Draw-Experiment

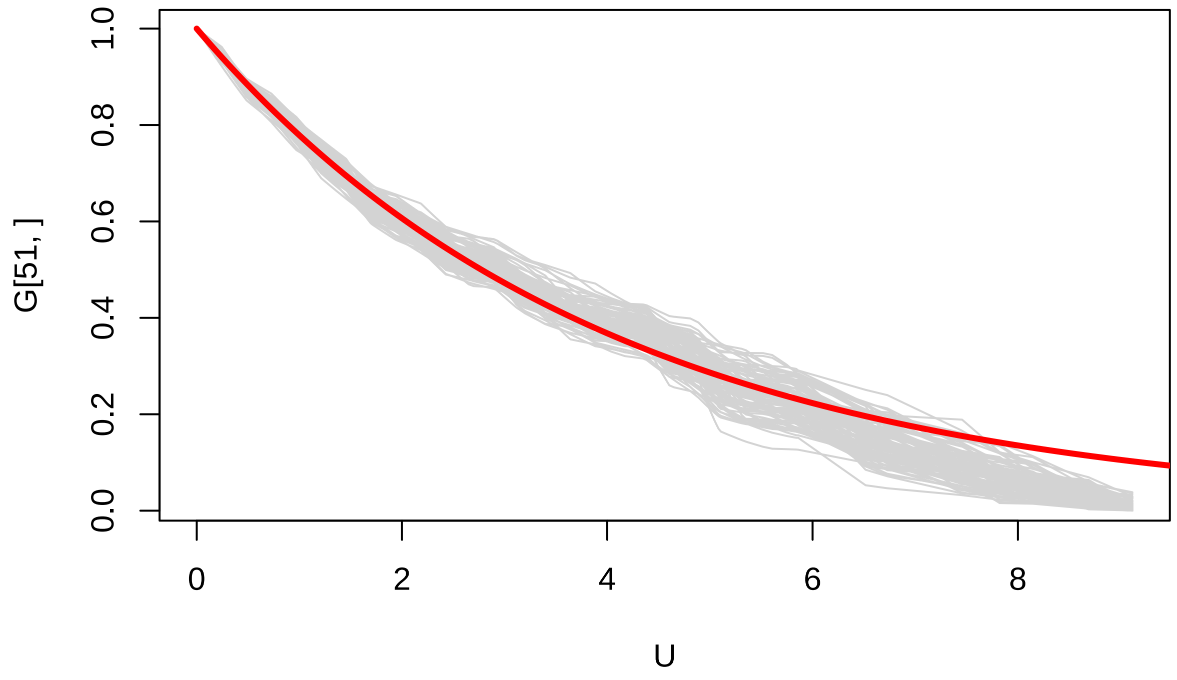
Hengde Ouyang

2023-08-20

Previous Code

```r
DrawLambdas <- function(nevents, nrisks, kappa0, delta_alpha) {
  nbins <- length(nevents)
  lambda_draw <- rep(NA, nbins)
  eta <- kappa0*delta_alpha
  bbeta <- nrisks - nevents + kappa0
  for(k in 1:nbins) {

    if(eta==1) {
      rax <- rbeta(1, shape1=bbeta[k], shape2=nevents[k] + 1)
      lambda_draw[k] <- -log(rax)
    } else {
      done <- FALSE
      while(!done) {
        theta <- rgamma(1, shape=eta + nevents[k], rate=bbeta[k])
        u <- runif(1)
        thresh <- log1p(-exp(-theta)) - log(theta)
        #print(c(thresh, nevents[k]*thresh, log(u)))
        # This approach doesn't work too well if
        #  nevents[k] is large
        if(log(u) < nevents[k]*thresh) {
          lambda_draw[k] <- theta
          done <- TRUE
        }
      }
    }
  }
  return(lambda_draw)
}



DrawCensSurv <- function(ndraw, U, delta, sgrid, kappa0, delta_alpha) {
  J <- length(sgrid)
  E <- R <- rep(NA, J-1)
  for(j in 2:J) {
    E[j-1] <- sum((1 - delta)*(U > sgrid[j-1])*(U <= sgrid[j]))
    R[j-1] <- sum(U > sgrid[j-1])
  }
  weight_ans <- matrix(NA, nrow=ndraw, ncol=length(U))
  for(k in 1:ndraw) {
      lambda.draw <- DrawLambdas(nevents=E, nrisks=R, kappa0=kappa0, delta_alpha=delta_alpha)
      CumHazFn <- approxfun(sgrid, cumsum(c(0, lambda.draw)))
      weight_ans[k,] <- exp(-CumHazFn(U))
  }
  return(weight_ans)
}
```

```r
Ctime <- rexp(1000, rate=0.25)
Ttime <- rexp(1000, rate=0.5)
U <- pmin(Ttime, Ctime)
delta <- ifelse(Ttime < Ctime, 1, 0)
sgrid <- seq(0, max(Ctime), length.out=100)

G <- DrawCensSurv(100, U, delta, sgrid, kappa0=1, delta_alpha=1)

## Example with looking at G in the 51st MCMC iteration
tt <- seq(0, 10, length.out=1000)
true_cens <- 1 - pexp(tt, rate=0.25)
plot(U, G[51,], type="n")
for(j in 1:100) {
    oo <- order(G[j,])
    lines(U[oo], G[j,oo], col="lightgrey")
}
lines(tt, true_cens, lwd=3, col="red")
```



Everything looks fine. But if we simply change the true distribution of C to Uniform Distribution, the graph will change.

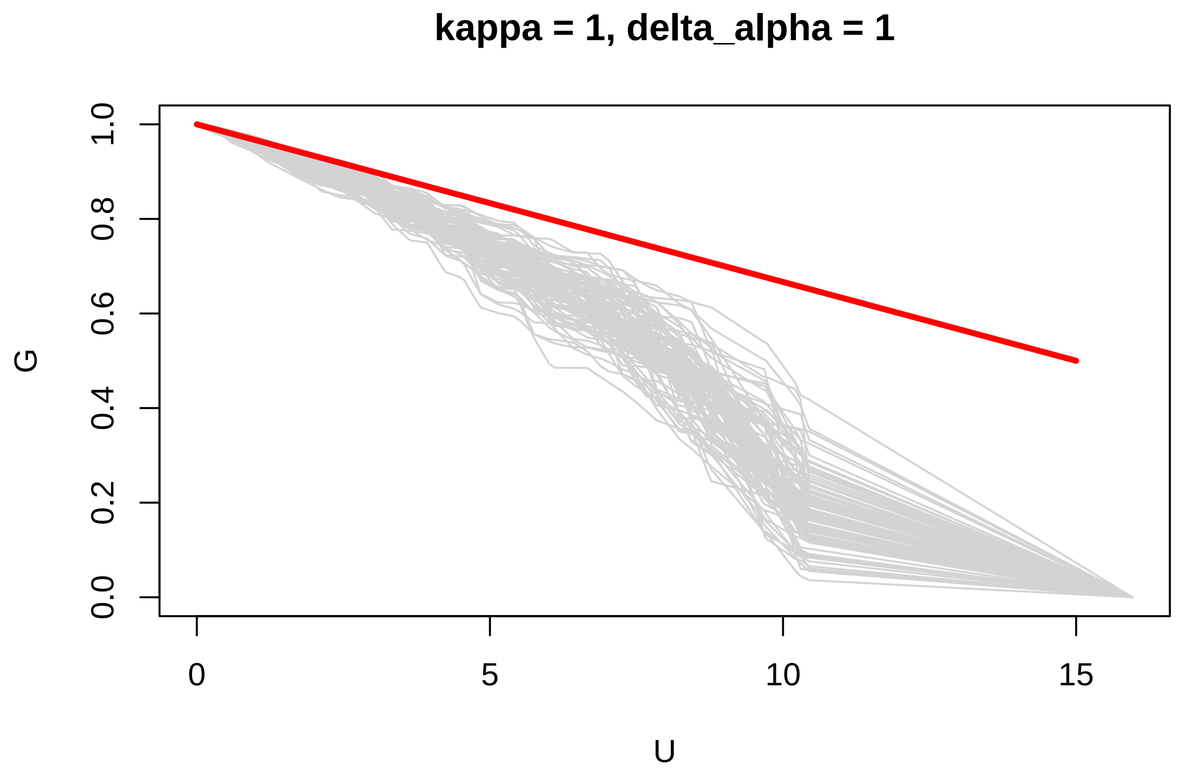
C follows Uniform Distribution

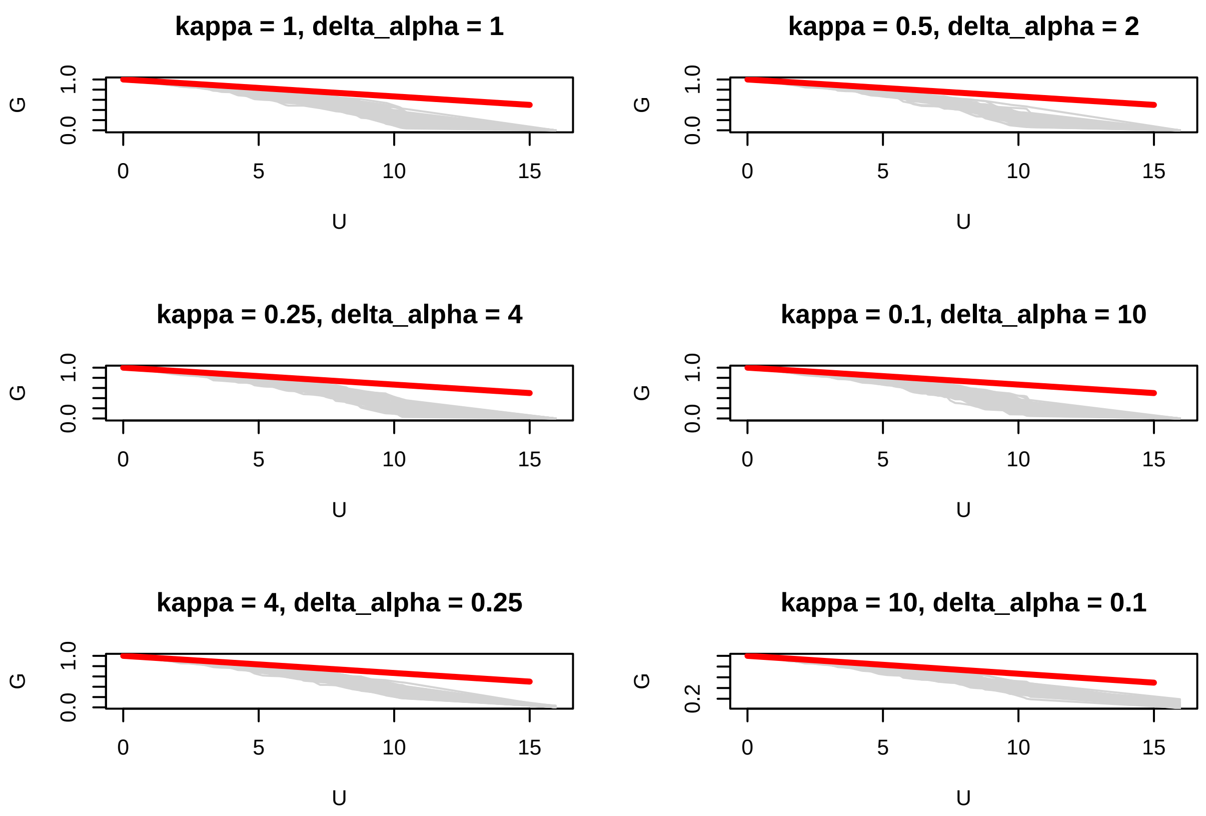
1. If we didn't change kappa and delta_alpha, we will observe the bias.

2. If we keep kappa*delta_alpha = 1, the bias are still remained.

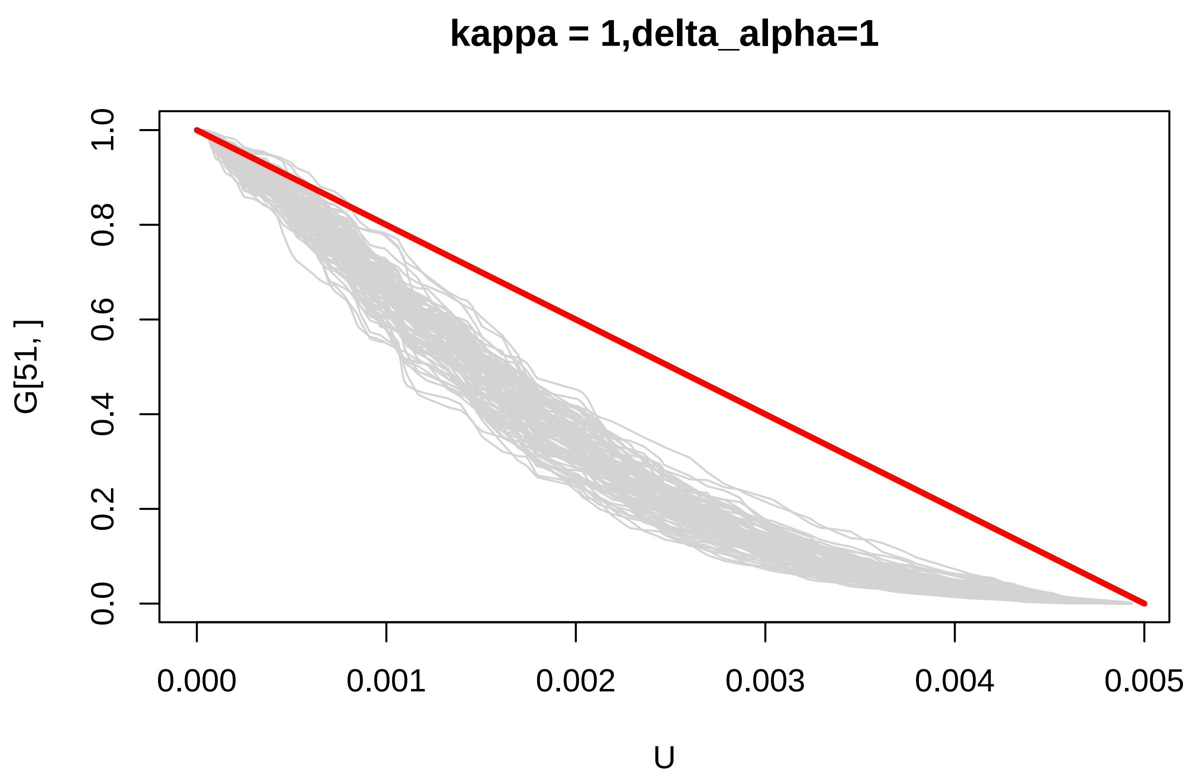
```r
Ctime <- runif(1000, 0,30)
Ttime <- rexp(1000, rate=0.5)
U <- pmin(Ttime, Ctime)
delta <- ifelse(Ttime < Ctime, 1, 0)
sgrid <- seq(0, max(Ctime), length.out=100)

G <- DrawCensSurv(100, U, delta, sgrid, kappa0=1, delta_alpha=1)

## Example with looking at G in the 51st MCMC iteration
tt <- seq(0, 15, length.out=1000)
true_cens <- 1 - punif(tt, 0,30)
```

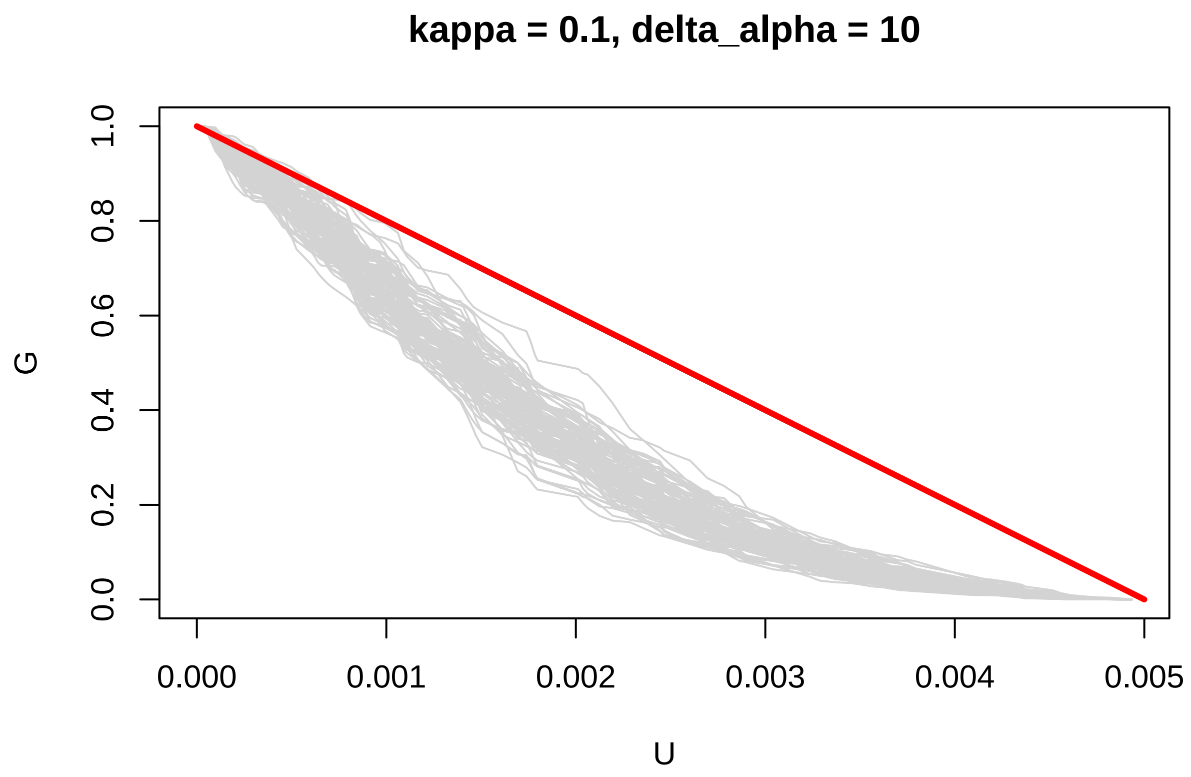
```r
plot(U, G[51,], type="n",main = "kappa = 1, delta_alpha = 1",ylab = "G")
for(j in 1:100) {
    oo <- order(G[j,])
    lines(U[oo], G[j,oo], col="lightgrey")
}
lines(tt, true_cens, lwd=3, col="red")
```



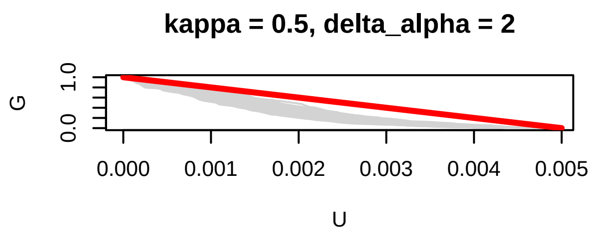

kappa = 1, delta_alpha = 1
kappa = 0.5, delta_alpha = 2
kappa = 0.25, delta_alpha = 4
kappa = 0.1, delta_alpha = 10
kappa = 4, delta_alpha = 0.25
kappa = 10, delta_alpha = 0.1
G
U
0.0
0.2
1.0
0
5
10
15

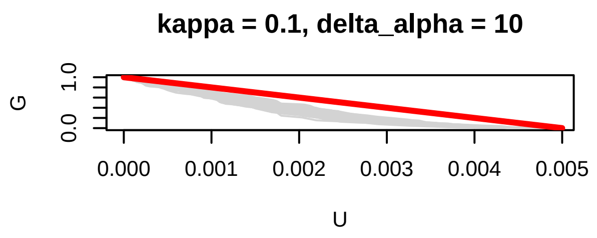
Using Cox Simulation Data

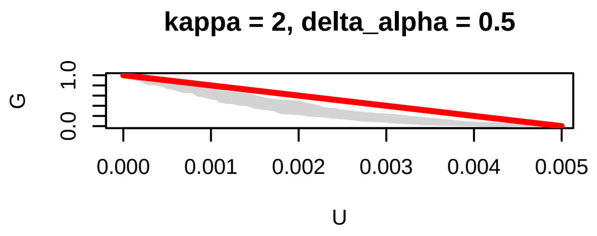


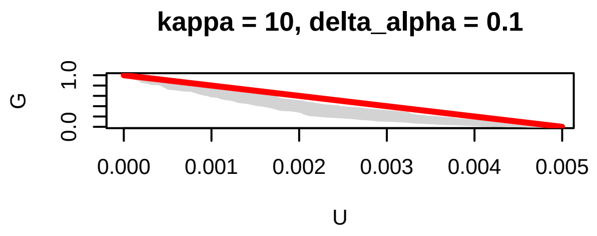


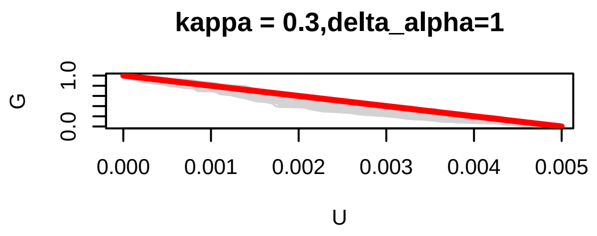
If kappa*delta_alpha does not have to be 1, we might have less bias

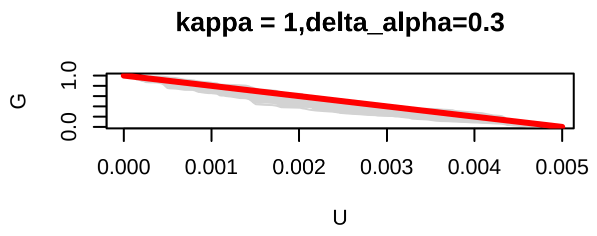


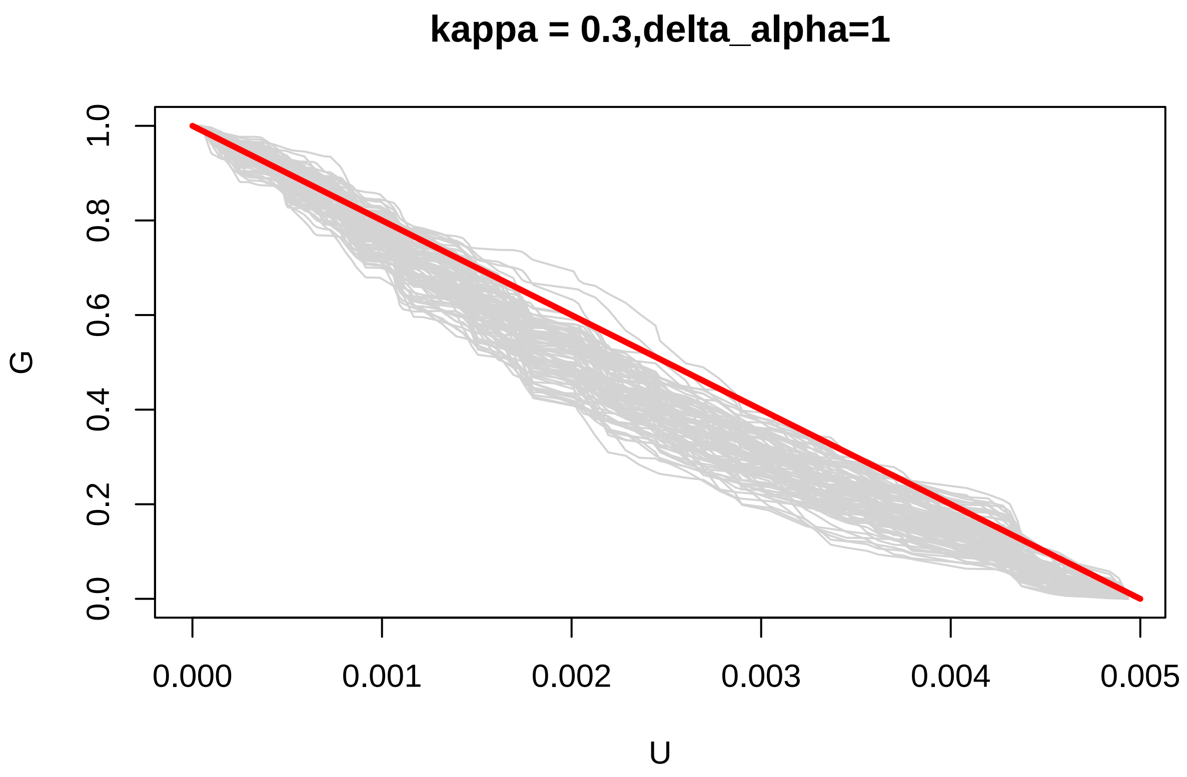


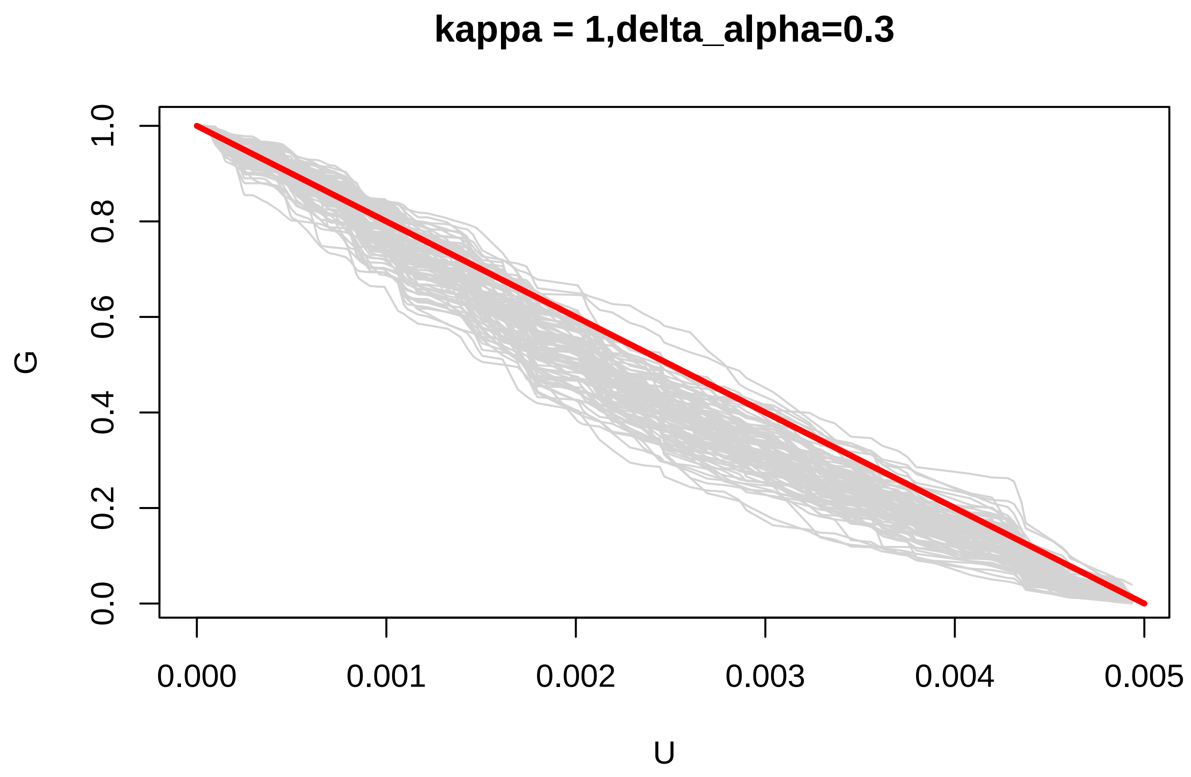









kappa = 0.3,delta_alpha=1
G
U
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.001
0.002
0.003
0.004
0.005


kappa = 1,delta_alpha=0.3
G
1.0
0.8
0.6
0.4
0.2
0.0
0.000
0.001
0.002
0.003
0.004
0.005
U