ALD/GIG

Ayush Agarwal

24/07/2020

ALD generation function

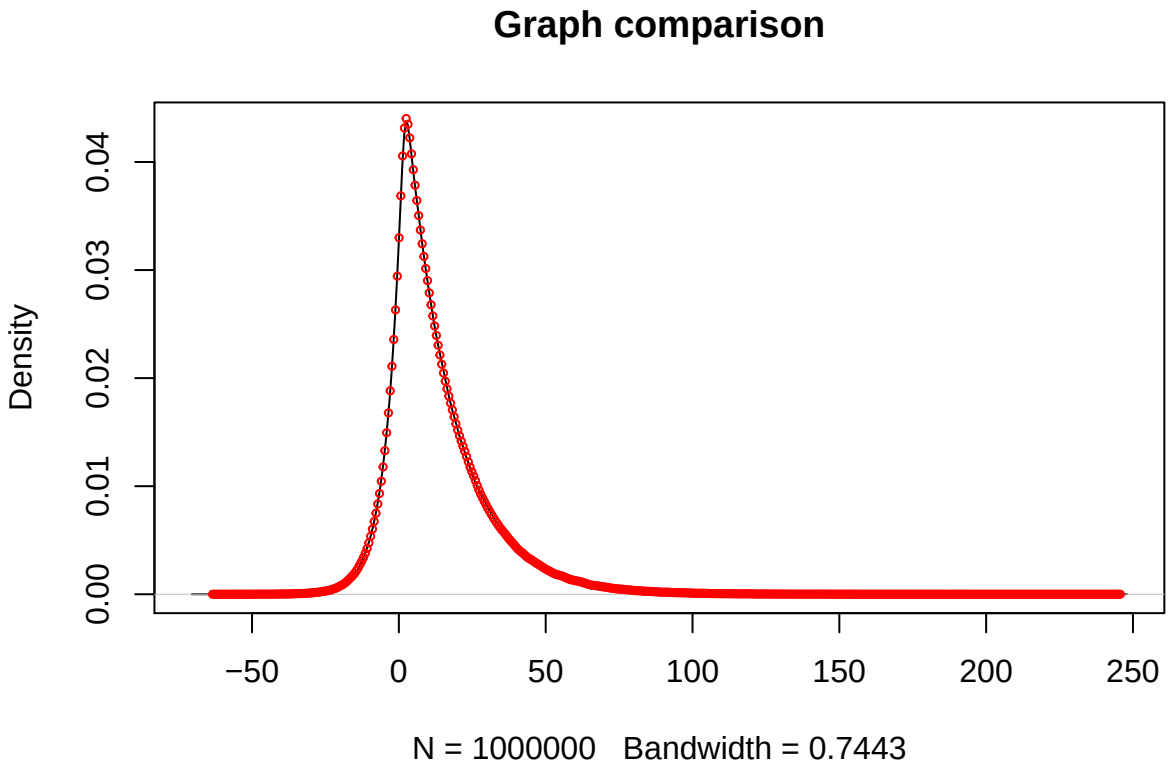
My implementation

This implementation corresponds to Equation (2) of README. Existing sampler in `ald` package.

Tests and Comparisons

Case1

```
n <- 1e6
mu <- 2
sigma <- 4

#check for p < 1/2
p <- 0.25
mean_exp = mu + (sigma*(1-2*p))/(p*(1-p))
var_exp = (sigma^2)*(1-2*p+2*p^2)/((p*(1-p))^2)

x <- rald_mix(n,mu,sigma,p)
y <- rALD(n,mu,sigma,p)

stopifnot(all.equal(mean_exp, mean(x), tolerance=0.01)) #mean is close to expected value
stopifnot(all.equal(var_exp, var(x)[1,1], tolerance=0.1)) #variance is close to close to expected value

stopifnot(all.equal(mean(x),mean(y),tolerance=0.01))  #both samplers agree
stopifnot(all.equal(var(y), var(x)[1,1], tolerance=0.1))

plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```



Speed

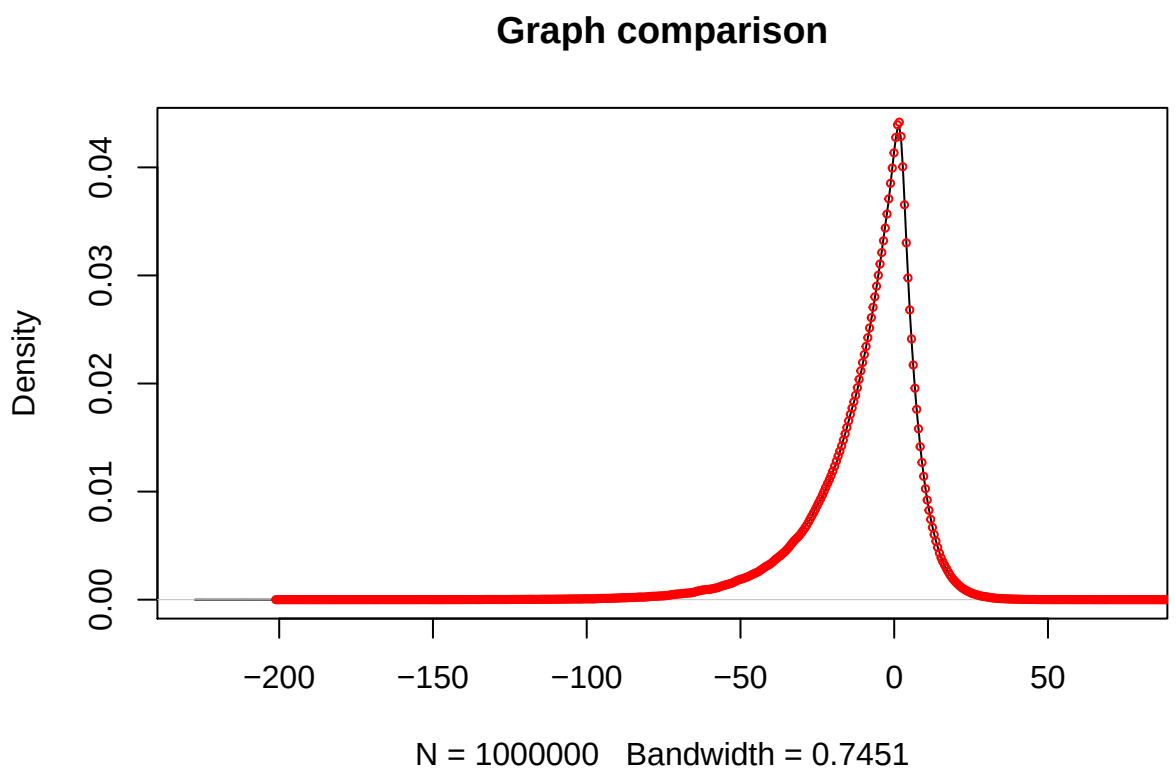
```
### speed, runs a 100x faster
mbm <- microbenchmark(my=rald_mix(1e5,mu,sigma,p),old=y <- rALD(1e5,mu,sigma,p),times=1)
print(mbm)
```

```
## Unit: milliseconds
##  expr        min         lq       mean     median         uq        max
##    my   29.92868   29.92868   29.92868   29.92868   29.92868   29.92868
##   old 1643.44445 1643.44445 1643.44445 1643.44445 1643.44445 1643.44445
##  neval
##      1
##      1
```

Case2

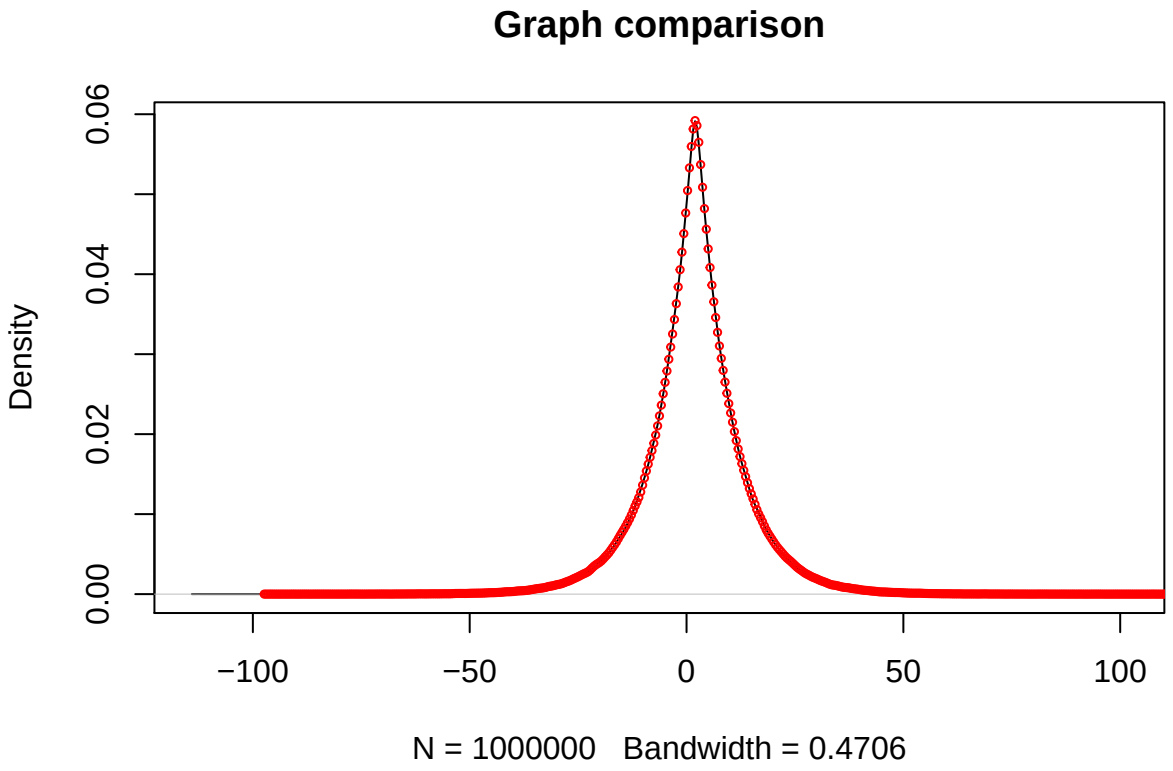
```
#check for p > 1/2
p <- 0.75
mean_exp = mu + (sigma*(1-2*p))/(p*(1-p))
var_exp = (sigma^2)*(1-2*p+2*p^2)/((p*(1-p))^2)

x <- rald_mix(n,mu,sigma,p)
y <- rALD(n,mu,sigma,p)

stopifnot(all.equal(mean_exp, mean(x), tolerance=0.01)) #mean is close to expected value
```

```r
stopifnot(all.equal(var_exp, var(x)[1,1], tolerance=0.1))  #variance is close to close to expected valu

stopifnot(all.equal(mean(x),mean(y),tolerance=0.01)) #both samplers agree
stopifnot(all.equal(var(y), var(x)[1,1], tolerance=0.1))

plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```



Case3

```r
#check for p = 1/2, equ to double laplace
p <- 0.50
mean_exp = mu + (sigma*(1-2*p))/(p*(1-p))
var_exp = (sigma^2)*(1-2*p+2*p^2)/((p*(1-p))^2)

x <- rald_mix(n,mu,sigma,p)
y <- rALD(n,mu,sigma,p)

stopifnot(all.equal(mean_exp, mean(x), tolerance=0.01)) #mean is close to expected value
stopifnot(all.equal(var_exp, var(x)[1,1], tolerance=0.1))  #variance is close to close to expected valu

stopifnot(all.equal(mean(x),mean(y),tolerance=0.01)) #both samplers agree
stopifnot(all.equal(var(y), var(x)[1,1], tolerance=0.1))

plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```



GIG generation function

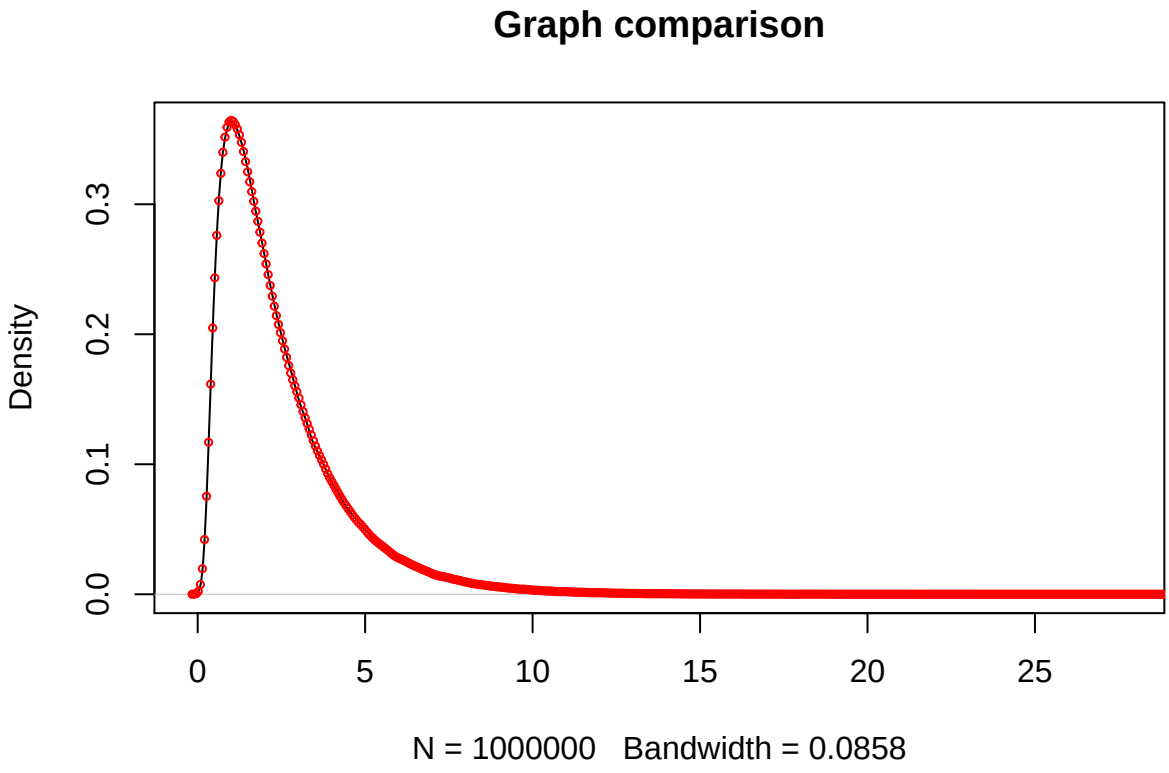
My implementation

This is my implementation of the GIG sampler, corresponding to Equation (7) in README. Existing implementation in `GIGrvg`

Tests and Comparisons

Case1

```
n <- 1e6
lambda = 0.5
a = 1
b = 2

x <- rgig_my(n,lambda,a,b)
y <- rgig(n,lambda,chi=b,psi=a)

stopifnot(all.equal(mean(x), mean(y), tolerance=0.01)) #mean is close to expected value
stopifnot(all.equal(var(x)[1,1], var(y), tolerance=0.1)) #variance is close to close to expected value

##plots
plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```



Speed Speed seems a bit slow on average but that is just due to conversions when returning back to R, in my use it'll be an internal function and hence the issue is to be neglected.

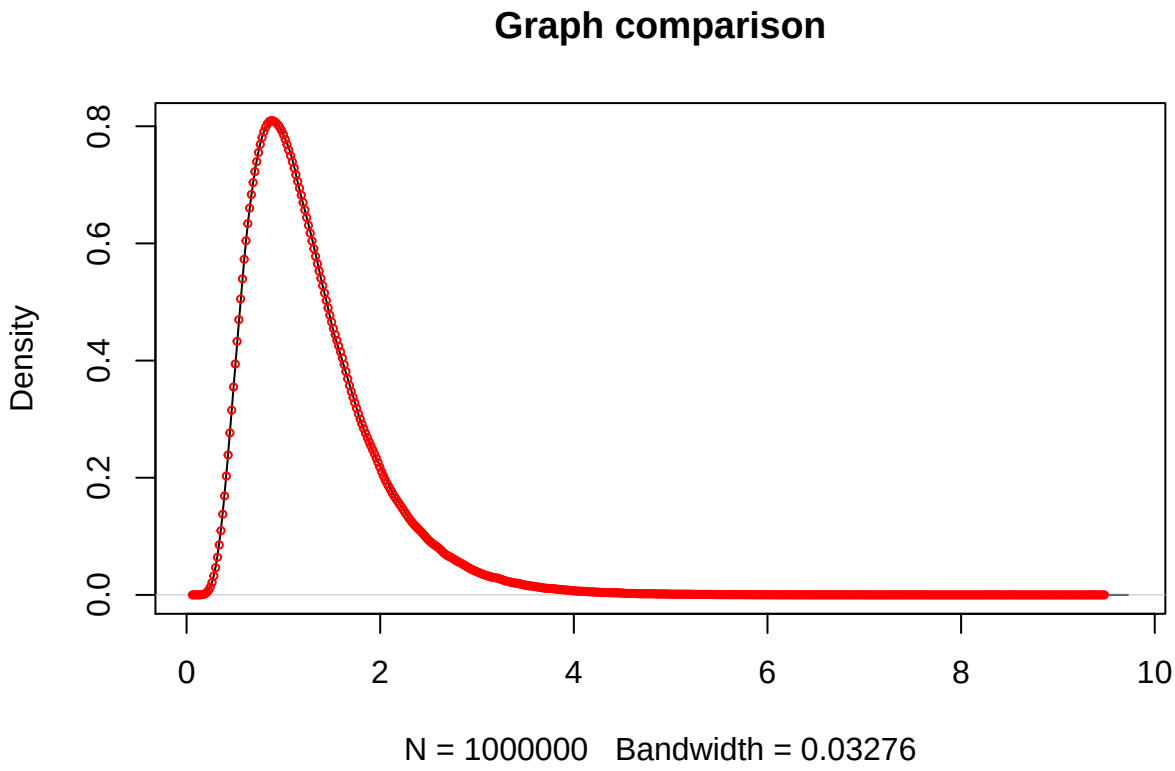
```
## speed comparison
mbm <- microbenchmark(my=rgig_my(n,lambda,a,b),his=rgig(n,lambda,chi=b,psi=a),times=5)
print(mbm)
```

```
## Unit: milliseconds
##  expr      min       lq     mean   median       uq      max neval
##    my 82.75027 86.35895 86.58966 86.54066 86.73875 90.55966     5
##   his 58.32522 61.12303 62.17425 61.36301 61.51641 68.54359     5
```

Case2

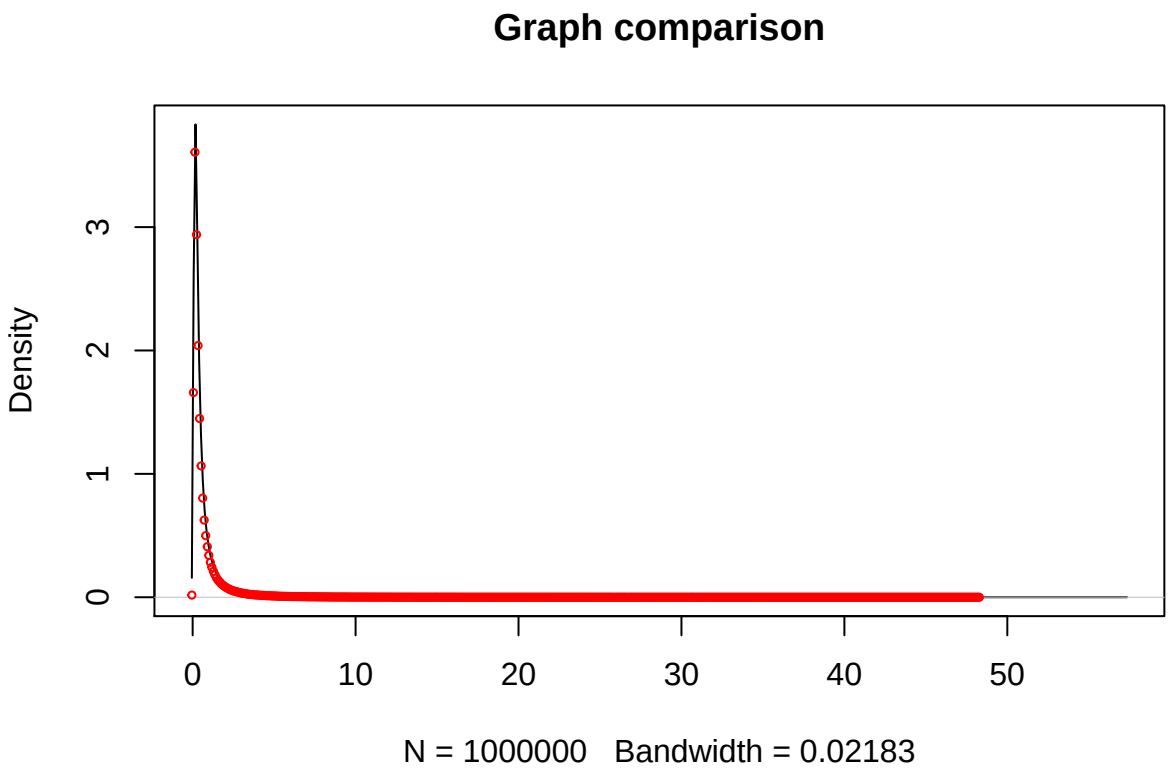
```
lambda = -0.50
a = 3
b = 5

x <- rgig_my(n,lambda,a,b)
y <- rgig(n,lambda,chi=b,psi=a)

stopifnot(all.equal(mean(x), mean(y), tolerance=0.01)) #mean is close to expected value
stopifnot(all.equal(var(x)[1,1], var(y), tolerance=0.1)) #variance is close to close to expected value

##plots
```

```r
plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```



Case3

```r
lambda = -1
a = 0.25
b = 0.50

x <- rgig_my(n,lambda,a,b)
y <- rgig(n,lambda,chi=b,psi=a)

stopifnot(all.equal(mean(x), mean(y), tolerance=0.01)) #mean is close to expected value
stopifnot(all.equal(var(x)[1,1], var(y), tolerance=0.1)) #variance is close to close to expected value

##plots
plot(density(x),col="black",main="Graph comparison")
lines(density(y),col="red",type="p",cex=0.5)
```


Graph comparison
Density
0
1
2
3
0
10
20
30
40
50
N = 1000000 Bandwidth = 0.02183