Power Calculations for Opinion Labeling in News Articles

W241, Experiments and Causality | Spring 2019 | Final Project

Authors ...

```
# load packages
library(data.table)
library(MASS)
library(ggplot2)
```

Introduction

This notebook is designed to test the statistical power of the experiment. These calculations help set expectations about the chances that a statistically significant effect will be detected, if one exists. This work seeks to measure statistical power by altering sample size, treatment effect size, and treatment effect distribution.

The results found here will help the experiment in two ways: First, it will inform the experimental design by influencing the N-sizes needed to detect an effect if one exists. Second, these results will provide context to the conclusions of the overall experiment, especially in cases where statistical significance is not found, by helping separate the possibility that an effect wasn't found because actually wasn't one vs. an effect not being found because the statistical power was low. Both of these outcomes will be applied to the overall experiment design and findings, as well as to those within blocks, and therefore with smaller sample sizes.

Testing Nulls

We'll begin by testing the power calculation on a null hypothesis by simulating idential distirbutions of responses, repeating this experiment multiple times (re-randomizing), and then calculating the percent of experiments where the p value is less than 0.05, indicating the frequency with which we observed statistical significance by random chance as opposed to an actual effect (false positives). We expect a result near 5%.

First, we'll generate data:

```
### GENERATE DATA ###
# Create survey category options
vecPolBias <- c('1_Very Liberal','2_Somewhat Liberal','3_Slightly Liberal',
                '4_Neutral','5_Slightly Conservative','6_Somewhat Conservative',
                '7_Very Conservative')
# The numbers preceeding the response options preserving ordinality by
# forcing alpha-numeric order.
vecPolBias
```

```
## [1] "1_Very Liberal"          "2_Somewhat Liberal"
## [3] "3_Slightly Liberal"      "4_Neutral"
## [5] "5_Slightly Conservative" "6_Somewhat Conservative"
## [7] "7_Very Conservative"
```

```
#Dynamically retrieve number of categories, and also convert to a vector
intNumCategories <- length(vecPolBias)
intNumCategories
```

```
## [1] 7
```

```
vecCategoryNums <- seq(intNumCategories)
vecCategoryNums
```

```
## [1] 1 2 3 4 5 6 7
```

```
# #Convert to data table
# dtOptions <- data.table('BiasNum'=vecCategoryNums, 'BiasRating'=vecPolBias)
# dtOptions
#
# #Note, our bias number is just used in data generation.
# #It is not meant to imply linearity between categories,
# #as doing so would defeat their ordinal-only nature.

# Set our probabilities of selecting responses

#We'll start with a uniform distribution (equal odds of selecting any response)
#Set probs for being in CONTROL GROUP
vecProbC <- rep(1/intNumCategories, intNumCategories) #Control probs.
vecProbC
```

```
## [1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571
```

```
#Set probs for being in TREATMENT GROUP
vecProbT <- vecProbC  #Treatment probs. Set the same, identical distributions
vecProbT
```

```
## [1] 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571 0.1428571
```

```
#Now let's run a quick observational test to see if we can generate random uniform choices
#... commenting out
#sample(x=vecPolBias, size=10, prob=vecProbC, replace=TRUE)
#sample(x=vecPolBias, size=10, prob=vecProbT, replace=TRUE)

#Let's further test by generating a larger sample and observing their frequency
intSampSize <- 100000
vecSampleDataC <- sample(x=vecPolBias, size=intSampSize, prob=vecProbC, replace=TRUE)
vecSampleDataT <- sample(x=vecPolBias, size=intSampSize, prob=vecProbT, replace=TRUE)
vecSampleData <- c(vecSampleDataC, vecSampleDataT)

dt <- data.table('outcome'=vecSampleData,
                 'treat'=c(rep(0, intSampSize), rep(1, intSampSize))
                 )
dt[, .N / intSampSize, keyby=.(treat,outcome)]
```

```
##     treat                 outcome      V1
##  1:     0          1_Very Liberal 0.14211
##  2:     0      2_Somewhat Liberal 0.14409
##  3:     0      3_Slightly Liberal 0.14593
##  4:     0               4_Neutral 0.14119
##  5:     0 5_Slightly Conservative 0.14163
##  6:     0 6_Somewhat Conservative 0.14209
##  7:     0     7_Very Conservative 0.14296
##  8:     1          1_Very Liberal 0.14286
##  9:     1      2_Somewhat Liberal 0.14328
## 10:     1      3_Slightly Liberal 0.14247
```

```
## 11:     1               4_Neutral 0.14059
## 12:     1 5_Slightly Conservative 0.14377
## 13:     1 6_Somewhat Conservative 0.14247
## 14:     1     7_Very Conservative 0.14456
```

```
#Yup, it works!

#Quick peek at our data ... looks good ... commenting out
#dt[sample(.N,10)]

#Finally, we'll make a function so we can repeatedly randomly generate this data
GenRandData <- function(intSampSize, vecProbC, vecProbT) {

  vecSampleDataC <- sample(x=vecPolBias, size=intSampSize, prob=vecProbC, replace=TRUE)
  vecSampleDataT <- sample(x=vecPolBias, size=intSampSize, prob=vecProbT, replace=TRUE)
  vecSampleData <- c(vecSampleDataC, vecSampleDataT)

  return(data.table('outcome'=vecSampleData,
                    'treat'=c(rep(0, intSampSize), rep(1, intSampSize))
                    )
         )
}

#Test our function ... it works ... commenting out
#test <- GenRandData(intSampSize, vecProbC, vecProbT)
#test[sample(.N,10)]
```

Now that we have our data, let's conduct our power tests.

```
### POWER TESTS ###

#We'll first generate a model along with some summary stats
#that eventually results in a p-Value.
intSampSize = 50000  #Very large sample produces consistent model
dt <- GenRandData(intSampSize, vecProbC, vecProbT)
m <- polr(as.factor(outcome) ~ treat, data = dt, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(outcome) ~ treat, data = dt, Hess = TRUE)
##
## Coefficients:
##           Value Std. Error t value
## treat -0.006188    0.01107 -0.5591
##
## Intercepts:
##                                                 Value     Std. Error
## 1_Very Liberal|2_Somewhat Liberal                 -1.7890    0.0106
## 2_Somewhat Liberal|3_Slightly Liberal             -0.9223    0.0089
## 3_Slightly Liberal|4_Neutral                      -0.2959    0.0085
## 4_Neutral|5_Slightly Conservative                  0.2754    0.0085
## 5_Slightly Conservative|6_Somewhat Conservative    0.9130    0.0089
## 6_Somewhat Conservative|7_Very Conservative        1.7840    0.0106
##                                                 t value
## 1_Very Liberal|2_Somewhat Liberal               -168.9672
## 2_Somewhat Liberal|3_Slightly Liberal           -103.2507
```

```
## 3_Slightly Liberal|4_Neutral                       -34.9856
## 4_Neutral|5_Slightly Conservative                   32.5854
## 5_Slightly Conservative|6_Somewhat Conservative    102.3125
## 6_Somewhat Conservative|7_Very Conservative        168.5912
##
## Residual Deviance: 389175.51
## AIC: 389189.51
```

```
tValue <- coef(summary(m))['treat', 't value']
tValue
```

```
## [1] -0.5591312
```

```
pValue <- pnorm(abs(tValue), lower.tail=F) * 2
pValue
```

```
## [1] 0.5760722
```

```
pValue < 0.05
```

```
## [1] FALSE
```

Now let's replicate this function to see how often it returns positive results.

```
#Now we'll wrap the above stuff in a function so we can replicate it
RepPVals <- function(intSampSize, vecProbC, vecProbT, pThresh=0.05) {

  dt <- GenRandData(intSampSize, vecProbC, vecProbT)
  m <- polr(as.factor(outcome) ~ treat, data = dt, Hess=TRUE)
  tValue <- coef(summary(m))['treat', 't value']
  pValue <- pnorm(abs(tValue), lower.tail=F) * 2

  #return(pValue)
  return(pValue < pThresh)

}

#Test the function
intSampSize = 500
RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05)
```

```
## [1] FALSE
```

```
#Now we replicate the function a bunch of times!
intSampSize = 500
intNumReps <- 1000
pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
numDetected <- sum(pValsNull) / intNumReps
numDetected
```

```
## [1] 0.062
```

Great! Having run this a few times, we verify that we get values close to 0.05 as expected. This exercise also gives us confidence that our p-value testing setup works as expected.

Testing Treatment Effects Against a Uniform Distribution

Now let's move on to rerunning this analysis when their actually *is* a difference between treatment and control. To generate this difference, we'll alter the current random uniform distirbution for the treatment

group, while keeping the control group the same, thus simulating a causal effect of subjects in the treatment group picking different choices.

Aside, we acknoweldge that starting from a random uniform distribution (equal probability of picking any response) for the control group is an assumption that may or may not be valid. We'll keep this assuption for now, but later on in this analysis we'll change it to see how the results change.

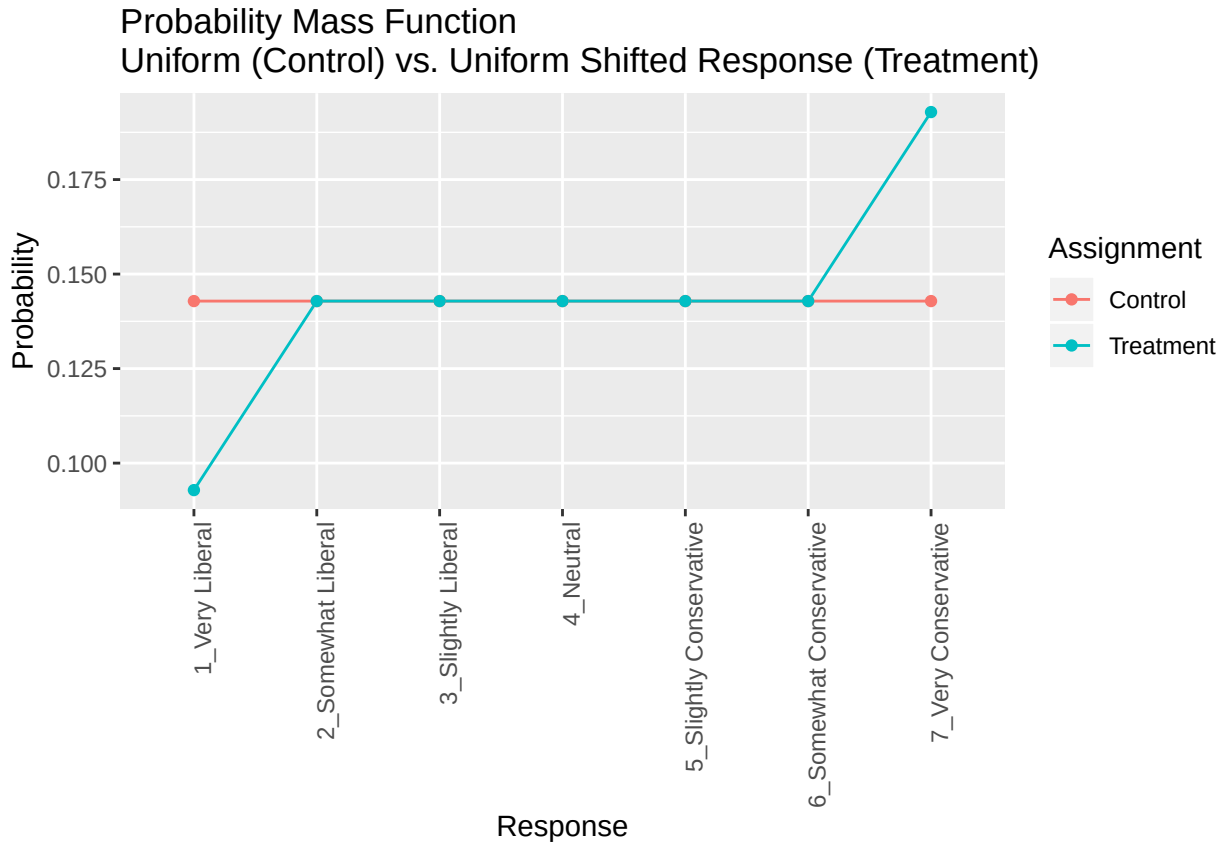
Uniform Shifted Response

First, we'll simulate a situation where we imagine every respondent has an X% probability of picking the next higher choice, and we'll refer to this change as a *uniform shift*. In this case, it means they'll shift to become more conservative, however, because our distribution is symmetric, it would work equally well or poorly in the liberal direction. Currently, every resopnse has a roughly 14% chance of being selected. Making this change will actually only result in the lowest response (Very Liberal) going down, and the highest response (Very Conservative) going up, because all the responses in the middle will have the same number of hypothetical respondents leave the for the next higher choice as they have that newly enter their choice from the lower response. Let's code this distribution now.

```
### UNIFORM SHIFTED DISTRIBUTION, GENERATION ###

#Uniform Shifted Responses
numShiftFactor <- 0.05
vecProbT <- vecProbC + c(-numShiftFactor,0,0,0,0,0,numShiftFactor)
vecProbT
```

```
## [1] 0.09285714 0.14285714 0.14285714 0.14285714 0.14285714 0.14285714
## [7] 0.19285714
```

```
#Let's visually compare our control and treatment distrubitons

#We'll create a function to do this here that we'll call later as well
PlotDists <- function(vecProbC, vecProbT, strSubtitle='') {

  dtPlot <- data.table('Response'=c(vecPolBias, vecPolBias),
                       'Assignment'=c(rep('Control',length(vecPolBias)),
                                      rep('Treatment',length(vecPolBias))),
                       'Probability'=c(vecProbC, vecProbT)
                       )

  ggplot(data=dtPlot, aes(x=Response, y=Probability, col=Assignment, group=Assignment)) +
   geom_line()+
   geom_point()+
   ggtitle(paste('Probability Mass Function\n',strSubtitle,sep='')) +
   theme(axis.text.x = element_text(angle = 90, hjust = 1))

}

PlotDists(vecProbC, vecProbT, 'Uniform (Control) vs. Uniform Shifted Response (Treatment)')
```



Having visualized our new treatment distribution, let's run our tests to see how frequently we detect effects.

```
### UNIFORM SHIFTED DISTRIBUTION, POWER CURVES ###

#Run a single test
intSampSize = 500
dt <- GenRandData(intSampSize, vecProbC, vecProbT)
m <- polr(as.factor(outcome) ~ treat, data = dt, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(outcome) ~ treat, data = dt, Hess = TRUE)
##
## Coefficients:
##        Value Std. Error t value
## treat 0.3009      0.111   2.711
##
## Intercepts:
##                                                 Value    Std. Error
## 1_Very Liberal|2_Somewhat Liberal                -1.8883   0.1114
## 2_Somewhat Liberal|3_Slightly Liberal            -0.8684   0.0894
## 3_Slightly Liberal|4_Neutral                     -0.2682   0.0849
## 4_Neutral|5_Slightly Conservative                 0.3732   0.0851
## 5_Slightly Conservative|6_Somewhat Conservative   1.0387   0.0900
## 6_Somewhat Conservative|7_Very Conservative       1.8135   0.1042
##                                                 t value
## 1_Very Liberal|2_Somewhat Liberal               -16.9433
```

```
## 2_Somewhat Liberal|3_Slightly Liberal            -9.7123
## 3_Slightly Liberal|4_Neutral                     -3.1570
## 4_Neutral|5_Slightly Conservative                 4.3863
## 5_Slightly Conservative|6_Somewhat Conservative  11.5366
## 6_Somewhat Conservative|7_Very Conservative      17.4122
##
## Residual Deviance: 3872.612
## AIC: 3886.612
```

```
tValue <- coef(summary(m))['treat', 't value']
tValue
```

```
## [1] 2.710617
```

```
pValue <- pnorm(abs(tValue), lower.tail=F) * 2
pValue
```

```
## [1] 0.006715807
```

```
pValue < 0.05
```

```
## [1] TRUE
```

```
#Now let's replicate it several times and see how many times it detects the difference.
intSampSize = 100
intNumReps <- 1000
pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
numDetected <- sum(pValsNull) / intNumReps
numDetected
```

```
## [1] 0.209
```

```
#Finally, let's generate "power curves" from replicating this simulation with various
#effect sizes and N-sizes

vecSampSizeTrials <- c(10, 25, 50, 100, 250, 500, 1000)
vecShiftFactorTrials <- seq(5) / 50  #(0.02, 0.04, 0.06, 0.08, 0.10)

#Start a data table with just our sample sizes (we'll fill in the rest as the experiment churns)
dtResultsUniShift <- data.table('NumResponses'=vecSampSizeTrials)
dtResultsUniShift
```

```
##    NumResponses
## 1:           10
## 2:           25
## 3:           50
## 4:          100
## 5:          250
## 6:          500
## 7:         1000
```

```
#Poor man's Grid Search to generate results ...
for (numShiftFactor in vecShiftFactorTrials){
  vecResults <- c(0) #Just a dummy value

  for (intSampSize in vecSampSizeTrials){

    vecProbT <- vecProbC + c(-numShiftFactor,0,0,0,0,0,numShiftFactor)
    pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
```

```
    numDetected <- sum(pValsNull) / intNumReps
    vecResults <- c(vecResults, numDetected)

    #print(intSampSize)
    #print(numShiftFactor)
    #print(numDetected)

  }
  print(vecResults[-1:-1])
  dtResultsUniShift[, paste('ATE',numShiftFactor,sep='') := vecResults[-1:-1] ]

}
```

```
## [1] 0.063 0.062 0.063 0.068 0.115 0.169 0.253
## [1] 0.070 0.072 0.098 0.128 0.273 0.479 0.766
## [1] 0.069 0.095 0.141 0.247 0.563 0.809 0.984
## [1] 0.085 0.140 0.241 0.416 0.739 0.973 1.000
## [1] 0.114 0.196 0.342 0.579 0.911 0.995 1.000
```

```
dtResultsUniShift
dtResultsUniShift #Not sure why, but it takes calling this twice to see the results
```

```
##    NumResponses ATE0.02 ATE0.04 ATE0.06 ATE0.08 ATE0.1
## 1:           10   0.063   0.070   0.069   0.085  0.114
## 2:           25   0.062   0.072   0.095   0.140  0.196
## 3:           50   0.063   0.098   0.141   0.241  0.342
## 4:          100   0.068   0.128   0.247   0.416  0.579
## 5:          250   0.115   0.273   0.563   0.739  0.911
## 6:          500   0.169   0.479   0.809   0.973  0.995
## 7:         1000   0.253   0.766   0.984   1.000  1.000
```

```
#Save results to disk
fwrite(dtResultsUniShift, 'dtResultsUniShift.csv')
```

Uniform Tilted Response

Now let's look at a tilted response.

```
### UNIFORM TILTED DISTRIBUTION GENERATION ###

#Uniform Tilted Responses
numMaxTiltFactor <- 0.05
vecProbT <- vecProbC + c(-numMaxTiltFactor,-numMaxTiltFactor*2/3,-numMaxTiltFactor*1/3,0,
                         numMaxTiltFactor*1/3,numMaxTiltFactor*2/3,numMaxTiltFactor)
vecProbT
```

```
## [1] 0.09285714 0.10952381 0.12619048 0.14285714 0.15952381 0.17619048
## [7] 0.19285714
```

```
#Visually compare our control and treatment distrubitons
PlotDists(vecProbC, vecProbT, 'Uniform (Control) vs. Uniform Shifted Response (Treatment)')
```

Now let's see how this tilted distribution runs in our power tests.

```
### UNIFORM TILTED DISTRIBUTION ###

#Run a single test
intSampSize = 500
dt <- GenRandData(intSampSize, vecProbC, vecProbT)
m <- polr(as.factor(outcome) ~ treat, data = dt, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(outcome) ~ treat, data = dt, Hess = TRUE)
##
## Coefficients:
##        Value Std. Error t value
## treat 0.3568     0.1112   3.208
##
## Intercepts:
##                                                 Value    Std. Error
## 1_Very Liberal|2_Somewhat Liberal                -1.8250   0.1099
## 2_Somewhat Liberal|3_Slightly Liberal            -0.9814   0.0909
## 3_Slightly Liberal|4_Neutral                     -0.3574   0.0850
## 4_Neutral|5_Slightly Conservative                 0.2762   0.0848
## 5_Slightly Conservative|6_Somewhat Conservative   0.9865   0.0903
## 6_Somewhat Conservative|7_Very Conservative       1.8046   0.1044
##                                                 t value
## 1_Very Liberal|2_Somewhat Liberal               -16.6031
```

```
## 2_Somewhat Liberal|3_Slightly Liberal           -10.7969
## 3_Slightly Liberal|4_Neutral                     -4.2034
## 4_Neutral|5_Slightly Conservative                 3.2572
## 5_Slightly Conservative|6_Somewhat Conservative  10.9300
## 6_Somewhat Conservative|7_Very Conservative      17.2859
##
## Residual Deviance: 3864.585
## AIC: 3878.585
```

```
tValue <- coef(summary(m))['treat', 't value']
tValue
```

```
## [1] 3.207762
```

```
pValue <- pnorm(abs(tValue), lower.tail=F) * 2
pValue
```

```
## [1] 0.001337722
```

```
pValue < 0.05
```

```
## [1] TRUE
```

```
#Now let's replicate it several times and see how many times it detects the difference.
intSampSize = 100
intNumReps <- 1000
pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
numDetected <- sum(pValsNull) / intNumReps
numDetected
```

```
## [1] 0.381
```

```
#Finally, let's generate "power curves" from replicating this simulation with various
#effect sizes and N-sizes

vecSampSizeTrials <- c(25, 50, 100, 250, 500, 1000)
#vecSampSizeTrials <- c(25, 50)
vecMaxTiltFactorTrials <- seq(5) / 50  #(0.02, 0.04, 0.06, 0.08, 0.10)
#vecMaxTiltFactorTrials <- seq(2) / 50  #(0.02, 0.04)

#Start a data table with just our sample sizes (we'll fill in the rest as the experiment churns)
dtResults <- data.table('NumResponses'=vecSampSizeTrials)
dtResults
```

```
##    NumResponses
## 1:           25
## 2:           50
## 3:          100
## 4:          250
## 5:          500
## 6:         1000
```

```
#Poor man's Grid Search to generate results ...
for (numMaxTiltFactor in vecMaxTiltFactorTrials){
  vecResults <- c(0) #Just a dummy value

  for (intSampSize in vecSampSizeTrials){

    vecProbT <- vecProbC + c(-numMaxTiltFactor,-numMaxTiltFactor*2/3,-numMaxTiltFactor*1/3,0,
```

```
                              numMaxTiltFactor*1/3,numMaxTiltFactor*2/3,numMaxTiltFactor)
    pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
    numDetected <- sum(pValsNull) / intNumReps
    vecResults <- c(vecResults, numDetected)

  }
  print(vecResults[-1:-1])
  dtResults[, paste('ATE',numMaxTiltFactor,sep='') := vecResults[-1:-1] ]

}
```

```
## [1] 0.073 0.060 0.103 0.171 0.331 0.543
## [1] 0.106 0.172 0.262 0.573 0.849 0.990
## [1] 0.184 0.308 0.540 0.888 0.999 1.000
## [1] 0.270 0.484 0.753 0.981 1.000 1.000
## [1] 0.415 0.632 0.914 0.999 1.000 1.000
```

```
dtResults
dtResults #Not sure why, but it takes calling this twice to see the results
```

```
##    NumResponses ATE0.02 ATE0.04 ATE0.06 ATE0.08 ATE0.1
## 1:           25   0.073   0.106   0.184   0.270  0.415
## 2:           50   0.060   0.172   0.308   0.484  0.632
## 3:          100   0.103   0.262   0.540   0.753  0.914
## 4:          250   0.171   0.573   0.888   0.981  0.999
## 5:          500   0.331   0.849   0.999   1.000  1.000
## 6:         1000   0.543   0.990   1.000   1.000  1.000
```

```
#Save results to disk
fwrite(dtResults, 'dtResultsUniTilt.csv')
```

Normally Distributed Responses

While the uniform distribution had the property of being totally random, that may not best mimic how actual users will respond. Instead, it's plausible that they'll cluster around a single response or group of responses, with a much lower chance of choosing other response options, which may be best illustrated with a normal distribution. Applying the treatment of opinion labels to subjects may then shift this distribution in one direction or the other. Here is what this will look like visually.

```
### NORMAL SHIFTED DISTRIBUTION GENERATION ###

#First create the control group distribution
vecCenteredNums <- vecCategoryNums - 4 #Centers 'neutral' category at 0
numNormSpike <- 0.7  #Adjusts the 'spikiness' of the normal distribution, higher is spikier
numNormShift <- 1  #Adjusts the shift of the treatment group

#Quick function to normalize the probability mass to sum to 1
NormalizeMass <- function(distIn) {
  return(distIn/sum(distIn))
}

vecProbC <- NormalizeMass(dnorm((vecCenteredNums-0) * numNormSpike))
vecProbC
```

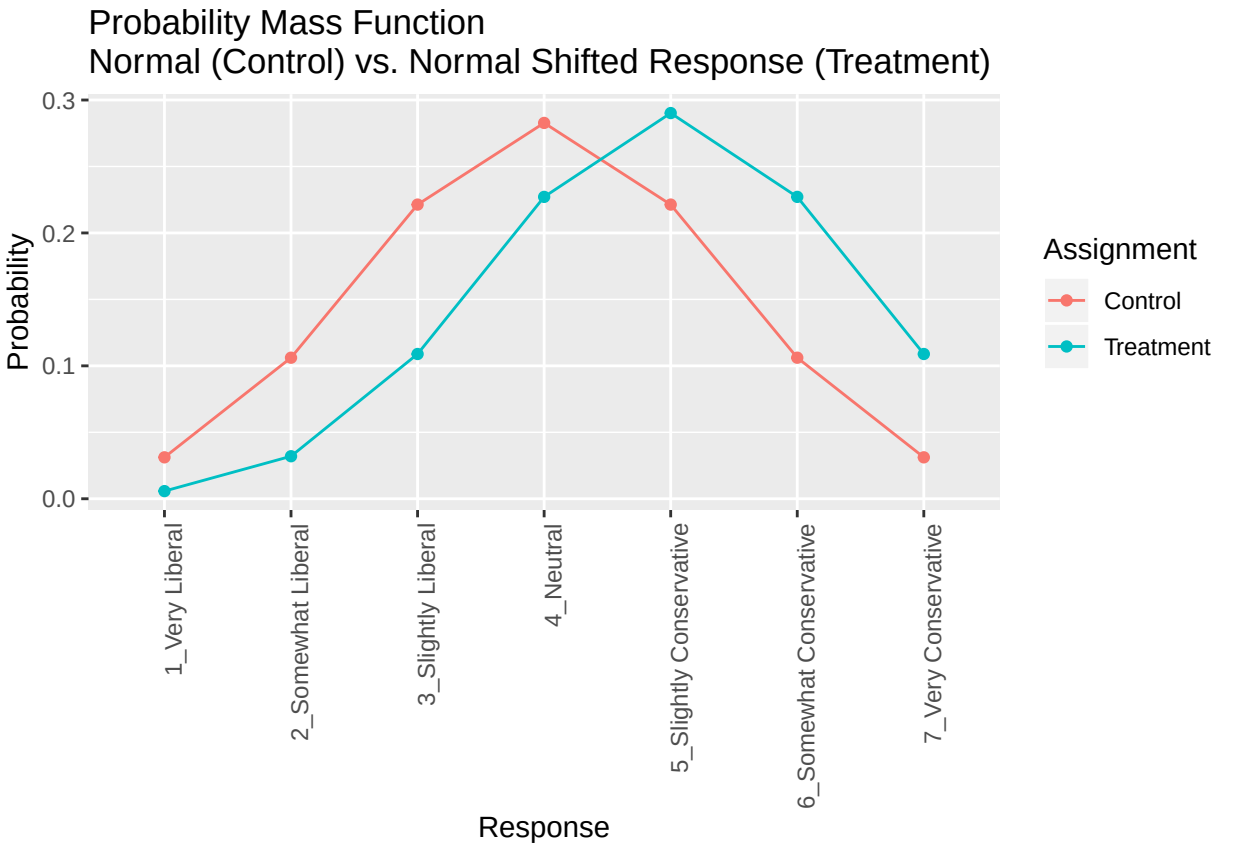
```
## [1] 0.03117475 0.10612404 0.22131977 0.28276286 0.22131977 0.10612404
```

```
## [7] 0.03117475
```

```
#Create the treatment group distribution
vecProbT <- NormalizeMass(dnorm((vecCenteredNums-numNormShift) * numNormSpike))
vecProbT
```

```
## [1] 0.005757512 0.031992627 0.108908217 0.227126125 0.290181178 0.227126125
## [7] 0.108908217
```

```
#Visually compare our control and treatment distrubitons
PlotDists(vecProbC, vecProbT, 'Normal (Control) vs. Normal Shifted Response (Treatment)')
```



Of note, the `numNormSpike` factor was set to 0.7 to give a small probability (as opposed to nearly 0) that the farthest outliers of Very Liberal and Very Conservative would occasionally be chosen.

Now that we see the distributions visually, we will calculate the Power probabilities at various effect sizes and sample sizes.

```
### NORMAL SHIFTED DISTRIBUTION POWER CURVES ###

#Run a single test
intSampSize = 500
dt <- GenRandData(intSampSize, vecProbC, vecProbT)
m <- polr(as.factor(outcome) ~ treat, data = dt, Hess=TRUE)
summary(m)
```

```
## Call:
## polr(formula = as.factor(outcome) ~ treat, data = dt, Hess = TRUE)
##
```

```
## Coefficients:
##       Value Std. Error t value
## treat  1.16     0.1174   9.888
##
## Intercepts:
##                                                 Value    Std. Error
## 1_Very Liberal|2_Somewhat Liberal                -3.7571   0.2632
## 2_Somewhat Liberal|3_Slightly Liberal            -1.9443   0.1217
## 3_Slightly Liberal|4_Neutral                     -0.5803   0.0878
## 4_Neutral|5_Slightly Conservative                 0.6267   0.0883
## 5_Slightly Conservative|6_Somewhat Conservative   1.8703   0.1046
## 6_Somewhat Conservative|7_Very Conservative       3.1404   0.1399
##                                                 t value
## 1_Very Liberal|2_Somewhat Liberal               -14.2722
## 2_Somewhat Liberal|3_Slightly Liberal           -15.9803
## 3_Slightly Liberal|4_Neutral                     -6.6054
## 4_Neutral|5_Slightly Conservative                 7.0945
## 5_Slightly Conservative|6_Somewhat Conservative  17.8881
## 6_Somewhat Conservative|7_Very Conservative      22.4403
##
## Residual Deviance: 3369.631
## AIC: 3383.631
```

```
tValue <- coef(summary(m))['treat', 't value']
tValue
```

```
## [1] 9.887553
```

```
pValue <- pnorm(abs(tValue), lower.tail=F) * 2
pValue
```

```
## [1] 4.714074e-23
```

```
pValue < 0.05
```

```
## [1] TRUE
```

```
#Now let's replicate it several times and see how many times it detects the difference.
intSampSize = 100
intNumReps <- 1000
pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
numDetected <- sum(pValsNull) / intNumReps
numDetected
```

```
## [1] 0.995
```

```
#Finally, let's generate "power curves" from replicating this simulation with various
#effect sizes and N-sizes

vecSampSizeTrials <- c(25, 50, 100, 250, 500, 1000)
#vecSampSizeTrials <- c(25, 50)
vecNormShiftTrials <- c(0.25, 0.5, 0.75, 1, 1.5, 2)
#vecNormShiftTrials <- c(0.25, 0.5)

#Start a data table with just our sample sizes (we'll fill in the rest as the experiment churns)
dtResults <- data.table('NumResponses'=vecSampSizeTrials)
dtResults
```

```
##    NumResponses
## 1:           25
## 2:           50
## 3:          100
## 4:          250
## 5:          500
## 6:         1000
```

```
#Poor man's Grid Search to generate results ...
for (numNormShift in vecNormShiftTrials){
  vecResults <- c(0) #Just a dummy value

  for (intSampSize in vecSampSizeTrials){

    vecProbT <- NormalizeMass(dnorm((vecCenteredNums-numNormShift) * numNormSpike))
    pValsNull <- replicate(intNumReps, RepPVals(intSampSize, vecProbC, vecProbT, pThresh=0.05))
    numDetected <- sum(pValsNull) / intNumReps
    vecResults <- c(vecResults, numDetected)

  }
  print(vecResults[-1:-1])
  dtResults[, paste('ATE',numNormShift,sep='') := vecResults[-1:-1] ]

}
```

```
## [1] 0.075 0.132 0.216 0.432 0.742 0.971
## [1] 0.186 0.361 0.645 0.945 1.000 1.000
## [1] 0.391 0.701 0.926 1.000 1.000 1.000
## [1] 0.631 0.905 0.996 1.000 1.000 1.000
## [1] 0.927 0.996 1.000 1.000 1.000 1.000
## [1] 0.994 1.000 1.000 1.000 1.000 1.000
```

```
dtResults
dtResults #Not sure why, but it takes calling this twice to see the results
```

```
##    NumResponses ATE0.25 ATE0.5 ATE0.75  ATE1 ATE1.5  ATE2
## 1:           25   0.075  0.186   0.391 0.631  0.927 0.994
## 2:           50   0.132  0.361   0.701 0.905  0.996 1.000
## 3:          100   0.216  0.645   0.926 0.996  1.000 1.000
## 4:          250   0.432  0.945   1.000 1.000  1.000 1.000
## 5:          500   0.742  1.000   1.000 1.000  1.000 1.000
## 6:         1000   0.971  1.000   1.000 1.000  1.000 1.000
```

```
#Save results to disk
fwrite(dtResults, 'dtResultsNormShift.csv')
```