Continuous CDFs and ANOVA with NNS

Fred Viole

1 Introduction

The following document provides a hands-on tutorial on discrete and continuous CDFs using partial moments and an extension to ANOVA analysis.

1.1 Install NNS

We need the latest version of NNS available on GitHub.

```
require(devtools); install_github('OVVO-Financial/NNS',ref = "NNS-Beta-Version")
require(NNS)
```

2 CDFs

Cumulative distribution functions (CDFs) represent the probability a variable X will take a value less than or equal to x.

$$F_X(x) = P(\leq x)$$

2.1 Empirical CDF

The empirical CDF is a simple construct, provided in the base package of R. We can generate an empirical CDF with the `ecdf` function and create a function `(P)` to return the CDF of a given value of X.

```
set.seed(123);x=rnorm(100)
ecdf(x)
```

```
## Empirical CDF
## Call: ecdf(x)
##  x[1:100] = -2.3092, -1.9666, -1.6867,  ...,  2.169, 2.1873
```

```
P=ecdf(x)
P(0);P(1)
```

```
## [1] 0.48
```

```
## [1] 0.83
```

2.1.1 Lower Partial Moment CDF (`LPM.CDF`)

The empirical CDF and Lower Partial Moment CDF (`LPM.CDF`) are identical when the degree term of the `LPM.CDF` is set to zero.

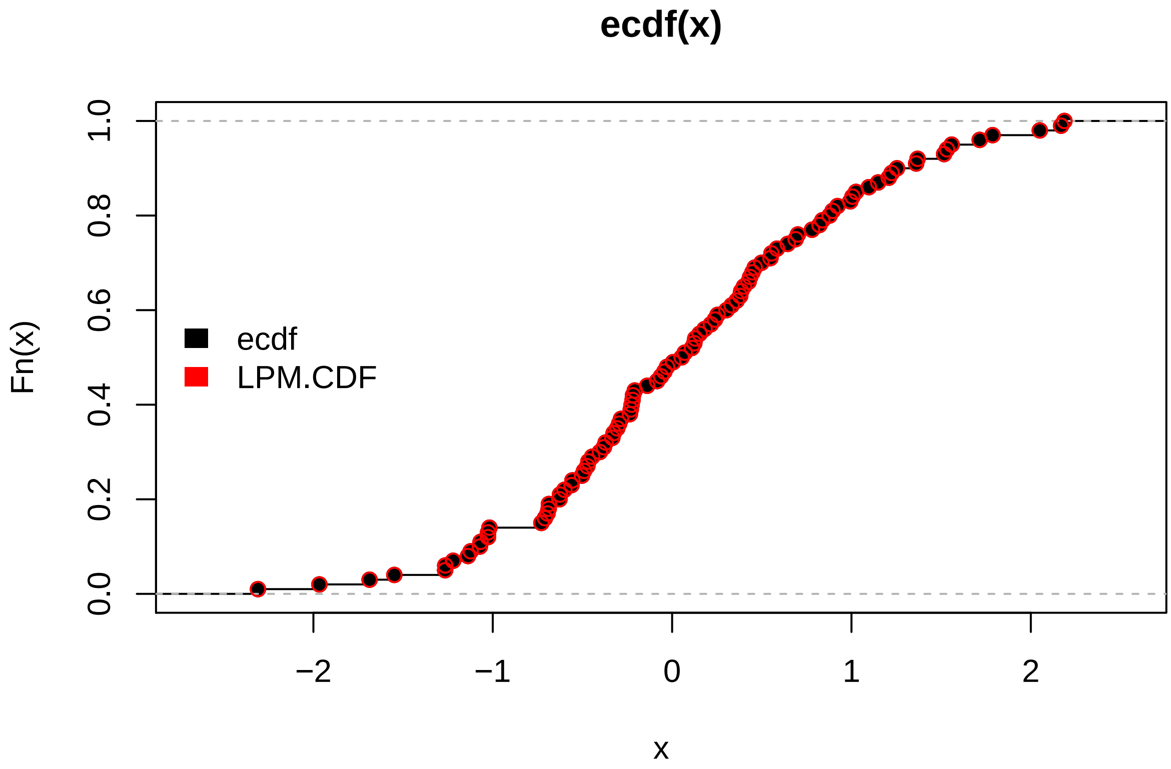
Using the same targets from our `ecdf` example above (0,1) we can compare `LPM.CDF`s.

```
LPM(0,0,x);LPM(0,1,x)
```

```
## [1] 0.48
## [1] 0.83
```

Calculating the probability for every value in X, we can plot both methods visualizing their identical results. `ecdf` function in black and `LPM.CDF` in red.

```r
plot(ecdf(x))
LPM.CDF=numeric()
for(i in 1:length(x)){LPM.CDF[i]=LPM(0,sort(x)[i],x)}
points(sort(x),LPM.CDF,col='red')
legend('left',legend = c('ecdf','LPM.CDF'),fill=c('black','red'),border=NA,bty='n')
```



2.1.2 Degree 1 LPM CDF (`LPM.1.CDF`)

We already noted how the `LPM.CDF` is equivalent to the empirical CDF when the LPM degree is set to zero. However, if we wish to consider the area below the point when generating a probability, set the degree equal to 1, for any target (t) creating `LPM.1.CDF`.

Degree 0 LPM:

$$LPM(0,t,X) = \frac{1}{N}\sum_{n=1}^{N}[max(t - X_n), 0]^0$$

Degree 1 LPM:

$$LPM(1,t,X) = \frac{1}{N}\sum_{n=1}^{N}[max(t - X_n), 0]^1$$

`LPM.CDF` is equivalent to the following form for any target (t) and variable X:

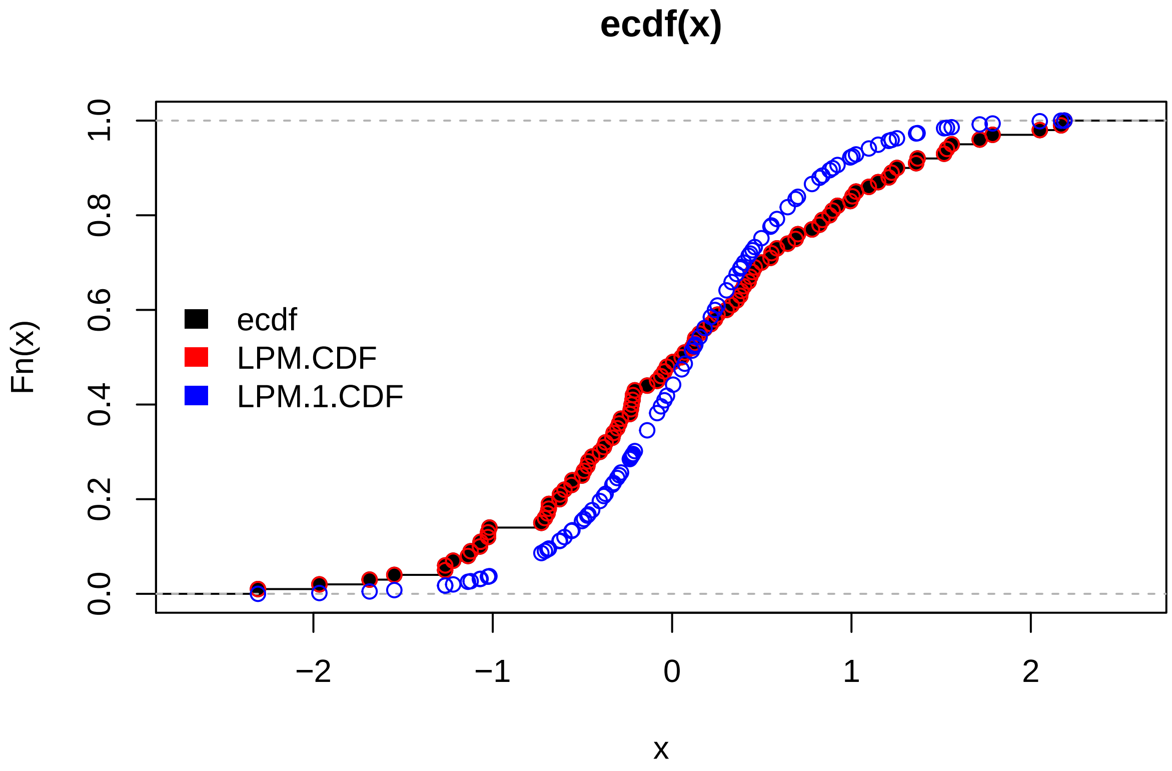
$$LPM(0,t,X) = \frac{LPM(0,t,X)}{LPM(0,t,X) + UPM(0,t,X)}$$

And `LPM.1.CDF` is equivalent to the following form for any target (t) and variable X:

$$LPM(1,t,X) = \frac{LPM(1,t,X)}{LPM(1,t,X) + UPM(1,t,X)}$$

Visualizing these CDFs together reveals some interesting properties.

```r
plot(ecdf(x))
LPM.CDF=numeric()
for(i in 1:length(x)){LPM.CDF[i]=LPM(0,sort(x)[i],x)}
points(sort(x),LPM.CDF,col='red')
LPM.1.CDF=numeric()
for(i in 1:length(x)){LPM.1.CDF[i]=LPM(1,sort(x)[i],x)/
(LPM(1,sort(x)[i],x)+UPM(1,sort(x)[i],x))}
points(sort(x),LPM.1.CDF,col='blue')
legend('left',legend = c('ecdf','LPM.CDF','LPM.1.CDF'),fill=c('black','red','blue'),
border=NA,bty='n')
```

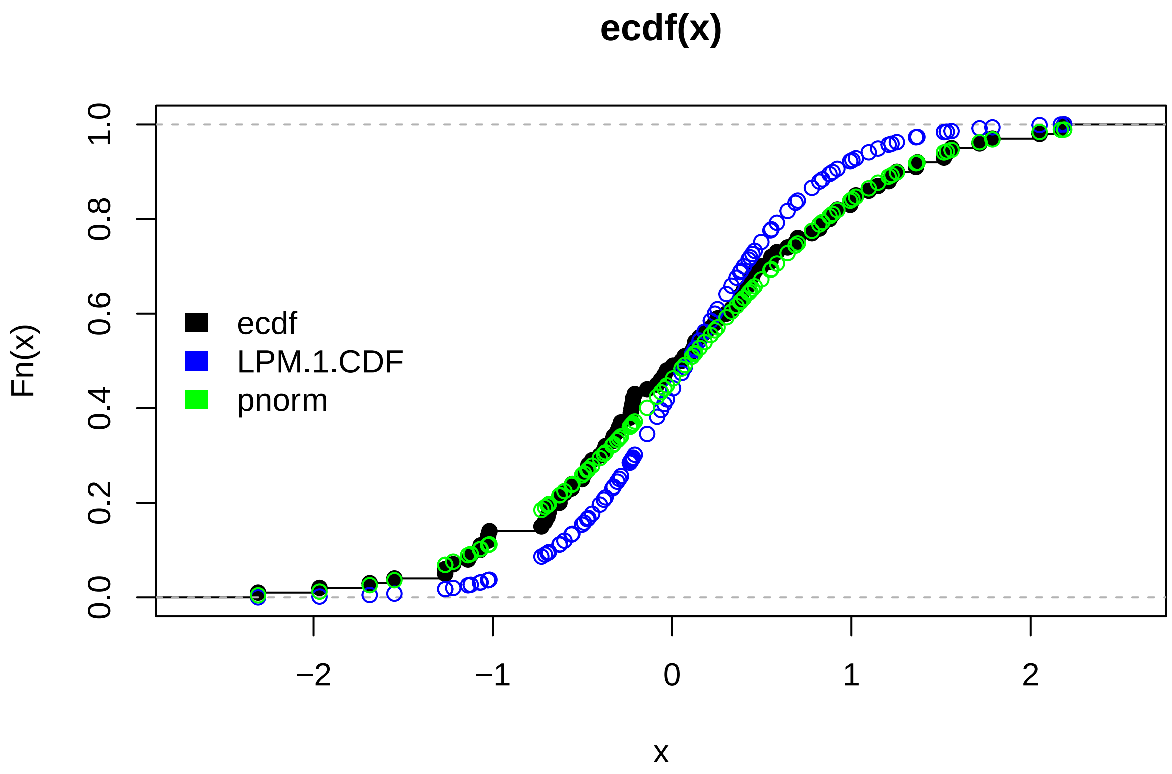


3 Comparing to Continuous CDF

R offers the `prnorm` function which returns the integral from -∞ to x where x is a Z-score.

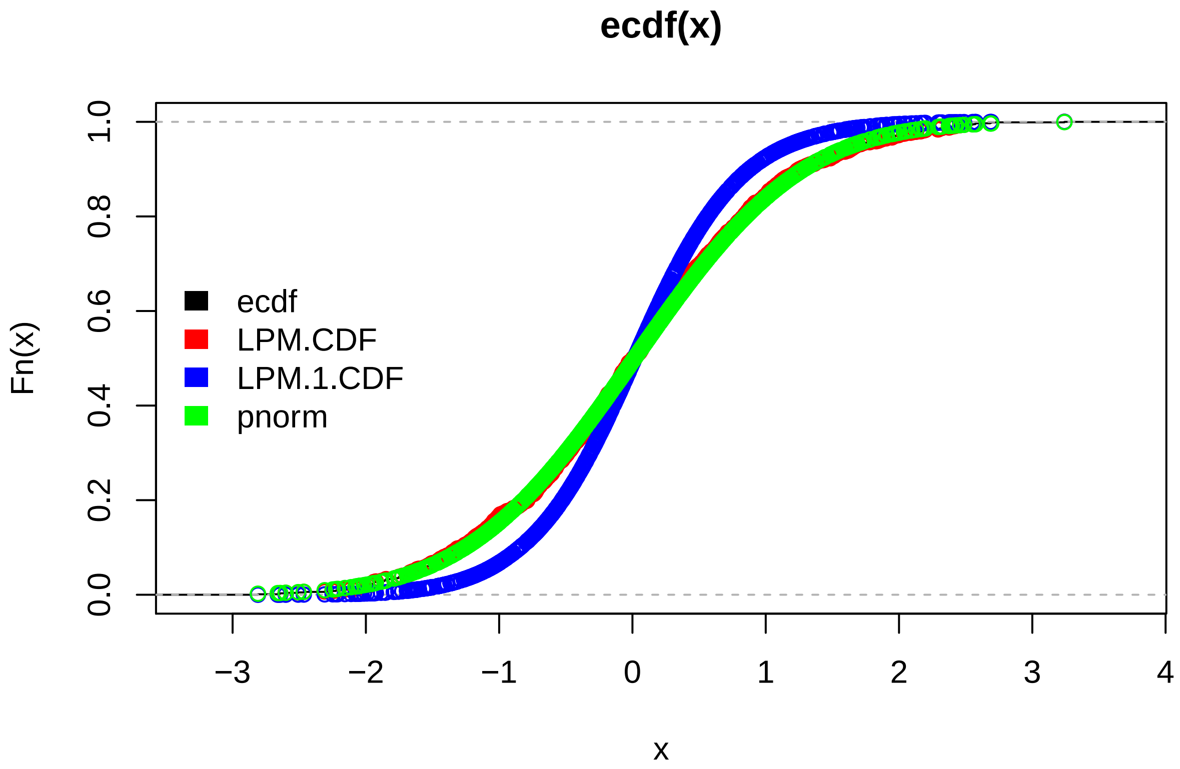
Adding this to our previous CDF estimates, we can see how `LPM.1.CDF` is more "S" shaped than the others.

```
plot(ecdf(x))
points(sort(x),LPM.1.CDF,col='blue')
points(sort(x),pnorm(sort(x),mean=mean(x),sd=sd(x)),col='green')
legend('left',legend = c('ecdf','LPM.1.CDF','pnorm'),fill=c('black','blue','green'),
border=NA,bty='n')
```



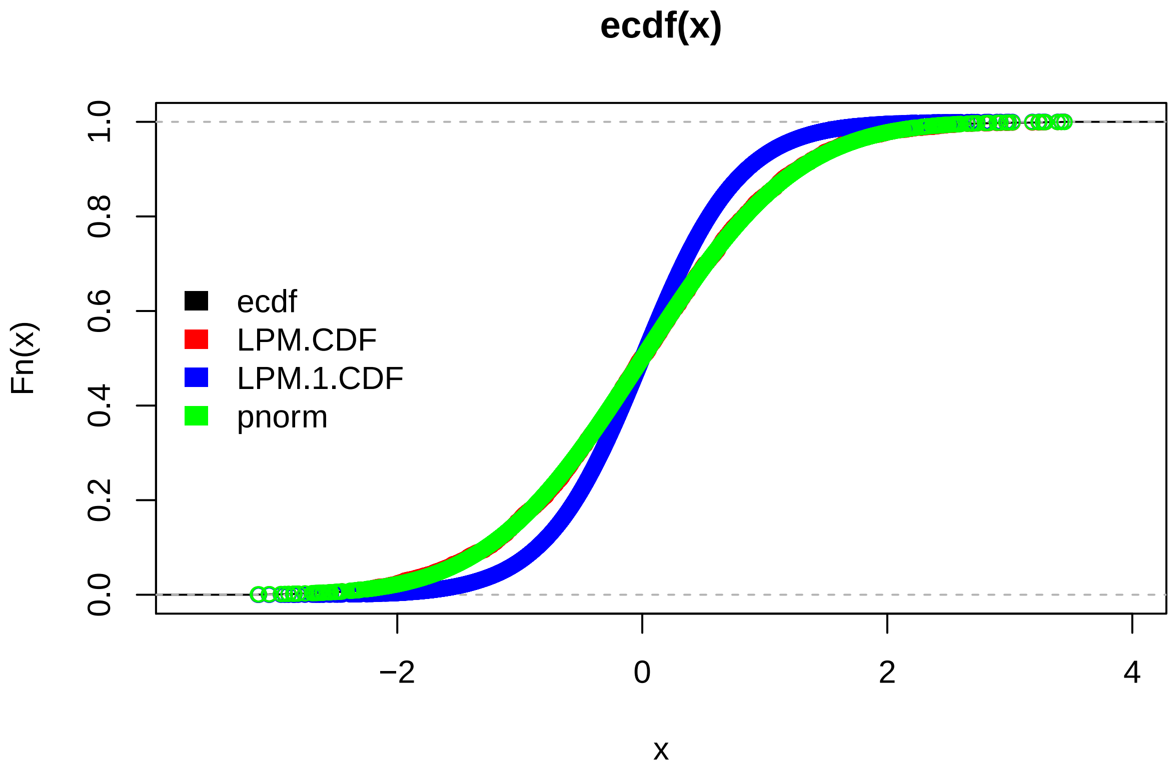
Extending the number of observations we can see how these functions behave. The `ecdf` and the `pnorm` are converging while `LPM.1.CDF` remains distinct:

```
set.seed(123);x=rnorm(1000)
plot(ecdf(x))
LPM.CDF=numeric()
for(i in 1:length(x)){LPM.CDF[i]=LPM(0,sort(x)[i],x)}
points(sort(x),LPM.CDF,col='red')
LPM.1.CDF=numeric()
for(i in 1:length(x)){LPM.1.CDF[i]=LPM.ratio(1,sort(x)[i],x)}
points(sort(x),LPM.1.CDF,col='blue')
points(sort(x),pnorm(sort(x),mean=mean(x),sd=sd(x)),col='green')
legend('left',legend = c('ecdf','LPM.CDF','LPM.1.CDF','pnorm'),fill=c('black','red',
'blue','green'),border=NA,bty='n')
```



Even more observations. . .

```
set.seed(123);x=rnorm(5000)
plot(ecdf(x))
LPM.CDF=numeric()
for(i in 1:length(x)){LPM.CDF[i]=LPM(0,sort(x)[i],x)}
points(sort(x),LPM.CDF,col='red')
LPM.1.CDF=numeric()
for(i in 1:length(x)){LPM.1.CDF[i]=LPM.ratio(1,sort(x)[i],x)}
points(sort(x),LPM.1.CDF,col='blue')
points(sort(x),pnorm(sort(x),mean=mean(x),sd=sd(x)),col='green')
legend('left',legend = c('ecdf','LPM.CDF','LPM.1.CDF','pnorm'),fill=c('black','red',
'blue','green'),border=NA,bty='n')
```



4 Properties

4.1 Smoothness

One property that immediately presents itself is the smoothness of `LPM.1.CDF`. Remember, this too is an empirical CDF, however, it is an area based probability and these area considerations are directly responsible for its distinct properties.

4.2 Mean Target

For every observation of every type of distribution, `LPM.1.CDF` is equal to 0.5 when the mean is used as a target.

$$\frac{LPM(1,\mu,X)}{LPM(1,\mu,X)+UPM(1,\mu,X)}=0.5$$

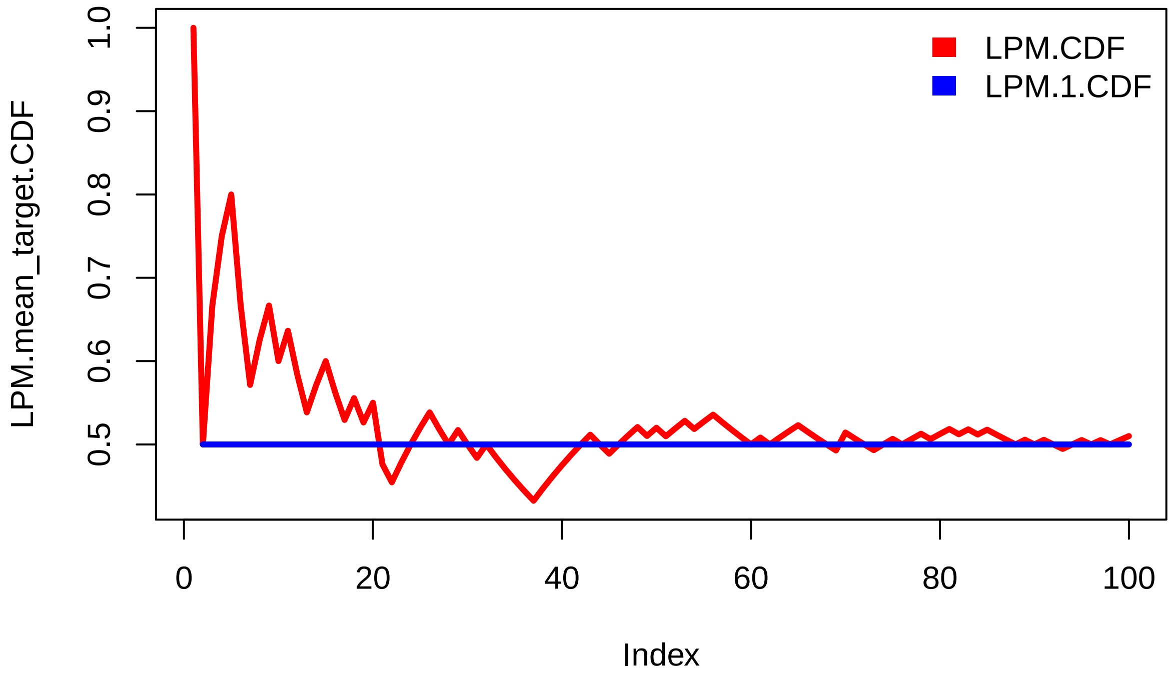
This bears repeating, "***For every observation of every type of distribution***". This is a very special property that can be used to compare means of distributions without underlying assumptions by using their CDFs, and serves as the basis of the `NNS.ANOVA` method.

5 Distribution Examples

The following examples highlight the LPM.1.CDF stability from a mean target.

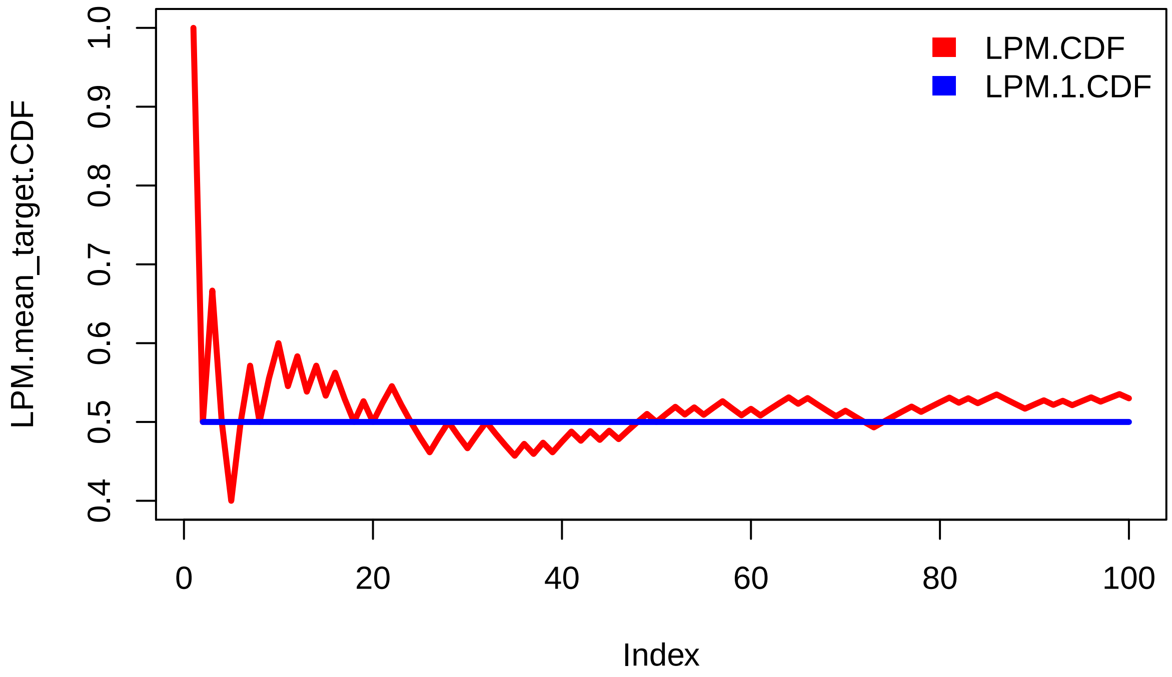
5.1 Normal Distribution

```r
set.seed(123);x=rnorm(100)
LPM.mean_target.CDF=numeric();LPM.1.mean_target.CDF=numeric()
for(i in 1:length(x)){
LPM.mean_target.CDF[i]=LPM(0,mean(x[1:i]),x[1:i]);
LPM.1.mean_target.CDF[i]=LPM.ratio(1,mean(x[1:i]),x[1:i])}
plot(LPM.mean_target.CDF,col='red',type = 'l',lwd=3)
lines((1:100),LPM.1.mean_target.CDF,col='blue',lwd=3)
legend('topright',legend = c('LPM.CDF','LPM.1.CDF'),fill=c('red','blue'),
border=NA,bty='n')
```



5.2 Uniform Distribution

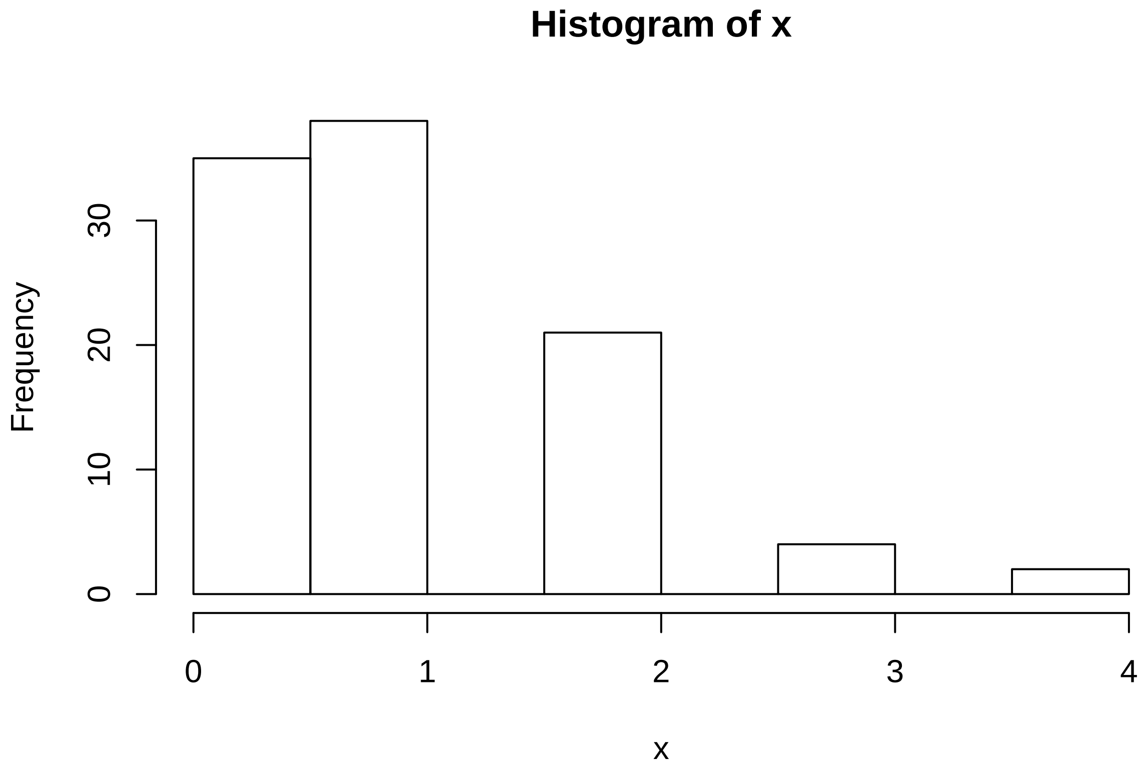
```
set.seed(123);x=runif(100)
LPM.mean_target.CDF=numeric();LPM.1.mean_target.CDF=numeric()
for(i in 1:length(x)){
LPM.mean_target.CDF[i]=LPM(0,mean(x[1:i]),x[1:i]);
LPM.1.mean_target.CDF[i]=LPM.ratio(1,mean(x[1:i]),x[1:i])}
plot(LPM.mean_target.CDF,col='red',type = 'l',lwd=3)
lines((1:100),LPM.1.mean_target.CDF,col='blue',lwd=3)
legend('topright',legend = c('LPM.CDF','LPM.1.CDF'),fill=c('red','blue'),
border=NA,bty='n')
```



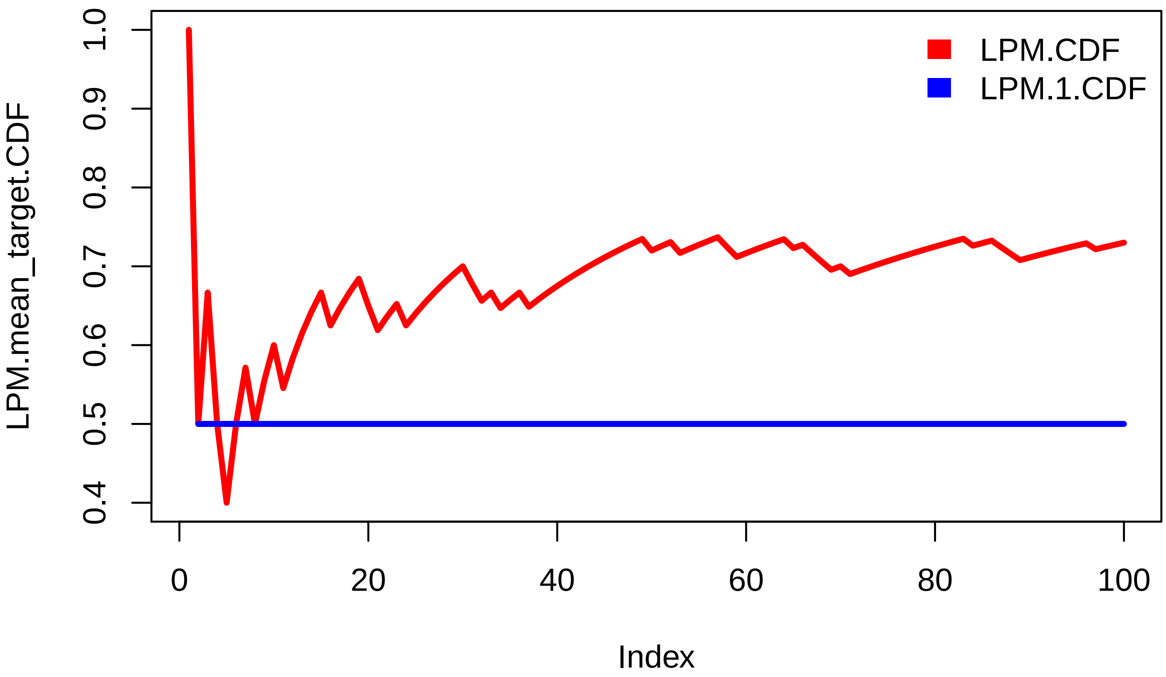
5.3 Poisson Distribution

This example demonstrates the non-association between frequency and area per the LPM CDFs from a mean target ($\lambda = 1$) in this distribution.

```r
set.seed(123);x=rpois(100,lambda = 1)
hist(x)
```



```r
set.seed(123);x=rpois(100,lambda = 1)
LPM.mean_target.CDF=numeric();LPM.1.mean_target.CDF=numeric()
for(i in 1:length(x)){
LPM.mean_target.CDF[i]=LPM(0,mean(x[1:i]),x[1:i]);
LPM.1.mean_target.CDF[i]=LPM.ratio(1,mean(x[1:i]),x[1:i])}
plot(LPM.mean_target.CDF,col='red',type = 'l',lwd=3)
lines((1:100),LPM.1.mean_target.CDF,col='blue',lwd=3)
legend('topright',legend = c('LPM.CDF','LPM.1.CDF'),fill=c('red','blue'),
border=NA,bty='n')
```


LPM.mean_target.CDF
1.0
0.9
0.8
0.7
0.6
0.5
0.4
0
20
40
60
80
100
Index
🟥 LPM.CDF
🟦 LPM.1.CDF

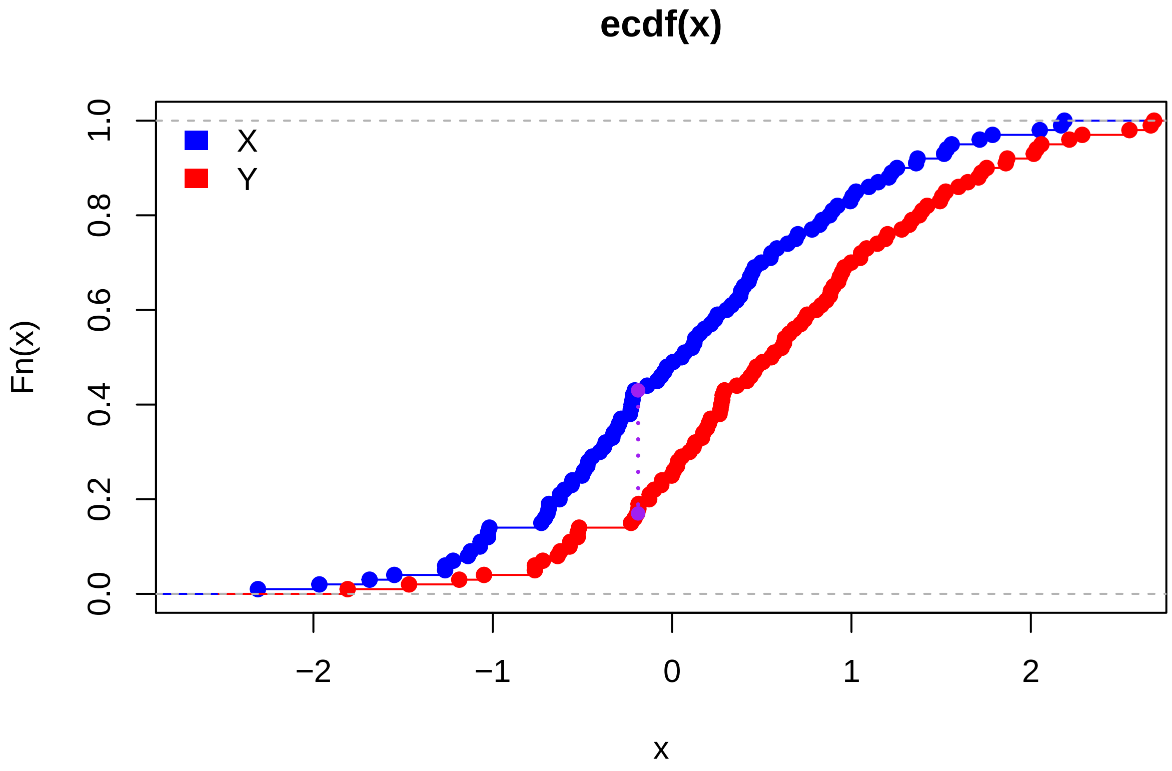
6 ANOVA Analysis

Using our previously derived information, we can effectively compare two means and ascertain a degree of certainty $[0, 1]$ whether the difference in means is zero via their CDFs. This avoids any and all distributional assumptions and requirements for the underlying data.

6.1 Kolmogorov-Smirnov (KS)

The KS statistic measures the maximum distance between the two empirical CDFs of two given samples in testing whether the samples were drawn from the same distribution. KS shown in purple below...

```r
set.seed(123);x=rnorm(100);y=x+.5
plot(ecdf(x),col='blue')
lines(ecdf(y),col='red')
legend('topleft',legend = c('X','Y'),fill=c('blue','red'),border=NA,bty='n')

points(c(x0, x0), c(y0, y1), pch=16, col="purple")
segments(x0, y0, x0, y1, col="purple", lty="dotted",lwd=2)
```



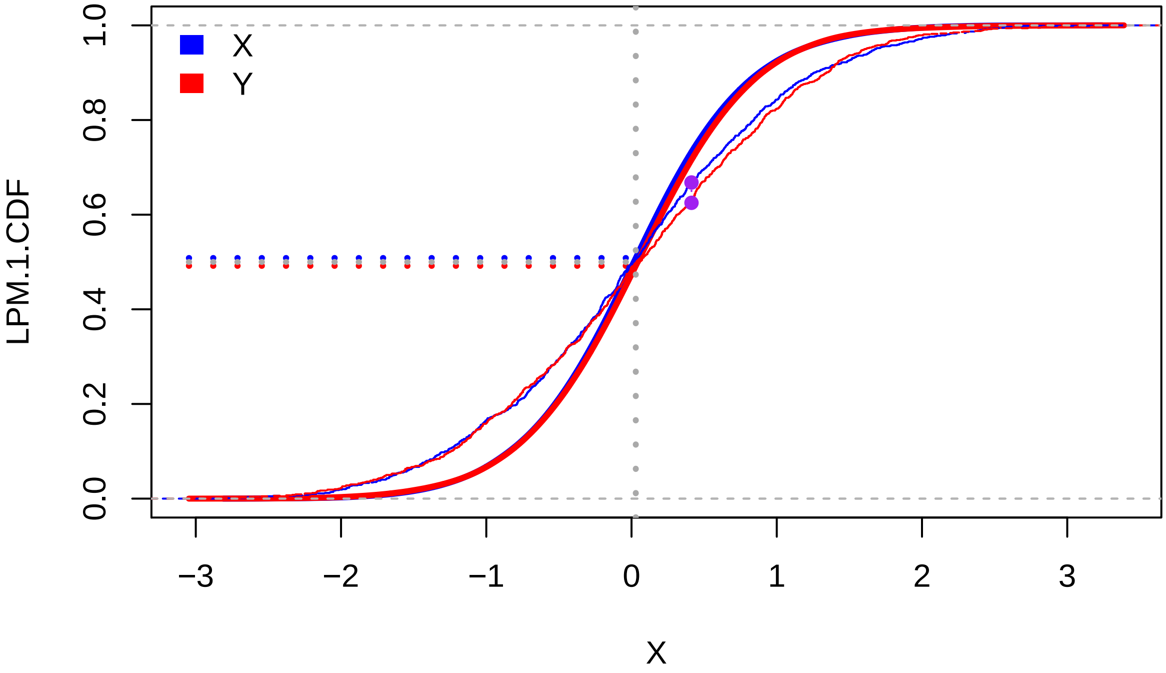
6.2 NNS ANOVA

NNS also analyzes the difference in CDFs of the two samples, but it is performed at the grand mean $(\frac{\mu_x+\mu_y}{2})$ between distributions, which can be generalized to multiple distributions. This specific grand mean point analysis coupled with the `LPM.1.CDF` is what distinguishes NNS from KS.

6.2.1 Same Population

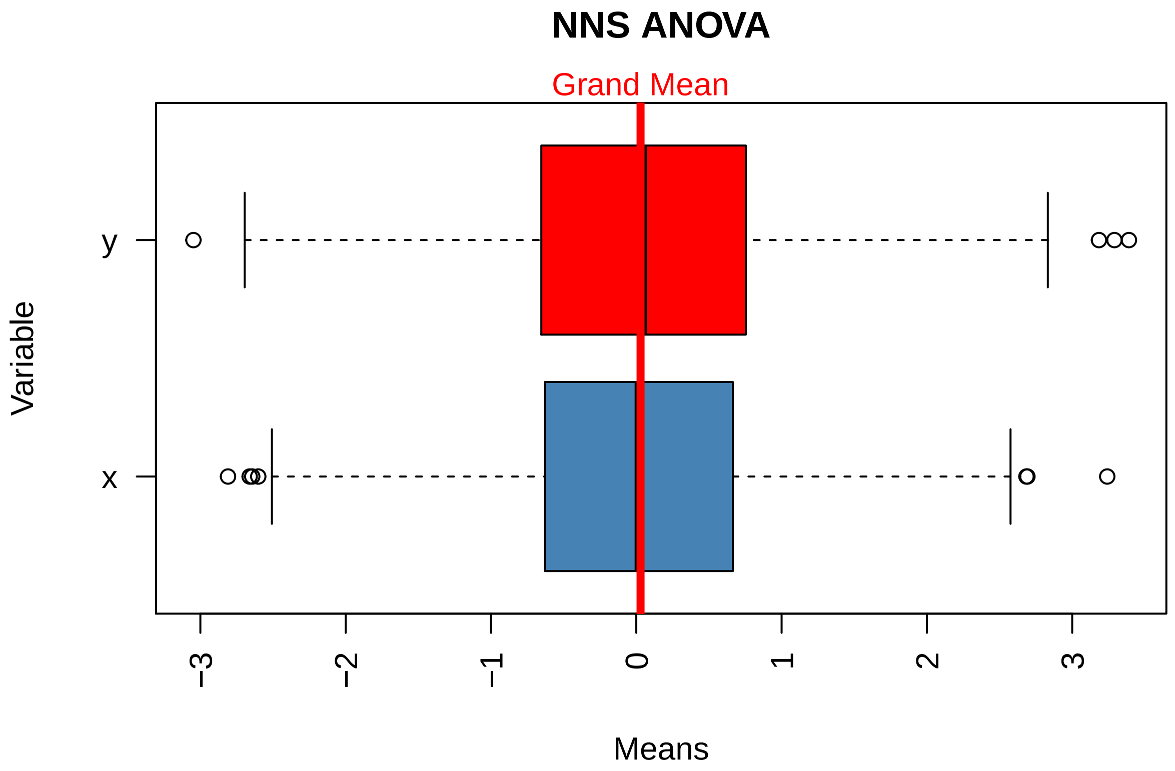
If both means were equal and the samples were from the same population, the difference between CDFs at the grand mean would be nil. ***We know unequivocally that the `LPM.1.CDF` will equal 0.5 from the mean of a given sample. Thus two samples with identical means will both have `LPM.1.CDF` equal to 0.5 if they are from the same population.*** The `LPM.1.CDF` lines are the thick blue and red lines (for X and Y respectively), while the ECDF lines for the KS test are the thin lines.

```r
set.seed(123);x=rnorm(1000);y=rnorm(1000)
LPM.1.CDF.x=numeric();LPM.1.CDF.y=numeric()
for(i in 1:length(x)){LPM.1.CDF.x[i]=LPM.ratio(1,sort(x)[i],x)
LPM.1.CDF.y[i]=LPM.ratio(1,sort(y)[i],y)}
plot(sort(x),LPM.1.CDF.x,col='blue',type='l',lwd=3,
     xlim=c(min(y),max(y)),xlab='X',ylab="LPM.1.CDF")
points(sort(y),LPM.1.CDF.y,col='red',type='l',lwd=3)
abline(v=mean(c(mean(x),mean(y))),col='darkgrey',lty=3,lwd=3)
target=mean(c(mean(x),mean(y)))
segments(x0=min(y),y0=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
         x1=target,y1=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
           col='red',lty=3,lwd=3)
segments(x0=min(y),y0=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
         x1=target,y1=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
           col='blue',lty=3,lwd=3)
segments(x0=min(y),y0=.5,
         x1=target,y1=.5,
           col='darkgrey',lty=3,lwd=3)

#KS line based off of ECDFs not LPM.1.CDF
lines(ecdf(x),lty=2,col='blue');lines(ecdf(y),lty=2,col='red')
cdf1 <- ecdf(x);cdf2 <- ecdf(y)
minMax <- seq(min(x, y), max(x, y), length.out=length(x))
x0<- minMax[which( abs(cdf1(minMax) - cdf2(minMax)) == max(abs(cdf1(minMax) - cdf2(minMax))) )]
y0 <- cdf1(x0)
y1 <- cdf2(x0)
points(c(x0, x0), c(y0, y1), pch=16, col="purple")
segments(x0, y0, x0, y1, col="purple", lty="dotted")
legend('topleft',legend = c('X','Y'),fill=c('blue','red'),border=NA,bty='n')
```



Comparing NNS CDF analysis (degree of certainty of the null hypothesis that the difference in means is zero) with a t-test (and its infamous p-value) and a ks.test:

```
NNS.ANOVA(cbind(x,y))
```



```
## Certainty
## 0.9672646
```

```
t.test(x,y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -0.5885, df = 1997.4, p-value = 0.5563
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.1141064  0.0614316
## sample estimates:
##  mean of x  mean of y
## 0.01612787 0.04246525
```

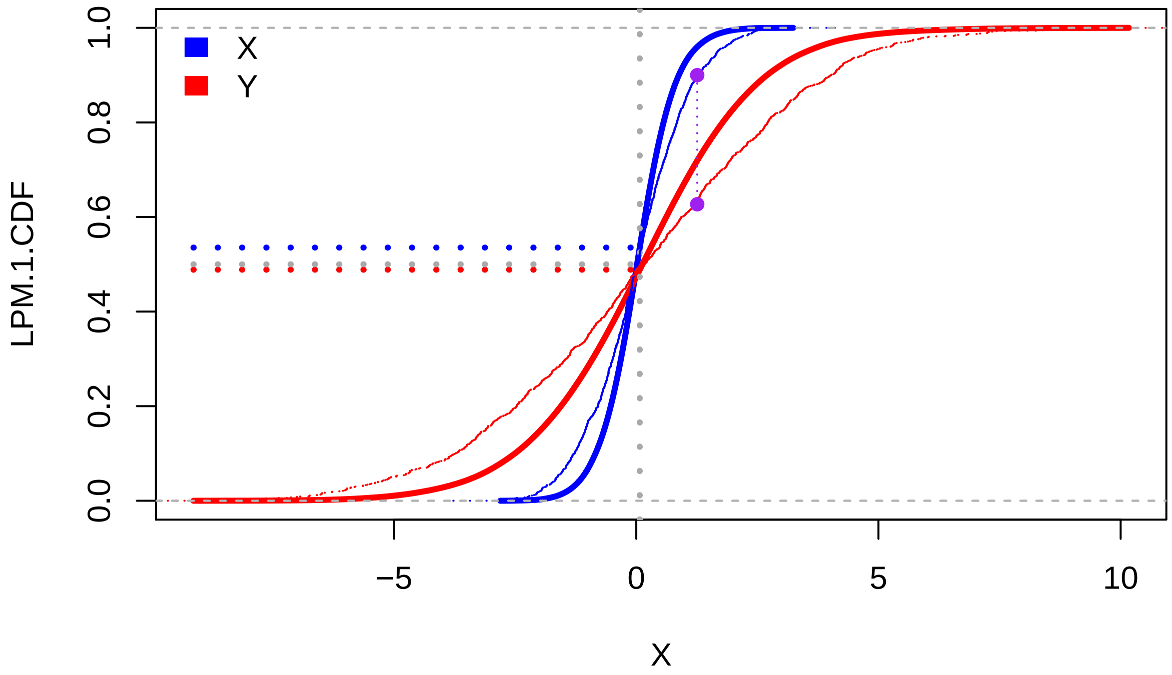
```
ks.test(x,y)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  x and y
## D = 0.043, p-value = 0.3136
## alternative hypothesis: two-sided
```

6.2.2 Different Populations

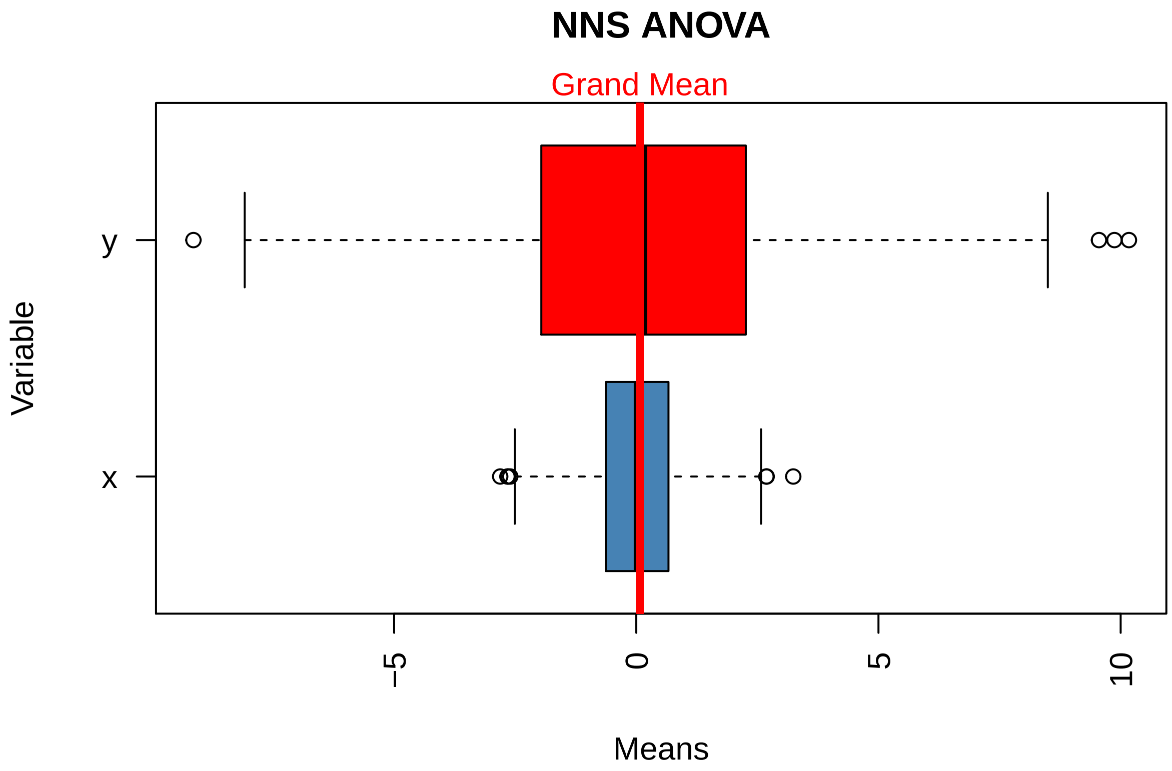
If both means were equal and the samples were from the different populations, the difference between CDFs at the grand mean would be evident.

```
set.seed(123);x=rnorm(1000);y=rnorm(1000,sd=3)
LPM.1.CDF.x=numeric();LPM.1.CDF.y=numeric()
for(i in 1:length(x)){LPM.1.CDF.x[i]=LPM.ratio(1,sort(x)[i],x)
LPM.1.CDF.y[i]=LPM.ratio(1,sort(y)[i],y)}
plot(sort(x),LPM.1.CDF.x,col='blue',type='l',lwd=3,
     xlim=c(min(y),max(y)),xlab='X',ylab="LPM.1.CDF")
points(sort(y),LPM.1.CDF.y,col='red',type='l',lwd=3)
abline(v=mean(c(mean(x),mean(y))),col='darkgrey',lty=3,lwd=3)
target=mean(c(mean(x),mean(y)))
segments(x0=min(y),y0=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
         x1=target,y1=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
           col='red',lty=3,lwd=3)
segments(x0=min(y),y0=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
         x1=target,y1=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
           col='blue',lty=3,lwd=3)
segments(x0=min(y),y0=.5,
         x1=target,y1=.5,
           col='darkgrey',lty=3,lwd=3)
#KS line
lines(ecdf(x),lty=3,col='blue');lines(ecdf(y),lty=3,col='red')
cdf1 <- ecdf(x);cdf2 <- ecdf(y)
minMax <- seq(min(x, y), max(x, y), length.out=length(x))
x0<- minMax[which( abs(cdf1(minMax) - cdf2(minMax)) == max(abs(cdf1(minMax) - cdf2(minMax))) )]
y0 <- cdf1(x0)
y1 <- cdf2(x0)
points(c(x0, x0), c(y0, y1), pch=16, col="purple")
segments(x0, y0, x0, y1, col="purple", lty="dotted")
legend('topleft',legend = c('X','Y'),fill=c('blue','red'),border=NA,bty='n')
```



Comparing NNS CDF analysis with a t-test and a KS-test:

```r
NNS.ANOVA(cbind(x,y))
```



```
## Certainty
## 0.9085942
```

```
t.test(x,y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -1.104, df = 1210.7, p-value = 0.2698
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.30900852  0.08647273
## sample estimates:
##  mean of x  mean of y
## 0.01612787 0.12739576
```

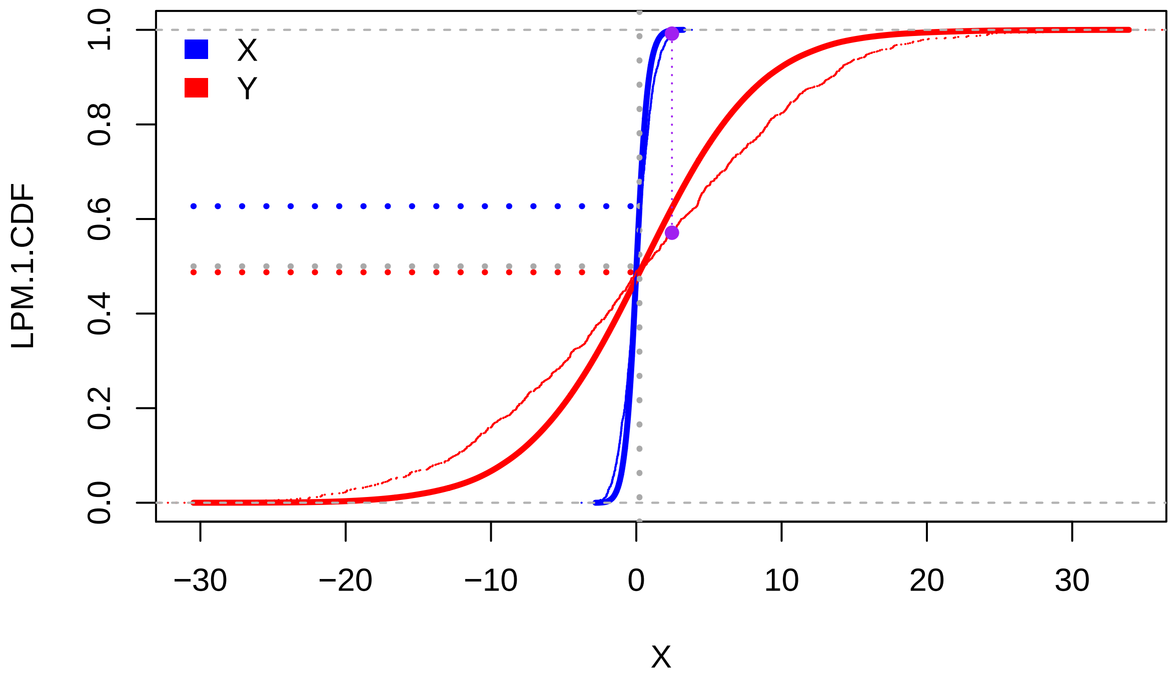
```
ks.test(x,y)
```

```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  x and y
## D = 0.273, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

6.2.3 VERY Different Populations

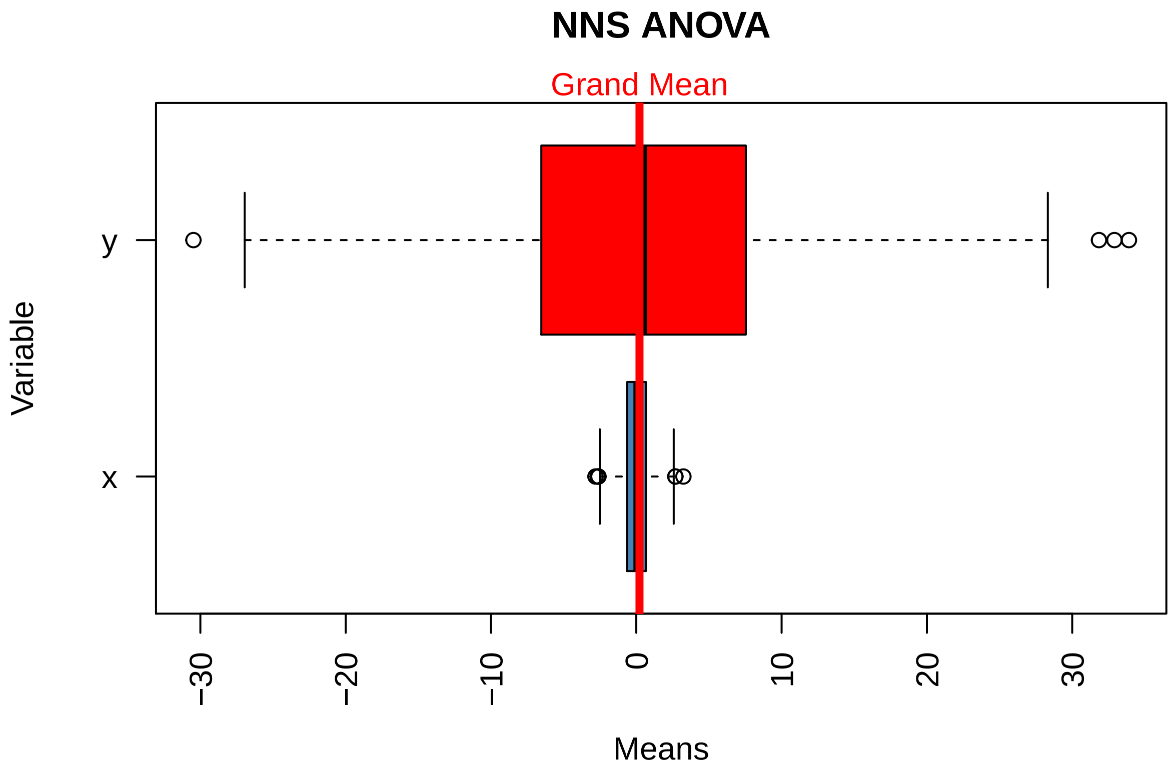
Note the difference between the NNS certainty vs. the t-test p-value…

```r
set.seed(123);x=rnorm(1000);y=rnorm(1000,sd=10)
LPM.1.CDF.x=numeric();LPM.1.CDF.y=numeric()
for(i in 1:length(x)){LPM.1.CDF.x[i]=LPM.ratio(1,sort(x)[i],x)
LPM.1.CDF.y[i]=LPM.ratio(1,sort(y)[i],y)}
plot(sort(x),LPM.1.CDF.x,col='blue',type='l',lwd=3,
     xlim=c(min(y),max(y)),xlab='X',ylab="LPM.1.CDF")
points(sort(y),LPM.1.CDF.y,col='red',type='l',lwd=3)
abline(v=mean(c(mean(x),mean(y))),col='darkgrey',lty=3,lwd=3)
target=mean(c(mean(x),mean(y)))
segments(x0=min(y),y0=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
         x1=target,y1=LPM(1,target,y)/(LPM(1,target,y)+UPM(1,target,y)),
           col='red',lty=3,lwd=3)
segments(x0=min(y),y0=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
         x1=target,y1=LPM(1,target,x)/(LPM(1,target,x)+UPM(1,target,x)),
           col='blue',lty=3,lwd=3)
segments(x0=min(y),y0=.5,
         x1=target,y1=.5,
           col='darkgrey',lty=3,lwd=3)
#KS line
lines(ecdf(x),lty=3,col='blue');lines(ecdf(y),lty=3,col='red')
cdf1 <- ecdf(x);cdf2 <- ecdf(y)
minMax <- seq(min(x, y), max(x, y), length.out=length(x))
x0<- minMax[which( abs(cdf1(minMax) - cdf2(minMax)) == max(abs(cdf1(minMax) - cdf2(minMax))) )]
y0 <- cdf1(x0)
y1 <- cdf2(x0)
points(c(x0, x0), c(y0, y1), pch=16, col="purple")
segments(x0, y0, x0, y1, col="purple", lty="dotted")
legend('topleft',legend = c('X','Y'),fill=c('blue','red'),border=NA,bty='n')
```



Comparing NNS CDF analysis with a t-test and KS-test:

```
NNS.ANOVA(cbind(x,y))
```



```
## Certainty
## 0.7401294
```

```
t.test(x,y)
```

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -1.2734, df = 1018.3, p-value = 0.2032
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -1.0380753  0.2210259
## sample estimates:
##  mean of x  mean of y
## 0.01612787 0.42465253
```

```
ks.test(x,y)
```

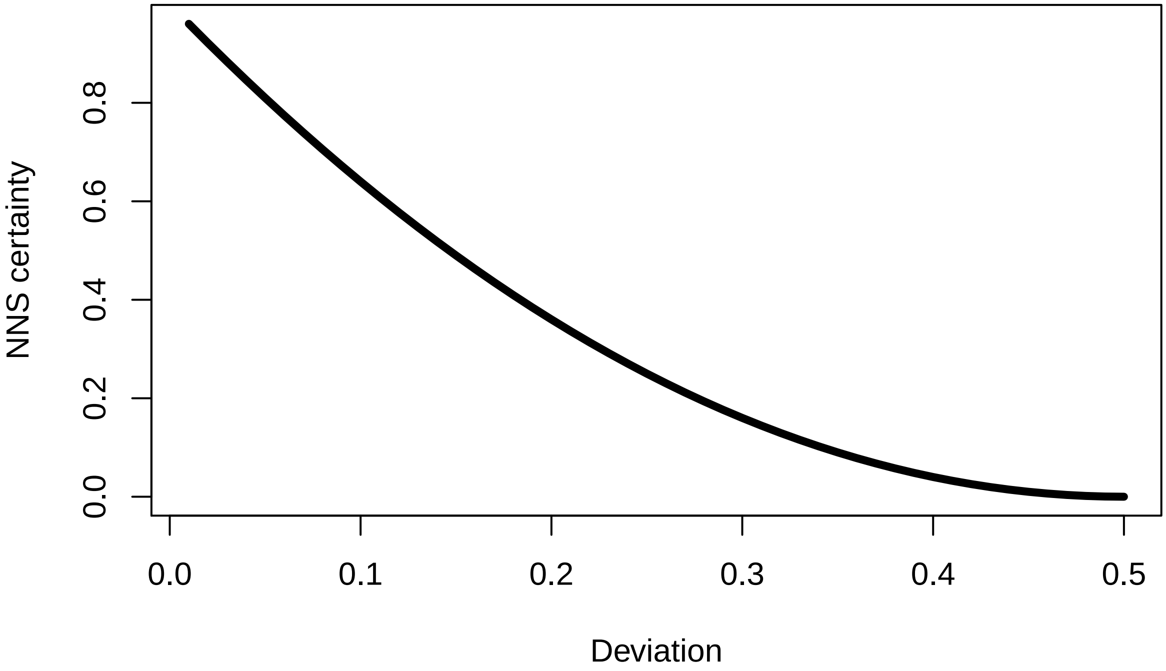
```
##
##  Two-sample Kolmogorov-Smirnov test
##
## data:  x and y
## D = 0.422, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

Of course t-tests are only informed by differences in means, while NNS and KS consider variances and other distribution characteristics. However, as we see, KS can evaluate differences far removed from the means while NNS simultaneously considers sample means explicitly.

7 NNS Certainties

Finally a note on the NNS certainty statistic. NNS uses the squared sum of absolute `LPM.1.CDF` deviations from the 0.5 null hypothesis. The range of these deviations is [0,.5] (representing a target below the sample range and above the sample range respectively.). The associated certainties over the range of possible deviations is clearly bound within the interval [0,1].

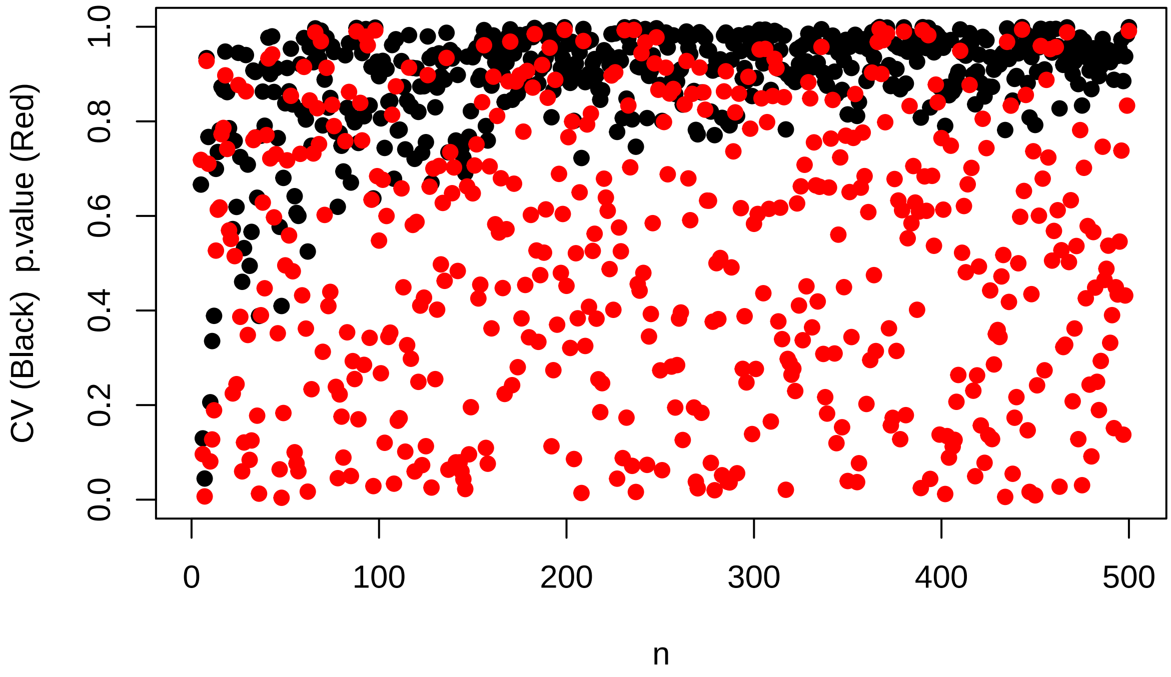
```
deviation=seq(.01,.5,.01)
plot(deviation,(0.5-deviation)^2/.25,type='l',
     lwd=4,xlab="Deviation",ylab="NNS certainty")
```



7.1 NNS Certainties vs. p values

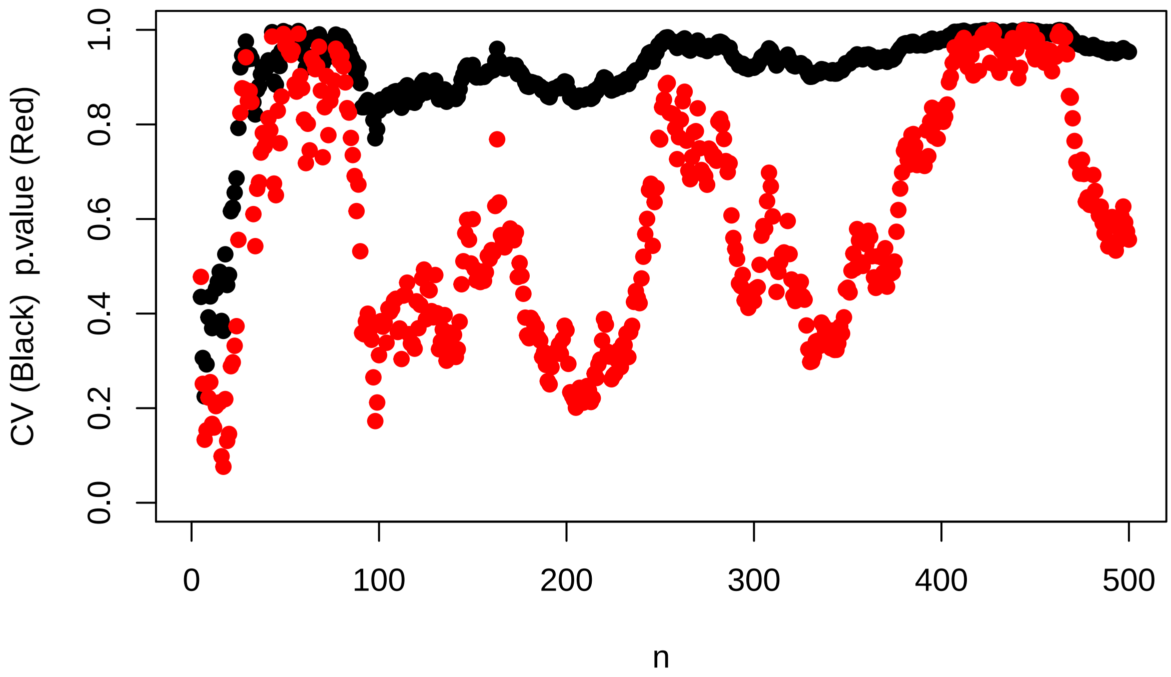
Below is a visualization of the stability of the NNS certainty statistic vs. the corresponding p value. The first example creates two standard normal distributions ranging in size from 5 to 500 observations.

```
set.seed(123)
certainties=numeric();p.values=numeric()
for(i in 5:500){
  x1=rnorm(i);x2=rnorm(i)
  certainties[i]=NNS.ANOVA(x1,x2,plot=FALSE)$Certainty
  p.values[i]=t.test(x1,x2)$p.value
}

plot(certainties,pch=19,xlab='n',ylab="CV (Black)  p.value (Red)",ylim=c(0,1))
points(p.values,col='red',pch=19)
```



Finally, we isolate a single distribution and increase the number of observations from 5 to 500. Again, note the stability of the NNS certainty statistic…

```
set.seed(123)
certainties=numeric();p.values=numeric()
x1=rnorm(500);x2=rnorm(500)
for(i in 5:500){
  certainties[i]=NNS.ANOVA(x1[1:i],x2[1:i],plot = FALSE)$Certainty
  p.values[i]=t.test(x1[1:i],x2[1:i])$p.value
}

plot(certainties,pch=19,xlab='n',ylab="CV (Black)  p.value (Red)",ylim=c(0,1))
points(p.values,col='red',pch=19)
```



8 References

To learn more about NNS statistics and their theoretical foundations, see "*Nonlinear Nonparametric Statistics: Using Partial Moments*" available on Amazon:

http://a.co/5bpHvUg

More hands-on NNS examples posted on GitHub:

https://github.com/OVVO-Financial/NNS/tree/NNS-Beta-Version/examples

Questions or comments welcomed at: ovvo.financial.systems@gmail.com