

Data Generation

Yuchen Li (li215), section 1UG

April 14, 2019

Contents

Team Powers	1
Standard Probability Distributions	1
Accept-Reject	1
Inverse CDF	2
Cross-Team Winning Probabilities	2
Seeding	3
Examples	3
Normal(0, 1) team powers	3
Normal(10, 0.01) team powers	3
Beta(3, 4) team powers using Accept-Reject	4
Exp(1) team powers using Inverse CDF	5

Team Powers

Standard Probability Distributions

Normal

```
genNormalPowers <- function(n, mean=0, sd=1) {  
  # INPUT:  
  # n is the number of teams  
  
  # OUTPUT:  
  # returns a vector of team powers, sorted in decreasing order  
  powers <- rnorm(n, mean, sd)  
  return(sort(abs(powers), decreasing=TRUE))  
}
```

Accept-Reject

```
# Reference: adapted from Yuchen Li (li215), HW2, Exercise 4  
  
acceptReject <- function(nsim, f, min, max, M) {  
  # INPUT:  
  # nsim is the number of simulations  
  # f is the target distribution  
  # min is the min value in the domain of f  
  # max is the max value in the domain of f
```

```

# max
# M >= sup{f(x)}

# OUTPUT:
# returns a vector of random variates sampled from f, using the
# Accept-Reject method with Unif(min, max) as the reference distribution
k1 = 0          # counter for accepted samples
j1 = 0          # number of iterations required to get desired sample size
y1 = numeric(nsim) # storing the sample
while(k1 < nsim){
  u = runif(1)
  x = runif(1, min, max) # random variate from reference distribution
  g1 = 1
  if (u < f(x) / M / g1) {
    # condition of accepting x in our sample
    k1 = k1 + 1
    y1[k1] = x
  }
  j1 = j1 + 1
}
return(sort(y1, decreasing=TRUE))
}

```

Inverse CDF

```

inverseCDF <- function(n, inv_cdf) {
  # INPUT:
  # n is the number of simulations
  # inv_cdf is the inverse CDF function for f

  # OUTPUT:
  # returns a vector of random variates sampled from PDF f,
  # using the Inverse CDF method
  u = runif(n)
  y = numeric(n)
  for (i in 1:n) {
    y[i] = inv_cdf(u[i])
  }
  return(sort(y, decreasing=TRUE))
}

```

Cross-Team Winning Probabilities

```

genCrossTeamWinningProbabilities <- function(powers) {
  # INPUT:
  # powers is the teams powers

  # OUTPUT:
  # returns an n x n matrix M where M_{ij} is the probability of team-i beating team-j
  n = length(powers)

```

```

probs = matrix(nrow=n, ncol=n)
for (i in 1:n) {
  for (j in 1:n) {
    probs[i,j] = powers[i] / (powers[i] + powers[j])
  }
}
return(probs)
}

```

Seeding

What are the other good methods than random selection? (In the data generation part, we do not have actual competition data yet.)

```

# Example
sample(1:8, size=2)

```

```
## [1] 1 8
```

Examples

Normal(0, 1) team powers

```

genCrossTeamWinningProbabilities(
  genNormalPowers(4)
)

```

```

##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.5000000 0.6846414 0.7788225 0.8540777
## [2,] 0.3153586 0.5000000 0.6186054 0.7294361
## [3,] 0.2211775 0.3813946 0.5000000 0.6243681
## [4,] 0.1459223 0.2705639 0.3756319 0.5000000

```

Normal(10, 0.01) team powers

Note the probabilities are closer to 0.5

```

genCrossTeamWinningProbabilities(
  genNormalPowers(4, mean=10, sd=0.1)
)

```

```

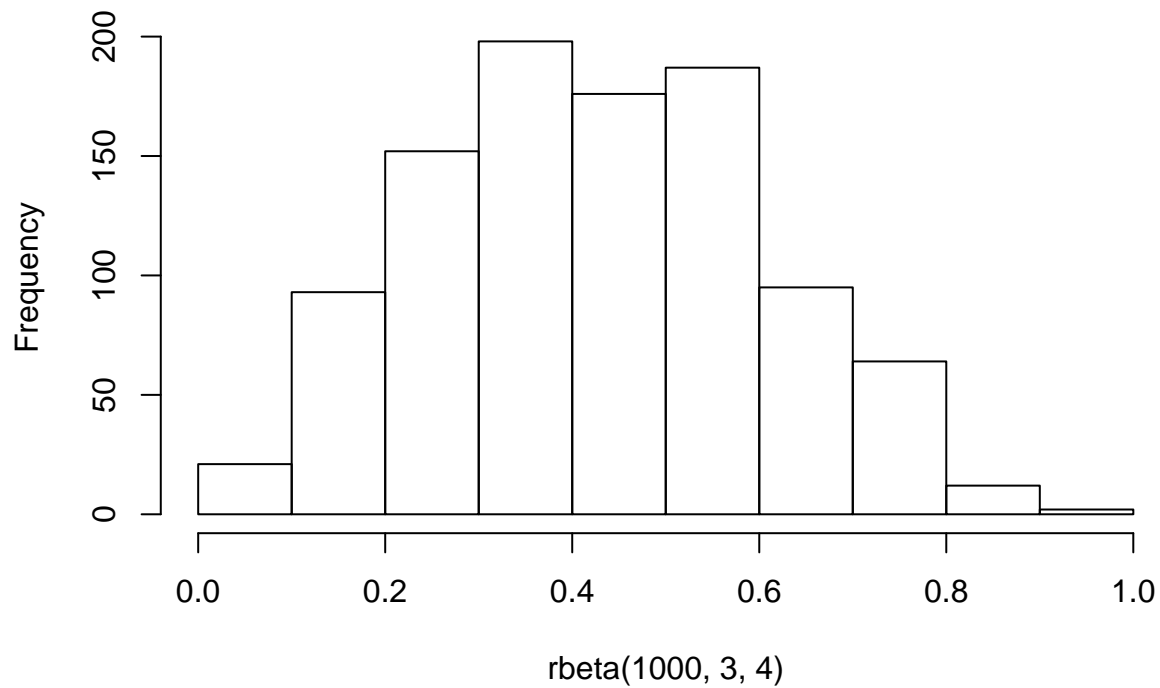
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.5000000 0.5003264 0.5017373 0.5029095
## [2,] 0.4996736 0.5000000 0.5014109 0.5025831
## [3,] 0.4982627 0.4985891 0.5000000 0.5011722
## [4,] 0.4970905 0.4974169 0.4988278 0.5000000

```

Beta(3, 4) team powers using Accept-Reject

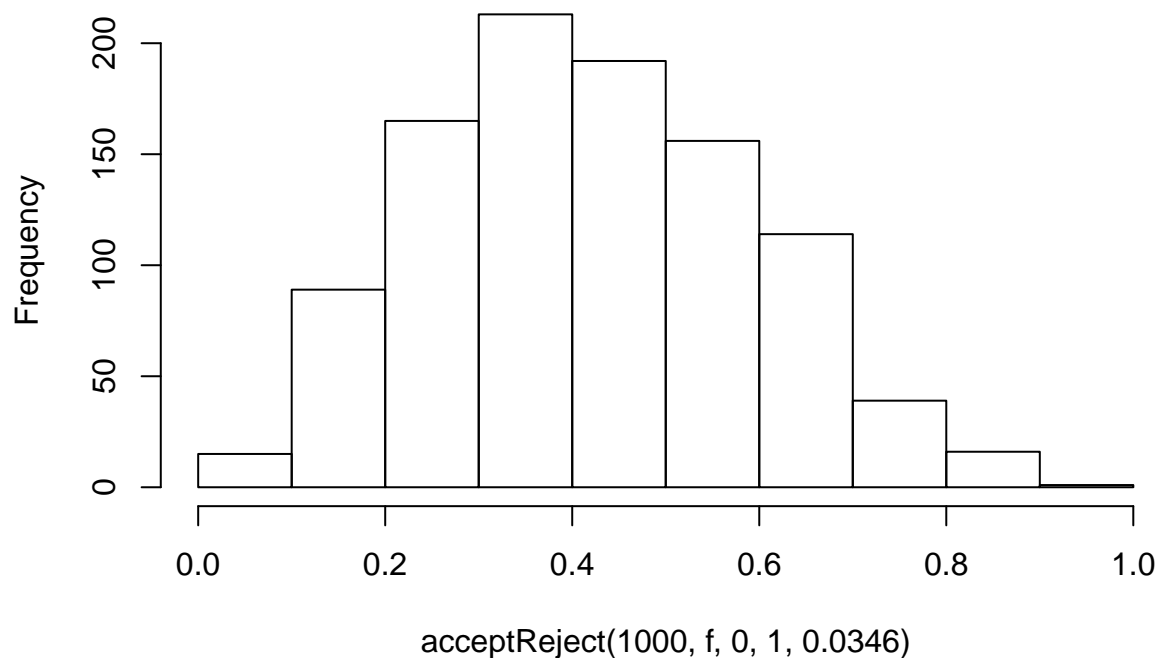
```
# Test `acceptReject`  
f <- function(x) {return(x^2 * (1-x)^3)}  
hist(rbeta(1000, 3, 4))
```

Histogram of rbeta(1000, 3, 4)



```
hist(acceptReject(1000, f, 0, 1, 0.0346))
```

Histogram of acceptReject(1000, f, 0, 1, 0.0346)



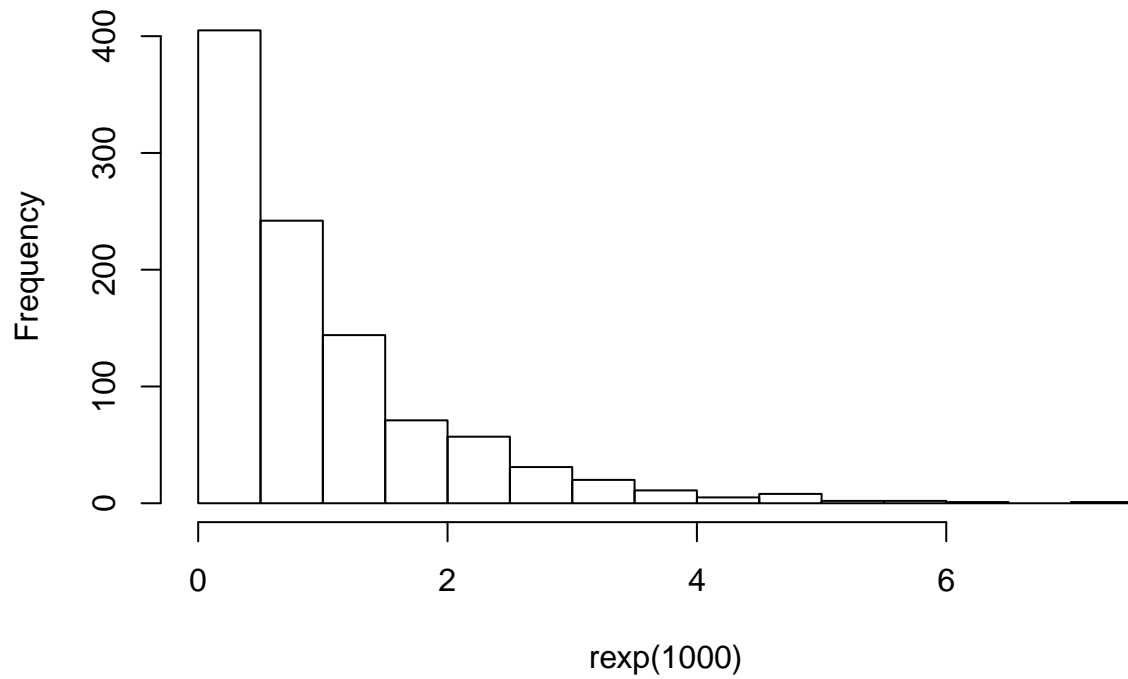
```
# Actual
genCrossTeamWinningProbabilities(
  acceptReject(4, f, 0, 1, 0.0346)
)

##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.5000000 0.5090574 0.5766003 0.6392323
## [2,] 0.4909426 0.5000000 0.5677309 0.6308349
## [3,] 0.4233997 0.4322691 0.5000000 0.5654230
## [4,] 0.3607677 0.3691651 0.4345770 0.5000000
```

Exp(1) team powers using Inverse CDF

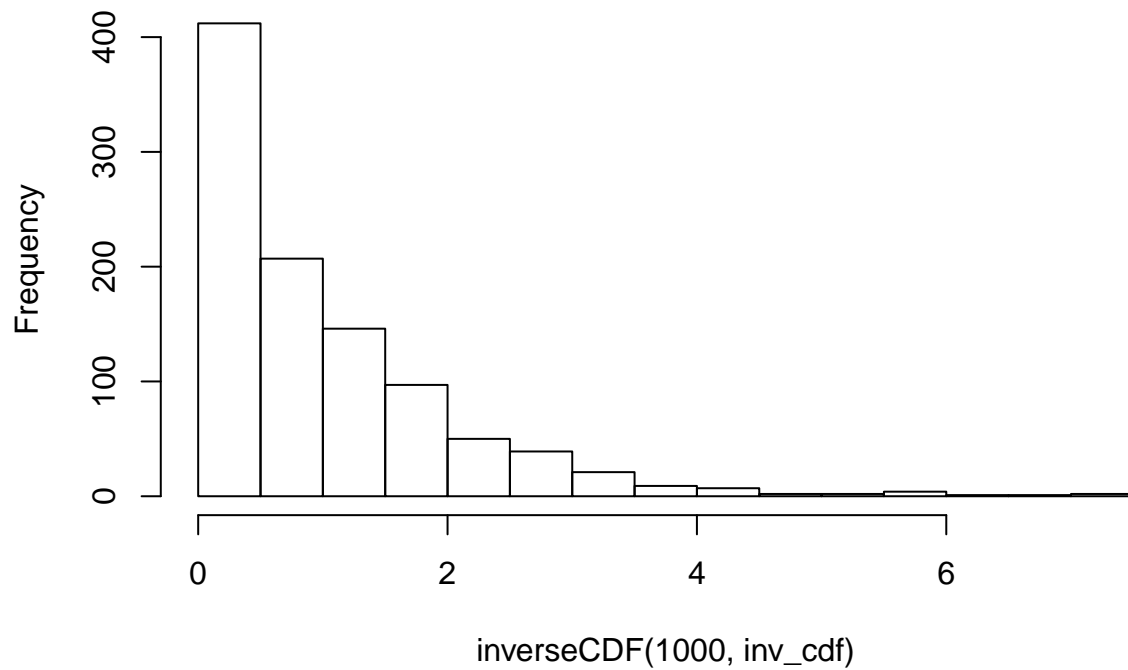
```
# Test `inverseCDF`
inv_cdf <- function(x) {return(-log(x))}
hist(rexp(1000))
```

Histogram of rexp(1000)



```
hist(inverseCDF(  
  1000,  
  inv_cdf  
)  
)
```

Histogram of inverseCDF(1000, inv_cdf)



```
# Actual
genCrossTeamWinningProbabilities(
  inverseCDF(
    4,
    inv_cdf
  )
)
```

```
##           [,1]      [,2]      [,3]      [,4]
## [1,] 0.5000000 0.5724035 0.6638494 0.7029643
## [2,] 0.4275965 0.5000000 0.5960015 0.6387147
## [3,] 0.3361506 0.4039985 0.5000000 0.5451164
## [4,] 0.2970357 0.3612853 0.4548836 0.5000000
```