Diagnostics

Jimmy G. Moore

October 12, 2018

Display Initial Results

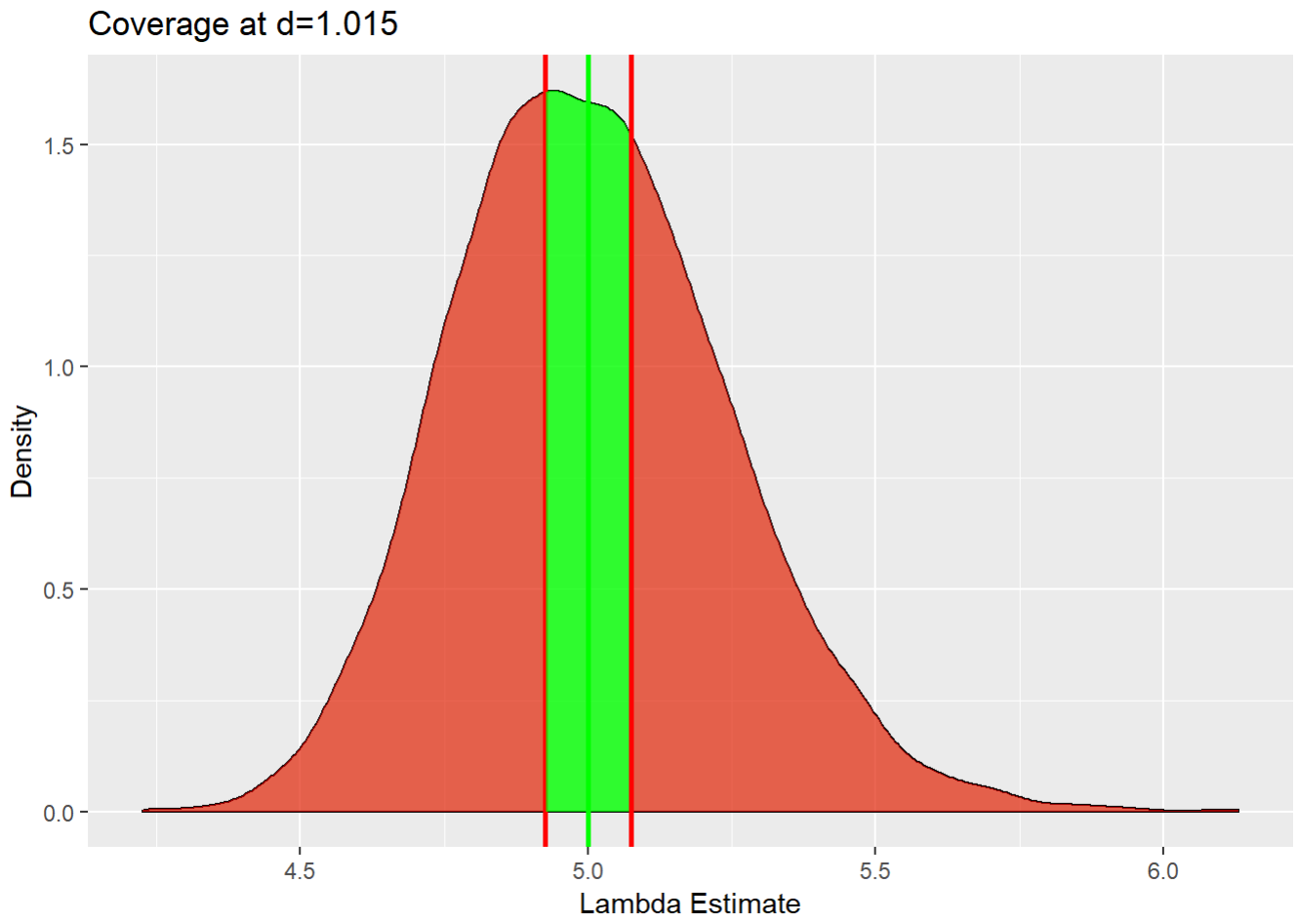
```
resultsTab = resultsTab[,(2:dim(resultsTab)[2])]
resultsTab
```

```
##        D Pilot    True.N Estimated.N Std.Error.of.N Estimated.True
## 1  1.021    50  58.86369     60.4180       7.987543       1.026405
## 2  1.020    50  64.83344     66.0640       8.862213       1.018980
## 3  1.019    50  71.76695     73.0138       9.681423       1.017374
## 4  1.018    50  79.88385     81.1802      10.027674       1.016228
## 5  1.017    50  89.47014     90.8640      10.556031       1.015579
## 6  1.016    50 100.90381    102.4634      11.468399       1.015456
## 7  1.015    50 114.69283    116.4360      11.891601       1.015199
## 8  1.014    50 131.53266    133.2364      12.994071       1.012953
## 9  1.013    50 152.39597    154.0946      13.594404       1.011146
## 10 1.012    50 178.67669    180.8808      14.861600       1.012336
## 11 1.011    50 212.42950    214.3936      16.531114       1.009246
## 12 1.010    50 256.78501    259.2968      17.949361       1.009782
##     N.TrueN Coverage.Probability Std.Err.Coverage Expected.Lambda
## 1  1.554308               0.2590        0.4381294        5.028948
## 2  1.230556               0.2356        0.4244157        5.027831
## 3  1.246845               0.2492        0.4325931        5.025090
## 4  1.296347               0.2400        0.4271258        5.019327
## 5  1.393861               0.2546        0.4356797        5.014619
## 6  1.559587               0.2316        0.4218970        5.010786
## 7  1.743171               0.2400        0.4271258        5.004509
## 8  1.703742               0.2470        0.4313099        5.005112
## 9  1.698626               0.2412        0.4278541        5.004037
## 10 2.204115               0.2548        0.4357923        4.997046
## 11 1.964097               0.2422        0.4284576        5.000379
## 12 2.511787               0.2458        0.4306036        4.995214
##    std.err.Lambda Time.to.compute
## 1       0.3311778            7.07
## 2       0.3231285            8.72
## 3       0.3147944           13.99
## 4       0.2936658           26.51
## 5       0.2745016           19.39
## 6       0.2637952           24.03
## 7       0.2397676           31.05
## 8       0.2279035           39.78
## 9       0.2055488           51.34
## 10      0.1918752           67.00
## 11      0.1794718           87.33
## 12      0.1606374          118.03
```

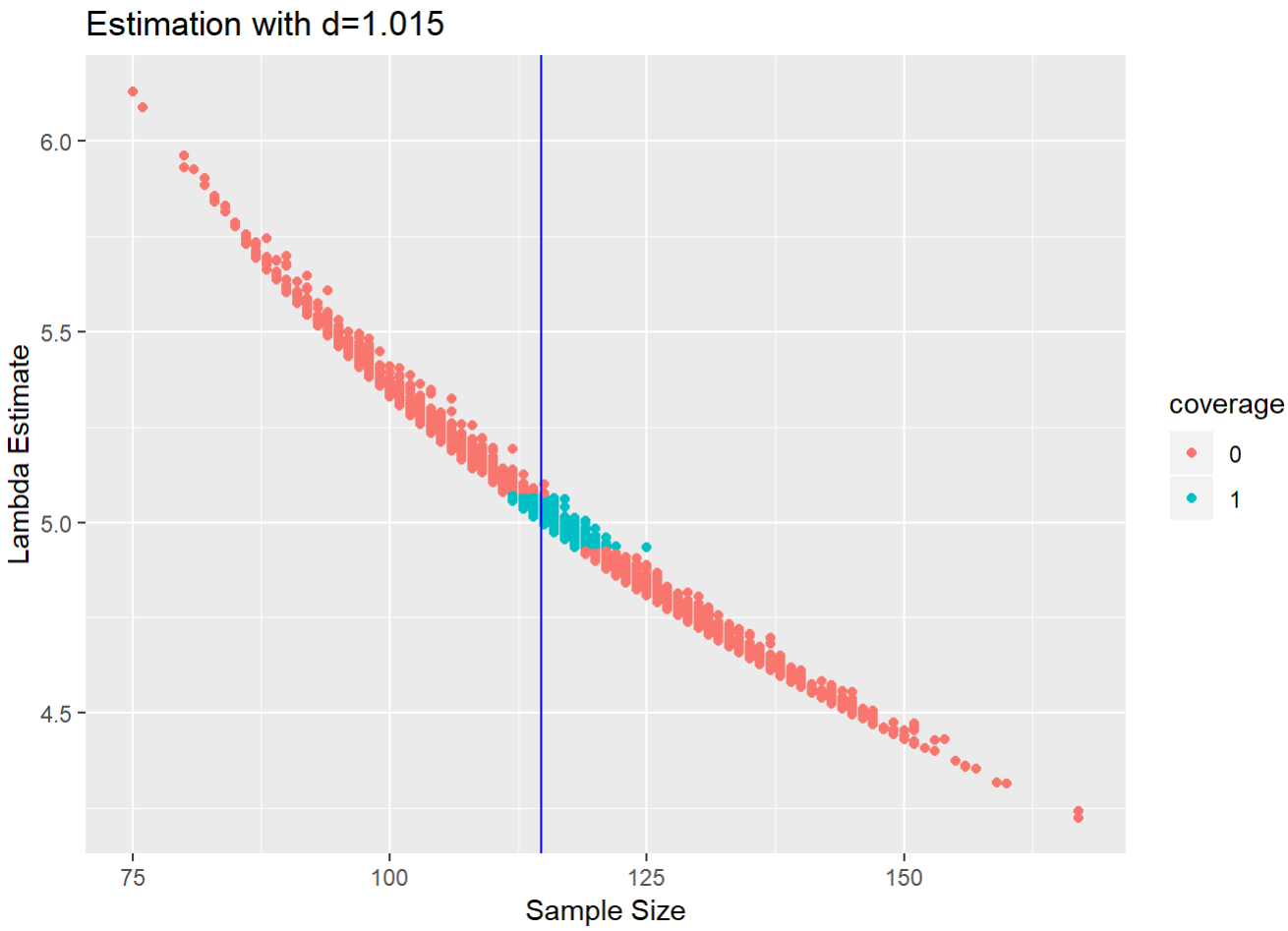
Show Performance of Lambda Estimate

For these Examples D=1.015

```
#Do Christmas Tree Plot on D value = 1.015
d15Frame = bigData[which(bigData$dLevels == '1.015'),]

library(ggplot2)
dIndex = 7
d = dVals[dIndex]
##### Cool Plot
#create a density plot with data
#Add line for population parameter (5)
#Add lower (5/d) and upper(5*d) bound lines
coolPlot = ggplot(data = d15Frame,aes(x=lambdaEst))+
  geom_density(fill = "green",alpha = 0.2)+
  geom_vline(aes(xintercept = 5),color = "green",size=1)+
  geom_vline(aes(xintercept = 5/d),color = "red",size=1)+
  geom_vline(aes(xintercept = 5*d),color = "red",size = 1)

#Not sure what this step does but is seems to store all data from geom_plot
subPlot = ggplot_build(coolPlot)$data[[1]]

#add colors based on subsets
coolPlot = coolPlot +
  geom_area(data = subset(subPlot,x>(5/d)& x<(5*d)),aes(x=x,y=y),fill = "green",alpha = 0.75)+
  geom_area(data = subset(subPlot,x>(5*d)),aes(x=x,y=y),fill = "red",alpha = 0.6)+
  geom_area(data = subset(subPlot,x<(5/d)),aes(x=x,y=y),fill = "red",alpha = 0.6)+
  xlab("Lambda Estimate")+
  ylab("Density")+
  ggtitle("Coverage at d=1.015")
coolPlot
```

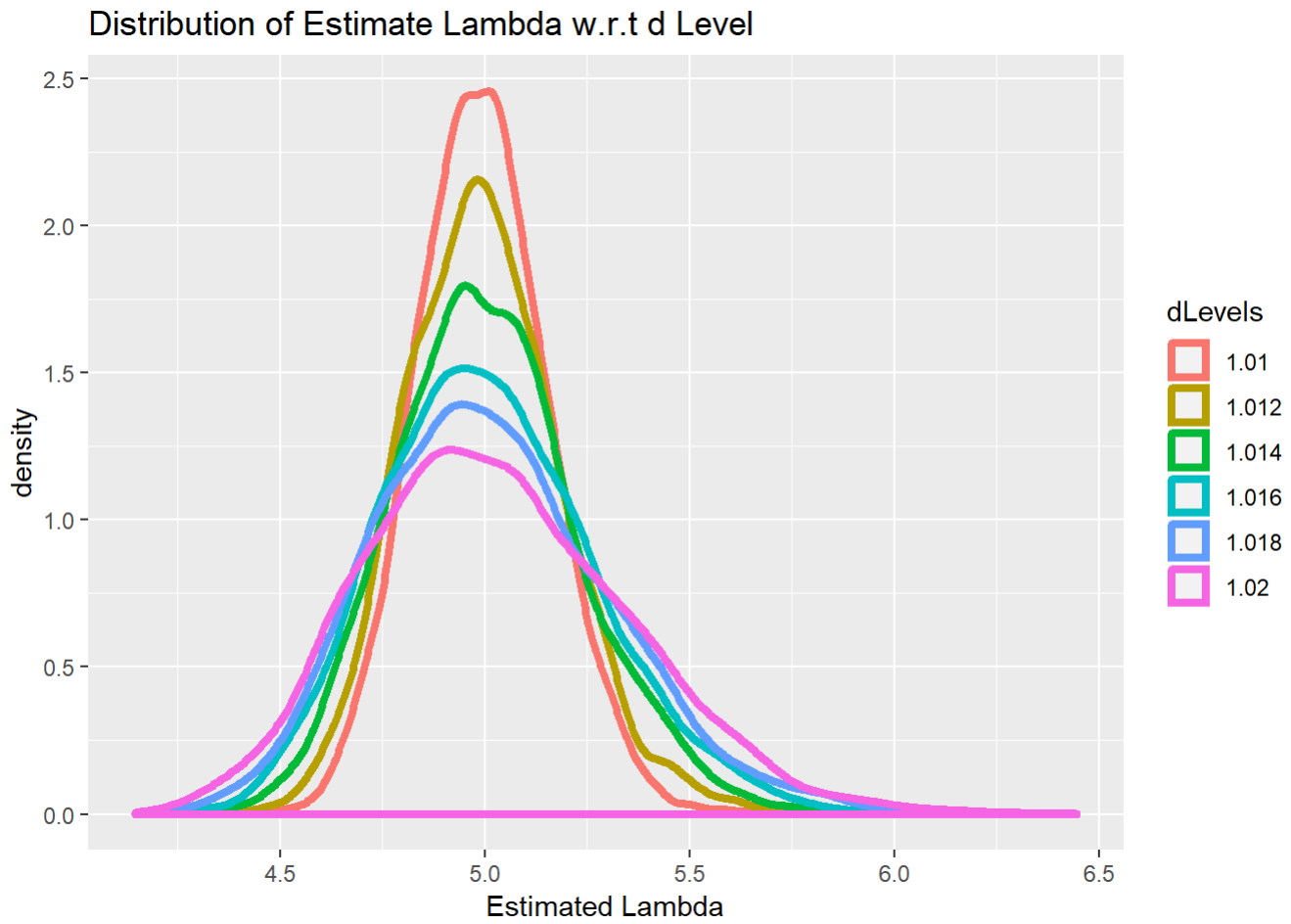



```
####
## Plot N on Lambda Est
ggplot(data = d15Frame,aes(x=nSize,y=lambdaEst,color = coverage))+
  geom_point()+
  geom_vline(aes(xintercept = 114.69),color = "blue",size = 0.5)+
  ggtitle("Estimation with d=1.015")+
  xlab("Sample Size")+
  ylab("Lambda Estimate")
```



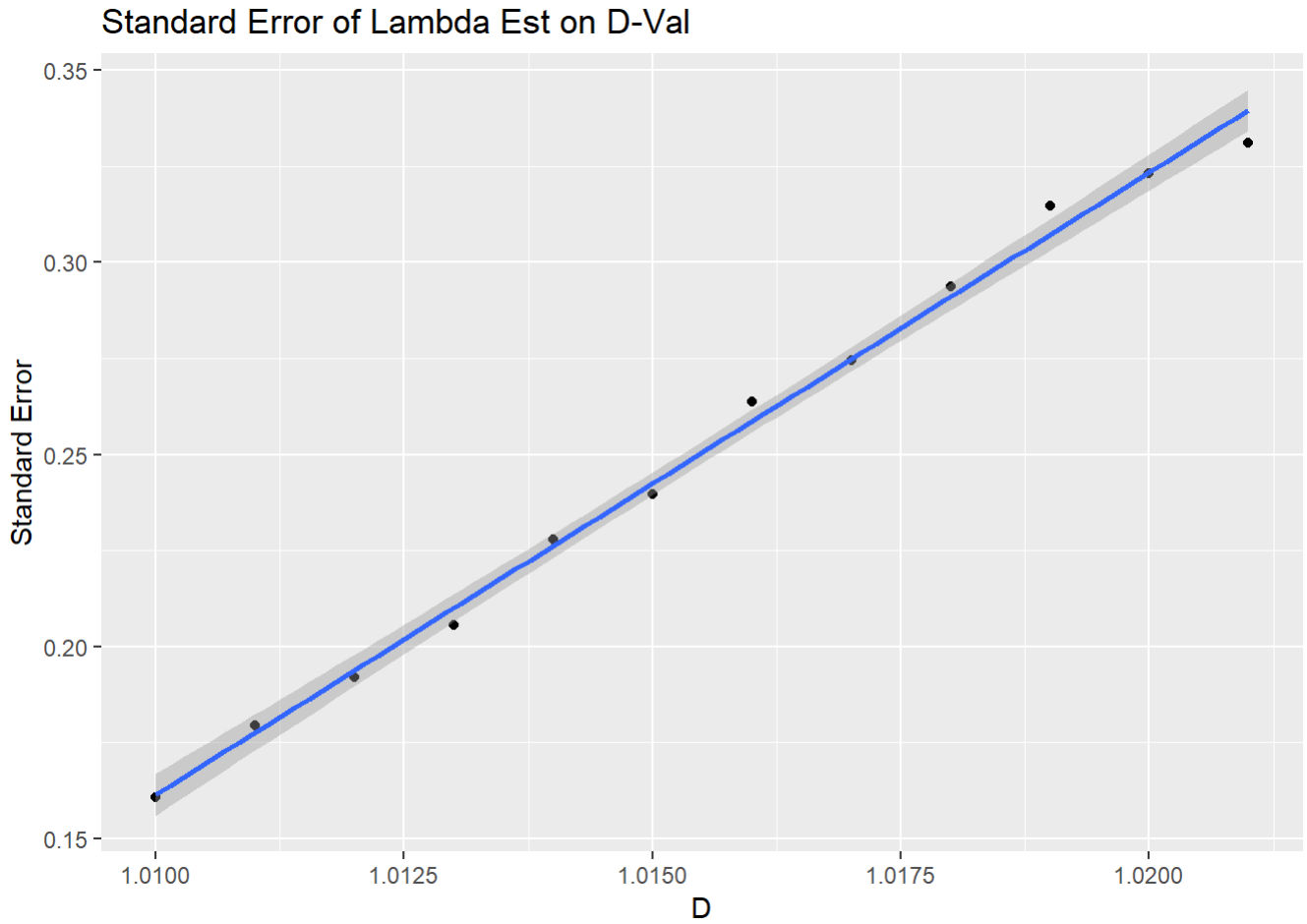
Check Standard Errors

```
halfData = bigData[which(bigData$dLevels == '1.02' | bigData$dLevels == '1.018' | bigData$dLevel
s =='1.016' | bigData$dLevels=='1.014' | bigData$dLevels == '1.012' | bigData$dLevels=='1.01'),]

ggplot(data = halfData,aes(x=lambdaEst, color = dLevels))+
  geom_density(size=1.5)+
  ggtitle("Distribution of Estimate Lambda w.r.t d Level")+
  xlab("Estimated Lambda")
```

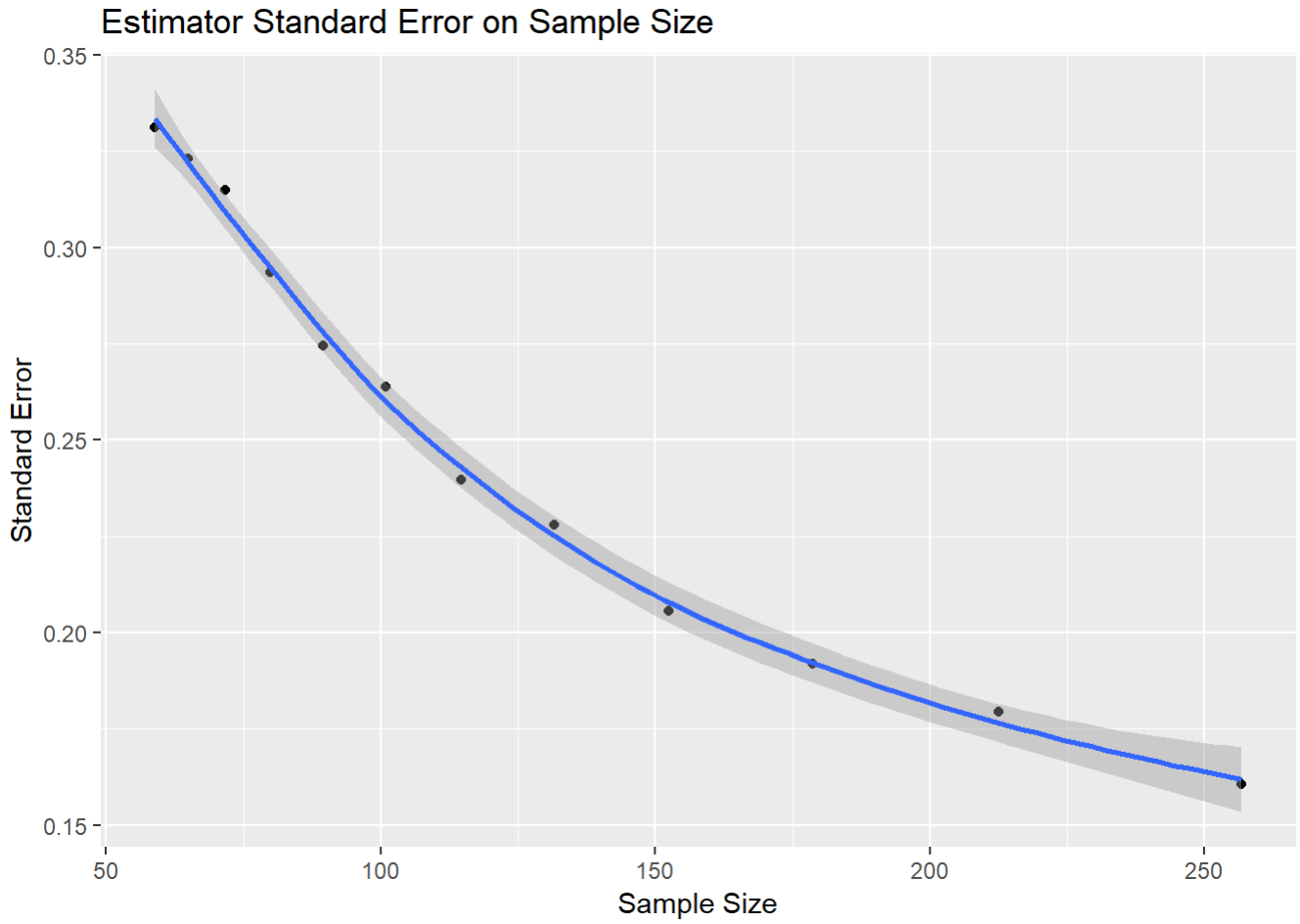


Association between $\hat{\lambda}$ standard error and d

Th First plot will show the relationship between standard deviation and our d value



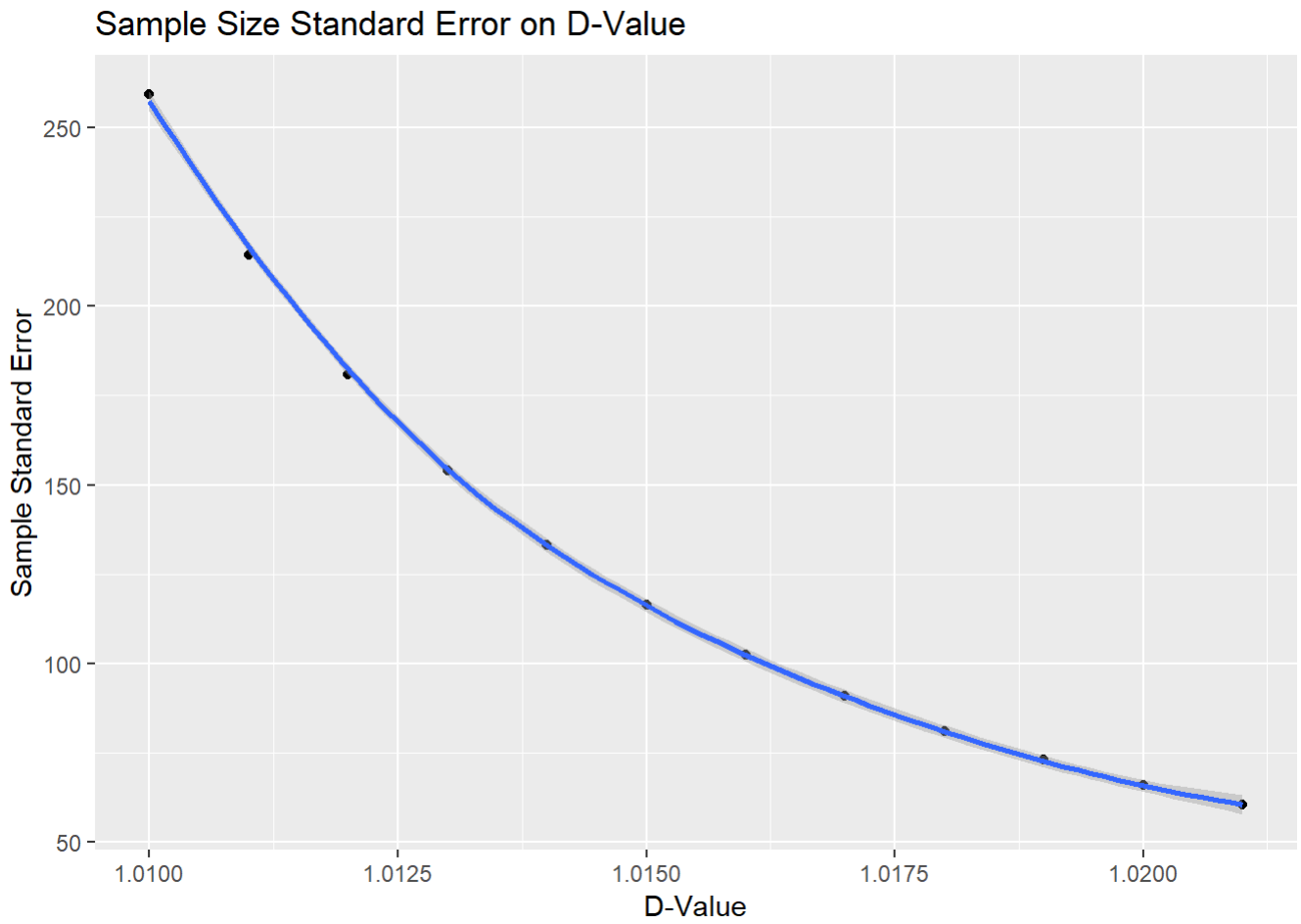
Next we will look at the realtionship between sample size and standard deviation.



The above plot shows that $\hat{\lambda}$ is consistent, meaning as sample size increases our variance converges to 0. Unfortunately, as our sample size increases, the width of our confidence interval gets smaller. Therefore we do not see a significant change in our coverage probability.

Next we will look at the realtionship between d-Value and standard error of our Sample Size estimation.

```
SSvsSD = data.frame(x = resultsTab[,1],
                    y = resultsTab[,4])

ggplot(data = SSvsSD, aes(x = x,y = y))+
  geom_point()+
  geom_smooth(method="loess",formula = y~log(x))+
  ggtitle("Sample Size Standard Error on D-Value")+
  xlab("D-Value")+
  ylab("Sample Standard Error")
```

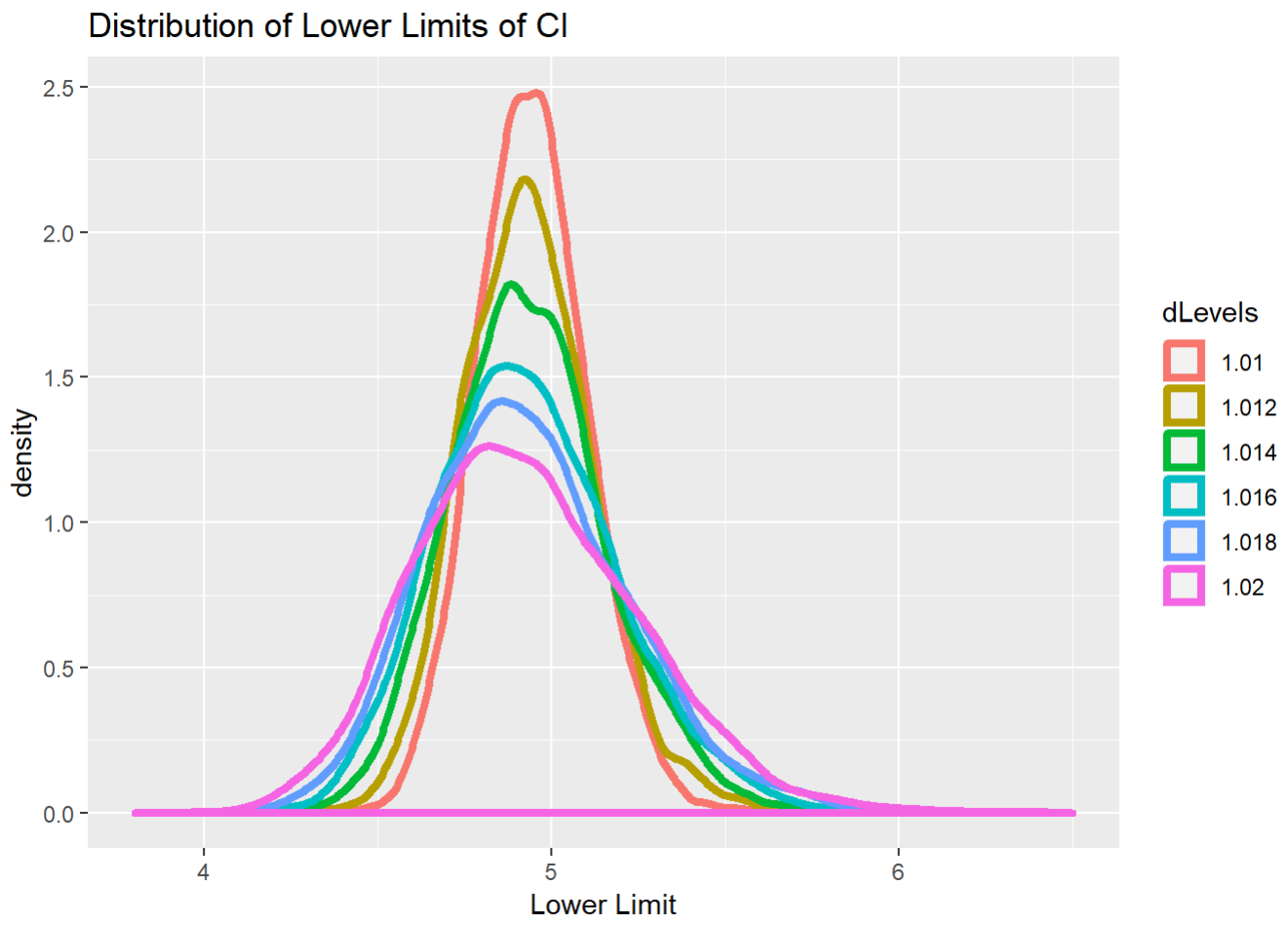


Analysis of Sequential Confidence Interval

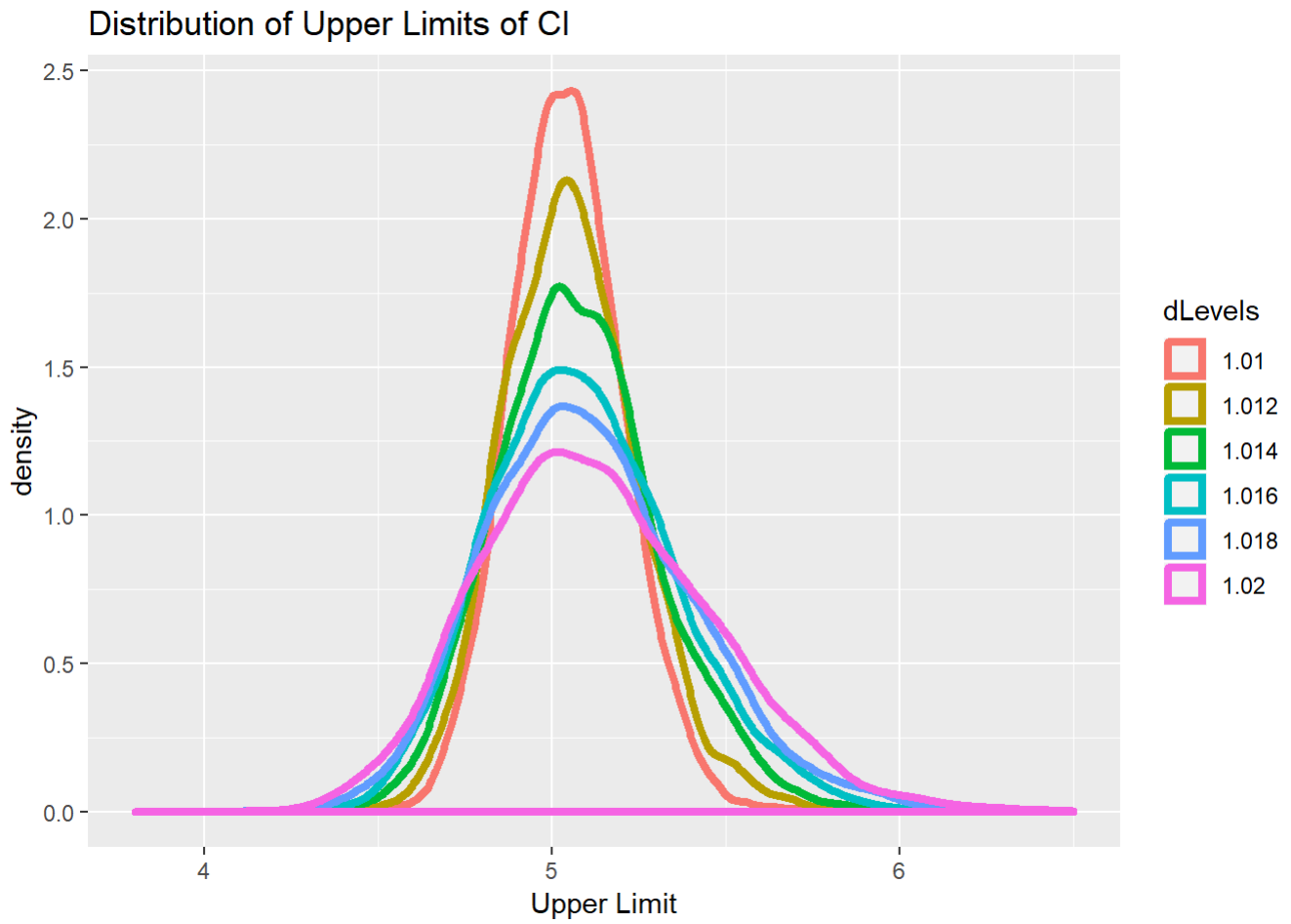
Distribution of Upper and Lower Bounds

First let us take a look at the distribution of our upper and lower bounds as they relate to d. To keep the plot from being too busy we have omitted half of the d values, however the general trend of the data is clear.

```
ggplot(data=halfData,aes(x=lower,color = dLevels))+
  geom_density(size = 1.5)+
  scale_x_continuous(limits = c(3.8,6.5))+
  ggtitle("Distribution of Lower Limits of CI")+
  xlab("Lower Limit")
```




```
ggplot(data=halfData,aes(x=upper,color = dLevels))+
  geom_density(size=1.5)+
  scale_x_continuous(limits = c(3.8,6.5))+
  ggtitle("Distribution of Upper Limits of CI")+
  xlab("Upper Limit")
```



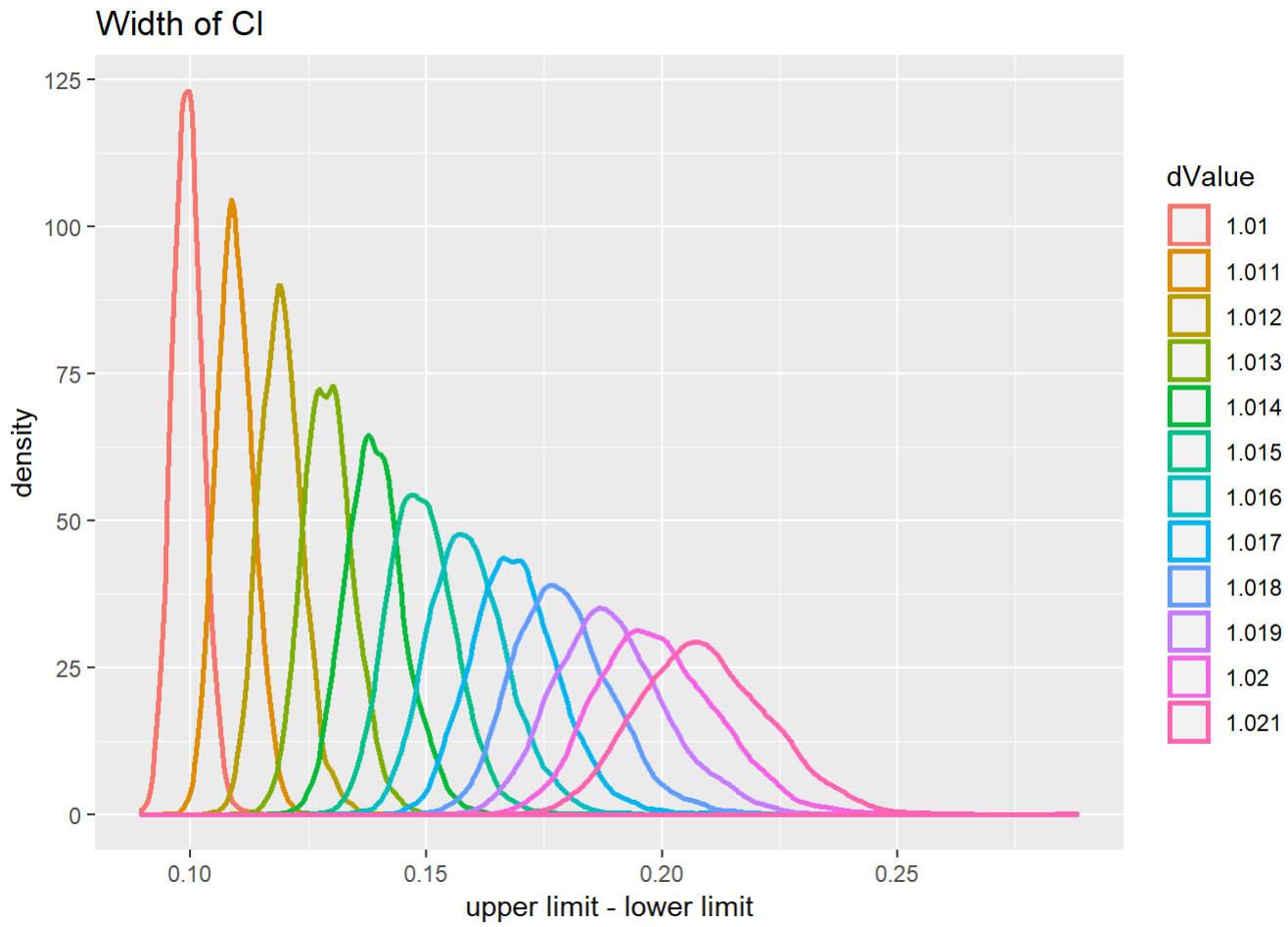
We can see that the the distributions ofthe lower and upper bounds of our confidence interval follow a similar shape to that of a our lambda estimator. This is because the bounds of CI are generated from our consistent estimator for lambda. Therefore, the variance in our bounds converges to 0 as sample size increases. To get another perspective on this we will next generate a table displaying distributional information for our upper and lower bounds.

```
boundsTable = matrix(0,nrow = 12, ncol = 9)
colnames(boundsTable) = c("dValue","Low Theo","Lower Mean","Low Theo/Obs","Lower StErr",
                          "Upper Theo","Upper Mean","U Theo/Obs","Upper StErr")
for(i in 1:12){
  boundsTable[i,1] = resultsTab[i,1]
  boundsTable[i,2] = 5/(resultsTab[i,1])
  boundsTable[i,3] = mean(bigData$lower[which(bigData$dLevels==dVals[i])])
  boundsTable[i,4] = boundsTable[i,2]/boundsTable[i,3]
  boundsTable[i,5] = sd(bigData$lower[which(bigData$dLevels==dVals[i])])
  boundsTable[i,6] = 5*(resultsTab[i,1])
  boundsTable[i,7] = mean(bigData$upper[which(bigData$dLevels==dVals[i])])
  boundsTable[i,8] = boundsTable[i,6]/boundsTable[i,7]
  boundsTable[i,9] = sd(bigData$upper[which(bigData$dLevels==dVals[i])])
}
boundsTable
```

```
##       dValue Low Theo Lower Mean Low Theo/Obs Lower StErr Upper Theo
##  [1,]  1.021 4.897160   4.925512    0.9942437   0.3243662      5.105
##  [2,]  1.020 4.901961   4.929246    0.9944646   0.3167926      5.100
##  [3,]  1.019 4.906771   4.931393    0.9950071   0.3089248      5.095
##  [4,]  1.018 4.911591   4.930577    0.9961494   0.2884733      5.090
##  [5,]  1.017 4.916421   4.930795    0.9970848   0.2699130      5.085
##  [6,]  1.016 4.921260   4.931876    0.9978475   0.2596409      5.080
##  [7,]  1.015 4.926108   4.930550    0.9990991   0.2362242      5.075
##  [8,]  1.014 4.930966   4.936008    0.9989787   0.2247569      5.070
##  [9,]  1.013 4.935834   4.939819    0.9991933   0.2029110      5.065
## [10,]  1.012 4.940711   4.937792    1.0005912   0.1896000      5.060
## [11,]  1.011 4.945598   4.945973    0.9999242   0.1775191      5.055
## [12,]  1.010 4.950495   4.945757    1.0009580   0.1590469      5.050
##       Upper Mean U Theo/Obs Upper StErr
##  [1,]   5.134556  0.9942437   0.3381326
##  [2,]   5.128388  0.9944646   0.3295911
##  [3,]   5.120566  0.9950071   0.3207755
##  [4,]   5.109675  0.9961494   0.2989518
##  [5,]   5.099867  0.9970848   0.2791681
##  [6,]   5.090958  0.9978475   0.2680159
##  [7,]   5.079576  0.9990991   0.2433641
##  [8,]   5.075183  0.9989787   0.2310942
##  [9,]   5.069089  0.9991933   0.2082210
## [10,]   5.057010  1.0005912   0.1941777
## [11,]   5.055383  0.9999242   0.1814460
## [12,]   5.045167  1.0009580   0.1622437
```

We can see that the standard error of our bouds distributions are decreasing with d value. Also, it can be noted that the standard error is symetric for both upper and lower bounds. Although the variability of our bounds estimators is decreasing, we also should note that our interval is getting small as d gets smaller. This is displayed in the proceeding figure.

```
widthDistribution = bigData$upper-bigData$lower
widthFrame = data.frame(dValue = bigData$dLevels,
                        width = widthDistribution)
ggplot(data = widthFrame, aes(x = width, color = dValue))+
  geom_density(size = 1)+
  ggtitle("Width of CI")+
  xlab("upper limit - lower limit")
```


Width of CI
125
100
75
50
25
0
density
0.10
0.15
0.20
0.25
upper limit - lower limit
dValue
1.01
1.011
1.012
1.013
1.014
1.015
1.016
1.017
1.018
1.019
1.02
1.021