Chapter 3: The total cophenetic index

Lucia Rotger

Contents

Packages required

The functions used in this chapter need these packages to be installed and loaded in the session

```
library(Zseq)
library(gmp)
library(ape)
library(CollessLike)
```

List of all binary trees

We have obtained all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ with the package *phylonetwork* for *Python*:

```
import phylonetwork.generators as gen
from phylonetwork.distances import cophenetic_distance as cophdist
from math import factorial

for n in range(3,9):
    taxa = [str(i+1) for i in range(n)]
    tg = gen.all_trees(taxa = taxa, binary = True, nested_taxa = False)
    trees = list(tg)
    newicks = []
    file = open("bintrees-n"+str(n)+".txt", "w+")
    for i in range(len(trees)):
        newicks.append(trees[i].eNewick())
        print >>file, newicks[i]
    file.close()
```

The resulting lists of trees can be consulted in the *List of Trees* folder of the GitHub repository.

General functions

Next functions are needed in some computations of the sections below.

```
big.factorial = function(n){
  if(n<2) return(1)
  return(Factorial(n+1)[n+1])
}

big.double.factorial = function(n){
  if(n<2) return(1)
  m = (n+2+n%%2)/2
  return(Factorial.Double(m,odd=(n%%2==1))[m])
}

big.binomial = function(n,k){
  return(big.factorial(n)/(big.factorial(k)*big.factorial(n-k)))
}

Cknk = function(k,n){
  return(big.binomial(n,k)*((big.double.factorial(2*k-3)*
          big.double.factorial(2*(n-k)-3))/(2*big.double.factorial(2*n-3))))
}
```

3.3 Expected value of Φ under the Yule model

The formula in Theorem 3.19 can be computed with the following function:

```
harmonic=function(n){return(sum(1/(1:n)))}
EYPhi=function(n){
  return(n*(n+1-2*harmonic(n)))
}
```

For $n = 3, ..., 20$ the results are:

```
sapply(3:20,EYPhi)
```

```
##  [1]   1.000000   3.333333   7.166667  12.600000  19.700000  28.514286
##  [7]  39.078571  51.420635  65.562698  81.522944  99.316522 118.956255
## [13] 140.453130 163.816672 189.055214 216.176109 245.185893 276.090414
```

To double-check the formula, we have computed the values of $E_Y(\Phi)$, for $n = 3, \ldots, 8$, from the cophenetic indices of all trees in the corresponding $\mathcal{BT}_n$.

First of all, we have considered all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ obtained with the package *phylonetwork* for *Python* in this section.

Moreover, we have to take into consideration the probabilities of each tree under the Yule model:

```
yule.prob = function(tree){
  if (class(tree)=="phylo")
    tree=graph.edgelist(tree$edge, directed=TRUE)
  sp = shortest.paths(tree,mode = "out")
  deg = degree(tree,mode="out")
  leaves = which(deg==0)
```

```
  n = length(leaves)
  k.node = function(node){
    subtree=which(sp[node,]<Inf)
    return(length(intersect(leaves,subtree)))
  }
  kappas = sapply(which(deg>0), k.node)
  value = (2^(n-1)/as.numeric(big.factorial(n)))*prod(1/(kappas-1))
  return(value)
}
```

Afterwards, we can compute the total cophenetic index with the package *CollessLike* and the expected value of the indices of each n

```
exp.yule = c()
for(n in 3:8){
  trees=read.tree(file=paste("./bintrees-n",n,".txt",sep=""))
  indices = sapply(trees, cophen.index)
  probs=sapply(trees, yule.prob)
  exp.vule[n]=sum(indices*probs)
}
```

Obtaining the following results:

n	3	4	5	6	7	8
$E_Y(\Phi_n)$	1	3.3333	7.1667	12.6	19.7	28.5143

agreeing with the figures given by our formula.

3.4 Expected value of Φ under the uniform model

The formula in Theorem 3.25 can be computed with the following function (`big.double.factorial` from common functions is needed):

```
EUPhi = function(n){
  return(as.numeric((n*(n-1)/4)*(big.double.factorial(2*n-2)/
                                 big.double.factorial(2*n-3)-2)))
}
```

For $n = 3, ..., 20$ the results are:

```
sapply(3:20,EUPhi)
```

```
##  [1]   1.000000   3.600000   8.285714  15.476190  25.545455  38.834499
##  [7]  55.658741  76.313040 101.075256 130.208893 163.965117 202.584342
## [13] 246.297504 295.327098 349.888046 410.188417 476.430046 548.809061
```

To double-check the formula, we have computed the values of $E_U(\Phi)$, for $n = 3, \ldots, 8$, from the cophenetic indices of all trees in the corresponding $\mathcal{BT}_n$.

First of all, we have considered all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ obtained with the package *phylonetwork* for *Python* in this section.

Afterwards, we can compute the total cophenetic index with the package *CollessLike* and the expected value, under the uniform model, of the indices of each n

```
exp.uni = c()
for(n in 3:8){
  trees=read.tree(file=paste("./bintrees-n",n,".txt",sep=""))
  indices = sapply(trees, cophen.index)
  exp.uni[n]=mean(indices)
}
```

Obtaining the following results:

n	3	4	5	6	7	8
$E_U(\Phi_n)$	1	3.6	8.2857	15.4762	25.5455	38.8345

agreeing with the figures given by our formula.

3.5 On the variance of Φ under the uniform model

Computing the variance of Φ_n using our formula

We can compute $\sigma^2_U(\Phi_n)$ using the recurrence formula for $E_U(\Phi_n^2)$ and the exact formula of $E_U(\Phi_2)^2$ and obtaining:

$$\sigma^2_U(\Phi_n) = E_U(\Phi_n^2) - E_U(\Phi_2)^2$$

The following functions are needed to compute the desired variance, in addition to the functions of the common block (see this section)

```
EUPhi = function(n){
    return(as.numeric((n*(n-1)/4)*(big.double.factorial(2*n-2)/
                                     big.double.factorial(2*n-3)-2)))
}

term.Phi = function(n){
  return(mul.bigq(as.bigq(n*(n-1)/2),(mul.bigq(as.bigq((49*n^3-57*n^2-22*n+24)/48),
                  big.double.factorial(2*n-4)/big.double.factorial(2*n-3))-
                    as.bigq((63*n^2-95*n+28)/30))))
}

compute.EUPhi2 = function(n.max=500){
  terms = lapply(2:n.max,term.Phi)
  terms = c(0,terms)
  exp.values = list(0)
  for(n in 2:n.max){
    sums = 0
    if(n>2){
      for(k in 2:(n-1)){
        sums = sums + Cknk(k,n)*exp.values[[k]]
      }
      sums = 2*sums
    }
    sums = sums + terms[[n]]
    exp.values[[n]] = sums
```

```
    print(n)
  }
  exp.values = sapply(exp.values, as.numeric)
  write.table(exp.values,file = paste("EU(Phi2)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(exp.values)
}

compute.varUPhi = function(exp.values,n.max=500){
  var.form = function(i)return(exp.values[i]-EUPhi(i)^2)
  var.values = sapply(1:n.max, var.form)
  write.table(var.values,file = paste("varU(Phi)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(var.values)
}
```

We have computed these variances up to $n = 1000$ with the following instructions:

```
exp.values.Phi = compute.EUPhi2(1000)
var.values.Phi = compute.varUPhi(exp.values.Phi,1000)
```

NOTE: these computations might take a long time to finish.

For $n = 3, ..., 20$ the results have been:

```
exp.values.Phi[3:20]
```

```
##  [1]      1.00000     13.60000     73.42857    259.09524    711.54545
##  [6]   1654.34965   3414.67413   6444.77869  11343.93737  18880.70867
## [11]  30015.50122  45923.39304  68017.17207  97970.57161 137741.67942
## [16] 189596.50276 256132.67433 340303.28664
```

```
var.values.Phi[3:20]
```

```
##  [1]     0.00000     0.64000     4.77551    19.58277    58.97521
##  [6]   146.23135   316.77865   621.09863  1127.72999  1926.35278
## [11]  3130.94150  4882.97724  7354.71170 10752.47676 15320.03476
## [16] 21341.96545 29147.08600 39111.90118
```

The rest of the values are available in the files "EU(Phi2)1000.txt" and "varU(Phi)1000.txt".

We have estimated the main order in the expansion of $\sigma_U^2(\Phi_n)$ as a function of n, by performing the minimum squares linear regression of $\ln(\sigma_U^2(\Phi_n))$ as a function of $\ln(n)$ for $n = 900, \ldots, 1000$,

```
summary(lm(log(var.values.Phi[900:1000])~log(900:1000)))
```

```
##
## Call:
## lm(formula = log(var.values.Phi[900:1000]) ~ log(900:1000))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.330e-05 -1.131e-05  4.161e-06  1.340e-05  1.671e-05
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.8743868  0.0003353  -11555   <2e-16 ***
## log(900:1000)  5.0657352  0.0000489  103586   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.509e-05 on 99 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 1.073e+10 on 1 and 99 DF,  p-value: < 2.2e-16
```

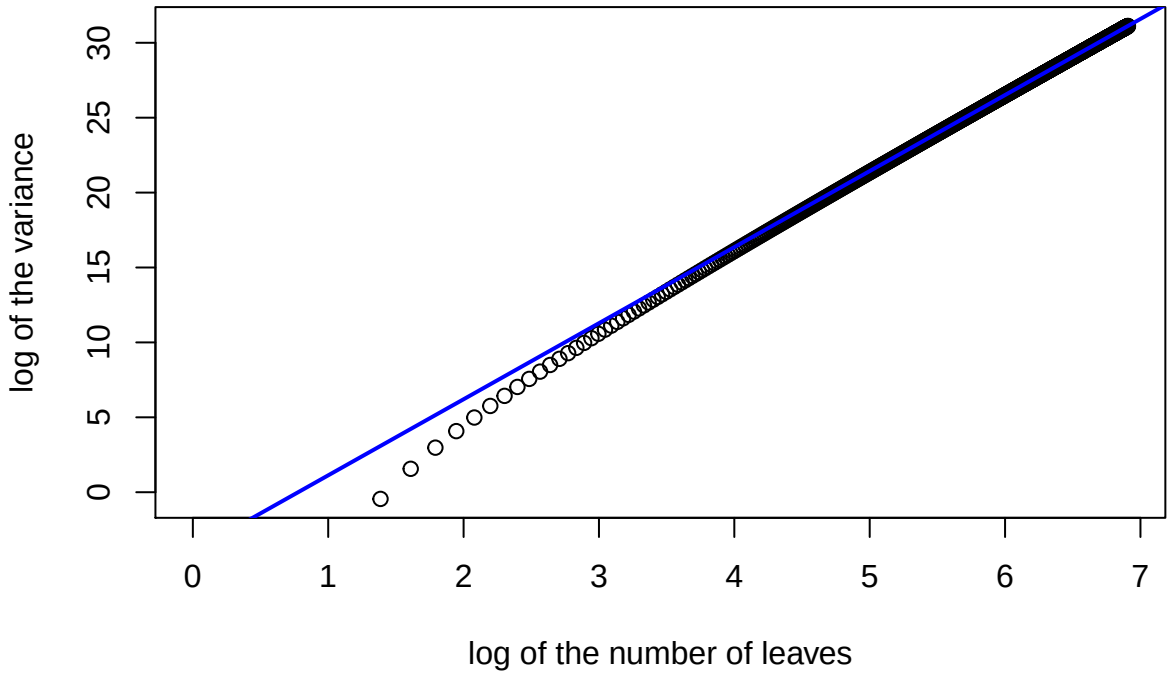
and the result has been

$$\ln(\sigma_U^2(\Phi_n)) \approx -3.8743868 + 5.0657352 \cdot \ln(n),$$

with a determination coefficient $R^2 \approx 1$. We conclude then that, according to our approximations, $\sigma_U^2(\Phi_n)$ is in $O(n^{5.0657})$. We conjecture that, actually, $\sigma_U^2(\Phi_n)$ is in $O(n^5)$, the order of $E_U(\Phi_n)^2$.

Next figure displays $\ln(\sigma_U^2(\Phi_n))$ as a function of $\ln(n)$, together with the corresponding regression line.

```
plot(log(1:1000),log(var.values.Phi),xlab="log of the number of leaves",
     ylab="log of the variance")
reg.phi=lm(log(var.values.Phi[500:1000])~log(500:1000))
abline(reg.phi,col="blue",lwd=2)
```



Computing the variance of Φ_n from the cophenetic indices

To double-check the recurrence, we have computed the values of $\sigma_U(\Phi_n)$, for $n = 3, \ldots, 8$, from the cophenetic indices of all trees in the corresponding $\mathcal{BT}_n$.

First of all, we have considered all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ obtained with the package *phylonetwork* for *Python* in this section.

Afterwards, we can compute the total cophenetic index with the package *CollessLike* and compute the variance of the indices of each n:

```
var.n = function (vec) return(var(vec)*(length(vec)-1)/length(vec))
trees = list()
all.cophen.index = list()
real.var.Phi = c()
for(n in 3:8){
  trees[[n]] = read.tree(file = paste("bintrees-n",n,".txt",sep = ""))
  all.cophen.index[[n]] = sapply(trees[[n]],cophen.index)
  real.var.Phi[n] = var.n(all.cophen.index[[n]])
  print(paste("var(Phi",n,") = ",real.var.Phi[n],sep = ""))
}
```

Obtaining the following results:

n	3	4	5	6	7	8
$\sigma^2_U(\Phi_n)$	0	0.64	4.7755	19.5828	58.9752	146.2314

agreeing with the figures given by our recurrence.

3.6 On the variance of S under the uniform model

Computing the variance of S_n using our formula

We can compute $\sigma^2_U(S_n)$ using the recurrence formula for $E_U(S_n^2)$ and the exact formula of $E_U(S_n)^2$ and obtaining:

$$\sigma^2_U(S_n) = E_U(S_n^2) - E_U(S_n)^2$$

The following functions are needed to compute the desired variance, in addition to the functions of the common block (see this section)

```
EUS = function(n){
    return(as.numeric(n*(big.double.factorial(2*n-2)/big.double.factorial(2*n-3)-1)))
}

term.S = function(n){
  return((5*n*2^(n-2)*big.factorial(n))/(big.double.factorial(2*n-3))-n*(5*n-2))
}

compute.EUS2 = function(n.max=500){
  terms = lapply(2:n.max,term.S)
  terms = c(0,terms)
  exp.values = list(0)
  for(n in 2:n.max){
    sums = 0
    if(n>2){
      for(k in 2:(n-1)){
        sums = sums + Cknk(k,n)*exp.values[[k]]
      }
      sums = 2*sums
```

```
    }
    sums = sums + terms[[n]]
    exp.values[[n]] = sums
    print(n)
  }
  exp.values = sapply(exp.values, as.numeric)
  write.table(exp.values,file = paste("EU(S2)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(exp.values)
}

compute.varUS = function(exp.values, n.max){
  var.form = function(i)return(exp.values[i]-EUS(i)^2)
  var.values = sapply(1:n.max,var.form)
  write.table(var.values,file = paste("varU(S)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(var.values)
}
```

We have computed these variances up to $n = 1000$ with the following instructions:

```
exp.values.S = compute.EUS2(1000)
var.values.S = compute.varUS(exp.values.S,1000)
```

NOTE: these computations might take a long time to finish.

For $n = 3, ..., 20$ the results have been:

```
exp.values.S[3:20]
```

```
##  [1]    25.0000    77.6000   177.2857   340.0952   582.4545   921.0816
##  [7]  1372.9245  1955.1189  2684.9571  3579.8650  4657.3837  5935.1556
## [13]  7430.9128  9162.4673 11147.7035 13404.5711 15951.0795 18805.2931
```

```
var.values.S[3:20]
```

```
##  [1]   0.0000000   0.1600000   0.7755102   2.2358277   4.9990817
##  [6]   9.5765183  16.5219346  26.4241938  39.9016992  57.5981796
## [11]  80.1793886 108.3304640 142.7537743 184.1671371 233.3023247
## [16] 290.9037954 357.7276063 434.5404734
```

The rest of the values are available in the files "EU(S2)1000.txt" and "varU(S)1000.txt".

We have estimated the main order in the expansion of $\sigma_U^2(S_n)$ as a function of n, by performing the minimum squares linear regression of $\ln(\sigma_U^2(S_n))$ as a function of $\ln(n)$ for $n = 900, \ldots, 1000$,

```
summary(lm(log(var.values.S[900:1000])~log(900:1000)))
```

```
##
## Call:
## lm(formula = log(var.values.S[900:1000]) ~ log(900:1000))
##
## Residuals:
##        Min         1Q     Median         3Q        Max
## -3.907e-05 -1.327e-05  4.879e-06  1.573e-05  1.961e-05
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept)   -2.347e+00  3.935e-04   -5965   <2e-16 ***
## log(900:1000)  3.079e+00  5.739e-05   53646   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.771e-05 on 99 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 2.878e+09 on 1 and 99 DF,  p-value: < 2.2e-16
```

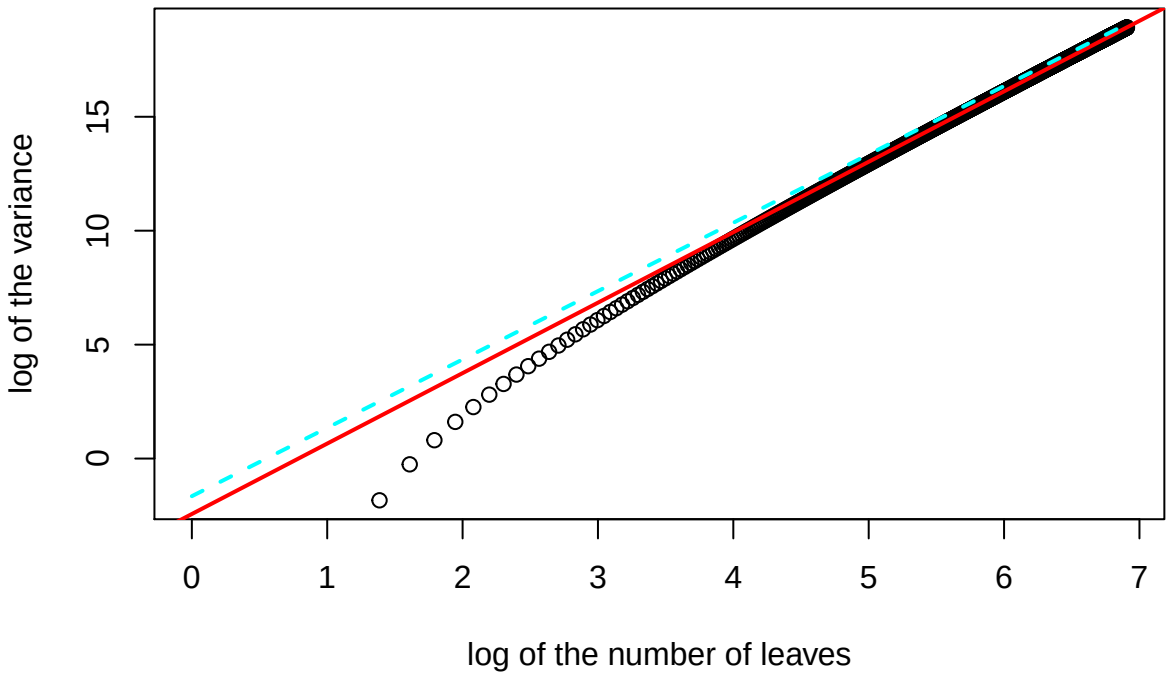
and the result has been

$$\ln(\sigma_U^2(S_n)) \approx -2.347121 + 3.078995 \cdot \ln(n),$$

with a determination coefficient $R^2 \approx 1$. We conclude then that, according to our approximations, $\sigma_U^2(S_n)$ is in $O(n^{3.078995})$. We conjecture that, actually, $\sigma_U^2(S_n)$ is in $O(n^3)$, the order of $E_U(S)^2$.

Next figure displays $\ln(\sigma_U^2(S_n))$ as a function of $\ln(n)$, together with the corresponding regression line and the approximation for $\sigma_U^2(S_n)$.

```
plot(log(1:1000),log(var.values.S),xlab="log of the number of leaves",
     ylab="log of the variance")
reg.S=lm(log(var.values.S[500:1000])~log(500:1000))
abline(reg.S,col="red",lwd=2)
sackin.approx = ((10-3*pi)/3)*(1:1000)^3
lines(log(1:1000),log(sackin.approx),col="cyan",lty=2,lwd=2)
```



Computing the variance of S from the Sackin indices

To double-check the recurrence, we have computed the values of $\sigma_U(S_n)$, for $n = 3, \ldots, 8$, from the Sackin indices of all trees in the corresponding $\mathcal{BT}_n$.

First of all, we have considered all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ obtained with the package *phylonetwork* for *Python* in the this section.

Afterwards, we can compute the Sackin index with the package *CollessLike* and compute the variance of the indices of each n:

```
var.n = function (vec) return(var(vec)*(length(vec)-1)/length(vec))
trees = list()
all.sackin.index = list()
real.var.sackin = c()
for(n in 3:8){
  trees[[n]] = read.tree(file = paste("bintrees-n",n,".txt",sep = ""))
  all.sackin.index[[n]] = sapply(trees[[n]],sackin.index)
  real.var.sackin[n] = var.n(all.sackin.index[[n]])
  print(paste("var(S_",n,") = ",real.var.sackin[n],sep = ""))
}
```

Obtaining the following results:

n	3	4	5	6	7	8
$\sigma^2_U(S_n)$	0	0.16	0.7755	2.2358	4.9991	9.5765

agreeing with the figures given by our recurrence.

3.7 On the covariance of S and Φ under the uniform model

Computing the covariance of S_n and Φ_n using our formula

We can compute $Cov_U(S_n, \Phi_n)$ using the recurrence formula for $E_U(S_n\Phi_n)$ and the exact formula of $E_U(S_n)$ and $E_U(\Phi)$, and obtaining:

$$Cov_U(S_n, \Phi_n) = E_U(S_n\Phi_n) - E_U(S_n)E_U(\Phi)$$

The following functions are needed to compute the covariance, in addition to the functions of the common block (see this section) and `EUPhi` from Section 3.5 and `EUS` from Section 3.6

```
term.cov = function(n){
  return(((13*n^2-9*n-2)*2^(n-5)*big.factorial(n))/(big.double.factorial(2*n-3))-
           (n*(n-1)/2)*(5*n-2))
}

compute.EUcov = function(n.max=500){
  terms = lapply(2:n.max,term.cov)
  terms = c(0,terms)
  exp.values = list(0)
  for(n in 2:n.max){
    sums = 0
```

```
    if(n>2){
      for(k in 2:(n-1)){
        sums = sums + Cknk(k,n)*exp.values[[k]]
      }
      sums = 2*sums
    }
    sums = sums + terms[[n]]
    exp.values[[n]] = sums
    print(n)
  }
  exp.values = sapply(exp.values, as.numeric)
  write.table(exp.values,file=paste("EU(SxPhi)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(exp.values)
}

compute.cov = function(exp.values,n.max = 500){
  cov.form = function(i)return(exp.values[i]-EUS(i)*EUPhi(i))
  cov.values = sapply(1:n.max, cov.form)
  write.table(cov.values,file = paste("covU(SPhi)",n.max,".txt",sep = ""),
              row.names = F,col.names = F)
  return(cov.values)
}
```

We have computed these covariances and correlations up to $n = 1000$ with the following instructions:

```
exp.values.cov = compute.EUcov(1000)
cov.values = compute.cov(exp.values.cov,1000)
```

NOTE: these computations might take a long time to finish.

For $n = 3, ..., 20$ the results have been:

```
exp.values.cov[3:20]
```

```
##  [1]     5.0000    32.0000   112.0000   291.0476   630.9091  1209.5478
##  [7]  2121.4881  3478.1045  5407.8581  8056.4954 11587.2180 16180.8299
## [13] 22035.8667 29368.7106 38413.6931 49423.1877 62667.6942 78435.9159
```

```
cov.values[3:20]
```

```
##  [1]    0.000000    0.320000    1.918367    6.580499   17.044077
##  [6]   37.089909   71.611701  126.671836  209.547294  328.768314
## [11]  494.151534  716.828809 1009.272571 1385.318372 1860.185104
## [16] 2450.493255 3174.281512 4051.021944
```

The rest of the values are available in the files "EU(SxPhi)1000.txt" and "covU(SPhi)1000.txt".

We have estimated the main order in the expansion of $Cov_U(S_n, \Phi_n)$ as a function of n, by performing the minimum squares linear regression of $\ln(Cov_U(S_n, \Phi_n))$ as a function of $\ln(n)$ for $n = 900, \ldots, 1000$,

```
summary(lm(log(cov.values[900:1000])~log(900:1000)))
```

```
##
## Call:
## lm(formula = log(cov.values[900:1000]) ~ log(900:1000))
##
## Residuals:
```

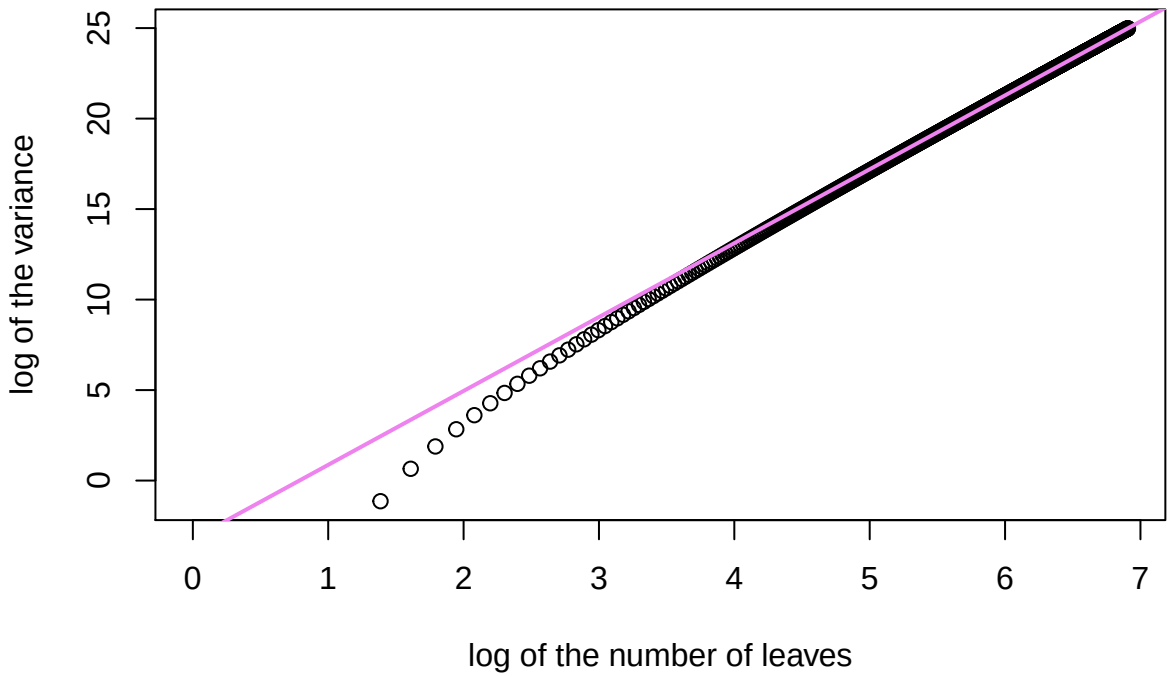
```
##        Min         1Q     Median         3Q        Max
## -3.553e-05 -1.207e-05  4.439e-06  1.430e-05  1.783e-05
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   -3.1333995  0.0003579   -8756   <2e-16 ***
## log(900:1000)  4.0709153  0.0000522   77993   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.611e-05 on 99 degrees of freedom
## Multiple R-squared:      1,  Adjusted R-squared:      1
## F-statistic: 6.083e+09 on 1 and 99 DF,  p-value: < 2.2e-16
```

and the result has been

$$\ln(Cov_U(S_n, \Phi_n)) \approx -3.1333995 + 4.0709153 \cdot \ln(n),$$

with a determination coefficient $R^2 \approx 1$. We conclude then that, according to our approximations, $Cov_U(S_n, \Phi_n)$ is in $O(n^{4.0709153})$. We conjecture that, actually, $Cov_U(S_n, \Phi_n)$ is in $O(n^4)$, the order of $E_U(S_n)E_U(\Phi_n)$. Next figure displays $\ln(Cov_U(S_n, \Phi_n))$ as a function of $\ln(n)$, together with the corresponding regression line.

```
plot(log(1:1000),log(cov.values),xlab="log of the number of leaves",
     ylab="log of the variance")
reg.cov=lm(log(cov.values[500:1000])~log(500:1000))
abline(reg.cov,col="violet",lwd=2)
```



Computing the covariance of S and Φ from the values of the indices

To double-check the recurrence, we have computed the values of $Cov_U(S_n\Phi_n)$, for $n = 3, \ldots, 8$, from the Sackin and cophenetic indices of all trees in the corresponding $\mathcal{BT}_n$.

First of all, we have considered all the phylogenetic trees in $\mathcal{BT}_n$ for $n = 3, \ldots, 8$ obtained with the package *phylonetwork* for *Python* in this section, then we have computed the Sackin index, `all.sackin.index`,and the cophenetic index, `all.cophen.index`, of all of them (correspondingly in Section 3.6 and Section 3.7).

Afterwards, we can compute the covariance of the indices for each n:

```
covariancesU = function(n){
  len = length(all.sackin.index[[n]])
  value = cov(all.sackin.index[[n]],all.cophen.index[[n]]*(len-1)/len)
  return(value)
}
real.cov.values = sapply(3:7,covariancesU)
```

Obtaining the following results:

n	3	4	5	6	7	8
$Cov_U(S_n, \Phi_n)$	0	0.32	1.9184	6.5805	17.0441	37.0899

agreeing with the figures given by our recurrence.

Now, since we know how to compute recurrently $Cov_U(S_n, \Phi_n)$, $\sigma^2_U(S_n)$ and $\sigma^2_U(\Phi_n)$, we can compute Pearson's correlation ρ of S and Φ under the uniform model for any desired $n \geq 4$, by means of

$$\rho_U(S_n, \Phi_n) = \frac{Cov_U(S_n, \Phi_n)}{\sigma_U(S_n) \cdot \sigma_U(\Phi_n)}.$$

Therefore

```
pearson.cor = function(n.max){
  return(cov.values[4:n.max]/sqrt(var.values.S[4:n.max]*var.values.Phi[4:n.max]))
}
```

For $n = 4, ..., 20$ the results are:

```
pearson.cor(20)
```

```
##  [1] 1.0000000 0.9968461 0.9944951 0.9926443 0.9911325 0.9898641 0.9887783
##  [8] 0.9878336 0.9870012 0.9862597 0.9855934 0.9849901 0.9844400 0.9839358
## [15] 0.9834711 0.9830410 0.9826413
```

3.8.1 The discriminative power of Φ

We have estimated the probability that a pair of trees $T_1, T_2 \in \mathcal{BT}_n$ have $I(T_1) = I(T_2)$, for $I = C, S, \Phi$. Notice that if T_1 and T_2 do have the same topology, then all these indices must be equal on them: therefore, any difference in these probabilities must be due to pairs of trees with different topology but having the same index.

To compute the balance indices we use the function `balance.indices` from the package *CollessLike* but modified to compute the clasical Colless index

```
balance.indices2 = function(tree){
  colless.coefficient = (log(0 + exp(1))+log(2 + exp(1)))/2
  values = balance.indices(tree)
  values[1] = values[1]/colless.coefficient
  return(values)
}
```

For every $n = 3, \ldots, 50$ we have chosen uniformly a set of N random pairs of trees in $\mathcal{BT}_n$ (for $n = 3, \ldots, 7$, we took $N = |\mathcal{BT}_n|$ and, for $n \geq 8$, we took $N = 3000$), and computed, for $I = C, S, \Phi$,

$$\widehat{p}_n(I) = \frac{\text{number of pairs } (T_1, T_2) \text{ with } n \text{ leaves such that } I(T_1) = I(T_2)}{N}.$$

The following functions compute this probabilities

```
are.tie = function(xx,yy) return(xx==yy)

exact.ties = function(){
  trees = list()
  all.indices = list()
  num.ties = c(0,0,0)
  prob.ties = list()
  for(n in 3:7){
    trees[[n]] = read.tree(file = paste("bintrees-n",n,".txt",sep=""))
    total.trees = length(trees[[n]])
    total.pairs = total.trees*(total.trees-1)/2
    all.indices[[n]] = matrix(sapply(trees[[n]],balance.indices2),ncol=3,byrow=T)
    num.ties[1]=sum(outer(all.indices[[n]][,1],all.indices[[n]][,1],are.tie))
    num.ties[2]=sum(outer(all.indices[[n]][,2],all.indices[[n]][,2],are.tie))
    num.ties[3]=sum(outer(all.indices[[n]][,3],all.indices[[n]][,3],are.tie))
    num.ties = (num.ties-total.trees)/2
    prob.ties[[n]] = num.ties/total.pairs
    print(paste("Ties for n =",n," : ",
                paste(c("p_C=","p_S=","p_Phi"),round(prob.ties[[n]],4),collapse=", "),sep=""))
  }
  return(prob.ties)
}

sim.ties.n = function(n,num.pairs.sim=3000){
  num.ties = c(0,0,0)
  for(i in 1:num.pairs.sim){
    t1 = rtree(n,rooted=TRUE)
    continue = TRUE
    while(continue){
      t2 = rtree(n,rooted=TRUE)
      continue = all.equal(t1,t2,use.length=FALSE,use.tip.label=FALSE)
    }
    t1.indices = balance.indices2(t1)
    t2.indices = balance.indices2(t2)
    num.ties = num.ties + (t1.indices==t2.indices)
  }
  print(paste("n =",n))
  print(paste("Ties :",num.ties))
  prob.ties = num.ties/num.pairs.sim
  print(paste("Prob :",round(prob.ties,4)))
```

```
  return(prob.ties)
}
```

Now, with this instructions we compute the probabilities for each $n = 3, \ldots, 50$

```
ties.1 = exact.ties()
ties.2 = lapply(8:50, sim.ties.n,num.pairs.sim=3000)
ties = matrix(c(unlist(ties.1),unlist(ties.2)),ncol=3,byrow=TRUE)
colnames(ties)=c("Colless","Sackin","Cophenetic")
rownames(ties)=3:50
```

For $n = 4, ..., 20$ the results are:

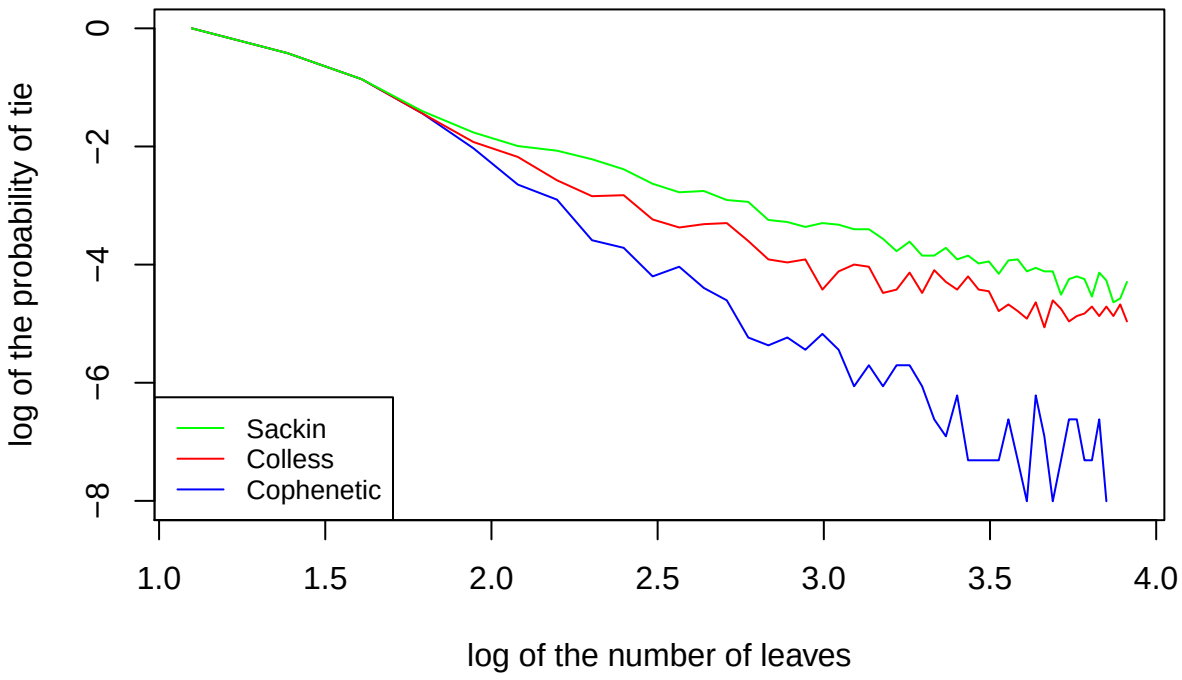
```
ties[1:18,]
```

	Colles	Sackin	Cophenetic
3	1.0000000	1.0000000	1.0000000
4	0.6571429	0.6571429	0.6571429
5	0.4230769	0.4230769	0.4230769
6	0.2372881	0.2463680	0.2372881
7	0.1459233	0.1716375	0.1312296
8	0.1133333	0.1363333	0.0710000
9	0.0763333	0.1260000	0.0550000
10	0.0583333	0.1090000	0.0276667
11	0.0593333	0.0920000	0.0243333
12	0.0393333	0.0720000	0.0150000
13	0.0343333	0.0623333	0.0176667
14	0.0363333	0.0636667	0.0123333
15	0.0370000	0.0546667	0.0100000
16	0.0273333	0.0530000	0.0053333
17	0.0200000	0.0390000	0.0046667
18	0.0190000	0.0376667	0.0053333
19	0.0200000	0.0346667	0.0043333
20	0.0120000	0.0370000	0.0056667

The whole table is available at "C3-table-ties.txt".

Next figure summarizes the results. It plots $\log(\hat{p}_n(I))$ for the three balance indices as a function of $\log(n)$. We can see that that Φ has the lowest relative frequency of ties.

```
plot(log(3:50),log(ties[,3]),type="l",
     xlab="log of the number of leaves",
     ylab="log of the probability of tie", col="blue")
lines(log(3:50),log(ties[,1]),col="red")
lines(log(3:50),log(ties[,2]),col="green")
legend("bottomleft", legend=c("Sackin","Colless","Cophenetic"),
       col=c("green","red", "blue"),lty=1, cex=0.8)
```



3.8.2 A test on TreeBASE

In this subsection we perform a simple test to check which of the models Yule or uniform is the one that better fits the TreeBASE using the total cophenetic index.

We have considerated all the trees from TreeBASE. The list is available in the *List of Trees* folder of the GitHub repository as a text file or as an `R` object.

```
# Option 1
tb.ape = read.tree(file = "./tb-newicks.txt")
# Option 2
load("./treeBASE-database.RData")
```

We have taken the numbers n of leaves for which the TreeBASE contains at least 20 binary phylogenetic trees with n leaves, and for each such n we have computed the mean of the total cophenetic indices of the corresponding binary trees.

```
bin.tb.ape=tb.ape[sapply(tb.ape,is.rooted)]
bin.tb.ape=bin.tb.ape[sapply(bin.tb.ape,is.binary)]
bin.tb.n = sapply(bin.tb.ape,Ntip)
leaves=as.numeric(names(which(table(bin.tb.n)>20)))
bin.tb.mean = c()
indices.tb = list()
for(k in leaves){
  trees = bin.tb.ape[bin.tb.n==k]
```

```
  indices.tb[[k]] = sapply(trees, cophen.index)
  value = mean( indices.tb[[k]] )
  bin.tb.mean = rbind(bin.tb.mean,c(k,value))
}
```

The results of this computations are available in "C3-table-tb-means.txt".

The following code computes $E_Y(\Phi_n)$ and $E_U(\Phi_n)$ for $n = 3, \ldots, 140$, using functions `EYPhi` and `EUPhi` from Section 3.3 and Section 3.4, respectively:

```
range.plot = 3:140
eyphi.values = sapply(range.plot, EYPhi)
euphi.values = sapply(range.plot, EUPhi)
```

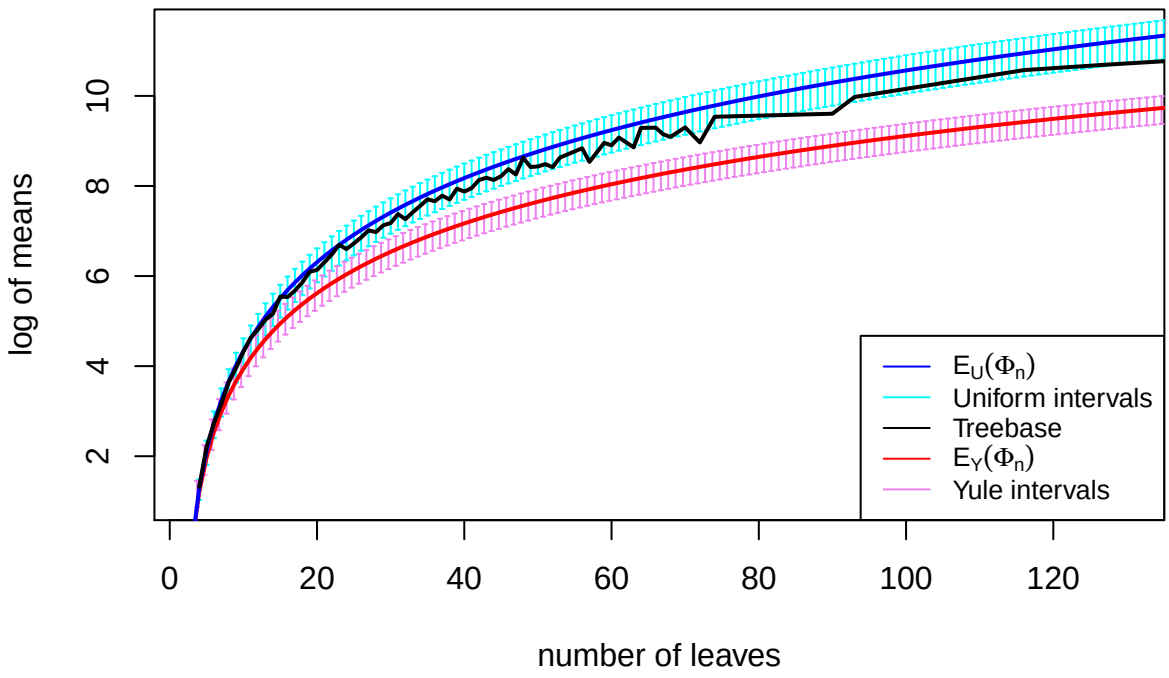
Using the computations of the variance of Φ_n under the uniform model (from this section) and the exact formula for $\sigma^2_Y(\Phi_n)$ (Formula 3.2) we can obtain the reference intervals for Φ_n that will be drawn in the figure.

```
harmonic2 = function(n){return(sum(1/((1:n)^2)))}
varYPhi = function(n){
  return((n^4-10*n^3+131*n^2-2*n)/12-4*n^2*harmonic2(n)-6*n*harmonic(n))
}
varYPhi.values = sapply(range.plot,varYPhi)
intY = cbind(range.plot,log(eyphi.values-1*sqrt(varYPhi.values)),
               log(eyphi.values+1*sqrt(varYPhi.values)))
intU = cbind(range.plot,log(euphi.values-1*sqrt(var.values.Phi[range.plot])),
               log(euphi.values+1*sqrt(var.values.Phi[range.plot])))

draw.intervals = function(range.plot,int.yule,int.uniform,delta=0){
  epsilon = 0.3
  for(i in range.plot){
    lines(c(i ,i ),int.uniform[i-2,2:3],col="cyan")
    lines(c(i-epsilon,i+epsilon),rep(int.uniform[i-2,2],2),col="cyan")
    lines(c(i-epsilon,i+epsilon),rep(int.uniform[i-2,3],2),col="cyan")
    lines(c(i ,i )-delta,int.yule[i-2,2:3],col="violet")
    lines(c(i-epsilon,i+epsilon)-delta,rep(int.yule[i-2,2],2),col="violet")
    lines(c(i-epsilon,i+epsilon)-delta,rep(int.yule[i-2,3],2),col="violet")
  }
}
```

Next figure shows the plot the log of these means as a function of n. We have added the curves of the log of the expected values of Φ_n under the Yule distribution (lower red curve) and under the uniform distribution (upper blue curve), again as a function of n, and the logarithms of the corresponding reference intervals for Φ_n (vertical segments).

```
plot(NULL,NULL,col="blue",xlab="number of leaves",
      ylab="log of means",xlim=c(3,130),ylim=c(1,11.5),
     type="l",lwd=2)
draw.intervals(range.plot,intY,intU,delta=0.3)
lines(range.plot,log(eyphi.values),col="red",lwd=2)
lines(range.plot,log(euphi.values),col="blue",lwd=2)
lines(bin.tb.mean[,1],log(bin.tb.mean[,2]),type="l",lwd=2)
legend("bottomright", legend=c(expression(E[U]*(Phi[n])),"Uniform intervals","Treebase",
        expression(E[Y]*(Phi[n])),"Yule intervals"),col=c( "blue","cyan","black","red",
        "violet"),lty=1,cex=0.8)
```



This figure shows that the total cophenetic indices of the binary phylogenetic trees in TreeBASE seem to be better explained by the uniform model than by the Yule model.