Importance sampling

On this report we denote D as the observed-part or extant-species, $+_i$ as the missing-part or extinct-species of the tree and D^+ is then the complete phylogenetic tree.

The EM algorithms consists on two steps. First, we calculate the conditional expectation:

$$Q(\theta|\theta^*) = E_{\theta^*}[logP(D^+|\theta)|D]$$

and then we perform the maximization:

$$\theta^{**} = argmax_{(\theta)} Q(\theta|\theta^*)$$

Given that the calculation of the conditional expectation is really hard (if not impossible), we use an approximation, sampling complete-phylogenies under a montecarlo approach. This simulations should be sampled from (real density)

$$f_{\theta^*}(+_i|D)$$

But instead we sample it from

$$g_{\theta^*}(+_i|D)$$

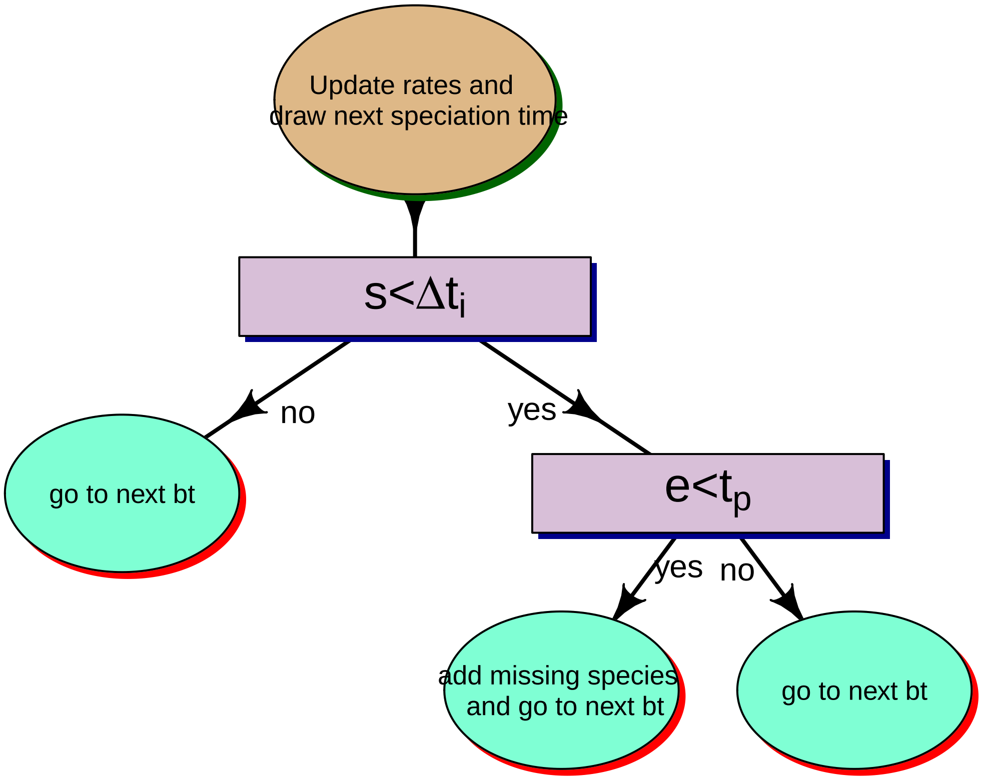
To correct this we re-weigh the approximation of the expectation by importance scaling:

$$w_i = \frac{f_{\theta^*}(+_i|D)}{g_{\theta^*}(+_i|D)} = \frac{f_i}{g_i}$$

Thus, the montecarlo re-weighted approximation has the form

$$E_{\theta^*}[logP(D^+|\theta)|D] \approx \frac{1}{N}\sum_{i=1}^{N} logP(D_i^+|\theta)\frac{f_i}{g_i}$$

- f_i is the density function (likelihood) of the complete phylogenetic tree.
- to calculate g_i we have 2 ways so far. The first one is related with the diagram above



we multiply the corresponding probabilities to have every outcome, this is done on the piece of code bellow

```
if (t_spe < cwt){
    t_ext = rexp(1,mu0)
    t_ext = cbt + t_spe + t_ext
    if (t_ext < ct){
      up = update_tree(wt=wt,t_spe = (cbt + t_spe), t_ext = t_ext, E = E, n = n)
      E = up$E
      n = up$n
      wt = up$wt
      fake = FALSE
      prob[i] = dexp(t_ext,rate=mu0,log=TRUE) + dexp(t_spe,rate=s,log = TRUE)
    }else{
      prob[i] = pexp(q = ct,rate = mu0,lower.tail = F,log.p = TRUE) + dexp(t_spe,rate=s, log = TRUE)
      fake = TRUE
      i = i-1
    }
  }else{
    fake = FALSE
    prob[i] = pexp(q = cwt,rate = s,lower.tail = F,log.p = TRUE)
  }
```

Note that we are working with multiplication of many densities, which in the end are very small values. To avoid cutting most numbers to zero we consider the following fact:

$$w_i = \frac{f_i}{g_i} = e^{log(f_i)-log(g_i)} = e^{loglik-\sum_j log(g_{i,j})}$$

and that is why on the code above the log of the densities is calculated. Considering that we calculate the weight in the following way

```
f_n = -llik(b=pars,n=n,E=E,t=wt)
weight = exp(f_n-sum(prob))
return(list(wt=wt,E=E,n=n,weight=weight,L=L,g=prob,f_n=f_n))
```

another way to calculate g is considering the probabilities of not having a new species on the interval Δt_i

$$\int_0^{\Delta t_i} s_{\lambda_i} e^{-s_{\lambda_i} t}[1-e^{-\mu(r_i-t)}]dt = s_{\lambda_i}\int_0^{\Delta t_i} e^{-s_{\lambda_i} t} - e^{t(\mu-s_{\lambda_i})-\mu r_i} = s_{\lambda_i}\int_0^{\Delta t_i} e^{-s_{\lambda_i} t}dt - s_{\lambda_i} e^{-\mu r_i}\int_0^{\Delta t_i} e^{t(\mu-s_{\lambda_i})}dt$$

$$= 1 - e^{-s_{\lambda_i}\Delta t} - \frac{s_{\lambda_i} e^{-\mu r_i}}{\mu - s_{\lambda_i}}[e^{\Delta t(\mu-s_{\lambda_i})} - 1]$$

so we do it with this function

```
convol <-function(wt,lambda,mu,remt){
  out = 1-exp(-lambda*wt)-lambda*exp(-mu*remt)/(mu-lambda)*(exp(wt*(mu-lambda))-1)
  return(out)
}
```

and we add this option to the code

```
 if (t_spe < cwt){
      t_ext = rexp(1,mu0)
      t_ext = cbt + t_spe + t_ext
      if (t_ext < ct){
        up = update_tree(wt=wt,t_spe = (cbt + t_spe), t_ext = t_ext, E = E, n = n)
        E = up$E
        n = up$n
        wt = up$wt
        fake = FALSE
        prob[i] = dexp(t_ext,rate=mu0,log=TRUE) + dexp(t_spe,rate=s,log = TRUE)
      }else{
        prob[i] = pexp(q = ct,rate = mu0,lower.tail = F,log.p = TRUE) + dexp(t_spe,rate=s,log = TRUE)
        if(v2){ prob[i] = log(convol(wt = t_spe,lambda = s,mu = mu,remt = ct-cbt))}
        fake = TRUE
        i = i-1
      }
    }else{
      fake = FALSE
      prob[i] = pexp(q = cwt,rate = s,lower.tail = F,log.p = TRUE)
      if(v2){prob[i] = log(convol(wt = t_spe,lambda = s,mu = mu,remt = ct-cbt))}
    }
```

Implementation

To have a better idea on how this implementation look like we do some simple observations.

We start simulating a whole tree

```
s <- sim_phyl()
```

Them we drop extinct species

```
s2 <- drop.fossil(s$newick)
```

we transform phylo format into branching-times, number of species, and topology vectors

```r
s2 <- phylo2p(s2)
```

Now we simulate a complete tree (extinct+extans species) based on the observed tree (extant species only)

```r
s3 <- rec_tree(wt=s2$wt)
```

and we observe the calculated weight for this tree

```r
s3$weight
```

```
## [1] -185.1205
```

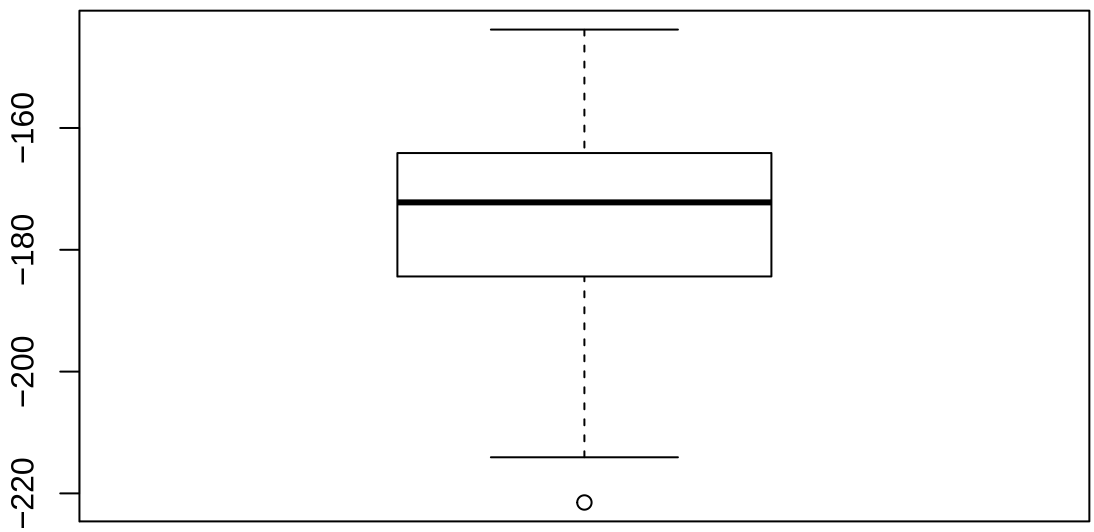
it seems very small

This is a random process, if we do it again we end with another complete-tree and its corresponding weight, we do it for 30 iterations just to have an idea of the variability of the weight

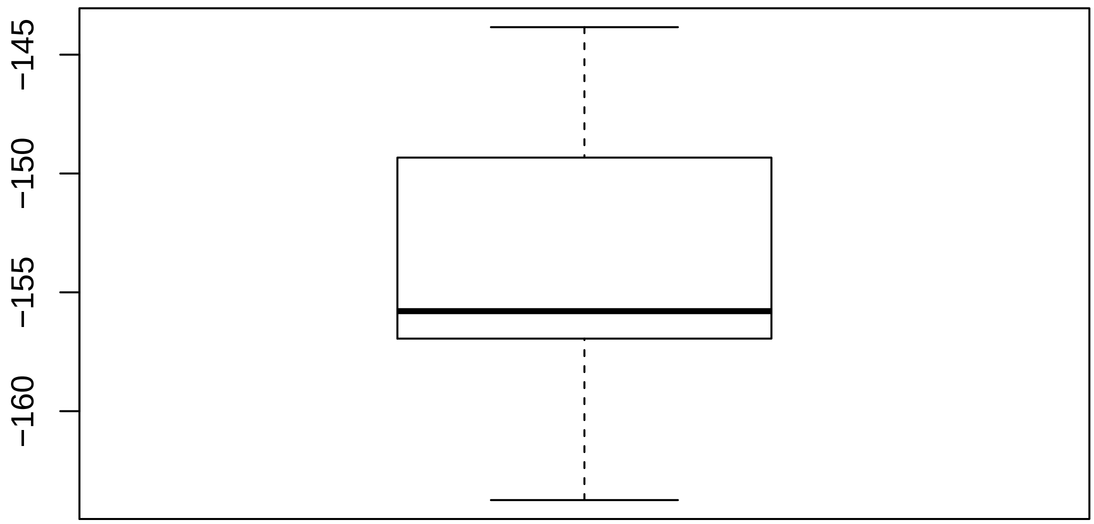
```r
w=0
nLTT=0
n_it = 100
st=sim_srt(wt=s2$wt,pars=c(0.8,0.0175,0.1),n_trees = n_it)
for(i in 1:n_it){
  st3 = st[[i]]
  w[i] = st3$weight
  rec = p2phylo(st3)
  nLTT[i] = nLTTstat(s$newick, rec, distance_method = "abs")
}
su=summary(w)
su
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -221.5  -184.2  -172.2  -174.6  -164.3  -143.8
```

```r
boxplot(w)
```

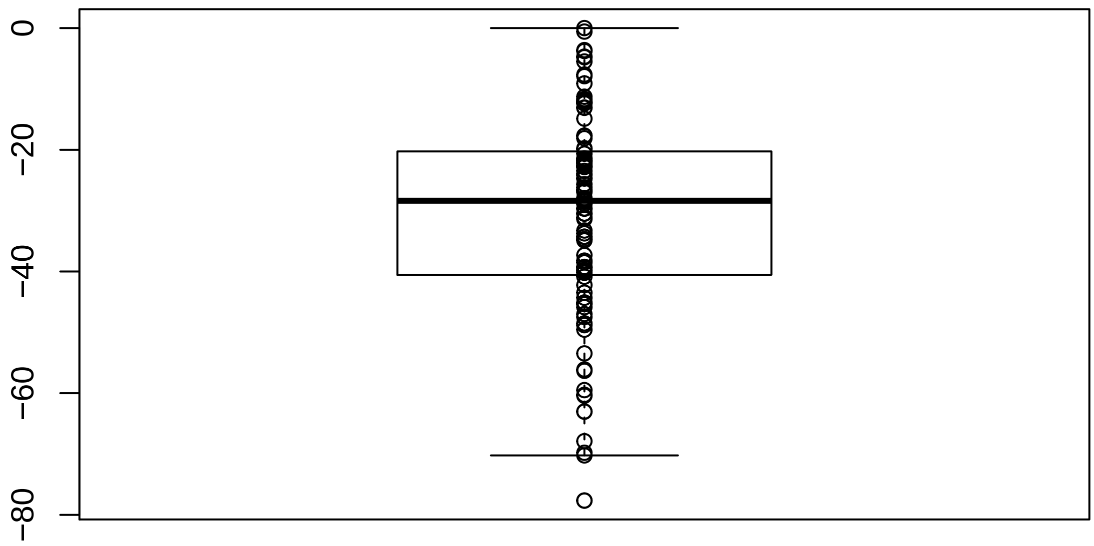



```r
q3 = w>su[5]
boxplot(w[q3])
```



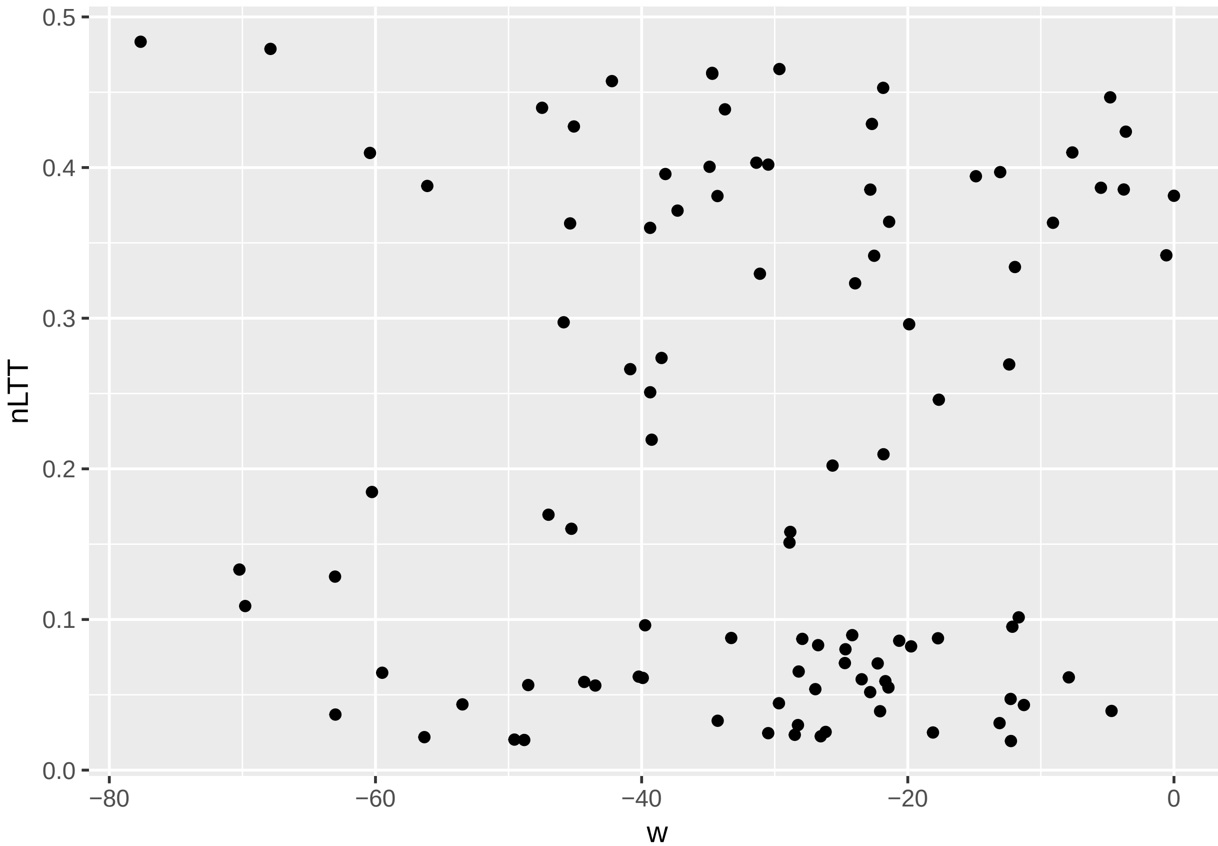
They seems very small to be the logarithm of the weight, we centreate it to avoid numerical issues

```
w = w - max(w)
boxplot(w)
points(rep(1,length(w)),w)
```

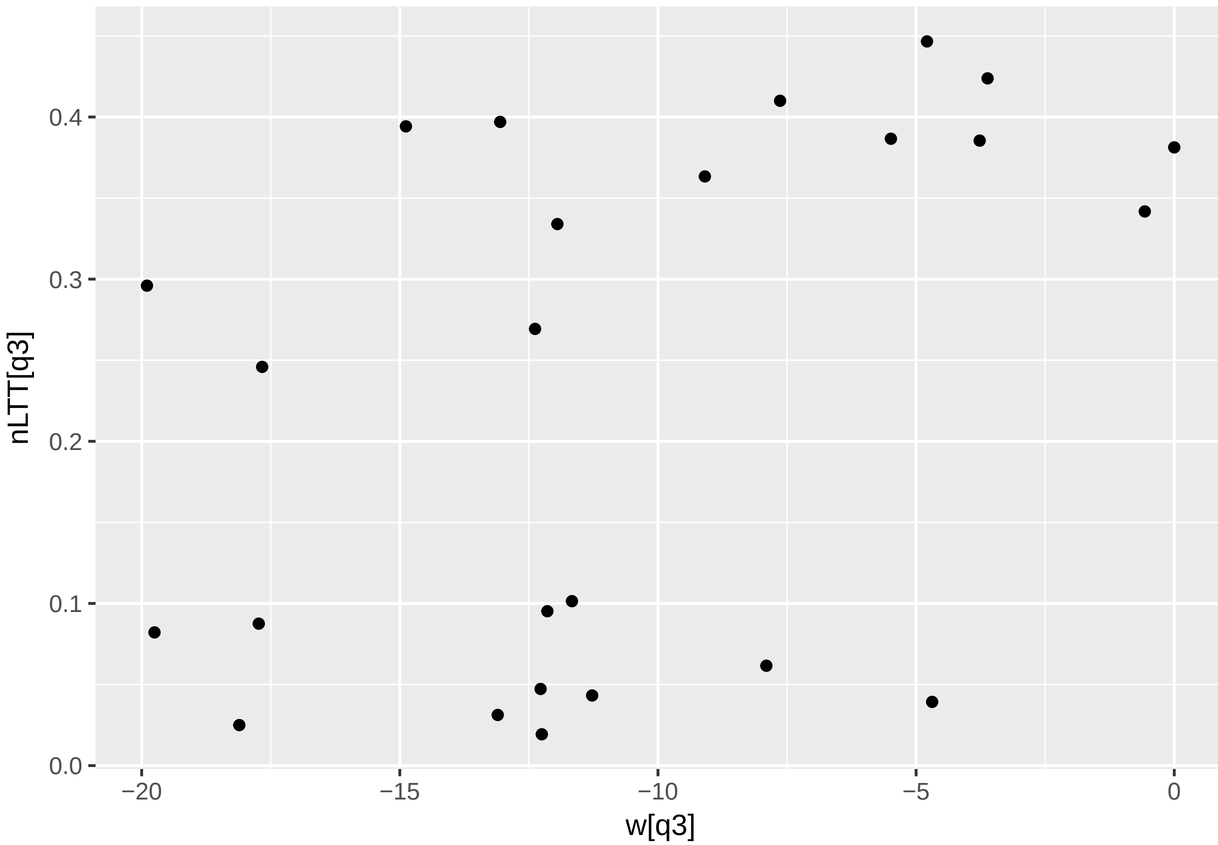


now we would like to check if trees with larger weight are really ''better trees". To do that we check the nLTT statistic

```
qplot(w,nLTT)
```



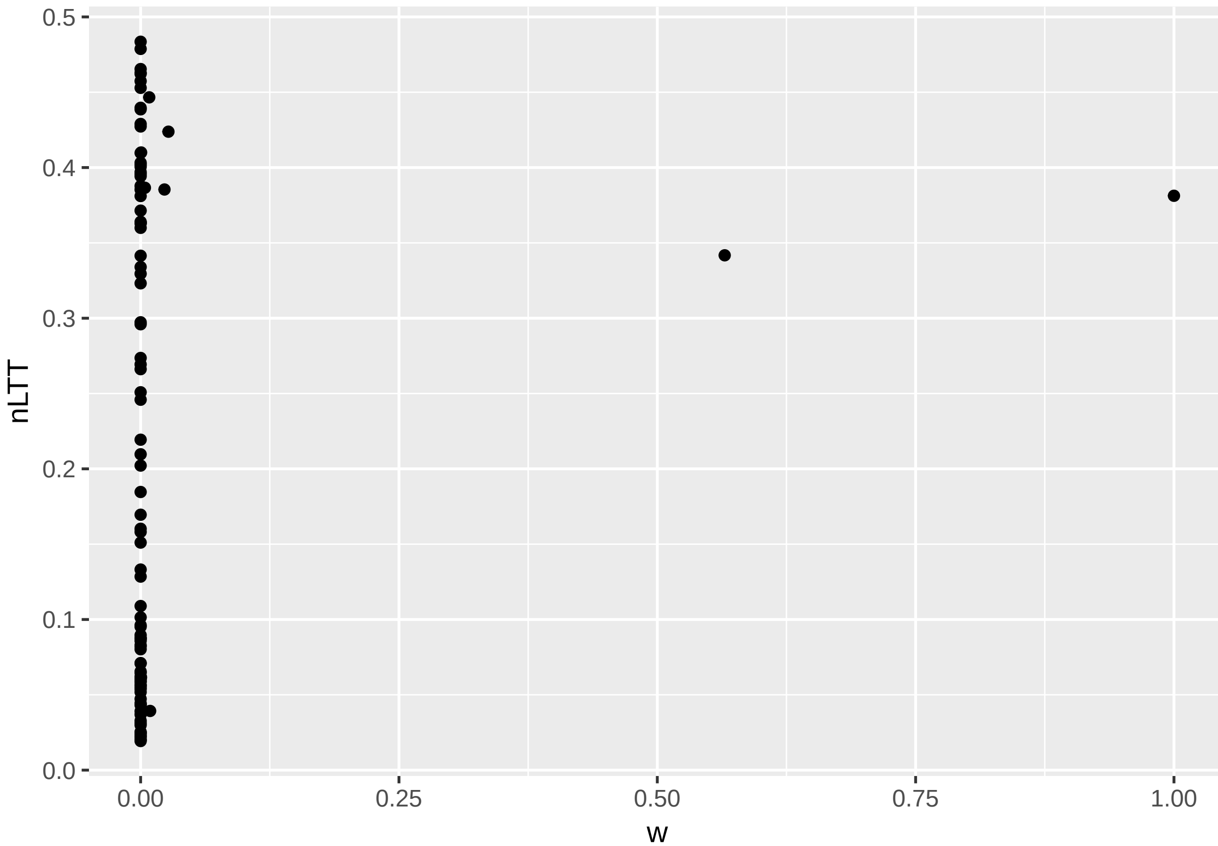

```
qplot(w[q3],nLTT[q3])
```



we can see that there is not relationship between the weight and the nLTT statistic. There is clearly something wrong about it.

Now let´s check the real weights, that is

```
w = exp(w)
qplot(w,nLTT)
```



---

now the same but with the second way

```
s3 <- rec_tree(wt=s2$wt,v2=T)
```

and we observe the calculated weight for this tree

```
s3$weight
```

```
## [1] -96.83191
```

it seems very small

This is a random process, if we do it again we end with another complete-tree and its corresponding weight, we do it for 30 iterations just to have an idea of the variability of the weight

```
w=0
for(i in 1:100){
st3 <- rec_tree(wt=s2$wt,v2=T)
w[i] = st3$weight
}
print(w)
```

```
##   [1]    0.00000  -93.05403  -95.08655  -85.05300    0.00000  -76.83888
##   [7]  -79.45153  -69.96837  -82.47960 -100.19506  -82.05855    0.00000
##  [13]  -84.03725    0.00000  -83.32769  -77.73729  -89.25524    0.00000
##  [19]  -95.47873    0.00000 -119.29149  -84.94644  -89.57649  -75.20586
##  [25]  -72.48798    0.00000  -91.56089  -84.17729    0.00000  -99.00831
##  [31]  -80.04266  -99.23032  -85.71132  -83.78256 -107.85903  -77.20945
##  [37]  -85.79230  -73.75916 -106.13564  -68.28258  -80.42210 -102.05702
```

```
## [43]  -98.70433   0.00000  -75.53201  -91.33869  -81.92048  -80.55669
## [49]  -91.65242  -94.97148 -207.37909  -84.68166   0.00000  -97.80541
## [55]  -72.16234 -102.37005  -82.10699  -92.84550  -75.18181  -95.20462
## [61]  -88.52505  -87.60505  -78.43630  -82.39785   0.00000   0.00000
## [67]  -78.82056  -91.99678 -102.26122 -123.33859  -86.07239  -83.06771
## [73]    0.00000  -88.21611  -91.52649  -79.08647   0.00000 -105.77050
## [79] -119.13902  -88.14998   0.00000  -92.80362  -83.25662 -146.10010
## [85] -112.67866 -119.97644  -95.95013  -76.82955 -102.89044   0.00000
## [91]  -82.02418 -114.73375  -96.38999 -101.82842  -77.69885 -103.14001
## [97]  -85.07219  -92.08088 -102.26961  -79.95846
```

```
w = w - max(w)
print(w)
```

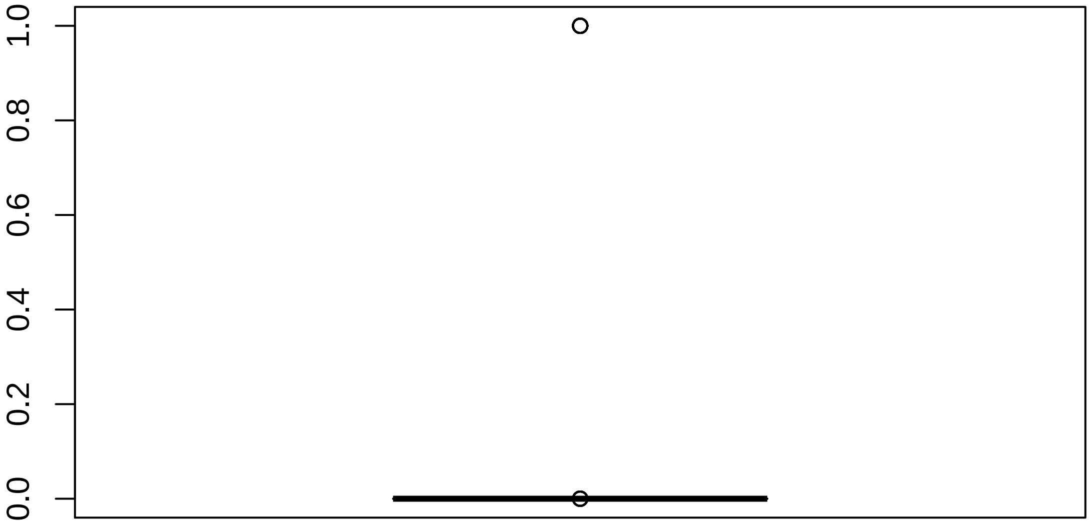
```
##   [1]    0.00000  -93.05403  -95.08655  -85.05300    0.00000  -76.83888
##   [7]  -79.45153  -69.96837  -82.47960 -100.19506  -82.05855    0.00000
##  [13]  -84.03725    0.00000  -83.32769  -77.73729  -89.25524    0.00000
##  [19]  -95.47873    0.00000 -119.29149  -84.94644  -89.57649  -75.20586
##  [25]  -72.48798    0.00000  -91.56089  -84.17729    0.00000  -99.00831
##  [31]  -80.04266  -99.23032  -85.71132  -83.78256 -107.85903  -77.20945
##  [37]  -85.79230  -73.75916 -106.13564  -68.28258  -80.42210 -102.05702
##  [43]  -98.70433    0.00000  -75.53201  -91.33869  -81.92048  -80.55669
##  [49]  -91.65242  -94.97148 -207.37909  -84.68166    0.00000  -97.80541
##  [55]  -72.16234 -102.37005  -82.10699  -92.84550  -75.18181  -95.20462
##  [61]  -88.52505  -87.60505  -78.43630  -82.39785    0.00000    0.00000
##  [67]  -78.82056  -91.99678 -102.26122 -123.33859  -86.07239  -83.06771
##  [73]    0.00000  -88.21611  -91.52649  -79.08647    0.00000 -105.77050
##  [79] -119.13902  -88.14998    0.00000  -92.80362  -83.25662 -146.10010
##  [85] -112.67866 -119.97644  -95.95013  -76.82955 -102.89044    0.00000
##  [91]  -82.02418 -114.73375  -96.38999 -101.82842  -77.69885 -103.14001
##  [97]  -85.07219  -92.08088 -102.26961  -79.95846
```

```
w = exp(w)
print(w)
```

```
##   [1] 1.000000e+00 3.864985e-41 5.063312e-42 1.153326e-37 1.000000e+00
##   [6] 4.258912e-34 3.123481e-35 4.103201e-31 1.512046e-36 3.060838e-44
##  [11] 2.303682e-36 1.000000e+00 3.184822e-37 1.000000e+00 6.475038e-37
##  [16] 1.734297e-34 1.725604e-39 1.000000e+00 3.420681e-42 1.000000e+00
##  [21] 1.557279e-52 1.283005e-37 1.251484e-39 2.180265e-33 3.302715e-32
##  [26] 1.000000e+00 1.720319e-40 2.768655e-37 1.000000e+00 1.002851e-43
##  [31] 1.729470e-35 8.031896e-44 5.971003e-38 4.108627e-37 1.436879e-47
##  [36] 2.940092e-34 5.506499e-38 9.264132e-33 8.051473e-47 2.214385e-30
##  [41] 1.183381e-35 4.755540e-45 1.359104e-43 1.000000e+00 1.573485e-33
##  [46] 2.148389e-40 2.644757e-36 1.034367e-35 1.569853e-40 5.680821e-42
##  [51] 8.637889e-91 1.671950e-37 1.000000e+00 3.339272e-43 4.573968e-32
##  [56] 3.477387e-45 2.194764e-36 4.761130e-41 2.233348e-33 4.499422e-42
##  [61] 3.581451e-39 8.986892e-39 8.620771e-35 1.640842e-36 1.000000e+00
##  [66] 1.000000e+00 5.870366e-35 1.112519e-40 3.877184e-45 2.721012e-54
##  [71] 4.161379e-38 8.397524e-37 1.000000e+00 4.877851e-39 1.780531e-40
##  [76] 4.499678e-35 1.000000e+00 1.159997e-46 1.813763e-52 5.211349e-39
##  [81] 1.000000e+00 4.964774e-41 6.951998e-37 3.544315e-64 1.159522e-49
##  [86] 7.850425e-53 2.134951e-42 4.298816e-34 2.066571e-45 1.000000e+00
##  [91] 2.384237e-36 1.485127e-50 1.375172e-42 5.976947e-45 1.802259e-34
##  [96] 1.610135e-45 1.131405e-37 1.022780e-40 3.844808e-45 1.881402e-35
```

So, a first observation, the values of w are really small and varies a lot. That is amaking a lot of numerical inestabilities.

```r
boxplot(w)
```




```r
summary(w)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    0.00    0.00    0.16    0.00    1.00
```

we can also see how few trees has almost the whole weight and defines the outcome