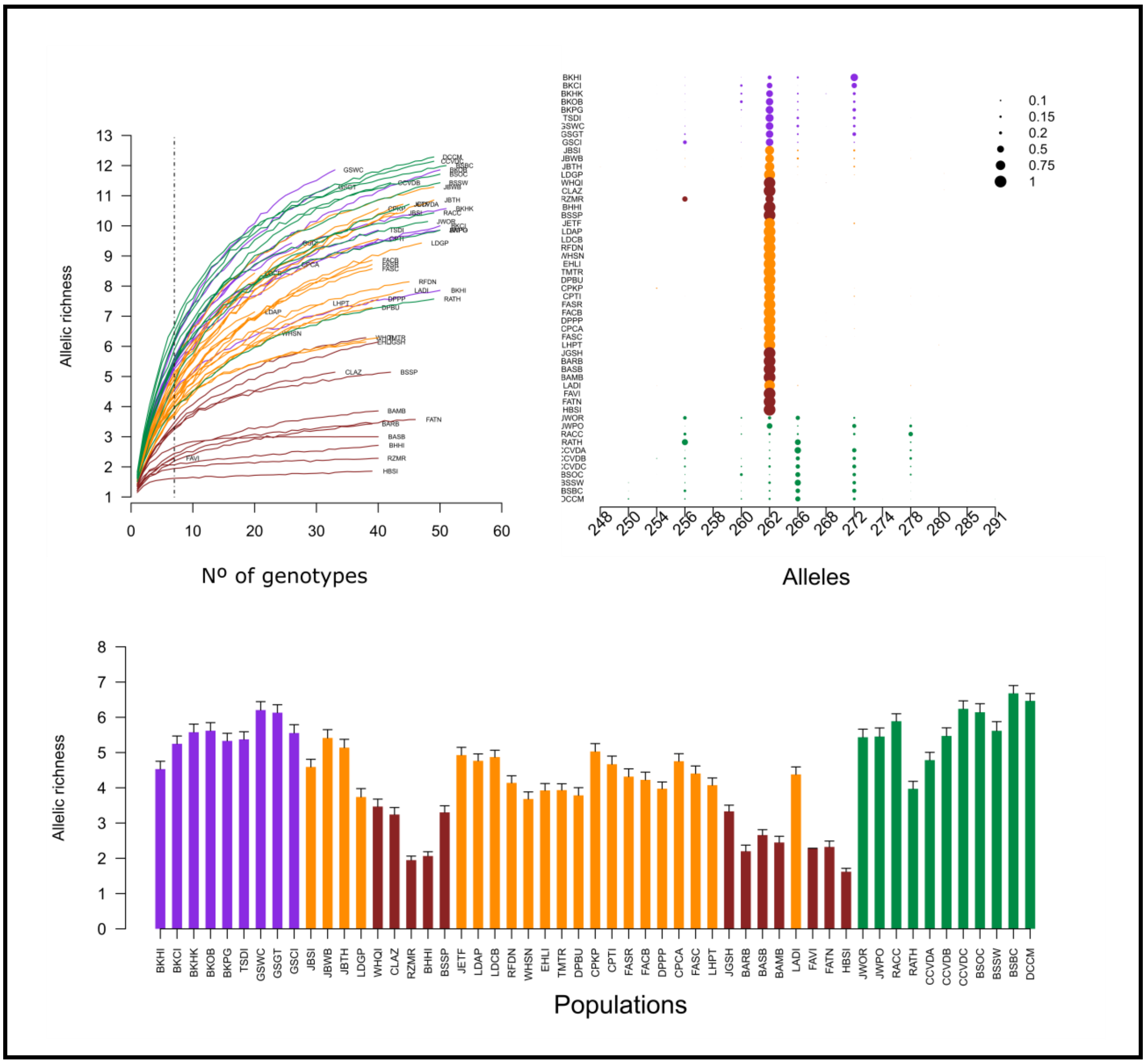
StandArich_v2.00

An R package to estimate population allelic richness using standardized sample size, plot allelic richness bar plot, allelic richness rarefaction curves and allele frequency distributions.

Author: Filipe Alberto, Dept. of Biological Sciences, University Wisconsin-Milwaukee

Date: 12-2021



What is StandArich2 for?

The package purpose is to standardize population sample size before comparing allelic richness (the mean number of alleles per locus) among different populations. The problem of unequal sample size is typical in clonal species where *G*, the number of different genotypes or genetic individuals, is the relevant statistic to standardize.

Independently of the sample design used for clonal organisms *G* is unpredictable at the time of sampling, even when the number of sample units is kept constant across populations. Therefore, for clonal species the input data file should contain a single copy of each multilocus genotype (*MLG*) present for each population. Usually, the *G* counts among populations for clonal species data sets vary much, thus a standardization of *G* is needed to compare meaningful estimates of allelic richness.

For non-clonal species a standardization is also necessary when *N* is variable across populations.

Graphical tools are also available to; 1) produce a bar plot of standardized allelic richness for *n* equal to the smallest sample size across all populations, 2) plot allelic richness rarefaction curves, i.e., the change in allelic richness as a function of *n*, and 2) plot populations' allele distributions and frequencies.

Changes with version 2

To obtain a standardized measure of allelic richness a single function call to *rgenotypes.arich* is now sufficient. The function *standarich* is no longer necessary. Other functionalities to plots are now available, like loci names in allele frequency plots and optional population names in sample size allelic richness rarefaction curves.

Data file

The text tab delimited file to use as input to StandArich2 is very simple: Lines correspond to individual genotypes. The 1st column has the population codes, the 2nd column contains the codes for individual within populations. The next columns contain the allele codes, in a two columns per locus format, in integers. Alleles are coded using three or two digits (but be consistent!). Missing data is represented by the value 999. Don't use 99 even if you use 2 digits to code alleles. There is no header in this file.

The package contains an example data (Exdata) file with three populations of the seagrass *Cymodocea nodosa*. A reduced version of this file (fewer genotypes per population) is shown below to exemplify the format.

Maurita	1	181	181	222	222	175	177
Maurita	2	181	181	222	222	177	177
Maurita	3	181	181	222	222	177	177
Pblanca	1	181	181	224	230	175	175
Pblanca	2	181	181	230	230	175	175
Pblanca	3	181	203	224	224	175	175
Pblanca	4	181	203	224	230	175	175
Pblanca	5	181	203	224	230	175	175
…							

To load such a file into R environment you can use the read.table( ) function; in the R console type:

```
test=read.table("myfile.txt", header=FALSE)
```

To load the example data file instead

```
data(Exdata)
```

Standardizing allelic richness estimates to similar *n* across populations

The function *rgenotypes.arich* runs a multiple random reduction (Leberg et al. 2002) of the number of individuals/genets in each population, by generating random subsamples for each population of varying size from *n* = 1 to *n* = *N*, *N* being the total number of different genotypes observed in each population. Each subsample of a given size *n* is a random replicate of the sample and there are *n.replicates* for each *n*. Below the general function call.

```
rtest=rgenotypes.arich( Popdata=test, n.replicates=100, loc.labels)
```

The first argument is the R object with your data (e.g., *test* following the example above reading data into R), the second *n.replicates* defines how many replicates per *n* permutation you want to use to resample each population. One hundred should be more than sufficient (this will affect how smooth the rarefaction curves are, see below). Finally, loc.labels takes a vector of class character containing the loci names. There is no default for *loc.labels* you need to provide a vector with loci names. If you don't know the loci names you can use the code below to quickly create a vector with "L.1","L.2"…."L.x" loci names.

```
loc.labels= paste("L", 1:((ncol(Exdata)/2)-2), sep=".")
```

or if you know how many loci are in the file (8 in the example data)

```
loc.labels= paste("L", 1:8, sep=".")
```

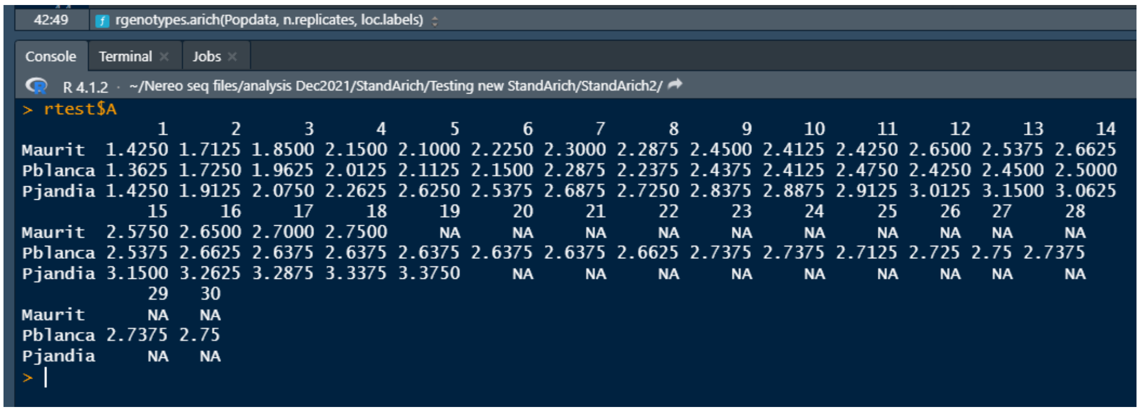
To run *rgenotypes.arich* using the example data, without knowing the loci names:

```
data(Exdata)  #loads the example data set

rtest=rgenotypes.arich(Exdata, 10, loc.labels = paste("L", 1:8, sep="."))
```

The function returns a list with three elements *R*, *A* and *A.sd*. *R* is a large table containing information for all permutations and replicates for each n level and population. The table contains number of alleles observed at each loci, the mean number of alleles observed across loci (i.e., allelic richness per permutation of size *n*), and the standard deviation of the number of alleles observed across loci. *A* is a table with the mean allelic richness across all permutations for each level of *n* for each population. Population data is organized by rows and different *n* values by columns (see figure below). Finally, *A.sd* is a table with the standard deviation across all permutations for each level of *n* for each population. The table format is identical to *A*, populations by rows and *n* by columns.

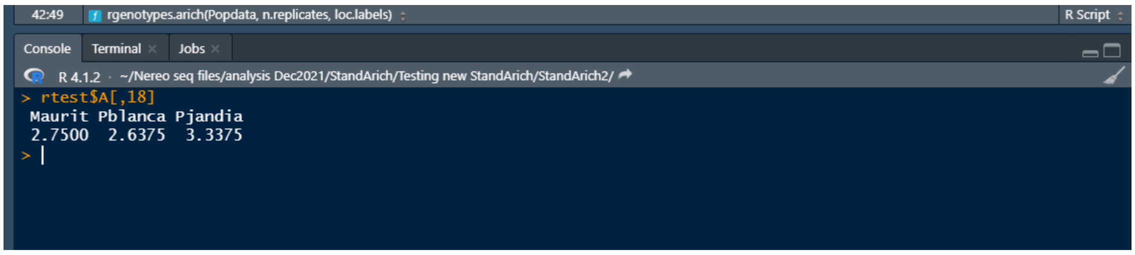
rtest$A


42:49
rgenotypes.arich(Popdata, n.replicates, loc.labels)
Console
Terminal
Jobs
R 4.1.2 · ~/Nereo seq files/analysis Dec2021/StandArich/Testing new StandArich/StandArich2/

```
> rtest$A
             1      2      3      4      5      6      7      8      9     10     11     12     13     14
Maurit  1.4250 1.7125 1.8500 2.1500 2.1000 2.2250 2.3000 2.2875 2.4500 2.4125 2.4250 2.6500 2.5375 2.6625
Pblanca 1.3625 1.7250 1.9625 2.0125 2.1125 2.1500 2.2875 2.2375 2.4375 2.4125 2.4750 2.4250 2.4500 2.5000
Pjandia 1.4250 1.9125 2.0750 2.2625 2.6250 2.5375 2.6875 2.7250 2.8375 2.8875 2.9125 3.0125 3.1500 3.0625
            15     16     17     18     19     20     21     22     23     24     25    26   27     28
Maurit  2.5750 2.6500 2.7000 2.7500     NA     NA     NA     NA     NA     NA     NA    NA   NA     NA
Pblanca 2.5375 2.6625 2.6375 2.6375 2.6375 2.6375 2.6375 2.6625 2.7375 2.7375 2.7125 2.725 2.75 2.7375
Pjandia 3.1500 3.2625 3.2875 3.3375 3.3750     NA     NA     NA     NA     NA     NA    NA   NA     NA
            29   30
Maurit      NA   NA
Pblanca 2.7375 2.75
Pjandia     NA   NA
> |
```

NA values are produced for allelic richness when the corresponding *n* value exceeds the population size *N* for that population. The largest *n* (column) without NA values for any population is also the sample size for the population with the smallest sample size. In this example that *n* value is 18. We should use this *n* value to report on standardized allelic richness. Therefore, to extract standardized allelic richness for *n*=18, we index the 18th column of *rtest$A*.

rtest$A[,18]


42:49
rgenotypes.arich(Popdata, n.replicates, loc.labels)
R Script
Console
Terminal
Jobs
R 4.1.2 · ~/Nereo seq files/analysis Dec2021/StandArich/Testing new StandArich/StandArich2/

```
> rtest$A[,18]
 Maurit Pblanca Pjandia
 2.7500  2.6375  3.3375
> |
```

Use the same idea to extract the standard deviation of allelic richness estimates from *rtest$A.sd[,18]*.

Plotting functions

<u>A bar plot of standardized allelic richness</u>

The function uses the list produced by rgenotypes.arich to print a bar plot of standardized allelic richness. Therefore, its main argument *Rlist* takes the object *rtest* created above with our example.

AR.barplot(Rlist, lwd=10, col="black", b.bar.x=5, popsize=0.7, error.bar=FALSE)

Other arguments to the function:

lwd - the width of each bar. The plot uses type="h" to produce the bars so this is the *lwd* argument used in R to set line width.

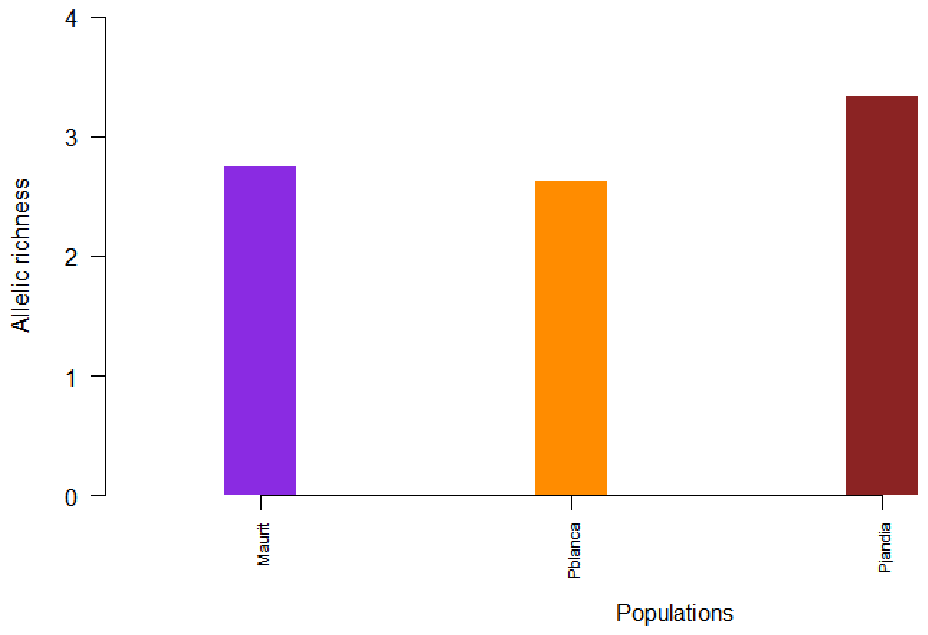
col – a vector of color names for each population. Defaults to black.

b.bar.x - the space in x plotting units between consecutive bars. Defaults to 5.

popsize - the size for pop names printed in the x axis, controled by par(cex=popsize). Defaults to 0.7.

error.bar - a logical value to print 95% confidence intervals. Defaults to FALSE.

```
cores<-c("blueviolet" ,  "darkorange"  , "brown4")
AR.barplot(rtest, col=cores, lwd=50 ,b.bar.x = 1)
```



Allelic richness rarefaction curves

The function *allele.genotype.plot* plots allelic richness rarefaction curves for each population, i.e., the change in allelic richness as a function of resampled sample size *n*. As such, its main argument is the table produced by *rgenotypes.arich* stored in the list slot *R*. There are other graphical arguments to control the plot.

```
allele.genotype.plot(results=rtest$R, g=18, xmin=0, xmax=50, xmark=10, main="",
    lwd=1, lty=1,lcol="black", print.pop=FALSE, pop.text.size=1)
```

Other arguments to the function are:

g - sets the *n* value to plot a vertical line over the plot to intersect at *n*. This helps visualize the standardized allelic richness values for a given *n*. Defaults to zero.

xmin - sets the lower limit of the x axis tick marks and dimensions. Defaults to zero.

xmax - sets the higher limit of the x axis tick marks and dimensions. Defaults to fifty.

xmark - sets the interval of tick marks for the x axis. Following the code for *seq(xmin,xmax,xmark).*

main - string to feed the main argument of plot.

lwd - line width for the lines in the plot. Can take a vector.

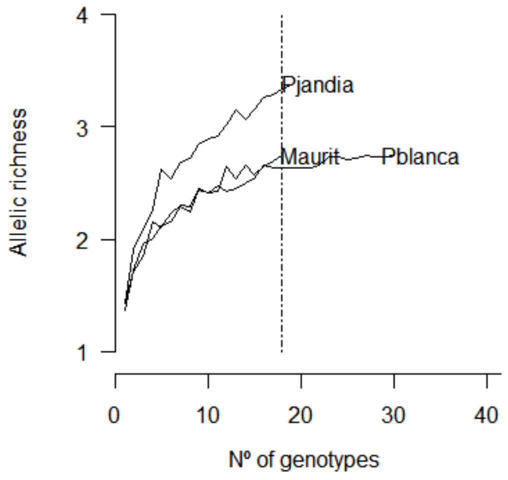
lty - line type for the lines in the plot. Can take a vector.

lcol - line colour. Can take a vector.

print.pop – takes a logical to decide if populations names should be printed in front of their corresponding lines. Defaults to FALSE.

pop.text.size - controls the size of population names text when print.pop=T. Defaults to 0.5.

The figure below was produced with the code above. To get smoother lines increase the number of replicates used in *rgenotypes.arich* when creating the *rtest$R* table used here.



Plotting populations’ allele distributions and frequencies

The function *plot.allele.freq* produces a much improved visual representation of allele frequencies when compared with a numerical table. If the input data contains populations ordered geographically, or in other meaningful order, genetic structure can be immediately visualized (when present).

```
plot.allele.freq(data=Exdata, LocusNames=paste("L", 1:8,  sep="."), Plogbase=20,
                textlegend=0.8, freqlegend=c(0.1,0.15,0.2,0.5,0.75,1), psize=10,
                poptext=1,allelesize=1, xaxispos=0,yplotlim=-3, alleleangle=45,
                pdfout=FALSE, pdfname="out.pdf")
```

The function returns as many plot windows as there are loci in your data file. Each plot represents a table of allele frequencies for each locus where the actual values are represented by dots of varying diameter. Allele codes are indicated on the x axis and population names on the y axis. You can decide to save the plots to a pdf file instead of visualizing them in R, using argument *pdfout* (NOTE that currently this is not working on RStudio).

The main argument of this function, *data*, takes the object with your data file (e.g., *Exdata*).

Other arguments to the function are:

LocusNames - a character vector with the names of loci

Plogbase - used to control the points cex argument to plot allele frequencies as points, as in cex=log(allelefreq+1, base=Plogbase)*psize). Defaults to 20.

textlegend - controls the cex argument for the legend text size.

freqlegend - a numeric vector with the allele frequencies used in the legend.

poptext - controls the cex argument for the font size of population names in the y-axis. Defaults to 0.5.

allelesize - controls the cex argument for the font size of allele names in the x-axis.

xaxispos - the pos argument for horizontal axis in the plot, see help(axis).

yplotlim - the ylim argument of the plot, see help(plot.default).

alleleangle - text rotation for allele names, in degrees. Defaults to 45.

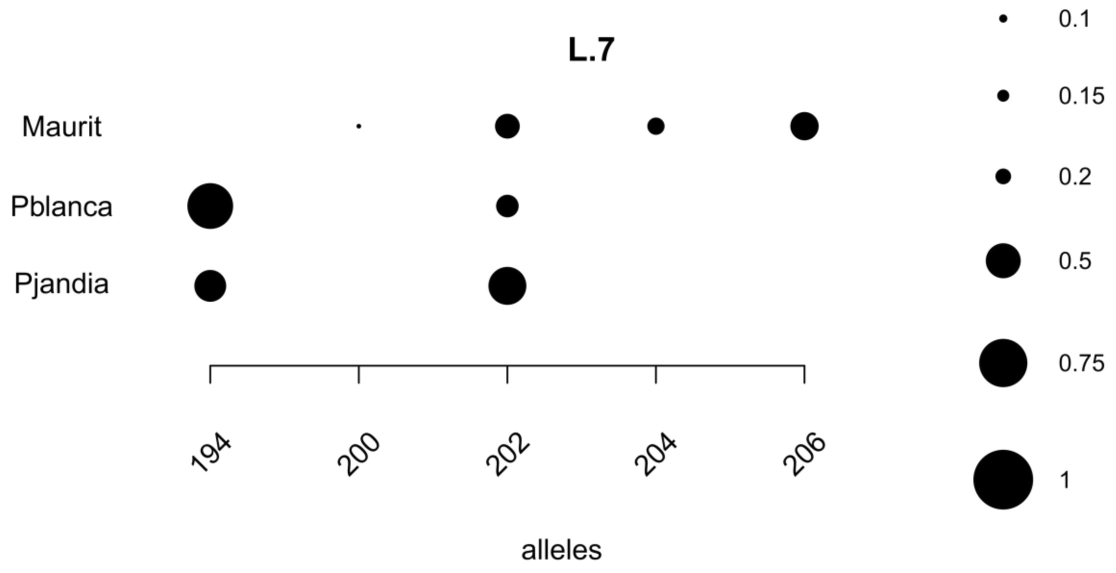
pdfout - logical value to control if a pdf with the plot should be written, defaults to *FALSE*. Currently the function to export pdf is not working in RStudio.

psize - Used to control the points cex argument to plot allele frequencies as points, as in *cex=log(allelefreq+1,base=Plogbase)*psize*. Defaults to 7.

pcol - color of points for each population takes a vector of with length equal to the number of populations. Defaults to black

pdfname - Name for the output pdf file to write the plots if pdfout is TRUE, defaults to out.pdf

```
plot.allele.freq(Exdata, LocusNames=paste("L", 1:8,  sep="."))
```



The figure below is for the seventh locus in the *Exdata*. It gets more interesting with more populations as you can see by the figure on the first page of this manual.

References

Alberto F, Arnaud-Haond S, Duarte CM, Serrao EA (2006) Genetic diversity of a clonal angiosperm near its range limit: the case of *Cymodocea nodosa* in the Canary Islands. *Marine Ecology Progress Series* 309: 117-29.

Leberg PL (2002) Estimating allelic richness: Effects of sample size and bottlenecks. *Molecular Ecology* 11: 2445-2449.

Petit RJ, El Mousadik A, Pons O (1998) Identifying populations for conservation on the basis of genetic markers. *Conservation Biology*, 12: 844-855.

R Development Core Team (2004) R: A language and environment for statistical computing. R foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-00-3, URL http://www.Rproject.org.