hmrk 7 Beimnet Taye

2023-03-08

Worked with Joan Shim and Lucas Yoshida

P1

1

- The input for the estimand is the underlying total population outputs of a given RV. The output is a measure of the RV values from the total population inputted (mean, median, etc).

2

- The input for the estimator is the sampled data of a given RV and the output is a measure of said data (mean, median, etc.) which is an estimate of the estimand.

3

- The estimand is the true value of a measure in a population. The estimator tries to estimate the estimand using sample data.

4.

Bias and Variance:

- $Bias = \mu(\hat{\psi}) - \psi$
- $= 2 - 5$
- $= -3$
- $Var(\hat{\psi}) = E[\hat{\psi}^2] - E[\hat{\psi}]^2$
- $= 4 - 4$
- $= 0$

P2

3

```r
dgp = function(n) {
tibble(X = rnorm(n)) # your code here
}
sigma_a2_estimator = function(data) {
n = nrow(data)
data %$% sum((X - mean(X))^2) / n
}
sigma_b2_estimator = function(data) {
n = nrow(data)
data %$% sum((X - mean(X))^2) / (n-1)
}

bias_raw <- function(dgp,n,rep =1000){
    map_df(1:rep, function(.x){
      obs <- dgp(n)
      return(
        tibble(
         sample_size = n,
         alpha = sigma_a2_estimator(obs),
         beta = sigma_b2_estimator(obs)
             )
        )
      }
    )
}

bias_eval <- function(dgp,n){
  bias_raw(dgp,n) %>%
    mutate(estimand = var(dgp(100000000)$X)) %>%
    summarize(alpha_m = mean(alpha),
              beta_m = mean(beta),
              true_var = mean(estimand),
              sample_size = mean(n),
              bias_a = alpha_m - true_var,
              bias_b = beta_m - true_var
              )

}

small <- bias_eval(dgp,10)
small
```

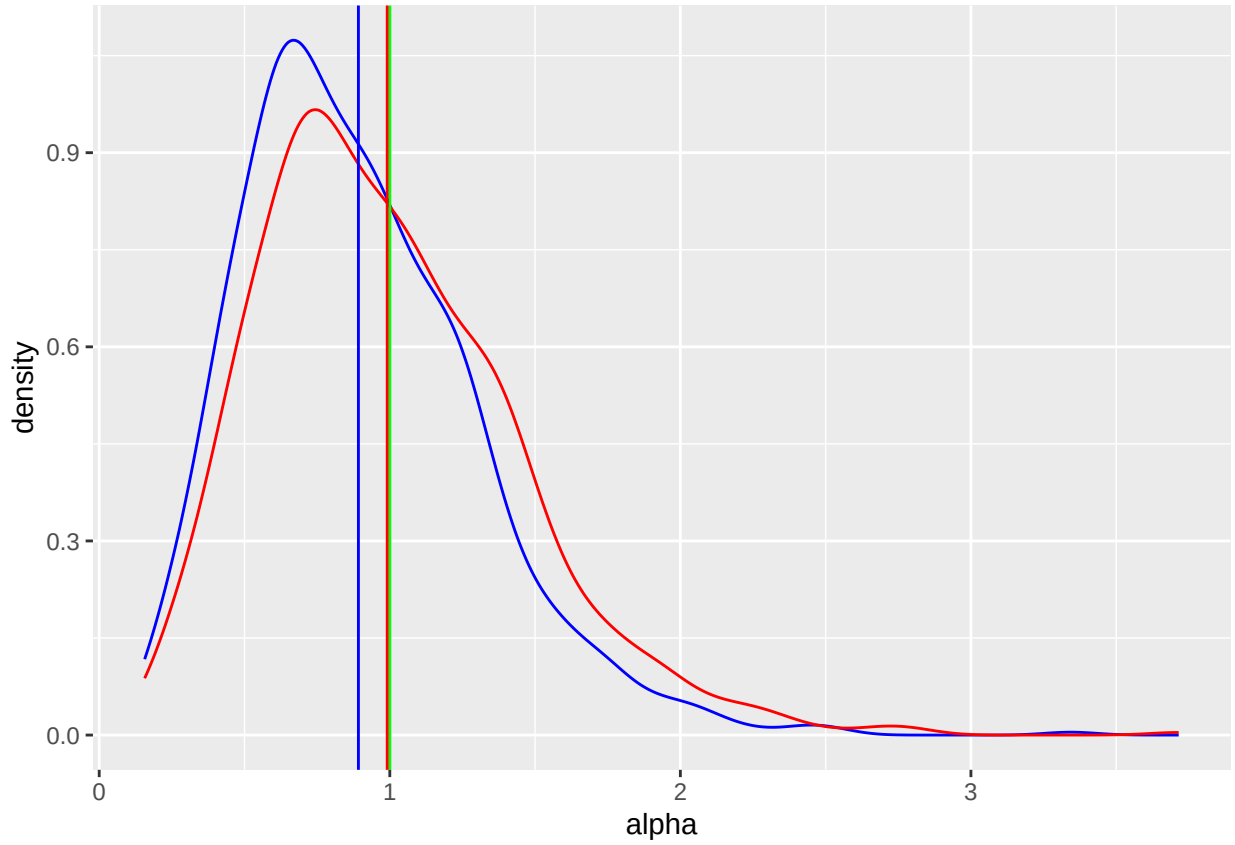
```
## # A tibble: 1 x 6
##   alpha_m beta_m true_var sample_size bias_a  bias_b
##     <dbl>  <dbl>    <dbl>       <dbl>  <dbl>   <dbl>
## 1   0.875  0.972     1.00          10 -0.125 -0.0278
```

```r
large <- bias_eval(dgp,100000)
large
```

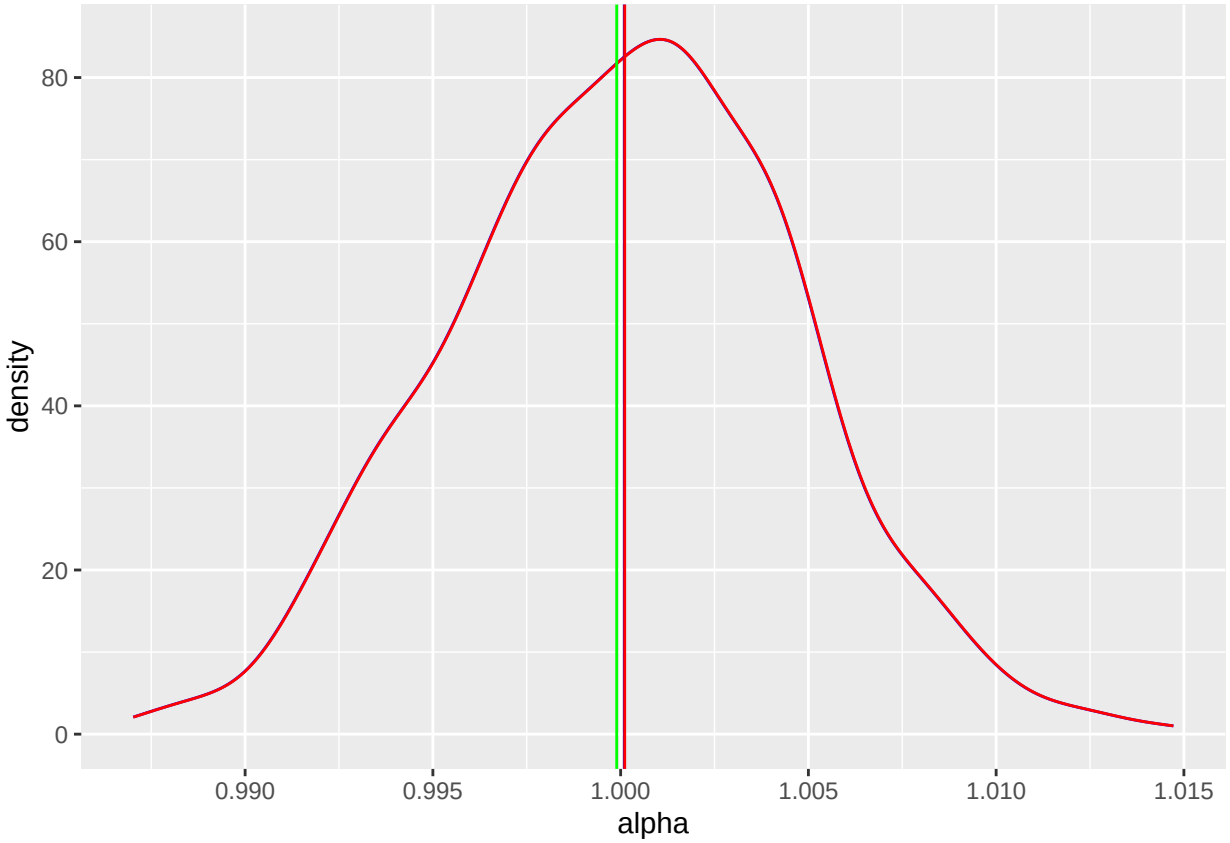
```
## # A tibble: 1 x 6
##   alpha_m beta_m true_var sample_size     bias_a     bias_b
##     <dbl>  <dbl>    <dbl>       <dbl>      <dbl>      <dbl>
```

```
## 1    1.00   1.00     1.00      100000 -0.0000277 -0.0000177
```

```
small_densities <- bias_raw(dgp,10) %>%
  ggplot() +
  geom_density(aes(x = alpha), color = "blue") +
  geom_density(aes(x = beta), color = "red") +
  geom_vline(xintercept = var(dgp(10000000)$X), color = "green") +
  geom_vline(aes(xintercept = mean(alpha)), color = "blue") +
  geom_vline(aes(xintercept = mean(beta)), color = "red")
small_densities
```




```
large_densities <- bias_raw(dgp,100000) %>%
  ggplot() +
  geom_density(aes(x = alpha), color = "blue") +
  geom_density(aes(x = beta), color = "red") +
  geom_vline(xintercept = var(dgp(10000000)$X), color = "green") +
  geom_vline(aes(xintercept = mean(alpha)), color = "blue") +
  geom_vline(aes(xintercept = mean(beta)), color = "red")
large_densities
```



- Based off the calculated tables the bias is larger with the alpha method when the sample size is small and close to 0 when the sample is larger. With the generated densities the mean for the alpha method is farther away from the estimand value of one for the smaller sample size.

4.

- The analytic proof is more robust and grounded in raw theory but is much harder to parse through. The computational method is a lot more readable and intuitive but scarifies some robustness.

5.

- The beta method while less biased than the alpha method actually has a higher amount of variance. This is called the bias-variance tradeoff and might be a reason you might want to use the alpha method instead.

P3

1.

```
logistic <- function(x){
  1/(1+exp(-x))
}
```

```
DGP_o <- function(n){tibble(
  X = runif(n),
  W = rbern(n, prob = logistic(X)),
  Y = rbern(n, prob = logistic(X+W))
)
}

data <- DGP_o(10000) %>%
  group_by(W) %>%
  summarize(one_probs = mean(Y),
            zero_probs = 1 - one_probs,
            odds = one_probs/zero_probs)
odds_ratio <- pull(data[2,4])/pull(data[1,4])
odds_ratio
```

```
## [1] 3.00397
```

P4

1.

```
dgp_components = list(
covariates = function(n) {
tibble(
X1 = rnorm(n),
X2 = rnorm(n),
X3 = rnorm(n),
  )
},

response = function(data) {
n = nrow(data)
data %>%
mutate(
D = rbern(n, logistic(-X1 - 2*X2 + 3*X3)),
  )
},

opiates = function(data) {
n = nrow(data)
data %>%
mutate(
Y = rbern(n, logistic(X1 + X2 - 2*X3 - 3)),
  )
}
)

true_dgp = function(n) {
dgp_components %$% {
covariates(n) %>%
response %>%
```

```
opiates
  }
}

observed_dgp = function(n) {
true_dgp(n) %>%
mutate(Y = ifelse(D, Y, NA))
}
```

```
Estimand_Y <- mean(true_dgp(100000)$Y)
Estimand_Y
```

```
## [1] 0.1584
```

2.

```
unadjusted = function(data){
mean(data$Y, na.rm=T)
}

regression = function(data){
data %>%
filter(!is.na(Y)) %>%
glm(Y ~ X1+X2+X3, data=., family=binomial) %>%
predict(data, type='response') %>%
mean()
}

weighted = function(data){
data %>%
glm(D ~ X1+X2+X3, data=., family=binomial) %>%
predict(data, type='response') %>%
mutate(data, p=.) %>%
filter(!is.na(Y)) %$%
    { sum(Y/p) / nrow(data) }

}
```

```
observed <- observed_dgp(500)

estimates <- tibble(
  unadjusted = unadjusted(observed),
  regression = regression(observed),
  weighted = weighted(observed)
)
estimates
```

```
## # A tibble: 1 x 3
##   unadjusted regression weighted
##        <dbl>      <dbl>    <dbl>
## 1     0.0153     0.0493   0.0109
```

3.

```
bias_rawd <- function(dgp,n=500,rep=10){
    map_df(1:rep, function(.x){
      obs <- dgp(n)
      return(
        tibble(
         sample_size = n,
         unadjusted = unadjusted(obs),
         regression = regression(obs),
         weighted = weighted(obs)
             )
        )
      }
    )
}

bias_evald <- function(dgp,true_dgp,n=500,rep=10) {
  bias_rawd(dgp, n=500,rep=10) %>%
    mutate(estimand = mean(true_dgp(1000000)$Y)) %>%
    summarize(bias_unadusted = mean(unadjusted)-mean(estimand),
              var_unadjusted = var(unadjusted),
              bias_regression =  mean(regression)-mean(estimand),
              var_regression = var(regression),
              bias_weighted = mean(unadjusted) - mean(estimand),
              var_weighted = var(weighted))
}
result <- bias_evald(observed_dgp,true_dgp, rep = 10000)
final <- tibble(Estimator = c("unadjusted","regression","weighted"),
                bias = as.numeric(c(result[1,1],result[1,3],result[1,5])),
                variance = as.numeric(c(result[1,2],result[1,4],result[1,6]))
                )
final
```

```
## # A tibble: 3 x 3
##   Estimator      bias variance
##   <chr>         <dbl>    <dbl>
## 1 unadjusted -0.135   0.000132
## 2 regression -0.00295 0.00153
## 3 weighted   -0.135   0.00250
```

4.

- I would use the regression estimator since it has the lowest amount of bias by a fairly large margin relative to the other two estimators. Yes it would since estimators in general are dependent on the underlying DGP. The best estimator we choose is usually based upon assumptions we make about the underlying distribution, such as how the underlying population data is distributed. If the DGP were to change our assumptions used to choose an estimator would likely no longer hold and we would have to make adjustments.

P5

1.

- $\sigma^2 = V[\hat{\mu}]$
- $= V[\frac{1}{n}\sum_i X_i]$
- $= \frac{1}{n^2}V[\sum_i X_i]$ Linearity of variance.
- $= \frac{1}{n^2}\sum_i V[X_i]$
- $= \frac{1}{n^2}V[X] * n$ The sum of the variances of an IID RV is the just the variance of the sum.
- $= \frac{V[X]n}{n^2}$
- $= \frac{V[X]}{n}$

2.

- We can arrive at this since variance is defined as:
- $V[\hat{X}] = E[(\hat{X} - E[\hat{X}])^2]$
- Using the above equation we can use the plug in estimators for expectation:
- $E[\hat{X}] = \frac{1}{n}\sum_i X_i$
- $V[\hat{X}] = \frac{1}{n}\sum_i(\hat{X} - (\frac{1}{n}\sum_i X_i))^2$
- Given:
- $\hat{\mu} = \frac{1}{n}\sum_i X_i$
- We can plug this in and we get:
- $V[\hat{X}] = \frac{1}{n}\sum_i(\hat{X} - \hat{\mu})^2$
- We can use the plug in estimator for the variance,derived in the previous question, into the equation:
- $\hat{\sigma^2} = \frac{V[\hat{X}]}{n}$
- $= \frac{\frac{1}{n}\sum_i(\hat{X}-\hat{\mu})^2}{n}$
- $= \frac{1}{n^2}\sum_i(\hat{X} - \hat{\mu})^2$
- Finally the Standard deviation of the sampling distribution, called the standard error, is just the square root of its variance:
- $\sigma = \sqrt{\frac{1}{n^2}\sum_i(\hat{X} - \hat{\mu})^2}$

3.

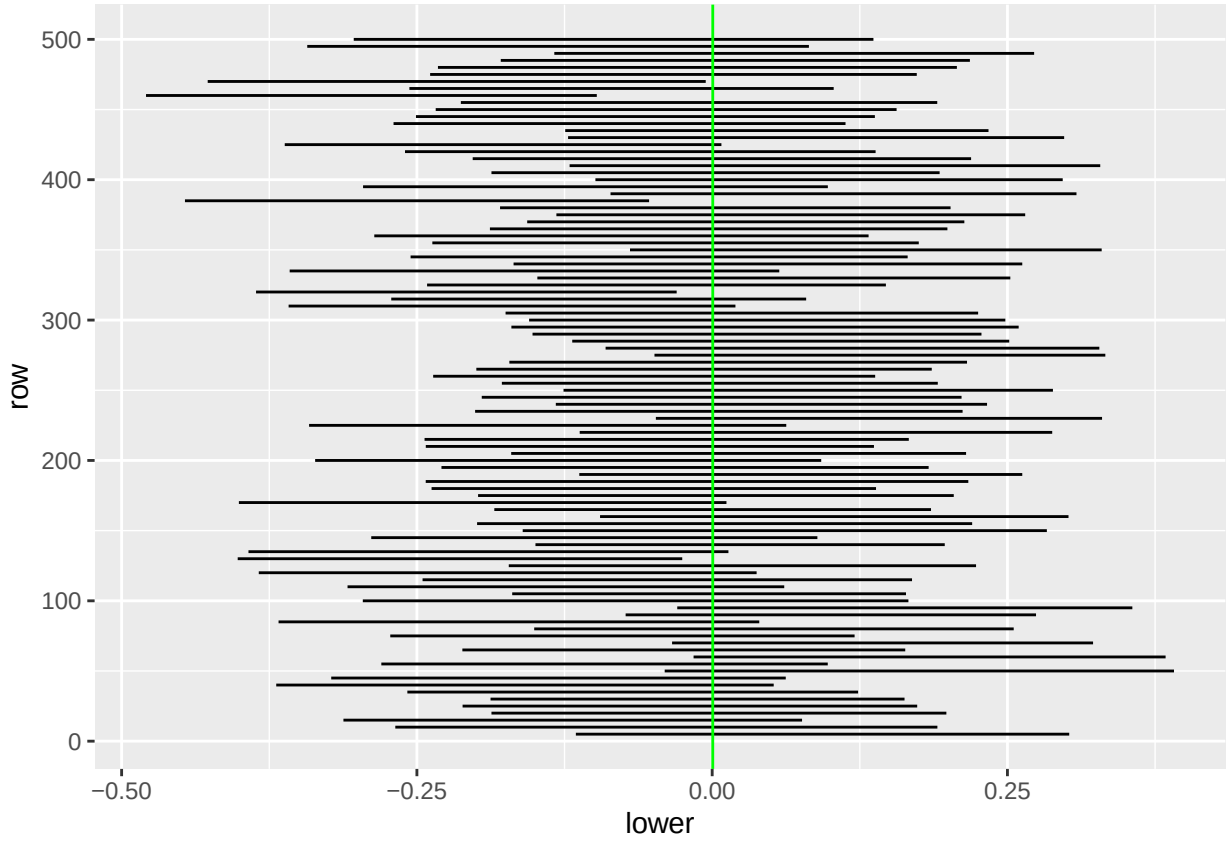
```
# Diff distributions
dgp_5 <- function(n) {
  return(rnorm(n))
}
dgp_6 <- function(n) {
  return(rexp(n))
}
dgp_7 <- function(n){
  return(runif(n))
}
#STE Calc
STE <- function(dgp,n) {
  sigma <- sqrt((1/n^2)*sum((dgp-mean(dgp))^2))
  return(sigma)
}
#table with interval values
interval <- function(dgp,n, rep=100) {
```

```
  map_df(1:rep,function(.x){
      data <- dgp(n)
      Sig <- STE(data,n)
      return(
        tibble(
        mu = mean(data),
        upper = mu + 2*Sig,
        lower = mu - 2*Sig,
        sample_size = n
        )
      )
    }
  ) %>%
    mutate(row = row_number()*5)
}
# interval visualization
int_visual <- function(dgp,n,rep=100){
  interval(dgp,n,rep) %>%
  ggplot() +
  geom_segment(aes(x = lower, xend = upper, y = row, yend = row)) +
  geom_vline(xintercept = mean(dgp(1000000)), color = "green")
}

# normal
# n = 100
normal_table_100 <- interval(dgp_5,100)
normal_table_100
```

```
## # A tibble: 100 x 5
##         mu  upper   lower sample_size   row
##      <dbl>  <dbl>   <dbl>       <dbl> <dbl>
##  1  0.228  0.433   0.0232         100     5
##  2 -0.0180 0.204  -0.240          100    10
##  3  0.169  0.367  -0.0285         100    15
##  4  0.144  0.349  -0.0612         100    20
##  5  0.0705 0.264  -0.122          100    25
##  6  0.0153 0.210  -0.179          100    30
##  7  0.0571 0.238  -0.124          100    35
##  8  0.0932 0.330  -0.143          100    40
##  9  0.127  0.316  -0.0631         100    45
## 10 -0.0949 0.0948 -0.285          100    50
## # ... with 90 more rows
```

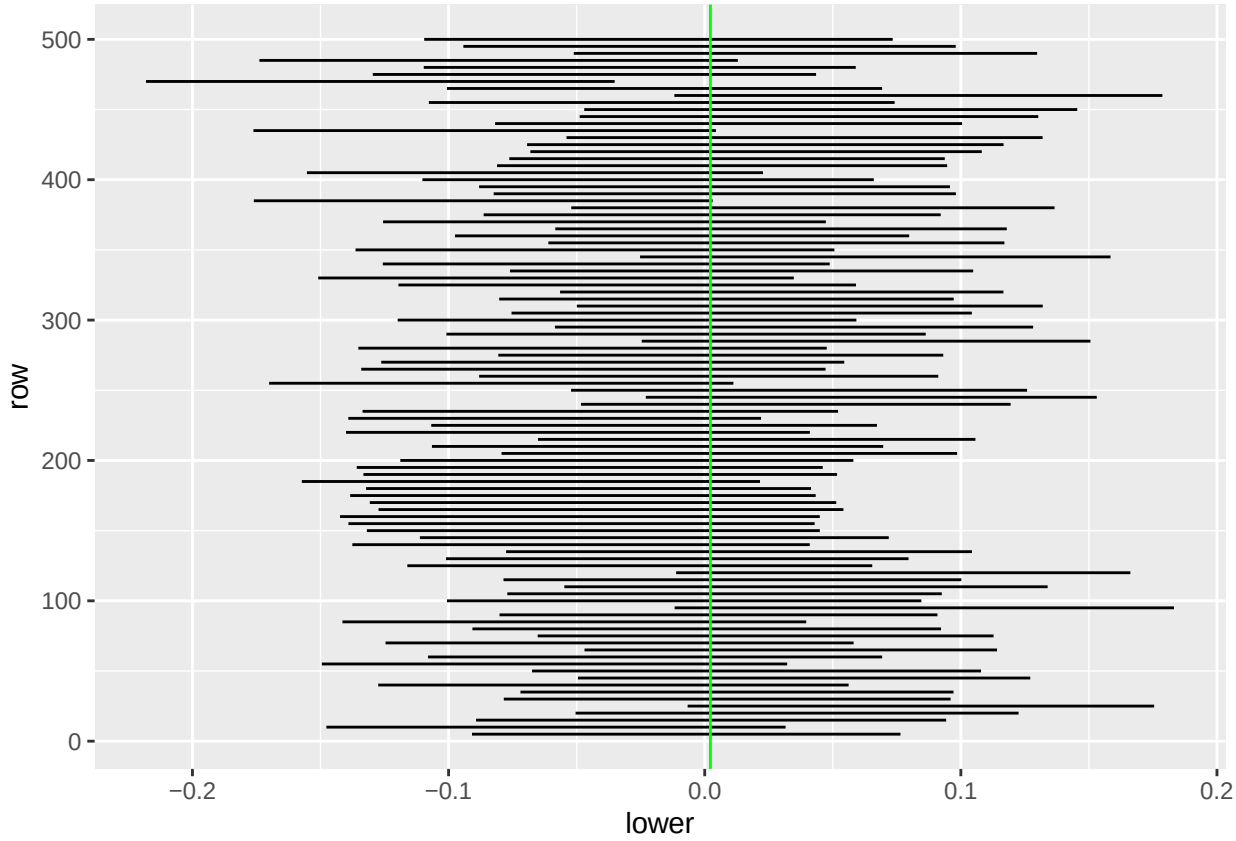
```
normal_visual_100 <- int_visual(dgp_5,100)
normal_visual_100
```




```
# n = 500
normal_table_500 <- interval(dgp_5,500)
normal_table_500
```

```
## # A tibble: 100 x 5
##          mu   upper    lower sample_size   row
##       <dbl>   <dbl>    <dbl>       <dbl> <dbl>
##  1  0.0302  0.119   -0.0584          500     5
##  2  0.0868  0.177   -0.00356         500    10
##  3 -0.0782  0.00824 -0.165           500    15
##  4  0.107   0.200    0.0145          500    20
##  5 -0.0301  0.0644  -0.125           500    25
##  6 -0.0410  0.0467  -0.129           500    30
##  7  0.0113  0.0979  -0.0752          500    35
##  8 -0.00572 0.0818  -0.0932          500    40
##  9 -0.0551  0.0366  -0.147           500    45
## 10 -0.0569  0.0315  -0.145           500    50
## # ... with 90 more rows
```

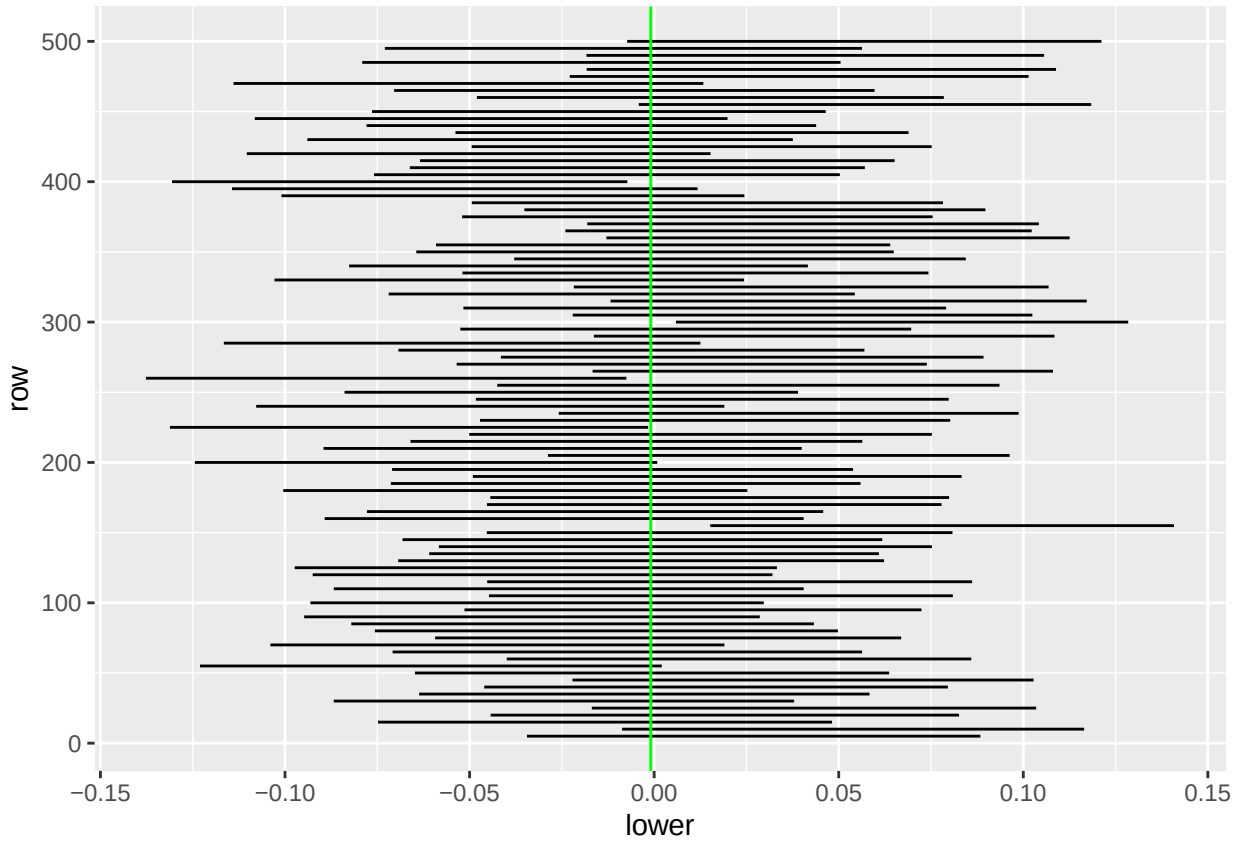
```
normal_visual_500 <- int_visual(dgp_5,500)
normal_visual_500
```




```
# n = 1000
normal_table_1000 <- interval(dgp_5,1000)
normal_table_1000
```

```
## # A tibble: 100 x 5
##          mu   upper    lower sample_size   row
##       <dbl>   <dbl>    <dbl>       <dbl> <dbl>
##  1 -0.0796  -0.0175 -0.142          1000     5
##  2 -0.0343   0.0298 -0.0983         1000    10
##  3  0.0442   0.109  -0.0203         1000    15
##  4 -0.00360  0.0615 -0.0687         1000    20
##  5  0.00919  0.0721 -0.0537         1000    25
##  6 -0.0366   0.0270 -0.100          1000    30
##  7  0.0689   0.133   0.00485        1000    35
##  8  0.00255  0.0653 -0.0602         1000    40
##  9 -0.00925  0.0553 -0.0738         1000    45
## 10 -0.0194   0.0433 -0.0820         1000    50
## # ... with 90 more rows
```

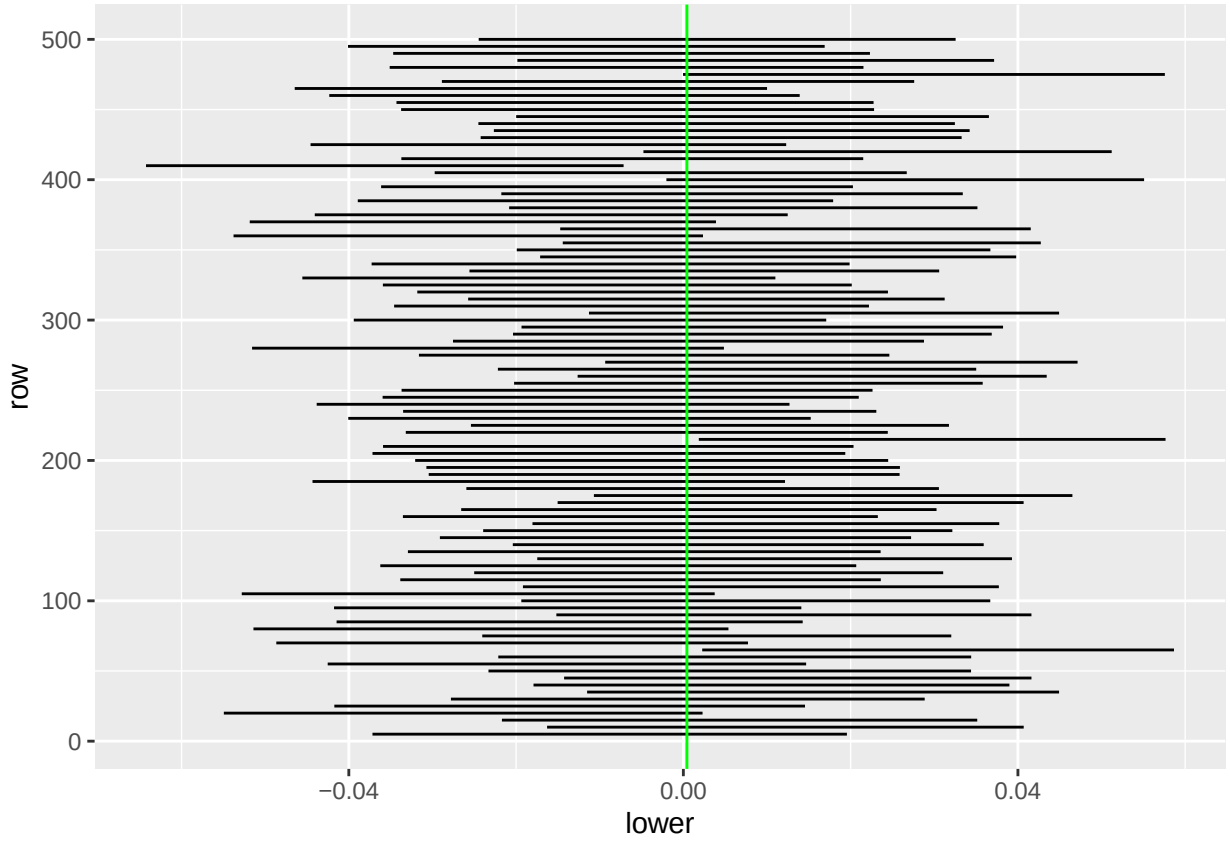
```
normal_visual_1000 <- int_visual(dgp_5,1000)
normal_visual_1000
```




```
# n = 5000
normal_table_5000 <- interval(dgp_5,5000)
normal_table_5000
```

```
## # A tibble: 100 x 5
##          mu  upper    lower sample_size   row
##       <dbl>  <dbl>    <dbl>       <dbl> <dbl>
##  1 -0.00598 0.0220 -0.0339         5000     5
##  2  0.00563 0.0336 -0.0223         5000    10
##  3  0.0137  0.0423 -0.0149         5000    15
##  4  0.0106  0.0388 -0.0176         5000    20
##  5  0.00145 0.0294 -0.0265         5000    25
##  6 -0.00193 0.0267 -0.0306         5000    30
##  7  0.0213  0.0496 -0.00693        5000    35
##  8  0.0141  0.0422 -0.0140         5000    40
##  9 -0.00761 0.0209 -0.0361         5000    45
## 10 -0.0132  0.0145 -0.0409         5000    50
## # ... with 90 more rows
```

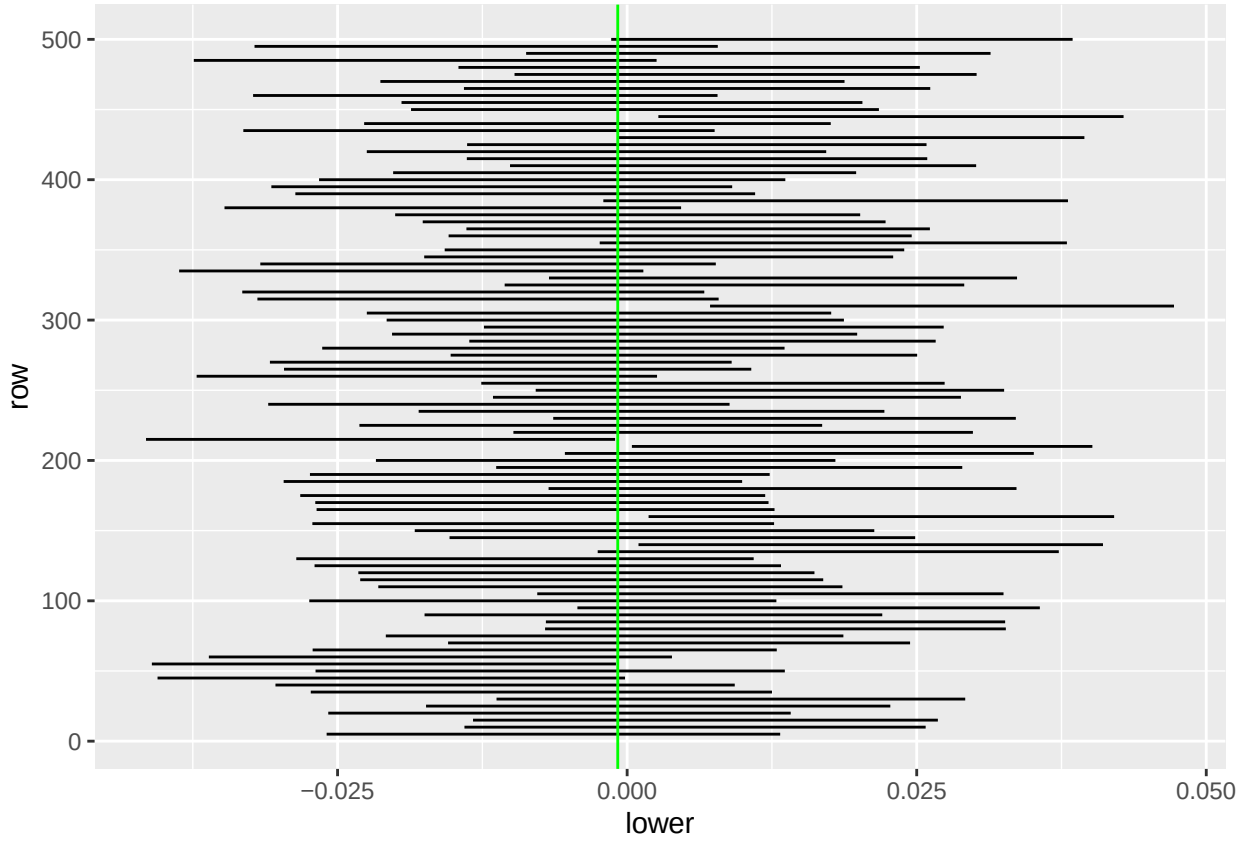
```
normal_visual_5000 <- int_visual(dgp_5,5000)
normal_visual_5000
```




```
# n = 10000
normal_table_10000 <- interval(dgp_5,10000)
normal_table_10000
```

```
## # A tibble: 100 x 5
##          mu   upper    lower sample_size   row
##       <dbl>   <dbl>    <dbl>       <dbl> <dbl>
##  1  0.0119  0.0318  -0.00800       10000     5
##  2 -0.00277 0.0172  -0.0228        10000    10
##  3 -0.0131  0.00698 -0.0331        10000    15
##  4  0.00134 0.0212  -0.0185        10000    20
##  5 -0.00351 0.0166  -0.0236        10000    25
##  6  0.0108  0.0306  -0.00903       10000    30
##  7  0.00767 0.0276  -0.0123        10000    35
##  8 -0.00264 0.0173  -0.0226        10000    40
##  9  0.0151  0.0352  -0.00493       10000    45
## 10 -0.0109  0.00932 -0.0311        10000    50
## # ... with 90 more rows
```

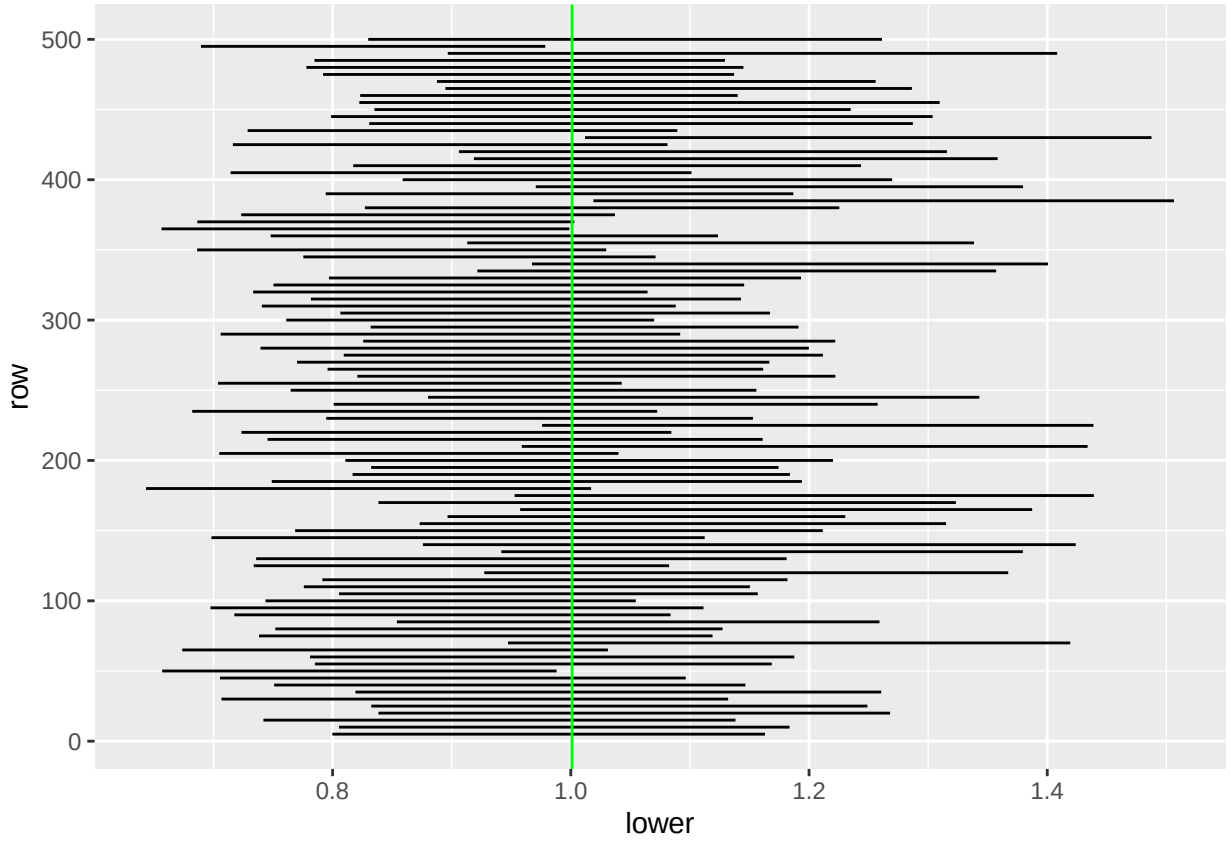
```
normal_visual_10000 <- int_visual(dgp_5,10000)
normal_visual_10000
```




```
# exponential
# n = 100
exp_table_100 <- interval(dgp_6,100)
exp_table_100
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.860  1.02 0.702         100     5
##  2 1.03   1.21 0.852         100    10
##  3 1.11   1.34 0.879         100    15
##  4 0.887  1.06 0.710         100    20
##  5 0.942  1.18 0.704         100    25
##  6 0.944  1.14 0.749         100    30
##  7 0.879  1.04 0.714         100    35
##  8 0.929  1.10 0.753         100    40
##  9 0.929  1.10 0.755         100    45
## 10 1.12   1.36 0.888         100    50
## # ... with 90 more rows
```

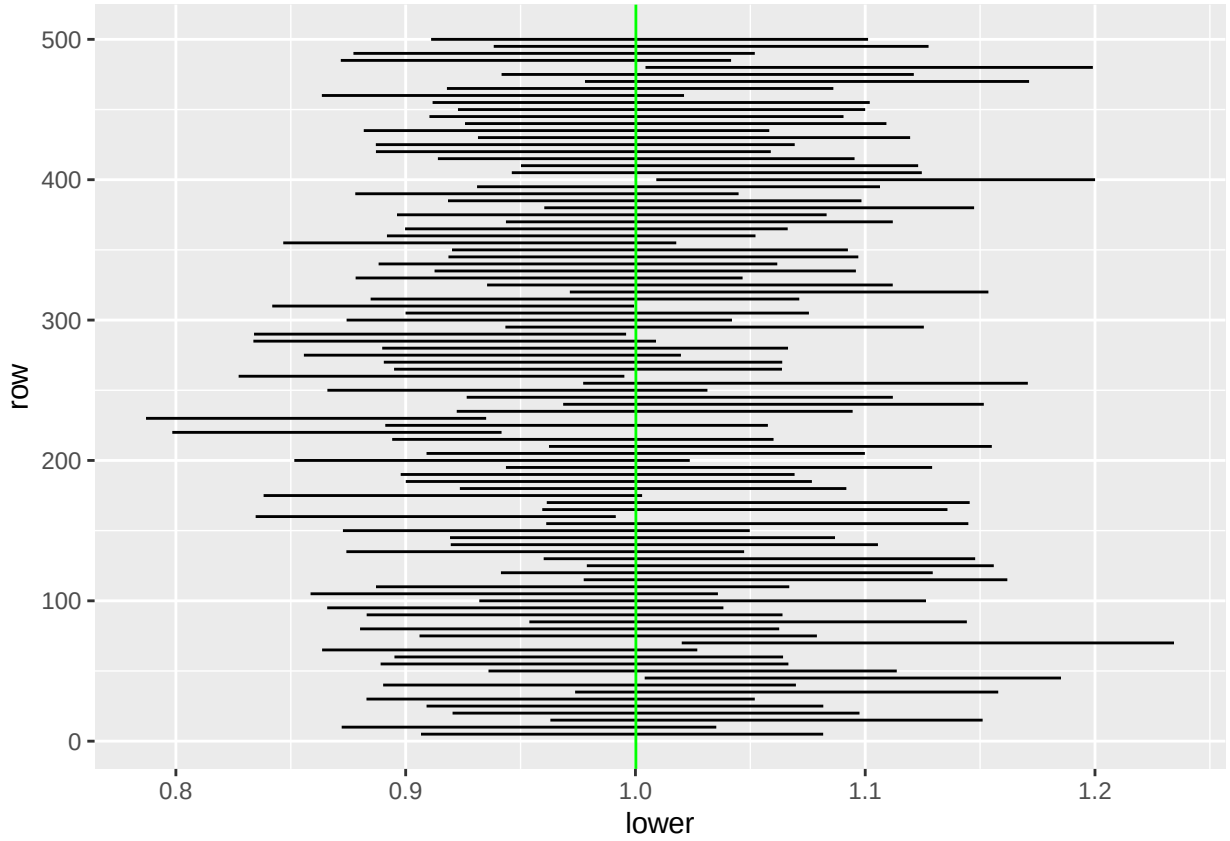
```
exp_visual_100 <- int_visual(dgp_6,100)
exp_visual_100
```




```
# n = 500
exp_table_500 <- interval(dgp_6,500)
exp_table_500
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.967 1.05  0.881         500     5
##  2 0.973 1.06  0.885         500    10
##  3 1.00  1.09  0.916         500    15
##  4 1.01  1.09  0.927         500    20
##  5 0.929 1.01  0.852         500    25
##  6 0.947 1.03  0.864         500    30
##  7 0.883 0.965 0.801         500    35
##  8 0.926 1.01  0.842         500    40
##  9 0.939 1.03  0.852         500    45
## 10 1.01  1.11  0.909         500    50
## # ... with 90 more rows
```

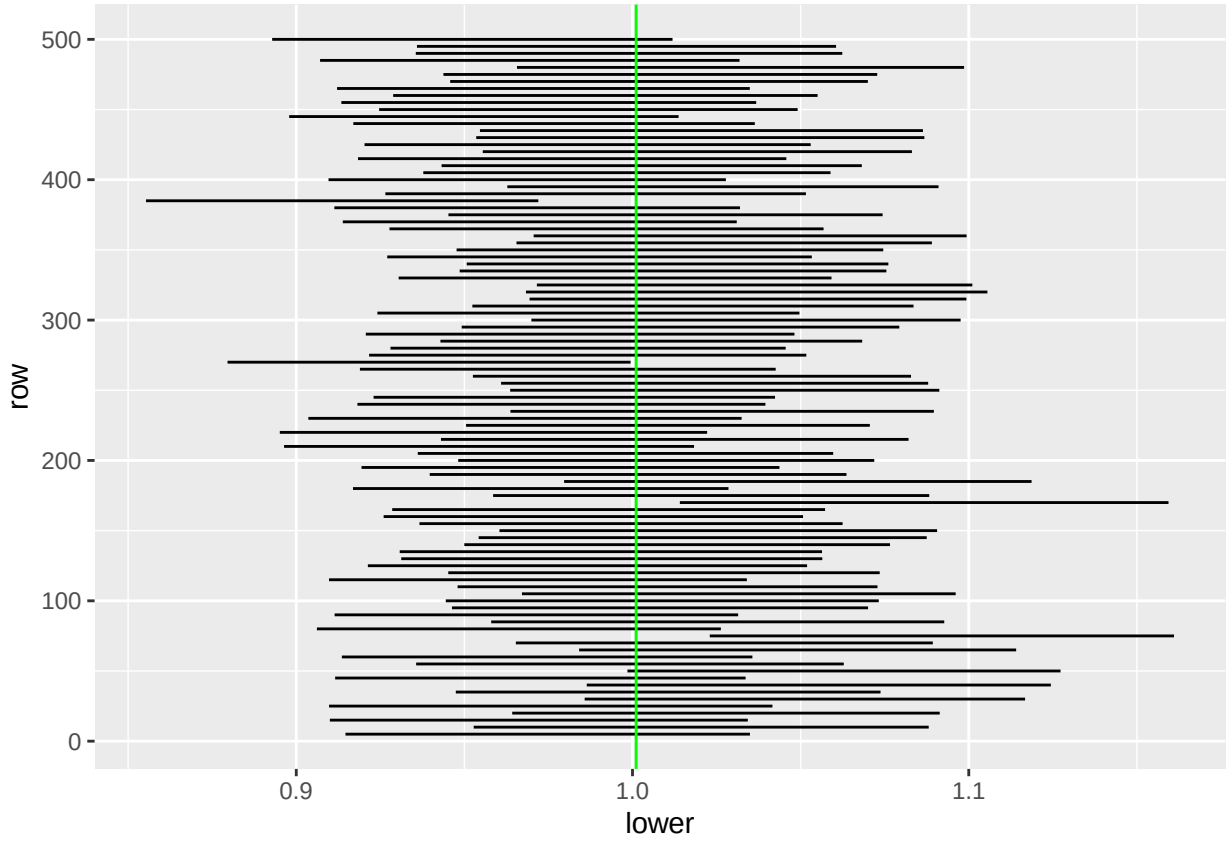
```
exp_visual_500 <- int_visual(dgp_6,500)
exp_visual_500
```




```
# n = 1000
exp_table_1000 <- interval(dgp_6,1000)
exp_table_1000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 1.03   1.10 0.963        1000     5
##  2 1.02   1.08 0.955        1000    10
##  3 1.04   1.10 0.976        1000    15
##  4 0.994  1.06 0.930        1000    20
##  5 1.05   1.12 0.980        1000    25
##  6 0.979  1.04 0.916        1000    30
##  7 0.990  1.06 0.923        1000    35
##  8 1.04   1.11 0.978        1000    40
##  9 0.977  1.04 0.918        1000    45
## 10 1.04   1.10 0.972        1000    50
## # ... with 90 more rows
```

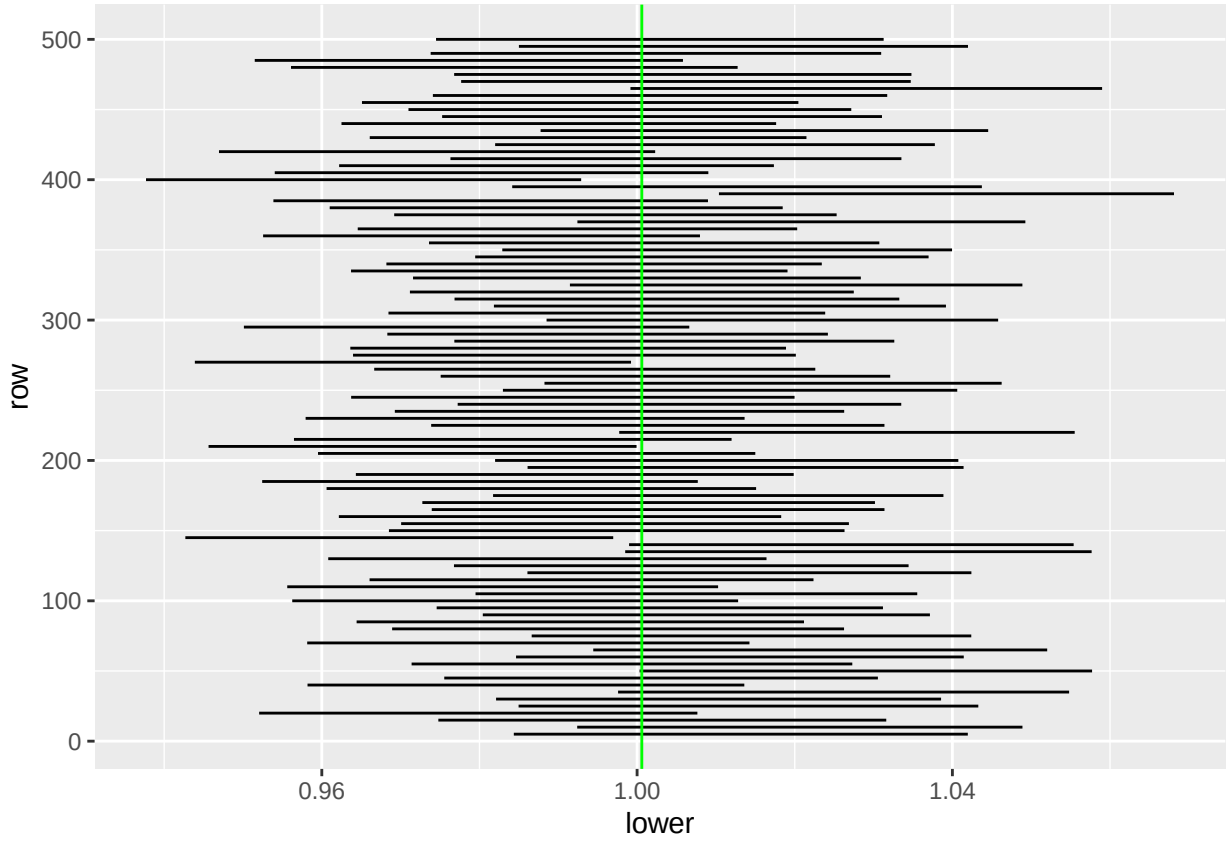
```
exp_visual_1000 <- int_visual(dgp_6,1000)
exp_visual_1000
```




```
# n = 5000
exp_table_5000 <- interval(dgp_6,5000)
exp_table_5000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 1.01   1.04 0.987        5000     5
##  2 0.991  1.02 0.963        5000    10
##  3 1.00   1.03 0.973        5000    15
##  4 1.02   1.05 0.990        5000    20
##  5 1.00   1.03 0.971        5000    25
##  6 0.991  1.02 0.963        5000    30
##  7 0.997  1.03 0.969        5000    35
##  8 1.02   1.05 0.993        5000    40
##  9 0.980  1.01 0.953        5000    45
## 10 0.989  1.02 0.961        5000    50
## # ... with 90 more rows
```

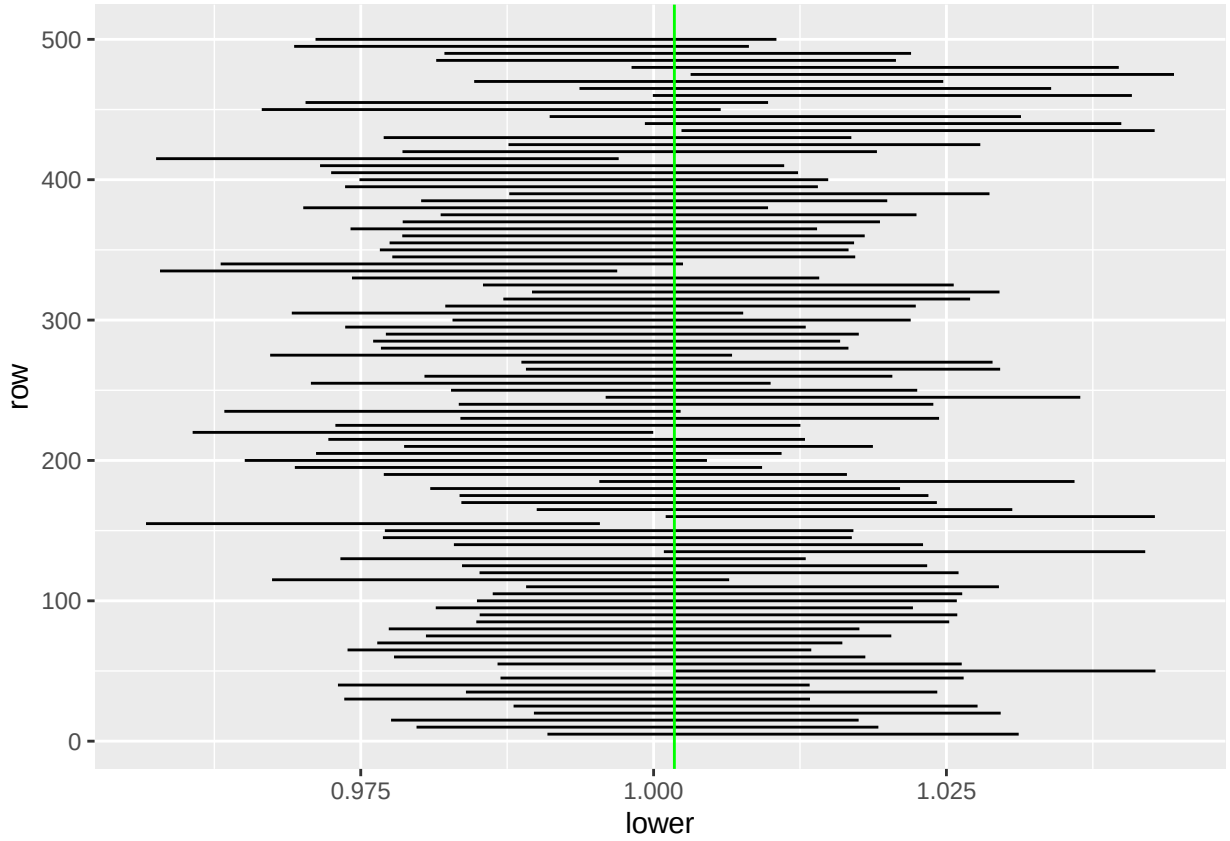
```
exp_visual_5000 <- int_visual(dgp_6,5000)
exp_visual_5000
```




```
# n = 10000
exp_table_10000 <- interval(dgp_6,10000)
exp_table_10000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.992  1.01 0.973       10000     5
##  2 1.00   1.02 0.983       10000    10
##  3 0.980  1.00 0.961       10000    15
##  4 1.00   1.02 0.981       10000    20
##  5 1.00   1.02 0.980       10000    25
##  6 1.01   1.03 0.992       10000    30
##  7 1.01   1.03 0.991       10000    35
##  8 0.993  1.01 0.973       10000    40
##  9 1.00   1.02 0.981       10000    45
## 10 1.01   1.03 0.992       10000    50
## # ... with 90 more rows
```

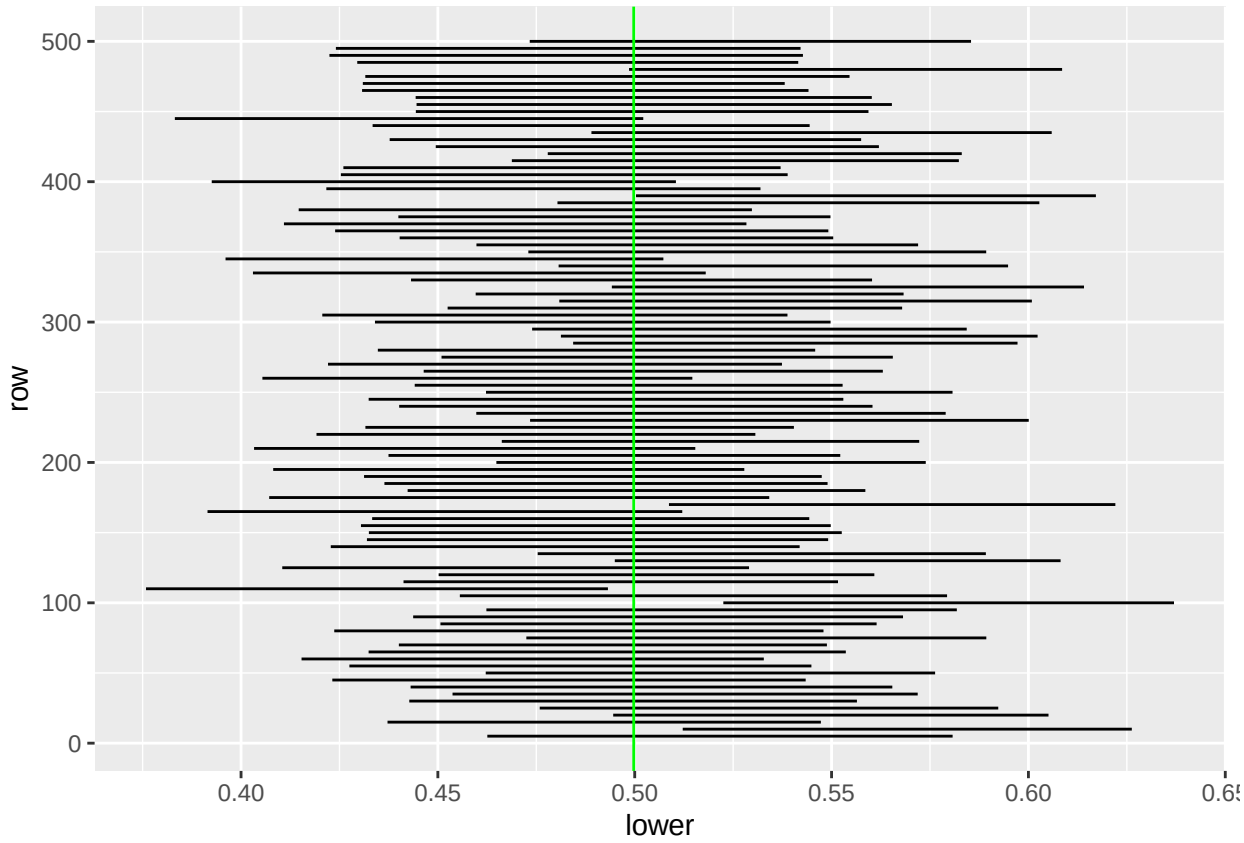
```
exp_visual_10000 <- int_visual(dgp_6,10000)
exp_visual_10000
```




```
# uniform
# n = 100
uni_table_100 <- interval(dgp_7,100)
uni_table_100
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.562 0.618 0.506         100     5
##  2 0.528 0.585 0.470         100    10
##  3 0.495 0.552 0.438         100    15
##  4 0.522 0.580 0.465         100    20
##  5 0.511 0.572 0.451         100    25
##  6 0.451 0.509 0.392         100    30
##  7 0.482 0.540 0.425         100    35
##  8 0.484 0.537 0.432         100    40
##  9 0.532 0.592 0.473         100    45
## 10 0.516 0.575 0.456         100    50
## # ... with 90 more rows
```

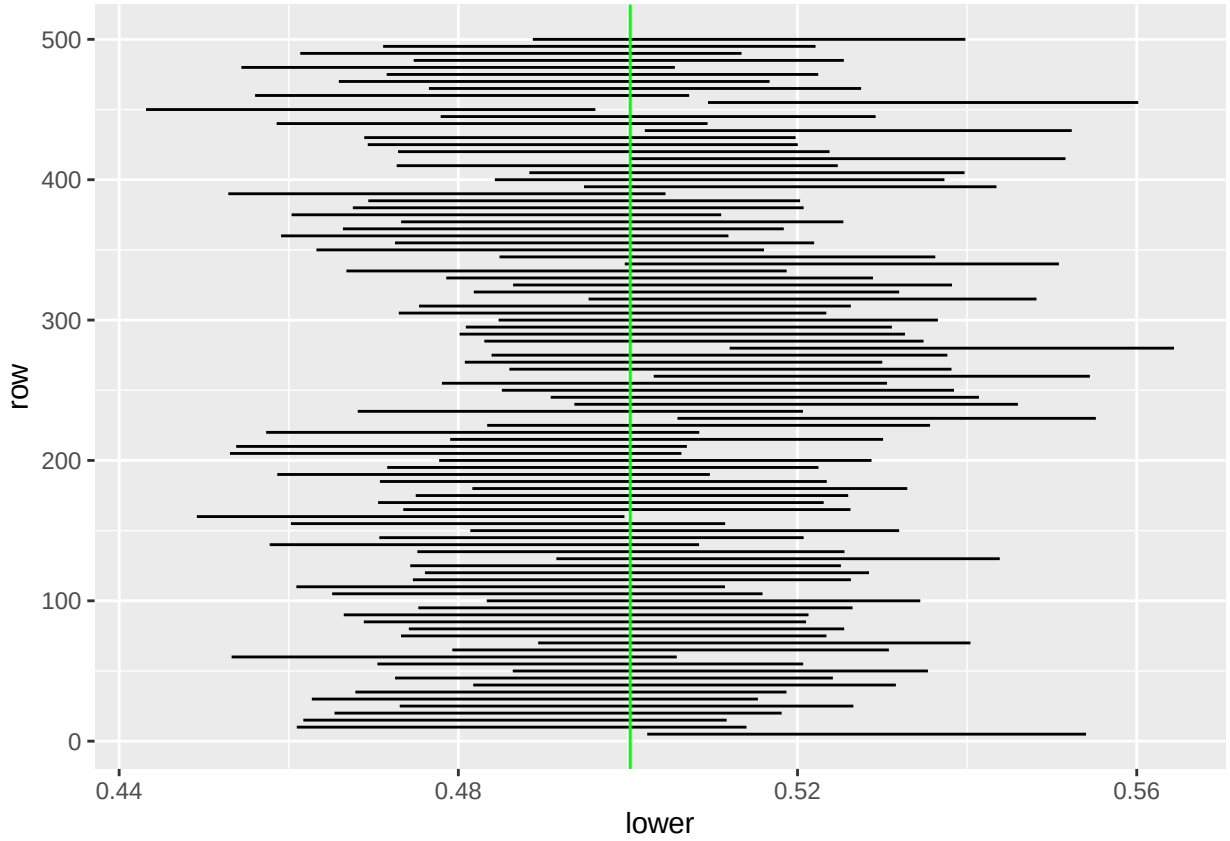
```
uni_visual_100 <- int_visual(dgp_7,100)
uni_visual_100
```




```
# n = 500
uni_table_500 <- interval(dgp_7,500)
uni_table_500
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.496 0.521 0.470         500     5
##  2 0.491 0.517 0.465         500    10
##  3 0.482 0.509 0.456         500    15
##  4 0.501 0.527 0.476         500    20
##  5 0.516 0.542 0.490         500    25
##  6 0.502 0.528 0.476         500    30
##  7 0.504 0.530 0.479         500    35
##  8 0.470 0.497 0.443         500    40
##  9 0.511 0.536 0.485         500    45
## 10 0.497 0.523 0.471         500    50
## # ... with 90 more rows
```

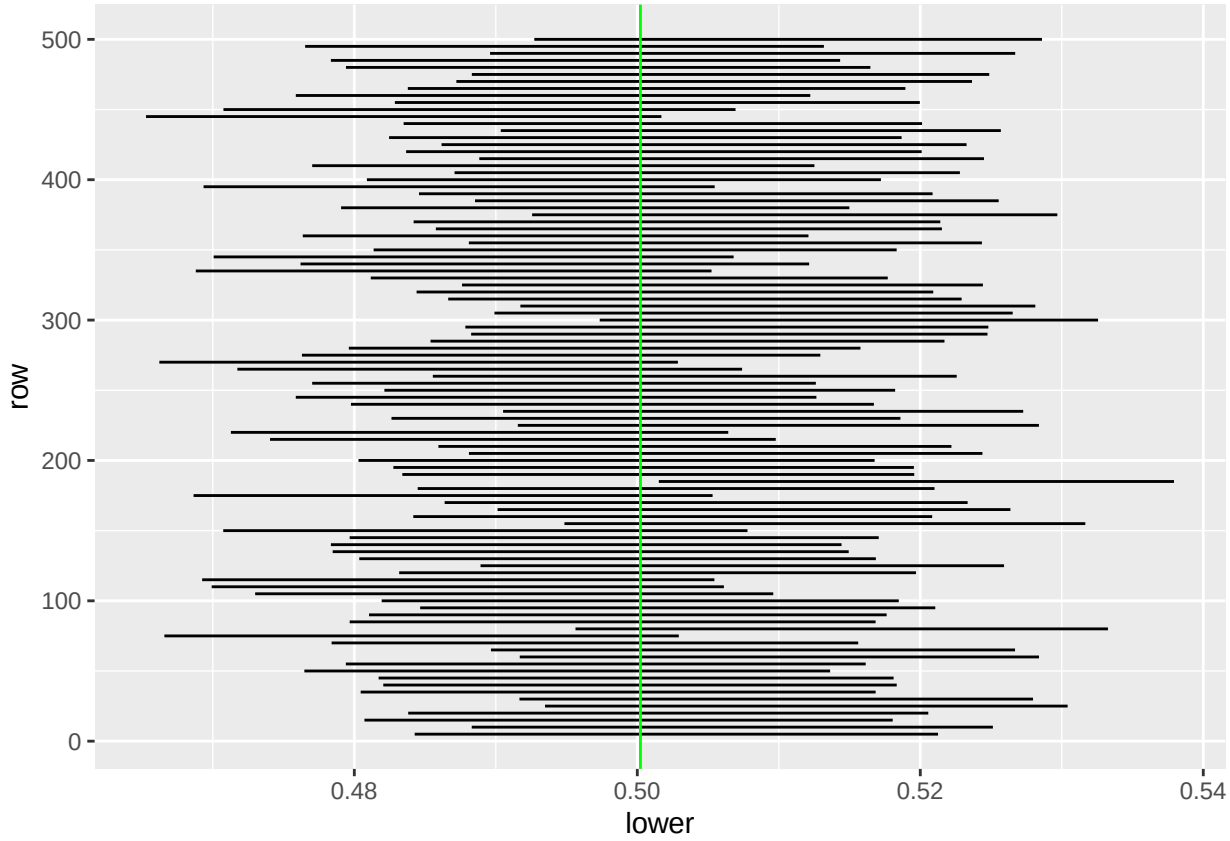
```
uni_visual_500 <- int_visual(dgp_7,500)
uni_visual_500
```




```
# n = 1000
uni_table_1000 <- interval(dgp_7,1000)
uni_table_1000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.492 0.510 0.474        1000     5
##  2 0.503 0.521 0.484        1000    10
##  3 0.502 0.521 0.484        1000    15
##  4 0.517 0.535 0.499        1000    20
##  5 0.527 0.545 0.510        1000    25
##  6 0.491 0.509 0.472        1000    30
##  7 0.498 0.516 0.480        1000    35
##  8 0.508 0.526 0.489        1000    40
##  9 0.494 0.512 0.477        1000    45
## 10 0.509 0.528 0.491        1000    50
## # ... with 90 more rows
```

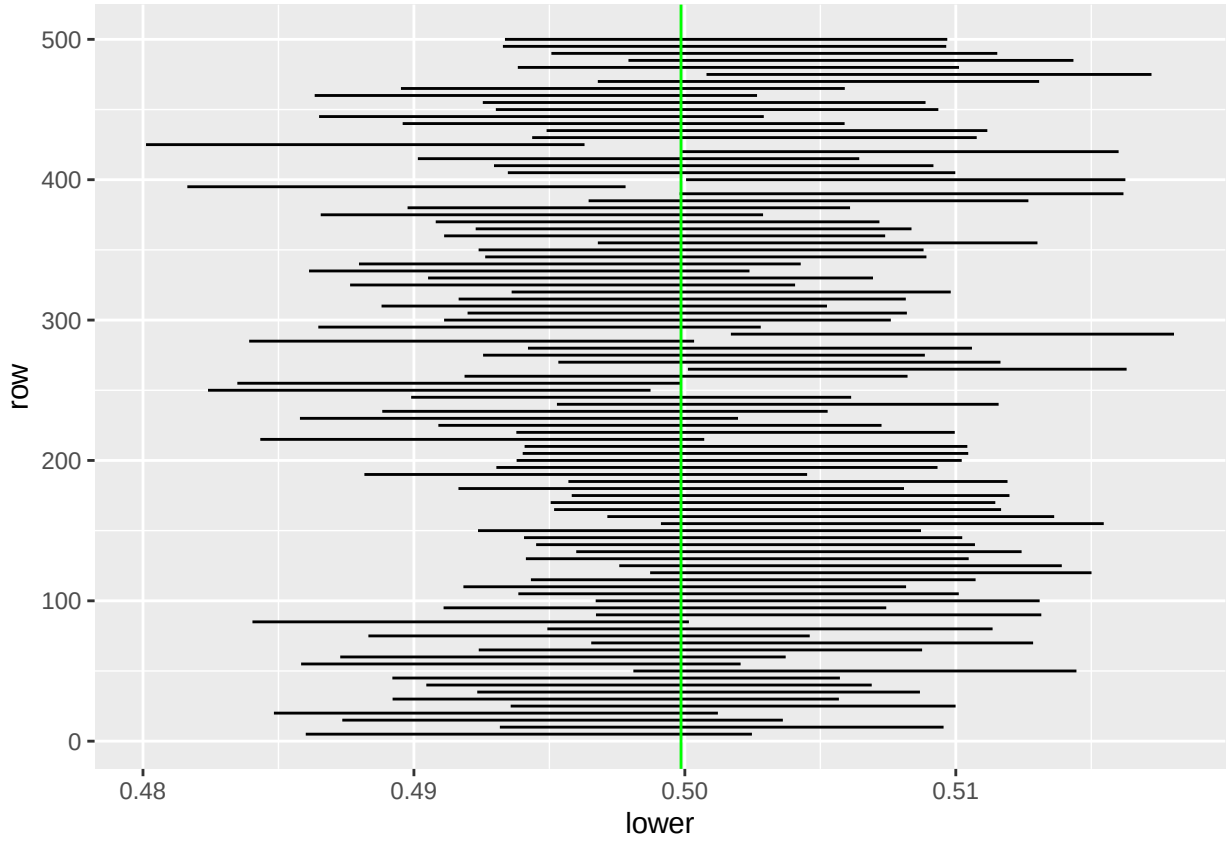
```
uni_visual_1000 <- int_visual(dgp_7,1000)
uni_visual_1000
```




```
# n = 5000
uni_table_5000 <- interval(dgp_7,5000)
uni_table_5000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.503 0.511 0.495        5000     5
##  2 0.504 0.512 0.496        5000    10
##  3 0.498 0.506 0.489        5000    15
##  4 0.503 0.511 0.495        5000    20
##  5 0.503 0.512 0.495        5000    25
##  6 0.496 0.505 0.488        5000    30
##  7 0.496 0.504 0.488        5000    35
##  8 0.500 0.508 0.492        5000    40
##  9 0.505 0.513 0.497        5000    45
## 10 0.499 0.507 0.491        5000    50
## # ... with 90 more rows
```

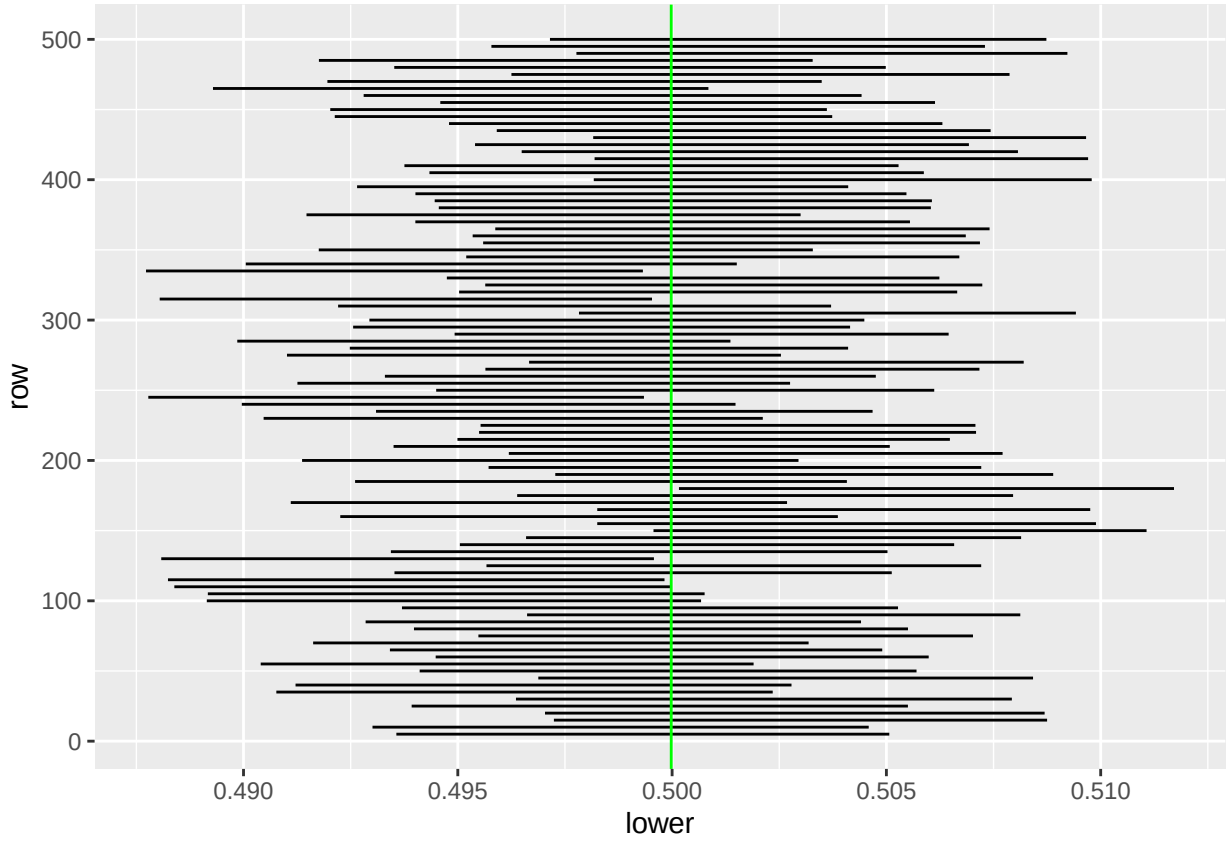
```
uni_visual_5000 <- int_visual(dgp_7,5000)
uni_visual_5000
```




```
# n = 10000
uni_table_10000 <- interval(dgp_7,10000)
uni_table_10000
```

```
## # A tibble: 100 x 5
##       mu upper lower sample_size   row
##    <dbl> <dbl> <dbl>       <dbl> <dbl>
##  1 0.502 0.508 0.497       10000     5
##  2 0.501 0.507 0.495       10000    10
##  3 0.500 0.505 0.494       10000    15
##  4 0.498 0.504 0.492       10000    20
##  5 0.500 0.506 0.494       10000    25
##  6 0.498 0.504 0.492       10000    30
##  7 0.503 0.509 0.498       10000    35
##  8 0.504 0.510 0.498       10000    40
##  9 0.500 0.506 0.495       10000    45
## 10 0.501 0.507 0.495       10000    50
## # ... with 90 more rows
```

```
uni_visual_10000 <- int_visual(dgp_7,10000)
uni_visual_10000
```



4.

```
data = read_xpt("CDQ_H.XPT") # replace w/ appropriate file path
prev_chest <- mean(data$CDQ001)
prev<- data %>%
  count(CDQ001) %>%
  mutate(prev = n/sum(n))
prev_chest <- pull(prev[1,3])
prev_chest
```

```
## [1] 0.2304063
```

5.

- You can try and calculate a 95% confidence interval to see if .3 is contained in the interval. If it is not then you can try to use that to say that the 30% measured is probably not the true value. A caveat to this argument is that by random chance (5% with 95% CI) our sample can generate an interval that does not contain the true value which in that case can in fact be 30%.