Using population strata in *adegenet* 2.0.0

Zhian N. Kamvar [*1] and Thibaut Jombart [†2]

[1] *Department of Botany and Plant Pathology, Oregon State University*

[2] *MRC Centre for Outbreak Analysis and Modelling. Imperial College London*

June 19, 2015

Abstract

This vignette provides a short tutorial on the use of the **strata** slot within GENIND and GENLIGHT objects. The slot was added in *adegenet* version 2.0-0 from the *poppr* package [1]. Sampling schemes are often not constrained to simply one population factor. They can be stratified over years, treatments, and geographic locations. The **strata** slot provides a way to store and combine any population strata related to your data.

[*] kamvarz@science.oregonstate.edu

[†] tjombart@imperial.ac.uk

Contents

1 Introduction to strata

in *adegenet*, the GENIND and GENLIGHT objects contain a slot called `strata` which can contain a data frame with the same number of rows as samples in your data set. This slot is used to define the population factor of the data. One convenient way of defining strata is to concatenate them using '_' and use them to define a single population in your data before you import it into *adegenet*. Examples of this format can be found in the supplementary data for [2] (file: *fsp_rob_pop.csv*) and at `http://dx.doi.org/10.6084/m9.figshare.877104`. Another way to define strata is to import them in a data frame as we will demonstrate below.

In this vignette, we will show you how to DEFINE populations strata, VIEW those strata to ensure that they are correctly defined, MANIPULATE your strata by adding and renaming them, and use these levels to SET THE POPULATION in your GENIND and GENLIGHT objects using the following methods:

Method	Function	Input	Result
split	`splitStrata`	formula	defined hierarchical levels
set	`strata`	data frame	new hierarchical levels
get	`strata`	formula	data frame
name	`nameStrata`	formula	new hierarchical level names
add	`addStrata`	vector or data frame	new hierarchical level

A NOTE ABOUT FORMULAS
The formulas used by genind objects always start with a ~ and levels are always separated by a /. Some examples are:
~Country/City/District
~Field/Year

In the next section, we'll explore two ways of defining population strata in your GENIND and GENLIGHT objects.

2 Defining strata

In this section, we will give examples of two ways to define a population strata. We will use the `microbov` data set which contains 704 samples of cattle from two countries, split into two species, and 15 breeds. The information on the population stratification is located in the **other** slot. Notice that the **strata** slot is currently empty.

```
library("adegenet")

## Loading required package:  ade4
##    ==========================
##    adegenet 2.0.0 is loaded
##    ==========================
##
##  - to start:  type '?adegenet'
##  - to browse the adegenet website:  type 'adegenetWeb()'
##  - to post questions/comments:  adegenet-forum@lists.r-forge.r-project.org
##  - to report bugs, request features, contribute:  http://goo.gl/dZuu5X

data(microbov)
microbov

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: - empty -
```

```
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe

lapply(other(microbov), summary) # Showing counts of each factor

## $coun
##  AF  FR
## 231 473
##
## $breed
##          Aubrac        Bazadais BlondeAquitaine          Borgou
##              50              47              61              50
##    BretPieNoire       Charolais          Gascon       Lagunaire
##              31              55              50              51
##        Limousin      MaineAnjou     Montbeliard           NDama
##              50              49              30              30
##          Salers           Somba            Zebu
##              50              50              50
##
## $spe
##  BI  BT
## 100 604
```

2.1 Using a data frame

By default, the **strata** slot in a GENIND object is empty, but it can be filled with a data frame as long as the number of rows match the number of samples in the data. We will use the information in the **other** slot to create the data frame.

```
strata_df <- data.frame(other(microbov)) # Create a data frame from the list.
head(strata_df)

##   coun  breed spe
## 1   AF Borgou  BI
## 2   AF Borgou  BI
## 3   AF Borgou  BI
## 4   AF Borgou  BI
## 5   AF Borgou  BI
## 6   AF Borgou  BI
```

Now that we have our strata defined in a data frame, we can set it as the population stratification by using the `strata` command:

```
strata(microbov) <- strata_df
microbov

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 3 columns ( coun, breed, spe )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe
```

We can now see that there are three strata defined in the `microbov` genind object.

2.2 Using `df2genind`

If you wanted to input strata from a file, you can use the field in the function `df2genind`. Let's use our microbov data set as an example. For this, we need 2 data frames: one that contains the genetic information and one that contains the strata.

```
micdf <- genind2df(microbov, sep = "/") # Convert to a data frame
micdf[1:6, 1:6] # genetic data

##                pop  INRA63   INRA5  ETH225  ILSTS5    HEL5
## AFBIBOR9503 Borgou 183/183 137/141 147/157 190/190 149/149
## AFBIBOR9504 Borgou 181/183 141/141 139/157 186/186 151/163
## AFBIBOR9505 Borgou 177/183 141/141 139/139 194/194 151/165
## AFBIBOR9506 Borgou 183/183 141/141 141/147 184/190 167/167
## AFBIBOR9507 Borgou 177/183 141/141 153/157 184/186 155/165
## AFBIBOR9508 Borgou 177/183 137/143 149/157 184/186 155/165

head(strata_df) # strata

##   coun  breed spe
## 1   AF Borgou  BI
## 2   AF Borgou  BI
## 3   AF Borgou  BI
```

```
## 4   AF Borgou  BI
## 5   AF Borgou  BI
## 6   AF Borgou  BI
```

Now that we have our data frames, we can import them using `df2genind`:

```
newmic <- df2genind(micdf[-1], sep = "/", pop = micdf$pop, strata = strata_df)
newmic

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: df2genind(X = micdf[-1], sep = "/", pop = micdf$pop, strata = strata_df)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 3 columns ( coun, breed, spe )
##    @hierarchy:  - empty -
##    @other: - empty -
```

2.3 Using an input file

As explained in Introduction to strata, one useful way to define strata for your data is to concatenate them using '_' and set that as your population factor in your original input file. This allows you to carry your strata with the genetic data even when it's not in R.

To demonstrate this, we will use the `microbov` data and import it using `df2genind`. Note that the code immediately following is simply to set up the data for the example. Normally your data would be in a text file. You do not need to understand the following code.

```
strata_comb <- do.call("paste", c(setNames(strata_df, NULL), sep = "_"))
head(strata_comb)        # Combined strata

## [1] "AF_Borgou_BI" "AF_Borgou_BI" "AF_Borgou_BI" "AF_Borgou_BI"
## [5] "AF_Borgou_BI" "AF_Borgou_BI"

micdf$pop <- strata_comb # replace the population with the combined strata.
micdf[1:6, 1:6]
```

```
##                      pop  INRA63   INRA5  ETH225  ILSTS5    HEL5
## AFBIBOR9503 AF_Borgou_BI 183/183 137/141 147/157 190/190 149/149
## AFBIBOR9504 AF_Borgou_BI 181/183 141/141 139/157 186/186 151/163
## AFBIBOR9505 AF_Borgou_BI 177/183 141/141 139/139 194/194 151/165
## AFBIBOR9506 AF_Borgou_BI 183/183 141/141 141/147 184/190 167/167
## AFBIBOR9507 AF_Borgou_BI 177/183 141/141 153/157 184/186 155/165
## AFBIBOR9508 AF_Borgou_BI 177/183 137/143 149/157 184/186 155/165
```

We are using a data frame as an example, so our input file would look a bit different if we were using a different file format like FSTAT, but the basic idea is the same. When we import our data, we have the population factor in the first column and the genotypes in the other columns. We can import our data this way:

```
mbov_from_df <- df2genind(micdf[-1], sep = "/", pop = micdf$pop)
mbov_from_df

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: df2genind(X = micdf[-1], sep = "/", pop = micdf$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: - empty -
##    @hierarchy:  - empty -
##    @other: - empty -
```

It says we don't have any strata set. We can use the population factor.

```
strata(mbov_from_df) <- data.frame(pop = pop(mbov_from_df))
mbov_from_df

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
```

```
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: df2genind(X = micdf[-1], sep = "/", pop = micdf$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 1 columns ( pop )
##    @hierarchy:  - empty -
##    @other: - empty -
```

We know that we have all of our strata combined into one level. We can split them by using the command `splitStrata`.

```
splitStrata(mbov_from_df) <- ~Country/Breed/Species
mbov_from_df

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: df2genind(X = micdf[-1], sep = "/", pop = micdf$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 3 columns ( Country, Breed, Species )
##    @hierarchy:  - empty -
##    @other: - empty -
```

And now we have our stratifications.

3 Viewing strata

If you wanted to view your strata to make sure that you made no spelling errors in your population definitions, you can extract the data frame from your genind object by using the function `strata`: Note, I'm wrapping the command in `head` to prevent all 704 rows from printing:

```
# Show the whole strata
head(strata(mbov_from_df))

##   Country  Breed Species
## 1      AF Borgou      BI
## 2      AF Borgou      BI
## 3      AF Borgou      BI
## 4      AF Borgou      BI
## 5      AF Borgou      BI
## 6      AF Borgou      BI
```

You can also hierarchically combine specific levels of strata by using a formula inside the function:

```
# Show only country and breed, hierarchically
head(strata(mbov_from_df, ~Country/Breed))

##   Country     Breed
## 1      AF AF_Borgou
## 2      AF AF_Borgou
## 3      AF AF_Borgou
## 4      AF AF_Borgou
## 5      AF AF_Borgou
## 6      AF AF_Borgou
```

If you don't want to combine the strata hierarchically, you can set `combine = FALSE`.

```
head(strata(mbov_from_df, ~Country/Breed, combine = FALSE))

##   Country  Breed
## 1      AF Borgou
## 2      AF Borgou
## 3      AF Borgou
## 4      AF Borgou
## 5      AF Borgou
## 6      AF Borgou
```

4 Defining populations with strata

The function `setPop` will eventually be your best friend. You can change the defined population simply by using a hierarchical formula.

```
popNames(microbov) # by breed
```

```
##  [1] "Borgou"          "Zebu"            "Lagunaire"
##  [4] "NDama"           "Somba"           "Aubrac"
##  [7] "Bazadais"        "BlondeAquitaine" "BretPieNoire"
## [10] "Charolais"       "Gascon"          "Limousin"
## [13] "MaineAnjou"      "Montbeliard"     "Salers"
```

```
setPop(microbov) <- ~coun/spe
popNames(microbov) # by country and species
```

```
## [1] "AF_BI" "AF_BT" "FR_BT"
```

```
setPop(microbov) <- ~breed
popNames(microbov) # back to breed
```

```
##  [1] "Borgou"          "Zebu"            "Lagunaire"
##  [4] "NDama"           "Somba"           "Aubrac"
##  [7] "Bazadais"        "BlondeAquitaine" "BretPieNoire"
## [10] "Charolais"       "Gascon"          "Limousin"
## [13] "MaineAnjou"      "Montbeliard"     "Salers"
```

5 Manipulating strata

You can also add and rename strata.

5.1 Adding levels

If you obtain a new stratification in your data, or want to stratify it by some new genetic characteristic, you can add a new level to your existing strata with the function `addStrata`.

```
microbov # Has three strata
```

```
## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
```

```
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 3 columns ( coun, breed, spe )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe

# Creating dummy strata
newstrat <- sample(paste0("P", 1:2), 704, replace = TRUE)

addStrata(microbov) <- data.frame(dummy = newstrat)
microbov

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 4 columns ( coun, breed, spe, dummy )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe
```

Note that you can add any number of strata by using a data frame.

5.2 Naming levels

Sometimes, the names of your strata are not what you want. You can rename the strata using the function `nameStrata`:

```
microbov # names are coun breed spe dummy

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
```

```
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 4 columns ( coun, breed, spe, dummy )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe

nameStrata(microbov) <- ~Country/Breed/Species/Dummy
microbov

## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 30-61)
##    @strata: a data frame with 4 columns ( Country, Breed, Species, Dummy )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe
```

6 Interpreting error messages

There are times when you might mis-type something or use capitalization when your data has none. In these instances functions will throw errors. Often, they can be difficult to interpret. This section will show some example warnings and how to interpret them.

6.1 setPop warnings/errors

```
setPop(microbov) <- ~country/breed

## Error:  One or more levels in the given strata is not present in the data
frame.
##  ---------------------------------------------------------------------------
## strata:  country, breed
## Data:  Country, Breed, Species, Dummy
```

This error message explains that we have supplied a stratification that does not exist in our data. Even though we got the words correct, R is case-sensitive, so we need to make sure our capitalization is correct:

```
setPop(microbov) <- ~Country/Breed
```

Here's another one that might come up:

```
setPop(microbov) <- Country/Breed

## Error in eval(expr, envir, enclos):  object 'Country' not found
```

This one is tricky because there is only one single character missing. This formula doesn't have the "∼". When this happens, R doesn't recognize it as a formula and thinks that you want to divide the variable `Country` by the variable `Breed`. Let's supposed that you actually had two variables defined as `Country` and `Breed`. Would you still be able to get away without including the ∼.

```
Country <- runif(10)
Breed <- runif(10)
setPop(microbov) <- Country/Breed

## Error:  formula must be a valid formula object.
```

Again, this is telling you that what you've presented is not a formula. This is fixed by placing the ∼ after the arrow. One last error that can come up with `setPop` is when you don't have any strata:

```
data("nancycats")
nancycats

## /// genind object \\\
##
##  // Basic content
```

```
##    @tab:  237 x 108 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 8-18)
##    @loc.fac: locus factor for the 108 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(nancycats)$tab, pop = truenames(nancycats)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 9-23)
##    @strata: - empty -
##    @hierarchy:  - empty -
##    @other: a list containing: xy

setPop(nancycats) <- ~group

## Warning:  Cannot set the population from an empty strata
```

Here, you get a warning instead of an error. This is telling you that your `@strata` slot is empty.

6.2 splitStrata warnings/errors

The `splitStrata` function allows you to import your strata via a standard format. If we revisit the microbov data from above, we can combine the strata to create an example:

```
setPop(microbov) <- ~Country/Breed/Species/Dummy
strata(microbov) <- data.frame(pop = pop(microbov))
microbov
```

```
## /// genind object \\\
##
##  // Basic content
##    @tab:  704 x 373 matrix of allele counts
##    @loc.nall: number of alleles per locus (range: 5-22)
##    @loc.fac: locus factor for the 373 columns of @tab
##    @all.names: list of allele names for each locus
##    @ploidy: ploidy of each individual  (range: 2-2)
##    @type:  codom
##    @call: genind(tab = truenames(microbov)$tab, pop = truenames(microbov)$pop)
##
##  // Optional content
##    @pop:  population of each individual (group size range: 12-31)
##    @strata: a data frame with 1 columns ( pop )
##    @hierarchy:  - empty -
##    @other: a list containing: coun  breed  spe
```

Now we see that microbov only has 1 stratification called "pop", but we know that it actually has 4. What happens if we only name 3 levels in `splitStrata`?

```
splitStrata(microbov) <- ~Country/Breed/Species

## Error:
##  Data has 4 levels of strata with the separator _ .
##  Here is the fist column of the data:  AF_Borgou_BI_P2
```

This error is showing us on the first line that we have specified too few strata to split (we supplied 3, but it needs 4). The second line shows us what the first row of the combined strata looks like so we can figure out what the possible names should be.

```
splitStrata(microbov) <- ~Country/Breed/Species/Dummy
```

Now we have everything split. What happens if we run the function a second time?

```
splitStrata(microbov) <- ~Country/Breed/Species/Dummy

## Warning:  Strata must be length 1.  Taking the first column.
## Error:
##  Data has 1 level of strata with the separator _ .
##  Here is the fist column of the data:  AF
```

This is a complicated one. Since this function is meant to be used only once, it can have side-effects when used multiple times. Here, we are given the error as we saw before, but we are also given a warning that "Strata must be length 1". This means that, when we use `splitStrata`, our strata slot must have a 1 column data frame in it. Since there are more than 1 columns, it is ignoring the others.

References

[1] Zhian N Kamvar, Jonah C Brooks, and Niklaus J Grünwald. Novel R tools for analysis of genome-wide population genetic data with emphasis on clonality. *Frontiers in Genetics*, 6:208, 2015.

[2] Ryan M Kepler, Todd A Ugine, Jude E Maul, Michel A Cavigelli, and Stephen A Rehner. Community composition and population genetics of insect pathogenic fungi in the genus *Metarhizium* from soils of a long-term agricultural research system. *Environmental microbiology*, 2015.