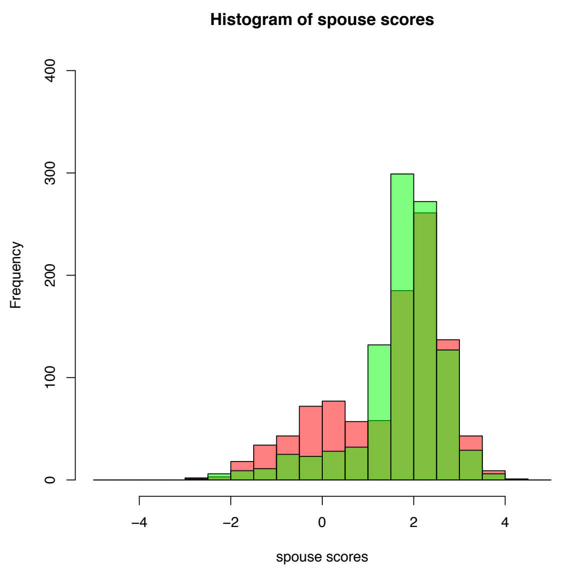
# Question 1
# One of the advantages of using a simulation to model your system is that you have access to all information about the system. In the above analyses, we only considered the mean scores for each n. For each of the two strategies considered above, prepare a histogram of the scores at the optimal n. Does this new information influence which strategy you might adopt or recommend?

# My answer:

```
# step1: determine the optimal n by simulation (assume I don't know the optimal n is 1/e*N for strategy 1)
# for strategy 1:
simulate.dating.1 <- function(N = 100, n = N/exp(1), niter = 1000, mean = TRUE) {
  spouse.score <- numeric(length = niter)
  for(case.idx in 1:niter) {
    score <- rnorm(N)
    optimal.score<- max(score)
    cutoff.score <- max(score[1:n])
    spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]

    if (is.na(spouse.index)) {
      spouse.index <- N
    }

    spouse.score[case.idx] <- (score[spouse.index] == optimal.score)
  }
  return(mean = mean(spouse.score))
}

trials <- 1:100
optimals.1 <- numeric(length(trials))
for(i in trials) {
  sampled <- seq(5, 95, 2)
  means.1 <- numeric(length = length(sampled))

  for(idx in 1:length(sampled)) {
    means.1[idx] <- simulate.dating.1(N=100, n = sampled[idx], niter = 1000, mean = TRUE)
  }

  optimals.1[i] <- sampled[which.max(means.1)]
}

optimal.n.1 <- mean(optimals.1) # the value is around 36.26

# for strategy 2:
simulate.dating.2 <- function(N = 100, n = N/exp(1), niter = 1000, mean = TRUE) {
  spouse.score <- numeric(length = niter)
  for(case.idx in 1:niter) {
    score <- rnorm(N)
    optimal.score<- max(score)
    cutoff.score <- max(score[1:n])
    spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]
    if (is.na(spouse.index)) {
      spouse.index <- N
    }

    spouse.score[case.idx] <- score[spouse.index]
  }
  return(mean = mean(spouse.score))
}

trials <- 1:100
optimals.2 <- numeric(length(trials))
for(i in trials) {
  sampled <- seq(5, 95, 2)
  means.2 <- numeric(length = length(sampled))

  for(idx in 1:length(sampled)) {
    means.2[idx] <- simulate.dating.2(N=100, n = sampled[idx], niter = 1000, mean = TRUE)
  }

  optimals.2[i] <- sampled[which.max(means.2)]
}

optimal.n.2 <- mean(optimals.2) # the value is around 14.96

# step2: plot the histograph:
hist.dating <- function(N = 100, n, niter = 1000) {
    spouse.scores <- numeric(length = niter)
    for (case.idx in 1:niter) {
      score <- rnorm(N)
      optimal.score <- max(score)
      cutoff.score <- max(score[1:n])
      spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]
      if (is.na(spouse.index)) {
        spouse.index <- N
      }
      spouse.scores[case.idx] <- score[spouse.index]
    }
    return(spouse.scores)
}
hist(hist.dating(100,optimal.n.1,niter = 1000),breaks=seq(-5,5,0.5),col=rgb(1,0,0,0.5),ylim=c(0,400),
main=paste("Histogram of", "spouse scores"), xlab = "spouse scores")# red
hist(hist.dating(100,optimal.n.2,niter = 1000),breaks=seq(-5,5,0.5),col=rgb(0,1,0,0.5),add=T)#green
```



#Summary: the 1st strategy scores are shown in red. It's slightly more likely to get an very high score(>2.5), but the scores are more spreading out, including more low scores(<0). Using the 2nd strategy (green bars) is more likely to get a decent score(between 1.0 and 2.5) and less likely to get a low score.
# Generally speaking, although using 1st strategy is more likely to find a soul mate with optimal score, but the chance to have a low score is also higher.Using the 2nd strategy is more likely to find someone with a decent score, less likely to get a low score.

###

# Question2: The simulation we described in class assumes that the candidates come from a normal distribution, with a standard deviation of 1. For each of the strategies, do the results change if the standard deviation is different? What about if the distribution is different, e.g., a uniform or poisson distribution? Can you reason out why?

# My answer:
# For strategy 1, changing the SD or changing the scores to uniform distribution won't change the optimal score; changing the scores to poisson distribution will change the optimal score. Because the density plots of normal distribution and uniform distribution are symmetric, and it won't change the optimal n when we consider the chance of getting the optimal score as the criteria for success.

# For strategy 2, changing the SD won't change the optimal score, but changing to poisson/uniform distribution will change the optimal score, because the chance to get a relatively decent score is changed. It changes the optimal n when we use the absolute score of the spouse as the criteria to determine n.

# This can be confirmed by simulating as well.

```
# The following is the coding for strategy 1 with different distributions
simulate.dating.1 <- function(N = 100, n = N/exp(1), niter = 1000, mean = TRUE) {
  spouse.score <- numeric(length = niter)
  for(case.idx in 1:niter) {
     score <- rnorm(N) # the optimal n is  36.26
    # for normal distribution with sd = 100
    # score <- rnorm(N, sd = 100) # the optimal n is 37.18
    # for poisson distribution
    # score <- rpois(100, 0.25) # the optimal n is 10.46
    # for uniform distribution
    # score <- runif(100, -1, 1) # the optimal n is 38.02
    optimal.score<- max(score)
    cutoff.score <- max(score[1:n])
    spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]

    if (is.na(spouse.index)) {
      spouse.index <- N
    }

    spouse.score[case.idx] <- (score[spouse.index] == optimal.score)
  }
  return(mean = mean(spouse.score))
}

trials <- 1:100
optimals <- numeric(length(trials))
for(i in trials) {
  sampled <- seq(5, 95, 2)
  means <- numeric(length = length(sampled))

  for(idx in 1:length(sampled)) {
    means[idx] <- simulate.dating.1(N=100, n = sampled[idx], niter = 1000, mean = TRUE)
  }

  optimals[i] <- sampled[which.max(means)]
}
optimal.n <- mean(optimals)

# The following is the coding for strategy 2 with different distributions
simulate.dating.2 <- function(N = 100, n = N/exp(1), niter = 1000, mean = TRUE) {
  spouse.score <- numeric(length = niter)
  for(case.idx in 1:niter) {
     score <- rnorm(N) # the optimal n is 14.96
    # for normal distribution with sd = 100
    # score <- rnorm(N, sd = 100) # the optimal n is 14.94
    # for poisson distribution
    # score <- rpois(100, 0.25) # the optimal n is 8.00
    # for uniform distribution
    #score <- runif(100, -1, 1) # the optimal n is 9.44
    optimal.score<- max(score)
    cutoff.score <- max(score[1:n])
    spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]

    if (is.na(spouse.index)) {
      spouse.index <- N
    }
    spouse.score[case.idx] <- score[spouse.index]
  }
  return(mean = mean(spouse.score))
}

trials <- 1:100
optimals <- numeric(length(trials))
for(i in trials) {
  sampled <- seq(5, 95, 2)
  means <- numeric(length = length(sampled))

  for(idx in 1:length(sampled)) {
    means[idx] <- simulate.dating.2(N=100, n = sampled[idx], niter = 1000, mean = TRUE)
  }

  optimals[i] <- sampled[which.max(means)]
}

optimal.n <- mean(optimals)

###########################################################################################
##########
```

# Question 3: One of the assumptions in the classic statement of this problem is that the candidate will always accept when you propose to them. How would you change the model to include the possibility of rejection? Do you think that a constant probability of rejection is a good model? It may be that if you find somebody very compatible, chances are good that they like you a lot too, and vice versa. How might you model the case where the chance of rejection is proportional to how much you like them?

# My answer:
# I'd like to use the first strategy, and consider it's a success only when a soulmate is found. The chance to be accepted fits binomial distribution. The acceptance rate is positively proportional to the score, i.e. the person is more likely to accept me when I like him a lot too.

```
simulate.dating.1.1 <- function(N = 100, n = N/exp(1), niter = 1000) {
  spouse.score <- numeric(length = niter)
  for(case.idx in 1:niter) {
    score <- rnorm(N)
    optimal.score<- max(score)
    cutoff.score <- max(score[1:n])
    spouse.index <- n + which(score[n+1:N] > cutoff.score)[1]

    # flip a coin to mimic the chance to be rejected, the probability of success is positively proportional to the
score.
    i <- 0
    while (!is.na(spouse.index)) {
      coinflip <- rbinom(1,1, prob = exp(score[spouse.index])/(2*exp(optimal.score)))
      if(coinflip == 1) {break}
    if(coinflip == 0) {spouse.index <- n + which(score[n+1:N] > cutoff.score)[2+i]}
    i <- i+1
  }
  spouse.index <- n + which(score[n+1:N] > cutoff.score)[1+i]

  if(is.na(spouse.index)){
    spouse.index <- N}
  # if the last person rejects me, I don't care, because it's considered a failure anyway.

  spouse.score[case.idx] <- (score[spouse.index] == optimal.score)
 }
 return(mean(spouse.score))
}

trials <- 1:100
optimals <- numeric(length(trials))
for(i in trials) {
 sampled <- seq(5, 95, 2)
 means <- numeric(length = length(sampled))

 for(idx in 1:length(sampled)) {
  means[idx] <- simulate.dating.1.1(N=100, n = sampled[idx], niter = 1000)
 }

 optimals[i] <- sampled[which.max(means)]
}
optimal.n <- mean(optimals) # the optimal n is 24.68

# The result shows that when considering the chance to be rejected, we should decrease the training pool size (n).
```