Introduction to PopGenReport

using PopGenReport Ver. 2.0

Gruber & Adamack (2014)

July 10, 2014

1 Overview

The main goal of PopGenReport is to simplify population genetic analysis in R. We do this by providing an easy to use entry point for conducting basic population genetic analyses in R. A basic workflow consists of only two steps:

1. Load your data into R from an existing data file that is formatted for some of the more common genetic analysis programs (e.g. Genepop, STRUCTURE, FSTAT, GENETIX) or as a simple CSV file.

2. Use the function `popgenreport` to generate a report containing a basic population genetic analysis of your dataset in an easy to read PDF format.

Output from the `popgenreport` function is provided in a number of formats including: a PDF report containing the complete analysis, each figure and a map of sampling locations is provided in PDF, PNG, and SNG file formats, tables are provided as CSV files. Additionally the R code that was used to generate your analysis is provided as an output which can be used as the beginnings of a more sophisticated and customised analysis of your data.

2 Installing PopGenReport

The full potential of PopGenReport is only available if your computer is setup to create pdf's using LaTeX, thus we recommend installing LaTeX as well on your machine. We note that `PopGenReport` does work without LaTeX and is able to produce all of the figure, table and map output files, but without LaTeX you will not be able to create the pdf report containing your complete analysis (see example in Adamack & Gruber (under review)). If you decide you do not want to install LaTeX on your machine, just skip the parts on the LaTeX installation below.

Additionally, you have the choice of installing R (and the `PopGenReport` package) on your computer or using a mobile version of the package. The advantage of the mobile package is that you do not need to install LaTeX on your machine (many versions take more than a Gb of space), but you are still able to choose the PDF report as an output option. Moreover the mobile version is a good choice if you want to run R (and `PopGenReport`) on different computers and/or computers where you do not have admin rights.

2.1 Full installation, admin rights needed

To install R and LaTeX on your computer you need to:

1. Download R (latest version) `http://cran.r-project.org`

2. [optional] Download an R editor (e.g. R-studio `http://www.rstudio.com/ide`, Tinn-r `http://www.sciviews.org/Tinn-R`)

3. Download a Latex installation (e.g. we recommend MiKTeX for Windows `http://miktex.org` and MacTeX for OSX `http://www.tug.org/mactex/`)

4. Install them in the following order: R, [Rstudio], MiKTeX/MacTeX following the installation instructions provided by the developers of each of the programs.

2.2 Mobile installation (currently Windows only)

To facilitate the installation process of the mobile version we have compiled a single zip file and made it available via our website: `www.popgenreport.org`. You can download the zip file from our website (currently windows only) and extract it into a folder of your local storage device

(e.g. on a usb-drive, external hard drive or the internal hard drive on your computer). If you prefer, you can create your own mobile version by downloading R and any mobile version of LaTeX. To use the mobileversion follow the steps below:

1. Download the file popgenreport.zip from our website: `http://www.popgenreport.org`. It includes a running version of R (3.0.1) and MiKTeX (portable 2.9.4757).

2. Extract the PopGenPack.zip file to your usb-drive, external hard drive or a directory on your local hard drive (e.g. E:\on windows).

3. Link R and MiKTeX. To do this you need to tell R where it can find MiKTeX. The location of MiKTeX depends on where you unzipped your zipfile to. If you unzipped it to your local usb-drive (for example E:\on windows then there should be an PopGenPack folder and the full command needed to tell R the location of MiKTeX is:

```
Sys.setenv(PATH = paste(Sys.getenv("PATH"),
"E:\\PopGenPack\\Miktex\\miktex\\bin\\", sep=.Platform$path.sep))
```

2.3 Testing your installation

Now you can check to see if your installation was successful.

1. Start R (or your R editor) and install the latest version of the `PopGenReport` package, by typing the following code into the R-console:

```
# Uncomment by deleting the hash from the install.packages command
# Needs to be done only once then it is permanently on your computer.
#install.packages("PopGenReport", repos="http://cran.rstudio.com/")
require(PopGenReport)
```

2. To check the installation type:

```
#a short test
data(bilby)
summary(bilby)


##
##  # Total number of genotypes:  71
##
##  # Population sample sizes:
## ACT NSW QLD
##  18  30  23
##
##  # Number of alleles per locus:
## L1 L2 L3
##  4  6  5
##
##  # Number of alleles per population:
##  1  2  3
## 14 13 14
##
```

```
##  # Percentage of missing data:
## [1] 0
##
##  # Observed heterozygosity:
##     L1     L2     L3
## 0.7183 0.7465 0.7183
##
##  # Expected heterozygosity:
##     L1     L2     L3
## 0.7081 0.6908 0.7098
```

If the output is similar to the above then `PopGenReport` should be installed correctly and we can start with our first analysis.

3 A basic analysis

For our first analysis, we follow the simple two step workflow mentioned above:

1. load your data into R using various file formats (genepop, structure or a simple text or Excel csv file)
2. use the function `popgenreport` to analyse your data [and create a nicely layouted pdf of your analysis (optional)

3.1 Load your data

We will be using an example dataset that is provided with `PopGenReport`, but you can also use follow along using your own dataset if it has a similar layout to our example dataset (which you'll see shortly).

To use the example dataset that is provided with `PopGenReport` we first need to locate the file on your computer and then read it into R and then create a genind object which will be used for all of the subsequent analyses.

You can find the path to the example file that comes with PopGenReport via:

```
paste(.libPaths()[1],"/PopGenReport/extdata/platypus1c.csv", sep="" )
```

For demonstration purposes we will first load the file as a data.frame into R so that you can see the layout of the example datafile. We will then show you how to use the `read.genetable` function to convert the datafile into a genind object and provide some tips on preparing your datafile if your dataset is in a similar format.

```
platy <- read.csv(paste(.libPaths()[1],"/PopGenReport/extdata/platypus1c.csv", sep="" ))
head(platy)

##    ind   pop    lat  long  group age   loci1   loci2   loci3   loci4
## 1 T158 Black -40.87 145.3 Female juv 141/145 243/243 159/173 263/265
## 2 T306 Black -40.86 145.3   Male  Ad 145/145 243/243 159/159 265/267
## 3 T305 Black -40.88 145.3 Female  Ad 143/145 243/243 159/159 265/265
## 4 T148 Black -40.99 145.4   Male  Ad 141/141 243/243 159/159 265/265
## 5 T149 Black -40.99 145.4 Female  Ad 141/145 243/245 159/171 265/267
## 6 T106  Brid -41.23 147.5   Male  Ad 145/147 243/245 159/171 267/267
```

```
##     loci5   loci6
## 1 215/215 192/192
## 2 219/219 192/194
## 3 215/215 194/194
## 4 215/219 192/194
## 5 215/215 194/196
## 6 215/215 192/192
```

Knowing the location of the example file, if you want, you can now explore it in more detail using a spreadsheet program such as Excel or Calc (OpenOffice) or any other program that is capable of opening an ascii file.

The first row of the file is a title row. Each of the following rows contains the data for a single sample/individual. The first column contains a unique identifier for the sample/individual. The title for this column must be "ind". The second column (optional, but strongly recommended) is the population that the animal belongs to. The title for the population column (if it is used) must be "pop". Separate columns are used for the latitude and longitude coordinates (given in decimal degrees) of each individual (optional) and there columns (if used) should be titled 'lat' and 'long'. As an optional alternative, Mercator coordinates (or grid points) can be used for individual spatial coordinates. If this option is used, the column titles should be 'x' and 'y'. Additional (optional) observations such as gender, age, phenotype, etc. can be placed in a contiguous block of columns. You are free to choose the column titles for these columns, but avoid using spaces and symbols in these titles as they have a tendency to create problems. Finally, comes the genetic data. Here you can use a single column per locus (with each of the alleles seperated by a defined separator ¡not commas¿) or in the format of a single allele per column). The column titles for the genetic data are up to you, but again we discourage the use of spaces and punctuation marks in the column titles.

It is **very, very** important follow these formatting instructions if you want to import your own data. Keep the spelling and case of the column titles ('ind', 'pop', 'lat', 'long', 'x', and 'y') consistent with the formatting shown in the examples. Additionally, provide your data in the same order of columns. The number of columns can vary (e.g. if you provide more or less additional information about your samples or if you have more or fewer loci). There is some error proofing in the import function, but it is best (easiest) to stick with the example format as closely as possible. From personal experience, if your dataset isn't being imported correctly, it is most likely because you have made the mistake of doing something like typing 'latitude', 'Lat', or 'LAT' instead of 'lat'.

If you have already formatted your data for use with programs like STRUCTURE, FSTAT, Genepop, or Genetix, you can import them without having to do any additional formatting by using the helper functions (e.g. `read.structure`) provided by `PopGenReport`, `adegenet`, and `pegas`. For more information please refer to the import functions implemented in the adegenet package. A shortcut is to type `?read.genetable` and then explore the links provided in the "See also" section.

ind	pop	lat	long	group	age	loci1	loci2	loci3	loci4	loci5	loci6
T158	Black	-40.87	145.28	Female	juv	141/145	243/243	159/173	263/265	215/215	192/192
T306	Black	-40.86	145.28	Male	Ad	145/145	243/243	159/159	265/267	219/219	192/194
T305	Black	-40.88	145.29	Female	Ad	143/145	243/243	159/159	265/265	215/215	194/194
T148	Black	-40.99	145.38	Male	Ad	141/141	243/243	159/159	265/265	215/219	192/194

Table 1: Example of a correctly formatted data set in a spreadsheet calculator (e.g. Excel). Please make sure to use the exactly same headings, such as 'ind', 'pop', ['lat', 'long' optional] for the first columns and use unique identifiers for your allel or loci headings. Save your data set as csv file. Please be aware to use a ', as separator. In some countries Excel uses ';' as default (e.g. Germany).

Now that we have a data file in the right format, we shall import it and do our first analysis. To do this we use the `read.genetable` function.

```
platy.gen <- read.genetable( paste(.libPaths()[1], "/PopGenReport/extdata/platypus1c.csv"
, sep="" ), ind=1, pop=2, lat=3, long=4, other.min=5, other.max=6, oneColPerAll=FALSE,
sep="/", ploidy=2)

platy.gen  #to check the data.

##
##    ######################
##    ### Genind object ###
##    ######################
## - genotypes of individuals -
##
## S4 class:  genind
## @call: df2genind(X = genes, sep = sep, ncode = ncode, ind.names = inds,
##     loc.names = colnames(genes), pop = pops, missing = missing,
##     ploidy = ploidy)
##
## @tab:  13 x 20 matrix of genotypes
##
## @ind.names: vector of  13 individual names
## @loc.names: vector of  6 locus names
## @loc.nall: number of alleles per locus
## @loc.fac: locus factor for the  20 columns of @tab
## @all.names: list of  6 components yielding allele names for each locus
## @ploidy:  2
## @type:  codom
##
## Optional contents:
## @pop:  factor giving the population of each individual
## @pop.names:  factor giving the population of each individual
##
## @other: a list containing: latlong  data
```

The read.gentable function has some arguments that need to be explained. First you need to provide the path to the file you want to load. Please be aware the R uses forward slashes '/' as a seperator to specify a path. So if your file is in a folder say "d:/temp/" then the first argument in read.genetable would be:

```
read.genetable('D:/temp/mydata.csv')
```

The following arguments specify the column number of specific entries (e.g. ind, lat and long). Some arguments are optional and some are compulsory. For a detailed description please refer to the help pages of the `?read.genetable` function. In brief

argument	description	optional
ind	**unique** identifier of individuals	no
pop	indentifier of subpopulations optional	yes
lat, long	coordinates in WGS84 (latitude and longitude)	yes
x,y	coordinates in Mercator projection (`?Mercator`)	yes
other.min, other.max	min to max column number with additional data	yes
oneColPerAll	are the alleles coded using a single or multiple columns	no
sep	character if alleles are within one column	yes
ncode	integer for the number of characters to code one genotype	yes
ploidy	integer indicating the degree of ploidy of the genotypes	yes
missing	default NA, char for missing data, check help page for options	yes

Table 2: arguments of `read.genetable`

In our data set the first column was unique identifiers ('ind'; ind=1), the second was population ('pop'; pop=2), coordinates were provided as latitidues and longitudes ('lat' and 'long') in columns 3 and 4 (lat=3, long=4), then we had some additional data in columns 5 and 6, (other.min=5, other.max=6) and the remainder of the columns were for our genotypes. They are coded as oneColPerAll=FALSE and therefore we need to specify a seperator "/". ncode is obvious but we could have specified it as being 3 (as we use 3 digits to code alleles) and finally ploidy=2 and missing was not specified (as we have no missing data, so the default NA is fine).

Again, it is **very, very** important follow the formatting requirements closely when you import your own data. Keep the headings exactly as shown in the examples and use the same column order (though not necessarily number of columns). The import function has some error proofing built in with respect to the formatting, but it is best to stick as closely to the the example format as possible.

3.2 Analysing your data

It is a good idea to check if your data import was successful. When you are first checking to see that your data has been imported correctly, there are a few details you can quickly check by typing type the name of the object `platy.gen` and then checking some of the entries. First, for the line @tab does the first number match the number of individuals in your data set? If the answer is yes then good, otherwise something didn't work right. The second number on the line is equivalent to the total number of alleles in your dataset. That may or may not be readily known beforehand, if you do know this number does it match? A second place to check is the line @loc.names. Does the size of the vector match up with the number of loci in your dataset? A third area to check is the lines for @pop and @pop.names. If these datasets are blank (in our example type `platy.gen@pop` and `platy.gen@pop.names`) it could indicate that there was a problem with importing your data (or you may not have provided population information for your data as it is an optional item). Finally, if you provided optional data such as latitudes and longitudes, ages, or phenotypic data etc., is that data listed on the line @other?

Once you are reasonably sure that the import was successful (or if something didn't go right, you can get an idea as to what went wrong) by looking at some of the details of your dataset. For example the number and names of individuals can be found in the `@ind.names` slot. If you have specified populations in your data set using the pop column we can check this using the `pop` function on our genind object.

```
platy.gen@ind.names  #check the number and names of individuals
```

```
##     01     02     03     04     05     06     07     08     09     10
## "T158" "T306" "T305" "T148" "T149" "T106" "T107" "T110" "T111" "T308"
##     11     12     13
## "T307" "T302" "T303"
```

```
pop(platy.gen) #similar to platy.gen@pop
```

```
##  [1] Black Black Black Black Black Brid  Brid  Brid  Brid  Cam   Cam
## [12] Cam   Cam
## Levels: Black Brid Cam
```

```
platy.gen@loc.names #names of all loci
```

```
##      L1      L2      L3      L4      L5      L6
## "loci1" "loci2" "loci3" "loci4" "loci5" "loci6"
```

```
platy.gen@loc.fac  #the number of allels per loci. e.g. the first locus has 4 alleles
```

```
##  [1] L1 L1 L1 L1 L2 L2 L3 L3 L3 L3 L4 L4 L4 L5 L5 L5 L6 L6 L6 L6
## Levels: L1 L2 L3 L4 L5 L6
```

```
table(platy.gen@loc.fac) # a better way to check the number of alleles per loci
```

```
##
## L1 L2 L3 L4 L5 L6
##  4  2  4  3  3  4
```

```
platy.gen@all.names #allele names for each loci
```

```
## $L1
##     1     2     3     4
## "141" "143" "145" "147"
##
## $L2
##     1     2
## "243" "245"
##
## $L3
##     1     2     3     4
## "159" "161" "171" "173"
##
## $L4
##     1     2     3
## "263" "265" "267"
##
## $L5
##     1     2     3
## "215" "217" "219"
##
## $L6
##     1     2     3     4
## "184" "192" "194" "196"
```

The genetic data for each individual is stored within the tab slot.

```
platy.gen@tab
```

```
##    L1.1 L1.2 L1.3 L1.4 L2.1 L2.2 L3.1 L3.2 L3.3 L3.4 L4.1 L4.2 L4.3 L5.1
## 01  0.5  0.0  0.5  0.0  1.0  0.0  0.5  0.0  0.0  0.5  0.5  0.5  0.0  1.0
## 02  0.0  0.0  1.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  0.5  0.5  0.0
## 03  0.0  0.5  0.5  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  1.0
## 04  1.0  0.0  0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.0  0.0  1.0  0.0  0.5
## 05  0.5  0.0  0.5  0.0  0.5  0.5  0.5  0.0  0.5  0.0  0.0  0.5  0.5  1.0
## 06  0.0  0.0  0.5  0.5  0.5  0.5  0.5  0.0  0.5  0.0  0.0  0.0  1.0  1.0
## 07  0.0  0.0  1.0  0.0  1.0  0.0  0.5  0.0  0.5  0.0  0.0  0.0  1.0  1.0
## 08  0.0  0.0  1.0  0.0  0.5  0.5  0.0  0.0  0.5  0.5  0.5  0.5  0.0  1.0
## 09  0.5  0.0  0.5  0.0  0.0  1.0  1.0  0.0  0.0  0.0  0.0  0.0  1.0  1.0
## 10  0.0  0.0  1.0  0.0  1.0  0.0  0.0  0.5  0.5  0.0  0.0  1.0  0.0  0.5
## 11  0.5  0.0  0.5  0.0  1.0  0.0  0.0  0.0  0.5  0.5  0.0  1.0  0.0  0.5
## 12  0.5  0.0  0.5  0.0  1.0  0.0  0.0  0.0  1.0  0.0  0.0  0.5  0.5  1.0
## 13  1.0  0.0  0.0  0.0  1.0  0.0  0.5  0.0  0.0  0.5  0.0  0.5  0.5  1.0
##    L5.2 L5.3 L6.1 L6.2 L6.3 L6.4
## 01  0.0  0.0  0.0  1.0  0.0  0.0
## 02  0.0  1.0  0.0  0.5  0.5  0.0
## 03  0.0  0.0  0.0  0.0  1.0  0.0
## 04  0.0  0.5  0.0  0.5  0.5  0.0
## 05  0.0  0.0  0.0  0.0  0.5  0.5
## 06  0.0  0.0  0.0  1.0  0.0  0.0
## 07  0.0  0.0  0.0  1.0  0.0  0.0
## 08  0.0  0.0  0.0  1.0  0.0  0.0
## 09  0.0  0.0  0.0  1.0  0.0  0.0
## 10  0.5  0.0  0.5  0.0  0.0  0.5
## 11  0.0  0.5  0.5  0.5  0.0  0.0
## 12  0.0  0.0  0.0  0.5  0.0  0.5
## 13  0.0  0.0  0.0  0.5  0.5  0.0
```

For a full description please refer to the adegent package (`?genind`) and check out some of the tutorial vignettes provided at `adegenet.r-forge.r-project.org`. In short each locus has its own column. As there are 4 alleles at the first loci, the first 4 columns are named L1.1 to L1.4 and belong to the first locus. The first individuals had the two alleles, 141 and 145, which correspond to alleles 1 and allele 3 at the first loci. Hence the first individual has an entry of 0.5 at L1.1 and L1.3. You can easily see that the second individuals is homozygote at the first loci (the entry was 145/145)

One big advantage of using the `read.genetable` function is the ability to include spatial data directly in a genind object. Spatial coordinates are necessary for a number of analysis such as spatial autocorrelation and for producing a map of your data. As the original genind object did not have a slot for spatial data, we used the "other" slot to include additional data such as coordinates for each individual (latlong or xy).

There are several items listed in the "other" slot. We can access the data that was stored in our additional columns and our coordinates here.

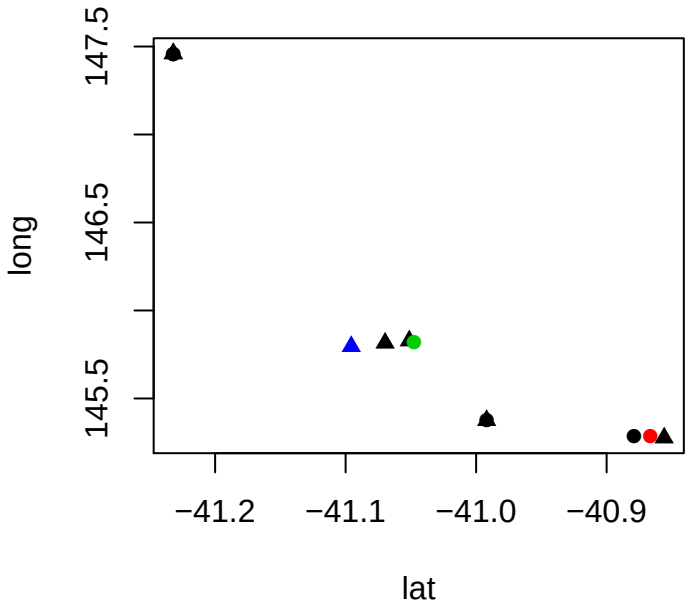
```
str(platy.gen@other)  #lists the content of the slot
```

```
## List of 2
##  $ latlong:'data.frame': 13 obs. of  2 variables:
```

```
##   ..$ lat : num [1:13] -40.9 -40.9 -40.9 -41 -41 ...
##   ..$ long: num [1:13] 145 145 145 145 145 ...
##  $ data   :'data.frame': 13 obs. of  2 variables:
##   ..$ group: Factor w/ 2 levels "Female","Male": 1 2 1 2 1 2 2 1 1 2 ...
##   ..$ age  : Factor w/ 4 levels "Ad","juv","Juv",..: 2 1 1 1 1 1 1 1 1 4 ...
```

We can produce a very crude plot of our sampling locations using the coordinates and set the symbols and colors of our individuals using the age and group columns.

```
plot(platy.gen@other$latlong, pch=as.numeric(platy.gen@other$data$group)+15, col=platy.gen@
```



Now that we are confident that the our dataset has been imported data was successful let's start our first population genetics analysis.

We would first like to get a simple overview on our genotypes, for example how many alleles are there in our dataset, how many individuals are there in each population sample, how many alleles are there in each population, etc. To do this we can use our "work horse" function `popgenreport`. The function has a lot of arguments (you can check them via the help page`?popgenreport`). Except for the first argument, all of the arguments are set to a default value and can be omitted if we want. The function's first argument is simply the name of our newly created genind object, which contains all of the data needed to do our analysis. To specify which analyses are performed, we use the `mk` switches (for a description of all of the available switches see the table below or check ?popgenreport). As we only want to some basic counts from our data set, we set the `mk.counts` switch to `TRUE`. To make it easier to find the results we specify an additional argument, namely the path.pgr argument, which specifies the main file path, where all of our results will be saved. We **STRONGLY RECOMMEND** that your main file path not have any spaces or punctuation marks in the folder names. If you do have spaces or punctuation marks in your folder names, the function will try to replace them with underscores and may create new folders on your drive if you have admin rights on that drive. If you don't have admin rights, the `popgenreport` function may fail to successfully run. Ommitting the path argument will cause your results to be saved in a temporary folder

that may be erased once you close your R session. Please specify an already existing folder (where you have write access) otherwise you will get an error message. Be aware if you run popgenreport for the first time, that your LaTeX installation may need to download additional packages (MiKTeX and other LaTeX installations trigger these downloads automatically). For this you need an internet connection (only once) and also some patience in case the response of the server is slow. **Please note: Due to technical reasons we always set** `mk.pdf=FALSE` (otherwise the vignette would not compile) when we run the function `popgenreport` during the tutorial (the technical reason is this manual is a vignette and you cannot run knitr within knitr, without causing errors...). If you want to have a pdf report set `mk.pdf=TRUE` in the examples.

```
### for technical reasons we set here mk.pdf=FALSE
### please set mk.pdf=TRUE if you want to have a report !!!!!!
platy.out1 <- popgenreport(platy.gen, mk.counts=TRUE, mk.pdf=FALSE,
                           foldername="platy")

## There is no  platy  folder. I am trying to create it;
## otherwise please create the folder manually.
## Compiling report...
## - General summary...
## Analysing data ...
## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/platy
```

If you did not install LaTeX on your system, you will get an error message that a pdf cannot be created. As was mentioned previously, you still can run `popgenreport` but you will need to specify that you do not want to create a PDF report by setting the switch `mk.pdf=FALSE`.

```
platy.out1 <- popgenreport(platy.gen, mk.counts=TRUE, mk.pdf=FALSE,
                           foldername="platy")

## Compiling report...
## - General summary...
## Analysing data ...
## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/platy
```

In both cases `popgenreport` creates an object called `platy.out1` that has all the results for this analysis. To get an overview of our results, we type the `summary` command.

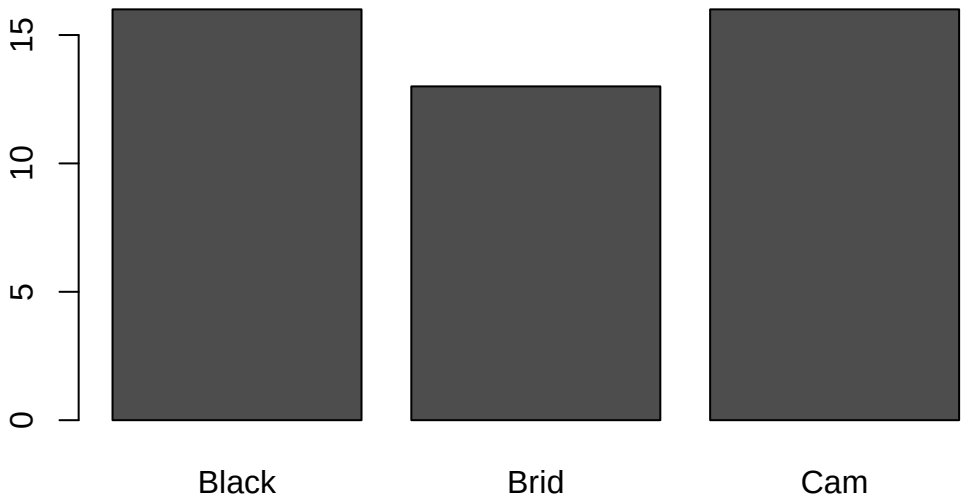
```
summary(platy.out1)

##        Length Class  Mode
## counts 7      -none- list

platy.out1

## $counts
## $counts$N
## [1] 13
##
## $counts$numbypop
##        Black Brid Cam
```

```
## Number     5    4   4
##
## $counts$totalleles
## [1] 20
##
## $counts$nallelesbypop
##                          Black Brid Cam
## Total number of alleles    16   13  16
##
## $counts$nallelesbyloc
##              loci1 loci2 loci3 loci4 loci5 loci6
## # of alleles     4     2     4     3     3     4
##
## $counts$meanalleles
## [1] 3.3
##
## $counts$missing
## [1] 0
```

So there are 7 objects in this list. To look at the content in more detail just type the name `platy.out1`.

We can use the object `platy.out1` to create some new outputs. For example we can plot the total number of alleles per population using a barplot.

```
barplot(platy.out1$counts$nallelesbypop)
```



Obviously this can be a bit tedious and most of these analysis are fairly simple. Thus, we developed the `popgenreport mk.counts` analysis to automate the process of doing these simple analyses. The function creates a suit of figures and tables that can be inspected in a subfolder called "results" (e.g. D:/Analysis/results if we specified D:/Analysis as path) . This folder will

be created if it does not exist so make sure you have write access in the specified path. If not path is specified the tempdir() folder of R is used. If you did install LaTeX there will also be a PDF report of your analysis with some explanations of what the different figures and tables show.

Using R, you can check the content of your results folder by typing:

```
dir(paste(tempdir(),"platy",sep=.Platform$file.sep))
```

```
##  [1] "PopGenReport-n_alleles_per_locus.pdf"
##  [2] "PopGenReport-n_alleles_per_locus.png"
##  [3] "PopGenReport-n_alleles_per_locus.svg"
##  [4] "PopGenReport-num_alleles_locus.csv"
##  [5] "PopGenReport-num_samp_by_loc.csv"
##  [6] "PopGenReport-pop_sampsz_vs_alleles.pdf"
##  [7] "PopGenReport-pop_sampsz_vs_alleles.png"
##  [8] "PopGenReport-pop_sampsz_vs_alleles.svg"
##  [9] "PopGenReport-tot_alleles_by_loc.csv"
## [10] "PopGenReport.rnw"
## [11] "PopGenReport.tex"
```

Feel free to inspect the report PopGenReport.pdf and all the files created using your prefered file explorer. The next section shows the pdf report you have just created.

3.3 PDF report created by `popgenreport` using `mk.counts=TRUE`

A Population Genetic Report

using PopGenReport Ver. 2.0

Adamack & Gruber (2014)

July 10, 2014

Contents

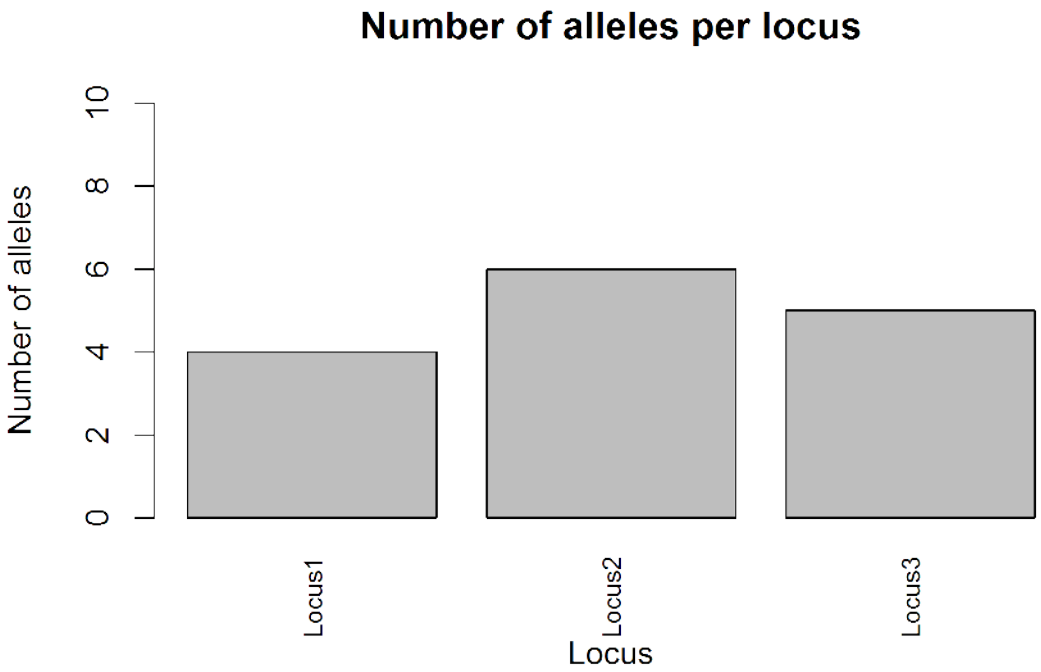
1 Counts **2**

1 Counts

This analysis looks at 71 individuals.

The mean number of alleles per locus (across all locations): 5

The percentage of missing data was 0%.



The individuals were sampled from the following locations in the following numbers:

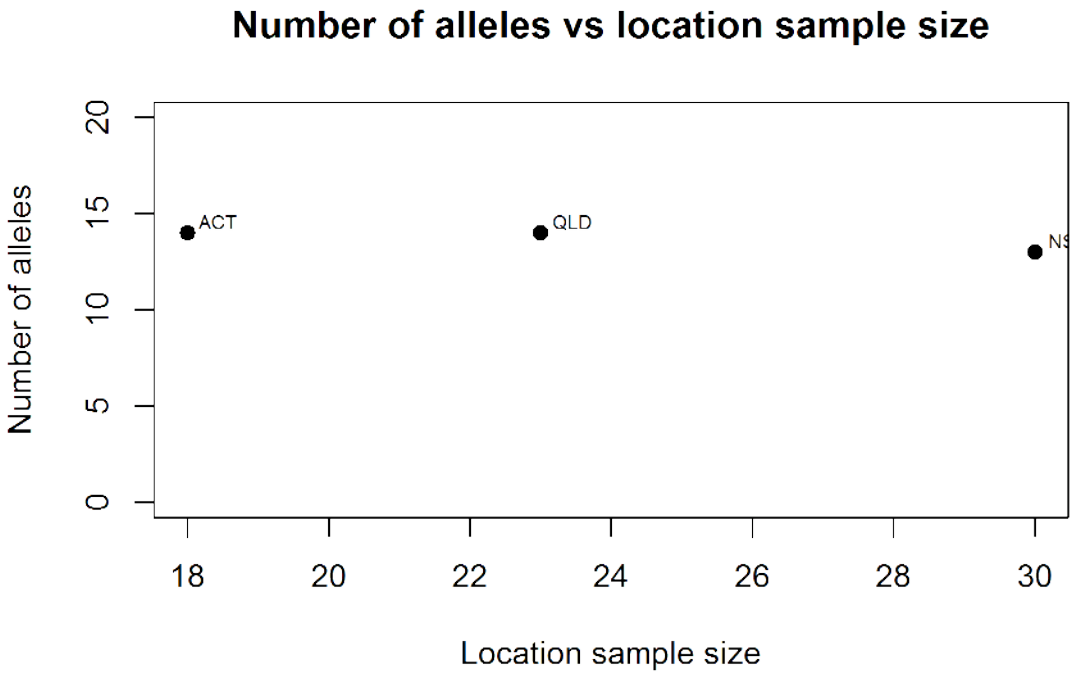
population	ACT	NSW	QLD
# ind	18	30	23

Table 1: Number of individuals per population

The total number of alleles sampled across all locations was 15
The total number of alleles seen in each sampling location was:

population	ACT	NSW	QLD
# alleles	14	13	14

Table 2: Number of alleles per population



The number of alleles per locus (across all subpopulations):

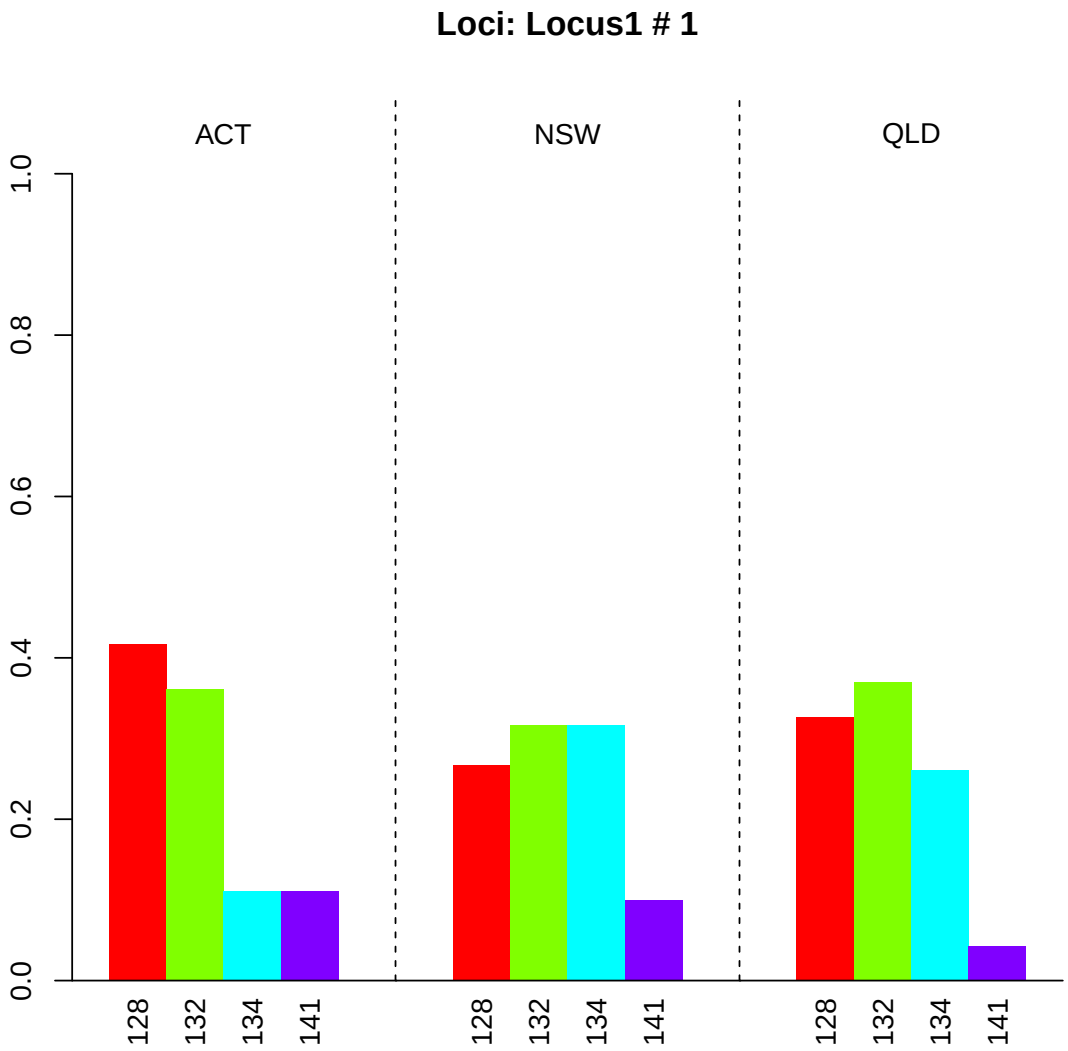
locus	Locus1	Locus2	Locus3
# alleles	4	6	5

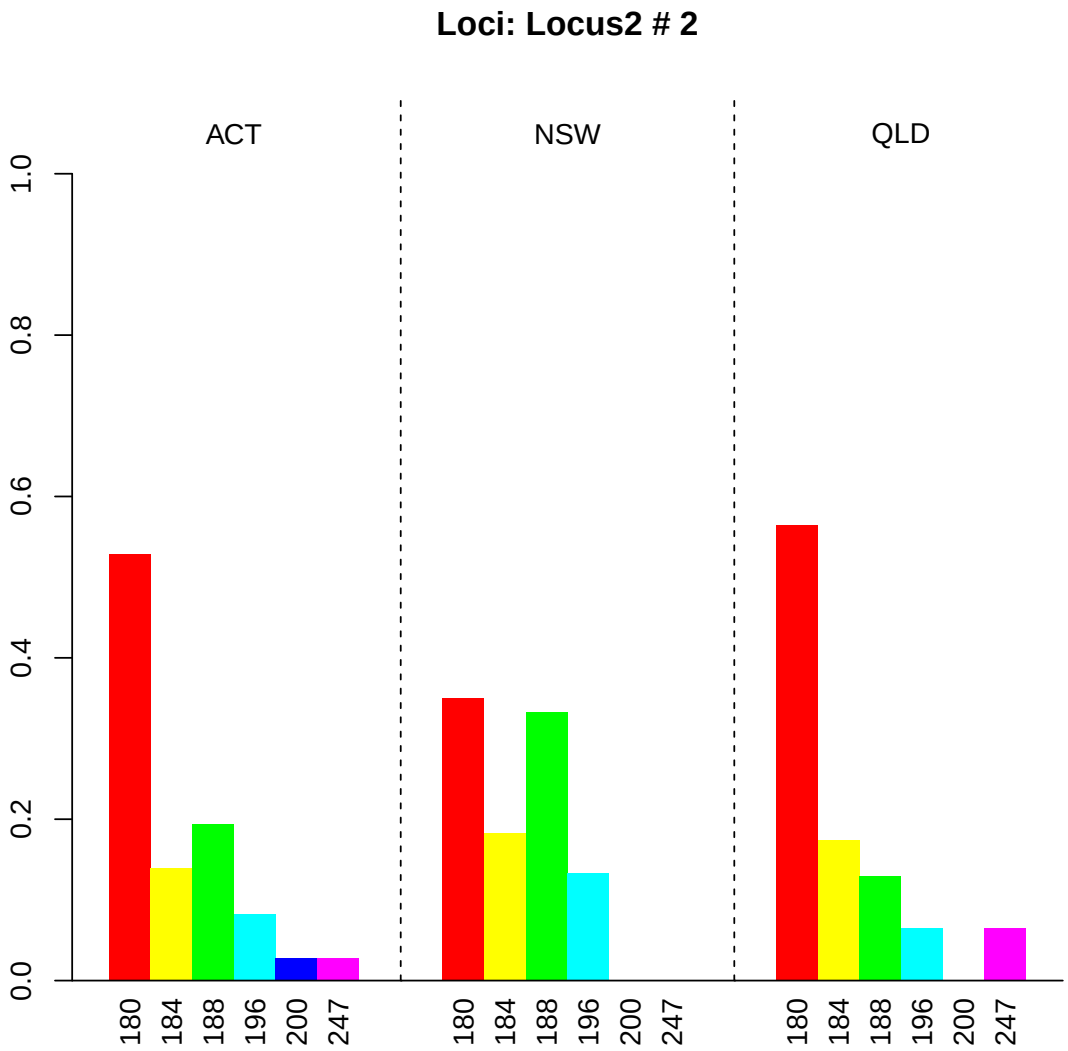
Table 3: Number of alleles per locus across all subpopulations

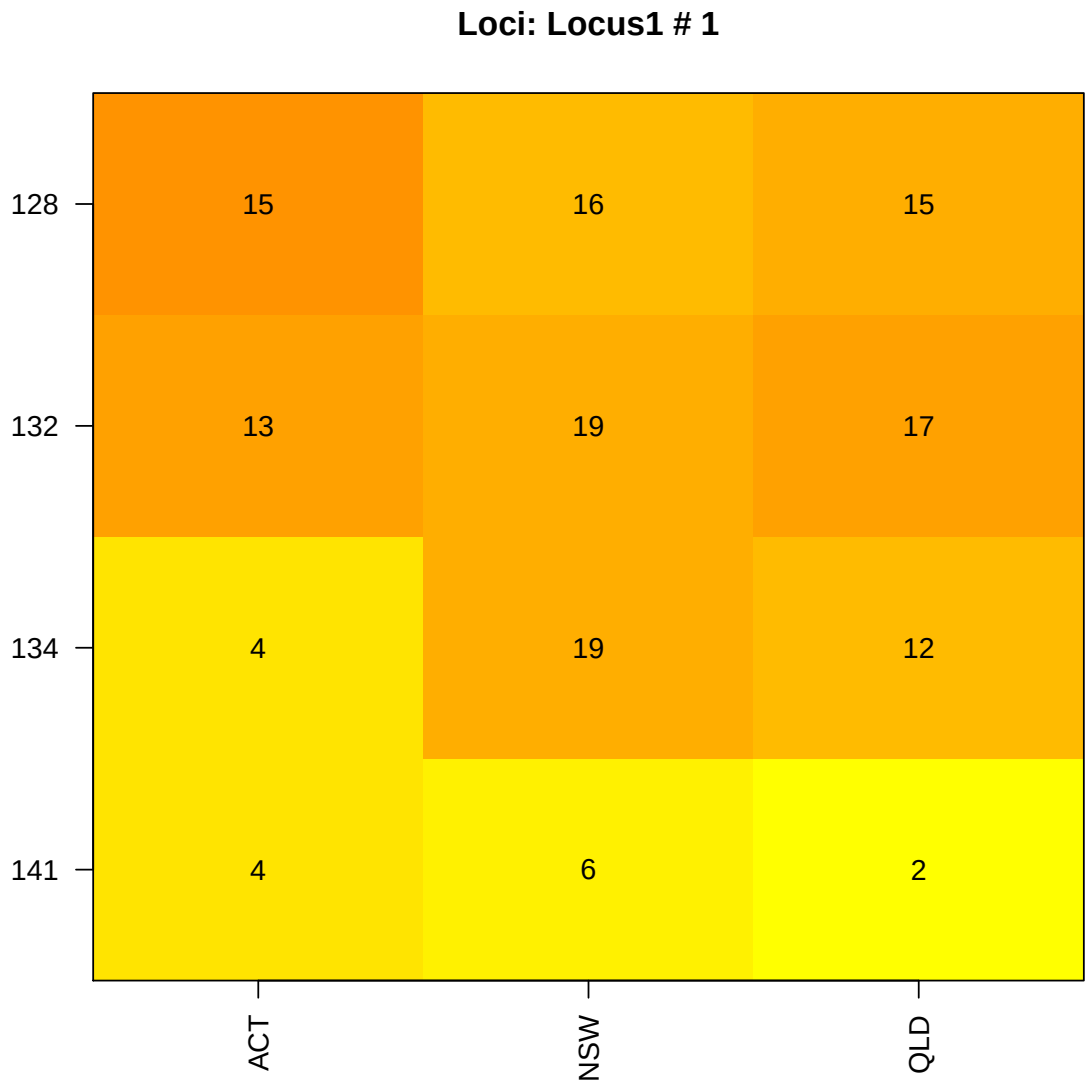
4 A complete analysis

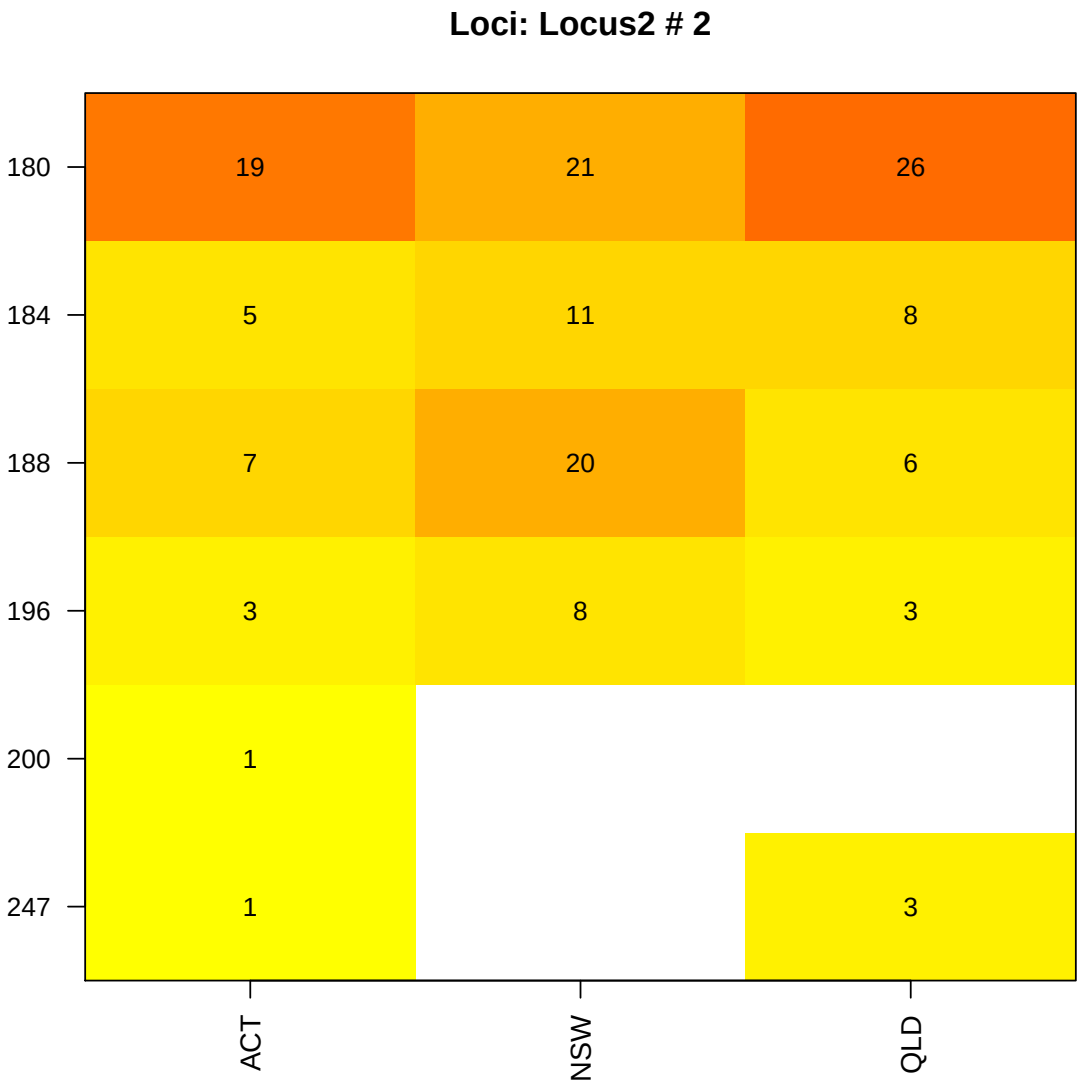
Now that you have learned the necessary steps to do a simply analysis, we will now create a full report, that demonstrates all of the standard outputs produced by popgenreport. Instead of using the `mk.counts=TRUE` switch we will instead use the `mk.complete=TRUE` to create a full report. To make it a bit more interesting we will use a bigger data set that comes with `popgenreport`. We load this data set using the data function as it is included with the `PopGenReport` pacakge as a saved genind object. In addition we would like to create an R file that shows how the whole analysis was done. The R file can be used as a start for performing a customised analysis.

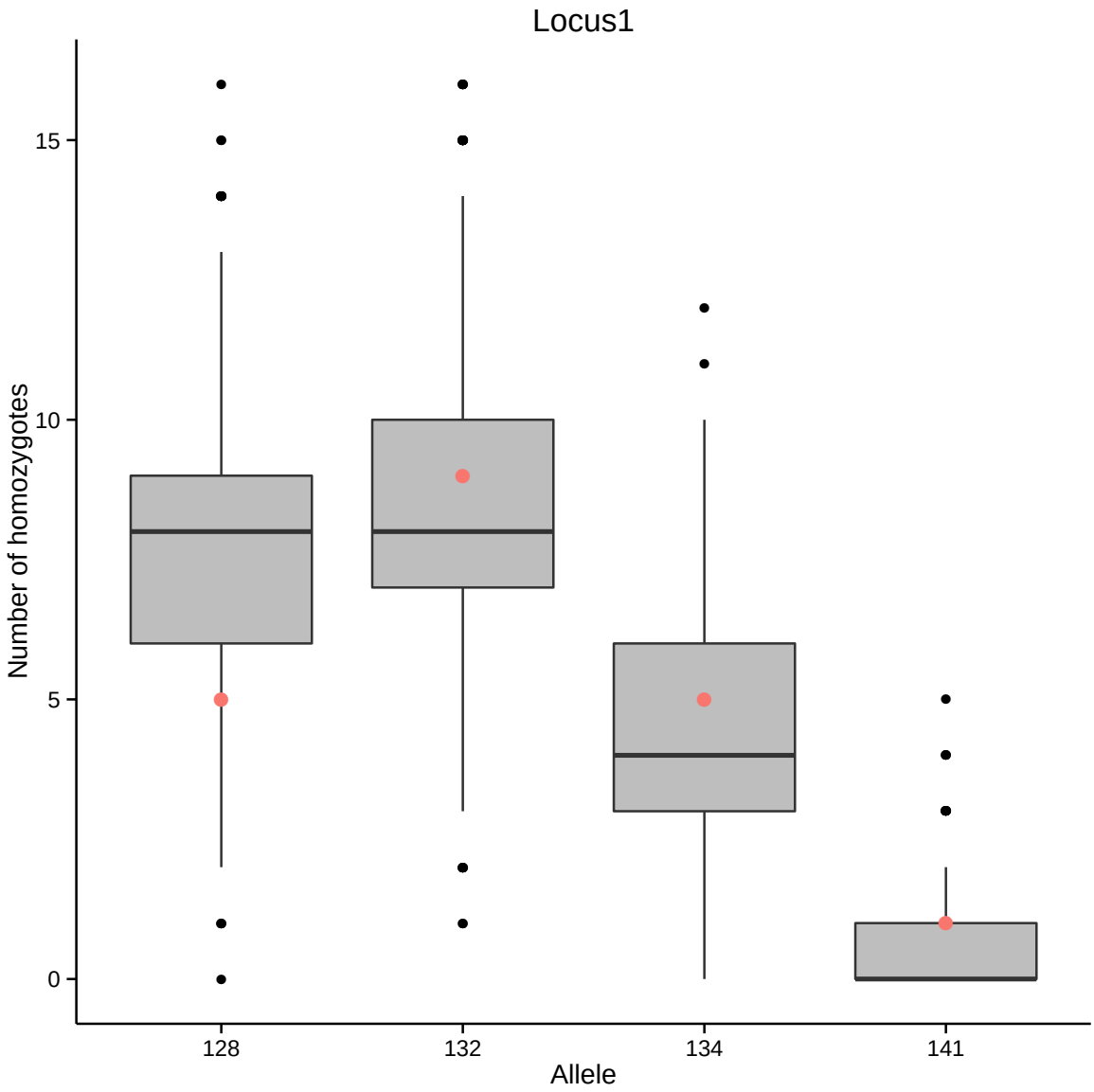
```
data(bilby)
### for technical reasons we set here mk.pdf=FALSE
### please set mk.pdf=TRUE if you want to have a report !!!!!!
bilby.complete <- popgenreport(bilby, mk.complete=TRUE, mk.Rcode=TRUE,
                               mk.pdf=FALSE)

## Compiling report...
## - General summary...
## - Map of individuals...
## - Statistics on population heterogeneity ...
## - Allelic distances ...
## - Pairwise Fst ...
## - Checking for null alleles ...
## - Allelic richness ...
## - Pairwise differentiations ...
## - Test for Hardy-Weinberg-Equilibrium ...
##   !! You may get warnings when running HWE tests, if the test is based
##  on an entry in the chi-square table which is less than five.!!
## - Kosman & Leonard 2005 genetic distances...
## - Smouse & Peakall 1999 genetic distances...
## - Spatial autocorrelation following Smouse & Peakall 1999 ...
## - Principal coordinate analysis following Jombart et al. 2009...
## Analysing data ...
```

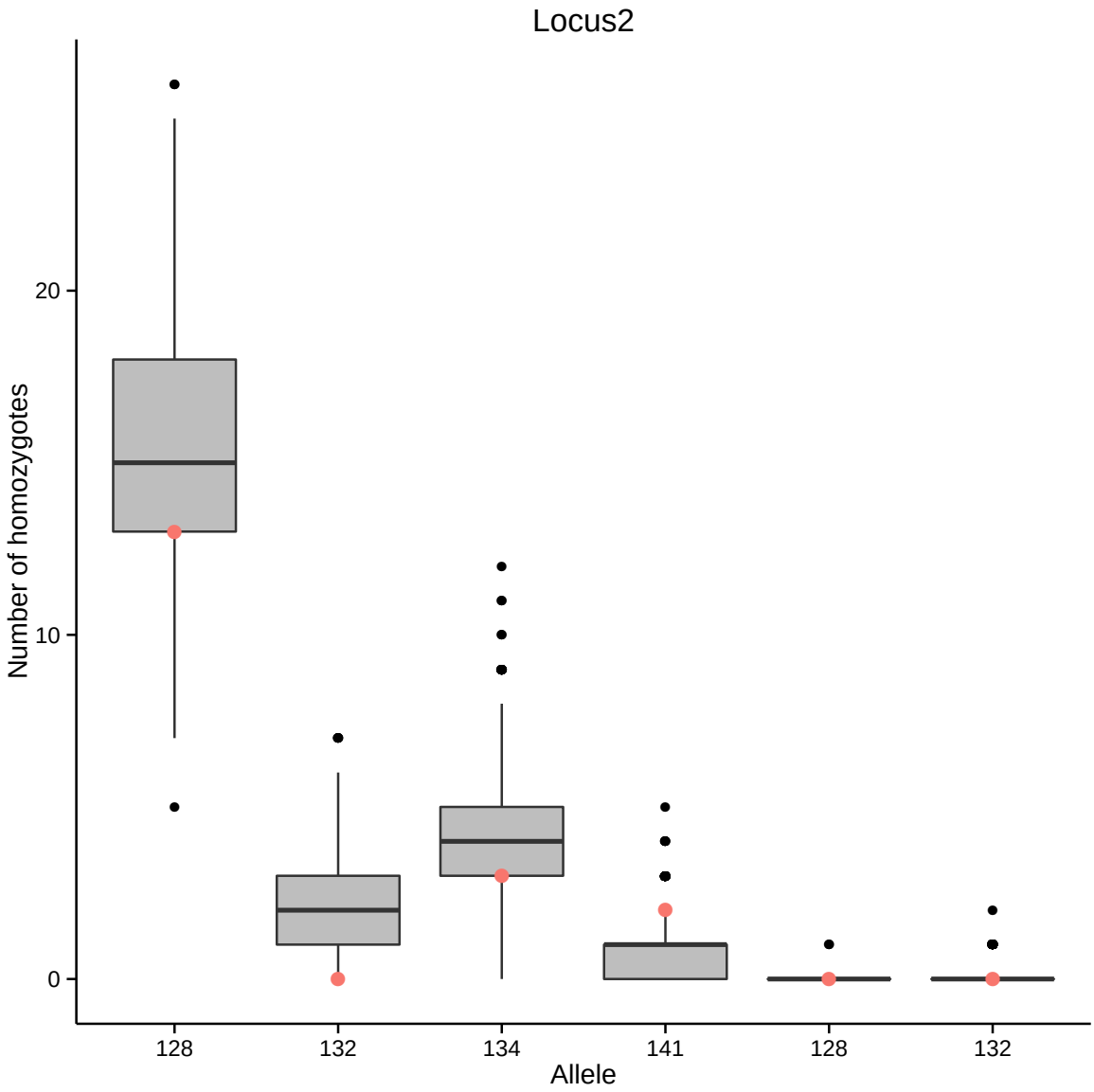

Loci: Locus1 # 1
ACT
NSW
QLD
1.0
0.8
0.6
0.4
0.2
0.0
128
132
134
141
128
132
134
141
128
132
134
141


Loci: Locus2 # 2
ACT
NSW
QLD
0.0
0.2
0.4
0.6
0.8
1.0
180
184
188
196
200
247


Loci: Locus1 # 1
128
132
134
141
15
16
15
13
19
17
4
19
12
4
6
2
ACT
NSW
QLD


Loci: Locus2 # 2
180
184
188
196
200
247
19
21
26
5
11
8
7
20
6
3
8
3
1
1
3
ACT
NSW
QLD


Locus1
Number of homozygotes
15
10
5
0
128
132
134
141
Allele




```
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect
## Warning:  Chi-squared approximation may be incorrect

## Creating R code from:  PopGenReport.rnw ...

## processing file:  PopGenReport.rnw
## output file:  PopGenReport.R

## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/results
```

Again you can find the output created in the specified folder. The full report on the bilby data set is attached to end of this pdf.

The table below gives an overview on the currently implemented analyses.

switch	description
mk.counts	provide overview counts of samples, loci, and alleles
mk.map	produce a map of individual locations; colours and point shapes can be customised
mk.lozihz	calculate observed and expected heterozygosity
mk.hwe	test for Hardy-Weinberg equilibrium
mk.fst	calculate F-statistics
mk.gd.smouse	individual pairwise genetic distances based on Smouse & Peakall (1999)
mk.gd.kosman	individual pairwise genetic distances based on Kosman & Leonhard (2005)
mk.pcoa	Principal component analysis following Jombart et al. (2009)
mk.spautocor	Spatial autocorrelation analysis following Smouse & Peakall (1999)
mk.allele.dist	allele distributions by loci and subpopulation; identify any private alleles
mk.null.all	check for null alleles
mk.allel.rich	allelic richness following El Mousadik and Petit (1996)
mk.differ.stats	population differentiation statistics (Nei's Gst, Hedrick's Gst, and Jost's D) from `mmod`
mk.custom	create your own analysis by editing the file custom.snw
mk.Rcode	to get the full R script that is used to generate the report
mk.complete	which you can customize for your analytical needs.
mk.pdf	create a PDF report

Table 3: arguments of `popgenreport`

4.1 Customise an analysis

4.1.1 Running subsets of data set

In addition to the report you also get all of the results back in a single large object from which you can extract data in order to perform more sophisticated and customised analyses or to simply create graphs that better match your needs.
When looking at the PDF report we see that there isn't much differentiation between the populations (e.g. pairwise Fst is quite low). It might be interesting to separate the analysis by sex and then check pairwise Fst for each sex seperately. To do this we need to run the popgenreport function for the each subset seperately.

```
### for technical reasons we set here mk.pdf=FALSE
### please set mk.pdf=TRUE if you want to have a report !!!!!!
bilby.fem <- popgenreport(bilby[bilby@other$sex=="Female"],
path.pgr=tempdir(), fname="Female", mk.fst=TRUE, mk.counts=FALSE,mk.pdf=FALSE)

## Compiling report...
## - Pairwise Fst ...
## Analysing data ...
## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/results

### for technical reasons we set here mk.pdf=FALSE
### please set mk.pdf=TRUE if you want to have a report !!!!!!
bilby.mal <- popgenreport(bilby[bilby@other$sex=="Male"],
path.pgr=tempdir(), fname="Male", mk.fst=TRUE, mk.counts=FALSE,mk.pdf=FALSE)
```

```
## Compiling report...
## - Pairwise Fst ...
## Analysing data ...
## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/results
```

We would like to see if our pairwise Fst for males are always higher than the ones for females. In our objects we find the pairwise Fsts for each sex in the slot: fstFSTpairwise. We then compare the two sets of pairwise Fsts by eye and we can see that the pairwise Fsts for males are generally lower (except in the pair NSW-ACT)

```
bilby.mal

## $fst
## $fst$FSTbyloci
##             Fit       Fst      Fis
## Locus3 -0.07639 -0.012227 -0.06339
## Locus2 -0.07402 -0.005745 -0.06788
## Locus1  0.10927  0.032200  0.07964
##
## $fst$FSTpairwise
##          ACT     NSW QLD
## ACT 0.000000      NA  NA
## NSW 0.035192 0.00000  NA
## QLD 0.008898 0.01805   0

bilby.fem

## $fst
## $fst$FSTbyloci
##             Fit       Fst      Fis
## Locus1 -0.17564 -0.039174 -0.13132
## Locus3  0.05915 -0.003187  0.06214
## Locus2 -0.08195  0.021151 -0.10533
##
## $fst$FSTpairwise
##         ACT     NSW QLD
## ACT 0.00000      NA  NA
## NSW 0.01271 0.00000  NA
## QLD 0.02143 0.02738   0

bilby.mal$fst$FSTpairwise> bilby.fem$fst$FSTpairwise

##       ACT   NSW   QLD
## ACT FALSE    NA    NA
## NSW  TRUE FALSE    NA
## QLD FALSE FALSE FALSE
```

4.1.2 Reprojecting spatial data and attaching it to a genind object

It may be that you have already imported your data into another program (e.g. structure) and you simply want to run popgenreport with it to produce a map of your study area. To

do this you will need to somehow attach your coordinates to your data set. We start with the usual situation that we have a data set and import it into R via `read.genetable`. This time the dataset is for Tasmanian tigers. There are rumors that a few Tasmanian tigers still roam across portions of the mainland of Australia and scats were collected in an attempt to detect their presence. You were asked to analyse the microsatellite data and produce a map from the sampling locations. Unfortunately the coordinates are not in the required geographic projection (e.g. latitudes and longitudes in decimal degrees), but instead were projected using the UTM system (UTM zone 55S). We first need to reproject the data so that `PopGenReport` is able to produce a map. To do this, we first load the data. Again we use a data set provided via the `PopGenReport` package and load it via the `read.genetable` function.

```
#### convert from utm 55S to latlong WGS84

tiger.gen <- read.genetable( paste(.libPaths()[1],"/PopGenReport/extdata/tiger.csv",
sep="" ), ind=1, pop=2, other.min=3, other.max=6, oneColPerAll=TRUE)


require(rgdal) #load package
xy <- as.matrix(tiger.gen@other$data[,2:1])  #y first then x
#projection from utm 55S to latlong WGS84
latslongs <-project(xy,
"+proj=utm +zone=55 +south +ellps=WGS84 +datum=WGS84 +units=m +no_defs",inv=T)

#add it to tiger.gen at the right place (again lats)
tiger.gen@other$latlong <- latslongs[,2:1] #again lat and then long
```

Now we attach the reprojected coordinates to `tiger.gen` and make our customised map. For all the available options, please check the `popgenreport` help page.

```
#customised map
### for technical reasons we set here mk.pdf=FALSE
### please set mk.pdf=TRUE if you want to have a report !!!!!!
popgenreport(tiger.gen, mk.map=TRUE,mk.counts=FALSE, path.pgr=tempdir(),
mapdotcolor="orange", mapdotsize=as.numeric(tiger.gen@other$data$sex),
maptype="roadmap", mapdotalpha=0.9,mk.pdf=FALSE)

## Compiling report...
## - Map of individuals...
## Analysing data ...
## All files are available in the folder:
## C:\Users\s425824\AppData\Local\Temp\RtmpKcPopj/results
## list()
```

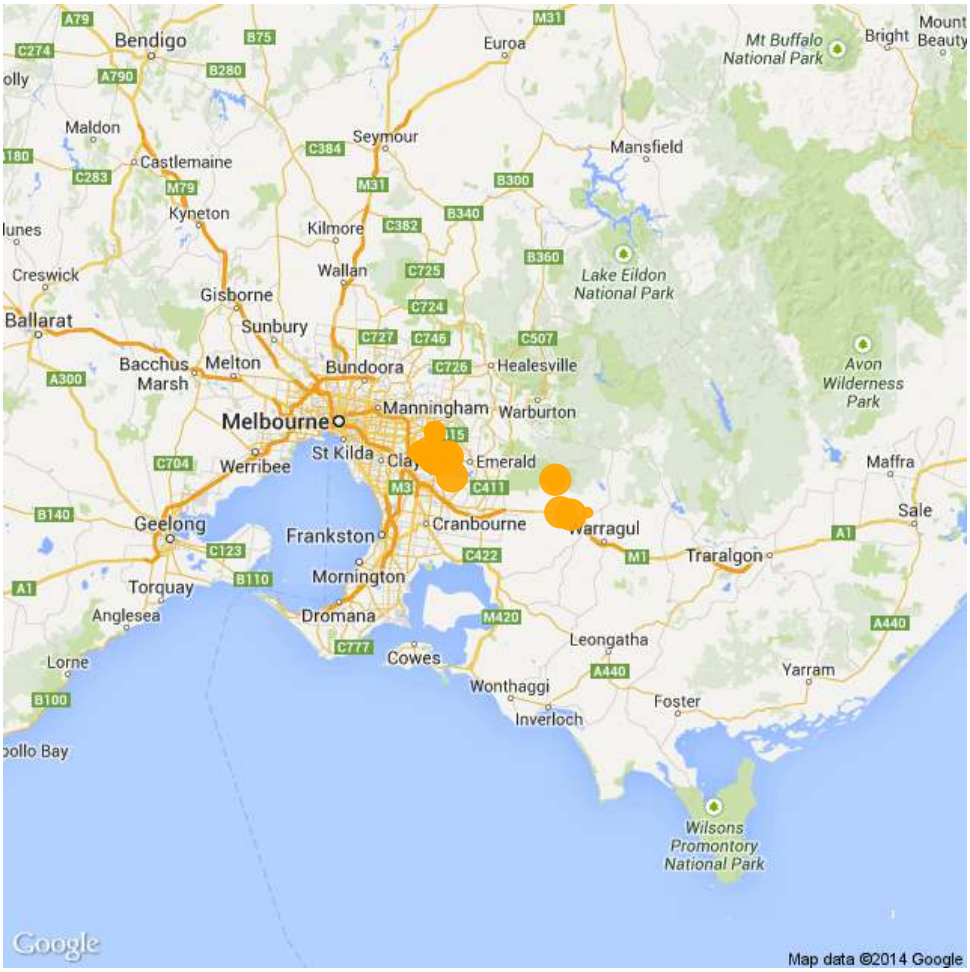


Figure 1: Occurrences of Tasmaninan Tiger scats

4.1.3 Creating your own analysis for inclusion in the PDF report

PopGenReport is meant to serve as a common framework for analysing genetic data. To facilitate this and to allow researchers to customise it for their own needs, we have included a switch in the `popgenreport` function, `mk.custom`. If this switch is set to "`TRUE`" a template snw file is run (custom.snw). This file is meant provide a template to demonstrate how a popgenreport function works. The popgenreport function basically combines a series of knitr files, each one performing a specified analysis and creates a series of outputs (pdf, figures and tables). It is straight forward to create your own type of analysis by editing the custom.snw file and including it in the report. The location of the custom.snw file can be found via: paste(.libPaths()[1],"/PopGenReport/swchunks/custom.snw",sep="""). Below you can see the whole file.

```
#custom.snw file
snw <- readLines(paste(.libPaths()[1],"/PopGenReport/swchunks/custom.snw", sep="" ))
```

snw

```
\section{Customised part using your own function}

\subsection{if you need a subsection }
You can use the "\texttt{mk.custom=TRUE}" switch to run your own customised part of the report.
Edit the file custom.snw in the library folder of the \texttt{PopGenReport} package
(\Sexpr{paste(.libPaths()[1],"/PopGenReport/swchunks/custom.snw",sep="")}).
Write your explanation using latex syntax.

<<echo=TRUE, results='markup', message=FALSE>>=
#to hide this output set echo=FALSE, results='hide' in the line above
#write in your code to calculate something here. Your genind object is accessed via the name 'cats'.

#create your own analysis
results <- summary(cats)$pop.eff

#create your table and use this for the report
#to print it as a table, it has to be a data.frame!!!
customtab <- data.frame(results)
@

<<echo=FALSE,results='asis'>>=
#here you print your results table, with a caption.
table_cap<-"Customised table: Number of samples per population"
custom.report.table<-xtable(customtab, cap=table_cap, digits=3)
print(custom.report.table,include.rownames=TRUE)

#and save your table as csv to the folder
write.csv(customtab, paste(cats@other$filename,"-cutomtab.csv", sep=""), row.names=TRUE)
@
\FloatBarrier

The plot below shows the number of samples per subpopuation as a pie chart.
Please note, the comment of the R help pages in regards to pie charts. Pie charts are a very bad
way of displaying information. The eye is good at judging linear measures and bad at judging
relative areas. A bar chart or dot chart is a preferable way of displaying this type of data.

<<custom_figure, echo=FALSE, fig.width=7, fig.height=4, fig.path=fig.path, dev=dev, dpi=dpi>>=
#this is how you create a plot of your results
pie(customtab[,1], labels=rownames(customtab))
@
\FloatBarrier

<<echo=FALSE, results='hide'>>=
#return your result to the popgenreport function
allresults$custom<-list(customtab)
@
\FloatBarrier

```

Everything up to line 8 in the file is a part of the LaTeX environment. We define the headings and provide some text for a description. To learn how to code LaTeX and how to write documents please refer to any of the the hundreds of manual you can find on the web (e.g `http://latex-project.org/guides/`). Line 9 - 19 are the first so-called "knitr chunks", which is basically normal R code. The genind object which contains all the data inside the popgenreport function is called "cats". When using data from your genind object in your custom function you will need to refer to the genind object 'cats' rather than the name of the genind object containing your data. In this simple example we create a summary and store one part of the output ($pop.eff) in the results. If you want to print your output as a table, you will need to convert your results into a data.frame and then print it via the `xtable` function (lines 23 - 28) in a new knitr-chunk. Lines 37 - 40 create a figure (a simple pie chart). The default figure settings is for all figures to be created in the PDF, SVG, and PNG file formats and be saved as files in the results folder. Line 43 - 46 is necessary if you want to make sure your result is returned by the `popgenreport` function so that you can use your results for further analysis. It is stored in the custom slot. To "play" with this file, it is best to create a backup of your files and then change parts of the code one by one and then run popgenreport using the `mk.custom=TRUE` switch.

5 Further reading and tutorials

There are a number of great resources how to use R and genetic data. We list them here, if you are aware of further tutorials please let us know.

- Tutorials in `PopGenReport`: `browseVignettes("PopGenReport")`
- Tutorials in `adegenet`: `browseVignettes("adegenet")`
- Tutorials in `mmod`: `browseVignettes("mmod")`
- Tutorials in `gap`: `browseVignettes("gap")`
- Overview on other R genetics packages: `http://cran.r-project.org/web/views/Genetics.html`
- There are plenty of general introductions to R. One we like is in the simpleR package from John Verzani (discontinued) `http://www.math.csi.cuny.edu/Statistics/R/simpleR`

6 Contacts

Please do not hesitate to contact the authors of `PopGenReport` if you have comments regarding the package. First of all we are certain there will still be "bugs" in the code as it is impossible to test it under all conditions. Additionally, we would like to expand the capabilities of PopGenReport in the future and comments on important functions that are missing are most welcome. For further updates please check our website: `www.popgenreport.org`.

Have fun running `PopGenReport` and may there be exciting results :-)

Bernd & Aaron

bernd.gruber@canberra.edu.au
aaron.adamack@canberra.edu.au

7 Attachement: Full pdf report on the bilby data set

A Population Genetic Report

using PopGenReport Ver. 2.0

Adamack & Gruber (2014)

July 10, 2014

Contents

1 Counts **2**

2 Sampling locations **4**

3 Population-wide test for heterozygosity **5**

4 Distribution of alleles by subpopulation and loci **5**
4.1 Allele frequency plots for all subpopulations and loci 5
4.2 Heatmaps of allele frequencies for all subpopulation and loci 6
4.3 Private Alleles 9
4.3.1 All private alleles 9
4.3.2 Number of private alleles by population 9

5 Calculation of F statistics for each locus across all sampling locations **9**

6 Computation of Nei's pairwise Fst between all pairs of populations **10**

7 Testing for null alleles **10**
7.1 Comparison of the observed number of homozygotes vs. expected 10
7.2 Frequency of null alleles 12
7.2.1 Determined using Chakraborty et al. (1992) 12
7.2.2 Determined using Brookfield (1996) 13

8 Allelic richness **13**

9 Hs and Ht based population differentiation statistics **13**

10 Testing for HWE for each combination of location and locus **15**
10.1 Testing HWE for each combination of location and locus 15
10.2 Combinations of location and locus that depart from HWE 15

11 Kosman and Leonard **15**

12 Genetic distance between individuals using Smouse and Peakall **16**

13 Spatial autocorrelation following Smouse and Pekall 1999 **17**

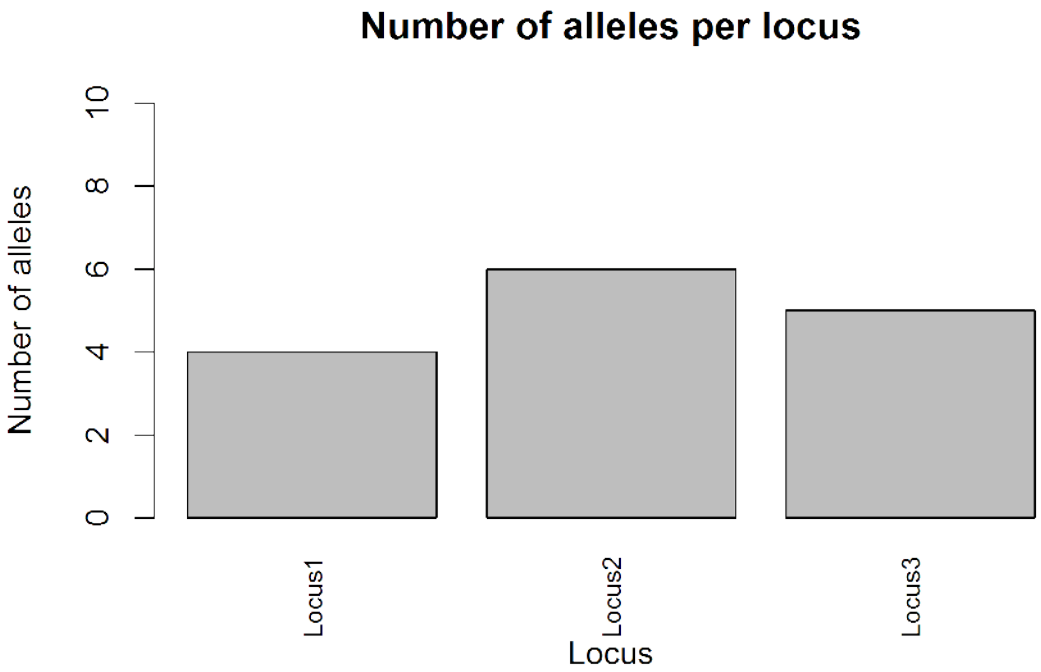
14 Principal Coordinate Analysis **17**

1 Counts

This analysis looks at 71 individuals.

The mean number of alleles per locus (across all locations): 5

The percentage of missing data was 0%.



The individuals were sampled from the following locations in the following numbers:

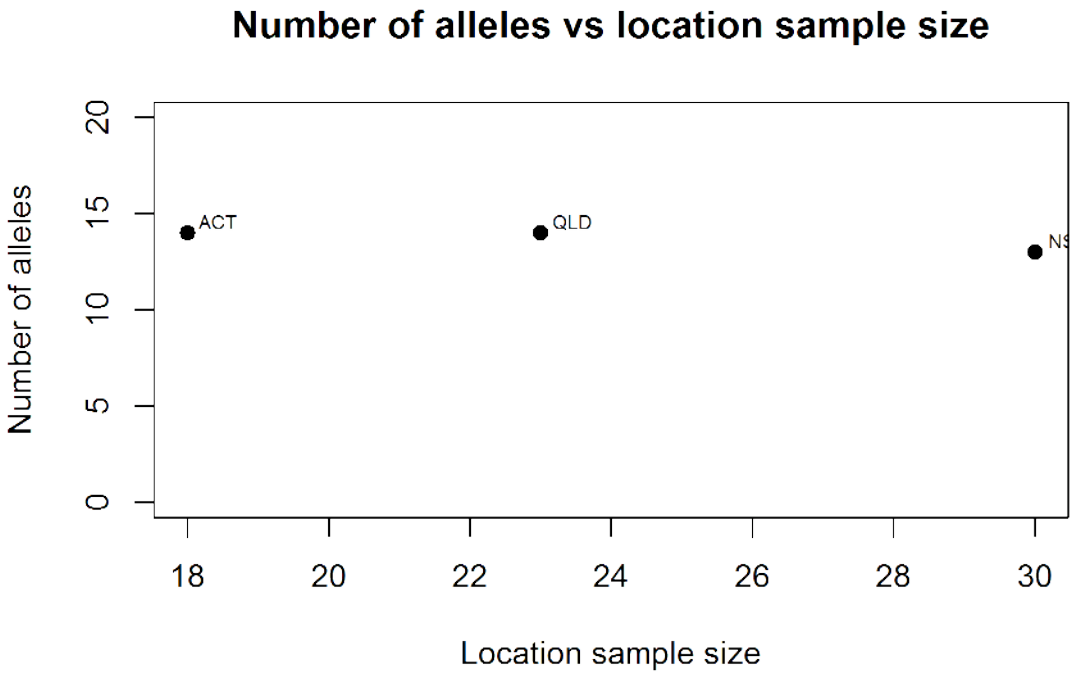
population	ACT	NSW	QLD
# ind	18	30	23

Table 1: Number of individuals per population

The total number of alleles sampled across all locations was 15
The total number of alleles seen in each sampling location was:

population	ACT	NSW	QLD
# alleles	14	13	14

Table 2: Number of alleles per population



The number of alleles per locus (across all subpopulations):

locus	Locus1	Locus2	Locus3
# alleles	4	6	5

Table 3: Number of alleles per locus across all subpopulations

2 Sampling locations

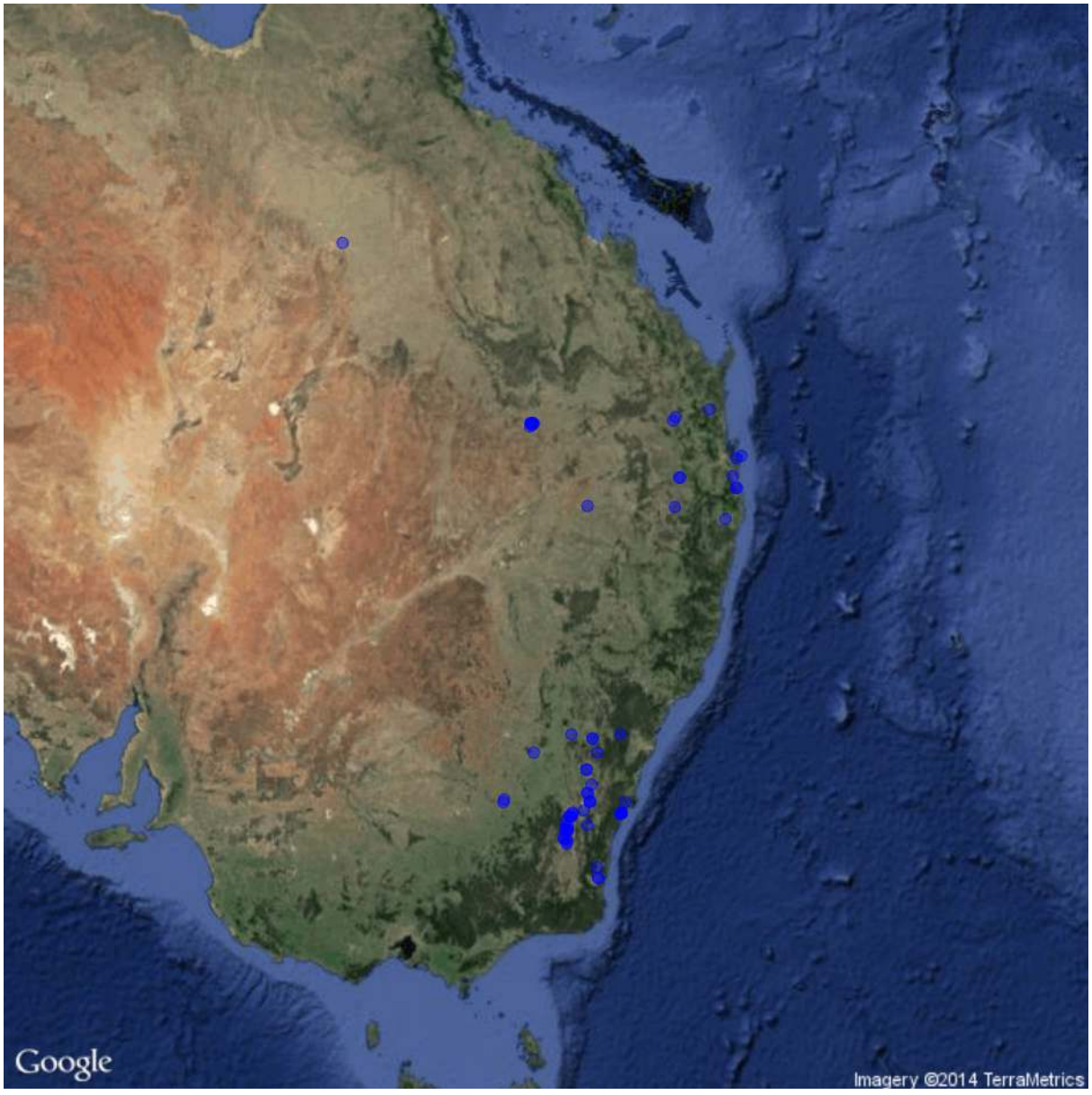


Figure 1: Sample locations

3 Population-wide test for heterozygosity

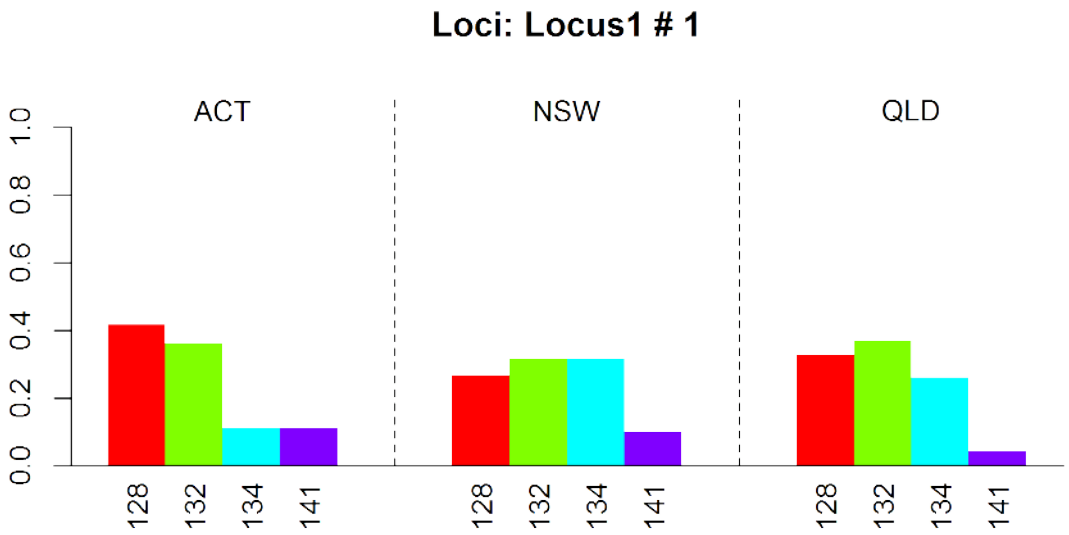
Locus	Expected	Observed	% difference
Locus1	0.708	0.718	-1.443
Locus2	0.691	0.746	-8.055
Locus3	0.710	0.718	-1.202

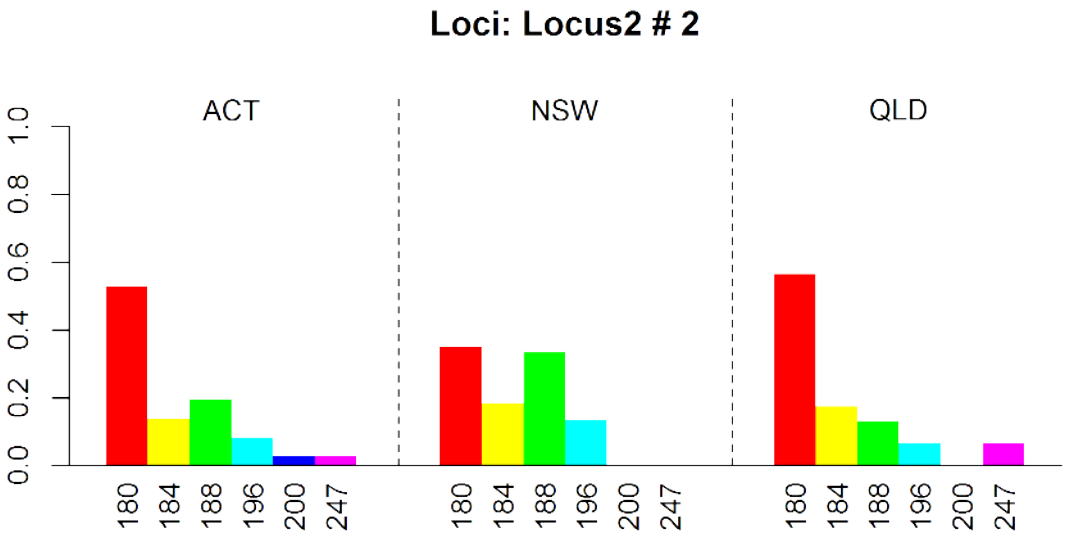
Table 4: The population-wide expected and observed heterozygosity and percent difference ((E-O)/E*100) at each locus

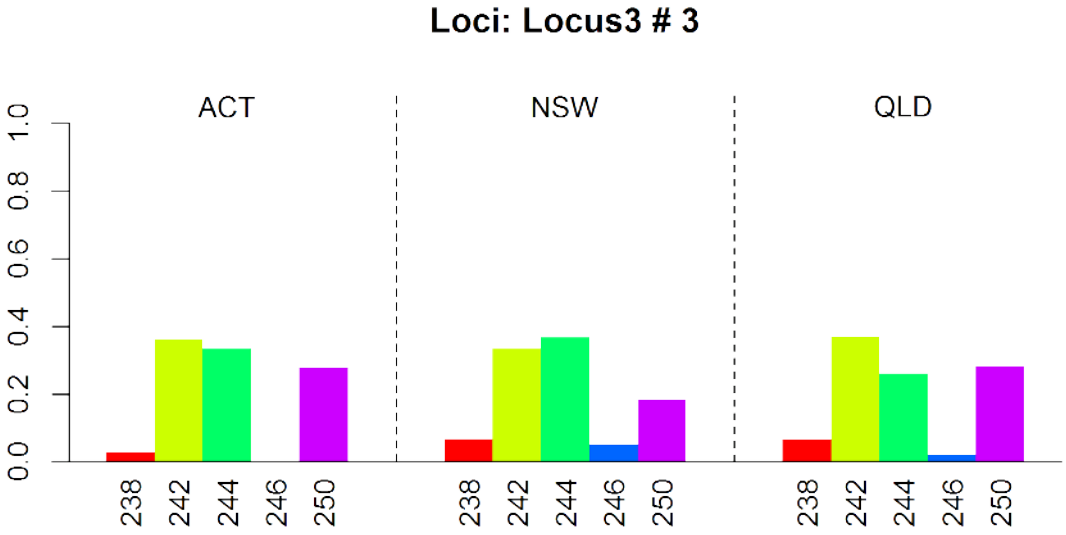
Bartlett test of homogeneity of variance. This test compares observed vs. expected heterozygosity. A significant result indicates that the population departs from HWE.
Bartlett's K-squared: 0.299, df = 1, p-value = 0.5846

4 Distribution of alleles by subpopulation and loci

4.1 Allele frequency plots for all subpopulations and loci







4.2 Heatmaps of allele frequencies for all subpopulation and loci

Cell colors indicate the proportion of the the total number of alleles in a subpopulation (e.g. 2N) that are of a particular allele type. The numbers within a cell are the counts of the number of alleles in a particular population.

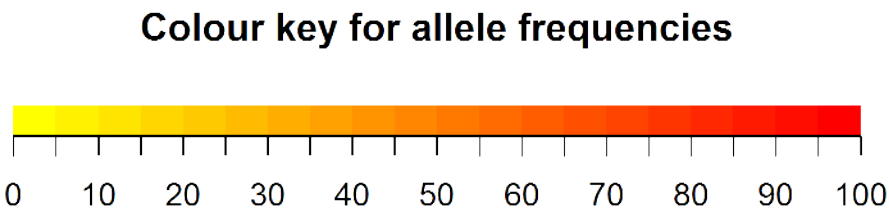
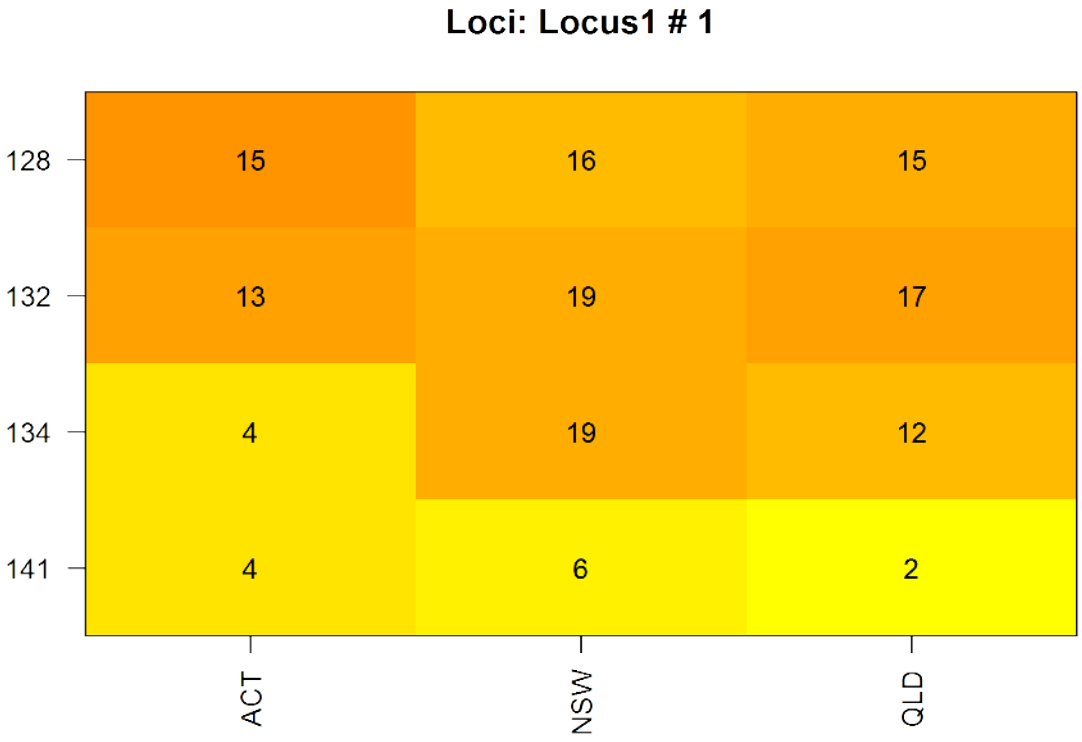
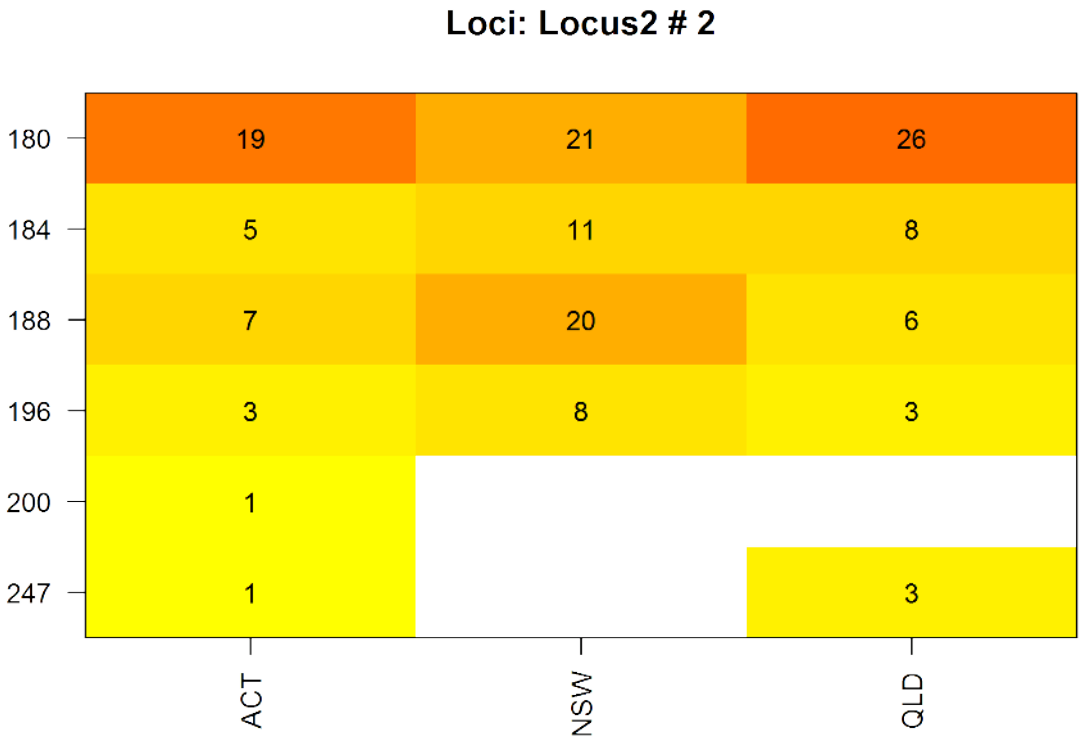
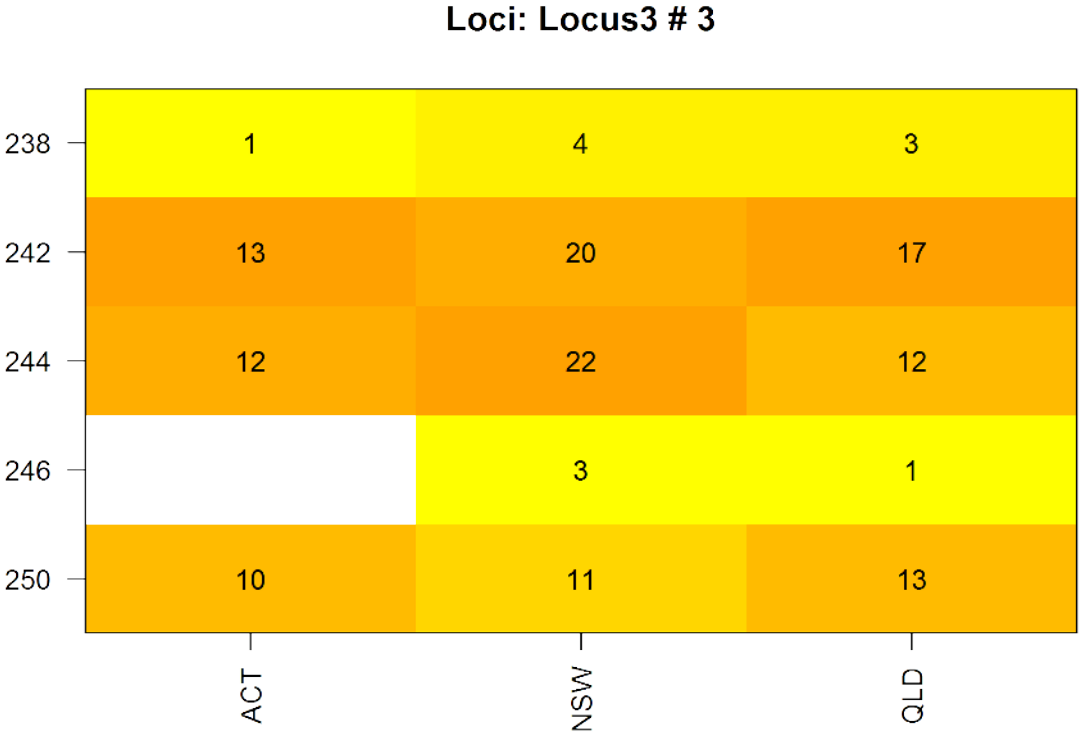


Figure 2: Color indicates the proportion of the total number of alleles in a subpopulation




Loci: Locus2 # 2
180
184
188
196
200
247
19
21
26
5
11
8
7
20
6
3
8
3
1
1
3
ACT
NSW
QLD


Loci: Locus3 # 3
238
242
244
246
250
1
4
3
13
20
17
12
22
12
3
1
10
11
13
ACT
NSW
QLD

4.3 Private Alleles

4.3.1 All private alleles

	Population	Allele
Locus2	ACT	200

Table 5: List of private alleles by locus and population

4.3.2 Number of private alleles by population

	ACT
Number of private alleles	1

Table 6: Number of private alleles by population

5 Calculation of F statistics for each locus across all sampling locations

	Fit	Fst	Fis
Locus3	-0.0084	-0.0103	0.0019
Locus1	-0.0060	0.0040	-0.0100
Locus2	-0.0645	0.0247	-0.0915

Table 7: Population wide Fit, Fst, and Fst values for each locus. The table is sorted in ascending order based on Fst.

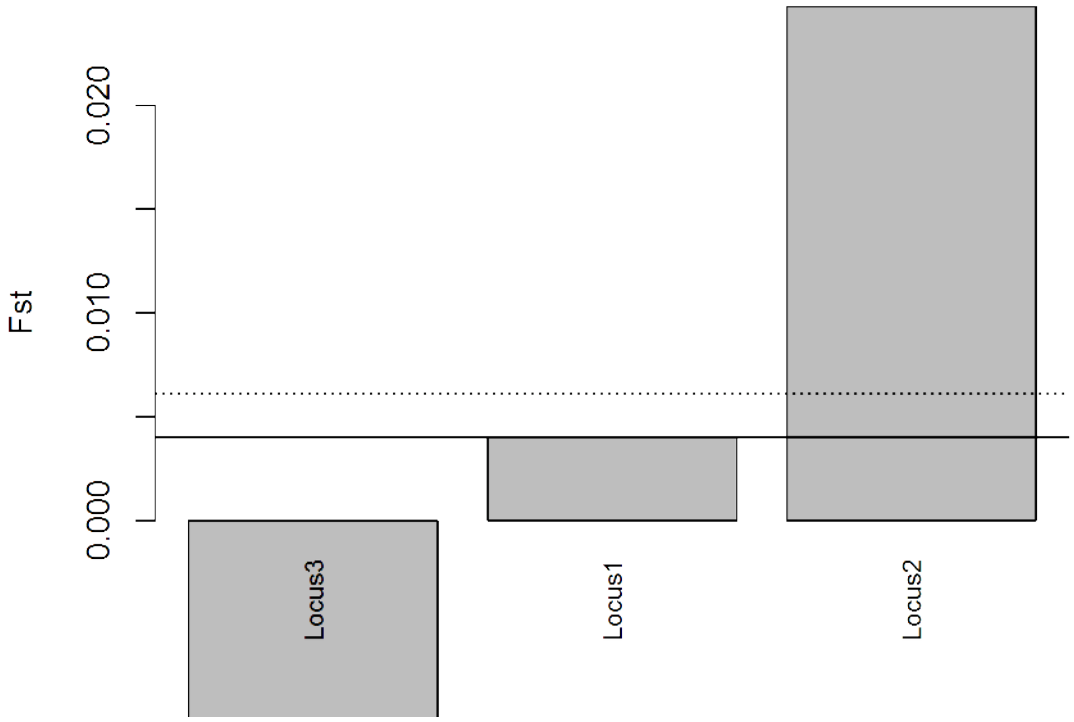


Figure 3: Fst across entire population at each locus. Solid line shows median Fst, dotted line shows mean Fst, dashed lines indicate 2.5th and 97.5th percentiles of Fst

6 Computation of Nei's pairwise Fst between all pairs of populations

	ACT	NSW	QLD
ACT	0.000		
NSW	0.015	0.000	
QLD	0.006	0.015	0.000

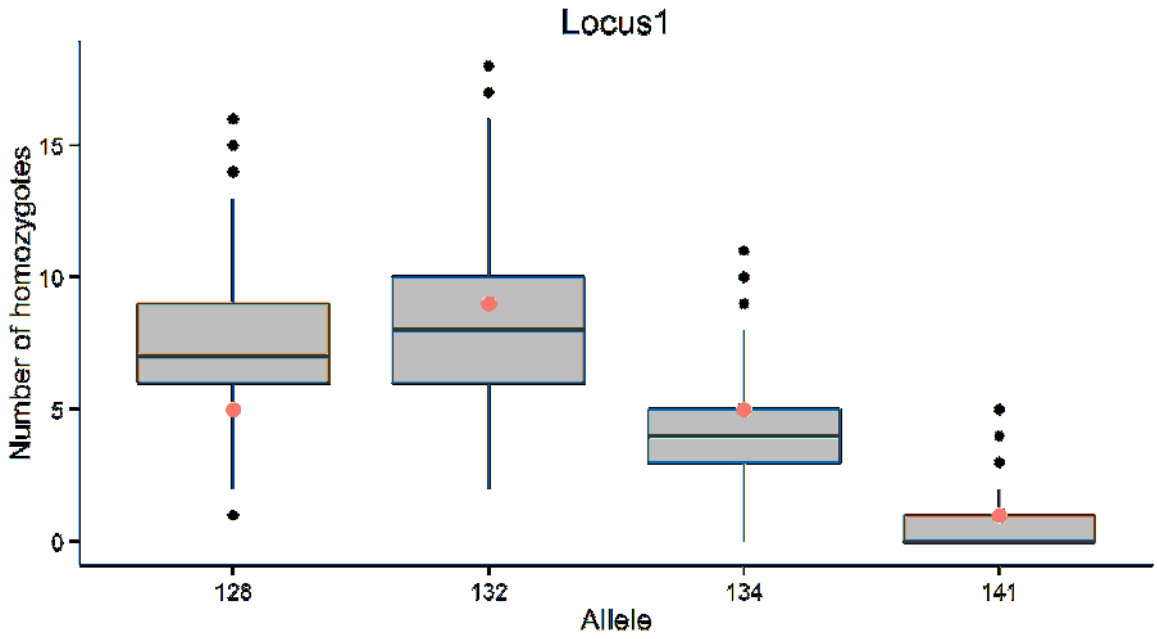
Table 8: Nei's pairwise Fst between all pairs of populations

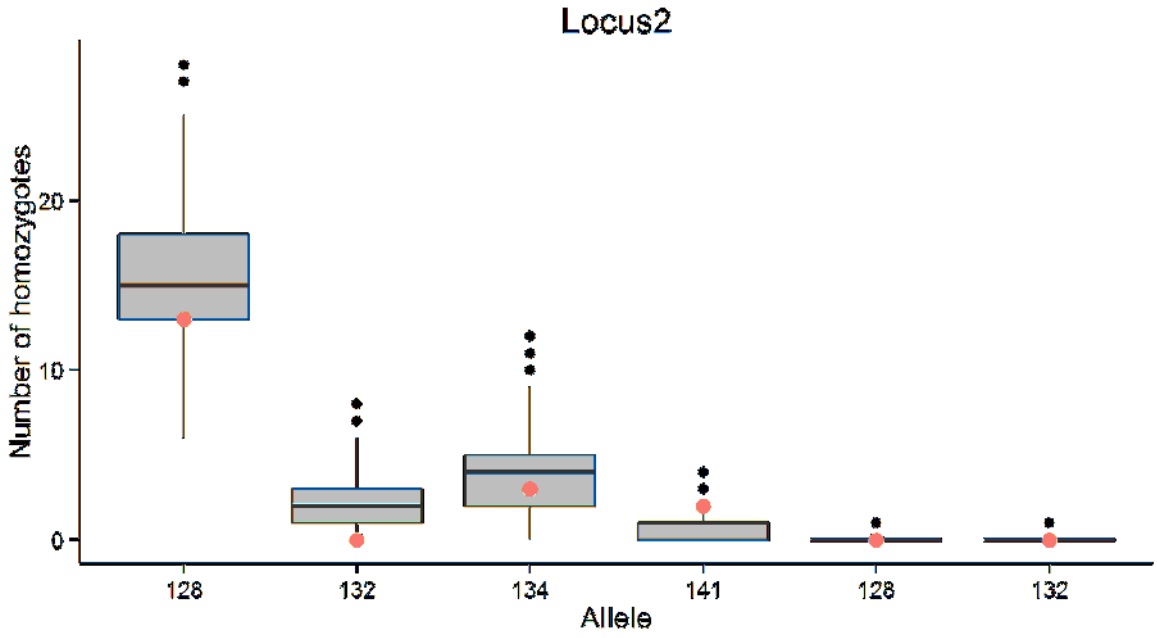
7 Testing for null alleles

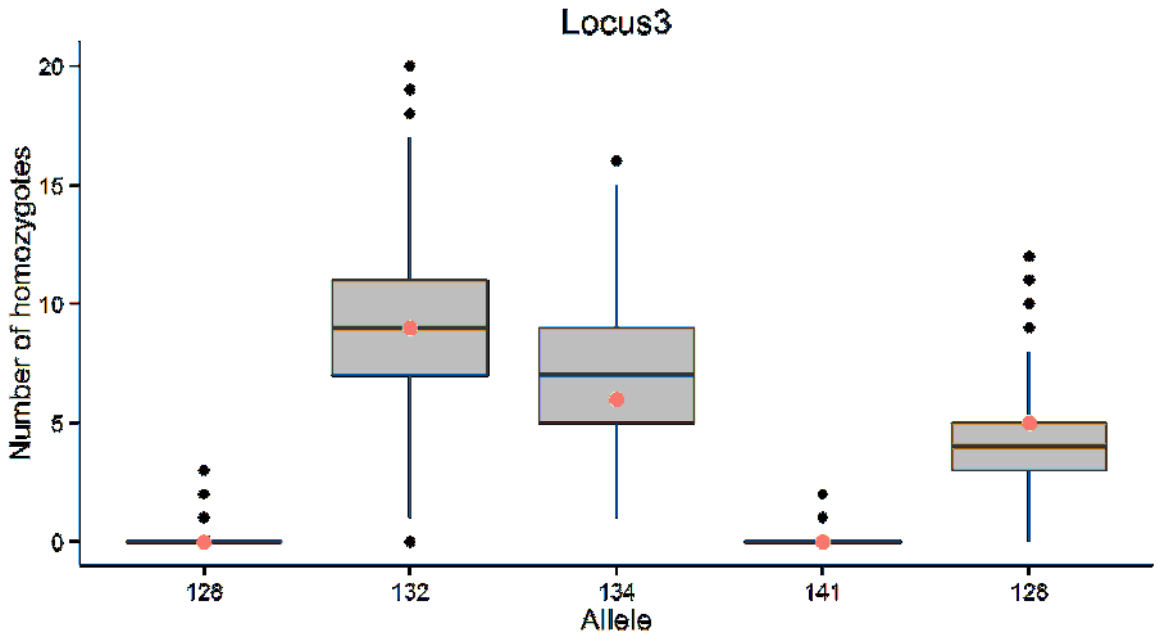
7.1 Comparison of the observed number of homozygotes vs. expected

Boxplots show the bootstrap distribution of the expected number of homozygotes for each allele with the boxes showing the 25th (lower edge), 50th (solid line), and 75th (upper edge) percentiles of the distribution and the whiskers showing 1.5 * the inter-quartile range. Solid black dots indicate outliers while red dots indicate the observed number of homozygotes for the allele. If the red dot is above the end of the whisker it suggests that there is an excess of homozygotes for that allele.

The probability of the observed number of homozygotes for each allele is available in the results object using null.all$homozygotes$probability.obs


Locus1
Number of homozygotes
15
10
5
0
128
132
134
141
Allele


Locus2
Number of homozygotes
20
10
0
128
132
134
141
128
132
Allele

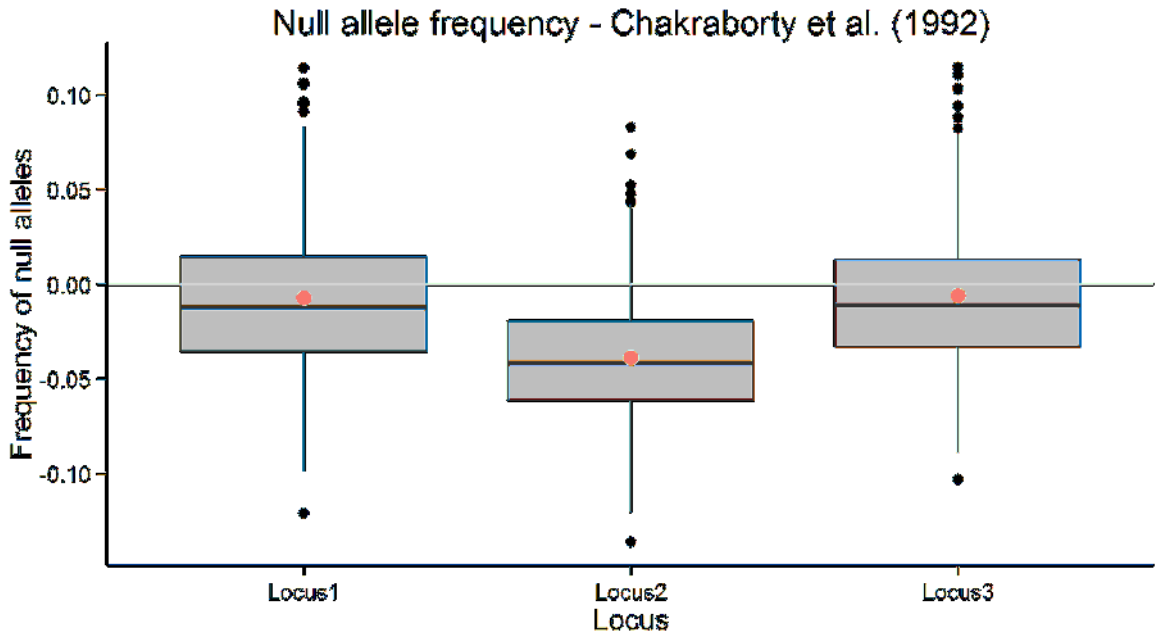


	Allele-1	Allele-2	Allele-3	Allele-4	Allele-5	Allele-6
Locus1	0.786	0.316	0.246	0.084		
Locus2	0.690	0.865	0.555	0.026	0.002	0.051
Locus3	0.213	0.389	0.622	0.053	0.205	

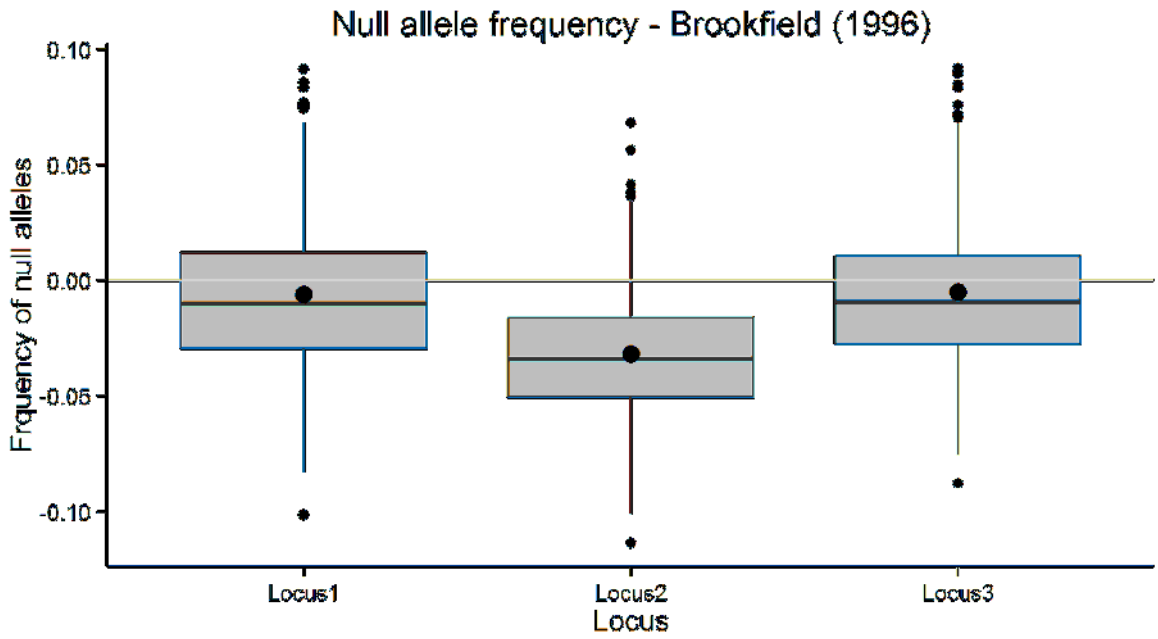
Table 9: Probability of the observed number of homozygotes

7.2 Frequency of null alleles

7.2.1 Determined using Chakraborty et al. (1992)



7.2.2 Determined using Brookfield (1996)



	Locus1	Locus2	Locus3
Observed frequency	-0.006	-0.032	-0.005
Median frequency	-0.010	-0.034	-0.009
2.5th percentile	-0.065	-0.083	-0.060
97.5th percentile	0.057	0.020	0.056

Table 10: Summary of null allele frequencies by locus for Brookfield (1996)

8 Allelic richness

Allelic richness for each locus and subpopulation was based on a subsample of 36 alleles.

	ACT	NSW	QLD
Locus1	4.000	3.997	3.957
Locus2	6.000	4.000	4.984
Locus3	4.000	4.919	4.775

Table 11: Allelic richness by locus and population

	ACT	NSW	QLD
Mean richness	4.667	4.305	4.572
Total richness	14.000	12.916	13.715

Table 12: Allelic richness summary statistics

9 Hs and Ht based population differentiation statistics

	Hs	Ht	Gst	Gprime_st	D
Locus1	0.708	0.710	0.003	0.017	0.012
Locus2	0.680	0.688	0.012	0.055	0.038
Locus3	0.718	0.712	-0.008	-0.041	-0.029

Table 13: Hs and Ht based estimates of differentiation: Gst, Gst and Dest for each locus

	Ht	Gst_est	Gprime_st	D_het	D_mean
0.702	0.704	0.002	0.012	0.008	

Table 14: Hs and Ht based global estimates of differentiation: Gst, Gst and Dest for each locus

	ACT	NSW	QLD
ACT			
NSW	0.025		
QLD	-0.027	0.029	

Table 15: mmod Jost's D pairwise

	ACT	NSW	QLD
ACT			
NSW	0.035		
QLD	-0.040	0.040	

Table 16: Pairwise Gst - Hedrick

	ACT	NSW	QLD
ACT			
NSW	0.005		
QLD	-0.006	0.006	

Table 17: Pairwise Gst - Nei

10 Testing for HWE for each combination of location and locus

10.1 Testing HWE for each combination of location and locus

The table below shows the p-value for the test of HWE for each combination of location and locus. The p-values shown here differ from those produced by GENALEX as the Chi-square test performed here includes the Yates continuity correction which GENALEX does not. As a large number of Chi-square tests are performed, $\alpha = 0.05$ cannot be used as Type I errors are likely to occur. Instead a Bonferroni adjustment is used $\alpha = (0.05/\ 9) = 0.0056$.

	ACT	NSW	QLD
Locus1	0.517	0.285	0.286
Locus2	0.167	0.029	0.800
Locus3	0.818	0.624	0.566

Table 18: Chi-square test of HWE p-values for each combination of location and locus

10.2 Combinations of location and locus that depart from HWE

There were no departures from HWE for any combination of sub-population and locus

11 Kosman and Leonard

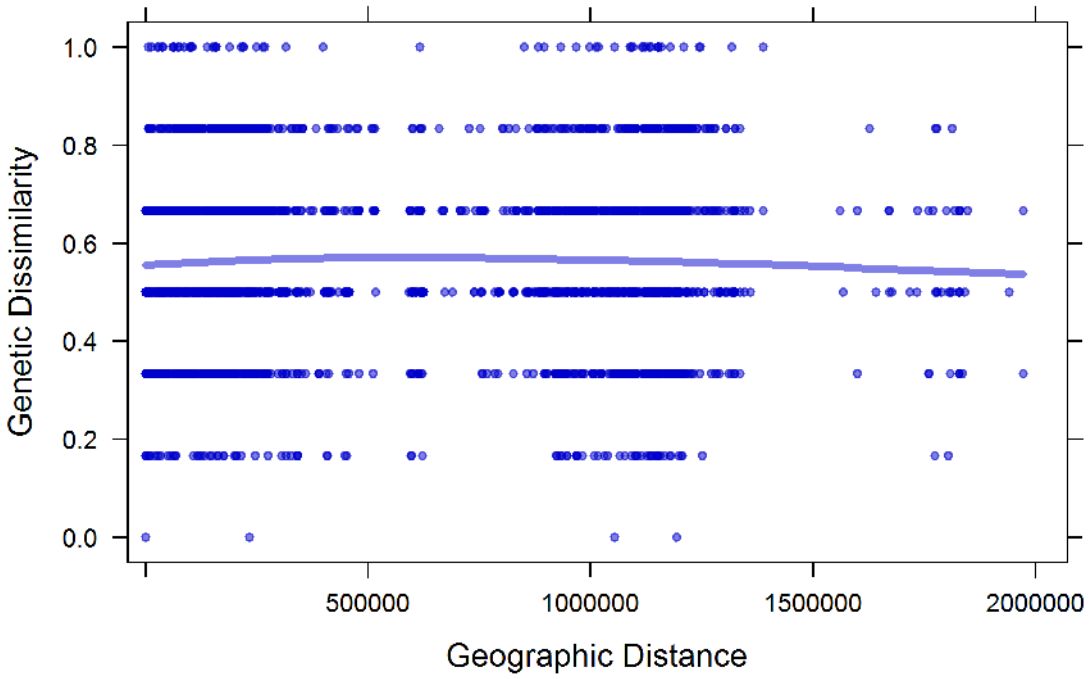


Figure 4: Genetic Dissimilarity (Kosman and Leonard 2005) vs Geographic Distance. The line represents the running average.

12 Genetic distance between individuals using Smouse and Peakall

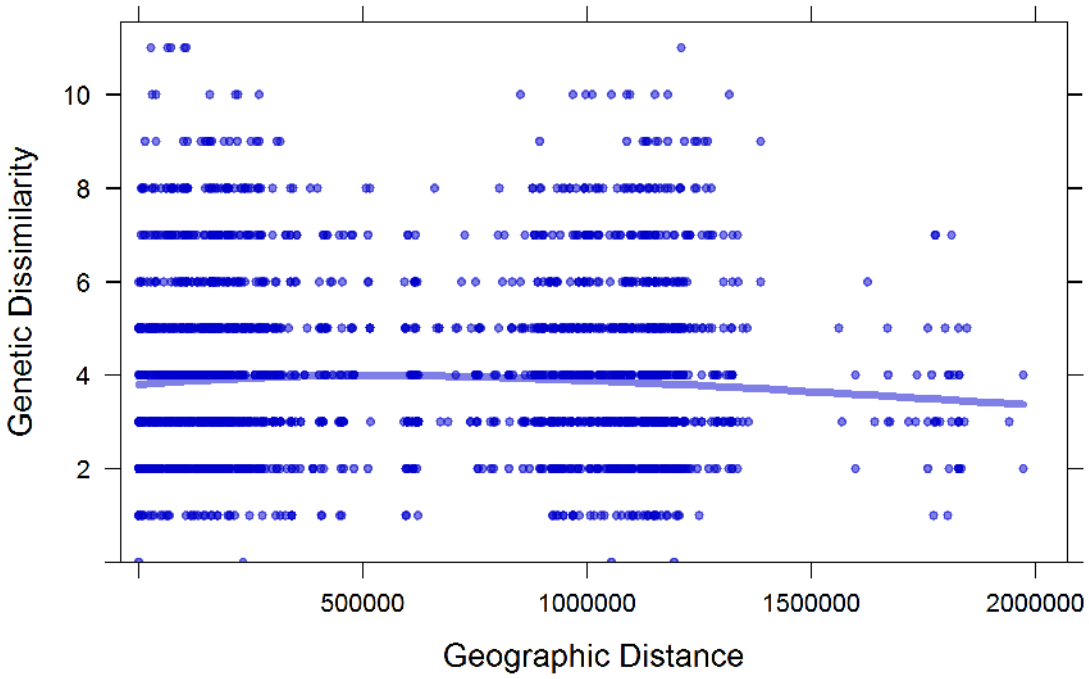


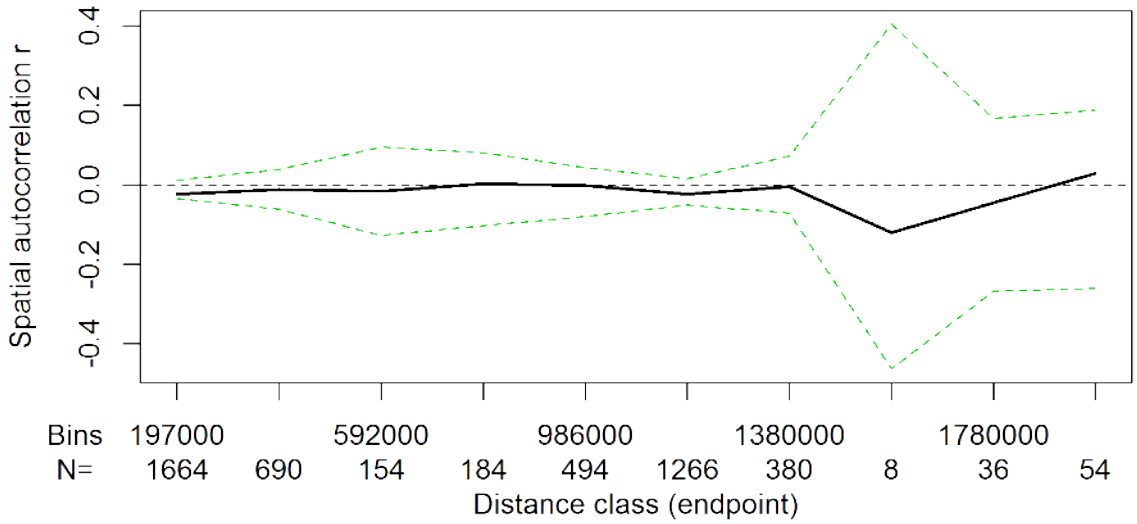
Figure 5: Genetic Dissimilarity (Smouse and Peakall 1999) vs Geographic Distance. The line represents the running average.

13 Spatial autocorrelation following Smouse and Pekall 1999

Global spatial autocorrelation is a multivariate approach combining all loci into a single analysis. The autocorrelation coefficient r is calculated for each pairwise genetic distance pairs for all specified distance classes. For more information see Smouse and Peakall 1999, Peakall et a. 2003 and Smouse et al. 2008. For full references refer to the help files by typing "?spautocor".

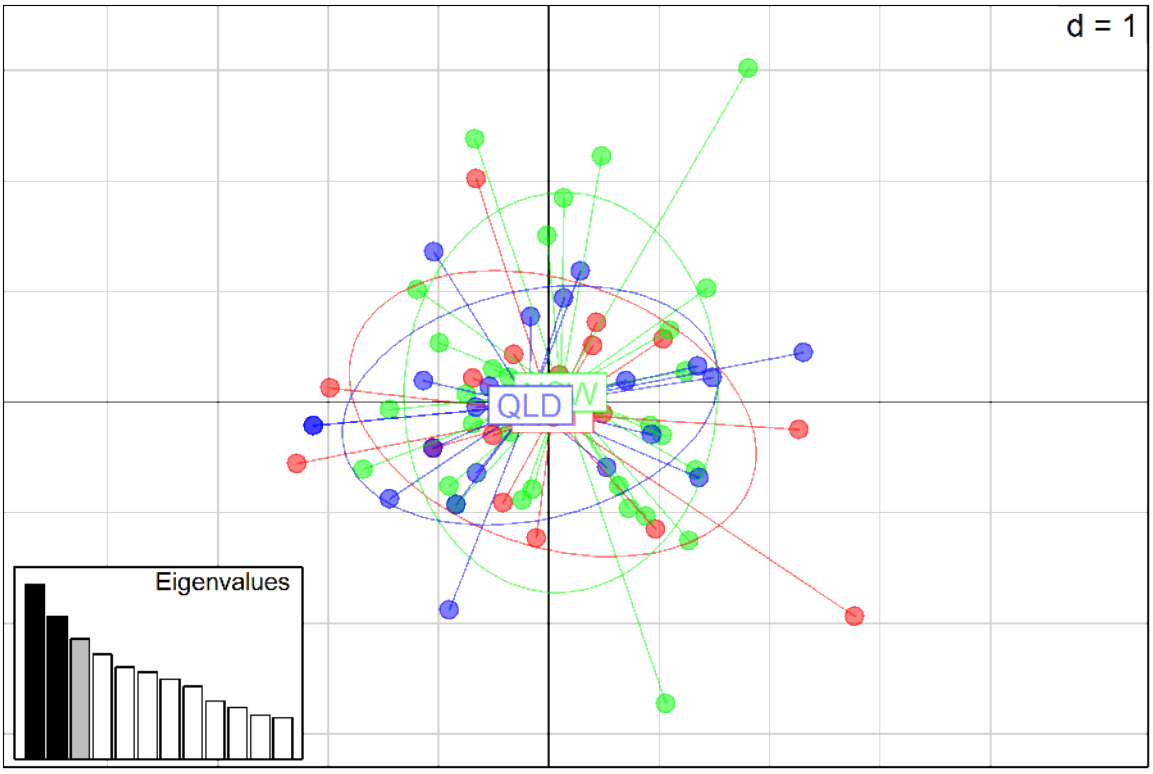
	bin	N	r	r.l	r.u
1	197300.000	1664	-0.023	-0.036	0.012
2	394600.000	690	-0.012	-0.061	0.038
3	591900.000	154	-0.017	-0.128	0.094
4	789200.000	184	0.002	-0.103	0.081
5	986500.000	494	-0.002	-0.080	0.044
6	1183800.000	1266	-0.023	-0.051	0.015
7	1381100.000	380	-0.004	-0.072	0.072
8	1578400.000	8	-0.120	-0.464	0.403
9	1775700.000	36	-0.045	-0.268	0.167
10	1973000.000	54	0.028	-0.260	0.187

Table 19: Spatial autocorrelation and bootstrap results using the approach of Smouse and Pekall 1999.



14 Principal Coordinate Analysis

Principal coordinate analysis. This plot, using the first two axes, visualises genetic diversity among sampled individuals. Missing data are replaced by the mean of the allele frequencies. Colors indicate subpopulation if specified in the genetic data set.


d = 1
QLD
Eigenvalues