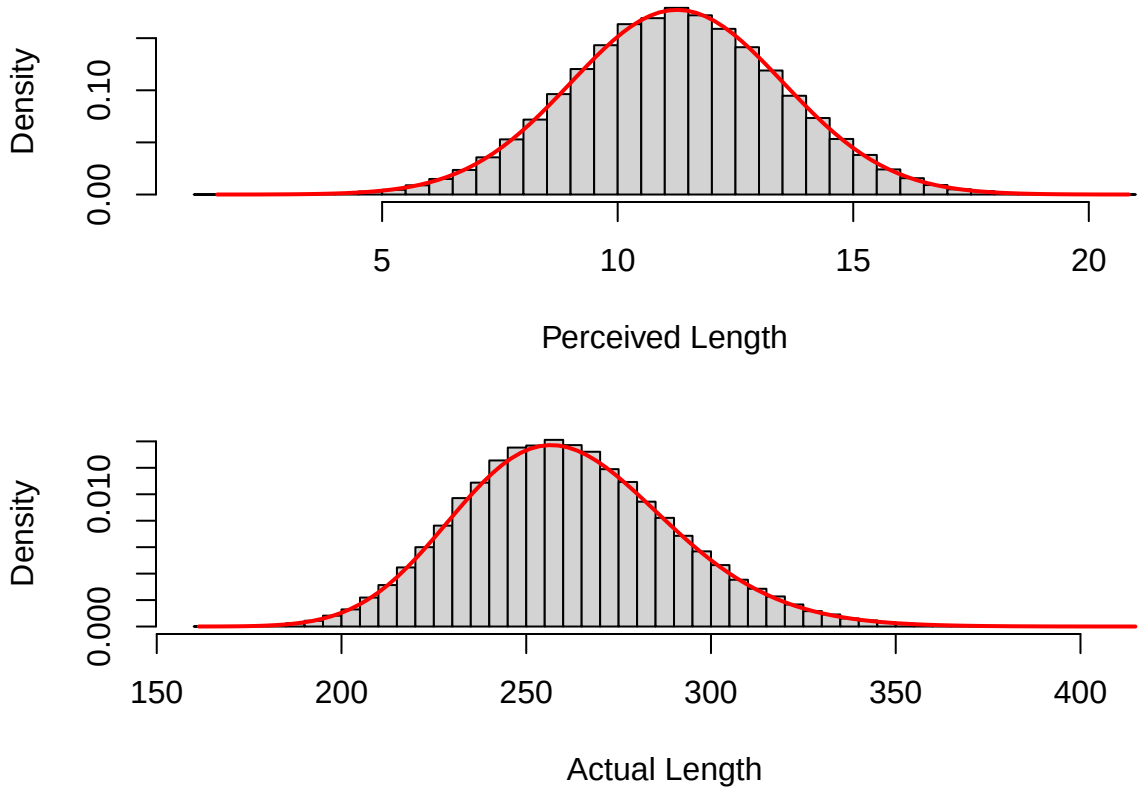
```
par(mfcol=c(1,1),mar=c(5,5,1,1))
xSeq=seq(0,22.52,length.out = 100)
plot(xSeq,psy2phy(xSeq),main="",
     type='l',xlab="Perceived",
     ylab="Actual")
slp=psy2phy(20)*.05
int=psy2phy(20)-20*slp
abline(int,slp,col="blue")
points(20,psy2phy(20),pch=4,col="blue")
```



So to get the density for a particular value of actual length (a) on the transformed distribution, we need to take the density of the corresponding point on the perceived distribution, phy2psy(a), and divide by the stretch factor at that point on the transformation function, a*.05. The dActMem() and dActNon() functions below calculate this density for category members and non-members. dActMem() is a log normal distribution. The only reason we can't just use the canned log normal distribution in R is that is assumes base e (Euler's number) instead of the 1.05 base for our transformation function. Comparing to random samples suggests that these functions work properly.

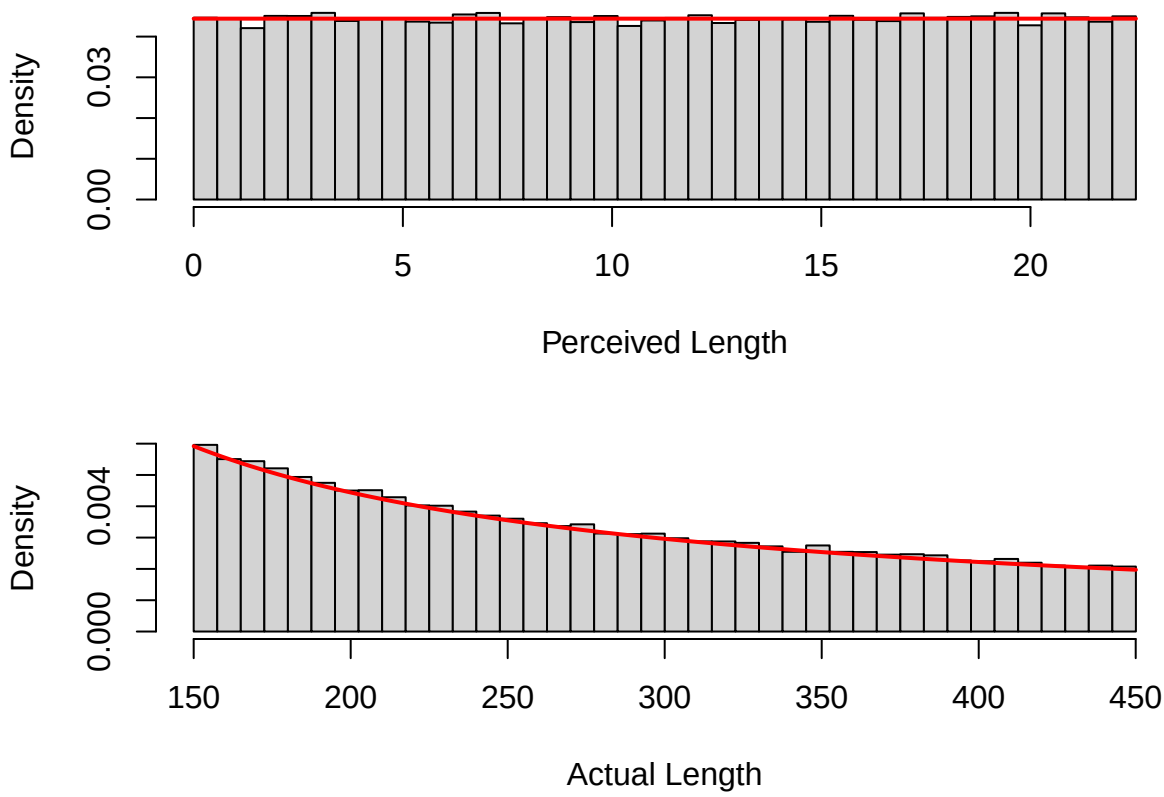
For category members:

```
par(mfcol=c(2,1),mar=c(5,5,1,1))

#Perception
dPerMem=function(p) dnorm(p,11.26,2.252)
hist(pMem,freq=F,40,
       main="",xlab="Perceived Length")
xSeq=seq(min(pMem),max(pMem),length.out=100)
points(xSeq,dPerMem(xSeq),type='l',col='red',lwd=2)

#Actual
dActMem=function(a) dnorm(phy2psy(a),11.26,2.252)/(.05*a)

hist(aMem,freq=F,40,
       main="",xlab="Actual Length")
xSeq=seq(min(aMem),max(aMem),length.out=100)
points(xSeq,dActMem(xSeq),type='l',col='red',lwd=2)
```



For category non-members:

```
par(mfcol=c(2,1),mar=c(5,5,1,1))

#Perception
dPerNon=function(p) dunif(p,0,22.52)
bounds=seq(0,22.52,length.out = 41)
hist(pNon,freq=F,breaks=bounds,
       main="",xlab="Perceived Length")
xSeq=seq(min(pNon),max(pNon),length.out=100)
points(xSeq,dPerNon(xSeq),type='l',col='red',lwd=2)

#Actual
dActNon=function(a) dunif(phy2psy(a),0,22.52)/(.05*a)

bounds=seq(150,max(aNon),length.out = 41)
hist(aNon,freq=F,breaks=bounds,
       main="",xlab="Actual Length")
xSeq=seq(min(aNon),max(aNon),length.out=100)
points(xSeq,dActNon(xSeq),type='l',col='red',lwd=2)
```


Density
0.00
0.03
0
5
10
15
20
Perceived Length
Density
0.000
0.004
150
200
250
300
350
400
450
Actual Length