252L - Problem Set 1

Klint Kanopka

January 26, 2018

1 Bernoulli Random Variables

Q. Compute all the correlations of the columns of this matrix (x1). What do you notice?

```
set.seed(12311)
x1<-matrix(rbinom(1000,1,.5),100,10)

cor(x1)
```

```
##               [,1]         [,2]         [,3]          [,4]         [,5]
##  [1,]  1.000000000  0.008410789 -0.009896948 -0.3032770731  0.094629567
##  [2,]  0.008410789  1.000000000  0.089886562 -0.0669602615 -0.057570773
##  [3,] -0.009896948  0.089886562  1.000000000  0.0295469723 -0.125809070
##  [4,] -0.303277073 -0.066960261  0.029546972  1.0000000000  0.089214650
##  [5,]  0.094629567 -0.057570773 -0.125809070  0.0892146505  1.000000000
##  [6,]  0.237526763  0.130744090 -0.001602564 -0.0525279507  0.035484609
##  [7,]  0.081814407 -0.074443750  0.047043222 -0.0988646639  0.060409150
##  [8,]  0.098085811 -0.016333199  0.059259270  0.0455242322 -0.103165975
##  [9,] -0.149960697 -0.107907043  0.053719716 -0.0004168548 -0.001638407
## [10,]  0.025551766  0.179665184  0.060860872  0.0365014114 -0.058030861
##               [,6]         [,7]        [,8]          [,9]       [,10]
##  [1,]  0.237526763  0.081814407  0.09808581 -0.1499606967  0.02555177
##  [2,]  0.130744090 -0.074443750 -0.01633320 -0.1079070433  0.17966518
##  [3,] -0.001602564  0.047043222  0.05925927  0.0537197158  0.06086087
##  [4,] -0.052527951 -0.098864664  0.04552423 -0.0004168548  0.03650141
##  [5,]  0.035484609  0.060409150 -0.10316597 -0.0016384067 -0.05803086
##  [6,]  1.000000000  0.087597723  0.09929932 -0.1904608106 -0.05925927
##  [7,]  0.087597723  1.000000000  0.02350749  0.0090628774  0.05755282
##  [8,]  0.099299317  0.023507488  1.00000000  0.0370118105  0.08043217
##  [9,] -0.190460811  0.009062877  0.03701181  1.0000000000  0.08500515
## [10,] -0.059259270  0.057552816  0.08043217  0.0850051472  1.00000000
```

The correlation matrix has 1's along the diagonal, which makes sense because:

$$\sigma(x, x) = 1$$

The matrix is also symmetric about the diagonal, which makes sense because:

$$\sigma(x, y) = \sigma(y, x)$$

The correlations are also relatively small, which makes sense because the data is generated randomly.

Q. Compute the row sums. What is the variation in row sums?

```
var(rowSums(x1))

## [1] 2.706667
```

The variance of the row sums is 2.706667. Given the context of the following question, this represents variance in total test scores across individuals.

Q. If you considered the 1s/0s correct and incorrect responses to test items (where the rows are people and the columns are items), does this seem like it could have come from a realistic scenario?

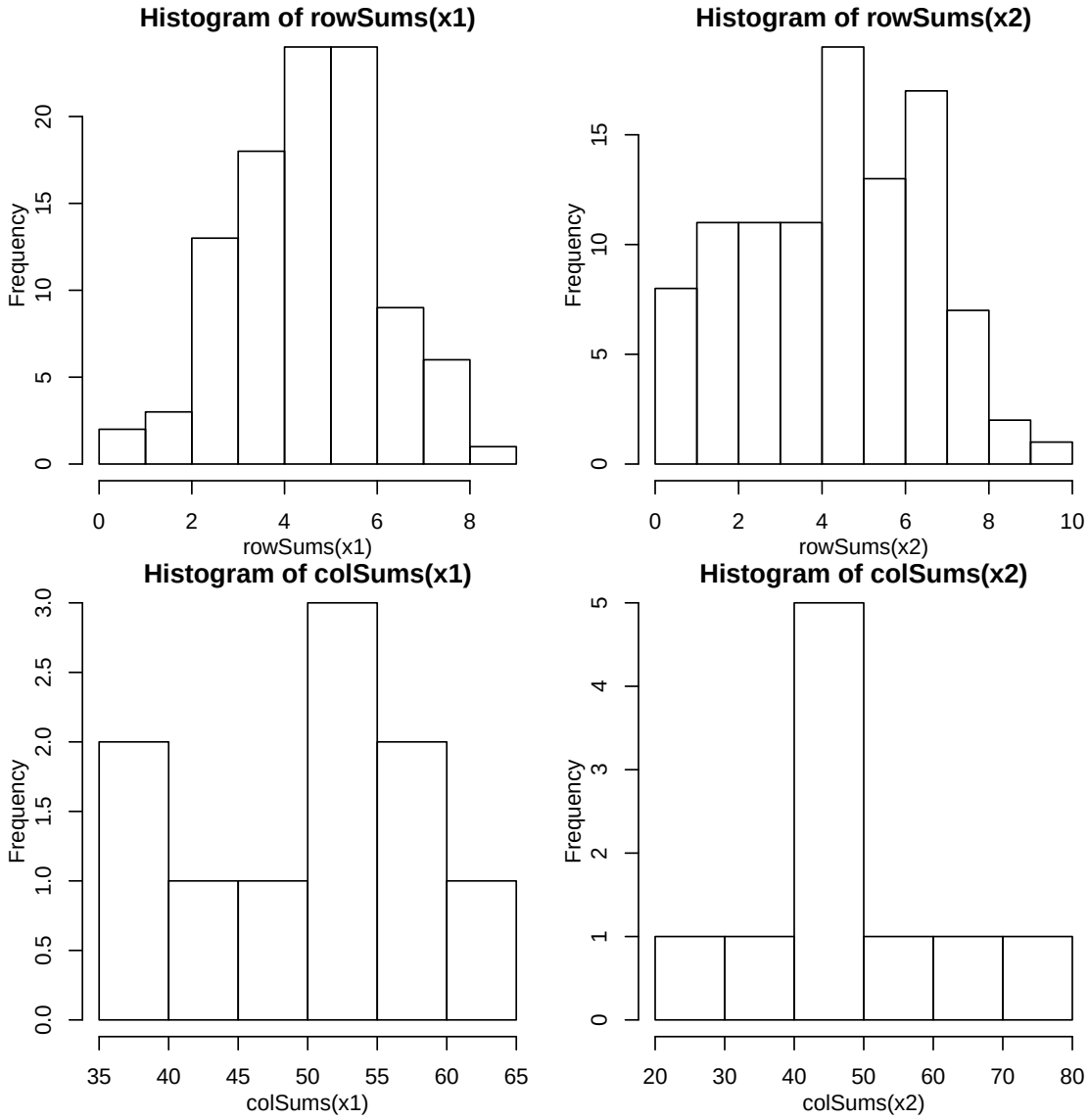
Probably not, because the correlations are so small. One might also expect a test to have "easy" and "hard" questions, which this randomly constructed matrix does not indicate when looking at the column sums.

Q. Now go back through the above questions and see what you make of this new matrix x2. Specifically, how does it compare to the first matrix x1 in terms of whether it seems like a realistic set of item responses? What characteristics (feel free to explore other features of the data) influence your opinion on this point?

```
var(rowSums(x2))

## [1] 5.111111
```

For x_2, the matrix still has 1's along the diagonal and is symmetric about the diagonal, but generally speaking the correlations are higher than they were in x_1.

The variance of the row sums is 5.111111. The variation in row sums here is higher. If you look at histograms of the row sums, the data is also somewhat more uniformly distributed in x_2 than in x_1, so it makes sense that the variance would be higher in this situation.

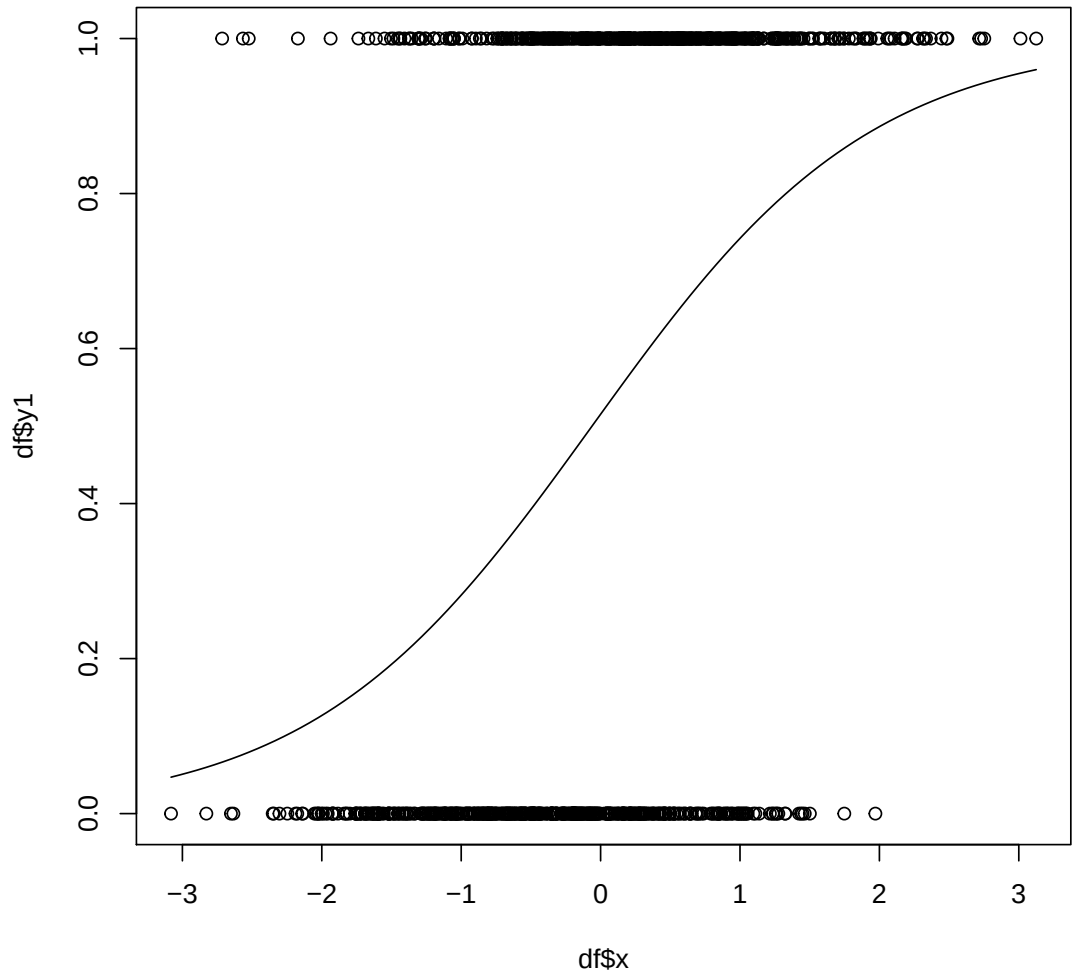


Looking at histograms of row and column sums in x_1 and x_2, x_2 is less clearly normally distributed, meaning that there are respondents at lower ability levels, but a sharper drop off after 7 (and 8) correct. More than anything, the difference in the variance between the column sums (61.7 in x_1 and 242.7 in x_2) implies that all the items in x_1 would have been of similar difficulty, whereas the items in x_2 are of varying difficulty. If you look at the histogram of column sums, however, you will see that this is blatantly untrue. The histogram of column sums for x_2, specifically, is the most damning piece of evidence in favor of x_1 being the "real test."

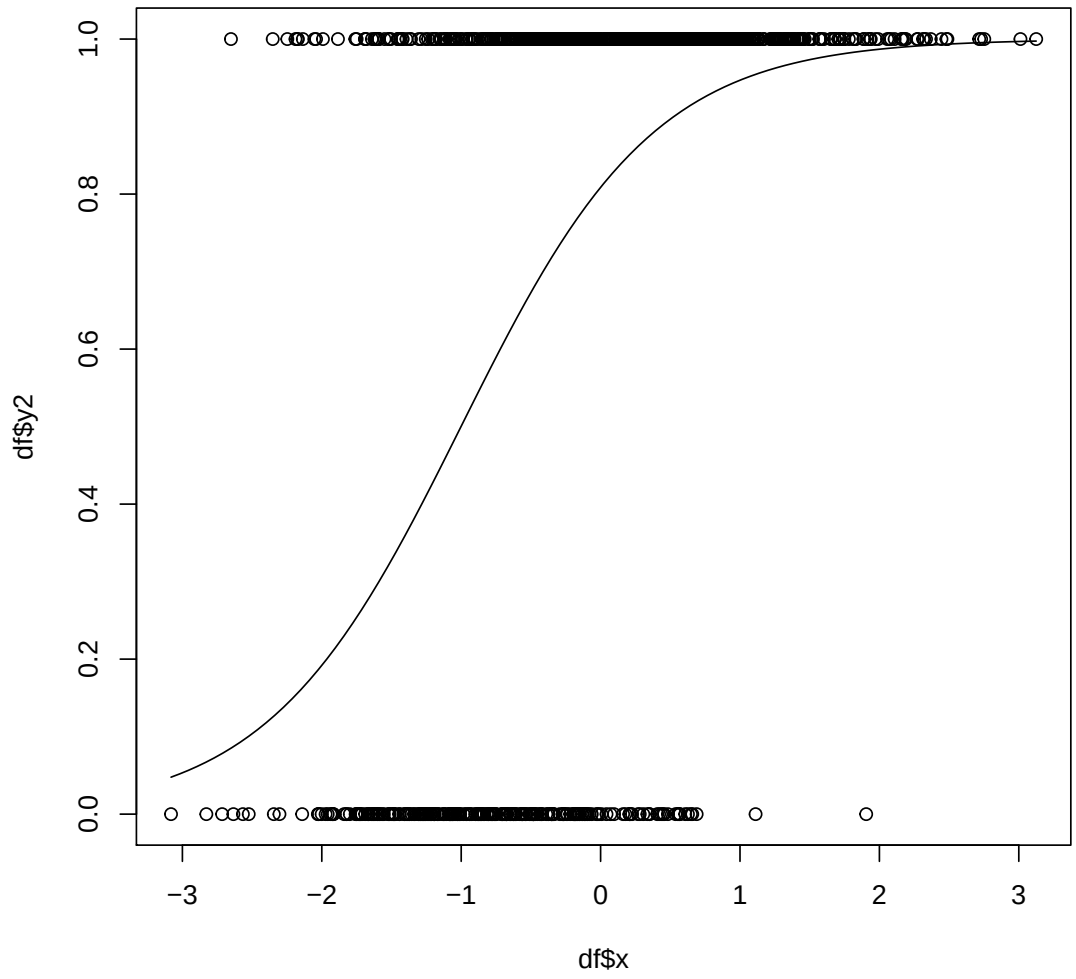
2 Logistic Regression

(1) How would you compare the association between y1/y2 & x? (2) How would you interpret the regression coefficients from (say) m1? (3) Do m1 and m2 show equivalent model fit? Can you notice anything peculiar about either y1 or y2 (in terms of their association with x)?

```
load("ps1-logreg.Rdata")

m1 <- glm(y1~x,df,family="binomial")
plot(df$x,df$y1)
curve(predict(m1,data.frame(x=x),type="resp"),add=TRUE)
```




```
m2 <- glm(y2~x,df,family="binomial")
plot(df$x,df$y2)
curve(predict(m2,data.frame(x=x),type="resp"),add=TRUE)
```



1. The association between x and y_2 is stronger than the association between x and y_1. This is because the AIC, null deviance and residual deviance are lower for y_2. In addition, when looking at plots, there is less overlap between the top and bottom sections in the plots of y_2 than in y_1. In y_2, the logistic curve has a steeper transition from minimum value to maximum value.

2. The regression coefficients, β_0 and β_1, are related to the shape of the logistic fit. Specifically, β_0 is related to the intercept. β_1 is a measure of test item discrimination. β_1 is related to the slope at the point of inflection, so a higher slope would indicate higher levels of discrimination. β_0 is related to difficulty, with a lower β_0 indicating an easier test item. β_0 is the odds of getting an answer correct when $x = 0$.

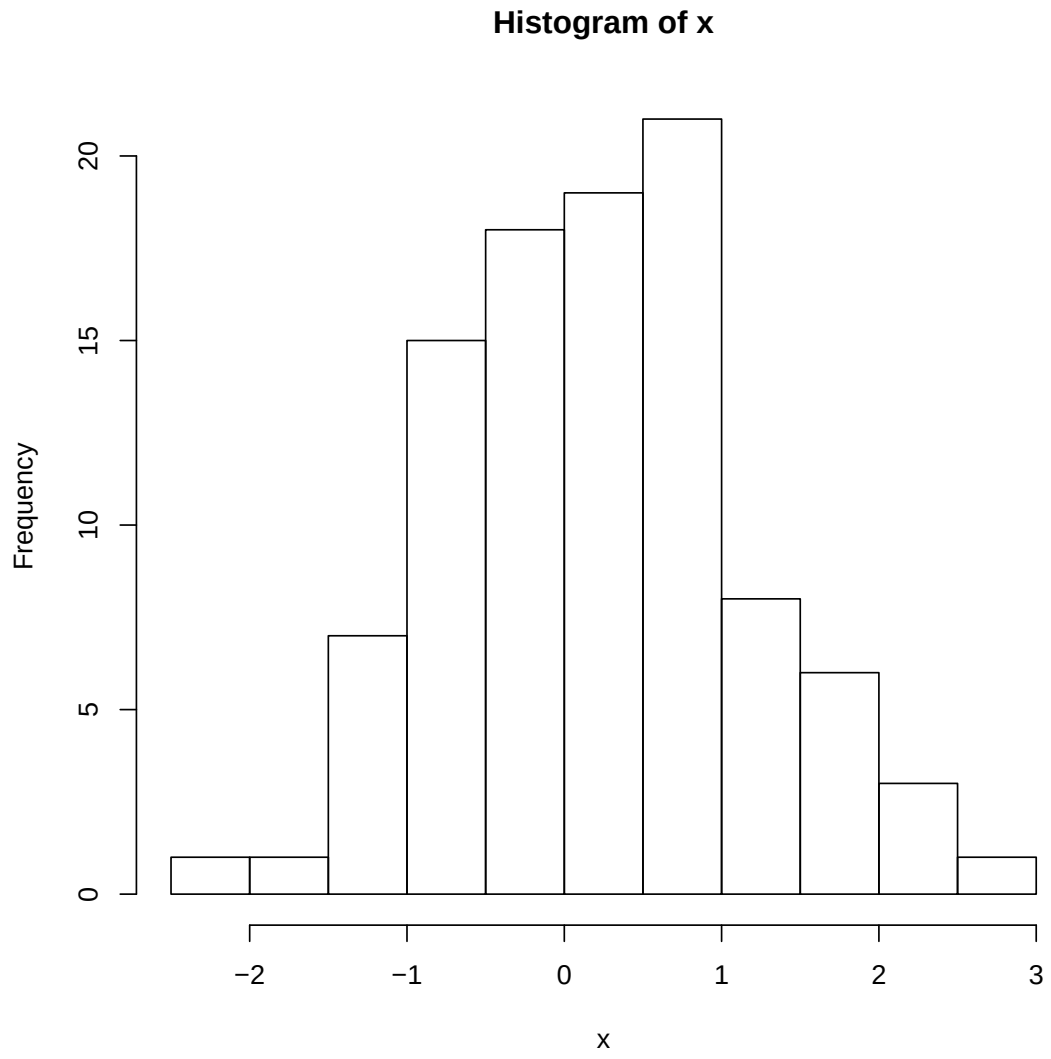
3. From the graphs above, m_2 appears to be a better model fit. One peculiar thing appears to be the enormous level of overlap between the two conditions predicted by x in y_1.

Table 1:

	Dependent variable:	
	y1	y2
	(1)	(2)
x	0.996***	1.440***
	(0.084)	(0.109)
Constant	0.061	1.441***
	(0.069)	(0.097)
Observations	1,000	1,000
Log Likelihood	−602.437	−448.444
Akaike Inf. Crit.	1,208.875	900.889
Note:	*p<0.1; **p<0.05; ***p<0.01	

3 Likelihood Exploration

Looks vaguely normalish, no?



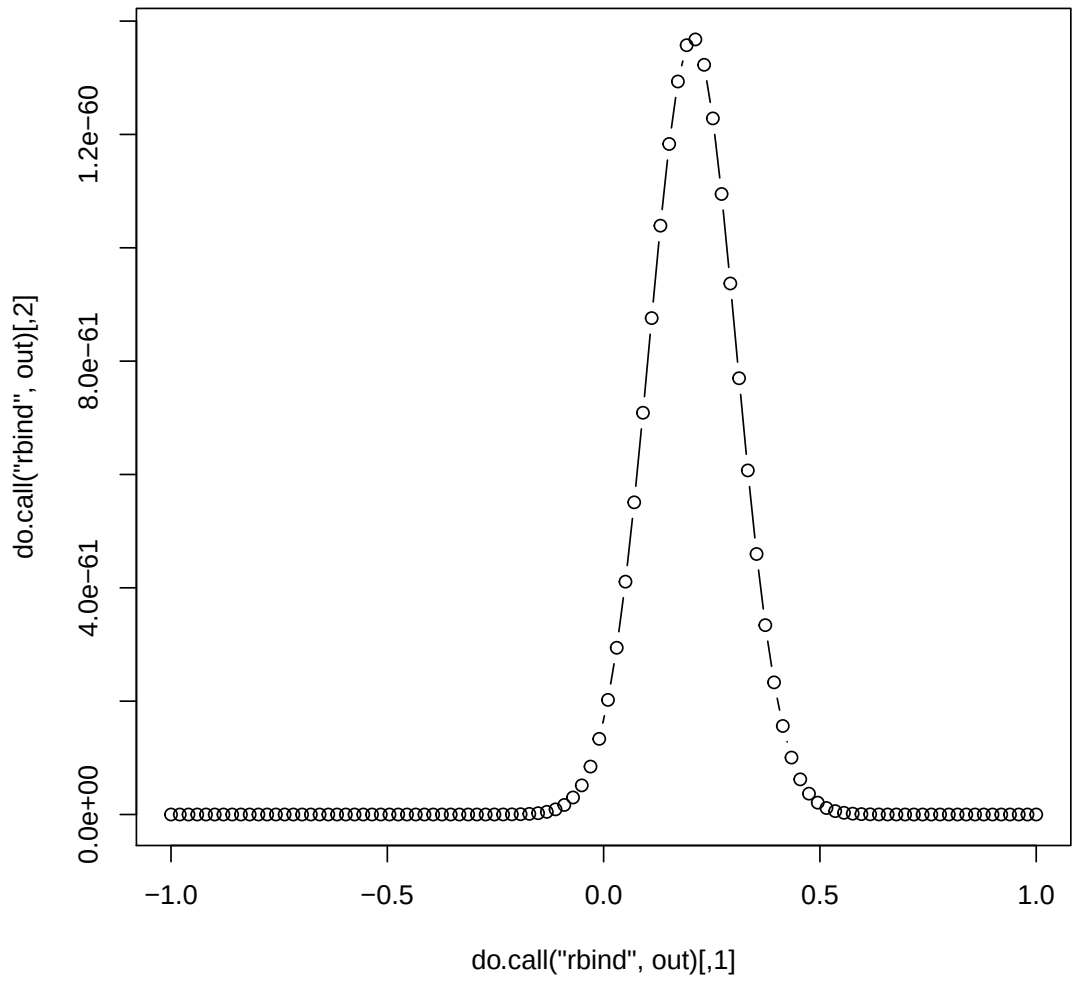
Yes, it does.

```
likelihood<-function(pars,x) {
    tmp<-exp(-(x-pars[1])^2/(2*pars[2]))
    tmp/sqrt(2*pars[2]*pi)
}
```

Let's think about what we get if the mean is unknown and the SD=1 Q. what do you notice?

```
out<-list()
```

```r
for (m in seq(-1,1,length.out=100)) {
    like<-rep(NA,length(x))
    for (i in 1:length(x)) {
        like[i]<-likelihood(c(m,1),x[i])
    }
    c(c(m,prod(like)))->out[[as.character(m) ]]
}
plot(do.call("rbind",out),type="b")
```



The slice of the likelihood surface shown here has a maximum value at our best estimate of the mean. It's also worth noting that the values for likelihood are really tiny, which is a good reason to use log-likelihood in computational applications.

Q. How do our estimates vary in accuracy as a function of the sample size (change 100 to something much bigger and much smaller in the top line)

```r
ll<-function(pars,x) {
    likelihood<-function(pars,x) {
        tmp<-exp(-(x-pars[1])^2/(2*pars[2]))
        tmp/sqrt(2*pars[2]*pi)
    }
    like<-rep(NA,length(x))
    for (i in 1:length(x)) {
        like[i]<-likelihood(pars,x[i])
    }
    -1*sum(log(like))
}
#these are the mean and variance estimates produced by MLE.
optim(par=c(-2,2),ll,x=x)$par

## [1] 0.2058405 0.9189130
```

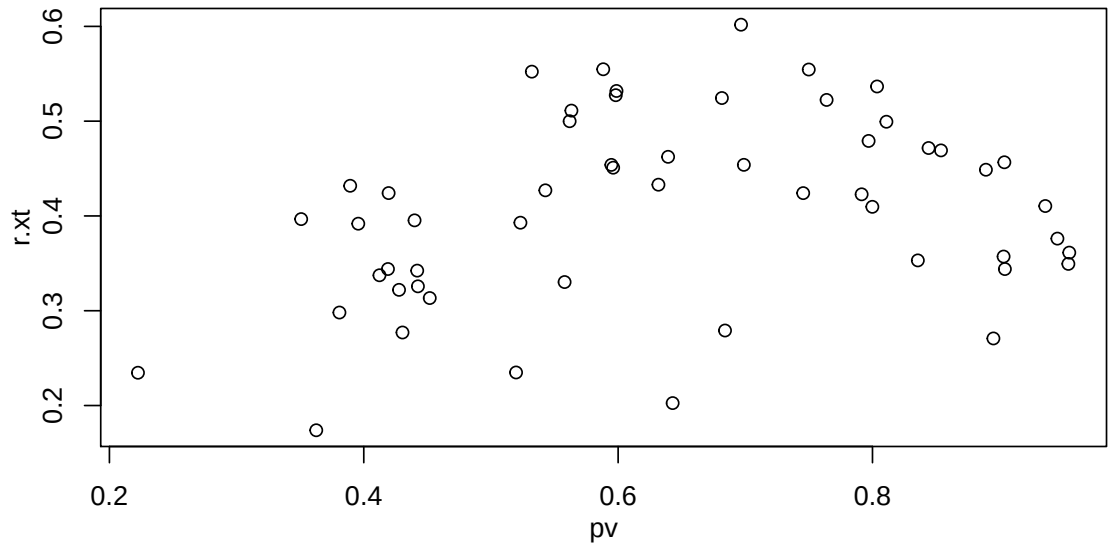
As the sample size increases, the mean approaches zero and the variance approaches one.

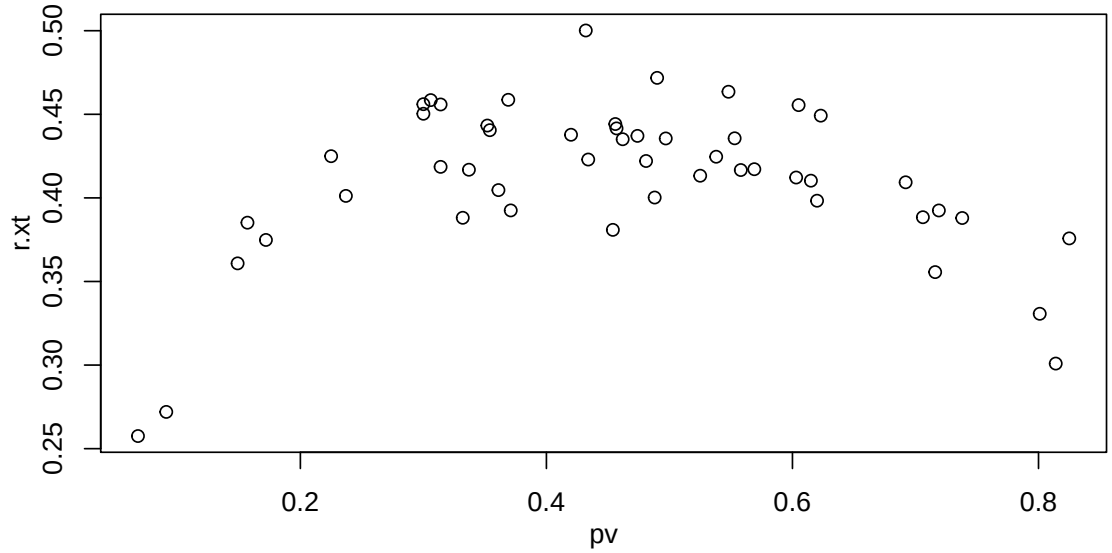
Sample Size	Mean	Variance
10	0.2335989	1.3643526
100	0.1675501	0.9539125
1000	-0.02798667	0.95039213
10000	0.0014444	1.0368603

4 Item Quality

Consider the item statistics (p-values & item-total correlations) discussed in the Crocker & Algina text. What do you think? As a point of contrast, consider them vis-a-vis the item statistics generated by this data:

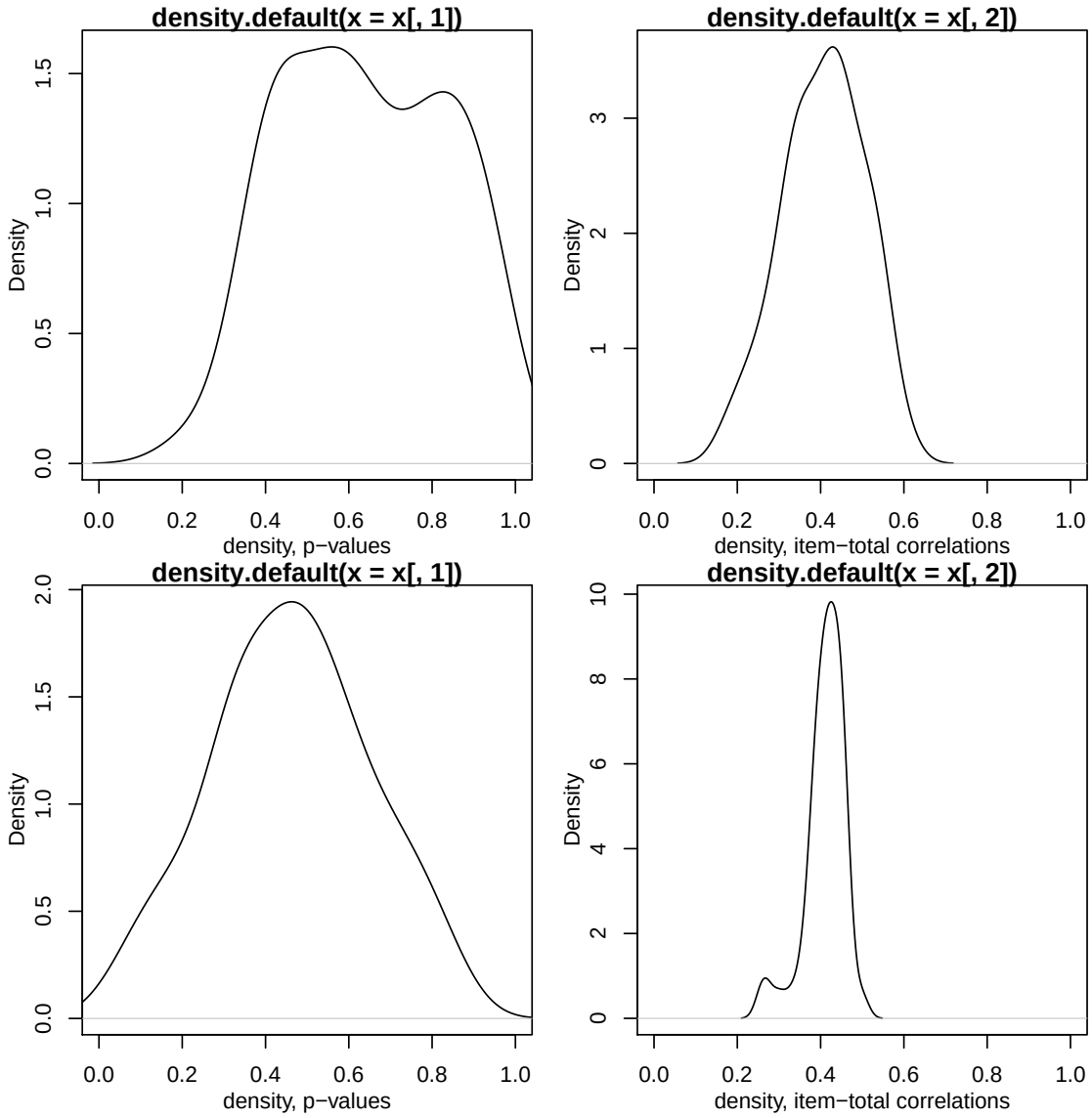
```
out<-list()

##CTT item analysis
##this function will compute CTT item statistics
item_analysis<-function(resp) {
    pv<-colMeans(resp,na.rm=TRUE)
    r.xt<-numeric()
    rowSums(resp,na.rm=TRUE)->ss
    for (i in 1:ncol(resp)) {
        cor(ss,resp[,i],use='p')->r.xt[i]
    }
    #return matrix of the p-values and the item/total correlations
    cbind(pv,r.xt)
}

resp1 <-read.table("emp-rasch.txt",header=FALSE)
out[[1]]<-item_analysis(resp1)

resp2 <-read.table("rasch.txt",header=FALSE)
out[[2]]<-item_analysis(resp2)

par(mfrow=c(2,1),mgp=c(2,1,0),mar=c(3,3,1,1))

plot(out[[1]])
plot(out[[2]])
```





77

```
par(mfrow=c(2,2),mgp=c(2,1,0),mar=c(3,3,1,1))
pf<-function(x) {
    plot(density(x[,1]),xlim=c(0,1),xlab="density, p-values")
    plot(density(x[,2]),xlim=c(0,1),xlab="density, item-total correlations")
}
lapply(out,pf)
```



The simulated data set (bottom row of graphs) has a fairly normal distribution of item difficulty, while the empirical set (top row of graphs) essentially only consists of items that are middling difficulty or very easy. When you look to the discrimination of the items (from the item-total correlation graphs), the simulated data set has a very consistently middling discrimination that is nearly the same for almost all items, while the empirical data set, once again, has a wider range of values, with a small number of more discriminating items and some items that are hardly discriminating at all.

The interpretation, then, is that the empirical data set does not represent a "good" instrument. There are not a variety of item difficulties and the items themselves are not discriminating.

5 Buffon's Needle

```
n <- 1000 #number of needles
d <- 1 #line spacing
l <- 1 #needle length

#matrix to hold all of the information about the needles
orient <- matrix(NA,n,4)
#generates the position of the lefthand side of the needle
orient[,1] <- runif(n,0,d)
#generates the angular orientation of the needle
orient[,2] <- runif(n,-pi/2, pi/2)
#determines the horizontal span of the needle
orient[,3] <- orient[,1]+l*cos(orient[,2])
#checks the location of the righthand side of the needle
orient[,4] <- orient[,3] >= d
#determines the probability that a needle crossed a line
p <- sum(orient[,4])/n
p

## [1] 0.632
```

The approach I took here was based upon the following logic:

1. This is a 1-dimensional problem. The only thing that actually matters is the projection of the length of a particular needle onto a line perpendicular to the lines on the surface.

2. It's only worth considering two lines, separated by a distance d.

3. Each needle lands with its leftmost edge some distance from that line on the left and makes some angle, θ, with the line we are projecting its length onto.

4. Since we always reference the leftmost edge of the needle, we only need to consider angles between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$.

5. If the distance between the leftmost edge and the line plus the projection of the length of the needle onto the perpendicular line is greater than the separation between the parallel lines, the needle will overlap a line.