Evaluation 1: get test properties

Janne Pott

13/03/2023

Summary of Simulations

What did I simulate?

Preparation: create all possible quadruples and all possible 4-way partitions (4WP) for $n = 6, ..., 10$ taxa

Simulation: for n taxa, there are $K_n = \binom{n}{4}$ possible quadruples. For each $k \in \{1, ..., K_n\}$, there are $M_k = \binom{K_n}{k}$ combinations of quadruples as input and 10,000 of these M_k combinations are used in the simulation. Each combination is tested for phylogenetic decisiveness according to 4WPP (truth) and fixing tacon traceability (test).

Here: Summary of the simulation results (barplots, negative predictive value, power).

Upper bound: it was already shown that given an input of $\binom{n}{4} - (n-4)$ quadruples, all sets are phylogenetic decisive:

- $n = 6$: $15 - 2 = 13$
- $n = 7$: $35 - 3 = 32$
- $n = 8$: $70 - 4 = 66$
- $n = 9$: $126 - 5 = 121$
- $n = 10$: $210 - 6 = 204$

Lower bound: there is no simple formula, but I use here my lower bound from my master thesis (set size that covers all triples or that covers all tuples adequately):

- $n = 6$: 6
- $n = 7$: 11
- $n = 8$: 14
- $n = 9$: 24
- $n = 10$: 30

Initialize

I use a file names *SourceFile.R* that contains all relevant R packages and user-/server-specific path to the R library. If using this code, you must make all the necessary changes within the template source file.

```
rm(list = ls())
time0<-Sys.time()

source("../SourceFile.R")
```

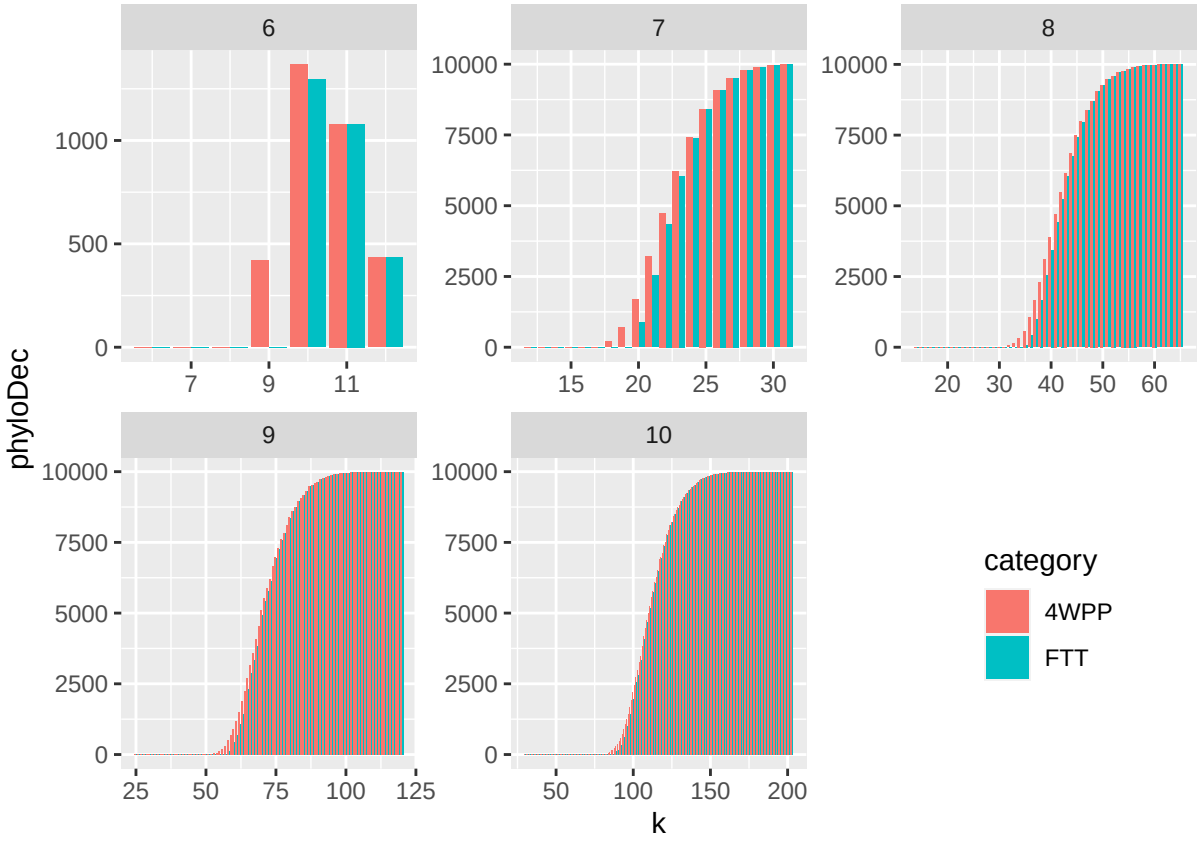
```
source("../helperFunctions/TestHelpRFunction.R")
source("../helperFunctions/ShiftLegendBarplot.R")

x_lowerBound = c()
x_upperBound = c()

for(i in 6:10){
  #i=7
  y = i %% 6

  if(y==0){
    min_quad = 0.25 * choose(i,3) + i/6
  }else if(y %in% c(2,4)){
    min_quad = 0.25 * choose(i,3)
  }else{
    min_quad = (1/6) * ((i-1)/2) * choose(i,2) + i/12
    min_quad = ceiling(min_quad)
  }
  x_lowerBound = c(x_lowerBound,min_quad)
  max_quad = choose(i,4) - (i-4)
  x_upperBound = c(x_upperBound,max_quad)
}
```

Get data

```
load("../results/02_SimulationResults_n06.RData")
load("../results/02_SimulationResults_n07.RData")
load("../results/02_SimulationResults_n08.RData")
load("../results/02_SimulationResults_n09.RData")
load("../results/02_SimulationResults_n10.RData")

SimulationResults_n06[,n := 6]
SimulationResults_n07[,n := 7]
SimulationResults_n08[,n := 8]
SimulationResults_n09[,n := 9]
SimulationResults_n10[,n := 10]

sim_n6 = SimulationResults_n06[k>=x_lowerBound[1] & k<=x_upperBound[1]]
sim_n7 = SimulationResults_n07[k>=x_lowerBound[2] & k<=x_upperBound[2]]
sim_n8 = SimulationResults_n08[k>=x_lowerBound[3] & k<=x_upperBound[3]]
sim_n9 = SimulationResults_n09[k>=x_lowerBound[4] & k<=x_upperBound[4]]
sim_n10 = SimulationResults_n10[k>=x_lowerBound[5] & k<=x_upperBound[5]]

sim = rbind(sim_n6,sim_n7,sim_n8,sim_n9,sim_n10)
table(sim$n)
#>
#>   6   7   8   9  10
#>   7  20  52  96 174
```

Bar plots

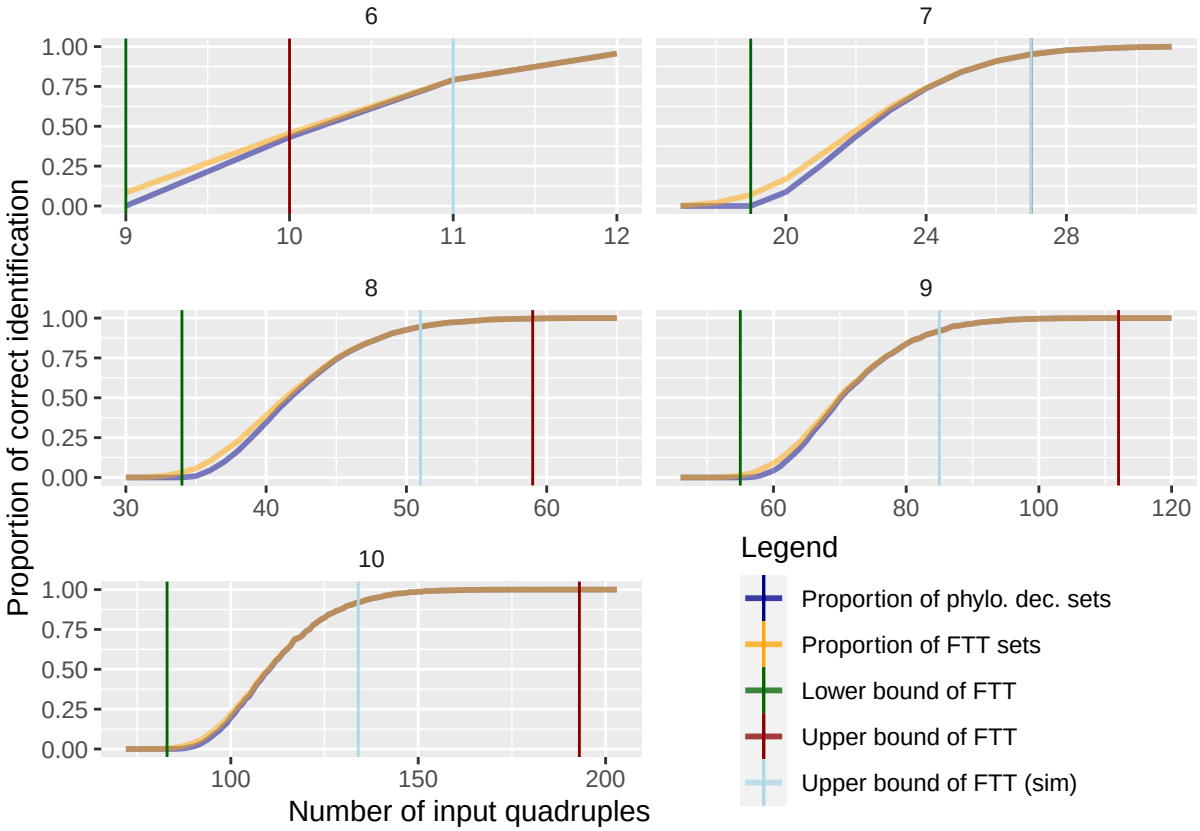
```
dumTab1 = foreach(i = 6:10)%do%{
  # i=6
  mySim = copy(sim)
  mySim = mySim[n == i,]
  x1 = dim(mySim)[1]

  PlotData = data.table(n = i,
                        k = rep(mySim$k,2),
                        phyloDec = c(mySim$NR_FTT,mySim$NR_PhyloDec),
                        category = c(rep("FTT",x1),rep("4WPP",x1)))
  ggplot(data=PlotData,aes(fill=category, y=phyloDec, x=k)) +
    geom_bar(position="dodge", stat="identity")

  PlotData
}
BarPlotData = rbindlist(dumTab1)

p1 = ggplot(data=BarPlotData,aes(fill=category, y=phyloDec, x=k)) +
  facet_wrap(vars(n), nrow = 2, scales = "free") +
  geom_bar(position="dodge", stat="identity")
grid.draw(ShiftLegendBarplot(p1))
```



Line plots

```
dumTab2 = foreach(i=6:10)%do%{
  #i=6
  dum = choose(i,4)
  dum2 = copy(sim)
  dum2 = dum2[n==i]
  dum2[,prop_4WPP := NR_PhyloDec/(NR_PhyloDec + NR_NotPhyloDec)]
  dum2[,prop_FTT := NR_FTT/(NR_PhyloDec + NR_NotPhyloDec)]
  #dum2 = dum2[!is.na(posRate)]
  dum2[,k2 := k/dum]
  dum2
}
LinePlotData2 = rbindlist(dumTab2)

dumTab3 = foreach(i=6:10)%do%{
  #i=6
  dum = choose(i,4)
  x = choose(i-1,3)-1
  y = x/dum
  a = choose(i,4) - (3*i-13)
  b = a/dum

  dum2 = copy(sim)
  dum2 = dum2[!is.na(posRate)]
  dum2 = dum2[n==i]
  dum2 = dum2[posRate==1,]
  v = min(dum2$k)
  w = v/dum
  res = data.table(n=i, vline1_abs = x, vline2_abs = a, vline3_abs = v,
                   vline1_rel = y, vline2_rel = b, vline3_rel = w)
}
data_vline = rbindlist(dumTab3)

p2 <- ggplot(LinePlotData2[!is.na(posRate),]) +
  geom_line(aes(x=k, y=prop_FTT, col=as.factor(1)),linewidth=1,alpha=0.5) +
  geom_line(aes(x=k, y=prop_4WPP,col=as.factor(2)),linewidth=1,alpha=0.5)+
  geom_vline(data = data_vline,aes(xintercept = vline1_abs,col =as.factor(3))) +
  geom_vline(data = data_vline,aes(xintercept = vline2_abs,col =as.factor(4))) +
  geom_vline(data = data_vline,aes(xintercept = vline3_abs,col =as.factor(5))) +
  facet_wrap(~ n, nrow = 3, scales = "free_x",strip.position = "top") +
  theme(strip.background = element_blank(),
        strip.placement = "outside")+
  labs(x="Number of input quadruples                                                    ",
       y = "Proportion of correct identification",
       color = "Legend") +
  scale_color_manual(values = c("darkblue","orange","darkgreen","darkred","lightblue"),
                     labels = c("Proportion of phylo. dec. sets",
                                "Proportion of FTT sets",
                                "Lower bound of FTT",
                                "Upper bound of FTT",
                                "Upper bound of FTT (sim)"))
#,"upper bound","upper bound (simulation)"
```

```
grid.draw(ShiftLegendBarplot(p2))
```



Contingency tables (overall)

- Prevalence = actual positive (P) / all (N+P)
- PPV = Positive Predictive Value = true positives (TP) / predicted positives (PP) = conditional probability P(true state positive | prediction positive)
- NPV = Negative Predictive Value = true negatives (TN) / predicted negatives = conditional probability P(true state negative | prediction negative)
- TPR = True Positive Rate = sensitivity = power = TP / P
- TNR = True Negative Rate = specificity = TN / N
- see also Wikipedia

```
dumTab2 = foreach(i = 6:10)%do%{
  # i=6
  mySim = copy(sim)
  mySim = mySim[n == i,]
  stats = TestHelpRFunction(P = sum(mySim$NR_PhyloDec),
                            N = sum(mySim$NR_NotPhyloDec),
                            PP = sum(mySim$NR_FTT))
  stats[,n:=i]
  stats
```

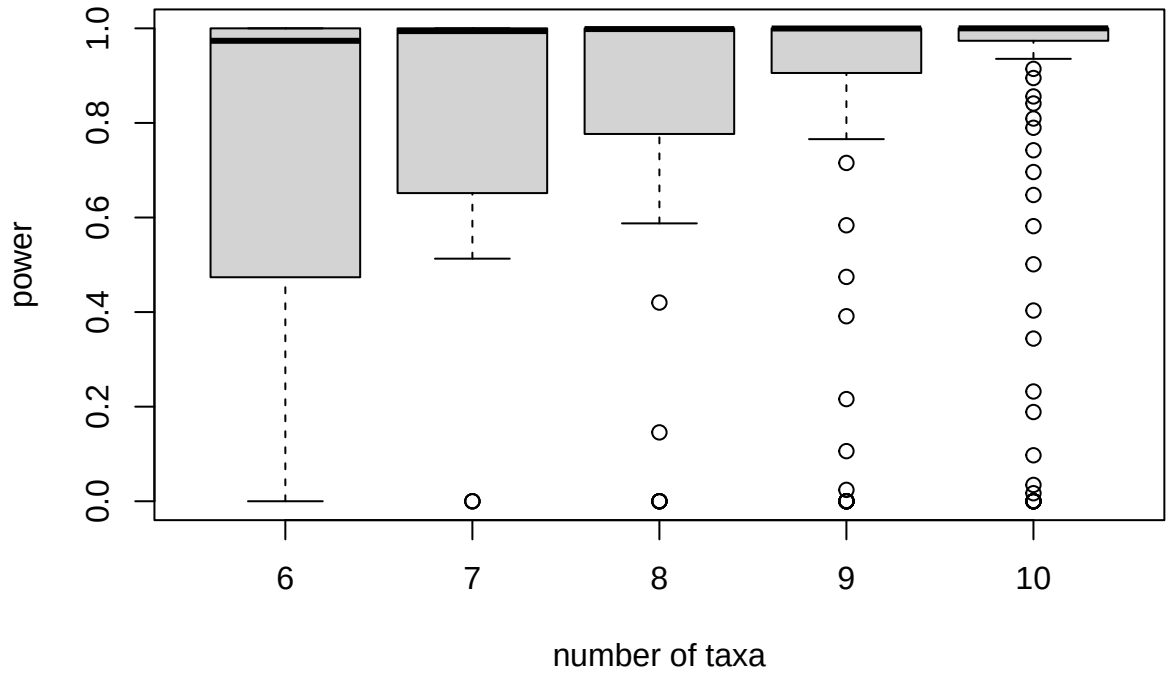
```
}
myStats = rbindlist(dumTab2)
myStats
#>    Prevalence PPV       NPV       TPR TNR  n
#> 1:  0.1192290   1 0.9802346 0.8510445   1  6
#> 2:  0.4541400   1 0.9728737 0.9664861   1  7
#> 3:  0.4524173   1 0.9833916 0.9795585   1  8
#> 4:  0.5146885   1 0.9877019 0.9882595   1  9
#> 5:  0.5270874   1 0.9934300 0.9940663   1 10
```

Fazit 1: NPV is about 98%, i.e. there is a 98% probability that a negative set is really not phylogenically decisive

Fazit 2: The power of the algorithm (TPR) is also about 98%.

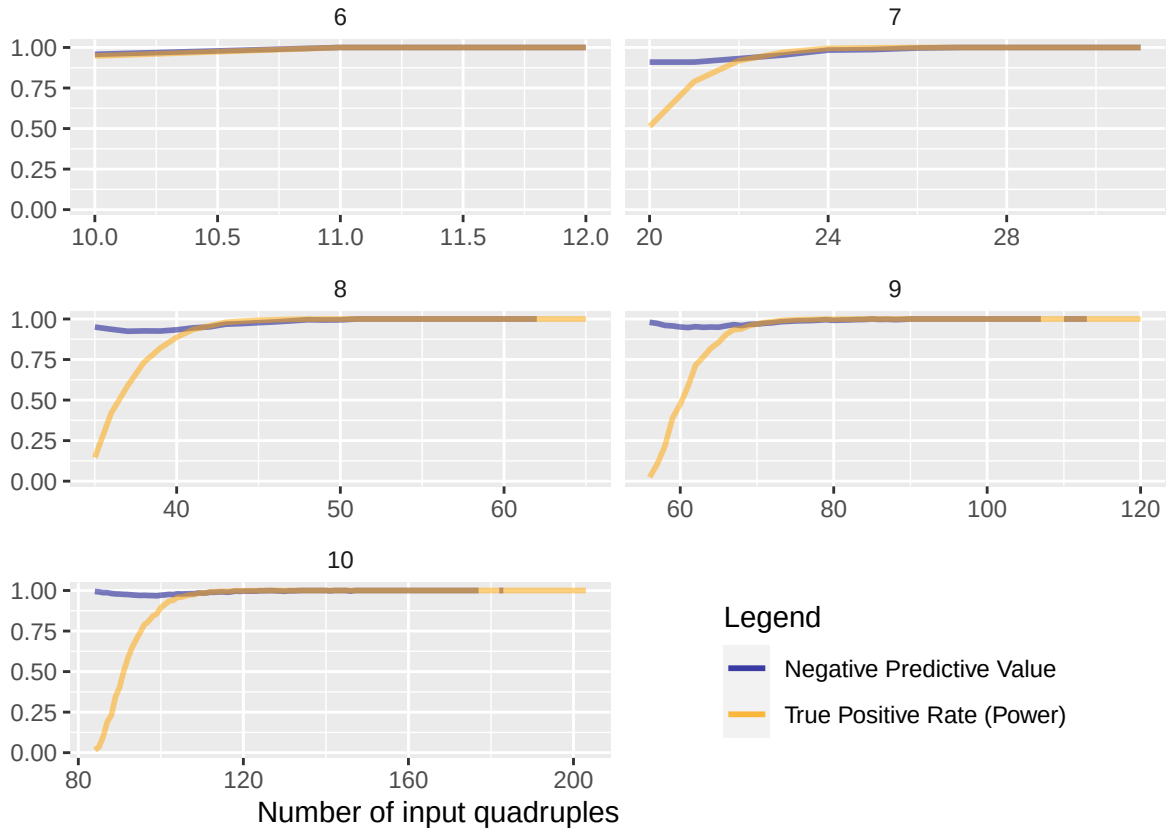
Contingency tables (per k)

```
dumTab3 = foreach(i = 6:10)%do%{
  # i=6
  mySim = copy(sim)
  mySim = mySim[n == i,]

  dumTab4 = foreach(k = 1:dim(mySim)[1])%do%{
    # k=1
    myRow = copy(mySim)
    myRow = myRow[k,]

    stats_k = TestHelpRFunction(P = myRow$NR_PhyloDec,
                                N = myRow$NR_NotPhyloDec,
                                PP = myRow$NR_FTT)
    stats_k[,n :=i]
    stats_k[,k := myRow$k]
    stats_k
  }
  myStats_k = rbindlist(dumTab4)
  myStats_k
}
myStats_k = rbindlist(dumTab3)

filt_TPR = !is.na(myStats_k$TPR)
filt_NPV = !is.na(myStats_k$NPV)

boxplot(myStats_k$TPR[filt_TPR] ~ myStats_k$n[filt_TPR],
        xlab = "number of taxa",
        ylab = "power")
```




```
boxplot(myStats_k$NPV[filt_NPV] ~ myStats_k$n[filt_NPV],
        xlab = "number of taxa",
        ylab = "NPV")

tab6 = myStats_k[n==6 & filt_TPR,summary(TPR)]
tab7 = myStats_k[n==7 & filt_TPR,summary(TPR)]
tab8 = myStats_k[n==8 & filt_TPR,summary(TPR)]
tab9 = myStats_k[n==9 & filt_TPR,summary(TPR)]
tab10 = myStats_k[n==10 & filt_TPR,summary(TPR)]
tab_TPR = rbind(tab6,tab7,tab8,tab9,tab10)

tab6 = myStats_k[n==6 & filt_NPV,summary(NPV)]
tab7 = myStats_k[n==7 & filt_NPV,summary(NPV)]
tab8 = myStats_k[n==8 & filt_NPV,summary(NPV)]
tab9 = myStats_k[n==9 & filt_NPV,summary(NPV)]
tab10 = myStats_k[n==10 & filt_NPV,summary(NPV)]
tab_NPV = rbind(tab6,tab7,tab8,tab9,tab10)

tabs = as.data.table(rbind(tab_NPV,tab_TPR))
tabs[,n:=rep(c(6:10),2)]
tabs[,Parameter:=c(rep("NPV",5),rep("TPR",5))]
tabs = tabs[,c(8,7,1:6)]
tabs
#>     Parameter  n      Min.   1st Qu.    Median      Mean 3rd Qu. Max.
#>  1:       NPV  6 0.9160839 0.9789104 1.0000000 0.9819864       1    1
#>  2:       NPV  7 0.9096485 0.9737353 0.9992500 0.9790416       1    1
```

```
#>  3:       NPV  8 0.9243381 0.9820736 1.0000000 0.9859275       1    1
#>  4:       NPV  9 0.9472723 0.9897121 1.0000000 0.9907297       1    1
#>  5:       NPV 10 0.9671249 0.9958745 1.0000000 0.9951519       1    1
#>  6:       TPR  6 0.0000000 0.7105263 0.9736842 0.7368421       1    1
#>  7:       TPR  7 0.0000000 0.6516322 0.9941978 0.7455668       1    1
#>  8:       TPR  8 0.0000000 0.7986817 0.9985225 0.7898741       1    1
#>  9:       TPR  9 0.0000000 0.9056010 0.9996693 0.8162004       1    1
#> 10:       TPR 10 0.0000000 0.9743057 1.0000000 0.8687601       1    1

# Plotting
LinePlotData3 = copy(myStats_k)
LinePlotData3 = LinePlotData3[!is.na(PPV)]

p3 <- ggplot(LinePlotData3) +
  geom_line(aes(x=k, y=NPV, col=as.factor(1)),linewidth=1,alpha=0.5) +
  geom_line(aes(x=k, y=TPR,col=as.factor(2)),linewidth=1,alpha=0.5)+
  facet_wrap(~ n, nrow = 3, scales = "free_x",strip.position = "top") +
  theme(strip.background = element_blank(),
        strip.placement = "outside")+
  labs(x="Number of input quadruples                                                    ",
       y = "",
       color = "Legend") +
  scale_color_manual(values = c("darkblue","orange"),
                     labels = c("Negative Predictive Value",
                                "True Positive Rate (Power)"))
#,"upper bound","upper bound (simulation)"


grid.draw(ShiftLegendBarplot(p3))
#> Warning: Removed 20 rows containing missing values (`geom_line()`).
```




```
test = copy(LinePlotData3)[TPR>0.9,]
test = test[!duplicated(n)]
test[,allQ := choose(n,4)]
test[,prop := k/allQ]
test
#>    Prevalence PPV       NPV       TPR TNR  n   k allQ      prop
#> 1:  0.4555445   1 0.9578207 0.9473684   1  6  10   15 0.6666667
#> 2:  0.4742000   1 0.9314438 0.9183889   1  7  22   35 0.6285714
#> 3:  0.4714000   1 0.9452790 0.9350870   1  8  41   70 0.5857143
#> 4:  0.3178000   1 0.9578770 0.9056010   1  9  66  126 0.5238095
#> 5:  0.2450000   1 0.9729381 0.9142857   1 10 101  210 0.4809524
```

Some total numbers

```
sumSim = sum(sim$NR_NotPhyloDec,sim$NR_PhyloDec)
sumPhyloDec = sum(sim$NR_PhyloDec)
sumFTT = sum(sim$NR_FTT)

sumPhyloDec/sumSim
#> [1] 0.504864

sumFTT/sumPhyloDec
#> [1] 0.9887465
```

Session Info

```
sessionInfo()
#> R version 4.2.2 (2022-10-31 ucrt)
#> Platform: x86_64-w64-mingw32/x64 (64-bit)
#> Running under: Windows 10 x64 (build 22621)
#>
#> Matrix products: default
#>
#> locale:
#> [1] LC_COLLATE=German_Germany.utf8  LC_CTYPE=German_Germany.utf8
#> [3] LC_MONETARY=German_Germany.utf8 LC_NUMERIC=C
#> [5] LC_TIME=German_Germany.utf8
#>
#> attached base packages:
#> [1] grid      stats     graphics  grDevices utils     datasets  methods
#> [8] base
#>
#> other attached packages:
#> [1] cowplot_1.1.1           gtable_0.3.1            ggplot2_3.4.1
#> [4] FixingTaxonTraceR_0.0.1 foreach_1.5.2           data.table_1.14.8
#>
#> loaded via a namespace (and not attached):
#>  [1] rstudioapi_0.14  knitr_1.42       magrittr_2.0.3   munsell_0.5.0
#>  [5] colorspace_2.1-0 R6_2.5.1         rlang_1.0.6      fastmap_1.1.1
#>  [9] fansi_1.0.4      highr_0.10       tools_4.2.2      xfun_0.37
#> [13] utf8_1.2.3       cli_3.6.0        withr_2.5.0      htmltools_0.5.4
#> [17] iterators_1.0.14 yaml_2.3.7       digest_0.6.31    tibble_3.2.0
#> [21] lifecycle_1.0.3  farver_2.1.1     vctrs_0.5.2      codetools_0.2-18
#> [25] glue_1.6.2       evaluate_0.20    rmarkdown_2.20   labeling_0.4.2
#> [29] compiler_4.2.2   pillar_1.8.1     scales_1.2.1     pkgconfig_2.0.3
message("\nTOTAL TIME : " ,round(difftime(Sys.time(),time0,units = "mins"),3)," minutes")
#>
#> TOTAL TIME : 0.091 minutes
```