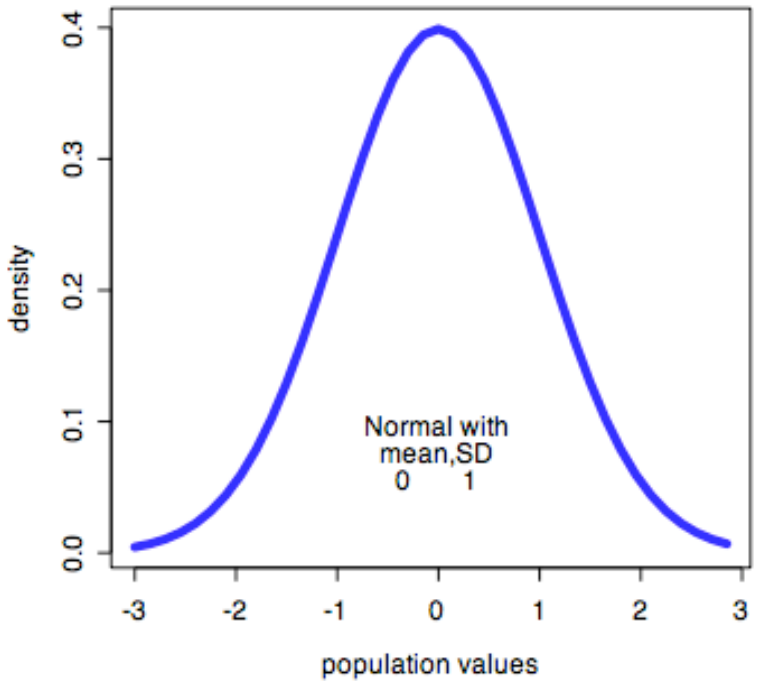
This file is a demonstration that CIs for continuous data have the property they are supposed to have.

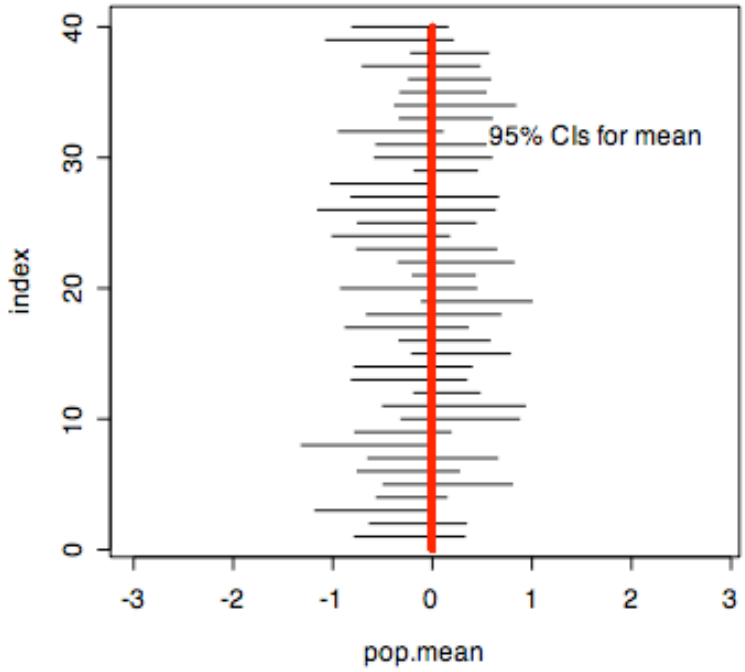
A program similar to the one for sample proportions can produce a picture of multiple Confidence Intervals for a continuous random variable. Note that the multiple CIs is not something one sees in practice – it is just to try to make the theory more transparent. When we have a data set with, say, n=25 values, we get only one 95% CI for the population mean, and the property it has is that CIs computed in this way will include (in the CI) the true population mean 95% of the time we take such a sample of n=25. The program just shows that the claim is true, even though when we actually use the technique, we will not know for sure if the one CI we compute does include the population mean, or not. All we know is that it usually will (95% of the time).

First lets assume the population is N(0,1) and we will use an R simulation to produce samples of size n=15. If we compute a 95% CI for the population mean, it should usually include the value 0 (the population mean) – in fact it should do it 95% of the time. We will use the situation I call Case 2 – we assume the population is normal but we imagine that we do not know the standard deviation and, of course, do not know the population mean that are trying to estimate. Note that n=15 is a smaller sample size than would allow us to use the CLT – that is why we have case 2 rather than case 3 (large sample). The multiple the program uses in this case is from Table A.8 = 2.145.

Jump to the figures on the next page – the first graph is the density of the N(0,1) distribution.



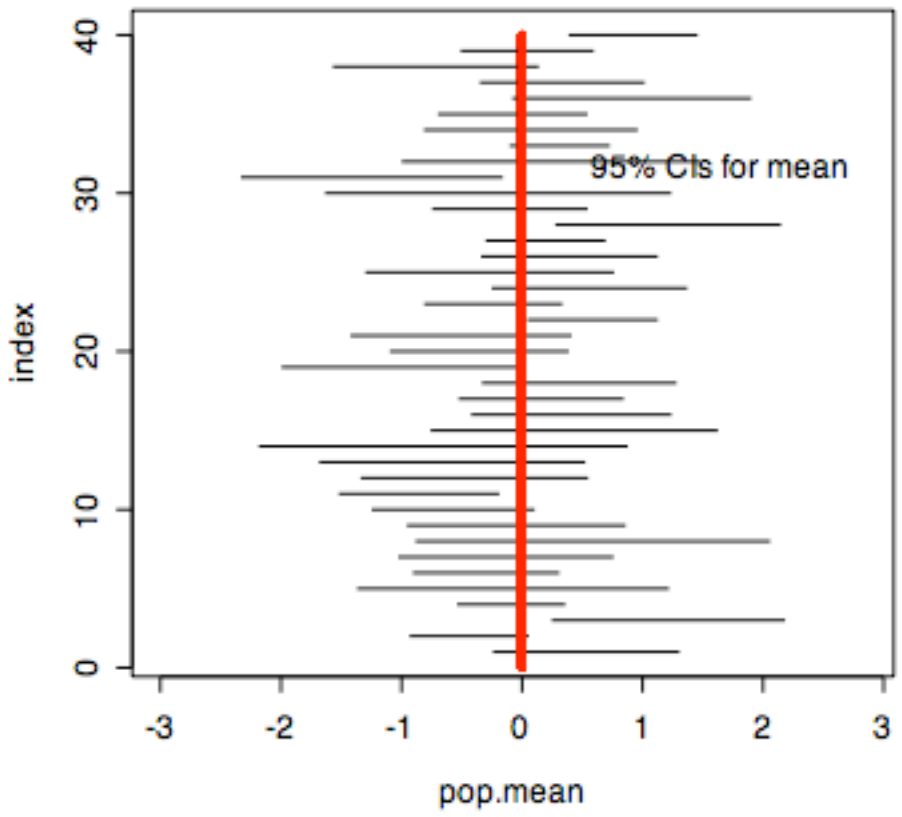
and here is a simulated set of 40 95% CIs. from the above population based on n=15 samples.



In this case ALL the CIs do include the pop. mean, but this was just good luck – the average would be 38 out of 40 (95%).

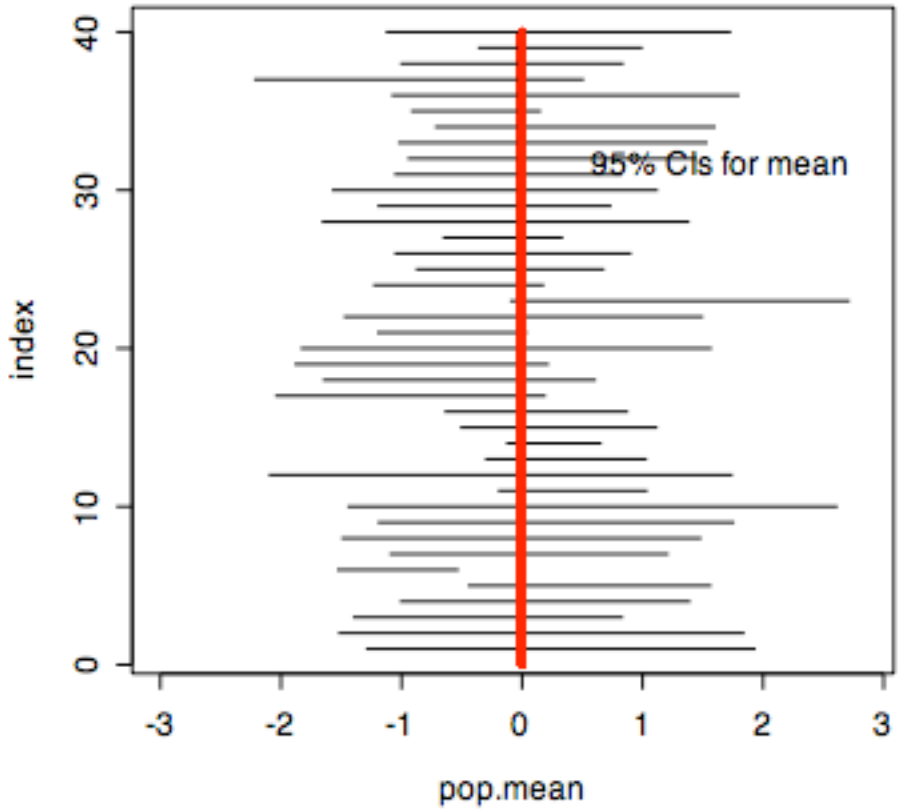
Note how narrow the CI relative to the population itself. This is because the CI is estiamting the location of the population mean, and this is a lot easier than estimating the range of the sample data. The SD of the sample data is $\sqrt{15}$ times larger than the SD of the sample mean, and it is the sample mean that is estimating the population mean, so the CI need not be as wide spread as the sample data.

Now what would happen if we used the large sample procedure for a 95% CI – that is use the Case 3 procedure when we should be using Case 2. The only difference is that the t-multiple replaces the normal table multiple. To exaggerate the effect we will use sample size n=5.



You can check that we only have 33 CIs that include the true mean. This is because we are using a multiple of 1.96 instead of 2.78 (the t multiple for 4 df). We need to allow for the variability in the sd estimate, and the the t multiple does this for us.

The next page shows the simulation result when we use the correct method – we have 39/40 CIs that include the pop mean in this instance.



Here is the program: - its not pretty or efficient, but it does the job!

```
> CIx
function (m=40,n=25,mean=0,sd=1,approx=F)
{
        x=matrix(ncol=n,nrow=m)
        sdv=1:m
        meanv=1:m
        x.up=matrix(ncol=1,nrow=m)
        x.down=matrix(ncol=1,nrow=m)
        index=1:m
        count=0
        for (i in 1:m){
        x[i,]=rnorm(n,mean,sd)
        sdv[i]=sd(x[i,])
        meanv[i]=mean(x[i,])
        if (approx==T) {multiple=2}
        if (approx==F) {multiple=qt(.975,(n-1))}
```

```
        x.up[i]=meanv[i]+multiple*sdv[i]/n^.5
        x.down[i]=meanv[i]-multiple*sdv[i]/n^.5
print(c(x.down[i],x.up[i]))
        if (is.between(x.down[i],mean,x.up[i])) {count=count+1}
        }
        temp=(-m/2):(m/2-1)
        pop.mean=mean+sd*(temp)/(m/6)
        plot(pop.mean,index,type="n")
        for (j in 1:m) {
                lines(c(x.down[j],x.up[j]),c(index[j],index[j]))
                lines(c(mean,mean),c(0,m),lw=3,col="red")
                }
                text(mean+1.65*sd,.8*m,"95% CIs for mean")
                print("number of intervals containing true mean=")
        print(count)
        print("out of")
        print(m)
        quartz()
        a=dnorm(pop.mean,mean=mean,sd=sd)
        plot(pop.mean,a,xlab="population values",ylab="density",col="blue", lwd=4,
type="l")
        text(mean,sd^(-1)/10,"Normal with")
        text(mean,sd^(-1)/10-.02*sd^(-1),"mean,SD")
        text(mean-sd/3,sd^(-1)/10-.04*sd^(-1),mean)
        text(mean+sd/3,sd^(-1)/10-.04*sd^(-1),sd)
        pause()
        graphics.off()
}
```