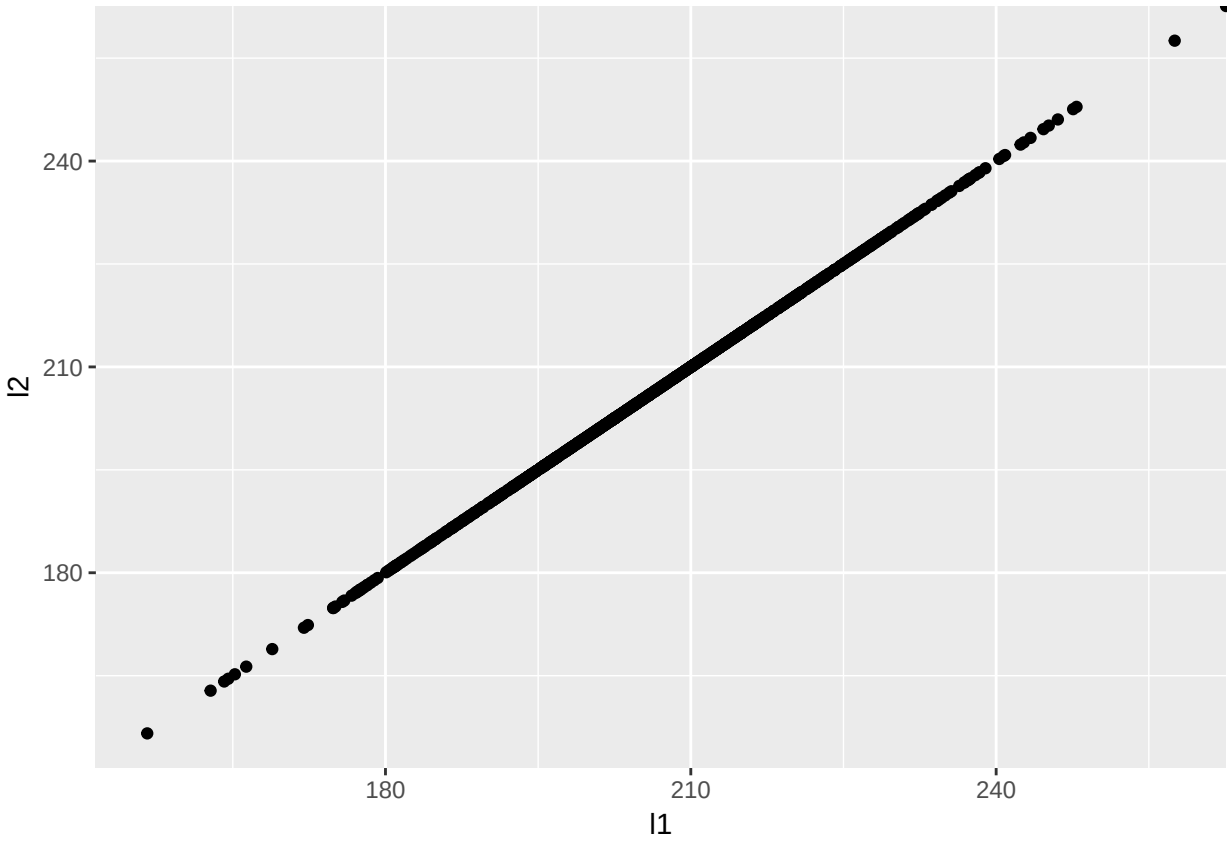
Tests

In order to search for possible bugs and test the package we have create alternative functions to the original ones which should be equivalent. There are a series of possible test to check that the package is doing the proper things to do

log-likelihood function

```
time = proc.time()
#dendroica
btdd = c(4.9999999998,4.806886544,4.70731246478,4.50735197578,4.37856240588,4.29594855558,4.19207515688
pars = c(3.2,0.3,40)
m = 1000
S = sim.sct(btdd,pars,m,print=FALSE)
l1 = vector(mode="numeric",length=m)
l2 = vector(mode="numeric",length=m)
for(i in 1:m){
  s = S$rec[[i]]
  l1[i] = nllik.tree(pars,s)
  l2[i] = nllik.tree2(pars,s)
}
qplot(l1,l2)
```



```
all.equal(l1,l2)
```

```
## [1] TRUE
```

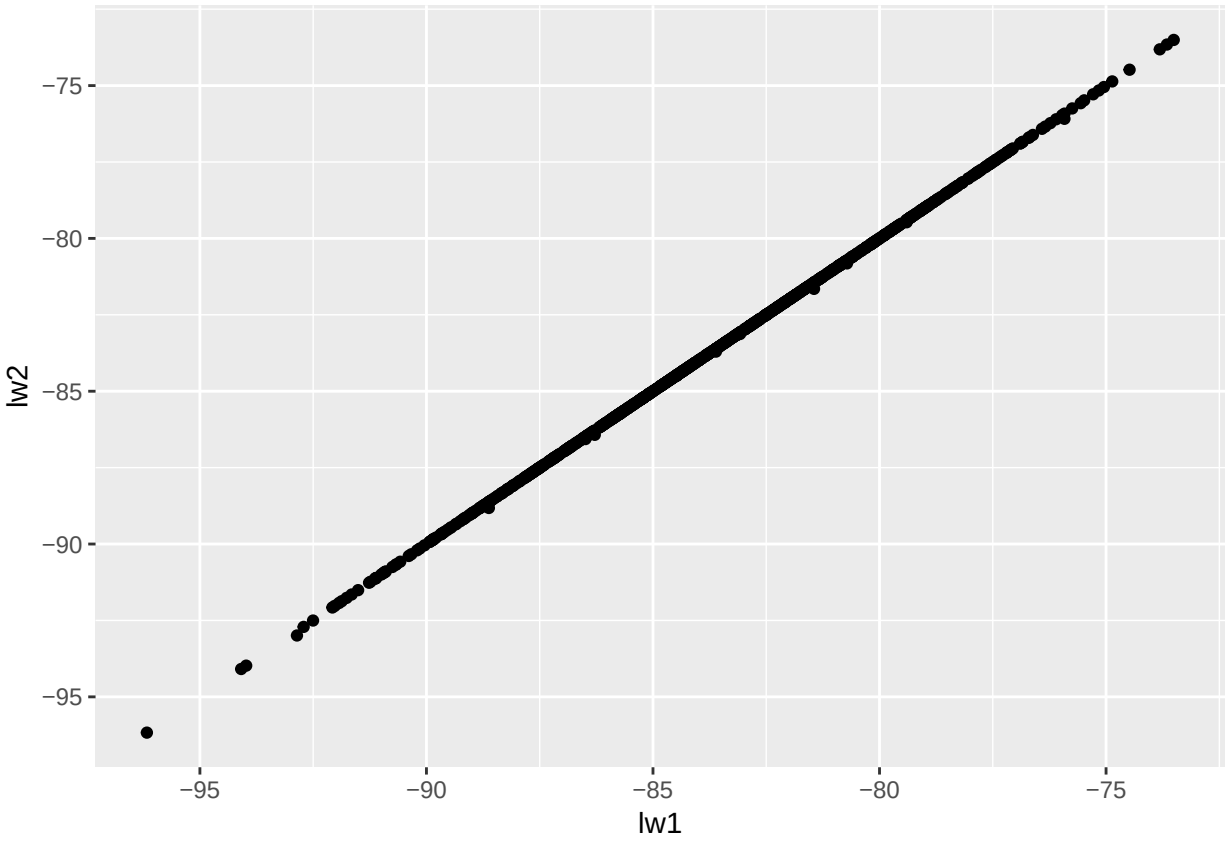
```
get.time(time)
```

```
## [1] 5.298
```

Importance sampling weights

To check if the importance sampling weights are well calculated we use the previous and the current method using the `sim.extinct_old` function.

```
lw1 = vector(mode="numeric",length=m)
lw2 = vector(mode="numeric",length=m)
w1 = vector(mode="numeric",length=m)
w2 = vector(mode="numeric",length=m)
for(i in 1:1000){
  si = sim.extinct_old(btdd,pars)
  w1[i] = si$weight
  w2[i] = si$weight2
  lw1[i] = si$logweight
  lw2[i] = si$logweight2
}
qplot(lw1,lw2)
```




```
all.equal(lw1,lw2)
```

```
## [1] "Mean relative difference: 2.491663e-05"
```

```
all.equal(w1,w2)
```

```
## [1] TRUE
```

now we check other ..

```
test.g <- function(df,pars){
  df$xi = 0
  df$xi[df$bte<df$bt[dim]] = 1
  dim = dim(df)[1]
  wtT = diff(c(0,df$bt))
  n.tree = list(wt=wtT,E=df$to[-dim])
  n.tree$E[n.tree$E==2] = 1
  E = n.tree$E
  n = c(2,2+cumsum(E)+cumsum(E-1))
  lambda = (pars[1]-(pars[1]-pars[2])*(n/pars[3]))
  mu = pars[2]
  s = lambda*n
  lsprob2 = da.prob2(wt=wtT,t_ext=df$t.ext,s=s,mu=pars[2],r=df$bt[dim]-c(0,df$bt[-dim]),n=n)
  ls2 = sum(lsprob2)
  lsprob = da.prob(xi=df$xi,wt=wtT,t_ext=df$t.ext,s=s,mu=mu0,r=df$bt[dim]-c(0,df$bt[-dim]),n=n)
  ls = sum(lsprob)
  return(list(ls=ls,ls2=ls2))
}

df = sim.extinct(btdd,pars)
test.g(df,pars)
```

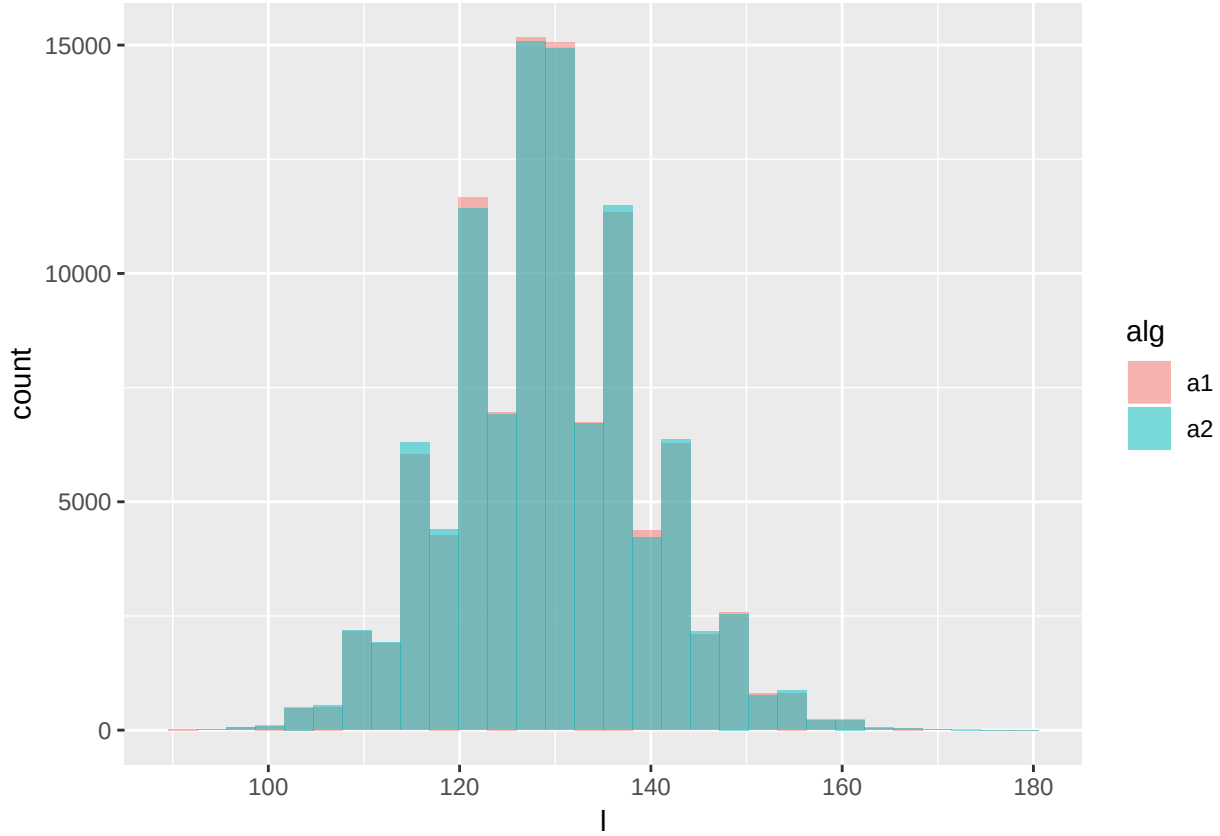
Data augmentation algorithm

We have two options on ..

```
no_cores <- detectCores()
cl <- makeCluster(no_cores)
registerDoParallel(cl)
m=100000
time = proc.time()
trees <- foreach(i = 1:m, combine = list) %dopar% {
  df =  emphasis::sim.extinct(brts = btdd,pars = pars)
  l1 = length(df$bt)
  return(list(l1=l1))
}
t1 = get.time(time)
l1 = sapply(trees,function(list) list$l1)
time = proc.time()
trees <- foreach(i = 1:m, combine = list) %dopar% {
  df =  emphasis::sim.extinct2(brts = btdd,pars = pars)
  l2 = length(df$bt)
  return(list(l2=l2))
}
l2 = sapply(trees,function(list) list$l2)
t2 = get.time(time)
stopCluster(cl)
```

```
dfh = data.frame(l=c(l1,l2),alg = c(rep("a1",length(l1)),rep("a2",length(l2))))
ggplot(dfh,aes(x=l,fill=alg)) + geom_histogram(alpha=0.5,position='identity')
```

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




```
summary(l1)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    90.0   122.0   130.0   129.4   136.0   176.0
```

```
summary(l2)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    90.0   122.0   130.0   129.4   136.0   178.0
```

```
t1
```

```
## [1] 111.107
```

```
t2
```

```
## [1] 263.588
```

two sampling algorithms

```
sim.extinct <- function(brts,pars,model='dd',seed=0){
  if(seed>0) set.seed(seed)
  wt = -diff(c(brts,0))
  ct = sum(wt)
```

```
lambda0 = pars[1]
mu0 = pars[2]
K = pars[3]
dim = length(wt)
ms = NULL # missing speciations, for now we just add time. When we consider topology we do it with sp
me = NULL # missing extinctions (in the uniform plane)
bt = NULL
bte = NULL
to = NULL
cbt = 0
N = 2
for(i in 1:dim){
  cwt = wt[i]
  cbt = sum(wt[0:(i-1)])
  key = 0
  while(key == 0){
    if(model == "dd"){  # diversity-dependence model
      lambda = max(1e-99, lambda0 - (lambda0-mu0)*N/K)
      mu = mu0
      s = N*lambda
    }else{print('Model not implemented yet, try dd')}
    t.spe = rexp(1,s)
    t.ext = extinction.processes(u=me,inits=ms,mu0=mu0)
    t_ext = ifelse(length(t.ext)>0,min(t.ext),Inf)-cbt  # if is not empty gives the waiting time for
    mint = min(t.spe,t_ext)
    if(mint < cwt){
      if(mint == t.spe){#speciation
        u = runif(1)
        if(u < pexp(ct-(cbt+t.spe),mu)){
          ms = c(ms,cbt+t.spe)
          me = c(me,u)
          bt = c(bt,cbt+t.spe)
          text = extinction.processes(u=u,inits=cbt+t.spe,mu0=mu0)
          bte = c(bte,text)
          to = c(to,1)
          N = N + 1
        }
        cwt = cwt - t.spe
        cbt = cbt + t.spe
      }
      else{#extinction
        extinctone = which(t.ext == min(t.ext))
        text = t.ext[extinctone]
        bt = c(bt,text)
        bte = c(bte,Inf)
        to = c(to,0)
        ms = ms[-extinctone]
        me = me[-extinctone]
        cwt = cwt - t_ext
        cbt = cbt + t_ext
        N = N-1
      }
    }
```

```
      else{
        key = 1
      }
    }
    N = N+1
  }
  df = data.frame(bt = c(bt,ct-brts),bte = c(bte, rep(Inf,length(wt))),to = c(to,rep(2,length(wt))))
  df = df[order(df$bt),]
  df$t.ext = df$bte-df$bt
  df = df[-1,]
  df = rbind(df,data.frame(bt=ct,bte=Inf,to=0,t.ext=Inf))
  return(df)
}

sim.extinct2 <- function(brts,pars,model='dd',seed=0){
  if(seed>0) set.seed(seed)
  wt = -diff(c(brts,0))
  ct = sum(wt)
  lambda0 = pars[1]
  mu0 = pars[2]
  K = pars[3]
  dim = length(wt)
  bt = NULL
  bte = NULL
  to = NULL
  N = 2
  for(i in 1:dim){
    cwt = wt[i]
    cbt = sum(wt[0:(i-1)])
    key = 0
    while(key == 0){
      if(model == "dd"){  # diversity-dependence model
        lambda = max(1e-99, lambda0 - (lambda0-mu0)*N/K)
        mu = mu0
        s = N*lambda
      }else{print('Model not implemented yet, try dd')}
      rns = rnhe(lambda=s,mu=mu,Ti=ct-cbt)
      t.spe = rns$rv
      ex = rns$ex
      sbte = bte[bte>cbt]
      t_ext = ifelse(length(sbte)>0,min(sbte),Inf) - cbt
      mint = min(t.spe,t_ext)
      if(mint < cwt){
        if(mint == t.spe){#speciation
          bt = c(bt,cbt+t.spe)
          text = cbt+t.spe+ex/mu0
          bte = c(bte,text)
          to = c(to,1)
          N = N + 1
          cwt = cwt - t.spe
          cbt = cbt + t.spe
        }
        else{#extinction
```

```
          bt = c(bt,cbt+t_ext)
          bte = c(bte,Inf)
          to = c(to,0)
          cwt = cwt - t_ext
          cbt = cbt + t_ext
          N = N-1
        }
      }
      else{
        key = 1
      }
    }
    N = N+1
  }
  df = data.frame(bt = c(bt,ct-brts),bte = c(bte, rep(Inf,length(wt))),to = c(to,rep(2,length(wt))))
  df = df[order(df$bt),]
  df$t.ext = df$bte-df$bt
  df = df[-1,]
  df = rbind(df,data.frame(bt=ct,bte=Inf,to=0,t.ext=Inf))
  return(df)
}
```