# 9_real data example 1

*Ziang Wang*

*7/15/2020*

## Example with 50 counts (Real data from the U.S. Census Bureau Website)

### Generate 50 counts and compute the total count

The data is about the Total Employement of 50 states in the U.S.in 2018.

```
# Get the data from the U.S. Census Bureau Website
# Total Employment in 2018
url = "https://www.census.gov/quickfacts/geo/chart/US/BZA110218"
counts <- read_html(url) %>% html_nodes(".qf-positive") %>% html_text()
counts <- counts[c(2*(1:length(counts))-1,2*(length(counts):1))][1:50]
```

```
set.seed(328328)
N <- as.numeric(gsub(",","",counts))
t <- sum(N)
N
```

```
##  [1]  1730817   261053  2549128  1043210 15223664  2423817  1528867
##  [8]   405809   539557  8669611  3975657   551681   597765  5524630
## [15]  2816081  1364250  1203434  1642234  1691552   516240  2366053
## [22]  3323852  3947891  2729492   944890  2533694   371239   845616
## [29]  1221809   612420  3739076   631393  8410206  3848565   346155
## [36]  4878062  1385228  1629432  5478025   661332   442449  1903609
## [43]   359771  2683214 10794596  1337574   261282  3386839  2847481
## [50]   554567
```

```
t
```

```
## [1] 128734869
```

### Define alpha and sampling functions

```
# Define alpha based on the geometric mechanism
alpha <- 1/exp(1)

# pdf of the doulbe geometric distribution
probs <- function(n, k = 0){
  p <- c()
  for(i in 1:length(n)){
    p[i] <- alpha^(abs(n[i] - k))*(1-alpha)/(1 + alpha)
  }
  return(p)
}

# Chop the noise so that i is less than or equal to 50 (?)
```

```r
# Set the first and the last probabilities(Boundaries) to adjust when using sample function
first_p = last_p = function(p){
  return(0.5*(1 - sum(p)))
}

# Define the function to sample noisy count (could be positive) from the dg distribution
samplenoise <- function(n, center = 0, i = 50){
  i_ = i-1
  return(sample(x = (-i + center):(i + center), size = n,
                prob = c(first_p(probs((-i_ + center):(i_ + center), center)),
                         probs((-i_ + center):(i_ + center), center),
                         last_p(probs((-i_ + center):(i_ + center), center)))))
}

# Define the function to sample posterior count (must be positive) from the dg distribution
samplepos <- function(n, center = 0, i = 50){

  # Two cases
  if (-i + center >= 0){
    result = samplenoise(n, center, i)
  } else {
    prob = probs(0:(i + center), center)
    result = sample(x = 0:(i + center), size = n, prob = prob / sum(prob))
  }
  return(result)
}

# Define the function to find the pi in multinomial idea
pi <- function(pos){
  p <- c()
  for(i in 1:length(pos)){
    p[i] <- pos[i] / sum(pos)
  }
  return(p)
}
```

### Find noisy counts and the posterior of the total and individuals

All algorithms share the same noisy counts.

1. Algorithm 1: New total and new components for each try
2. Algorithm 2: New total with fixed components
3. Algorithm 3: Only use noise counts (Not Bayesian). If noisy count is negative, use the posterior mode.

```r
set.seed(1)

# Generate n counts
n = 10000

# Create vectors for results
p1_total <- c()
p2_total <- c()
t3_noise <- c()
```

```r
p1_N <- matrix(data = NA, n, 50)
N3_noise <- matrix(data = NA, n, 50)

for(i in 1:n){

  # Noisy count of the total
  t.noise <- samplenoise(1, center = t, i = 50)

  # Algorithm 1

    ## Posterior count of the total
  p1_total[i] <- samplepos(1, center = t.noise, i = 50)

    ## Sample the individual noisy counts
  N_noise <- c()
  for(j in 1:length(N)){
    N_noise[j] <- samplenoise(1, center = N[j], i = 50)
  }
    ## Sample the individual posterior counts
  posterior_N <- c()
  for(k in 1:length(N_noise)){
    posterior_N[k] <- samplepos(n = 1, center = N_noise[k], i = 50)
  }

  p1_N[i,] <- posterior_N

  # Algorithm 2
  p2_total[i] <- p1_total[i]

  # Algorithm 3
  t3_noise[i] <- t.noise
  N3_noise[i,] <- N_noise
}
```

```r
# Algorithm 3 Adjustments

# Function used to find the posterior mode
Mode <- function(x) {
  ux <- unique(x)
  ux[which.max(tabulate(match(x, ux)))]
}

# Substitute negative noisy counts with the posterior mode of that state
for(i in 1:n){
  for(j in 1:50){
    if(N3_noise[i,j] < 0){
      N3_noise[i,j] <- Mode(p1_N[,j])
    }
  }
}
```

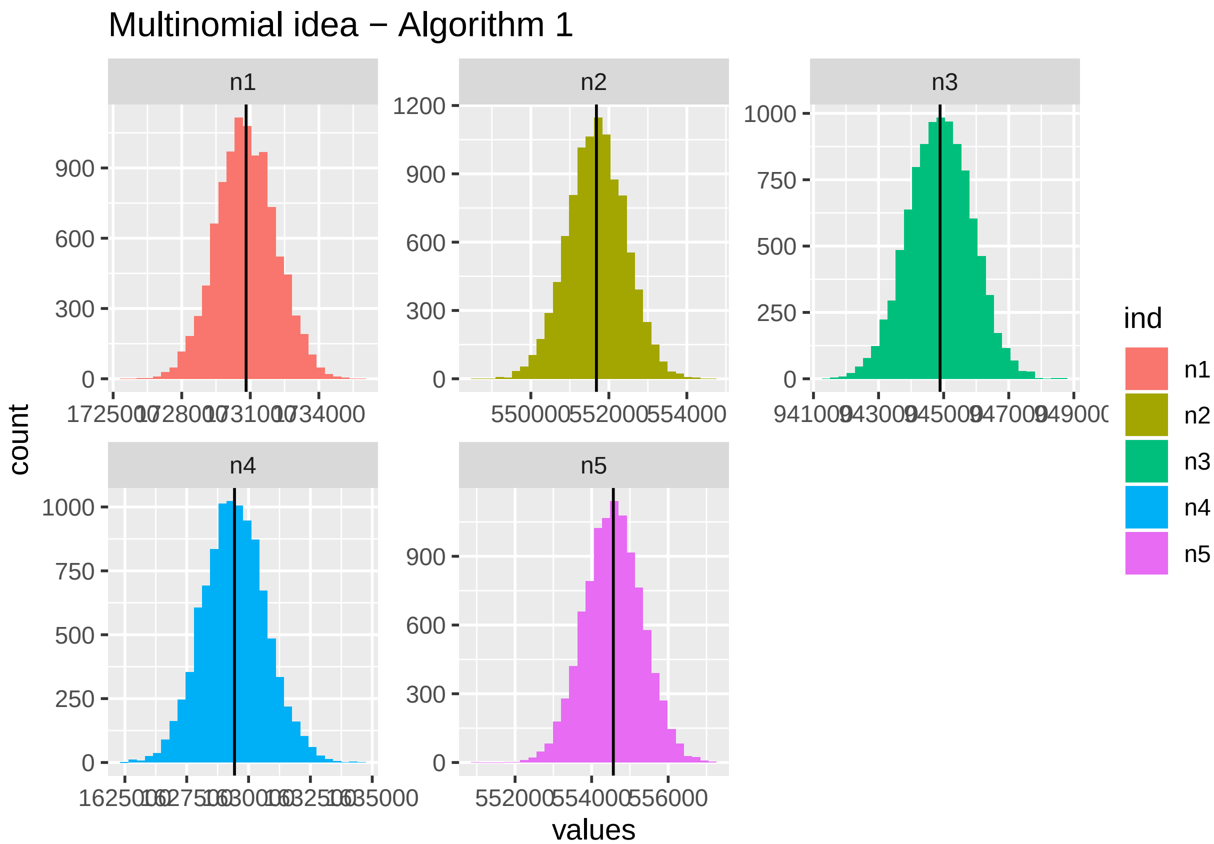
## Multinomial idea

### Generate each individual count of multinomial idea

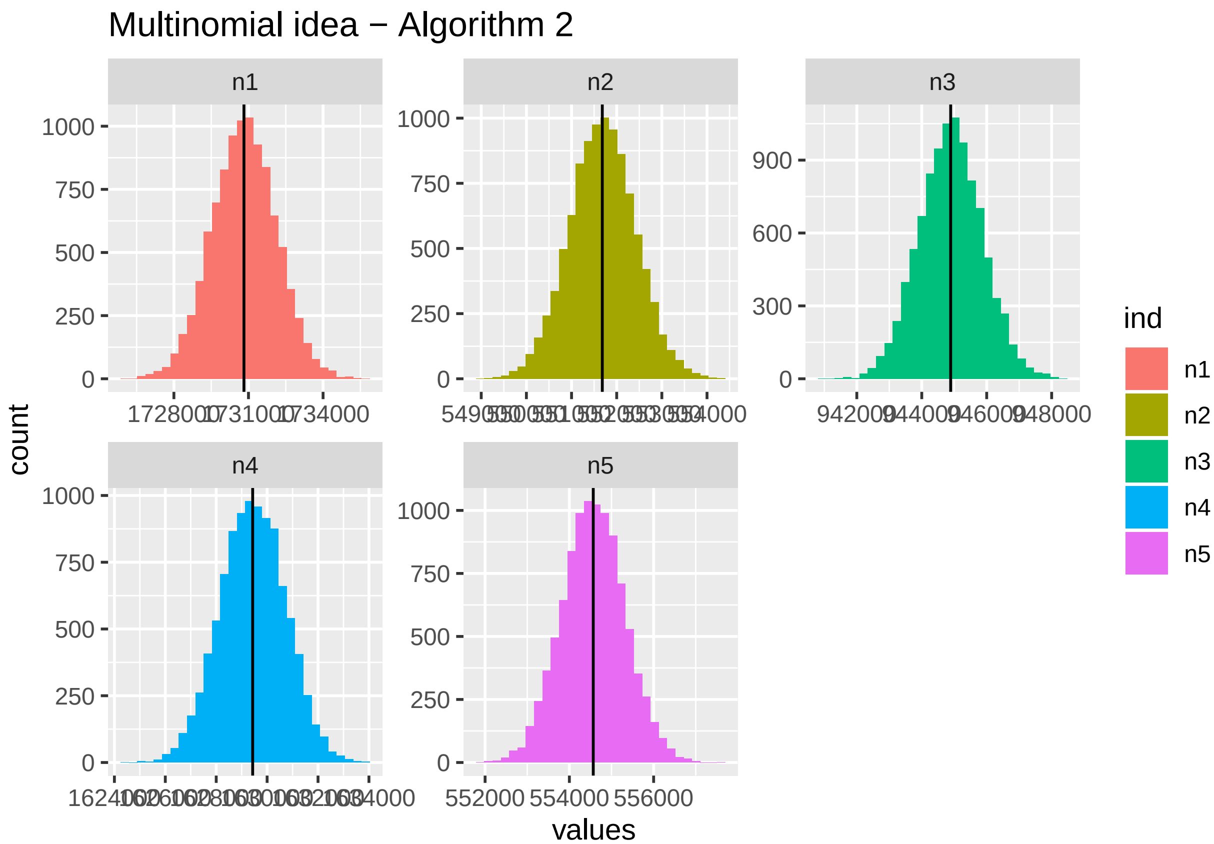
```
set.seed(1)

multi_1 <- matrix(data = NA, 50, n)
multi_2 <- matrix(data = NA, 50, n)
multi_3 <- matrix(data = NA, 50, n)

for(i in 1:n){
  # Algorithm 1
  multi_1[, i] <- rmultinom(n = 1, size = p1_total[i], prob = pi(p1_N[i,]))

  # Algorithm 2: Probabilities are the same for all 10000 runs.
  multi_2[, i] <- rmultinom(n = 1, size = p2_total[i], prob = pi(p1_N[1,]))

  # Algorithm 3: Only use noisy counts for each run
  multi_3[, i] <- rmultinom(n = 1, size = t3_noise[i], prob = pi(N3_noise[i,]))
}
```

### Plot the results

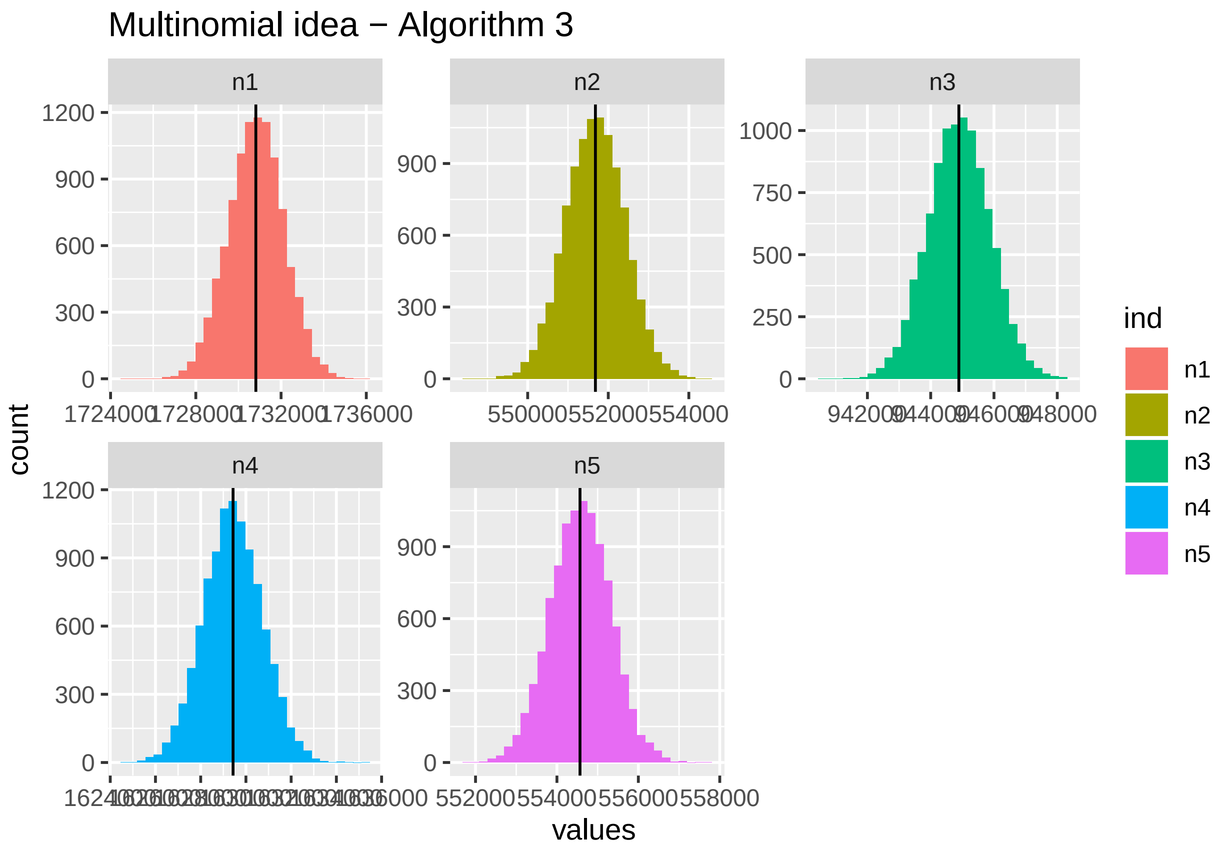
```
# Algorithm 1
result1 <- data.frame(n1 =multi_1[1,], n2 = multi_1[12,], n3 = multi_1[25,],
                      n4 = multi_1[38,], n5 = multi_1[50,])
results1 <- stack(result1)

truevalue <- data.frame(ind = c("n1", "n2", "n3", "n4", "n5"),
                        true = c(N[1], N[12], N[25], N[38], N[50]))

ggplot(data = results1, aes(values, fill = ind)) +
  geom_histogram() +
  facet_wrap(~ind, scales = "free") +
  geom_vline(data = truevalue, aes(xintercept = true)) +
  ggtitle("Multinomial idea - Algorithm 1")
```




```r
# Algorithm 2
result2 <- data.frame(n1 =multi_2[1,], n2 = multi_2[12,], n3 = multi_2[25,],
                      n4 = multi_2[38,], n5 = multi_2[50,])
results2 <- stack(result2)

ggplot(data = results2, aes(values, fill = ind)) +
  geom_histogram() +
  facet_wrap(~ind, scales = "free") +
  geom_vline(data = truevalue, aes(xintercept = true)) +
  ggtitle("Multinomial idea - Algorithm 2")
```




```r
# Algorithm 3
result3 <- data.frame(n1 =multi_3[1,], n2 = multi_3[12,], n3 = multi_3[25,],
                      n4 = multi_3[38,], n5 = multi_3[50,])
results3 <- stack(result3)

ggplot(data = results3, aes(values, fill = ind)) +
  geom_histogram() +
  facet_wrap(~ind, scales = "free") +
  geom_vline(data = truevalue, aes(xintercept = true)) +
  ggtitle("Multinomial idea - Algorithm 3")
```




### Comparison

### Variance

```r
df_variance <- data.frame(n1 = c(var(result1$n1), var(result2$n1), var(result3$n1)),
                          n2 = c(var(result1$n2), var(result2$n2), var(result3$n2)),
                          n3 = c(var(result1$n3), var(result2$n3), var(result3$n3)),
                          n4 = c(var(result1$n4), var(result2$n4), var(result3$n4)),
                          n5 = c(var(result1$n5), var(result2$n5), var(result3$n5)))

rownames(df_variance) <- c("Algorithm 1", "Algorithm 2", "Algorithm 3")
kable(df_variance, digits = 4, caption = "Variances with the multinomial idea")
```

Table 1: Variances with the multinomial idea

| | n1 | n2 | n3 | n4 | n5 |
|---|---|---|---|---|---|
| Algorithm 1 | 1698327 | 557229.4 | 955978.8 | 1605238 | 558714.9 |
| Algorithm 2 | 1671832 | 535338.0 | 944843.7 | 1624088 | 548134.7 |
| Algorithm 3 | 1687512 | 539428.7 | 961045.7 | 1669455 | 538233.4 |

### Bias

```r
df_bias <- data.frame(n1 = c(bias(result1$n1, rep(N[1], n)), bias(result2$n1, rep(N[1], n)),
                             bias(result3$n1, rep(N[1], n))),
                      n2 = c(bias(result1$n2, rep(N[12], n)), bias(result2$n2, rep(N[12], n)),
```

```
                            bias(result3$n2, rep(N[12], n))),
                     n3 = c(bias(result1$n3, rep(N[25], n)), bias(result2$n3, rep(N[25], n)),
                            bias(result3$n3, rep(N[25], n))),
                     n4 = c(bias(result1$n4, rep(N[38], n)), bias(result2$n4, rep(N[38], n)),
                            bias(result3$n4, rep(N[38], n))),
                     n5 = c(bias(result1$n5, rep(N[50], n)), bias(result2$n5, rep(N[50], n)),
                            bias(result3$n5, rep(N[50], n))))

rownames(df_bias) <- c("Algorithm 1", "Algorithm 2", "Algorithm 3")
kable(df_bias, digits = 4, caption = "Bias with the multinomial idea")
```

Table 2: Bias with the multinomial idea

| | n1 | n2 | n3 | n4 | n5 |
|---|---|---|---|---|---|
| Algorithm 1 | 8.3049 | 4.7117 | -0.7292 | -3.7677 | 0.7500 |
| Algorithm 2 | 7.1376 | 5.3906 | 6.4192 | 12.1823 | 0.5266 |
| Algorithm 3 | 9.5990 | -3.1670 | 6.5854 | -6.7299 | 11.3924 |

**Mean Squared Error**

```
df_mse <- data.frame(n1 = c(mean((result1$n1 - N[1])^2), mean((result2$n1 - N[1])^2),
                            mean((result3$n1 - N[1])^2)),
                     n2 = c(mean((result1$n2 - N[12])^2), mean((result2$n2 - N[12])^2),
                            mean((result3$n2 - N[12])^2)),
                     n3 = c(mean((result1$n3 - N[25])^2), mean((result2$n3 - N[25])^2),
                            mean((result3$n3 - N[25])^2)),
                     n4 = c(mean((result1$n4 - N[38])^2), mean((result2$n4 - N[38])^2),
                            mean((result3$n4 - N[38])^2)),
                     n5 = c(mean((result1$n5 - N[50])^2), mean((result2$n5 - N[50])^2),
                            mean((result3$n5 - N[50])^2)))

rownames(df_mse) <- c("Algorithm 1", "Algorithm 2", "Algorithm 3")
kable(df_mse, digits = 4, caption = "MSE with the multinomial idea")
```

Table 3: MSE with the multinomial idea

| | n1 | n2 | n3 | n4 | n5 |
|---|---|---|---|---|---|
| Algorithm 1 | 1698226 | 557195.9 | 955883.7 | 1605091 | 558659.6 |
| Algorithm 2 | 1671716 | 535313.5 | 944790.4 | 1624074 | 548080.1 |
| Algorithm 3 | 1687436 | 539384.8 | 960993.0 | 1669334 | 538309.4 |