# Problem Set 1

Iris Zhong

1/13/2022

## Question 1 Exploration of collections of bernoulli variables

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

Let's pretend that x1 is item response data from a test. So 1s and 0s are correct/incorrect responses (rows are people and columns are items).

For fun we can look at the correlations across items and the variation in row sums (ie, total scores)

```
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
```

```r
var(rowSums(x1))
```

```
## [1] 2.706667
```

## 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? How might we know?

No. The correlations between the items are very weak (very close to 0). From the data generating process, it's clear that 1s and 0s are generated randomly, both with 50% of probability. In a realistic scenario, the numbers should not be random, and items should have stronger correlations, as people doing well in one item should somewhat indicate that they could do well in other items too.

> Feel free to ignore this chunk of code (skip ahead to below question). I'm going to generate a new set of data.

```r
set.seed(12311)
th<-matrix(rnorm(100),100,10,byrow=FALSE)
diff<-matrix<-matrix(rnorm(10),100,10,byrow=TRUE)
kern<- exp(th - diff)
pr<-kern/(1+kern)
test<-matrix(runif(1000),100,10)
x2<-ifelse(pr>test,1,0)
```

```r
cor(x2)
```

```
##             [,1]       [,2]       [,3]        [,4]       [,5]        [,6]
##  [1,] 1.00000000 0.05225302 0.31098566  0.08849268 0.17610919  0.11892066
##  [2,] 0.05225302 1.00000000 0.26505302  0.03714521 0.12156613  0.19791667
##  [3,] 0.31098566 0.26505302 1.00000000  0.17914327 0.17914327  0.06001200
##  [4,] 0.08849268 0.03714521 0.17914327  1.00000000 0.05582923  0.13169665
##  [5,] 0.17610919 0.12156613 0.17914327  0.05582923 1.00000000  0.08104409
##  [6,] 0.11892066 0.19791667 0.06001200  0.13169665 0.08104409  1.00000000
##  [7,] 0.17435686 0.25832804 0.22615825  0.10837438 0.06732348  0.17221869
##  [8,] 0.07143616 0.13262503 0.05803086 -0.01959216 0.06204183 -0.04029115
##  [9,] 0.11133735 0.42075805 0.11903823  0.21353066 0.05114993  0.08014439
## [10,] 0.12808759 0.08071308 0.14553789  0.19806502 0.10679082  0.13514748
##             [,7]        [,8]        [,9]       [,10]
##  [1,] 0.17435686  0.07143616  0.11133735  0.12808759
##  [2,] 0.25832804  0.13262503  0.42075805  0.08071308
##  [3,] 0.22615825  0.05803086  0.11903823  0.14553789
##  [4,] 0.10837438 -0.01959216  0.21353066  0.19806502
##  [5,] 0.06732348  0.06204183  0.05114993  0.10679082
##  [6,] 0.17221869 -0.04029115  0.08014439  0.13514748
##  [7,] 1.00000000  0.14204314  0.15182598  0.21266889
##  [8,] 0.14204314  1.00000000  0.33582739  0.13068593
##  [9,] 0.15182598  0.33582739  1.00000000 -0.01399044
## [10,] 0.21266889  0.13068593 -0.01399044  1.00000000
```

```r
var(rowSums(x2))
```

```
## [1] 5.111111
```

**Q. Now, let's ask the same question of the new matrix x2. Does it seem like realistic item response data? 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 influence your opinion on this point?**

Since the correlations are substantially higher, it looks more like a realistic item response dataset.

**Q. How would you characterize the key difference between x1 and x2 in terms of what we can observe if we blind ourselves to the data generating process?**

The size of correlation coefficients.

## Question 2 Logistic Regression

```r
load(here("data","ps1-logreg.Rdata"))
```

```r
m1 <- glm(y1 ~ x,df, family = "binomial")
m2 <- glm(y2 ~ x,df, family = "binomial")
```

```r
summary(m1)
```

```
##
## Call:
## glm(formula = y1 ~ x, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.0732  -1.0230   0.3943   1.0042   2.3292
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  0.06109    0.06940    0.88    0.379
## x            0.99636    0.08424   11.83   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1386.1  on 999  degrees of freedom
## Residual deviance: 1204.9  on 998  degrees of freedom
## AIC: 1208.9
##
## Number of Fisher Scoring iterations: 3
```

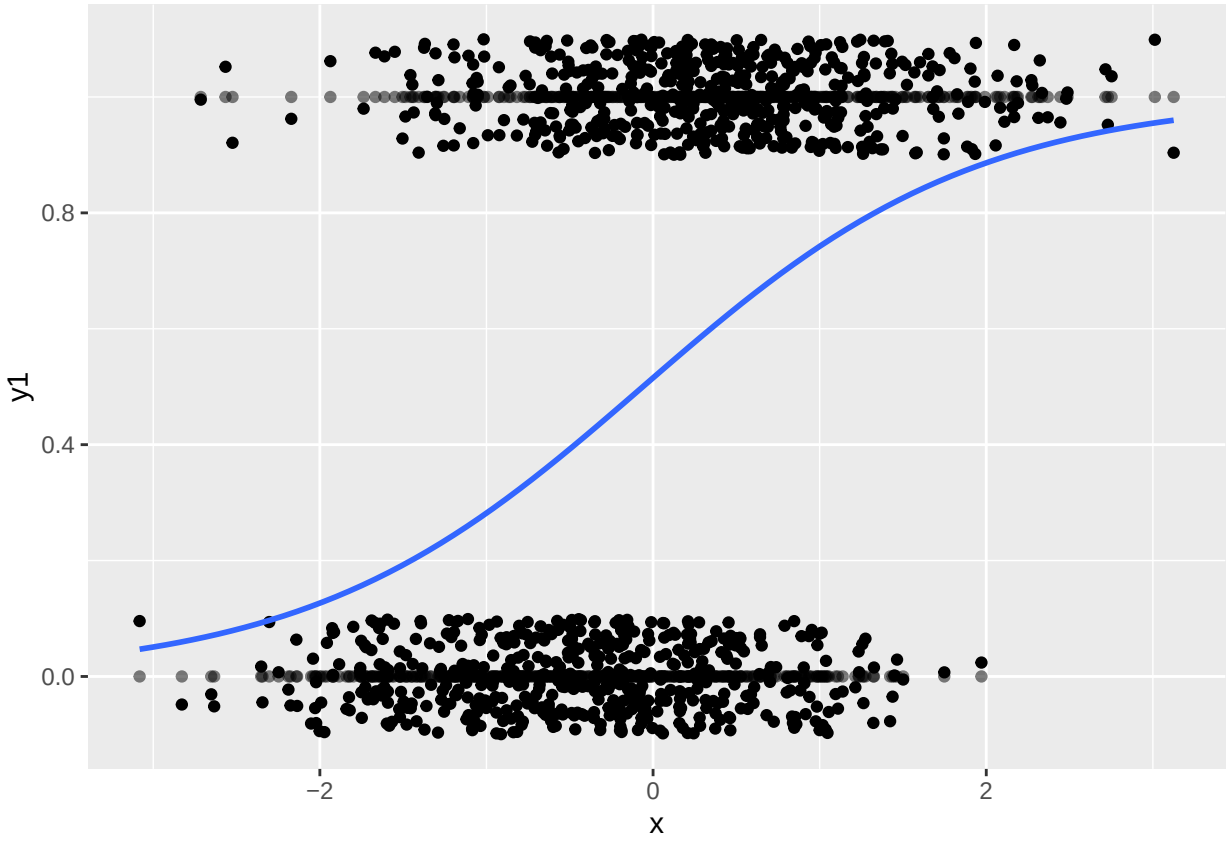
```r
summary(m2)
```

```
##
## Call:
## glm(formula = y2 ~ x, family = "binomial", data = df)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8981  -0.6553   0.4267   0.6887   2.2202
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  1.44146    0.09658   14.93   <2e-16 ***
## x            1.43951    0.10921   13.18   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1152.34  on 999  degrees of freedom
## Residual deviance:  896.89  on 998  degrees of freedom
## AIC: 900.89
##
## Number of Fisher Scoring iterations: 5
```

### (A) How would you compare the association between y1 or y2 & x?

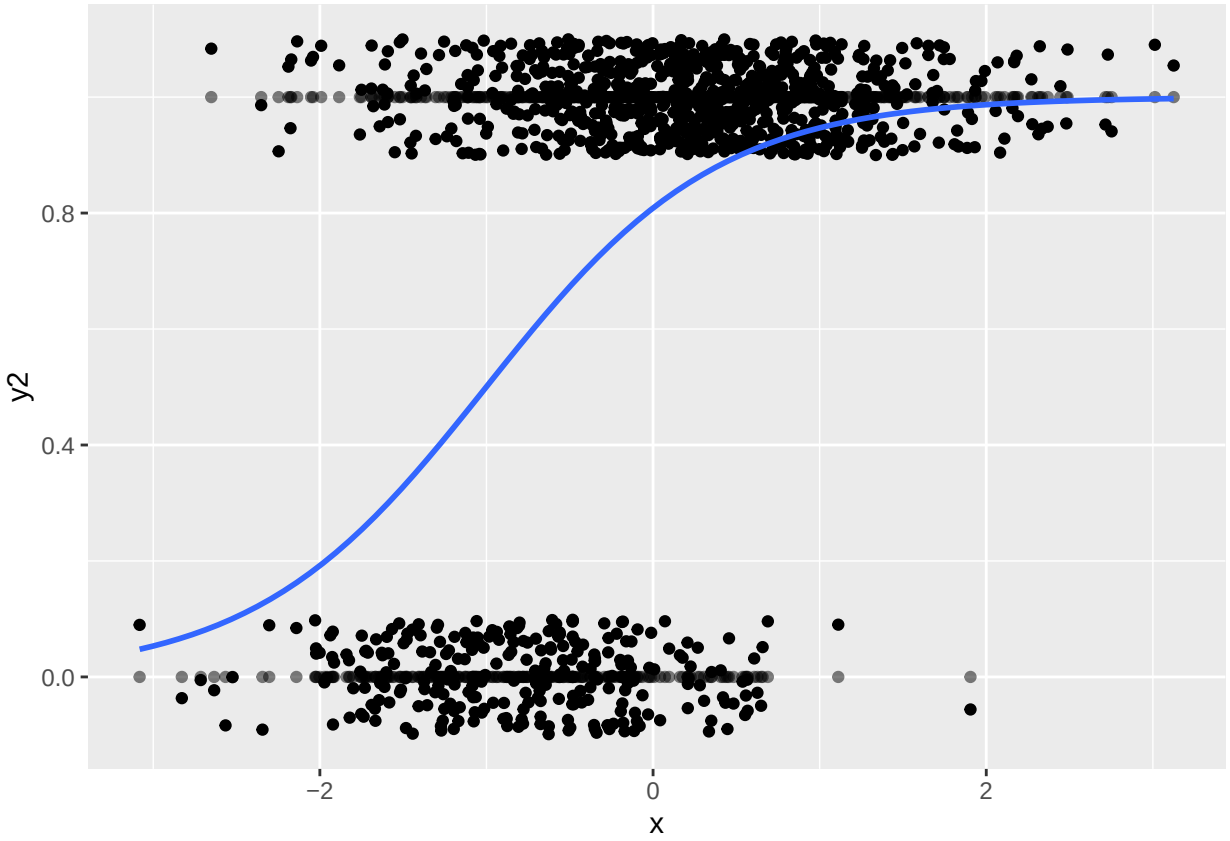
```r
ggplot(df, aes(x = x, y = y1)) +
  geom_point(alpha=.5) +
  geom_jitter(height = 0.1) +
  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




```
ggplot(df, aes(x = x, y = y2)) +
  geom_point(alpha=.5) +
  geom_jitter(height = 0.1) +
  stat_smooth(method="glm", se=FALSE, method.args = list(family=binomial))
```

```
## `geom_smooth()` using formula 'y ~ x'
```




The intercept in model 1 is smaller than that in model 2, so a prediction of y = 0 is more prevalent in model 1.

The curve of the regression line is flatter in model 1 and steeper in model 2, which is also evident from the slope estimates in the models.

## (B) How would you interpret the regression coefficients from (say) m1?

For m1: odds ratio = $e^{\hat{\beta}_1} = 2.7084053$

When x increases by 1 unit, the odds of y is 2.71 times as large as before.

## (C) 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)? [Note: This one is sneaky. I'd encourage you to avoid fit statistics and look at techniques for model diagnostics (e.g., residuals).]

From the plots, I'd predict that m2 has a better fit. In m1, it looks like there are more instances where the data points with same x values have different y values.

Check confusion matrix:

```
m1.prob = predict(m1, df[1], type="response")
m1.prob[m1.prob > .5] <- 1
m1.prob[m1.prob <= .5] <- 0
table(m1.prob, df$y1)
```

```
##
## m1.prob   0   1
##       0 329 155
##       1 164 352
```

```
m2.prob = predict(m2, df[1], type="response")
m2.prob[m2.prob > .5] <- 1
m2.prob[m2.prob <= .5] <- 0
table(m2.prob, df$y2)
```

```
##
## m2.prob   0   1
##       0 110  62
##       1 153 675
```

From the confusion matrix, m2 has a higher predicted accuracy. In particular, m1 has substantially more cases when it predicts 0 but the actual value is 1.

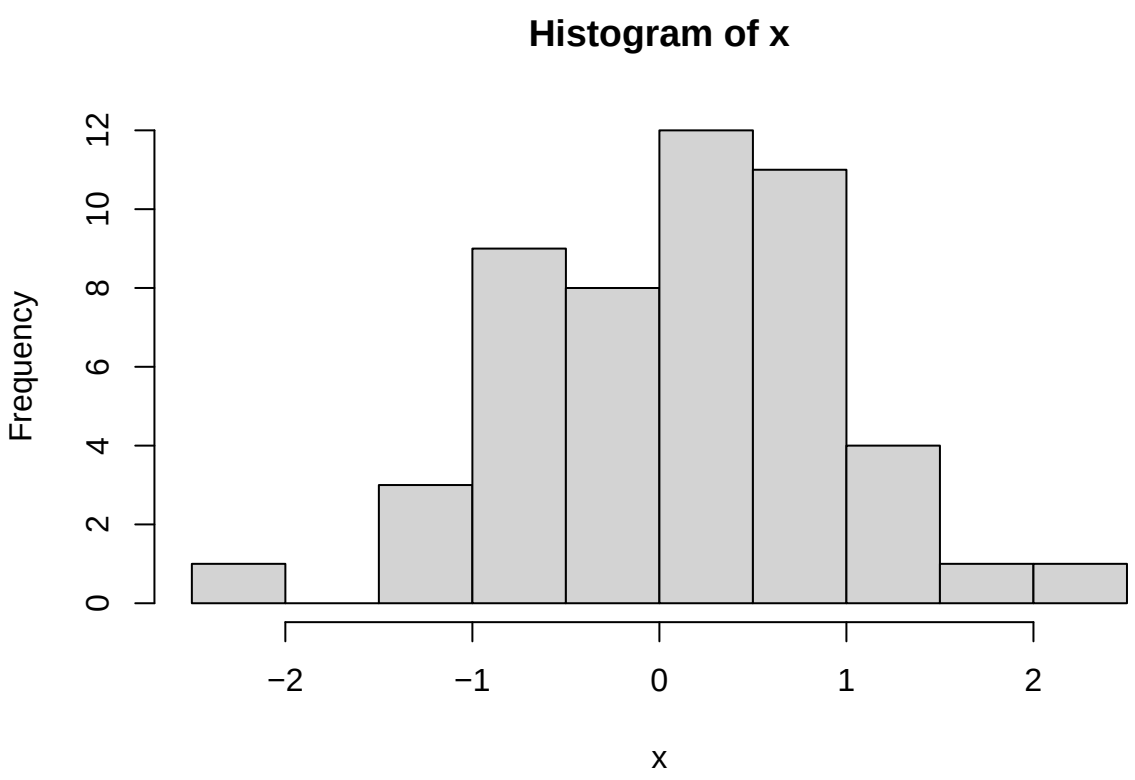
# Q3 Likelihood exploration

Suppose we just observed x, a bunch of random numbers.

```
x <- rnorm(50)
```

We first want to see what the distribution looks like. We can do this:

```
hist(x)
```




Looks vaguely normalish, no? [Of course, you can see that I'm drawing variates from the normal distribution, so this isn't surprising. Pretend you didn't see that part!]

So what if we wanted to estimate the mean and var of the normal distribution that may have generated this set of draws. How do we do this? The key statistical technique is to consider the likelihood. Let's start by writing a function that computes the likelihood for "x" in a normal with unknown mean and var (collectively, "pars").

```
likelihood <- function(pars,x) { #see the first eqn here, http://mathworld.wolfram.com/NormalDistributi
  # pars[1] = mu, pars[2] = sigma
  # prob of x in a normal dist that has a mean of mu and sd of sigma
    tmp <- exp(-(x-pars[1])^2/(2*pars[2]))
    tmp / sqrt(2*pars[2]*pi)
}
```

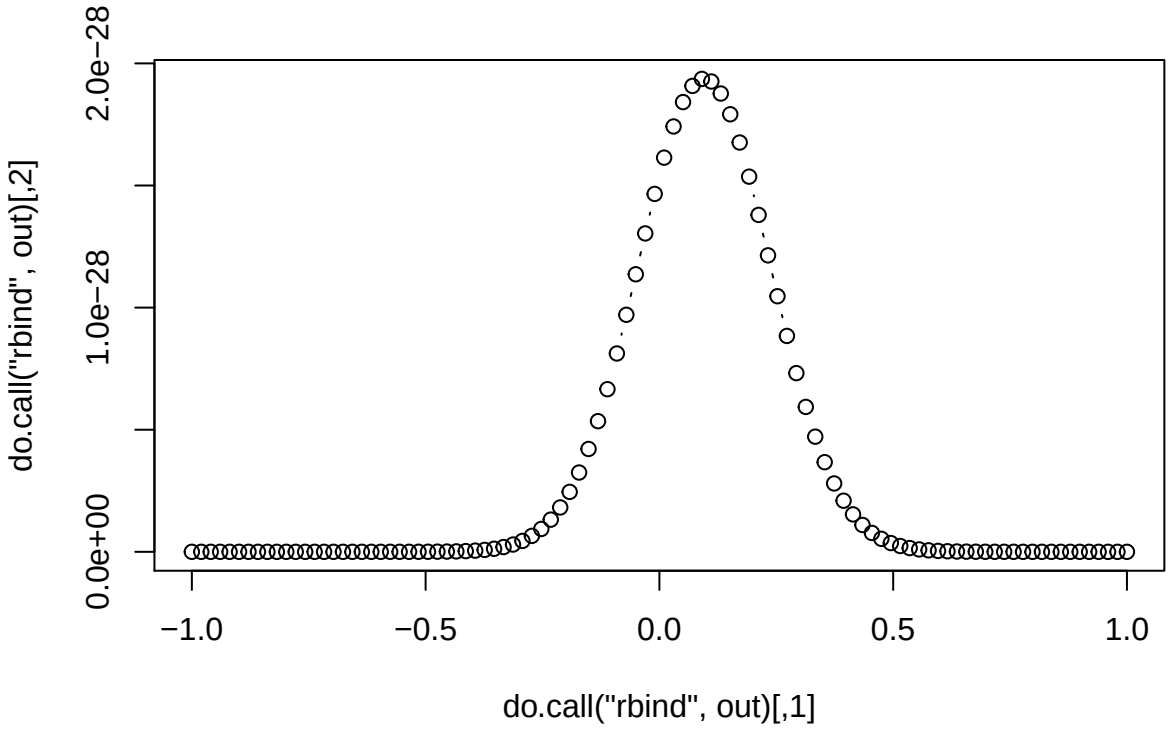
To completely evaluate this function, we would need to know x and pars. We only know x (this is the problem of estimation in general: the values in pars are not known to us!).

With known x, we can think of the likelihood as the "probability" of observing the draws x from a normal distribution with parameters pars.

That is, we are thinking of the likelihood as a function of pars (x is known).

Let's think about what we get if the mean is unknown and the SD = 1

```r
out<-list()
character_m = list()
for (m in seq(-1,1,length.out=100)) {
    like<-rep(NA,length(x)) # empty list
    for (i in 1:length(x)) { # for each x:
        like[i]<-likelihood(c(m,1),x[i]) # likelihood of x_i when mean = m and sd = 1
    }
    character_m <- append(character_m, as.character(m))
    c(c(m,prod(like)))->out[[as.character(m)]] # likelihood of x when mean = m and sd = 1
}
plot(do.call("rbind",out),type="b") #this is a likelihood surface where we're seeing the likelihood as
```




### Q. what do you notice?

The likelihood of x is the highest when it is under a normal distribution with a mean of approximately 0 (which is our true mean).

> From a computational perspective, working with the products of small numbers is very unstable. So we instead work with the sum of the logs.
>
> Why is this ok? First of all, log(xy)=log(x) + log(y). Second, log(f(x)) is a monotic transformation of f(x). So if we maximize log(f(x)) funtion, then we've also maximized f(x) ##Below is a function that will do this.

```
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))
}
optim(par=c(-2,2),ll,x=x)$par #these are the mean and variance estimates produced by maximum likelihood
```

```
## [1] 0.09563549 0.71446064
```

**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 call to "rnorm" at the top)? What does this connect to in your understanding of estimation theory (think standard error)?**

When the sample size gets larger, the estimates are more accurate. Since $s = \frac{\sigma}{\sqrt{n}}$, when the sample size increases, standard error should decrease, and we are more certain about our estimates.

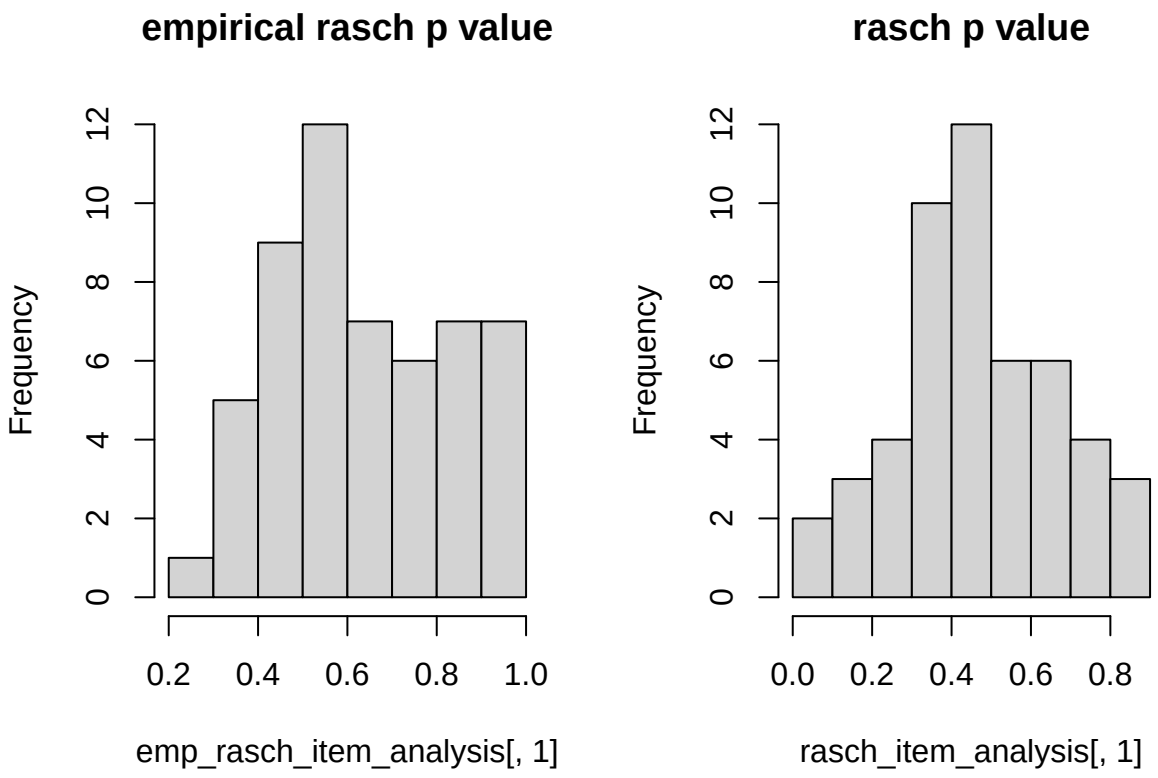
# Q4 Item Quality

```
emp_rasch <- read.delim(here("data","emp-rasch.txt"), sep = " ", header = F)
rasch <- read.delim(here("data","rasch.txt"), sep = " ", header = F)
```

```
# from in class code
item_analysis<-function(resp) { #'resp' is just a generic item response matrix, rows are people, column
    pv<-colMeans(resp,na.rm=TRUE) #simple "p-values", which in psychometrics tends to just mean the mea
    r.xt<-numeric() #initializing a vector
    rowSums(resp,na.rm=TRUE)->ss #these are the sum scores/observed scores
    for (i in 1:ncol(resp)) {
        cor(ss,resp[,i],use='p')->r.xt[i] #this is the correlations between the i-th item (resp[,i]) an
    }
    cbind(pv,r.xt) #returning a matrix consisting of the p-values and the item/total correlations
}
```

```
emp_rasch_item_analysis <- item_analysis(emp_rasch)
rasch_item_analysis <- item_analysis(rasch)
```

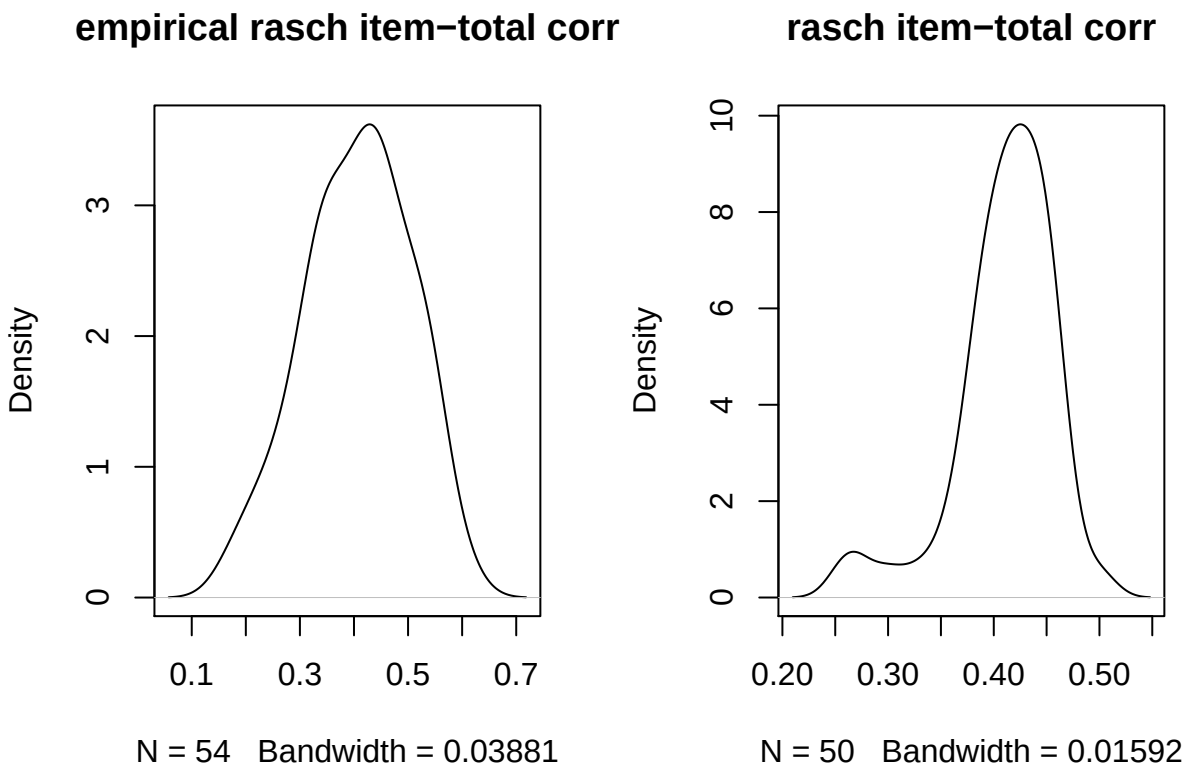
```
par(mfrow=c(1,2))
hist(emp_rasch_item_analysis[,1], main = "empirical rasch p value")
hist(rasch_item_analysis[,1], main = "rasch p value")
```




The empirical rasch data has most items with a p-value of approximately 0.5. It looks less normal than the rasch data, with more items that are on the difficult end (high p-value). Its p-value ranges from about 0.2 to 1.0.

The rasch data has a more normal distribution and centers at around 0.5. It also has a wider range of p-value from 0 to 1.

```
par(mfrow=c(1,2))
plot(density(emp_rasch_item_analysis[,2]), main = "empirical rasch item-total corr")
plot(density(rasch_item_analysis[,2]), main = "rasch item-total corr")
```




Item-total correlation in the empirical rasch dataset has a nice bell-shaped distribution, ranging from about 0.1 to 0.7. Since no item-total correlation is close to 0, the items are acceptable.

Item-total correlation distribution of the rasch dataset has a smaller range from 0.2 to 0.55. There is also a small bump at about 0.25. Similarly, because no item-total correlation is close to 0, the items are acceptable.

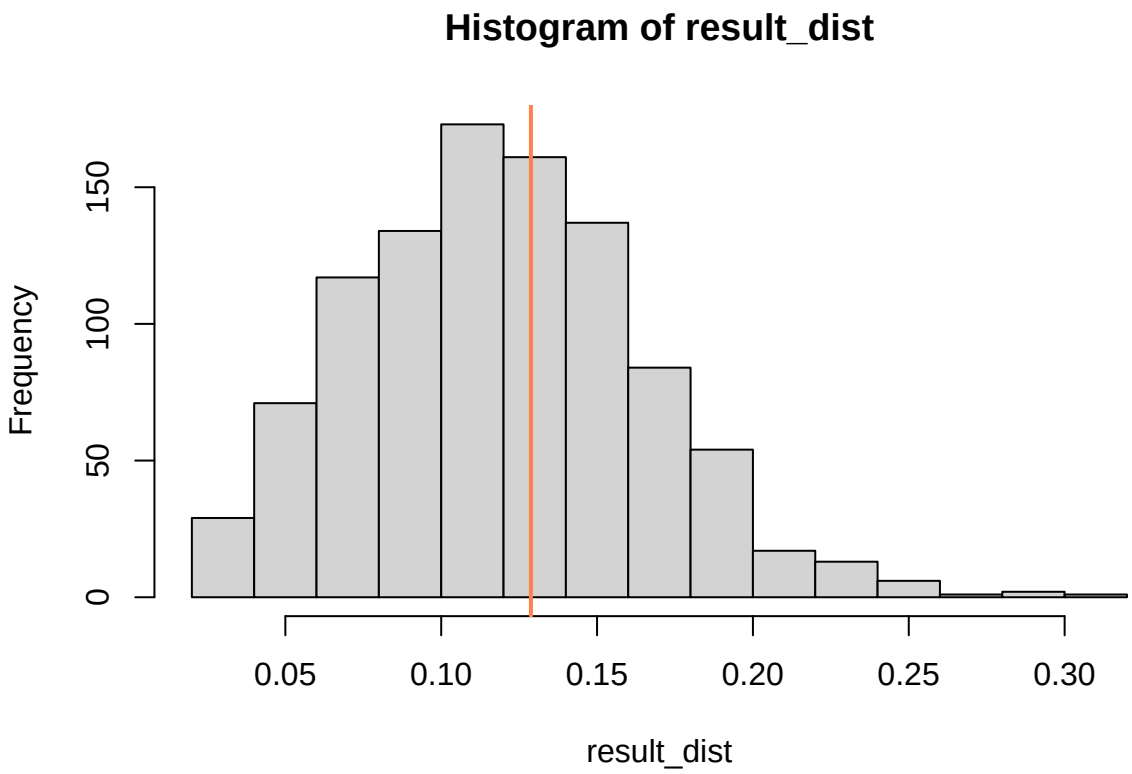
## Bonus Question

```
d <- 5 # distance
l <- 1 # length of needle
simulate_n <- 1000 # simulate how many times?
sample_n <- 50 # sample size in each simulation?
```

```
intersect_function <- function(x) {

  if (min(x1, x2) <= x & max(x1, x2) >= x) {return(1)}
  else {return(0)}

}
```

```
result_dist <- c()
set.seed(252)

for (i in 1:simulate_n) {
```

```
  result_sample <- c() # list of 0 and 1 for each j
  for (j in 1:sample_n){
    result_temp <- c()

    # create coordinates for the needle ends

    x1 <- runif(1, -100, 100) # a random number from -100 to 100
    y1 <- runif(1, -100, 100)
    theta <- runif(1, 0, pi) # a random angle
    x2 <- x1 + l * cos(theta) # find the other coordinate
    y2 <- y1 + l * sin(theta)

    # create parallel lines
    initial_line <- runif(1, -120, -110)
    line_list <- seq(from = initial_line, to = -initial_line, by = d) # create list of parallel lines

    # check intersection
    intersect_yes <- sapply(line_list, intersect_function) # for each line, does it intersect with the
    if (sum(intersect_yes) != 0) {
      result_temp <- append(result_temp, 1)
    }

    else {result_temp <- append(result_temp, 0)} # whether j intersects

    result_sample <- append(result_sample,result_temp)

  }
result_dist <- append(result_dist,mean(result_sample))

}

hist(result_dist)
abline(v = mean(result_dist),col = "coral", lwd = 2)
```




```
mosaic::favstats(result_dist)
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##   min  Q1 median   Q3  max    mean         sd    n missing
##  0.02 0.1   0.12 0.16 0.32 0.12878 0.04661206 1000       0
```