On the Distribution of Prime Numbers

Viole, Fred
fred.viole@gmail.com

April 25, 2017

INTRODUCTION

We present several contentions with the Prime Number Theorem (PNT) and offer alternative explanations. The first section of this paper deals with the probability a number is prime. We link this probability to the floor of the square root of the number. We then illustrate how incorporating the floor of the square root of the number generates a step function that more accurately approximates $\pi(x)$ than $\frac{x}{ln(x)}$.

The second section of this paper deals with convergences. We demonstrate increased accuracy to $\pi(x)$ by identifying the optimal coefficient in our function. We then transpose the convergence argument to the Log base such that at infinity, $\frac{x}{ln(x)}$ is valid, but not for any finite number.

Finally, we look at the historical treatment of convergences and the substitution of limit conditions derived from Euler's product formula.

1 Probability a Number is Prime

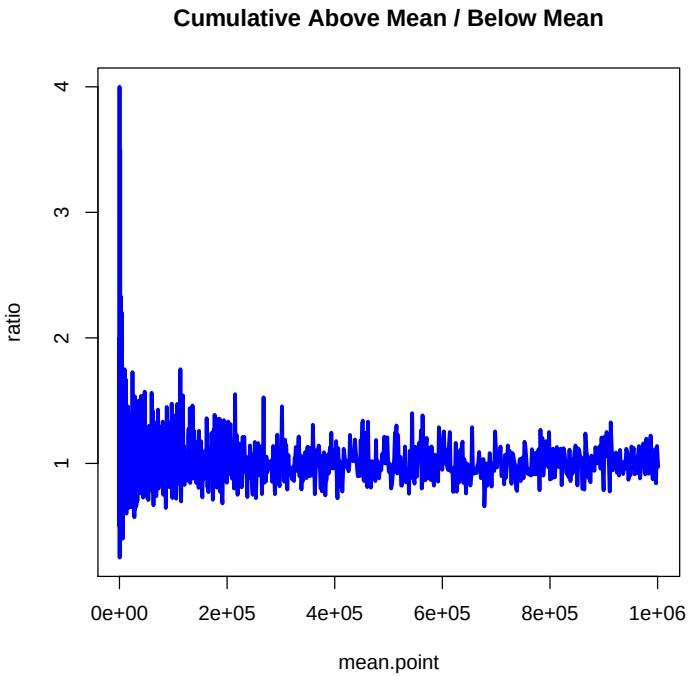
The probability a number is prime is equal to the size of its square root, since one factor must reside within the range $[0, \sqrt{x}]$. If two numbers have the same whole number part of their square root, they have equal probabilities of being prime.

For example, 101 and 119 have the same probability of being prime. Both numbers have 10 whole numbers in their square roots, 10.0499 and 10.9071 respectively. Thus the probability a number is prime over the range from 100 to 121 is uniform (for all numbers ending in 1, 3, 7, and 9). PNT asserts that the probability a number is prime is equivalent to $\frac{1}{ln(x)}$. This simply cannot be, and this declining probability over the range rather than the theoretical uniform leads to a systemic underestimate of primes (especially when ranges get larger via the distance between squares of larger numbers) as noticed with $\frac{x}{ln(x)}$.

1.1 Uniform Probability Assumption

We perform two tests to demonstrate the uniform probability assumption using p and the interval mean, $\mu_{interval}$. When testing the uniform probability assumption, we found the ratio of prime numbers above the mean from two whole numbers squared to the prime numbers below this mean asymptotic to 1. We only tested several hundred thousand, but this could be an area of further research to further bolster this assumption presented in Figure 1 below.

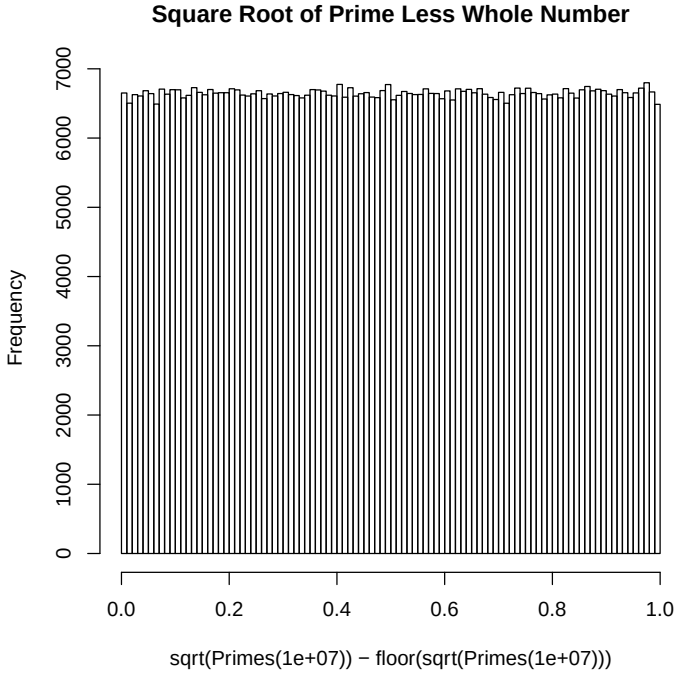
```
> require(pracma);require(numbers);require(NNS)
> ratio=numeric()
> mean.point=numeric()
> for(i in 1:1000){
+   mean.point[i]=mean(c(i^2,(i+1)^2))
+   ratio[i]=sum(Primes(i^2,(i+1)^2)<mean.point[i])/
+     sum(Primes(i^2,(i+1)^2)>mean.point[i])}
> plot(mean.point,ratio,type = 'l',col='blue',lwd=3,
+ main = "Cumulative Above Mean / Below Mean")
```



Viewing this uniformity assumption from a different perspective, the square root of a prime number has an equal probability of ending in a .25 or .9, since the range is reset with every perfect square. In fact, when taking the decimal of the prime number for the first 664,579 primes ($x = 10,000,000$), we found that the decimal is uniformly distributed. If $\frac{x}{ln(x)}$ were a viable approximation, then

there should exist more .01 primes than .99 primes, accompanied by a non-linear cumulative output. However, this is not the case as shown in Figure 2.

```
> hist(sqrt(Primes(1e7))-floor(sqrt(Primes(1e7))),breaks = 99,
+ main="Square Root of Prime Less Whole Number")
```



As the number gets larger, this uniformity holds for any equal sized decimal range between the perfect squares. For example, the number of primes with square roots ending in $[.01, .25] = [.25, .5] = [.5, .75] = [.75, .00]$.

2 Convergences

We will demonstrate for each x, an optimum log base exists such that $\pi(x) = \frac{x}{log_{OPTIMUMBase}(x)}$ and this optimum logarithmic base is itself a convergent function with limit e. The rate of convergence of Equation (1), often given as the definition of e, is pretty quick.

$$\lim_{x\to\infty}\left(1+\frac{1}{x}\right)^{(1+x)} = e \tag{1}$$

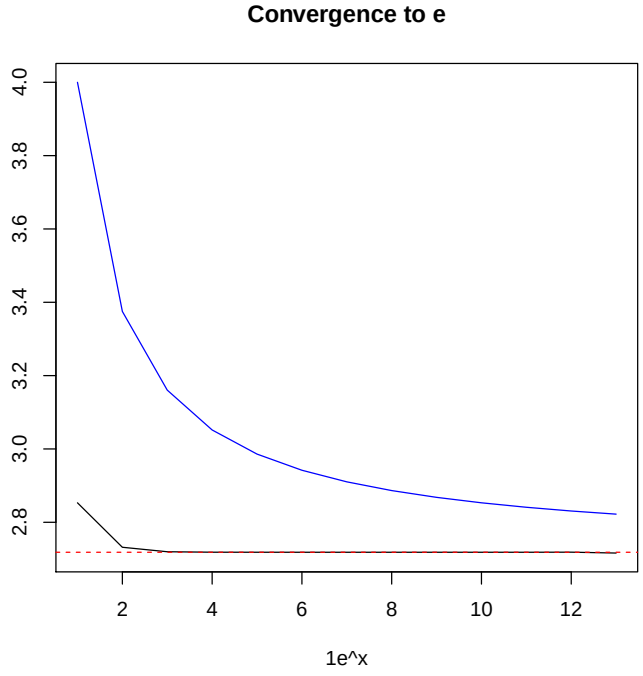
Using Equation (1) as the log base helps somewhat versus $\frac{x}{ln(x)}$, but not to

the accuracy desired. We can slow this convergence down further by incorporating additional logs to x as shown in Equation (2).

$$\lim_{x\to\infty}\left(1+\frac{1}{log_{10}(x)}\right)^{(1+log_{10}(x))} = e \tag{2}$$

Figure 3 below shows a comparison of the two convergences:

```
> eq.1=numeric()
> eq.2=numeric()
> for(i in seq(1,13,1)){
+   eq.1[i]=(1+(1/(1*10^i)))^(1+(1*10^i))
+   eq.2[i]=(1+(1/log(1*10^i,10)))^(1+log(1*10^i,10))
+ }
> plot(eq.1,type = 'l',ylim = c(min(eq.1),max(eq.2)),
+ main = "Convergence to e",ylab = '',xlab = '1e^x' )
> lines(eq.2,col='blue')
> abline(h=exp(1),lwd=1,col='red',lty=2)
>
```



2.1 Incorporating $\lfloor\sqrt{x}\rfloor$ into $\pi(x)$

Equation (3) represents our estimate of $\pi(x)$ by incorporating $\lfloor\sqrt{x}\rfloor$ to compensate for the probability a number is prime via its whole number square root

along with the slower convergence to e.

$$VF(x) = \left\lfloor \frac{\lfloor\sqrt{x}\rfloor^2}{ln\left(\frac{\lfloor\sqrt{x}\rfloor^2}{\left(1+\frac{1}{log_{10}(x)}\right)^{(1+log_{10}(x))}}\right)} \right\rfloor \tag{3}$$

$\lfloor\sqrt{x}\rfloor^2$ is the floor of the square root of the number to be estimated, squared. This may seem counterintuitive at first glance, but it reconciles a probability issue with the prime number theorem (PNT) as discussed in Section 1.

Here is the R-function for Equation(3), henceforth `VF`:

```
> Viole.Function <- function(x,base=exp(1)){
+
+   numerator<- floor(sqrt(x))^2
+
+   denominator<- log(numerator /
+               (1+(1/log(x,10)))^(1+log(x,10)),base=base)
+
+   max(0,floor(numerator/denominator))
+ }
```

2.2 OPTIMUM Log Base

At infinity, all estimates are equal since all series are asymptotic to e. Below is a comparison to the optimum log base $\pi(x) = \frac{x}{log_{OPTIMUMBase}(x)}$ versus implied log base from various function estimates of $\pi(x)$.

For example, the optimum base (b) of $\frac{x}{log_b(x)}$ when $x = 1000$ can be derived with the `uniroot` function in R.

```
> uniroot(function(x) length(Primes(1000))-(1000/log(1000,base=x)),
+ lower = 2, upper = 4, tol = 1e-9)$root
```

```
[1] 3.191538
```

We can now expand this optimum base to see its convergence properties to e and see what the implied bases are for `VF(x)` and `li(x)`.

```
> optimum.base=numeric()
> x.lnx.base=numeric()
> VF.base=numeric()
> li.x.base=numeric()
> for(i in seq(2,8,1)){
+ optimum.base[i]=uniroot(function(x)
+ length(Primes(1*10^i))-((1*10^i)/log((1*10^i),base=x)),
+ lower = 2, upper = 4, tol = 1e-9)$root
```

```
+ VF.base[i]=uniroot(function(x)
+ Viole.Function(1*10^i)-((1*10^i)/log((1*10^i),base=x)),
+ lower = 2, upper = 4, tol = 1e-9)$root
+ li.x.base[i]=uniroot(function(x)
+ (li(1*10^i)-li(2))-((1*10^i)/log((1*10^i),base=x)),
+ lower = 2, upper = 4, tol = 1e-9)$root
+ }
> optimum.bases=cbind(optimum.base,VF.base,li.x.base)
> optimum.bases[complete.cases(optimum.bases),]

     optimum.base  VF.base li.x.base
[1,]     3.162278 3.801894  3.816098
[2,]     3.191538 3.191538  3.386051
[3,]     3.101701 3.118890  3.148015
[4,]     3.017172 3.014741  3.029969
[5,]     2.957931 2.958585  2.963187
[6,]     2.918806 2.917725  2.920399
[7,]     2.890123 2.889433  2.890525
```

Equation (1) is too fast of a convergence to e, and would far outpace the optimum or Equation (3) (henceforth `VF(x)`). However, we can see that even the `VF(x)` estimate needs to be lowered once we get into the realm of large numbers, presumably at another decreasing rate converging to e. There is also no a priori reason why this log base estimate isn't itself functional towards a convergence to e with a decreasing base (<10) as we attempted to capture with the use of the ln outer term in the denominator shown in Equation (3). This log base divergence is why $\pi(x) > \frac{x}{ln(x)} for all x < \infty$, *and this nested log base convergence issue is the reason no specific distribution of prime numbers can exist.*

This is a strikingly similar argument towards the basic fact that every number has an increasing number of prime factors less than its square root, and that these factors cannot be summarily dismissed (and no decision rule for primality exists due to the uniform probability over the interval between perfect squares per Section 1), or that an ever increasing (infinite) number of non-trivial zero's are needed for accuracy in Riemann's $R(x)$; regardless if they all reside on $s = 1/2$ in $\zeta(s)$ as also suggested by the square root's influence on prime probability.

$$\begin{aligned} Li(x) &\sim \sum_{k=0}^{\infty} \frac{k!x}{ln(x)^(k+1)} \\ &\sim \frac{x}{ln(x)} + \frac{x}{ln(x)^2} + \frac{x}{ln(x)^3} + \cdots \end{aligned} \tag{4}$$

$Li(x)$ also adds to the log base in decreasing increments via its expanded exponential series. However, the addition is generally too great, leading to log base values exceeding the optimum and ultimately leading to overestimates of

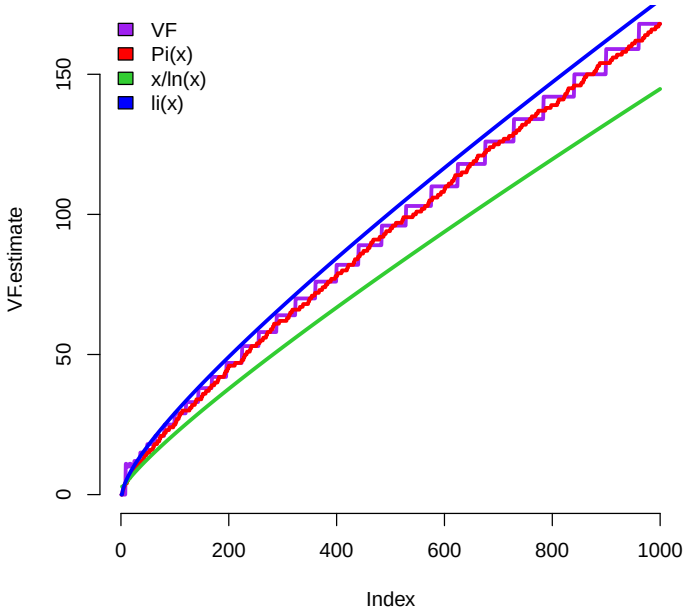
$\pi(x)$. The inflexibility inherent in its numerator factorials leads to increments that cannot replicate the optimum base, even with averaging techniques. We can see the convergence to the optimum for $Li(x)$, but the limit is e, not a dynamically induced optimum log base value. Thus, it is consistent with the notion that $Li(x)$ will in turn have lower values than the optimum log base and underestimate $\pi(x)$ for larger values of x. It is not conjectured that an infinite amount of crosses exist, as that would require a very dynamic shift in the implied log base convergence function underlying $Li(x)$ or of $\pi(x)$. Furthermore, when averaging $Li(x)$ to manipulate the effective log base rate, slowing a divergence does not necessarily guarantee a synchronization or convergence to the optimum log base rate (from above or below).

2.3 Intersection of $\pi(x)$ and VF(x)

If VF(x) were a valid estimate of $\pi(x)$, then $\pi(x)$ should cross VF(x) once in each interval from underneath as VF(x) is a step function; overestimating $\pi(x)$ below the mean of the interval between two squares, and underestimating $\pi(x)$ above the mean. The step comes from the next whole number in the square root of x. This square root step function undoubtedly has some relation to the Riemann Hypothesis as the critical strip, $s = 1/2$ is consistent with the real part of the line being a square root in the zeta function $\zeta(s)$.

2.3.1 `VF(x)` Estimate Visualization

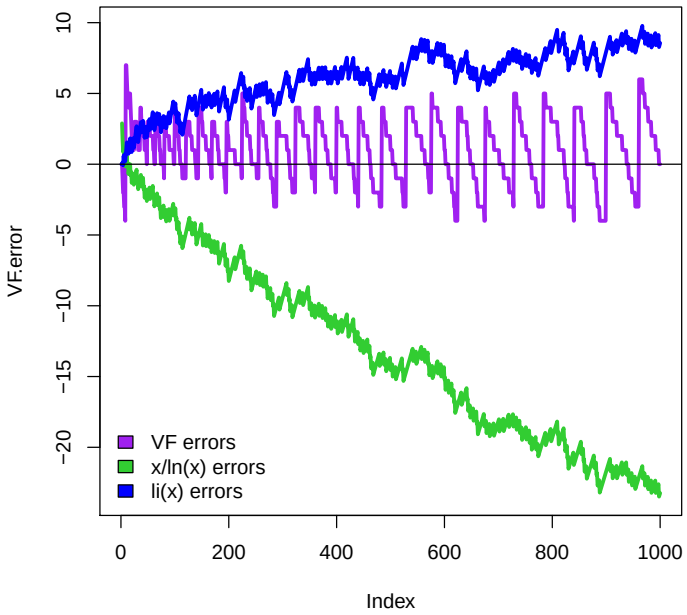
```
> n=1000
> VF.estimate=numeric()
> pi.actual=numeric()
> x.lnx=numeric()
> li.x=numeric()
> for(i in 1:n){
+   VF.estimate[i]=Viole.Function(i)
+   pi.actual[i]=length(Primes(1,i)[Primes(1,i)<i])
+   x.lnx[i]=i/log(i)
+   li.x[i]=max(0,Re(li(i)-li(2)))
+ }
> plot(VF.estimate,col='purple',type='l',lwd=3,bty='n')
> lines(pi.actual,col='red',lwd=3)
> lines(x.lnx,col='limegreen',lwd=3)
> lines(li.x,col='blue',lwd=3)
> legend('topleft',legend=c("VF","Pi(x)","x/ln(x)","li(x)"),
+ fill = c('purple','red','limegreen','blue'),
+ bty='n')
```



2.3.2 `VF(x)` Errors

And we can compare the errors of both methods:

```
> VF.error=VF.estimate-pi.actual
> x.lnx.error=x.lnx-pi.actual
> li.x.error=li.x-pi.actual
> plot(VF.error,type = 'l',col='purple',lwd=3,ylim = c(min(x.lnx.error),
+ max(c(li.x.error,VF.error))))
> lines(x.lnx.error,col='limegreen',lwd=3)
> lines(li.x.error,col='blue',lwd=3)
> abline(h=0)
> legend('bottomleft',legend=c("VF errors","x/ln(x) errors","li(x) errors"),
+ fill = c('purple','limegreen','blue'),
+ bty='n')
```

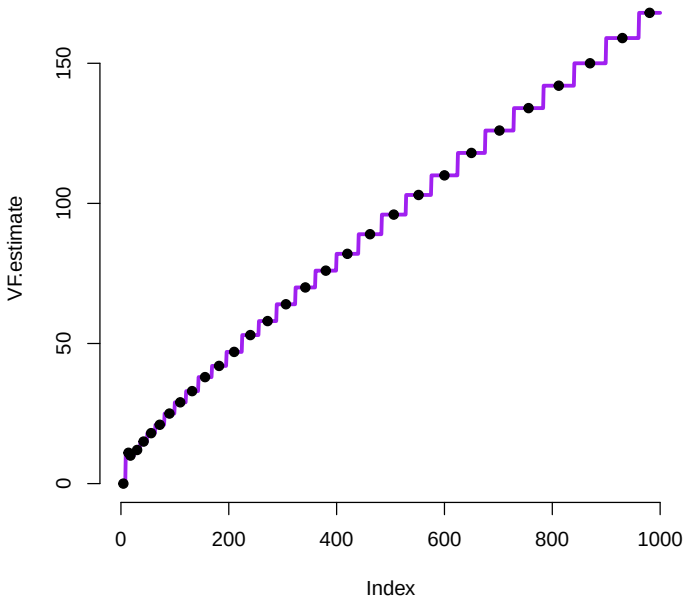


3 Midpoints of `VF(x)`

We can improve these errors by reducing the VF(x) step function to a continuous line. BY isolating the midpionts of `VF(x)`, we can then fit them and interpolate estimates.[1]

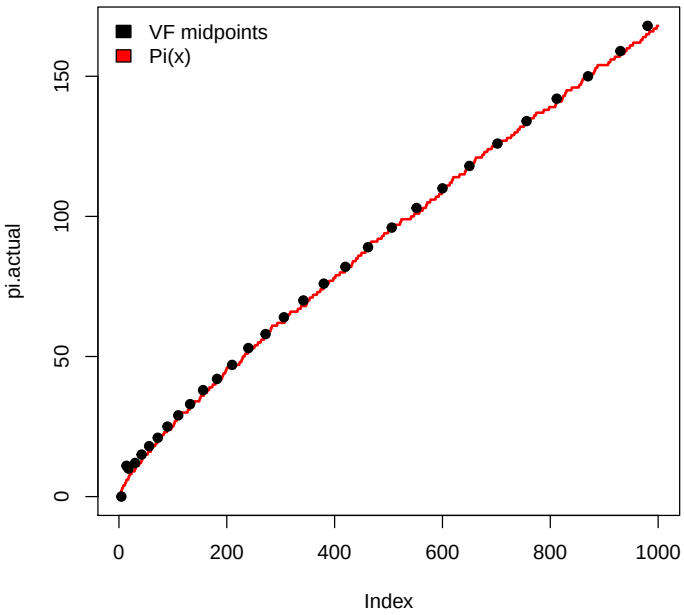
```
> midpoint.df=cbind((1:n),VF.estimate)
> midpoints=aggregate(x = midpoint.df[,1], by =list(midpoint.df[,2]), FUN = "mean")
> plot(VF.estimate,col='purple',type='l',lwd=3,bty='n')
> points(midpoints$x,midpoints$Group.1,pch=19)
```

[1] Connecting all points linearly and interpolating all integer values within $[1, x]$ will likely be faster than `NNS.reg`.



Now we add back $\pi(x)$ and isolate our VF(x) midpoints.

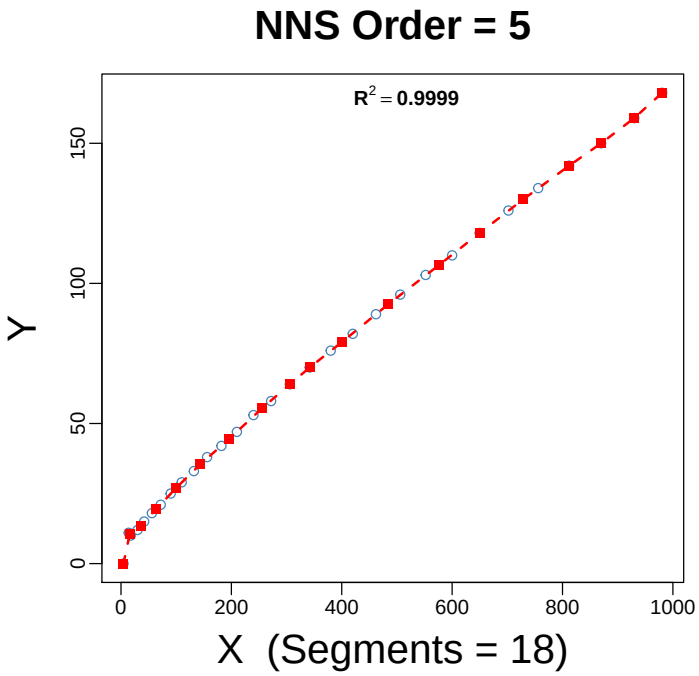
```
> plot(pi.actual,col='red',type = 'l',lwd=2)
> points(midpoints$x,midpoints$Group.1,pch=19)
> legend('topleft',legend=c("VF midpoints","Pi(x)"),fill = c('black','red'),
+ bty='n')
```



Here is the nonlinear regression fitting our VF(x) midpoints.[2]

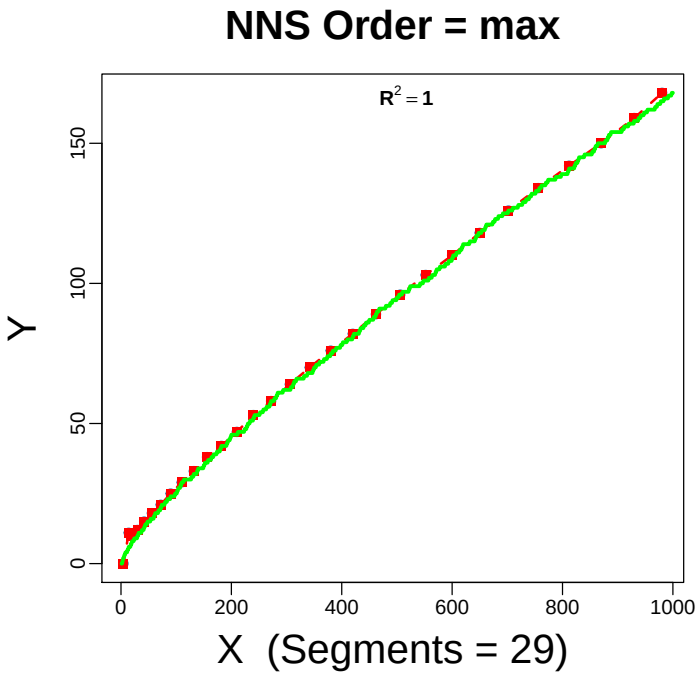
```
> NNS.reg(midpoints$x,midpoints$Group.1)$R2
```

[2]Surprisingly, these midpoints would not succumb to an exponential fit...hence the nonlinear regression.



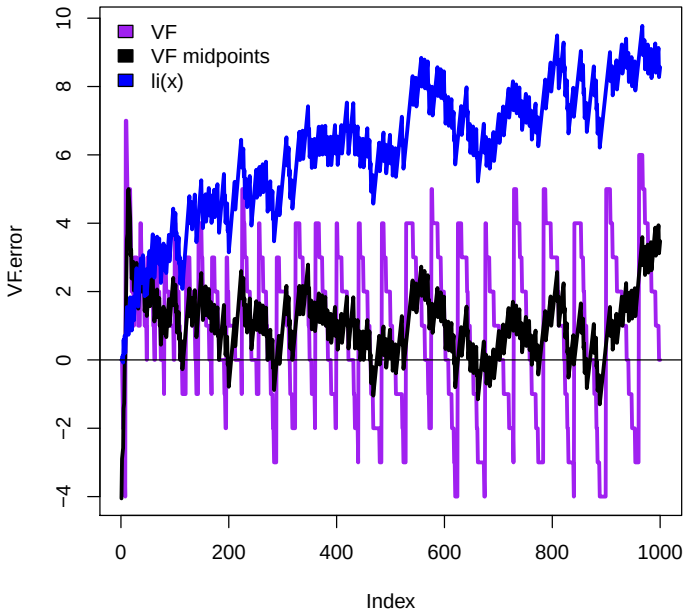
With $\pi(x)$ overlayed in green this time, we can barely see our fitted VF(x) midpoints regression.

```
> NNS.reg(midpoints$x,midpoints$Group.1,order='max')$R2
> lines(pi.actual,col='green',lwd=3)
```



So let's compute our errors from our VF(x) midpoints regression to $\pi(x)$ and see if we improved our estimate over the naive step VF(x). We need to generate our interpolated estimates using the 'point.est' parameter in `NNS.reg`.

```
> NNS.VF.errors=NNS.reg(midpoints$x,midpoints$Group.1,point.est = c(1:n),plot = F,
+ order='max')$Point.est
> plot(VF.error,type = 'l',col='purple',lwd=3,
+ ylim = c(min(c(VF.error,li.x.error)),
+ max(c(VF.error,li.x.error))))
> lines(NNS.VF.errors-pi.actual,col='black',lwd=3)
> lines(li.x.error,col='blue',lwd=3)
> legend('topleft',legend=c("VF","VF midpoints","li(x)"),
+ fill = c('purple','black','blue'),
+ bty='n',horiz = F)
> abline(h=0)
```

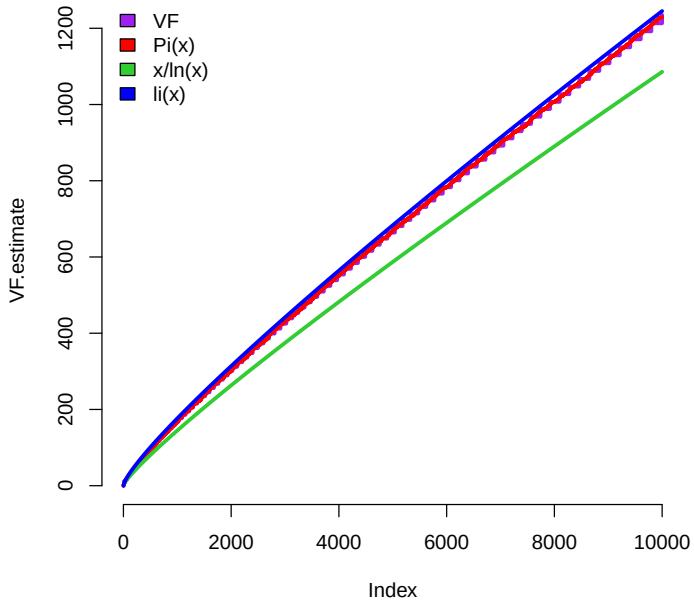


3.1 Increased x

We will increase x to 10,000 to see if the fit still holds well, and if our `VF(X)` midpoint error terms are still well behaved.

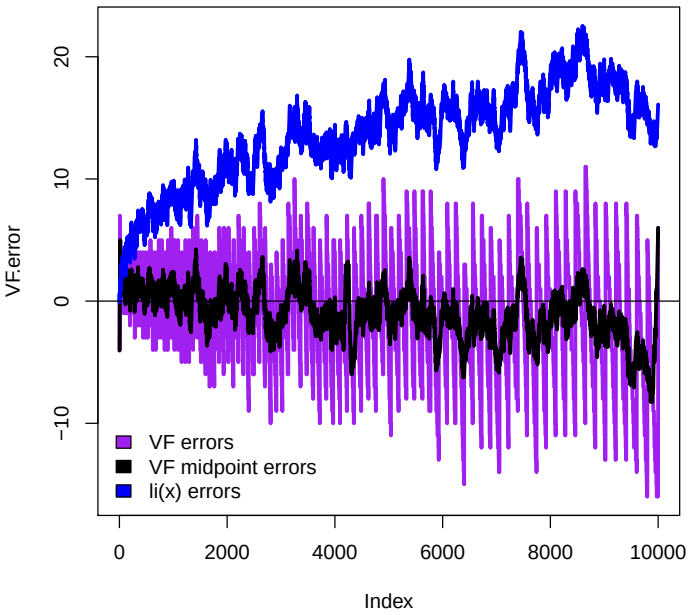
```
> n=10000
> VF.estimate=numeric()
> pi.actual=numeric()
> x.lnx=numeric()
> li.x=numeric()
> for(i in 1:n){
+   VF.estimate[i]=Viole.Function(i)
+   pi.actual[i]=length(Primes(1,i)[Primes(1,i)<i])
+   x.lnx[i]=i/log(i)
+   li.x[i]=max(0,Re(li(i)-li(2)))
+ }
> plot(VF.estimate,col='purple',type='l',lwd=3,bty='n')
> lines(pi.actual,col='red',lwd=3)
> lines(x.lnx,col='limegreen',lwd=3)
> lines(li.x,col='blue',lwd=3)
> legend('topleft',legend=c("VF","Pi(x)","x/ln(x)","li(x)"),
+ fill = c('purple','red','limegreen','blue'),
```

```
+ bty='n')
> VF.error=VF.estimate-pi.actual
> x.lnx.error=x.lnx-pi.actual
> li.x.error=li.x-pi.actual
> midpoint.df=cbind((1:n),VF.estimate)
> midpoints=aggregate(x = midpoint.df[,1], by = list(midpoint.df[,2]), FUN = "mean")
```



Fit looks good, are the errors well behaved?

```
> NNS.VF.errors=NNS.reg(midpoints$x,midpoints$Group.1,point.est = c(1:n),plot = F,
+ order='max')$Point.est
> plot(VF.error,type = 'l',col='purple',lwd=3,
+ ylim = c(min(VF.error),
+ max(c(li.x.error,VF.error))))
> lines(NNS.VF.errors-pi.actual,col='black',lwd=3)
> lines(li.x.error,col='blue',lwd=3)
> abline(h=0)
> legend('bottomleft',legend=c("VF errors","VF midpoint errors","li(x) errors"),
+ fill = c('purple','black','blue'),
+ bty='n',horiz = F)
```



They are.

Limits & Comments

A final thought on primes which takes us back all the way to Euler's product formula:

$$\sum_{n=1}^{\infty} \frac{1}{n^s} = \prod_{p=\texttt{prime}} \frac{1}{1-p^{-s}}$$

For any finite number n, there is no equality. This is also realized from the fact that e is an irrational number in the natural log base. Therefore, whatever number one stops at it in utilizing e, an underestimate of $\pi(x)$ is guaranteed.

Unfortunately, limits were used to perform analysis with beautiful accompanying work since Gauss's initial prime conjecture and Euler's subsequent proof. However, *the story of the prime numbers is at its heart a convergence story; and to use limits is to simply tell a different story.*

This work would be greatly advanced by a more capable language than R, but we hope the provided code is readable enough to be ported to other languages.

We look to extend the analysis to much larger numbers with dynamic log bases and ultimately compare to Riemann's estimate.