Sampling Property Tests

```r
library(TreeSampleR)
library(igraph)
library(MASS)
library(tidyverse)
```

```r
## Generate a graph and arbitrarily select two edges to illustrate the testing procedure

num_vert = 10
A_1<- matrix(rep(1, num_vert*num_vert), num_vert, num_vert)
A_2<- matrix(rep(1, num_vert*num_vert), num_vert, num_vert)

A <- matrix(0, 2*num_vert, 2*num_vert)
A[1:num_vert, 1:num_vert] <- A_1
A[(num_vert + 1):(2*num_vert), (num_vert + 1):(2*num_vert)] <- A_2
diag(A) <- 0


## select two edges
edge1 = c(num_vert, num_vert + 1)
edge2 = c(num_vert %/% 2, num_vert + num_vert %/% 2)

A[edge1[1], edge1[2]] <- 1
A[edge1[2], edge1[1]] <- 1

A[edge2[1], edge2[2]] <- 1
A[edge2[2], edge2[1]] <- 1
```

```r
## Sampling 1000 spanning trees over 1000 batches on the above adjacency matrix using all 3
set.seed(100)

N = 1000
```

```r
reps = 1000

accel.edge1.list = base::rep(0, reps)
accel.edge2.list = base::rep(0, reps)

normal.edge1.list = base::rep(0, reps)
normal.edge2.list = base::rep(0, reps)

wilson.edge1.list = base::rep(0, reps)
wilson.edge2.list = base::rep(0, reps)

for (j in 1:reps) {
  if ((j %% 50) == 0) {
    print("progress at ")
    print(j/reps)
  }
  accel.edge1 = 0
  accel.edge2 = 0

  normal.edge1 = 0
  normal.edge2 = 0

  wilson.edge1 = 0
  wilson.edge2 = 0

  degrees = colSums(A)[1:10]
  normalize = sum(degrees)
  prob = degrees/normalize

  ## accel loop
  for (i in 1:N){
    init = 0
    a1 <- fast_cover(A, init, num_vert*2)
    if (a1[edge1[1], edge1[2]] == 1){
      accel.edge1 = accel.edge1 + 1
    }
    if (a1[edge2[1], edge2[2]] == 1){
      accel.edge2 = accel.edge2 + 1
    }
  }

  ## normal loop
  for (i in 1:N){
    init = 1 # which(rmultinom(1, 1, prob) > 0)
    a2<- aldous_broder(A,  init)
    if (a2[edge1[1], edge1[2]] == 1){
      normal.edge1 = normal.edge1 + 1
    }
    if (a2[edge2[1], edge2[2]] == 1){
      normal.edge2 = normal.edge2 + 1
    }
  }

   ## wilson loop
  for (i in 1:N){
    init = 1 # which(rmultinom(1, 1, prob) > 0)
    a3<- wilson(A,  init)
    if (a3[edge1[1], edge1[2]] == 1){
       wilson.edge1 = wilson.edge1 + 1
    }
    if (a3[edge2[1], edge2[2]] == 1){
      wilson.edge2 = wilson.edge2 + 1
    }
  }

  accel.edge1.list[j] = accel.edge1
  accel.edge2.list[j] = accel.edge2

  normal.edge1.list[j] = normal.edge1
  normal.edge2.list[j] = normal.edge2

  wilson.edge1.list[j] = wilson.edge1
  wilson.edge2.list[j] = wilson.edge2
}
```

```
[1] "progress at "
[1] 0.05
[1] "progress at "
[1] 0.1
[1] "progress at "
[1] 0.15
[1] "progress at "
[1] 0.2
[1] "progress at "
```

```
[1] 0.25
[1] "progress at "
[1] 0.3
[1] "progress at "
[1] 0.35
[1] "progress at "
[1] 0.4
[1] "progress at "
[1] 0.45
[1] "progress at "
[1] 0.5
[1] "progress at "
[1] 0.55
[1] "progress at "
[1] 0.6
[1] "progress at "
[1] 0.65
[1] "progress at "
[1] 0.7
[1] "progress at "
[1] 0.75
[1] "progress at "
[1] 0.8
[1] "progress at "
[1] 0.85
[1] "progress at "
[1] 0.9
[1] "progress at "
[1] 0.95
[1] "progress at "
[1] 1
```

```r
save(accel.edge1.list , accel.edge2.list, file = "saved_tree_samples/accel_edge_inclusion_sam

save(normal.edge1.list , normal.edge2.list,  file = "saved_tree_samples/normal_edge_inclusion

save(wilson.edge1.list , wilson.edge2.list,  file = "saved_tree_samples/wilson_edge_inclusion
```

Sources:

e.g. effective resistance formula https://cs-people.bu.edu/orecchia/CS591fa16/lecture7.pdf
e.g. Marginal inclusion probability formula: e.g. http://cs-www.cs.yale.edu/homes/spielman/sagt/sagt.pdf
Chapter 13.5

```r
## define function that calculates marginal edge inclusion probability

D = diag(colSums(A))
L = D - A
L_inv = ginv(L)

get_marginal_probability <- function(u, v, L_inv, A){
  w = A[u, v]
  n = dim(A)[1]
  chi_uv = base::rep(0, n)
  chi_uv[u] = 1
  chi_uv[v] = -1
  effective_resistance = t(chi_uv) %*% L_inv %*% chi_uv
  marginal_probability = effective_resistance * w
  marginal_probability
}
```

```r
## calculate the exact marginal edge inclusion probability of the two edges
edge1 = c(num_vert, num_vert + 1)
edge2 = c(num_vert %/% 2, num_vert + num_vert %/% 2)
edge_1_exact_p = get_marginal_probability(edge1[1], edge1[2], L_inv, A)
edge_2_exact_p = get_marginal_probability(edge2[1], edge2[2], L_inv, A)
print(edge_1_exact_p)
```

```
          [,1]
[1,] 0.5833333
```

```r
print(edge_2_exact_p)
```

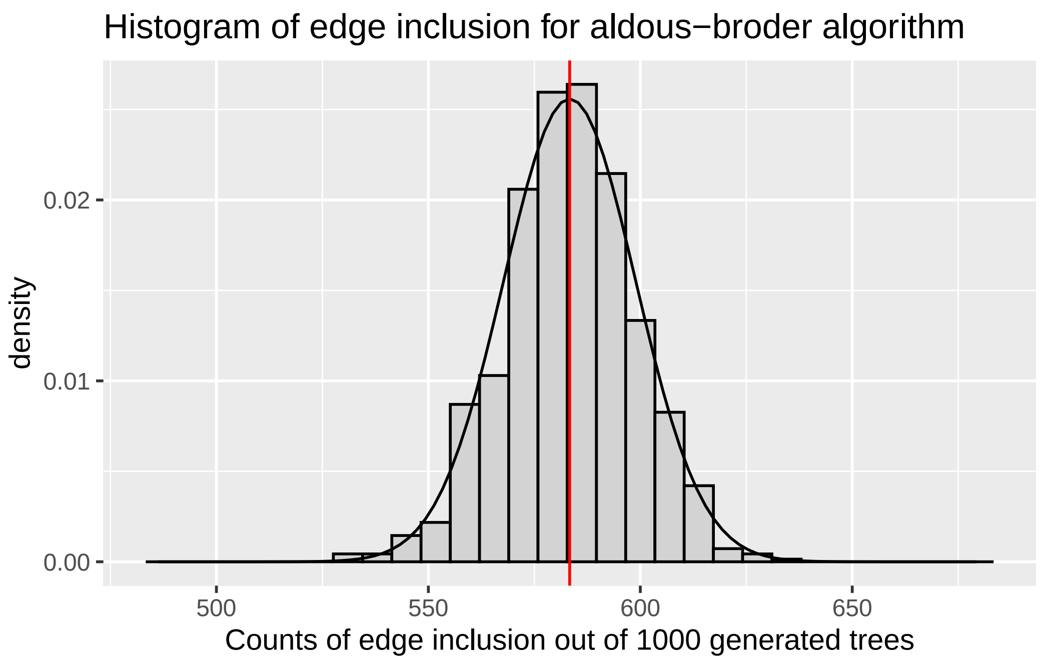
```
          [,1]
[1,] 0.5833333
```

```r
## plot the empirical edge inclusion frequencies for edge 1 and compare with theoretical val
p.1 = get_marginal_probability(edge1[1], edge1[2], L_inv, A)

data.normal <- data.frame(x= normal.edge1.list)

#create histogram and overlay normal curve
ggplot(data.normal, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.1, sd= sqrt(N*p.1*(1-p.1)))) +
  xlim(N*p.1 -N/10 , N*p.1 +N/10 ) +
  geom_vline(xintercept=N*p.1, color = "red") +
  ggtitle("Histogram of edge inclusion for aldous-broder algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

```
Warning: The dot-dot notation (`..density..`) was deprecated in ggplot2 3.4.0.
i Please use `after_stat(density)` instead.
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```

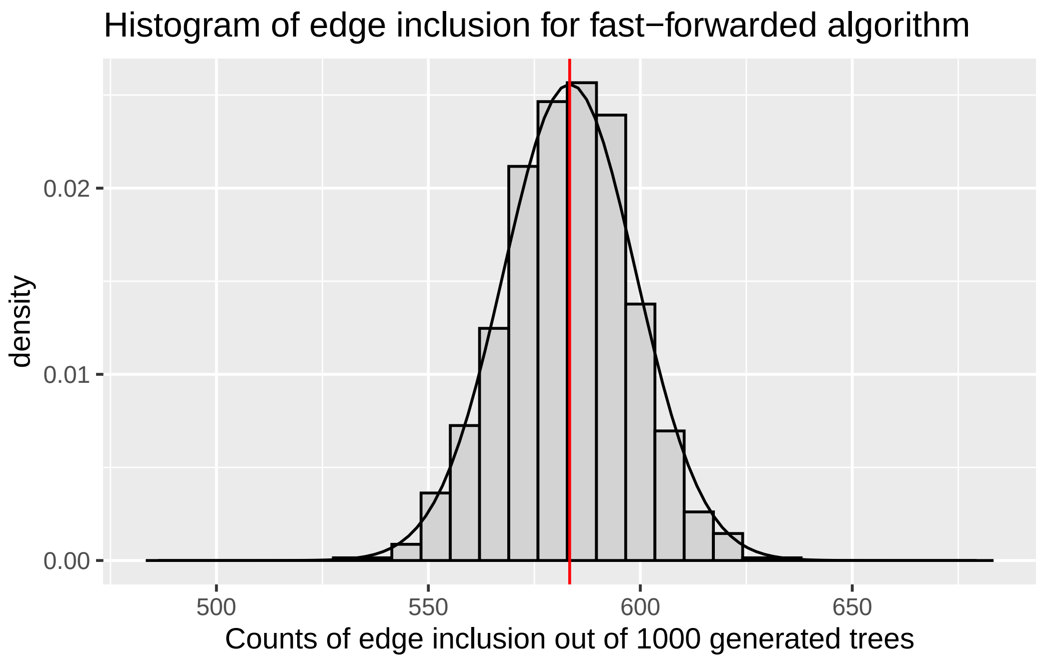


```
#overlay fast forward curve on histogram
data.accel <- data.frame(x= accel.edge1.list)

#create histogram and overlay fast forward curve
ggplot(data.accel, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.1, sd= sqrt(N*p.1*(1-p.1)))) +
```

```r
  xlim(N*p.1 -N/10 , N*p.1 +N/10 ) +
  geom_vline(xintercept=N*p.1, color = "red") +
  ggtitle("Histogram of edge inclusion for fast-forwarded algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

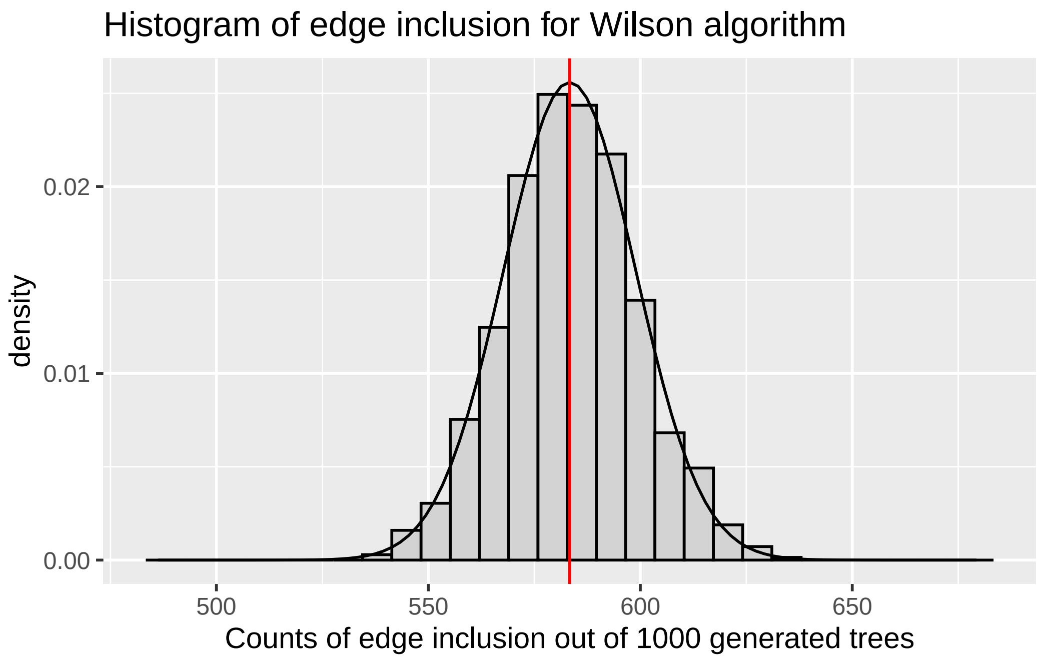
```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



```r
data.wilson <- data.frame(x= wilson.edge1.list)
#create histogram and overlay Wilson curve
ggplot(data.wilson, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.1, sd= sqrt(N*p.1*(1-p.1)))) +
  xlim(N*p.1 -N/10 , N*p.1 +N/10 ) +
  geom_vline(xintercept=N*p.1, color = "red") +
  ggtitle("Histogram of edge inclusion for Wilson algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

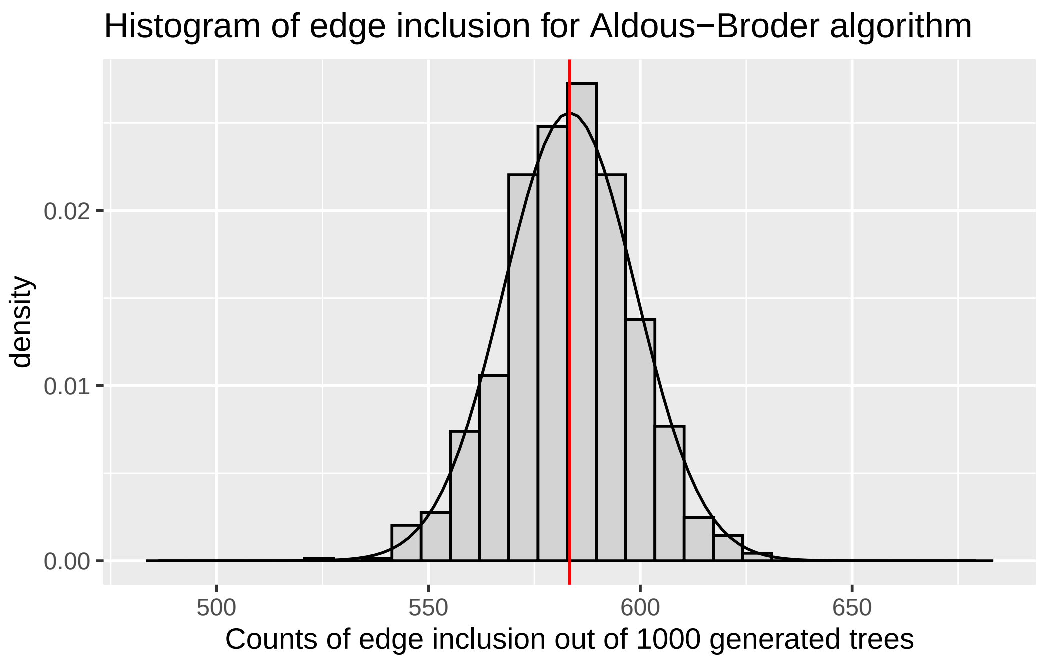
```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



```
## plot the empirical edge inclusion frequencies for edge 2 and compare with theoretical val
p.2 = get_marginal_probability(edge2[1], edge2[2], L_inv, A)

data.normal <- data.frame(x= normal.edge2.list)

#create histogram and overlay normal curve
ggplot(data.normal, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.2, sd= sqrt(N*p.2*(1-p.2)))) +
  xlim(N*p.1 -N/10 , N*p.1 +N/10) +
  geom_vline(xintercept=1000*p.2, color = "red") +
  ggtitle("Histogram of edge inclusion for Aldous-Broder algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

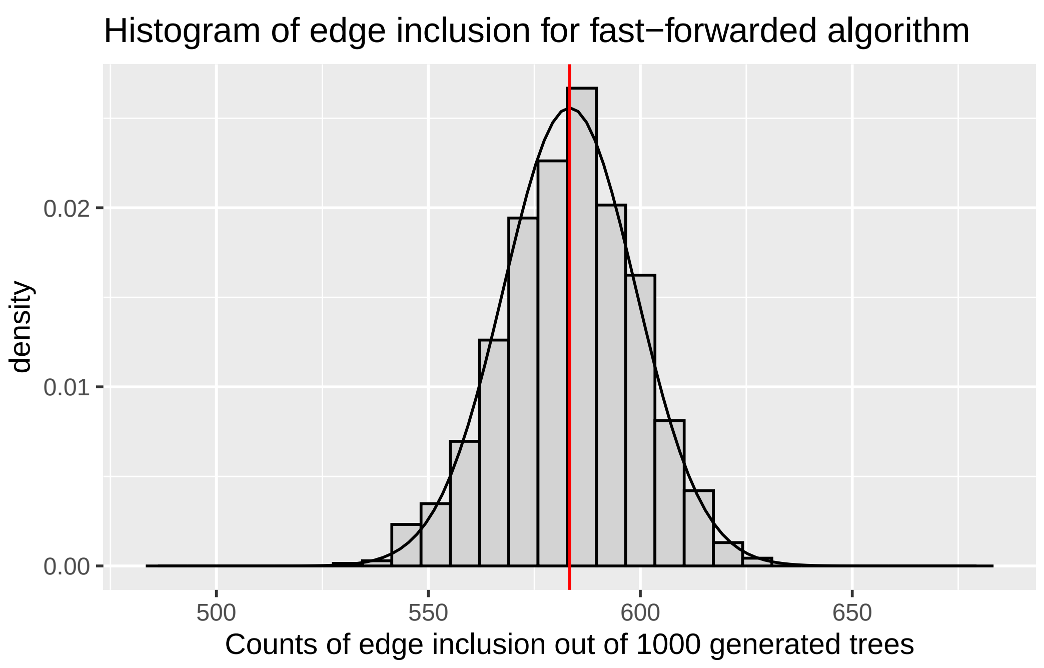
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



```r
#overlay accel curve on histogram
data.accel <- data.frame(x= accel.edge2.list)

#create histogram and overlay fast forward curve
ggplot(data.accel, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.2, sd= sqrt(N*p.2*(1-p.2)))) +
  xlim(N*p.1 -N/10 , N*p.1 +N/10) +
  geom_vline(xintercept=1000*p.2, color = "red") +
  ggtitle("Histogram of edge inclusion for fast-forwarded algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

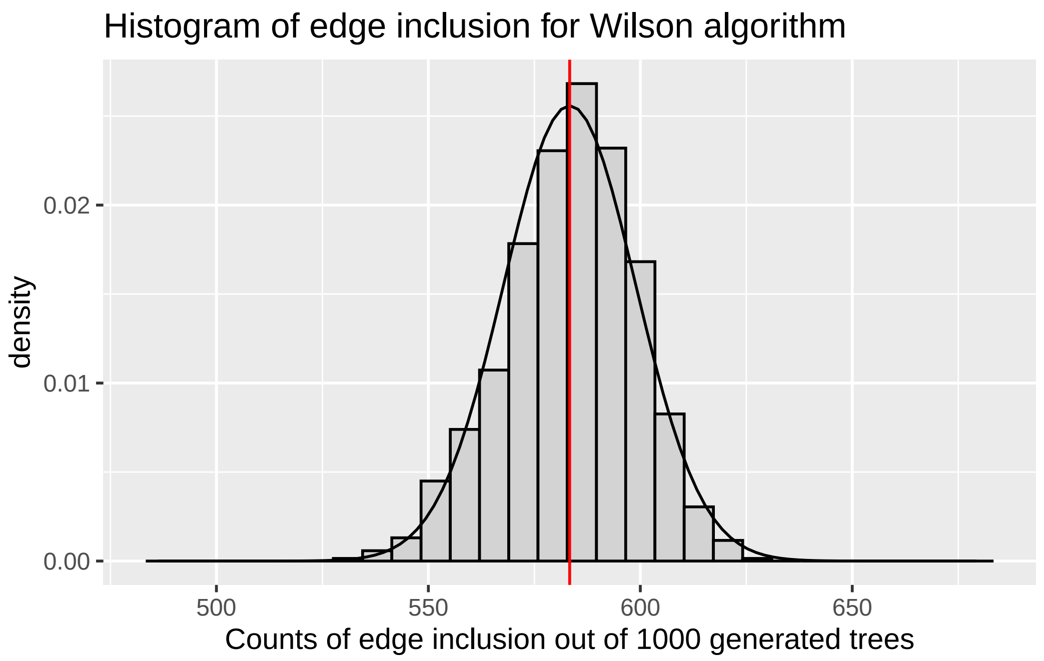
```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



```
data.wilson <- data.frame(x= wilson.edge2.list)
#create histogram and overlay Wilson curve
ggplot(data.wilson, aes(x)) +
  geom_histogram(aes(y = ..density..), fill='lightgray', col='black') +
  stat_function(fun = dnorm, args = list(mean= N*p.2, sd= sqrt(N*p.2*(1-p.2)))) +
  xlim(N*p.2 -N/10 , N*p.2 +N/10 ) +
  geom_vline(xintercept=N*p.2, color = "red") +
  ggtitle("Histogram of edge inclusion for Wilson algorithm") +
  xlab("Counts of edge inclusion out of 1000 generated trees")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
Warning: Removed 2 rows containing missing values or values outside the scale range
(`geom_bar()`).
```



All 3 algorithms offers excellent match with the theoretical value. Moreover, the distribution of the empirical frequency estimates matches up with the normal curve (with mean and variances computed from theoretical value under the central limit theorem) well. Provides useful evidence that algorithms are implemented correctly.