pophelper 1.0.0
Demonstration and workflow

Roy M Francis
roy.m.francis@outlook.com

February 20, 2014

This vignette aims to demonstrate the use of R package `pophelper`. This package contains functions that are useful for processing output results from two programs used in the analysis of population structure: 'STRUCTURE' [4] and 'TESS' [2].

The programs 'STRUCTURE' and 'TESS' are two popular programs used to differentiate populations, to determine population structure and to reveal population assignment at an individual level using molecular markers. These programs use allelic frequency information to assign individuals to a predefined number of populations (K). The assignment is usually run for a range of K such as from K=2 to K=10. Multiple repeats are also usually carried out for each K. Each output file for each repeat of K showing the assignment probabilities of all individuals is referred to as the run file or cluster file. `pophelper` has a set of functions such as tabulating runs, summarising runs, plotting runs etc that can be applied to these run files. Several other programs exist for determining population structures such as 'BAPS', 'INSTRUCT', 'GENELAND' etc which will not be covered for now.

This vignette covers the use of all important functions in the `pophelper` package. The demonstration is ordered in the manner of a typical workflow. This demonstration has been performed using R version 3.0.2 (2013-09-25) on a Windows 7 Platform x86/64. This vignette has been compiled using $\LaTeX\,2_\varepsilon$.

Input and output codes are printed in a font different from body text like `this`. Input codes always start with '`>`'. If codes break off to the next lines, then they are preceeded by '`+`'. Output results are in `this` font but do not start with '`>`'.

1 Installation

The first step is to install the `pophelper` library, which is typically only once. The dependent R packages required for the `pophelper` library as also automatically installed.

```
> install.packages("pophelper",dep=T)
```

The next step is to load the library. This is usually done everytime when starting up R.

```
> library(pophelper)
```

The installation and version can be verified using the following command.

```
> packageDescription("pophelper", fields = "Version")
```

```
[1] "1.0.0"
```

For functions where one or more files need to be selected, the selection can be performed interactively. Windows users can use `choose.files(multi=T)`. Mac users can use `file.choose()` for single selection and `tk_choose.files()` from `tcltk` package for multiple selection.

The next step to set the working directory. The working directory is a folder that usually contains the run files of interest so as to allow R to access it. Functions may produce outputs such as text files or images which will be exported to the working directory. The working directory can be set by running the command below and then selecting the folder interactively in the popup window.

```
setwd(choose.dir())
```

For this demonstration, the run files will be loaded from the `pophelper` library. Therefore, the working directory will be used for output generated from various `pophelper` functions.

Standard help and documentation for functions are obtained using `?`.

```
> ?tabulateRunsStructure
> ?summariseRunsStructure
> ?collectRunsTess
```

2 STRUCTURE Functions

In this section, we will deal with STRUCTURE output files and the functions that apply to these. The `pophelper` library must be loaded and the working directory must be set.

2.1 tabulateRunsStructure

A typical STRUCTURE run produces a large number of output run files as seen in Fig.1.

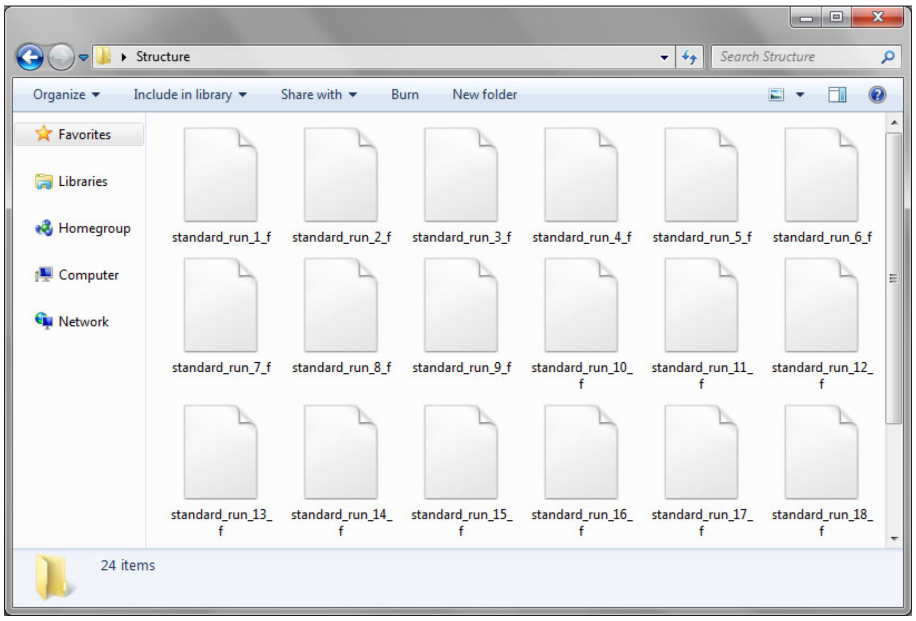

Figure 1: Typical view of Structure output run files.

The function `tabulateRunsStructure()` can be used to select any number of run files in a folder and produce a table of runs with various parameters. The only mandatory argument required for this function is a character vector of filenames. This function can be run as shown below and the files to be tabulated can be selected interactively using `choose.files(multi=TRUE)`. The results can be pointed to a variable `df` for further use.

For this demonstration, we will use the sample structure files accompanied with this library.

```
> flist<-list.files(path=system.file("/files/structure",package="pophelper"),full.names=T)
> df<-tabulateRunsStructure(files=flist)
> #common usage
> #df<-tabulateRunsStructure(files=choose.files(multi=TRUE))
> #another way
> #files=choose.files(multi=TRUE)
> #df<-tabulateRunsStructure(files=files)
```

This function produces a tabulated result and shown below.

```
> df
```

```
                  file k ind loci    elpd    mvll    vll    mva burnin  reps
1   structure_run_1_f 2 149   25 -7509.5 -7387.2  244.6 0.1280  1e+05 1e+06
10  structure_run_2_f 2 149   25 -7508.5 -7387.2  242.6 0.1270  1e+05 1e+06
17  structure_run_9_f 2 149   25 -7510.1 -7387.4  245.3 0.1330  1e+05 1e+06
2  structure_run_10_f 3 149   25 -7476.5 -7268.7  415.4 0.1212  1e+05 1e+06
11  structure_run_3_f 3 149   25 -7475.8 -7268.8  414.1 0.1201  1e+05 1e+06
12  structure_run_4_f 3 149   25 -7475.7 -7268.9  413.6 0.1218  1e+05 1e+06
3  structure_run_11_f 4 149   25 -7599.0 -7264.7  668.7 0.2225  1e+05 1e+06
7  structure_run_15_f 4 149   25 -7665.9 -7258.4  815.1 0.2082  1e+05 1e+06
13  structure_run_5_f 4 149   25 -7687.5 -7260.2  854.7 0.2130  1e+05 1e+06
4  structure_run_12_f 5 149   25 -7828.5 -7205.4 1246.2 0.2182  1e+05 1e+06
8  structure_run_16_f 5 149   25 -7709.8 -7208.7 1002.3 0.2266  1e+05 1e+06
14  structure_run_6_f 5 149   25 -7692.4 -7210.5  963.9 0.2308  1e+05 1e+06
5  structure_run_13_f 6 149   25 -8023.3 -7158.0 1730.6 0.2483  1e+05 1e+06
9  structure_run_17_f 6 149   25 -7963.6 -7164.9 1597.3 0.2581  1e+05 1e+06
15  structure_run_7_f 6 149   25 -7970.6 -7160.4 1620.5 0.2811  1e+05 1e+06
6  structure_run_14_f 7 149   25 -8571.2 -7137.5 2867.4 0.2095  1e+05 1e+06
16  structure_run_8_f 7 149   25 -8656.8 -7144.2 3025.1 0.2280  1e+05 1e+06
```

The `tabulateRunsStructure()` function produces a table listing all selected files showing 10 columns namely file names, value of K, number of individuals, number of loci, estimated ln probability of data, mean value of ln likelihood, variance of ln likelihood, mean value of alpha, number of burn-in and number of repeats. The table is sorted by loci, ind and K. Missing values are given NA.

By default, the `tabulateRunsStructure()` function prints the number of selected files as `Number of files selected: 10`. This can be turned off using the `quiet` argument as such `tabulateRunsStructure(files=files,quiet=T)`. The tabulated output can be written to the working directory as a text file by setting the argument `writetable` like so `tabulateRunsStructure(files = choose.files(multi=TRUE),writetable=T)`.

2.2 summariseRunsStructure

The table produced using `tabulateRunsStructure()` can be further collapsed by K and number of runs. The output table from `tabulateRunsStructure()` can be passed as input to `summariseRunsStructure()`. The summarised table has 6 columns namely Mean estimated ln probability of data, standard deviation, value of K, number of runs for each K, estimated ln probability of data plus standard deviation, estimated ln probability of data minus standard deviation.

```
> #usage
> df1<-summariseRunsStructure(data=df)
> #variant
> #summariseRunsStructure(data=tabulateRunsStructure(files=choose.files(multi=TRUE)))
> df1
```

```
   meanelpd         sd k runs ind loci   maxelpd   minelpd
1 -7509.367  0.8082904 2    3 149   25 -7508.558 -7510.175
2 -7476.000  0.4358899 3    3 149   25 -7475.564 -7476.436
3 -7650.800 46.1418465 4    3 149   25 -7604.658 -7696.942
4 -7743.567 74.0671542 5    3 149   25 -7669.500 -7817.634
5 -7985.833 32.6353081 6    3 149   25 -7953.198 -8018.469
6 -8614.000 60.5283405 7    2 149   25 -8553.472 -8674.528
```

The summarised runs can be written to the working directory as a text file by setting the argument `writetable=T` like so `summariseRunsStructure(data=df,writetable=T)`.

2.3 evannoMethodStructure

The Evanno method [1] is used to estimate the number of K. The method is based on Evanno et al., (2005) listed in references. The summarised runs table output from `summariseRunsStructure()` function can be input to `evannoMethodStructure()`. The `evannoMethodStructure()` function creates an Evanno derivative plot if suitable conditions are met. A resulting table is also returned. The plot can be written to file using argument `exportplot=T`. The table can be written to file using `writetable=T`.

```
> #usage
> #df2<-evannoMethodStructure(data=df1)
> #to export a plot
> #evannoMethodStructure(data=df1,exportplot=T)
> #evannoMethodStructure(data=df1,doplot=F)
> #to export plot and table
> evannoMethodStructure(data=df1,exportplot=T,writetable=T,na.rm=T)
```

```
evannoMethodStructure.txt exported
evannoMethodStructure.png exported
   meanelpd         sd k runs ind loci   maxelpd   minelpd       lnk1
1 -7509.367  0.8082904 2    3 149   25 -7508.558 -7510.175         NA
2 -7476.000  0.4358899 3    3 149   25 -7475.564 -7476.436   33.36667
3 -7650.800 46.1418465 4    3 149   25 -7604.658 -7696.942 -174.80000
4 -7743.567 74.0671542 5    3 149   25 -7669.500 -7817.634  -92.76667
5 -7985.833 32.6353081 6    3 149   25 -7953.198 -8018.469 -242.26667
6 -8614.000 60.5283405 7    2 149   25 -8553.472 -8674.528 -628.16667
     lnk1max    lnk1min      lnk2   lnk2max   lnk2min     deltaK BestK
1         NA         NA        NA        NA        NA         NA
2   33.73907   32.99427 208.16667 253.50022 162.83311 477.567086     *
3 -129.09404 -220.50596  82.03333  99.81398  64.25268   1.777851
4  -64.84136 -120.69197 149.50000 163.00654 135.99346   2.018439
5 -200.83482 -283.69851 385.90000 399.43881 372.36119  11.824616
6 -600.27363 -656.05970        NA        NA        NA         NA
```

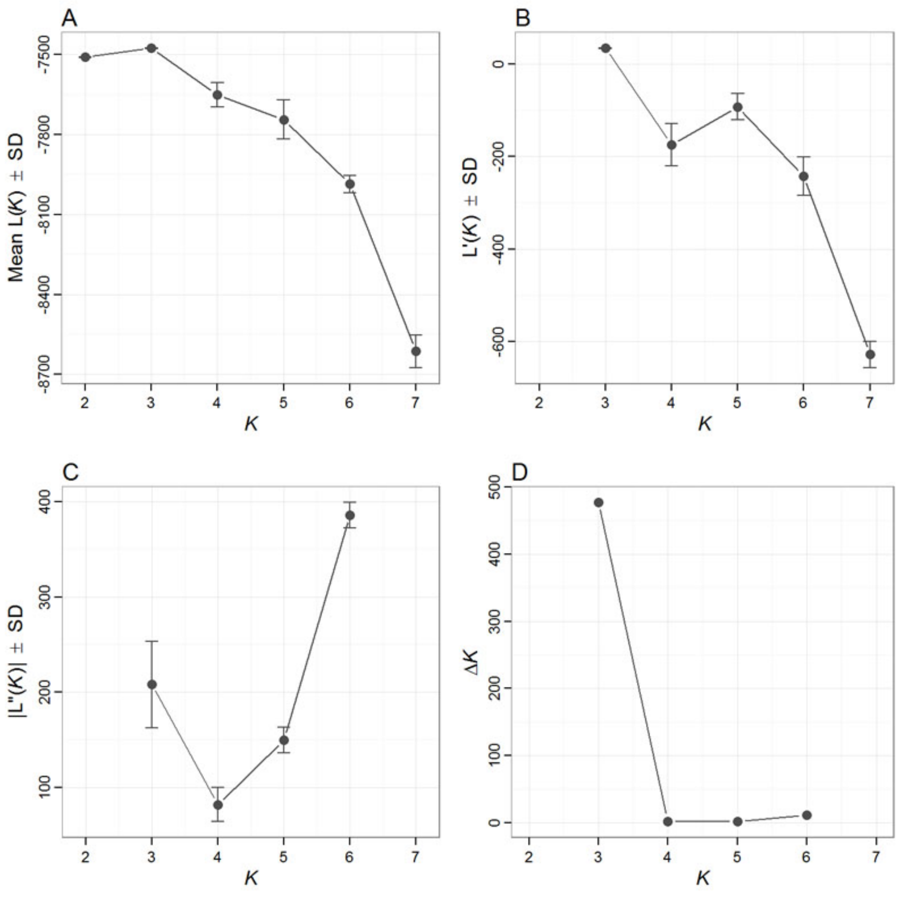

Figure 2: Plots produced from the Evanno method.

The Evanno method can be computed only if these criteria are met: At least 3 values of K must be available, values of K must be sequential (ie; there must not be missing values of K), number of individuals and loci must be same in all runs. If number of repeats for any K is less than 2, then results may not be reliable. In case the Evanno method cannot be computed, a plot of ELPD over K is produced referred to as the kPlot.

From the command above, a png file and text file is exported to the working directory. The peak of deltaK in Fig. plot (D) is usually used to estimate the value of K.

2.4 runsToDfStructure

STRUCTURE files can be converted to R dataframes using the function `runsToDfStructure()`. If one file is selected, a dataframe is returned. If multiple files are selected, then a list of dataframes are returned.

```
> #usage
> df3<-runsToDfStructure(files=flist)
> #select files interactively
> #df3<-runsToDfStructure(files=choose.files(multi=TRUE))
> head(df3[[1]])
> head(df3[[3]])
```

The dataframe contains the assignment probabilities of all individuals into K populations denoted as Cluster1, Cluster 2 etc.

2.5 clumppExportStructure

When multiple repeats are run for each K in STRUCTURE, the order of clusters may be jumbled for each run. Therefore, when plotting multiple runs within each K, the colours cannot be assigned correctly. The software CLUMPP helps to overcome this issue by reordering the clusters correctly. To read more about CLUMPP [2].

```
> #usage
> clumppExportStructure(files=flist)
> #select files interactively
> #clumppExportStructure(files=choose.files(multi=TRUE))
> #optionally change folder name
> #clumppExportStructure(files=flist,prefix="Set1")
```

This function `clumppExportStructure()` takes multiple runs for each K and combines them into a single file and generates a parameter file for easy use with CLUMPP. A combined file and a parameter file are generated for each value of K in separate directories. The name for the folder can be change optionally if required using the argument `prefix="something"`.

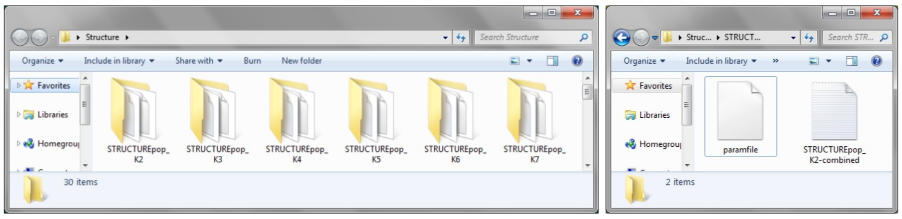

Figure 3: Folders created from clumpp export and the contents of each folder.

The CLUMPP executable can be copied and pasted into each of these folder and run by double clicking on it. This generates three output files: aligned file, merged file and misc file. This step must be carried out manually to follow the rest of the demonstration.

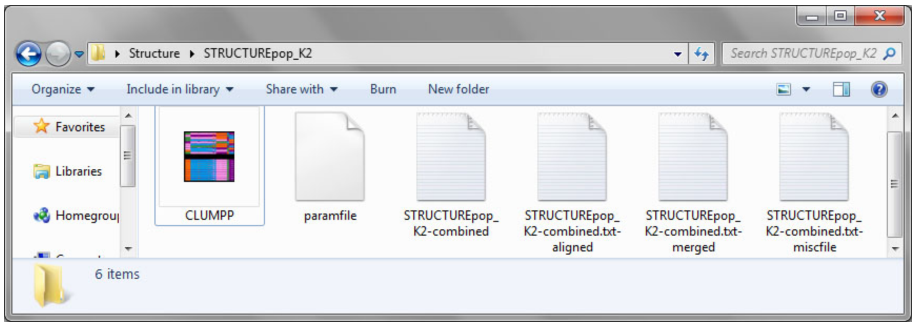

Figure 4: Folder showing CLUMPP results: aligned file, merged file and misc file.

The aligned file contains all the runs in the combined file after realignment of clusters. In contrast, the merged file contains only one table which merges all the aligned runs to create a consensus run. The merged file makes sense only if all the aligned runs have similar assignments. The miscfile contains run parameters and other details.

2.6 collectClumppOutput

The CLUMPP output files are now distributed in multiple folders. The aligned, merged or both files can be copied from multiple folders into a single folder for further analyses using the function `collectClumppOutput()`. The working directory is set suitably before running this function. This function need a `prefix` argument which denotes the prefix used in the previous function or the text before the underscore. For ex. a directory named STRUCTUREpop_K2 has the prefix STRUCTUREpop.

```
> #usage
> #setwd(choose.dir())
> #collectClumppOutput(prefix="STRUCTUREpop",filetype="aligned")
> #collectClumppOutput(prefix="STRUCTUREpop",filetype="merged")
> #working directory can also be set using runsdir argument
> #collectClumppOutput(prefix="STRUCTUREpop",filetype="both",runsdir=choose.dir())
>
> collectClumppOutput(prefix="STRUCTUREpop",filetype="both")
```

Collecting the aligned and merged files into a single folder can be helpful in plotting these files. From the command above, both aligned and merged files are copied into a new directory STRUCTUREpop-both within the working directory.

2.7 plotRuns

The function `plotRuns()` is used to create barplots from STRUCTURE run files, TESS run files, combined files, aligned files or merged files. The default usage below (usage 1) creates barplots of all selected files individually/separately. To create a single output figure with all selected files joined together, set argument `imgoutput="join"`.

```
> #usage 1 #plot first 2 runs
> #plotRuns(files=flist[1:2])
> #same as above
> plotRuns(files=flist[1:2],imgoutput="sep")
> #usage 2, join files into one figure
> plotRuns(files=flist[1:2],imgoutput="join")
> #
```

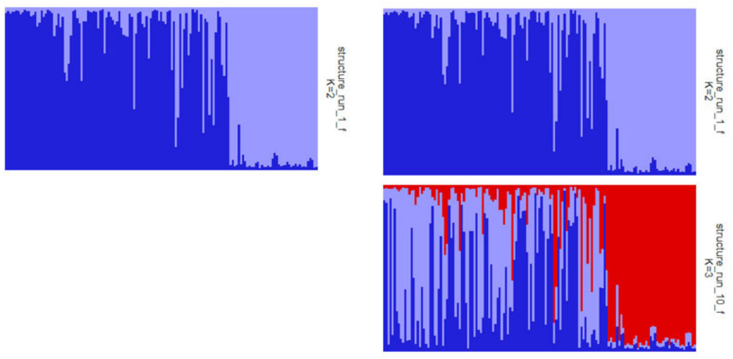

Figure 5: (A) Left: Single run plotted separately. (B) Right: Two runs joined together in one image.

Population labels can be added to this barplot by providing a vector of labels. The length must be equal to the number of individuals. The labelling is still experimental and may not always give perfect results. This function has numerous arguments to tweak the plot as required. In this demonstration, we will use the labels table in the `pophelper` library. In case of separate plots and joined plots, the labels are shown only once at the bottom of the plot.

```
> pops<-read.delim(system.file("files/structurepoplabels.txt",package="pophelper"),header=F)
> head(pops$V1)
> length(pops$V1)
> #print label
> #plotRuns(files=flist,poplab=pops$V1)
> #same as above
> plotRuns(files=flist[1],imgoutput="sep",poplab=pops$V1)
> #usage 2, join files into one figure
> plotRuns(files=flist[1:2],imgoutput="join",poplab=pops$V1)
> #adjust angle of labels
> #plotRuns(files=flist[1],imgoutput="sep",poplab=pops$V1,labangle=90)
> #
```

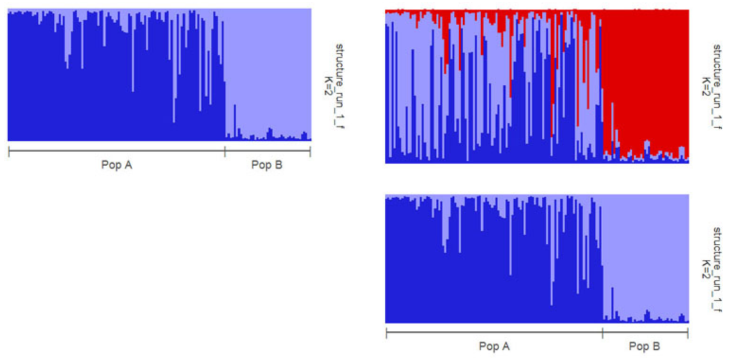

Figure 6: (A) Left: Single run plotted separately with pop labels. (B) Right: Two runs joined together in one image with pop labels.

In similar manner, the `plotsRuns` function can be used to plot from combined, aligned or merged files. For this set argument `imgoutput="tab"`.

```
> tabs1<-c(system.file("files/STRUCTUREpop_K4-combined.txt",package="pophelper"),
+          system.file("files/STRUCTUREpop_K4-combined-aligned.txt",package="pophelper"),
+          system.file("files/STRUCTUREpop_K4-combined-merged.txt",package="pophelper"))
> #usage
> plotRuns(files=tabs1,imgoutput="tab")
> #with labels
> plotRuns(files=tabs1,imgoutput="tab",poplab=pops$V1)
> #
```

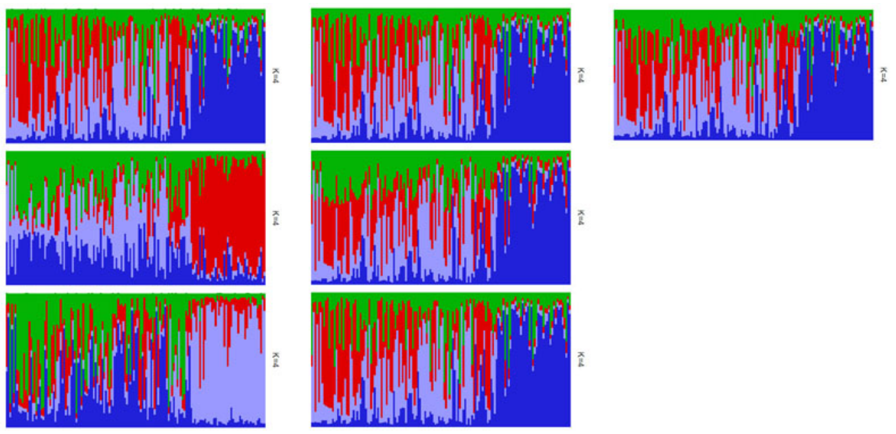

Figure 7: (A) Left: Combined files (Three STRUCTURE runs for K=4). (B) Middle: Aligned files (Three STRUCTURE runs for K=4 aligned using CLUMPP). (C) Right: Merged file (Three runs for K=4 merged into one table/figure using CLUMPP)

2.8 plotMultiline

The `plotMultiline` function is also used to plot STRUCTURE runs, TESS runs and table files as barplots. This function automatically identifies these three input file types. The barplots are plotted in multiple rows to enable easier identification of individuals. The figure is produced on A4 size by default. The number of samples per line (`spl`) and the number of lines per page (`lpp`) can be specified.

```
> plotMultiline(flist[1])
> #manually modified
> plotMultiline(flist[1],spl=75,lpp=10)
```

Note that this is a slow function and takes several minutes to run depending on number of individuals and number of files selected. The results from above commands is shown below.

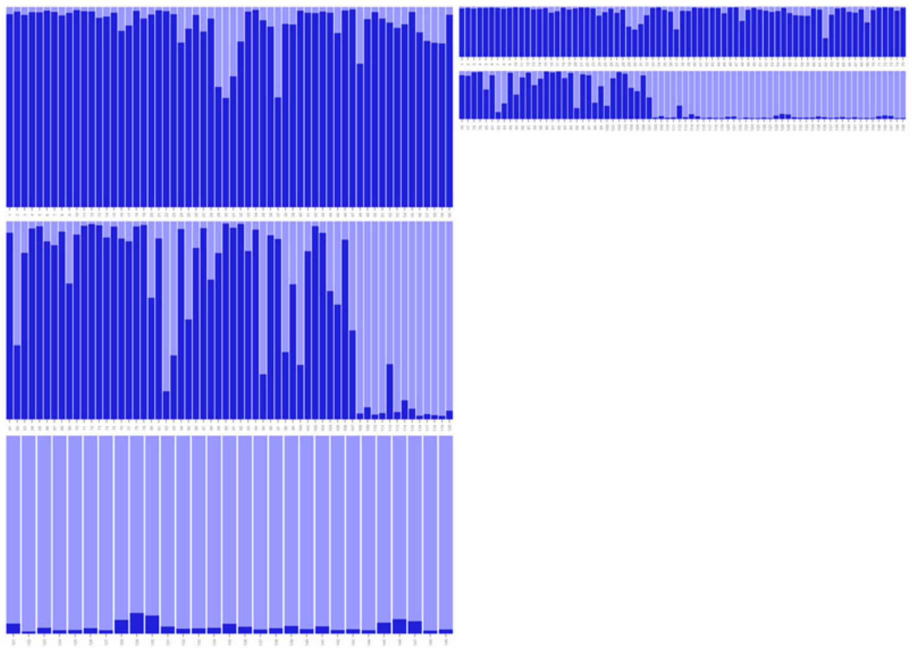

Figure 8: Left: Default output from `plotMultiline`. Right: Modified output.

3 TESS functions

In this section, we will deal with TESS output files and the functions that apply to these.

3.1 collectRunsTess

Unlike STRUCTURE runs which are exported into a single directory, TESS output files are exported into separate directories by run. This means that one needs to go into individual folders to obtain the run file. The function `collectRunsTess()` collects TESS cluster files from individual run folders into one new folder and renamed by run.

```
> #usage
> #collectRunsTess(runsdir = choose.dir())
```

Within each TESS run folder, the function searches for filename ending with 'TR.txt' as the cluster file. This file is copied to the new folder and renamed as the name of the respective run directory. Therefore, do not manually rename original run files or directories.

3.2 tabulateRunsTess

The `pophelper` library must be loaded and the working directory must be set appropriately. The function `tabulateRunsTess()` can be used to select any number of TESS run files in a folder and produce a table of runs with filename, K and number of individuals. The only mandatory argument required for this function is a vector of filenames. This can be run as below and the files to be tabulated can be selected interactively using the function `choose.files(multi=TRUE)`. The results can be pointed to a variable `df`.

For this demonstration, we will use the sample TESS files accompanied with this library.

```
> flist1<-list.files(path=system.file("files/tess",package="pophelper"),full.names=T)
> df<-tabulateRunsTess(files=flist1)
> #common usage
> #df<-tabulateRunsTess(files=choose.files(multi=TRUE))
> #another way
> #flist=choose.files(multi=TRUE)
> #df<-tabulateRunsTess(files=flist1)
> #write table as text file
> #df<-tabulateRunsTess(files=choose.files(multi=TRUE),writetable=T)
```

This function produces a tabulated result and shown below. The table is sorted by ind and K.

```
> head(df)
```

```
                  file k ind
1  TESS_RUN_000001.txt 2  75
8  TESS_RUN_000008.txt 2  75
15 TESS_RUN_000015.txt 2  75
2  TESS_RUN_000002.txt 3  75
9  TESS_RUN_000009.txt 3  75
16 TESS_RUN_000016.txt 3  75
```

By default, the `tabulateRunsTess()` function prints the number of selected files as `Number of files selected:  10`. This can be turned off using the `quiet` argument as such `tabulateRunsTess(files=files),quiet=T`. The tabulated output can be written to the working directory as a text file by setting the option `writetable` like so `tabulateRunsTess(files = choose.files(multi=TRUE),writetable=T)`.

3.3 summariseRunsTess

This function further condenses the tabulated table by number of runs.

```
> df2<-summariseRunsTess(df)
> head(df2)
```

```
  k runs ind
1 2    3  75
2 3    3  75
3 4    3  75
4 5    3  75
5 6    3  75
6 7    3  75
```

3.4 runsToDfTess

TESS files can be converted to R dataframes using the function `runsToDfTess()`. If one file is selected, a dataframe is returned. If multiple files are selected, then a list of dataframes are returned.

```
> #usage
> df3<-runsToDfTess(files=flist1)
> #select files interactively
> #df3<-runsToDfTess(files=choose.files(multi=TRUE))
> head(df3[[1]])
> head(df3[[3]])
```

The dataframe contains the assignment probabilities of all individuals into K populations denoted as Cluster1, Cluster 2 etc.

3.5 clumppExportTess

When multiple repeats are run for each K in TESS, the order of clusters may be jumbled for each run. Therefore, when plotting multiple runs within each K, the colours cannot be assigned correctly. The software CLUMPP helps to overcome this issue by reordering the clusters correctly.

```
> #usage
> clumppExportTess(files=flist1)
> #select files interactively
> #clumppExportTess(files=choose.files(multi=TRUE))
> #optionally change folder name
> #clumppExportTess(files=flist1,prefix="Set1")
```

This function `clumppExportTess()` takes multiple runs for each K and combines them into a single file and generates a parameter file for easy use with CLUMPP. A combined file and a parameter file are generated for each value of K in separate directories. The name for the folder can be change optionally if required using the argument `prefix="something"`.

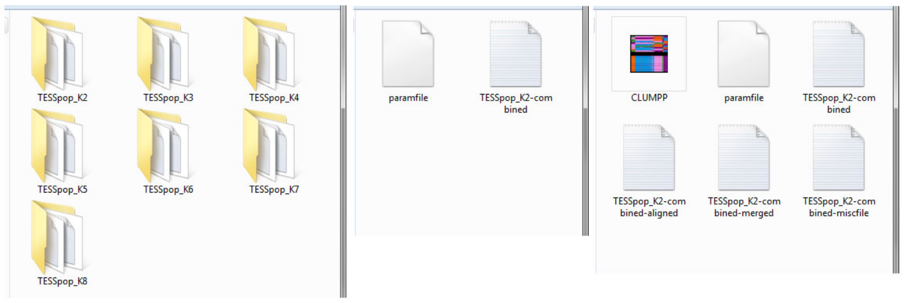

Figure 9: Left: Output generated from function `clumppExportTess()`. Middle: Output inside a sample directory. Right: Files generated by running CLUMPP.

The CLUMPP executable can be copied and pasted into each of these folder and run by double clicking on them. This generates three output files: aligned file, merged file and misc file. This step must be carried out manually to follow the rest of the demonstration.

The aligned file contained all the runs in the combined file after realignment of clusters. In contrast, the merged file contains only one table which merges all the aligned runs to create a consensus run. The merged file makes sense only if all the aligned runs have similar assignments. The miscfile contains run parameters and other details.

3.6 collectClumppOutput

Same function as covered in section STRUCTURE functions. The Clumpp output files are now distributed in multiple folders. The aligned, merged or both files can be copied from multiple folders into a single folder for further analyses using the function `collectClumppOutput()`. The working directory is set suitably before running this function. This function need a `prefix` argument which denotes the prefix used in the previous function or the text before the underscore. For ex. a directory named TESSpop_K2 has the prefix TESSpop.

```
> #usage
> #setwd(choose.dir())
> #collectClumppOutput(prefix="TESSpop",filetype="aligned")
> #collectClumppOutput(prefix="TESSpop",filetype="merged")
> #working directory can also be set using runsdir argument
> #collectClumppOutput(prefix="TESSpop",filetype="both",runsdir=choose.dir())
>
> collectClumppOutput(prefix="TESSpop",filetype="both")
```

Collecting the aligned and merged files into a single folder can be helpful in plotting these files. From the command above, both aligned and merged files are copied into a new directory TESSpop-both within the working directory.

3.7 plotRuns

Same function as covered in section STRUCTURE functions. The function `plotRuns()` is used to create barplots from TESS run files, combined files, aligned files or merged files. The default usage below (usage 1) creates barplots of all selected files individually/separately. To create a single output figure with all selected files joined together, set argument `imgoutput="join"`.

```
> #usage 1 #plot first 2 runs
> #plotRuns(files=flist1[1:2])
> #same as above
> plotRuns(files=flist1[1:2],imgoutput="sep")
> #usage 2, join files into one figure
> plotRuns(files=flist1[1:2],imgoutput="join")
> #
```

Population labels can be added to this barplot by providing a vector of labels. The length must be equal to the number of individuals. In this demonstration, we will create some sample labels. In case of separate plots and joined plots, the labels are shown only once at the bottom of the plot.

```
> labs1 <- factor(c(rep("PopA",30),rep("PopB",45)))
> length(labs1)
> #print label
> #plotRuns(files=flist1,poplab=labs)
> #same as above
> plotRuns(files=flist1[1],imgoutput="sep",poplab=labs1)
> #usage 2, join files into one figure
> plotRuns(files=flist1[1:2],imgoutput="join",poplab=labs1)
> #adjust angle of labels
> #plotRuns(files=flist1[1],imgoutput="sep",poplab=labs,labangle=90)
> #
```

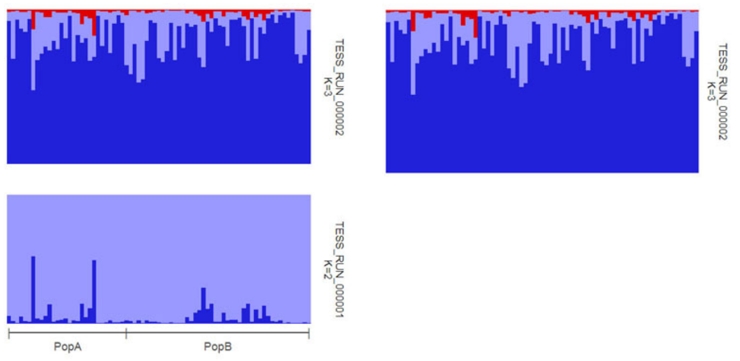

Figure 10: Sample figures from plotting TESS runs.

In similar manner, the `plotsRuns` function can be used to plot from combined, aligned or merged files. For this set argument `imgoutput="tab"`.

```
> tabs2<-c(system.file("files/TESSpop_K5-combined.txt",package="pophelper"),
+          system.file("files/TESSpop_K5-combined-aligned.txt",package="pophelper"),
+          system.file("files/TESSpop_K5-combined-merged.txt",package="pophelper"))
> #usage
> plotRuns(files=tabs2,imgoutput="tab")
> #with labels
> plotRuns(files=tabs2,imgoutput="tab",poplab=labs1)
> #
```

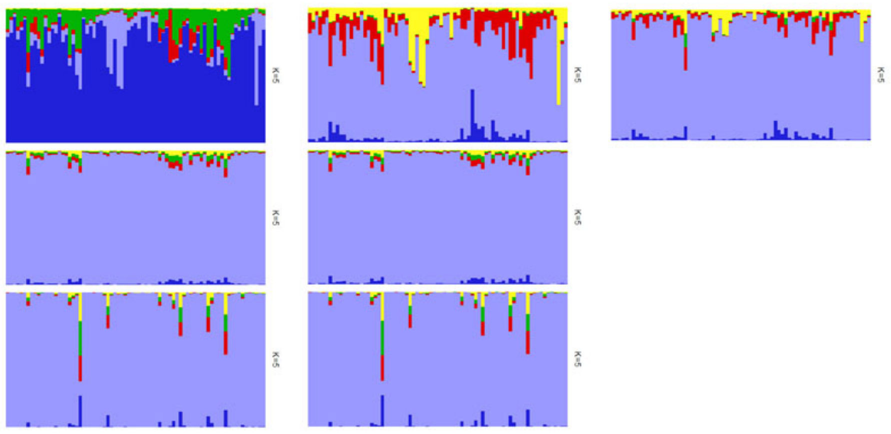

Figure 11: (A) Left: Combined files (Three TESS runs for K=5). (B) Middle: Aligned files (Three TESS runs for K=5 aligned using CLUMPP). (C) Right: Merged file (Three runs for K=5 merged into one table/figure using CLUMPP)

3.8 plotMultiline

The `plotMultiline` function is also used to plot STRUCTURE runs, TESS runs and table files as barplots. This function automatically identifies these three input file types. The barplots are plotted in multiple rows to enable easier identification of individuals. The figure is produced on A4 size by default. The number of samples per line (`spl`) and the number of lines per page (`lpp`) can be specified.

```
> plotMultiline(flist[1])
> plotMultiline(flist[1],spl=75,lpp=10)
```

Note that this is a slow function and takes several minutes to run depending on number of individuals and number of files selected.

4 Working code

Here is a list of all functions in a order typical of workflow.

4.1 Structure

```
> #choose files
> flist<-choose.files(multi=TRUE)
> #tabulate runs
> df1<-tabulateRunsStructure(flist)
> #summarise runs
> df2<-summariseRunsStructure(df1)
> #Evanno method
> evannoMethodStructure(df2,exportplot=T)
> #clumpp export
> clumppExportStructure(flist)
> #collect clumpp output
> collectClumppOutput(prefix="STRUCTUREpop",filetype="aligned")
> #plot runs
> plotRuns(files=flist)
> plotRuns(files=flist,imgoutput="join")
> plotRuns(files=choose.files(multi=TRUE),imgoutput="tab")
> #plot multiline
> plotMultiline(flist[1])
> plotMultiline(flist[1],spl=70,lpp=10)
> #structure runs to data frame
> runsToDfStructure(flist)
```

4.2 Tess

```
> #collect TESS output
> collectRunsTess(runsdir=choose.dir())
> #choose files
> flist<-choose.files(multi=TRUE)
> #tabulate runs
> df1<-tabulateRunsTess(flist)
> #clumpp export
> clumppExportTess(flist)
> #collect clumpp output
> collectClumppOutput(prefix="TESSpop",filetype="aligned")
> #plot runs
> plotRuns(files=flist)
> plotRuns(files=flist,imgoutput="join")
> plotRuns(files=choose.files(multi=TRUE),imgoutput="tab")
> #plot multiline
> plotMultiline(flist[1])
> plotMultiline(flist[1],spl=70,lpp=10)
> #TESS runs to data frame
> runsToDfTess(flist)
```

Some of the same functionalities in this package have been implented in the online tool `StructureHarvester` by other authors. For those who prefer a GUI, this may be useful. See useful links.

5 References

1. Evanno, G., Regnaut, S., and Goudet, J. (2005). Detecting the number of clusters of individuals using the software STRUCTURE: a simulation study. Molecular ecology, 14(8), 2611-2620
(Link)

2. FranÃğois, O., and Durand, E. (2010). Spatially explicit Bayesian clustering models in population genetics. Molecular Ecology Resources, 10(5), 773-784.
(Link)

3. Jakobsson, M., and Rosenberg, N. A. (2007). CLUMPP: a cluster matching and permutation program for dealing with label switching and multimodality in analysis of population structure. Bioinformatics, 23(14), 1801-1806.
(Link)

4. Pritchard, J. K., Stephens, M., and Donnelly, P. (2000). Inference of population structure using multilocus genotype data. Genetics, 155(2), 945-959.
(Link)

6 Useful links

STRUCTURE program
`http://pritchardlab.stanford.edu/structure.html`

TESS program
`http://membres-timc.imag.fr/Olivier.Francois/tess.html`

CLUMPP program
`http://www.stanford.edu/group/rosenberglab/clumpp.html`

Structure Harvester
`http://taylor0.biology.ucla.edu/structureHarvester`

Disclaimer