# An Investigation into Racial Profiling in an Airport Setting

**INTRODUCTION**

By definition, racial profiling is an actuarial method that conditions an individual's prior probability of criminal behavior explicitly on his or her race, ethnicity, nationality, or religion (Press, 2010). Some think this is an effective method of catching criminals but other think using these stereotypes is unethical and prejudice. Racial profiling is controversial especially when we are unsure if it is even effective.

**BACKGROUND INFORMATION**

The idea for this project came from the article, "To Catch a Terrorist: Can Ethnic Profiling Work?" by William Press. He defines two different methods of selecting passengers for secondary screening within an airport setting. The two methods are uniform random sampling and importance sampling. Uniform random sampling is assuming that every passenger has the same probability of being a security threat. Using this method, a number of passengers are randomly selected for a secondary screening based upon allocated resources. Importance sampling uses a prior probability of being a security threat for each passenger. In the context of our problem, the prior probability could be based on many different things including race, religion, name, or even how they paid for their ticket. Using this method, resources still only allow a certain number of passengers to be give a secondary screening but if a person is thought as twice as likely to be a security threat over another passenger, they will stop this passenger twice as often.

Press argues that in an airport setting, racial profiling or as he calls it importance sampling is no more effective that simply randomly selecting passengers at catching security threats. He measured their effectiveness by mathematically deriving the average number of checkpoints a security threat could get through without being caught.

Table 1- Effectiveness of Sampling Methods

| **Uniform Sampling** | **Importance Sampling** |
|---|---|

| | |
|---|---|
| $\mu = \sum_{i=1}^{N} \frac{p_i}{M/N} = \frac{N}{M}$ | $\mu = \sum_{i=1}^{N} \frac{p_i}{Mp_i} = \frac{N}{M}$ |

Where N equals numbers of passengers to pass through security checkpoint, $p_i$ is the individual prior probability for each passenger, and M is the amount of passengers based on limited resources that can have a secondary screening. This means for when resources allow for 1 out of every 7 passengers to be given a secondary screening, a security threat will on average get through 6 checkpoints without being stopped or equivalently will be caught on the $7^{th}$ checkpoint. This results in the same effectiveness for both methods.

**SIMULATION**

The main goal of this project was to create a function that would run each of the sampling methods on a data set to replicate the results of Press's article. Using R, I created a separate function for the different sampling methods that created a matrix of data, set up the screening method, ran multiple iterations, and then reported the results. The function needs the number of iterations, number of passengers, number of security threats, and the sample rates specified or can be run with the set values.

Table 2 – Code for Function Set Up

| Uniform Sampling | Importance Sampling |
|---|---|
| # num.iter: number of simulation<br># num.p: number of people<br># num.t: number of security threats<br># k: sample rate<br>uniform = function(num.iter=100, num.p=1000, num.t=10, k=5) | # num.iter: number of simulation<br># num.p: number of people<br># num.t: number of security threats<br># k: sample rate<br>significance = function(num.iter=100, num.p=1000, num.t=10, k=5) |

The code to create the matrix of data differed between the two functions. For the uniform function, the data matrix had two columns. The first column was an ID column that

went from 1 to number of passengers specified by the function. The second column was security threat status where 0 meant not a threat and 1 meant they were a security threat. To assign the security threat status to different passengers I used the sample function, which randomly selected ID's from the first column of the matrix based upon the number of security threat specified by the function. I then assigned those randomly selected ID numbers a 1 in the security threat column.

For the Importance function, the data matrix had three columns. The first twos columns were the same as above, the ID number and the security threat status denoted by 0 or 1. The third column was a prior probability. For my project I decided to select half to be at a higher risk of being a security threat, and have their probability of being a security threat be double so they would be selected twice as often. To do this I used the sample function to select half of the number passengers and assigned them the higher probability. Then using the sample function again but this time using their prior probability to select ID numbers to be security threat. I then assigned those selected ID numbers a 1 in the security threat column.

Table 3- Code for Setting Up Data Matrix

| **Uniform Sampling** | **Importance Sampling** |
|---|---|
| #creates simulation data, ID and security threat status<br>#(0:no security threat, 1:security threat)<br>x=cbind(1:num.p, rep(0,num.p))<br>whichrows=sample(1:num.p, num.t, replace = FALSE)<br>x[whichrows,2]=1 | #creates simulation data<br>#ID, security threat status, probability a threat based on race<br>#security threat status: (0:no security threat, 1:security threat)<br>x=cbind(1:num.p, rep(0,num.p), rep((2/(3*num.p)),num.p))<br>prob.t = sample(1:num.p, num.p/2, replace= FALSE)<br>x[prob.t,3]=(4/(3*num.p))<br>whichrows=sample(1:num.p, num.t, replace = FALSE, x[,3])<br>x[whichrows,2]=1 |

The next step after setting up the data matrix is to develop the screening processes. For both screening methods we set up a for loop, so it would continue to go through the cycle of

checkpoints. We also set up an if statement that checks to see if there are an 1's in the second column of the data matrix or equivalently if there are any security threats left. If there are security threats left in the data the loop continues, if not it stops. After this check it actually starts the screening process. It calculates how many people can be stopped at the current checkpoint by using the rate of sampling specified by the function. We then use the sample function to select ID's that will be given a secondary screening. For the uniform function it is done randomly but for the importance function the row of the data matrix with prior probabilities is used in this selection. We then create a new data matrix that does not contain the security threats that were caught during that checkpoint. This data matrix is only updated if a security threat is caught during that checkpoint. The function also creates and updates an output matrix. This matrix has two columns, the first being the number of the checkpoint, and the second being how many security threats were caught at that checkpoint. We use this output matrix to calculate the average checkpoint a security threat was caught at during this iteration.

Table 4 – Code for Screening Methods

| **Uniform Sampling** | **Importance Sampling** |
|---|---|
| #intializing matrix for each simulation<br>output.mat = cbind(rep(0,100), rep(0,100))<br><br>#looping through checkpoints for a single simulation<br>for( i in 1:100)<br>{<br><br>#checks to see if all security threats are caught<br>if (sum(newx[,2] > 0))<br>{<br><br>#selecting how many passengers to sample<br>p=round(dim(newx)[1]/k,0)<br><br>#selecting which passengers to sample<br>r=sample(1:dim(newx)[1], p, replace=FALSE)<br><br>#updates matrix with checkpoint numbers<br>output.mat[i,1]=i<br><br>#updates matrix with number of security threats caught each checkpoint | #intializing matrix for each simulation<br>output.mat = cbind(rep(0,100), rep(0,100))<br><br>#looping through checkpoints for a single simulation<br>for( i in 1:100)<br>{<br><br>#checks to see if all security threats are caught<br>if (sum(newx[,2] > 0))<br>{<br><br>#selecting how many passengers to sample<br>p=round(dim(newx)[1]/k,0)<br><br>#selecting which passengers to sample using significance sampling<br>r=sample(1:dim(newx)[1], p, replace=FALSE, newx[,3])<br><br>#updates matrix with checkpoint numbers<br>output.mat[i,1]=i |

```
      output.mat[i,2]=sum(newx[r,2])

      #updates newx if not all security threats are caught
      if(output.mat[i,2] > 0)
      {
        newx = newx[-r[newx[r,2]==1],]
      }
     }
   }
   #keeps only the checkpoint where security threats
were caught
   output.mat.keep = output.mat[output.mat[,1]>0,]

   #creates list of average number of checkpoint a
security got through without
   # being selected for each simulation
   avg.sim[j] =
sum(output.mat.keep[,1]*output.mat.keep[,2])/sum(x[,2
])

  }
```

```
      #updates matrix with number of security threats
caught each checkpoint
      output.mat[i,2]=sum(newx[r,2])

      #updates newx if not all security threats are caught
      if(output.mat[i,2] > 0)
      {
        newx = newx[-r[newx[r,2]==1],]
      }
     }
   }
   #keeps only the checkpoint where security threats
were caught
   output.mat.keep = output.mat[output.mat[,1]>0,]

   #creates list of average number of checkpoint a
security got through without
   # being selected for each simulation
   avg.sim[j] =
sum(output.mat.keep[,1]*output.mat.keep[,2])/sum(x[,2
])

  }
```

The function contains another for loop to run iterations to get reliable results. To report results the functions creates an array of the average checkpoint for each iteration. The array is updated after each iteration and is reported after the whole simulation. This array of averages can be used for analysis and to create graphs.

Table 5 –Code for Running Iterations and Reporting Results

| **Uniform Sampling** | **Importance sampling** |
|---|---|
| #initializing array for simulation mean output<br>avg.sim=rep(-1,num.iter)<br><br>#looping through each simulation<br>for(j in 1:num.iter)<br>{<br><br>(code for setting up data matrix)<br><br>(code for setting up screening method)<br><br>}<br><br>#gives the average of all the simulations<br>avg.uniform = sum(avg.sim)/num.iter | #initializing array for simulation mean output<br>avg.sim=rep(-1,num.iter)<br><br>#looping through each simulation<br>for(j in 1:num.iter)<br>{<br><br>(code for setting up data matrix)<br><br>(code for setting up screening method)<br><br>}<br><br>#gives the average of all the simulations<br>avg.sig = sum(avg.sim)/num.iter |

| | |
|---|---|
| #prints the average of all the simulations to use for statistical analysis<br>print(avg.uniform)<br><br>#returns an array of the averages for each simulation<br>return(avg.sim) | #prints the average of all the simulations to use for statistical analysis<br>print(avg.sig)<br><br>#returns an array of the averages for each simulation<br>return(avg.sim) |

**DISSCUSSION**

The main goal of this project is to replicate the results from the paper through simulation. To make sure the results are reliable, the examples will have varied samples rates and number of terrorists. The first example assumes 10,000 passengers for 365 iterations where 1 in 1000 is a security threat with the sample rate varied.

Table 6- Means for Example with Varied Sample Rate

| **When Resources Allow for 1 out of 7 to be Given a Secondary Screening** | | **When Resources Allow for 1 out of 10 to be Given a Secondary Screening** | |
|---|---|---|---|
| Method | Mean Checkpoint | Method | Mean |
| Theory | 7 | Theory | 10 |
| Uniform Sampling | 6.848 | Uniform Sampling | 10.03 |
| Importance Sampling | 7.035 | Importance Sampling | 10.01 |

Table 6 shows the average checkpoint that a security threat would be caught at when I out of 7 can be given a secondary screening is checkpoint 7 and when 1 out of 10 can be given a secondary screening is checkpoint 10. The results for both levels of resources seems to agree with the averages Press mathematically derived in his article. I also wanted to look into varying the number of security threats to see how the results compared. This example assumed 10,000 passengers for 365 iterations with a sample rate of 10 and varied number of threats.

Table 7- Means for Example with Varied Security Threats

| **Where 1 in 2000 is a security threat** | **Where 1 in 100 is a security threat** |
|---|---|

| Method | Mean Checkpoint | Method | Mean |
|---|---|---|---|
| Theory | 10 | Theory | 10 |
| Uniform Sampling | 9.971 | Uniform Sampling | 9.863 |
| Importance Sampling | 10.237 | Importance Sampling | 9.954 |

Table 7 shows that when the 1 out 10 people can be given a secondary screening, on average a security threat will be caught at the $10^{th}$ checkpoint even when the number of security threats changes. It seems that as long as the sample rate stays the same, they average security checkpoint a security threat is caught at will not vary when the number of security threats vary. These results seem to agree with Press's article.

Table 6 and 7 easily show that the means are similar across the methods of screening but they do not give any information about the distributions of the different methods. The probability distribution plots of the first example with varied sample rates will give us insight into the distributions, shown below in Figure 8.

Figure 8-PDF Plots for Examples with Varied Sample Rate

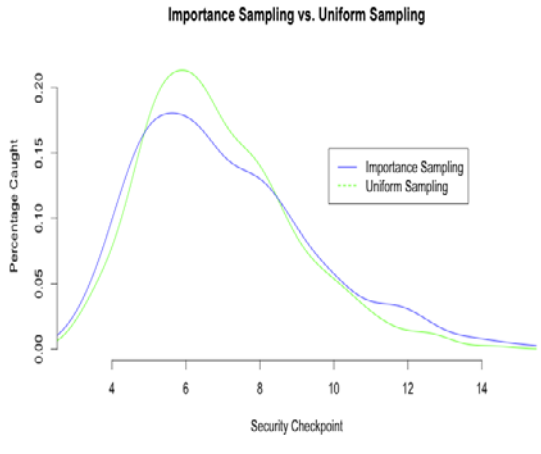
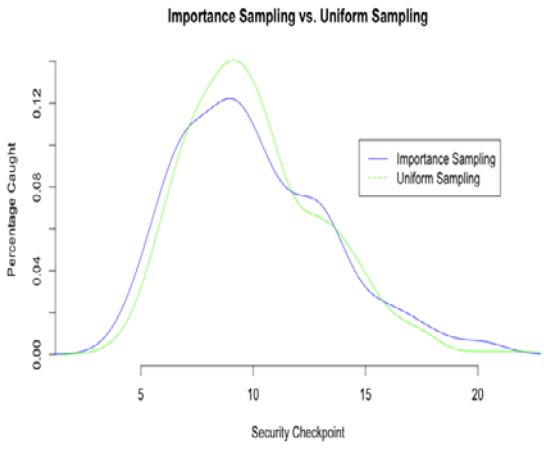
| **When Resources Allow for 1 out of 7 to be Given a Secondary Screening** | **When Resources Allow for 1 out of 10 to be Given a Secondary Screening** |
|---|---|
|  |  |

Figure 8 shows the percentage of security threats that were caught at each of the security checkpoints. The distributions for the screening methods follow a very similar trend. Both distributions are skewed right. This means the data peaks before the average and then has a gradual decrease and the highest percentage of security threats caught is actually a few checkpoints before the average. The plots also show that the uniform sampling distribution peaks much higher than the importance sampling distribution for both sample rates. Another way to compare the distributions is with cumulative distribution plots, shown below in figure 9.

Figure 9-CDF Plots of Example with Varied Sample Rate

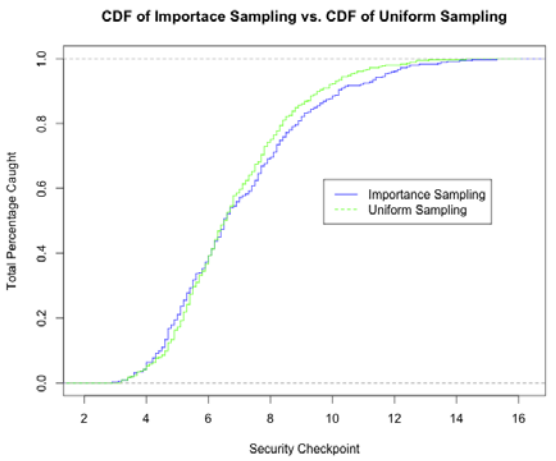
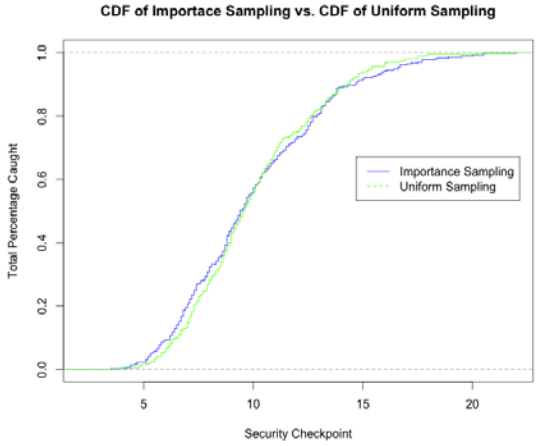
| When Resources Allow for 1 out of 7 to be Given a Secondary Screening | When Resources Allow for 1 out of 10 to be Given a Secondary Screening |
|---|---|
|  |  |





Figure 9 shows the total percentage of security threats that are caught before and at the current checkpoint. The graphs reiterate that both of the screening methods have very similar distributions and that there are apparent trends across both sample rates. It is notable that the importance sampling method is actually more effective at catching potential security threats until a few checkpoints before the average checkpoint the security threats are caught at. After this checkpoint the sampling methods switch and uniform sampling is more effective until all of the security threats are caught.

Even though we know that there are specific trends that are apparent across sample rates, we do not know why these trends occur. For an extension of this project it would be interesting to look at what the theoretical standard deviations for each sampling method should be. Deriving the standard deviations could explain more about the distributions of the data. Another extension of this project could be to look into the best possible method of sampling. Press describes a method with in his article where the square root of the prior probability is used. He mathematically derived the average checkpoint a security threat would be caught for this method and his results state this would be the best possible method. It would be interesting to create a function for this method and compare the means along with the other sampling methods.

**CONCLUSION**

We were successfully able to recreate the results from Press's article through simulation within R. We were able to show that our simulation generated averages that seem to agree with the average checkpoint a security threat would be caught according to Press. Therefore, the results show importance sampling is no more effective than uniform random sampling within airport security. It is also interesting that even though the distributions follow very similar trends importance sampling is more effective until a few checkpoints before the average checkpoint a security threat would be caught at. After this checkpoint though, uniform sampling is more effective until all security threats are caught. It would be interesting to find out why the distributions behave this way and also to create a function for a sampling method that is more effective that uniform or importance sampling.

**REFERENCE**

Press, W. (2010). To Catch a Terrorist: Can Ethnic Profiling Work? Significance, 7(4), 164-167.

**APPENDIX**

## Uniform Sampling Code

```
#Fuction runs a simulation to return the average number of checkpoint a security
#threat gets though without being stopped
# num.iter: number of simulation
# num.p: number of people
# num.t: number of security threats
# k: sample rate
uniform = function(num.iter=100, num.p=1000, num.t=10, k=5)
{
 #initializing array for simulation mean output
 avg.sim=rep(-1,num.iter)

 #looping through each simulation
 for(j in 1:num.iter)
 {
  #creates simulation data, ID and security threat status
  #(0:no security threat, 1:security threat)
  x=cbind(1:num.p, rep(0,num.p))
  whichrows=sample(1:num.p, num.t, replace = FALSE)
  x[whichrows,2]=1

  #copies x into newx
  newx=x

  #intializing matrix for each simulation
  output.mat = cbind(rep(0,100), rep(0,100))

  #looping through checkpoints for a single simulation
  for( i in 1:100)
  {

   #checks to see if all security threats are caught
   if (sum(newx[,2] > 0))
   {

    #selecting how many passengers to sample
    p=round(dim(newx)[1]/k,0)

    #selecting which passengers to sample
    r=sample(1:dim(newx)[1], p, replace=FALSE)

    #updates matrix with checkpoint numbers
    output.mat[i,1]=i

    #updates matrix with number of security threats caught each checkpoint
    output.mat[i,2]=sum(newx[r,2])

    #updates newx if not all security threats are caught
    if(output.mat[i,2] > 0)
    {
     newx = newx[-r[newx[r,2]==1],]
    }
   }
  }
  #keeps only the checkpoint where security threats were caught
  output.mat.keep = output.mat[output.mat[,1]>0,]
```

```
  #creates list of average number of checkpoint a security got through without
  # being selected for each simulation
  avg.sim[j] = sum(output.mat.keep[,1]*output.mat.keep[,2])/sum(x[,2])

 }

 #gives the average of all the simulations
 avg.uniform = sum(avg.sim)/num.iter

 #prints the average of all the simulations to use for statistical analysis
 print(avg.uniform)

 #returns an array of the averages for each simulation
 return(avg.sim)
}
```

## Importance Sampling Code

```
#Function runs a simulation to return the average number of checkpoint a security
#threat gets though without being stopped
# num.iter: number of simulation
# num.p: number of people
# num.t: number of security threats
# k: sample rate
significance = function(num.iter=100, num.p=1000, num.t=10, k=5)
{
 #initializing array for simulation mean output
 avg.sim=rep(-1,num.iter)

 #looping through each simulation
 for(j in 1:num.iter)
 {
  #creates simulation data
  #ID, security threat status, probability a threat based on race
  #security threat status: (0:no security threat, 1:security threat)
  x=cbind(1:num.p, rep(0,num.p), rep((2/(3*num.p)),num.p))
  prob.t = sample(1:num.p, num.p/2, replace= FALSE)
  x[prob.t,3]=(4/(3*num.p))
  whichrows=sample(1:num.p, num.t, replace = FALSE, x[,3])
  x[whichrows,2]=1

  #copies x into newx
  newx=x

  #intializing matrix for each simulation
  output.mat = cbind(rep(0,100), rep(0,100))

  #looping through checkpoints for a single simulation
  for( i in 1:100)
  {

   #checks to see if all security threats are caught
   if (sum(newx[,2] > 0))
   {

    #selecting how many passengers to sample
    p=round(dim(newx)[1]/k,0)
```

```
      #selecting which passengers to sample using significance sampling
      r=sample(1:dim(newx)[1], p, replace=FALSE, newx[,3])

      #updates matrix with checkpoint numbers
      output.mat[i,1]=i

      #updates matrix with number of security threats caught each checkpoint
      output.mat[i,2]=sum(newx[r,2])

      #updates newx if not all security threats are caught
      if(output.mat[i,2] > 0)
      {
        newx = newx[-r[newx[r,2]==1],]
      }
    }
   }
   #keeps only the checkpoint where security threats were caught
   output.mat.keep = output.mat[output.mat[,1]>0,]

   #creates list of average number of checkpoint a security got through without
   # being selected for each simulation
   avg.sim[j] = sum(output.mat.keep[,1]*output.mat.keep[,2])/sum(x[,2])

 }

 #gives the average of all the simulations
 avg.sig = sum(avg.sim)/num.iter

 #prints the average of all the simulations to use for statistical analysis
 print(avg.sig)

 #returns an array of the averages for each simulation
 return(avg.sim)
}
```