

SNPolisher™ Package

USER GUIDE

Publication Number MAN0017790

Revision D.0

For Research Use Only. Not for use in diagnostic procedures.

ThermoFisher
S C I E N T I F I C



Affymetrix Inc. | 3450 Central Expressway | Santa Clara, CA 95051 | USA

For descriptions of symbols on product labels or product documents, go to [thermofisher.com/symbols-definition](https://www.thermofisher.com/symbols-definition).

The information in this guide is subject to change without notice.

DISCLAIMER: TO THE EXTENT ALLOWED BY LAW, THERMO FISHER SCIENTIFIC INC. AND/OR ITS AFFILIATE(S) WILL NOT BE LIABLE FOR SPECIAL, INCIDENTAL, INDIRECT, PUNITIVE, MULTIPLE, OR CONSEQUENTIAL DAMAGES IN CONNECTION WITH OR ARISING FROM THIS DOCUMENT, INCLUDING YOUR USE OF IT.

Revision history: Pub. No. MAN0017790

Revision	Date	Description
D.0	30 September 2020	Document updated to removed the functions: OTV_Caller, Ps_Call_Adjust, Ballele_Freq_Test, and add the functions: Ps_Vis_Multi_ID, CN_Visualization.
C.0	22 January 2019	Updated descriptions of multiallelic and copy-number genotype handling for all functions.
B.0	26 October 2018	Updated Ps_Metrics and Ps_Visualization to add a new input parameter, and Ps_Metrics to add a new output column. Updated the descriptions of Ps_Metrics, Ps_Classification, and Ps_Visualization with information on the new parameters and columns.

Important Licensing Information: These products may be covered by one or more Limited Use Label Licenses. By use of these products, you accept the terms and conditions of all applicable Limited Use Label Licenses.

TRADEMARKS: All trademarks are the property of Thermo Fisher Scientific and its subsidiaries unless otherwise specified.

©2020 Thermo Fisher Scientific Inc. All rights reserved.

Contents

1	Acknowledgements	5
2	Overview	6
2.1	What is R?	6
2.2	Packages and Libraries	8
3	Before Running SNPolisher	9
3.1	Downloading SNPolisher	11
3.2	Installing SNPolisher in R	11
3.3	Installing SNPolisher in RStudio	13
3.4	SNPolisher Support	14
4	SNPolisher Functions	15
4.0	Tips For Running SNPolisher	15
4.0.1	Copying and Pasting in R	15
4.0.2	Inputs, Arguments, and Outputs	15
4.0.3	Working Directories	16
4.0.4	Help Files	17
4.0.5	APT Files	18
4.0.6	Differences Between the SNPolisher and AxAS Versions of Cluster Plots	19
4.0.7	Out of Memory Errors in R	19
4.1	APT File Formats	21
4.1.1	Dynamic Call Code Assignments	21
4.1.2	Multiallelic SNPs	22
4.1.3	Special SNPs	24
4.2	Ps_Visualization	26
4.2.1	Inputs and Arguments	26
4.2.1.1	Selecting Colors in R	35
4.2.1.2	Plotting Multiallelic SNPs	36
4.2.1.3	Gender-Separated Plots	41
4.2.1.4	Batch Plotting	42
4.2.2	Outputs	48
4.3	Ps_Vis_Density	49
4.3.1	Inputs and Arguments	49
4.3.1.1	Density Algorithm	53
4.3.2	Outputs	53
4.4	Ps_Vis_Multi_ID	54
4.4.1	Inputs and Arguments	54
4.4.2	Outputs	58
4.5	CN_Visualization	59
4.5.1	Inputs and Arguments	59
4.5.1.1	Batch Histograms	60
4.5.1.2	Plate Effects	61
4.5.1.3	Heatmap	62
4.5.1.4	Plate Histograms	63
4.5.1.5	Plate Histograms with Truth	64
4.5.2	Outputs	66
4.6	Autotetraploid Functions	67
4.6.1	fitTetra_Input	67
4.6.1.1	Inputs and Arguments	68
4.6.1.2	Outputs	68
4.6.2	fitTetra	68
4.6.2.1	Inputs and Arguments	69

4.6.2.2	Outputs	70
4.6.3	fitTetra_Output	70
4.6.3.1	Inputs and Arguments	71
4.6.3.2	Outputs	71
4.7	Ps_Extract	72
4.7.1	Inputs and Arguments	72
4.7.2	Outputs	73
5	Vignettes	74
5.1	Introductory Vignettes	74
5.2	Vignette: Human Data	75
5.3	Vignette: Wheat Data	85
5.4	Vignette: Case-Control Data	91
5.5	Vignette: Batch Data	92
5.6	Vignette: Multiallelic Data	99
5.7	Vignette: Copy Number Data	101
5.8	Vignette: Autotetraploid Data	103
6	References	104
	Appendices	106
	Appendix A Installation	106
A.1	Windows	106
A.1.1	R	106
A.1.2	R Studio	106
A.1.3	Perl	108
A.2	Mac OS X	109
A.2.1	R	109
A.2.2	R Studio	110
A.2.3	Perl	111
A.3	Linux	112
A.3.1	R	112
A.3.2	RStudio	113
A.3.3	Perl	113
	Appendix B R Basics for SNPlisher	114

1 Acknowledgements

The authors would like to thank Hong Gao, Yan Lu, Dorothy Oliver, Cat Barts, Ram Varma, Anu Mittal, Manasi Shah, Marcos Woehrmann, Alexa Barnes, Ali Pirani, Teresa Webster, and Robin Walters for their work on `SNPolisher` functions and helpful suggestions.

2 Overview

SNPolisher is an R package that provides SNP visualization tools and reformats APT genotype output for use with the **fitTetra** package and reformats **fitTetra** output for visualization in **SNPolisher**. **SNPolisher** addresses challenges posed when genotyping the human genome, and addresses new challenges posed by the complexity of plant and animal genomes. Genome-wide scans of nucleotide variation in human, plant and animal genomes provide an increasing number of associations with complex traits. However, the variants detected often have small effects, and small sources of systematic or random error can cause spurious results or obscure real effects. The need for careful attention to data quality from genotyping microarray data is critical to the success of a study, and requires visualizing SNPs and top genome-wide association study (GWAS) hits. This guide uses the term *SNP* to refer to bi-allelic or multi-allelic markers which can be both SNPs and indels.

Problems with genotype calls may occur when genotyping populations with a high proportion of low minor allele frequency (MAF) SNPs. Non-human genomes, especially plants, are likely to produce more than three genotypes at a some loci due to structural diversity within and between populations, polyploid genomes, a high percentage of genomic repetitive elements, and/or interfering mutations near a SNP locus. In addition, high levels of inbreeding within a population can lead to unexpected genotype clusters; SNPs that are challenging, including those for which mis-clustering events produce miscalls when calling genotypes; and SNPs at loci which produces more than three genotypes.

SNPolisher addresses these problems by providing functions for visualizing genotyping results generated by Applied Biosystems Array Power Tools (APT) software, provided for Affymetrix Axiom®myDesign custom and catalog arrays. **SNPolisher** includes a set of functions which enable the user to draw and review cluster plots for SNPs in a probeset list. SNPs with high quality cluster properties can then be taken forward into downstream analysis.

Currently, there are seven functions: (i) drawing genotype cluster plots for each SNP, (ii) drawing density genotype cluster plots for each SNP, (iii) drawing genotype cluster plots for each probeset that makes up a multiallelic SNP, (iv) drawing plots for genomic regions with copy-number variation, (v) reformatting Axiom genotyping output files for use with the autotetraploid genotype calling algorithm **fitTetra**, (vi) reformatting **fitTetra** output for use with **SNPolisher**, and (vii) extracting data for a list of SNPs from larger genotype files. **SNPolisher** is designed to take standard Axiom genotyping output files from APT as input files.

The seven functions are named *Ps_Visualization*, *Ps_Vis_Density*, *Ps_Vis_Multi_ID*, *CN_Visualization*, *fitTetra_Input*, *fitTetra_Output*, and *Ps_Extract*. *Ps_Visualization* is designed to be run on the results of the Axiom Best Practices Workflow, which is available in APT or Axiom™ Analysis Suite (AxAS).

Figure 1 shows an example of one type of cluster plot.

2.1 What is R?

R is an open-source programming language and environment designed for statistical analysis and graphics. R is licensed under the GNU General Public License and is free for all users. R is one of the commonly used statistical programming languages in the statistics community (as well as Stata, SAS, and SPSS), and is known for producing high quality graphics. The R Project (<http://www.r-project.org>) [14] provides package and binary files for installation on personal computers and servers.

R is an *interpreted language*. Users type input at the command-line prompt, and the input is interpreted for the compiler. R is command-line input only - there is no graphical user interface (GUI), a.k.a. point-and-click. There are optional graphical user overlays that make it easier to use R.

RStudio (or another graphical interface for R) is necessary for all of the functionality in **SNPolisher**. We recommend that users install RStudio to accompany R (see appendix A). RStudio displays multiple R windows in one interface, including history, workspace, plots, and packages. The package vignettes are easily available when using RStudio but are not available from the command-line prompt in R without a GUI. Additionally, RStudio allows users to copy and paste into the R command line when it is not possible to copy and paste in the R installation (see figure 2).

To enter commands in R, the user types a command at the prompt and hits the return key. R displays the result directly below the command. When the user creates a plot, it is generated in a separate window

called the plot window and no results are returned at the command line unless specified (see figure 3).

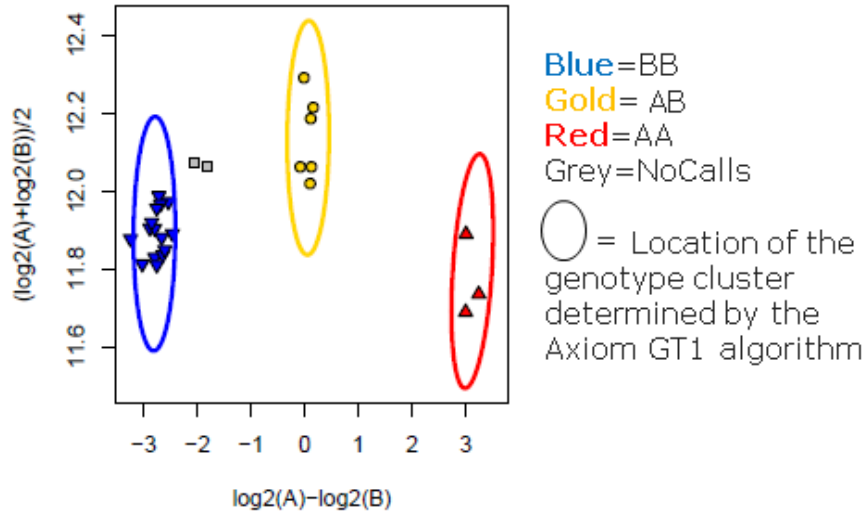
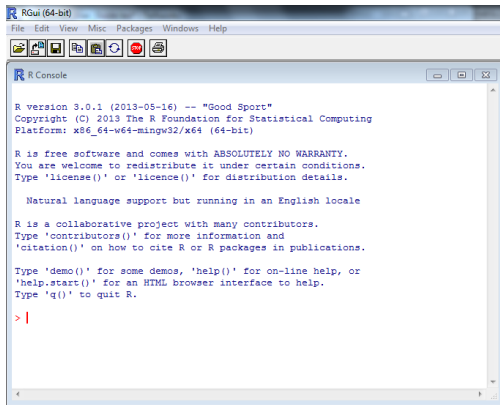
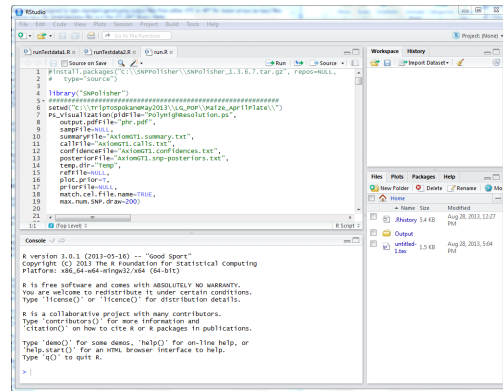


Figure 1: Contrast vs Size cluster plot: the cluster plot displays the genotypes of multiple samples for a single SNP marker. Each point represents an individual sample and each color represents a separate genotype (AA, AB, or BB). The X axis is $Contrast = \text{Log}_2\left\{\frac{A_{\text{signal}}}{B_{\text{signal}}}\right\}$. The Y axis is $Size = \left\{\frac{\text{Log}_2(A_{\text{signal}})+\text{Log}_2(B_{\text{signal}})}{2}\right\}$.



(a) The R user interface - command line prompt.



(b) The RStudio user interface - multiple windows are available to the user.

Figure 2: R versus RStudio user interfaces

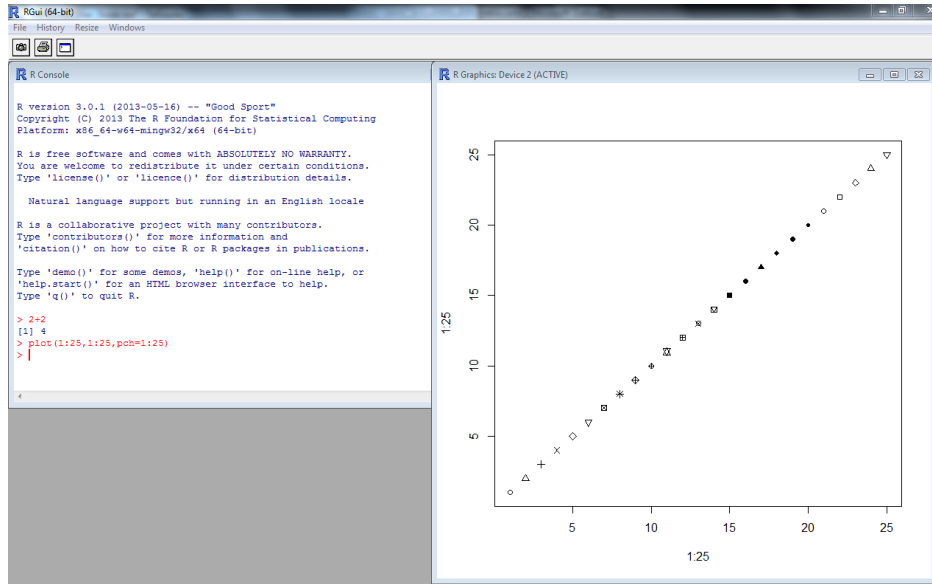


Figure 3: R command line result and plotting window.

2.2 Packages and Libraries

Because R is an open-source project, many users have developed and contributed their own sets of R functions. These sets of functions are called packages. Generally, people write a package when they need to use a function over and over again and they do not want to type in the code for making the function each time they open R. For example, the functions in *SNPolisher* are very useful but users would not want to have to reprogram *Ps_Visualization* whenever they need to plot a probeset. Instead, R users can bundle the code into a package for easier access. Many people upload their packages to the Comprehensive R Archive Network (<http://cran.r-project.org>) [15] for other users to download and install.

The term *package* refers to a collection of R functions and compiled code, and may include data sets as well. All packages uploaded to CRAN must be well formatted and include certain files, such as help files and a manual. When a package is downloaded and installed, the directory that it is stored in is called the library. R can easily load the information and data stored in a library after a package has been installed. The `library` command is used to tell R to load the functions and code in a library:

```
> library(SNPolisher)
```

See the R Project for more information on packages [17].

3 Before Running SNPolisher

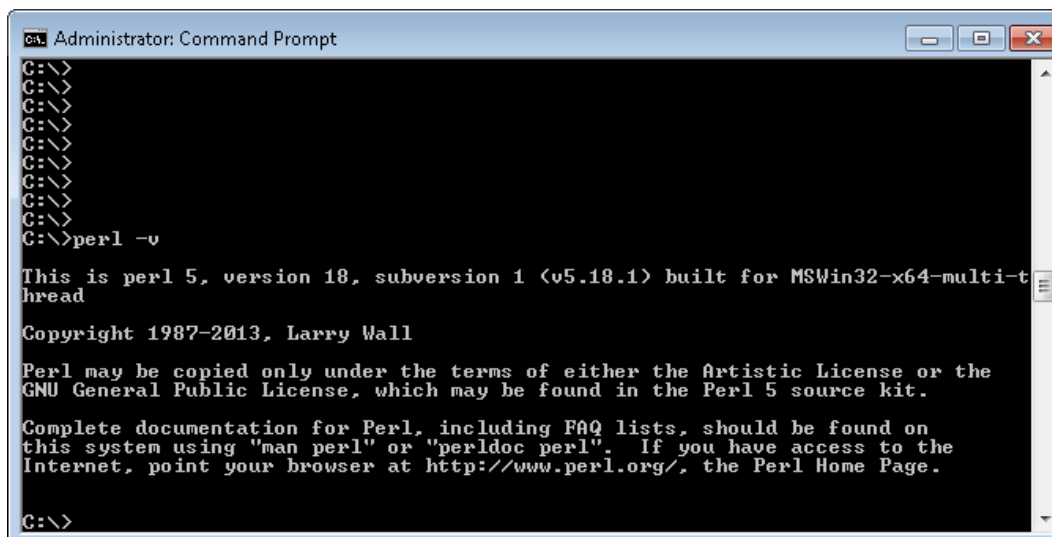
In the Windows environment, R/RStudio and perl must be installed; in Mac OS X and linux, perl is already installed and R/RStudio must be installed. See appendix A for instructions on installing R, RStudio, and perl. Once R/RStudio and perl are installed, the `SNPolisher` package should be installed. Although it is not necessary to install RStudio to run `SNPolisher`, we recommend using RStudio. Installation instructions are given for R and RStudio are in sections 3.2 and 3.3.

It is very important to install recent versions of R and perl or the `SNPolisher` functions will not run correctly. The installed version of R must be 3.1.0 or later, and the installed version of perl must be 5.20.0 or later. Note that `SNPolisher` will return errors that are caused by using an old version of perl but which do not state that the installed version of perl is too old. The user should check which version of perl is being used and update it if necessary.

Several of the `SNPolisher` functions have been designed to handle very large data sets. To run `fitTetra_Input` and `fitTetra_Output` with summary files larger than 100 MB, a 64-bit version of perl must be installed. The functions will not run correctly with a 32-bit version on large files. To see which version of perl is installed, open the terminal (OS X) or the command prompt (Windows), and type:

```
> perl -v
```

The result will show which version of perl is installed. Figure 4 shows the results on a Windows machine. This version is 5.18.1 and it was built for `MSWin32-x64-multi-thread`, where the `x64` indicates that it is a 64-bit version.



```
Administrator: Command Prompt
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>
C:\>perl -v
This is perl 5, version 18, subversion 1 (v5.18.1) built for MSWin32-x64-multi-thread
Copyright 1987-2013, Larry Wall

Perl may be copied only under the terms of either the Artistic License or the
GNU General Public License, which may be found in the Perl 5 source kit.

Complete documentation for Perl, including FAQ lists, should be found on
this system using "man perl" or "perldoc perl". If you have access to the
Internet, point your browser at http://www.perl.org/, the Perl Home Page.

C:\>
```

Figure 4: Output from `perl -v`.

`SNPolisher` uses the output from APT genotyping of Axiom arrays as the input files. The Axiom Genotyping Solution Data Analysis Guide [8] should be used for all Axiom arrays. The goal of the Best Practices Genotyping Analysis Workflow in the Data Analysis Guide is to identify and remove poor quality samples in order to generate well-clustered SNPs for downstream analysis. A detailed description of the Best Practices Genotyping Analysis Workflow is available at [8].

The files needed to run `SNPolisher` are generated from steps 1 – 7 in the Best Practices Genotyping Analysis Workflow (see figure 5). These steps encompass first pass genotyping for initial SNP assessment, and are described in detail in [8]. See [8] for more information on using APT with the Best Practices Genotyping Analysis Workflow.

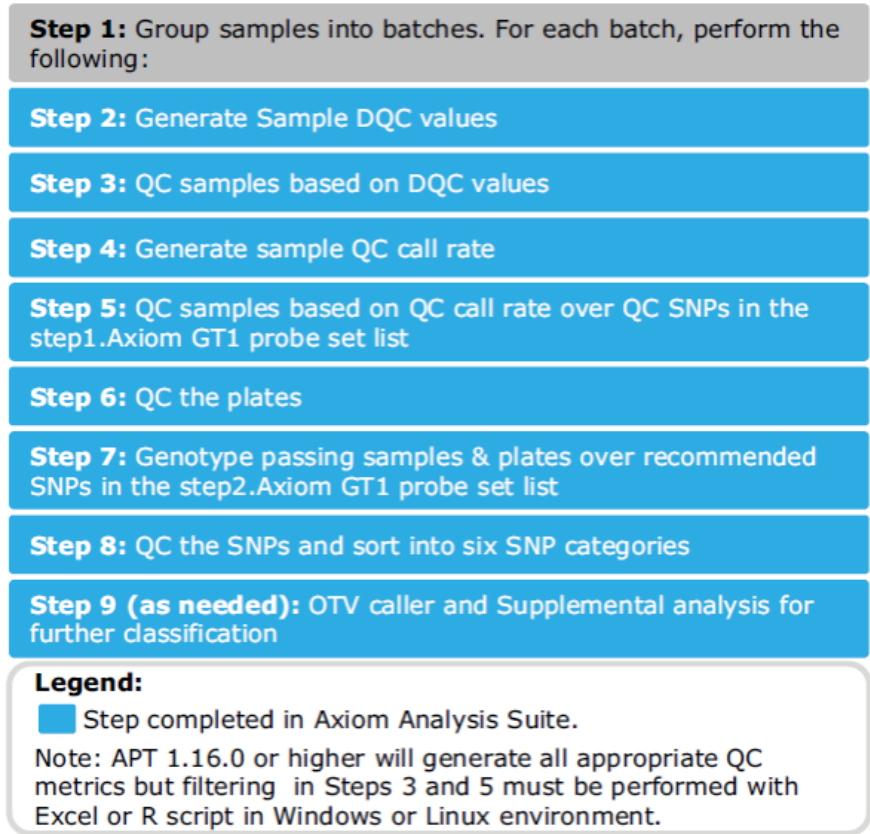


Figure 5: The Best Practices Genotyping Analysis Workflow

3.1 Downloading SNPolisher

The R package files to install SNPolisher are available on the Thermo Fisher website (<http://www.thermofisher.com/>). Select “Sign In” at the top of the website to register your email address with Thermo Fisher. Once you have registered, SNPolisher can be downloaded from the “DevNet Tools” page. From the home page, select “Life Sciences”, then “Microarray Analysis”, then “Microarray Analysis Partners & Programs”, then “Affymetrix Developers Network”, and then “Affymetrix DevNet Tools”. SNPolisher is available under “Analysis Tools”. Download the zipped SNPolisher folder (SNPolisher_package.zip file).

The zipped folder contains the R package file (SNPolisher_XXXX.tar.gz file, where XXXX is the release number); the user guide; the license, copyright, and readme files; a PDF with colors for use in R; and a data folder with example data for running in R.

The zipped folder is not a package binary and cannot be used to install SNPolisher. The folder must be unzipped to access the package file (SNPolisher_XXXX.tar.gz). If the zipped package folder is used in an installation attempt, R will return an error message that the package file is corrupted and cannot be installed. Unzip the folder and then redo the installation with the SNPolisher_XXXX.tar.gz file.

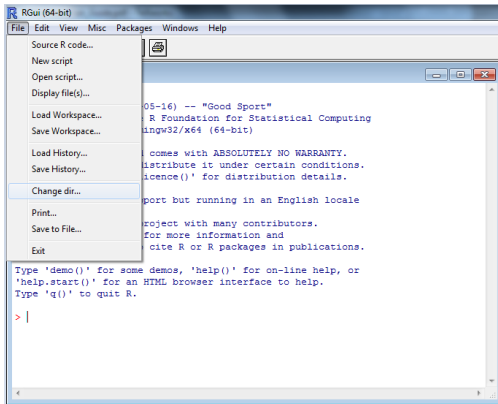
3.2 Installing SNPolisher in R

Before installing a package, R must know where the downloaded package file is located on the computer relative to the location that R is working in (the *working directory*). R has a default location on the computer that it loads and stores data from. There are two options to load data from or store data in a different folder: change the working directory to that folder, or tell R what the path is from the working directory to that folder. Changing the working directory is the recommended option for users with little experience working with command line software.

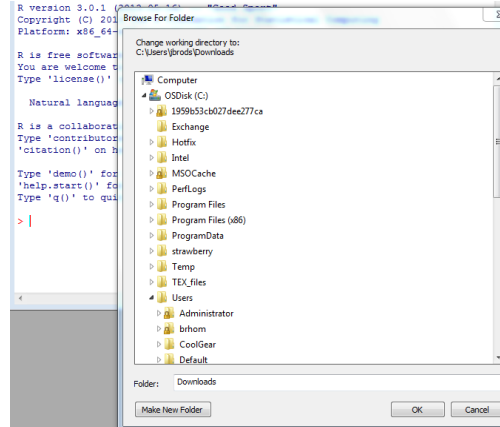
To change the working directory, click on “File” and then “Change dir...” (see figure 6a). A window for browsing to the folder where the package file is saved will appear. When the correct folder is selected, click on “OK” (see figure 6b). This will set the working directory to be the folder where the package was downloaded. The working directory will stay as this location until the end of the R session or until it is changed again. To install SNPolisher, type:

```
> install.packages("SNPolisher_3.0.tar.gz",repos=NULL,type="source")
```

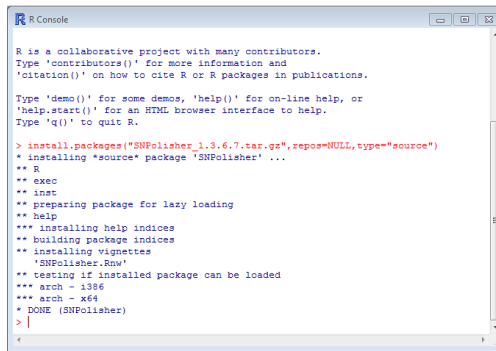
Be sure to use `type="source"` or R will attempt to download the package from CRAN instead of looking for an already downloaded package source file.



(a) Changing the working directory.



(b) Changing the working directory.



(c) Installed package in R.

Figure 6: Installing SNPolisher in R.

To tell R the path from the working directory to a folder, we need to know what the current working directory is. The command `getwd()` stands for “get working directory”. To change the working directory by hand, use the command `setwd()`, which stands for “set working directory”. Once we know the working directory, we can tell R where the downloaded file is.

```
> getwd()
[1] "C:/Users/name/Documents"
```

In this example, the file is in the *Downloads* folder. We need to tell R that the package file is in `C:/Users/name/Downloads`. *Documents* and *Downloads* both live in the *name* folder, so we need to tell R to go up one level to the *name* folder and then go down one level into the *Downloads* folder to find the package file. In R, two periods followed by a slash mean to go up one level: `../` Now that we know where the file we want to install is relative to the working directory, we can tell R where to find the file when we use the `install.packages` command:

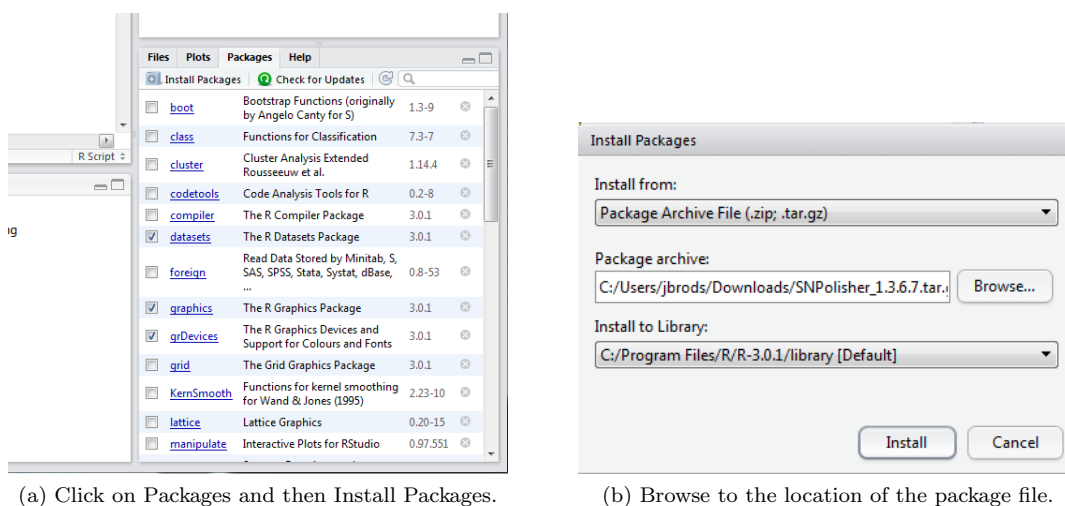
```
> install.packages("../Downloads/SNPolisher_3.0.tar.gz",repos=NULL,type="source")
```

If we wanted to change the working directory to be the *Downloads* folder before installing the *SNPolisher* package, we would type:

```
> setwd("../Downloads/")
> install.packages("SNPolisher_3.0.tar.gz",repos=NULL,type="source")
```

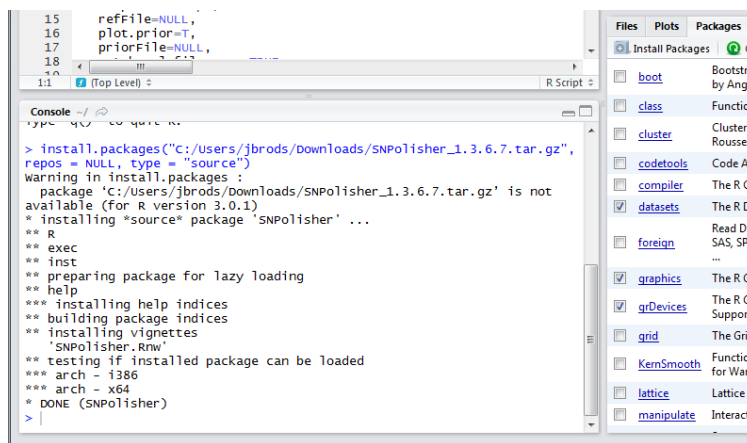
3.3 Installing SNPolisher in RStudio

RStudio is a GUI that calls R, so both R and RStudio must be installed. See appendix A for installation instructions. There is a menu for installing packages in RStudio. In the files window in the bottom right-hand corner, click on the *Packages* tab and then on *Install Packages* (see figure 7a). This will bring up a dialog box (see figure 7b). Set the first drop-down menu to “Package Archive File” instead of “Repository”. Click on the “Browse” button to locate the *SNPolisher tar.gz* file. Do not change the library location from the default. Click on “Install”. The console window will show that the package is installed (see figure 7c). It is fine if the version of R is newer than the one that *SNPolisher* was written for.



(a) Click on Packages and then Install Packages.

(b) Browse to the location of the package file.



(c) The console window will display that the package has been installed.

Figure 7: Installing SNPolisher in RStudio.

Because RStudio is a simply nicer way of displaying the R console and various other R windows, all typed commands work exactly the same in RStudio as in R. To display or change the working directory from the command line, follow the instructions in section 3.2. To change the working directory through the menu, click on “Session”, “Set Working Directory”, and “Choose Directory” (see figure 8).

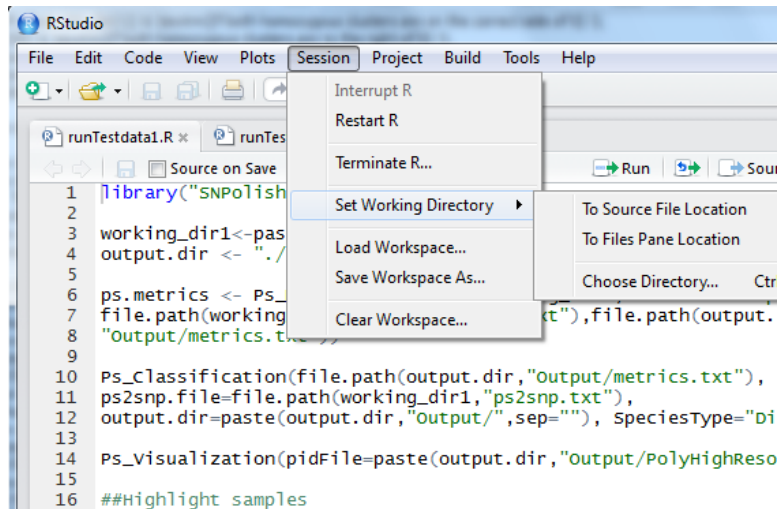


Figure 8: Changing the working directory in RStudio.

3.4 SNPolar Support

For Thermo Fisher clients, questions about `SNPolisher` and `SNPolisher` functions should be directed to your local FAS or to AFFYsupport@thermofisher.com. To contact support on the telephone, please call 1-800-955-6288.

4 SNPolisher Functions

SNPolisher contains functions for performing post-processing visualization on the genotyping results produced by APT genotyping for Axiom arrays. APT does not contain any plotting functions and APT users should use SNPolisher to produce plots. AxAS users may either use the SNP Cluster Graph function, or switch to SNPolisher and use the visualization functions in the package.

4.0 Tips For Running SNPolisher

4.0.1 Copying and Pasting in R

Copying and pasting into R from formatted files (e.g. Word documents) can produce unintended results. R expects plain text as the input and cannot understand most formatting characters. R treats formatting as typed text and turns formatting characters into data.

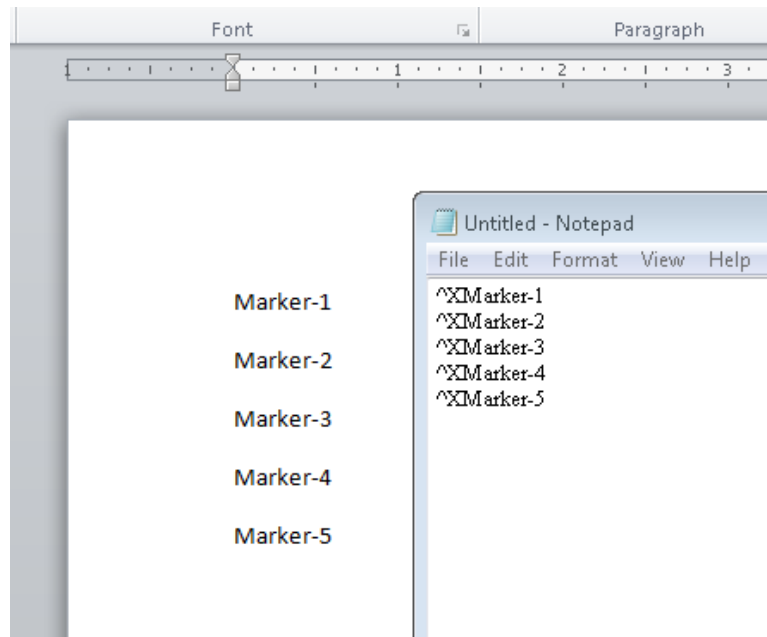


Figure 9: The same data file in Word and Notepad.

PDF files are highly formatted, including line breaks and hidden characters. Do not copy and paste from a PDF into R! If SNPolisher functions produce strange errors, check if the files have any hidden characters or formatting that R cannot process. Read the files into R and print them to the screen to see what R is reading.

The most common type of file that has formatting problems is lists of probesets or SNPs (PID files) because many lists are copied from emails, Word documents, or Excel files. To check that the formatting is not problematic, read the file in:

```
> hold <- read.table("list.ps", header=T, sep="\t")
```

To have R print the list, type the name of the list at the prompt:

```
> hold
```

4.0.2 Inputs, Arguments, and Outputs

The files that are used with a function in R are called *inputs*. The options for a function in R are called *arguments*. R functions take arguments (written before the equal sign) and are passed inputs (written after

the equal sign), written inside parentheses. This tells R to run the function using those arguments and to load the corresponding input for each argument.

```
> Ps_Visualization(callFile="AxiomGT1.calls.txt",  
  posteriorFile="AxiomGT1.snp-posteriors.txt")
```

Each argument has a name (e.g. the argument *posteriorFile*). When values are passed to a function using argument names, R knows exactly what to do with each value. If the function is passed values without any argument names, then R assumes that the values are entered in the same order that the arguments were written by the programmer when the function was first created. A function's help file will list the default order of the arguments.

If the example above was written without arguments, R would only understand that the calls file comes first and the posteriors file comes second if the function has been written to take the calls file before the posteriors file.

```
> Ps_Visualization("AxiomGT1.calls.txt","AxiomGT1.snp-posteriors.txt")
```

For *Ps_Visualization*, the calls file is the first argument and the calls file is the posteriors argument, so this command will run without problems. However, the next example command will not run because R will think that the posteriors file matches up to the *callFile* argument and will try to run the function using the posteriors file as the calls file. The user will get an error due to the different formatting in the calls and posteriors files:

```
> Ps_Visualization("AxiomGT1.snp-posteriors.txt","AxiomGT1.calls.txt")  
Error in data.frame(..., check.names = FALSE) :  
  arguments imply differing number of rows: 500, 0
```

To reduce any possible confusion about which values are being used by which arguments, we strongly recommend that the user types out each argument name that is being used. Then the order of the arguments will not matter. The following example will run even though the arguments are not in the expected order because the user has told R which input files should be used with which argument:

```
> Ps_Visualization(posteriorFile="AxiomGT1.snp-posteriors.txt",  
  callFile="AxiomGT1.calls.txt")
```

The functions in *SNPolisher* produce flat text files. The delimiting character is a tab. These output files can be opened in a plain text reader (like notepad) or in any program that can handle flat text files (like Word or Excel). *SNPolisher* output files may also be read back into R for more analysis if desired.

4.0.3 Working Directories

R working directories were briefly discussed in section 3.2 and 3.3. *SNPolisher* functions can produce a large number of output files, and it is important to keep track of which folder the files are written to and what they are named. Because *SNPolisher* functions have default names for most output files, it is possible to accidentally overwrite previously produced *SNPolisher* output when running a new *SNPolisher* section.

R allows a user to give a complete path to a file instead of only reading files in and outputting files to the working directory. This means that a user can load files from a folder that is not the working directory, and output files to a folder that is not the working directory. This is important when using *SNPolisher* because many of the input files are the APT outputs. For example, the input files for *Ps_Visualization* are the summary, posteriors and calls files produced by Step 7 of the Best Practices Genotyping Analysis Workflow. The user must either change the working directory to be the folder that contains the APT output files or must tell R where those files are. Directions for changing working directories were given in sections 3.2 and 3.3.

To tell R to use input files that are not in the working directory or to output files to somewhere other than the working directory, the user needs to know the path to the different folder. First, determine what the working directory is:


```
> getwd()
[1] "C:/Users/name/Documents"
```

Second, figure out what location the folder has relative to the working directory. If the input files are in `C:/Users/name/APT/Axiom/Output`, then the path from the working directory (the *Documents* folder) to the *Output* folder is `../APT/Axiom/Output`. To use the calls file as one of the inputs in *Ps_Visualization*, type `../APT/Axiom/Output/AxiomGT1.calls.txt` instead of `AxiomGT1.calls.txt`.

The user may also wish to put the *SNPolisher* output files in a different folder than either the working directory or the folder with the APT Axiom output files. Simply use the same process and give R the path from the working directory to the desired output folder. If the user wants to output the files to `C:/Users/name/SNPolisher/Output`, then the path from the working directory (the *Documents* folder) would be `../SNPolisher/Output`. If the user wants to store the output from *Ps_Visualization* as a file named *clusters.pdf* in the *SNPolisher/Output* folder, then the relative path and file name would be `../SNPolisher/Output/clusters.pdf`.

Assume that the user has changed the working directory to be the APT Axiom output folder and wants to put the output from *Ps_Visualization* in a folder for *SNPolisher* output. The working directory is:

```
> getwd()
[1] "C:/Users/name/APT/Axiom/Output"
```

and the desired output directory is `C:/Users/name/SNPolisher/Output`. Then the command to run *Ps_Visualization* is:

```
Ps_Visualization(posteriorFile="AxiomGT1.snp-posteriors.txt",
  callFile="AxiomGT1.calls.txt",
  summaryFile="AxiomGT1.summary.txt",
  output.File="../../../../SNPolisher/Output/clusters.pdf")
```

The output file *clusters.pdf* will be created in `C:/Users/name/SNPolisher/Output`.

R also accepts complete paths instead of paths relative to the working directory. The user could have produced the exact same output by typing the complete path to the output file:

```
Ps_Visualization(posteriorFile="AxiomGT1.snp-posteriors.txt",
  callFile="AxiomGT1.calls.txt",
  summaryFile="AxiomGT1.summary.txt",
  output.metricsFile="C:/Users/name/SNPolisher/Output/clusters.pdf")
```

4.0.4 Help Files

To find out more information about any function in R, type the help command or a question mark with that function name at the command line:

```
> help(Ps_Visualization)
> ?Ps_Visualization
```

In R, typing only the name of a function at the prompt without any parentheses is the command to print the R code for that function. For more complicated functions, this can result in R printing hundreds of lines of code.

```
> Ps_Visualization
```

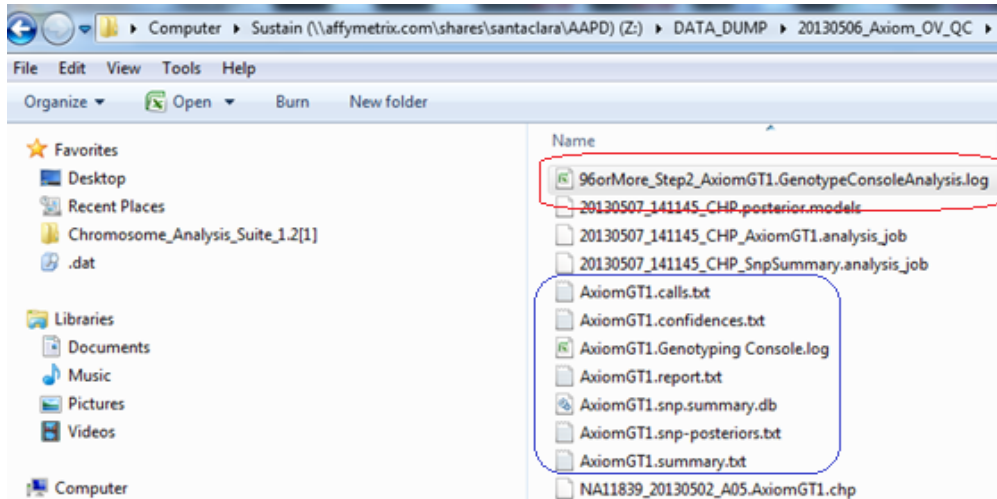


Figure 10: The output files from Step 2 genotyping in APT: the file in the red circle is the Step 2 analysis log, and the files in the blue circle are the AxiomGT1 output files.

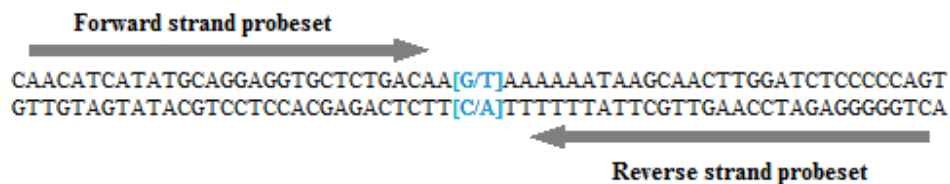
4.0.5 APT Files

In order to run *SNPpolisher*, the summary, calls, confidences, and posteriors files must be created by APT during Step 7 of the Best Practices Genotyping Analysis Workflow (“Step2_AxiomGT1”). Please take care to use the Step2_AxiomGT1 output files as input for *SNPpolisher* (enclosed in blue box in figure 10). The default names for these files are *AxiomGT1.summary.txt*, *AxiomGT1.calls.txt*, *AxiomGT1.confidences.txt*, and *AxiomGT1.snp-posteriors.txt*. *SNPpolisher* needs access to these files.

In APT, these files are located in the output folder selected using the `--out-dir` option with `apt-genotype-axiom`.

APT does not automatically produce the posteriors and summary files. Users must make sure to instruct APT to produce these files. When running Step 7 of the Best Practices Genotyping Analysis Workflow, make sure to use the `--write-models` and `--summaries` options when running `apt-genotype-axiom`.

See the Best Practices Genotyping Analysis Workflow, example `apt-genotype-axiom` script for Step 7 [8], for more details.



(a) A SNP site (blue) can be interrogated with two distinct probe sequences. The “best” probeset for a SNP is selected using classification types.

probeset_id	snpid
AX-11126382	Affx-6877593
AX-11126400	Affx-7010457
AX-11126410	Affx-7010457
AX-11126411	Affx-7054365

(b) A *ps2snp* file opened in Excel.

Figure 11: Forward and reverse probesets for a SNP, and a *ps2snp* file. SNP *Affx-7010457* has two probesets.

The file *ps2snp.txt* is an optional input file that should be used when analyzing results from Axiom arrays which include *de novo* SNP content or multiple probesets per SNP. The ps2snp file contains a list of all probesets on the array and the SNPs they belong to. This list must be used with *Ps_Classification* to select the best probeset per SNP. If an array only has one probeset per SNP for the entire array, then a ps2snp file is not generated.

A SNP may have multiple probesets because a SNP site can be interrogated with both a forward and reverse strand probeset (see figure 11). A file whose name includes the string “ps2snp” should be included in the “Analysis Library File” folder for any array with multiple probesets per SNP (see [8] for more information). This file contains two columns named *probeset_id* and *snp_id*. If this file is missing for an array with multiple probesets per SNP, please contact Thermo Fisher support.

4.0.6 Differences Between the SNPolisher and AxAS Versions of Cluster Plots

AxAS runs the complete Best Practices Genotyping Analysis Workflow and has a visualization function that is similar to *Ps_Visualization*. Both functions produce graphs with the same basic colors and shapes, but each function has special features. The SNP Cluster Plot function in AxAS includes the *lasso* tool that allows the user to interact with a plot and select samples by circling them with the mouse. *Ps_Visualization* has a different system for plotting multiallelic probesets that includes all samples and calls in one plot. *Ps_Visualization* can also plot large data sets with multiple batches of samples genotyped across the same SNPs.

Note that the R package functions can take the Axiom summary, calls, confidences, and posteriors files as input when they are compressed in the gzip compression file format (“gzipped”) while Axiom Analysis Suite cannot.

4.0.7 Out of Memory Errors in R

Although R has many wonderful qualities to recommend it, it is also famous for poor memory usage. Out of memory errors in R are similar to this:

```
Error: cannot allocate vector of size 8.1 Gb
Reached total allocation of 6135Mb: see help(memory.size)
```

Out of memory errors are generally caused by two different scenarios: the user accidentally attempted to create a data object in R that really was larger than the amount of RAM on the computer, or there was not enough *contiguous* bits of RAM to store a new object. The first scenario can be solved by either acquiring more RAM or reducing the size of the object. The second scenario can occur even when there is technically enough memory to store the new data object. In this case, the problem is that R does not have access to enough memory that is not “split up”. One possible solution is to increase the amount of memory R has access to.

We have several suggestions for dealing with the out-of-memory errors that can occur when running SNPolisher with large Axiom files:

Make sure to install a 64-bit version of R and perl Memory usage is limited to 4 Gb (linux) or 2 Gb (older versions of Windows) on a 32-bit installation. In contrast, memory usage is limited to 8 Tb on a 64-bit installation. Do not install a 32-bit version of R. Note: if a 64-bit version of R or perl is installed on a computer with a 32-bit OS, then the 32-bit memory limits still apply.

Set the amount of memory manually This can be a complicated procedure, and the steps are different depending on what OS R is installed on (Windows, Mac, or linux). To see the amount of memory available, use the `memory.size` command with the `max` argument. When `max=FALSE`, R returns the amount of memory in Mb currently in use. When `max=TRUE`, R returns the maximum amount of memory in Mb available from the OS. When `max=NA`, R returns the limit on memory in Mb. To increase the memory limit on Windows, use the command `memory.limit` with argument `size` set in Mb. To increase the memory limit on unix/linux, use `ulimit` or `limit`.

Compress the Axiom output files The SNPolisher functions can take gzipped files as input. To reduce the size of the files, they may be compressed using the gzip compression file format. For Mac OS X

and Unix/Linux, use the `gzip` command on the command line. For Windows, compression software such as 7zip ([31]) is needed.

Create smaller versions of the Axiom output files The `SNPolisher` functions in the Best Practices Genotyping Analysis Workflow are SNP-independent: each SNP or probeset is evaluated separately from all other probesets. Because of this independence, the Axiom output files (summary, calls, confidences, and posteriors) can be split up into multiple files without affecting the results of the Best Practices workflow. Make sure that the smaller versions of the Axiom output files have the same set of probesets across the smaller summary, calls, confidences, and posteriors files so that some probesets are not inadvertently left out of the analysis.

For a very thorough explanation of memory usage in R, see Hadley Wickham's description and exercises ([23]).

4.1 APT File Formats

The `SNPoli` package can take inputs with copy-number aware genotyping and multiallelic SNPs. Due to this, there are more possible input files with different file formats than in previous versions. The major formatting updates involve call code assignments and multiallelic priors and posteriors.

4.1.1 Dynamic Call Code Assignments

Numeric call codes were previously static assignments, and only a certain set of genotypes were assigned call codes: OTV (-2), NoCall (-1), AA (0), AB (1), and BB (2). With the introduction of copy-number aware genotype calls and multiallelic calls, numeric call codes are assigned dynamically in APT during the genotyping process. This means that the number call code 7 may indicate the genotype *CD* in one calls file and the genotype *BF* in another calls file. The dynamically assigned call codes are listed in the header comments of each calls file. The previous set of assigned call codes listed above (-2 to 2) have been reserved and will always be assigned to the listed genotype calls. Older calls file will not have the call codes listed, and `SNPoli` will use the reserved call codes for these files. `SNPoli` cannot process any calls file that have more numeric call codes than the reserved call codes but which do not have the call codes listed in the header. All `SNPoli` functions will stop with an error message in this case.

```
##call-code-1=OTV_1:-4:1
##call-code-2=NoCall_1:-3:1
##call-code-3=OTV:-2:2
##call-code-4=NoCall:-1:2
##call-code-5=AA:0:2
##call-code-6=AB:1:2
##call-code-7=BB:2:2
##call-code-8=ZeroCN:3:0
##call-code-9=A:4:1
##call-code-10=B:5:1
##call-code-11=C:6:1
##call-code-12=AC:7:2
##call-code-13=BC:8:2
##call-code-14=CC:9:2
##call-code-15=D:10:1
##call-code-16=AD:11:2
##call-code-17=BD:12:2
##call-code-18=CD:13:2
##call-code-19=DD:14:2
##call-code-20=E:15:1
##call-code-21=AE:16:2
##call-code-22=BE:17:2
##call-code-23=CE:18:2
##call-code-24=DE:19:2
##call-code-25=EE:20:2
```

Figure 12: The raw numeric call code assignment in the header of a calls file.

Autotetraploid data produced by running the *fitTetra* functions do not have a call code assignment table. There is another set of reserved numeric call codes that are used only with autotetraploid data: AAAA is 0, AAAB is 1, AABB is 2, ABBB is 3, and BBBB is 4. NoCall and OTV remain -1 and -2. The `SNPoli` functions can handle autotetraploid data automatically as long as the posteriors file is supplied.

4.1.2 Multiallelic SNPs

Multiallelic SNPs can potentially have many more genotypes called than biallelic SNPs. This change in genotyping options affects the calls, summary, posteriors, and priors files. The calls files now have dynamic call code assignments and have as many call codes as there are genotypes across a set of SNPs. The summary files contain one row per allele per probeset. In previous versions, this meant that each probeset had two rows of intensity values in a summary file. Now, a probeset has as many rows as there are alleles genotyped for that probeset. There are still two rows per probeset for a biallelic SNP, and there is now a variable number of rows for multiallelic SNPs. If a multiallelic SNP has alleles A, B, and C, then there will be three rows in the summary file. If a multiallelic SNP has alleles A, B, C, and D, there will be four rows in the summary file.

One major difference between biallelic SNPs and multiallelic SNPs is that the data transformation for multiallelic SNPs is not size versus contrast: it is base 2 logarithmic transformation for each allele. If a multiallelic SNP has alleles A, B, and C, then the transformation is $\log_2(A)$ versus $\log_2(B)$ versus $\log_2(C)$. For a discussion of plotting multiallelic SNPs, see 4.2.

The formatting for multiallelic posteriors and priors files is very different from the biallelic posteriors and priors files. Biallelic and multiallelic posteriors and priors data are not combined together in one file like the calls and summary data. There is one row of data per cluster per probeset, not just one row per probeset. If a multiallelic probeset has alleles A, B, and C, then the multiallelic posteriors file will have one row for each of the clusters: AA, AB, BB, AC, BC, CC. Each row has a value for “copynumber” (usually 2) and for “nAlleles” which is the total number of alleles genotyped in the probeset. In this example, there are three alleles so the value of “nAlleles” is 3 for all rows of the probeset.

The means per cluster are the mean value of the cluster for each of the SNP’s alleles. The value of “AA–meanB” is the mean location in $\log_2(B)$ space of the AA cluster. The covariance values are presented in a similar fashion: the value of “AA–varB” is the variance in $\log_2(B)$ space of the AA cluster and “AA–covarAB” is the covariance in $\log_2(A)$ and $\log_2(B)$ space of the AA cluster. See figure 13 for a figure of what the means and covariances are for a multiallelic SNP.

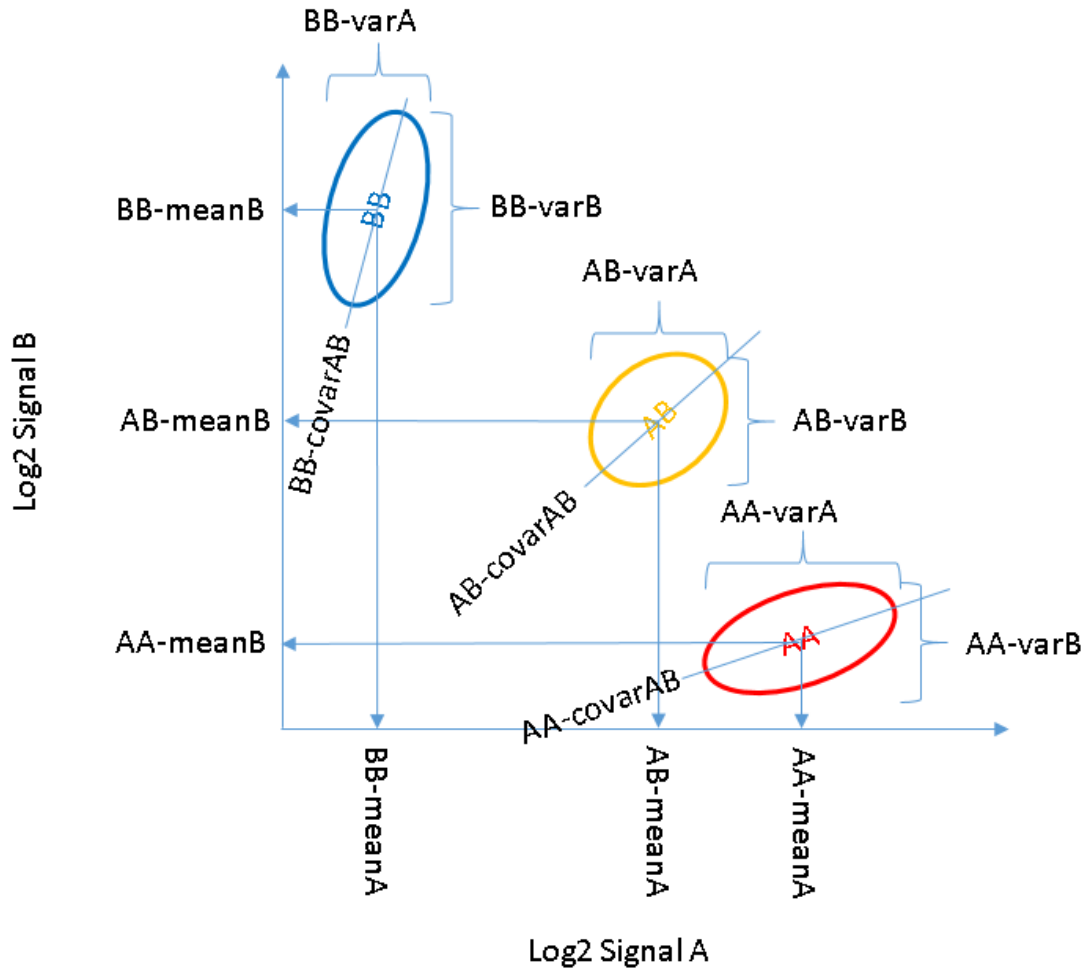


Figure 13: The visual interpretation of multiallelic mean and covariances.

The mean and covariance values are formatted as numeric values pasted together with a comma, while each column in the multiallelic posteriors and priors files is separated with tabs. The number of means and covariances per cluster depend on the number of alleles per SNP. For a SNP with three alleles, each cluster has three means and 6 covariances. A SNP with four alleles has four means and 10 covariances per cluster. Because the number of means and covariances is different for SNPs with a different number of alleles, the order in which the means and covariances appear is given in the headers. Figure 14 shows the headers with the orders of the means and covariances for SNPs with 3, 4, and 5 alleles. The first SNP in the file is also shown. It has 4 alleles and has 4 means and 10 covariance values per cluster.

```

##%data-order-mean-nalleles-3=A,B,C
##%data-order-mean-nalleles-4=A,B,C,D
##%data-order-mean-nalleles-5=A,B,C,D,E
##%data-order-covariance-nalleles-3=varA,covAB,covAC,varB,covBC,varC
##%data-order-covariance-nalleles-4=varA,covAB,covAC,covAD,varB,covBC,covBD,varC,covCD,varD
##%data-order-covariance-nalleles-5=varA,covAB,covAC,covAD,covAE,varB,covBC,covBD,covBE,varC,covCD,covCE,varD,covDE,varE

```

probeset_id	copynumber	nAlleles	cluster	mean	nObsMean	covariance	nObsVar
AX-165679670	2	4	AA	11.971223,9.23	0.2	0.030000,0.000000,0	1
AX-165679670	2	4	AB	11.771223,10.7	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	AC	11.771223,9.23	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	AD	11.771223,9.23	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	BB	10.271197,10.9	106	0.006743,0.004664,0	106
AX-165679670	2	4	BC	10.271223,10.7	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	BD	10.271223,10.7	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	CC	10.271223,9.23	0.2	0.030000,0.000000,0	1
AX-165679670	2	4	CD	10.271223,9.23	0.3	0.030000,0.000000,0	1
AX-165679670	2	4	DD	10.271223,9.23	0.2	0.030000,0.000000,0	1

Figure 14: The multiallelic posteriors for one SNP, with the mean and covariance orders in the header of the multiallelic posteriors file.

If the headers of a calls or posteriors file is missing, **SNPolisher** will use default values. For call codes, this is the standard values listed in 4.1.1. The default order of the means and covariances is alphabetical by allele. If the headers have been removed from a multiallelic posteriors or priors file and the order of the means and covariances is not the default order, then any results from a **SNPolisher** function will be incorrect. Users must be careful not to delete the headers of the calls and multiallelic posteriors and priors files.

The multiallelic genotypes are produced from combining the results of multiple biallelic probesets that interrogate all of the possible alleles for a multiallelic SNP. The final Axiom genotyping calls, confidences, posteriors, and summary data are given for the final multiallelic data. *Ps_Vis_Multi_ID* is designed to plot all probesets that have contributed to the assigned genotype calls of a multiallelic SNP. This is the only function where a 4-column p2snp file is required.

4.1.3 Special SNPs

Non-autosomal SNPs are referred to as *special SNPs* and are handled differently than the autosomal SNPs. There are seven genomic regions that have special SNPs: the non-pseudoautosomal region of the X chromosome (non-PAR X), the pseudoautosomal region of the X chromosome (PAR), the Y chromosome (Y), the Z chromosome (Z), the W chromosome (W), mitochondrial SNPs (MT), and chloroplast SNPs (CP). These regions can have a variable number of allele copies based on gender, or have fewer than two allele copies for all genders. The special SNPs file lists all special SNPs for an array and is included as part of the library files for each array. The report file lists the genders for the samples that are genotyped and a new report file is generated when samples are genotyped.

Region	Gender	Copy Number	Notes
non-PAR X	female	2	
non-PAR X	male	1	
PAR	all	2	treated as autosomal
Y	female	0	
Y	male	1	
Z	female	1	avian species only
Z	male	2	avian species only
W	female	1	avian species only
W	male	0	avian species only
MT	all	1	
CP	all	1	

Table 1: Special SNPs with gender and copy number information.

In the `SNPolisher` functions, samples are separated by gender for non-PAR X, Z, Y, and W probesets when the special SNPs and report files are supplied. MT and CP probesets are treated identically; non-PAR X and Z probesets are treated identically except that the male and female handling is switched; and Y and W probesets are treated identically except that the male and female handling is switched. The different treatment by gender is because of the difference in copy number by gender. The expected behavior of male and female samples are different due to the different number of alleles and the `SNPolisher` functions take this into account when plotting the genotype clusters.

4.2 Ps_Visualization

Ps_Visualization generates plots of the genotype cluster patterns for a set of selected SNPs. Plots may include the posterior and prior information. OTV genotypes, reference genotypes, and confidence levels may also be included. The cluster plots can help quality check SNPs and identify underlying genotyping problems.

Major updates were made to *Ps_Visualization* with the release of **SNP_{olisher} 3.0**. *Ps_Visualization* plots biallelic SNPs, multiallelic SNPs, batch plots, and gender-separated plots for special SNPs; users can set the output file type and the output directory, individual genotype colors for biallelic plots, the size of the output plots, whether all biallelic pairwise combinations should be plotted for multiallelic SNPs, and separate labels for each batch per probeset; and users can supply sample lookup files, presence/absence allele files, and sample QC metrics files.

4.2.1 Inputs and Arguments

Ps_Visualization takes three required arguments: the pid file (*pidFile*), the summary file (*summaryFile*), and either a calls (*callFile*) or a confidences file (*confidenceFile*). These arguments provide *Ps_Visualization* with a list of SNPs to plot, the locations of each sample that was genotyped for each SNP, and data that is used to color the samples. All other input files and parameters are optional.

`Ps_Visualization(pidFile,summaryFile,callFile,confidenceFile)`

- *pidFile*: text file listing probeset IDs
- *summaryFile*: Axiom summary file (usually “AxiomGT1.summary.txt”)
- *callFile*: Axiom calls file (usually “AxiomGT1.calls.txt”)
- *confidenceFile*: Axiom confidences file (usually “AxiomGT1.confidences.txt”)

Due to the changes in how genotype call codes are assigned (see section 4.1), the calls file must have the header comments which contain the list of calls that appear in the file and which numeric call codes are assigned to those calls. This assignment is dynamic and is different for each calls file. The call codes -2, -1, 0, 1, and 2 are reserved for OTV, NoCall, AA, AB, and BB. All other genotype calls will have different call codes from each genotyping run. For autotetraploid data, the call codes 0, 1, 2, 3, and 4 are reserved for AAAA, AAAB, AABB, ABBB, and BBBB.

If the call code assignments are missing, *Ps_Visualization* will use the reserved call codes. This ensures that older calls files without headers will still be handled correctly for plotting. Any calls file with call codes that are not in the reserved set of call codes but without headers cannot be used for plotting.

The list of probeset IDs (*pidFile*) can either be the list of probesets produced by the Axiom Best Practices Workflow for one category or a list of probesets selected by the user. The first line of the *pidFile* should always be “probeset_id”.

The many optional arguments allow the user to specify various aspects of the plots and output files. Some arguments handle the input file options (table 2); some arguments handle the shapes, colors, and X and Y limits (tables 3 and 4); and some arguments handle the options for the output files (table 5).

The summary, calls, posteriors, and confidences files may be gzipped and used as input files. R handles reading compressed gzipped files internally without needing to expand them.

The posterior file holds the information about the size and shape of the posterior ellipses for each biallelic probeset. The multiallelic posterior file holds the same information for multiallelic probesets (but with a different formatting than the biallelic posteriors file). The prior file and multiallelic priors file hold the information about the size and shape of the prior ellipses for the probesets.

The multiallelic priors and posteriors files must contain the APT header comments which include the order of the means and covariances for each allele number. If the header comments are missing, *Ps_Visualization* will use the generic set of mean and covariance orders, which handles up to 12 alleles. If there are more than 12 alleles or if the mean and covariance order is not the standard order, then the priors and posteriors will be plotted incorrectly. A warning message is output when the generic mean and covariance orders are used.

The generic values for multiallelic priors and posteriors are not included in the multiallelic priors and posteriors file. A file containing the generic values is included in the `Perl` subfolder of the **SNP_{olisher}**

Argument	Description
posteriorFile	Axiom biallelic posteriors file (usually “AxiomGT1.snp–posteriors.txt”)
multiallele.posteriorFile	Axiom multiallelic posteriors File (usually “AxiomGT1.snp–posteriors.multi.txt”)
priorFile	Axiom biallelic priors file (usually “AxiomGT1.priors.txt”)
multiallele.priorFile	Axiom multiallelic priors file (usually “AxiomGT1.priors.multi.txt”)
refFile	file with the reference genotype calls
specialSNPsFile	file listing the special SNPs (report file must be supplied with the special SNPs file)
reportFile	file listing the genders of the samples (special SNPs file must be supplied with the report file)
sampleFile	file listing the samples to be highlighted and the sample color
labelsFile	file listing the labels to be used as main plot titles instead of SNP names
sampleLookupFile	file matching up the same samples in the calls and references files that have different names
sampleQCFile	file with a continuous QC sample metric used to adjust the opacity of sample colors in the plots
presenceAbsenceFile	file listing the presence and background allele of the subset of probesets that are plotted with the PAV transformation
temp.dir	name of the temporary directory created by <i>Ps_Visualization</i>
keep.temp.dir	flag to keep or remove the temporary directory
use.temp.dir	flag to use an already existing temporary directory

Table 2: The input file arguments for *Ps_Visualization*.

package. To access this file outside of *Ps_Visualization*, the user must navigate to the library location where the package is installed and find the `Perl` subfolder.

The reference file holds the information for any known reference genotype calls for the samples. The source of reference genotypes may be a SNP discovery project such as HapMap or the 1000 genomes project, or it may be genotypes produced in an NGS project for the user’s samples. The user must create the *refFile* with the same format and genotype codes as the calls file. Samples without reference genotypes have their reference plotted as a gray square for “No Call”. If there are very few reference genotypes, the reference plot will contain mostly gray squares. If the vast majority of the reference calls are NoCalls, the *refNoCalls* option can be used to only plot assigned reference calls. The reference file must contain reference calls for at least one sample in the calls file. The default value is NULL.

The sample lookup file contains the probesets that appear in the calls and reference files, but which have been assigned different names in each file. There should be two columns separated by a tab, and the two columns must be titled “Tag” and “SampleName”. *Ps_Visualization* will match up the assigned genotype call and the reference genotype call for the same sample with different names when this file is provided. If the reference and calls files have the same sample names, the sample lookup file is not necessary and should not be supplied.

The special SNPs file has information about which chromosome the SNPs or probesets are located on. There should be two or more columns separated by a tab, and two columns must be titled “probeset_id” and “chr”. The report file contains a list of the samples and which gender they are. There should be two or more columns separated by a tab, and two columns must be titled “Sample” and “ComputedGender” or “cel_files” and “computed_gender”. Gender-separated plots can be used to inspect the locations of the male and female samples and to determine if a special SNP has produced clusters in the expected locations by gender. Both the report file and the special SNPS files are necessary to produce gender-separated plots. If a special SNPs

file is provided but a report file is not, *Ps_Visualization* will output a warning that gender-separated plots will not be made. See **Gender-separated Plotting** (4.2.1.3) for more information. The default values are NULL.

Ps_Visualization will output a warning if a special SNPs file or a report file are provided without the other file. Both a special SNPs file and a report file are required to create gender-separated plots.

The sample file contains the list of samples that should be highlighted. There should be two or more columns separated by a tab, and two columns must be titled “sample” and “color”. The same color may be used for all samples or a different color may be used for different groups of samples. A sample file with only one column is no longer an option as it was in previous versions. Colors may be written as names, hexidecimals, or in RGB format. See **Selecting Colors in R** (4.2.1.1) for more information on how R formats colors. The default value is NULL.

The labels file contains the list of titles for the plots instead of the default titles of the SNP name. There should be two or more columns separated by a tab, and two columns must be titled “probeset_id” and “label”. The labels file can list one SNP up to the full set of plotted SNPs. Any listed SNPs that are not in the *pidFile* are ignored, so a user may prepare one larger labels file and use that with multiple different *pid* files. The default value is NULL.

The sample QC file one continuous-valued QC metric with a value for each sample. The colors used in the cluster plots are adjusted in opacity (complete opaque to completely see-through) with opacity values scaled to the range of the sample QC metric. Use this option carefully, as differences in opacity may not produce the expected changes in color.

The presence/absence file is a library file listing which allele is present and which allele is background for some probesets. There should be two columns separated by a tab, and the two columns must be titled “probeset_id” and “presence_allele”. Any biallelic probeset that is listed in the presence/absence file will be plotted with the *PAV* transformation, regardless of which transformation the user has selected for the other probesets, while biallelic probesets that are not listed in the file will be plotted with the user-selected transformation.

Temporary Directories and Files

The user should give *Ps_Visualization* the name of a temporary directory which will be used for outputting one file per SNP (*temp.dir*). If no directory is given, the default used is **Temp/**. The accompanying flag *keep.temp.dir* indicates whether the temporary directory should be kept or deleted at the end of *Ps_Visualization* (default is **FALSE**). The option *use.temp.dir* is a flag to indicate if there is an already created temporary directory with files for all SNPs to be plotted that should be used instead of creating a new temporary directory (default is **FALSE**). This option should be used when a directory has all of the individual SNP files already created, which allows *Ps_Visualization* to skip the step of extracting the data for each SNP from the summary, calls, confidences, and reference files. If such a directory exists, the user should input the location of the directory to *temp.dir*.

Ps_Visualization generates one temporary file per SNP to be plotted. These temporary files are stored in the temporary directory. The files contain the summary, calls, confidences, and reference data for each SNP. The file names are the probeset names. Each file contains one row per data type (calls, confidences, references) and one row per allele line (summary). The first column is the SNP name. The second column is the data type. The third column contains the data values from the files, separated by commas.

probeset_id	file_type	data
AX-11143316	summary	AX-11143316-A,1725.63159,2304.90820,2288.48169
AX-11143316	summary	AX-11143316-B,451.13852,446.98730,453.54608,431
AX-11143316	calls	AX-11143316,0,0,0,0,1,0,0,2,0,0,1,0,1,0,1,0,0,0,0,0,

Figure 15: The temporary file for a SNP from *Ps_Visualization*.

The arguments *plot.intensity* (default is **TRUE**), *plot.genotype* (default is **TRUE**), and *plot.ref* (default is **FALSE**) control if plots are made with the intensity data (e.g. A versus B), the transformed data (e.g. contrast versus size), or reference data. The default values are to plot the intensity and genotype calls (**TRUE**), which matches the previous versions of *Ps_Visualization*. Plotting both types of data produces the default two

Argument	Description
plot.intensity	flag indicating if intensity plots should be made
plot.genotype	flag indicating if genotype plots should be made
plot.calls	flag indicating if genotype calls should be used to assign shapes and colors in plots
plotNoCalls	flag indicating if NoCalls should be plotted with genotype calls
plot.conf	flag indicating if confidences should be used to assign colors in plots
plot.ref	flag indicating if reference calls should be used to assign shapes and colors in plots
refNoCalls	flag indicating if NoCalls should be plotted with reference calls
plot.prior	flag indicating if prior ellipses should be plotted
plot.prior.ref	flag indicating if prior ellipses should be plotted on reference plots
plot.posterior	flag indicating if posterior ellipses should be plotted
plot.posterior.ref	flag indicating if posterior ellipses should be plotted on reference plots
plot.pca	flag indicating if the PCA plots should be plotted for multiallelic probesets
transform	which data transformation should be used when plotting genotype plots
K	variable used with several transformations
autotetraploid	flag indicating if the data comes from an autotetraploid species
unknownGender	flag indicating if samples with unknown gender should be included in gender-separated plots
confidences.cutoff	calls with confidences larger than this option are plotted as NoCall
conf.thresh	threshold values for confidences plotting
conf.col	colors for confidences plotting

Table 3: The arguments for controlling shape, color, and plotting types in *Ps_Visualization*.

plots per probeset. Plotting both types of data with the reference data produces the previously standard four plots per probeset.

The arguments *plot.calls* (default is `TRUE`) and *plot.conf* (default is `FALSE`) control what values are used to color the samples in the plots. *plot.calls* colors the samples by genotype calls, while *plot.conf* colors the samples by which bin each sample’s confidence value falls into. If there are a lot of NoCalls in a calls plot, the *plotNoCalls* argument can be used to only plot genotype calls that are not NoCall. Similarly, the *refNoCalls* option can be used to plot only calls that aren’t NoCall in the reference plots. The argument *conf.thresh* takes 5 threshold values for sorting the confidences into four bins. Threshold values must be in the range [0,1]. The default values are (0,0.01,0.05,0.10,0.15). *conf.col* takes the list of four colors for coloring samples when *plot.conf* is `TRUE`. The default is `c("green", "gold", "darkorange1", "firebrick1")`.

The logical operators *plot.prior* and *plot.posterior* indicate if the prior and posterior ellipses are plotted (default is `FALSE` for priors and `TRUE` for posteriors). If *plot.prior* or *plot.posterior* are `FALSE`, *Ps_Visualization* ignores *priorFile*, *multiallele.priorFile*, *posteriorFile*, and *multiallele.posteriorFile*. If *plot.prior* or *plot.posterior* are `TRUE` and *priorFile*, *multiallele.priorFile*, *posteriorFile*, or *multiallele.posteriorFile* are not supplied, then *Ps_Visualization* plots a generic prior or posterior for each argument that doesn’t have an input file. Prior and posterior ellipses are plotted in the same colors as the clusters. Prior ellipses are dashed and posterior ellipses are solid. Note that if a cluster is empty and the prior and posterior ellipses are in the same location, then the posterior ellipsis will sit directly on top of the prior ellipses and it will appear that the prior was not plotted. See figure 23 for an example of what this looks like. In multiallelic plots, the generic priors are plotted with a slight shrinkage parameter to make them visible when the generic posteriors are plotted as well.

The flag *plot.pca* controls if the principal component analysis (PCA) plots should be created for multiallelic probesets. The flag *unknownGender* controls if samples with unknown genders should be plotted in gender-separated plots.

transform specifies which data transformation is used for the calls and confidences plots. There are seven transformations: “log2”, “MvA”, “CES”, “CSS”, “RvT”, “MvA-sqrt”, and “polar”. The default is “MvA” (aka

size versus contrast). K is a parameter used with the “CES” and “CSS” transforms and must be an integer value from 1 or 5 (default is 2). With the default “MvA” transform, the X axis is $contrast = \text{Log}_2\left\{\frac{A_{signal}}{B_{signal}}\right\}$ and the Y axis is $size = \left\{\frac{\text{Log}_2(A_{signal}) + \text{Log}_2(B_{signal})}{2}\right\}$.

The “CES” transform should be used with autotetraploid data. *Ps_Visualization* can automatically determine if the data is autotetraploid from the posteriors file. If the posteriors file is missing, the autotetraploid argument must be set to `TRUE` and the transform option must be set to “CES”. When the data is autotetraploid, the standard genotype call codes produced by `fitTetra_Output` (4.6.2.2) are used. NoCall and OTV samples are still colored grey and cyan, respectively. AAAA samples are plotted as red triangles, AAAB samples are gold circles, AABB samples are blue inverted triangles, ABBB samples are dark green pluses, and BBBB samples are purple crosses. If *transform* is not set to “CES”, the plots will be visibly incorrect. In this case, the user should double-check that the correct data transformation was used.

If a posteriors file is not supplied and *autotetraploid* is not set to `TRUE`, *Ps_Visualization* will not be able to tell that the data is from an autotetraploid species. In this case, the incorrect call shapes and colors will be used and the ABBBB and BBBB samples will be treated as though they are haploid (i.e. A and B calls). If haploid calls appear instead of autotetraploid, *autotetraploid* should be set to `TRUE`.

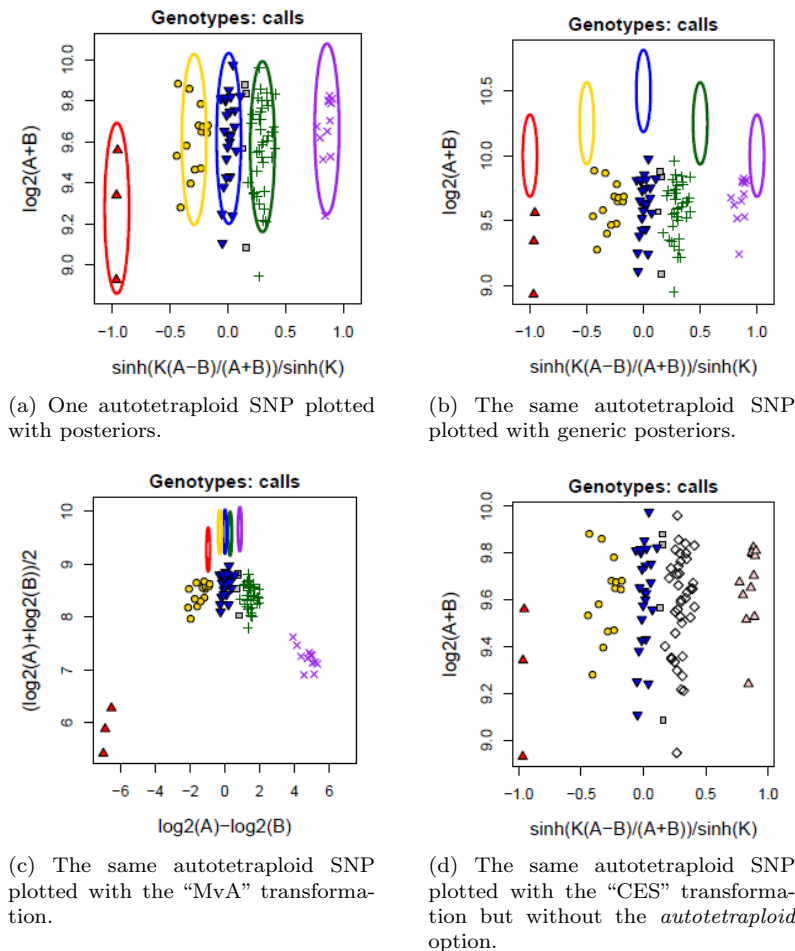


Figure 16: One autotetraploid SNP plotted with posteriors, generic posteriors, and the “MvA” and “CES” transformations.

confidences.cutoff controls if a sample is plotted as a “No Call” instead of using the call value in the calls file when the matching confidences value is larger than *confidences.cutoff*. To plot all samples using their given calls in the calls file (including “No Call”), set *confidences.cutoff* to be 1. If a confidences file is

supplied but no value is given for *confidences.cutoff*, then the default value of 1 will be used. This will result in no changes to the calls.

A confidences file is necessary to use the *plot.conf*s and *confidences.cutoff* options. If a confidences file is not supplied with these options, *Ps_Visualization* will stop with an error message. A gzipped confidences file may be used as input.

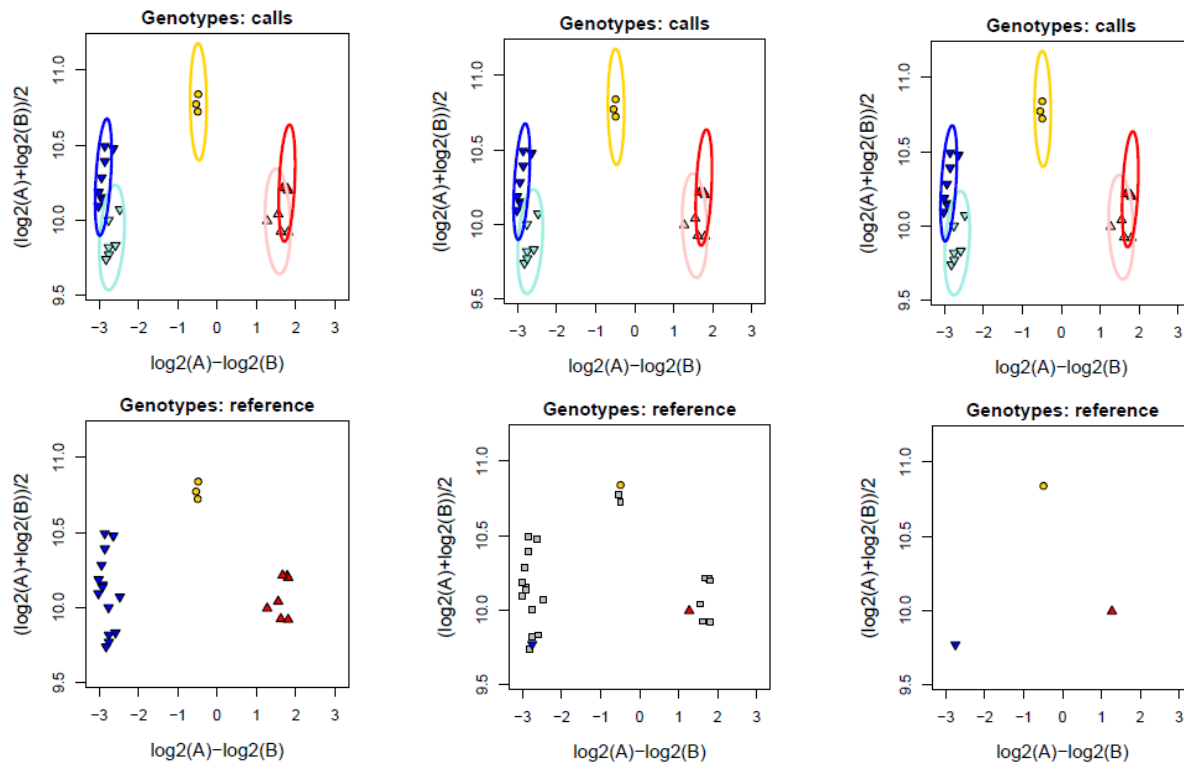
Argument	Description
col.AA	the color for AA calls
col.AB	the color for AB calls
col.BB	the color for BB calls
col.A	the color for A calls
col.B	the color for B calls
col.CN0	the color for CN0 calls
col.OTV	the color for OTV calls
col.NC	the color for NoCalls
col.OTV1	the color for OTV1 calls
col.NC1	the color for NC1 calls
arrow.col	the color for the PCA arrows
xlim.intensity	file-wide X range limits for intensity plots
yylim.intensity	file-wide Y range limits for intensity plots
zlim.intensity	file-wide Z range limits for intensity plots
xlim.genotype	file-wide X range limits for genotype plots
yylim.genotype	file-wide Y range limits for genotype plots
zlim.genotype	file-wide Z range limits for genotype plots
axis.mirror	flag for mirroring the X axis range and Y axis range (intensity plots only)
vertical.line	flag that controls if a light grey, dashed vertical line is drawn through $X = 0$ for calls plots
max.num.SNP.draw	maximum number of SNPs per file (default is 5000)
cex.main	size of the magnification of the main title (default is 1.2)
cex.lab	size of the magnification of the labels (default is 1.2)
num.rows	number of rows in the plots file (default is 4)
num.cols	number of columns in the plots file (default is 6)

Table 4: The arguments for controlling shape, color, and plotting types in *Ps_Visualization*.

col.AA, *col.AB*, *col.BB*, *col.A*, *col.B*, *col.CN0*, *col.OTV*, *col.NC*, *col.OTV1*, and *col.NC1* allow the user to set the colors for each of the standard diploid and haploid calls separately. Note that these arguments only apply to biallelic SNPs; the user cannot change the colors for multiallelic SNPs. The default values are:

- AA: red triangle
- A: light red triangle
- AB: golden circle
- BB: blue inverted triangle
- B: light blue inverted triangle
- NC: grey square
- NC1: light grey square
- OTV: cyan diamond
- OTV1: light aqua diamond
- CN0: white diamond

Colors can be specified either using names, in RGB format, or in hexadecimal format. See **Selecting Colors in R** (4.2.1.1) for more information on how color names are supplied in R.



(a) One SNP plotted with reference genotypes.

(b) Same SNP plotted with different reference genotypes and *refNoCalls* set to **FALSE**.

(c) Same SNP plotted with different reference genotypes and *refNoCalls* set to **TRUE**.

Figure 17: One SNP plotted with reference genotypes, with and without the *refNoCalls* option.

xlim.intensity, *yylim.intensity*, *zlim.intensity*, *xlim.genotype*, *yylim.genotype*, and *zlim.genotype* allow the user to set file-wide X, Y, and Z range limits for intensity and genotype plots. For example, if the user wants all genotype plots to cover the range $(-5, 4.5)$ for the X axis with genotype plots, then *xlim.genotype* should be set to $c(-5, 4.5)$. The range limits must consist of two numbers, where the first number must be smaller than the second number. These options are for the entire set of plots and cannot be used to set range limits for individual plots. Using the range limits on a set of SNPs allows the user to compare multiple SNP plots on the same ranges. The Z range limits are only used in multiallelic plots with 3 axes.

axis.mirror is a logical value that controls whether the X axis range and Y axis range should be the same for intensity plots. The default value is TRUE, which is the same behavior as in previous versions of SNPolisher.

vertical.line is a logical value that controls if a light grey, dashed vertical line is drawn through the point $X = 0$ for genotype plots.

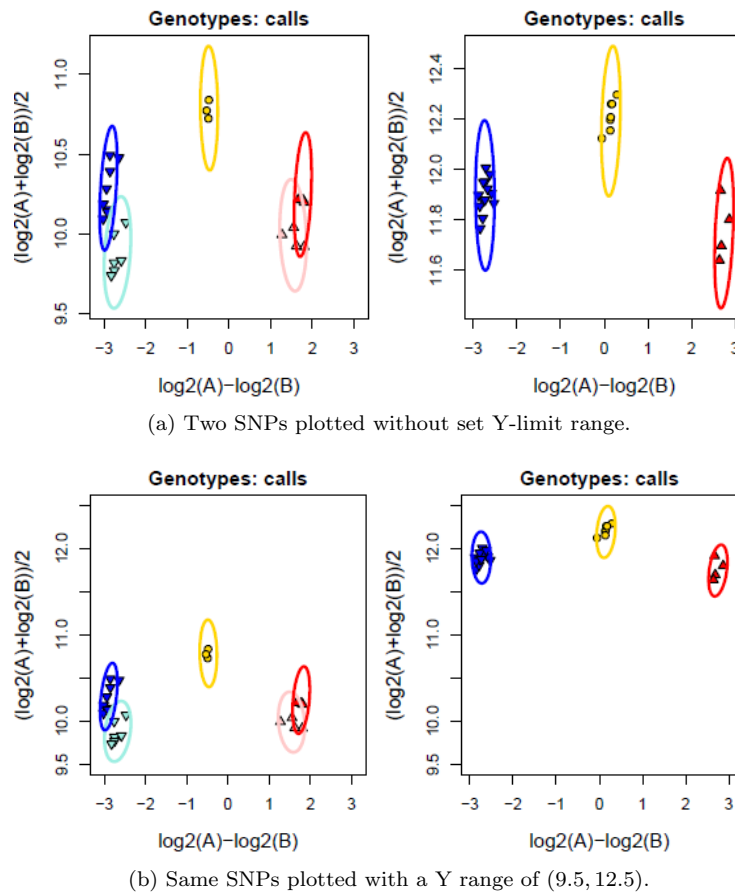


Figure 18: Two SNPs plotted with and without specific Y-limits for the range.

cex.main and *cex.lab* control the size of the magnification of the main title and the labels relative to the size automatically determined by R when setting up plots. The default value is 1.2, i.e. slightly larger than what R sets the font size to be. *num.rows* and *num.cols* control the number of rows and columns in the file when more than one plot is made in a file. The defaults are 4 rows and 6 columns.

Ps_Visualization has been updated with new options for the output files produced.

plot.width and *plot.height* control the size of the output file in inches. The default values are 15.5 inches for width and 12 inches for height. All changes to width and height values must be given in inches.

There are now three different options for output file type: pdf, png, and svg. The *plot.type* option controls what file type is created. The default value is pdf.

The user can also control if all plots are written to one file or if each plot is written to an individual file.

Argument	Description
plot.width	width in inches of the output file (default is 15.5 inches)
plot.height	height in inches of the output file (default is 12 inches)
plot.type	file type of the output file (default is “pdf”)
plot.all	flag that controls if all plots are written to one file or one file per plot is produced (default is TRUE)
square	flag that controls if plots are square when the number of rows or columns is increased (default is FALSE)

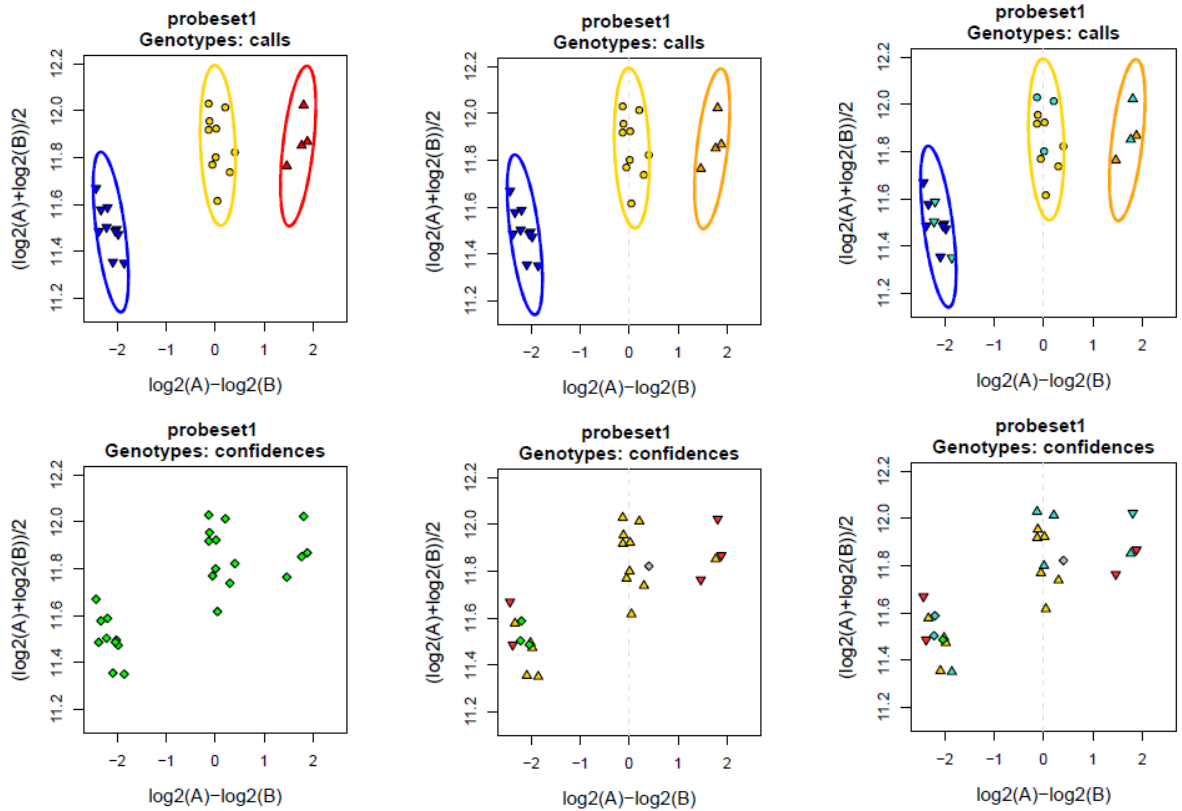
Table 5: The arguments for output files in *Ps_Visualization*.

The *plot.all* option allows users to call *Ps_Visualization* as part of a web app or other type of programs. The default value of TRUE means that all plots are written to one file. When *plot.all* is FALSE, the file names are set to be the probeset name and the plot type (e.g. “AX-1234.calls.pdf”). If more than one batch is plotted, then the batch number is appended (e.g. “AX-1234_batch_2.calls.pdf”). If the probeset is a multiallelic probeset, then the biallelic pairwise combination number is appended (e.g. “AX-1234_batch_2_ma_3.calls.pdf”). If the probeset is a non-PAR X or Y probeset and gender-separated plots are output, the number of the gender-separated plot is appended (e.g. “AX-1234_batch_2_ma_3_gender_1.calls.pdf”). The user cannot change the output names in this version of *Ps_Visualization*. The batch and gender parts of the file name are not used if a probeset is not a special SNP or has more than one batch (e.g. AX-1234_ma_3.calls.pdf).

square is a logical value that allows the user to tell *Ps_Visualization* that the individual plots should be plotted as close to square as possible when the number of rows or columns is increased. The default value is FALSE and should only be set to TRUE if there are fewer than 10 or 12 rows or columns. If there are many rows or columns and *square* is TRUE, the resulting plots will be very small.

The flag *batch* has been deprecated. Note that *Ps_Visualization* automatically detects if multiple batches have been supplied, so this parameter no longer has an effect. See **Batch Plotting** (4.2.1.4) for more information on batch plotting.

Figure 19 displays how using the arguments changes the plots produced by *Ps_Visualization*. Subfigure 19a shows the default color values for genotype calls and confidences plotted for a biallelic SNP. The *labelsFile* has been used to change the main title to “probeset1”. Subfigure 19b shows the same SNP with *col.AA* set to **orange**. *vertical.line* has been set to TRUE. The confidences threshold values have been changed from the defaults of $c(0, 0.01, 0.05, 0.10, 0.15)$ to $c(0, 1e-05, 1.5e-05, 2e-05, 3e-05, 1)$. This displays how well-genotyped probesets will have very small confidences, indicating a high confidence in the genotype call assignment. The default values of *confs.thresh* are set to visually identify samples with problematic genotype calls but these values do not visually separate samples with very small confidence values. Subfigure 19c shows 8 samples highlighted with the color “turquoise”. The *sampleFile* was used to highlight these samples.



(a) One SNP plotted with all default values and a plot label (“probeset1”).

(b) Same SNP plotted with new AA color and updated confidences threshold.

(c) Same SNP plotted with new AA color and updated confidences threshold and highlighted samples.

Figure 19: One SNP plotted with defaults, new colors, and highlighted samples.

Although there are many different options and arguments to use with *Ps_Visualization* and the commands to run *Ps_Visualization* can be long, it is not confusing to create plots when the user knows what the different options do. To produce subfigure 19c, the following command was run:

```
Ps_Visualization(output.File="test.pdf",callFile="AxiomGT1.calls.txt",
summaryFile="AxiomGT1.summary.txt",posteriorFile="AxiomGT1.snp-posteriors.txt",
plot.genotype=TRUE,plot.intensity=FALSE,plot.conf=TRUE,
col.AA="orange",sampleFile="samples.txt",labelsFile="labels.txt",
confs.thresh=c(0,1e-05,1.5e-05,2e-05,3e-05,1),vertical.line=TRUE)
```

4.2.1.1 Selecting Colors in R The encoding for colors in R is quite complicated. Colors can be entered as numeric values, hexadecimal values, nine-digit (RGB = red, green, blue) values, or names. Many R users find it easiest to refer to colors by name. A PDF of the full set of R colors along with their names is available at [24]. To see what names exist in R for the shades of a certain color (e.g. red), use the command:

```
> colors()[grep("red",colors())]
 [1] "darkred"          "indianred"        "indianred1"       "indianred2"
 [5] "indianred3"      "indianred4"       "mediumvioletred"  "orangered"
 [9] "orangered1"      "orangered2"       "orangered3"       "orangered4"
[13] "palevioletred"   "palevioletred1"   "palevioletred2"   "palevioletred3"
[17] "palevioletred4"  "red"              "red1"             "red2"
```

```
[21] "red3"           "red4"           "violetred"      "violetred1"
[25] "violetred2"      "violetred3"     "violetred4"
```

The command `colors()` returns all of the 657 color names. To generate a PDF containing the numeric values, names, hexadecimal, and RGB values for each color, use the command:

```
> source("http://research.stowers-institute.org/efg/R/Color/Chart/ColorChart.R")
```

which will generate a PDF named “ColorChart.pdf” in the current working directory (see figure 20) [25]. This pdf is included in the `SNPolisher` package folder as well.

51	chartreuse4	#458B00
52	chocolate	#D2691E
53	chocolate1	#FF7F24
54	chocolate2	#EE7621
55	chocolate3	#CD661D
56	chocolate4	#8B4513
57	coral	#FF7F50

Figure 20: PDF of R colors with numbers and names.

The website <http://research.stowers-institute.org/efg/R/Color/Chart/> provides more information on selecting colors for scientific graphics, etc. We recommend reading more on color usage before changing the default colors for *Ps_Visualization*.

4.2.1.2 Plotting Multiallelic SNPs Biallelic SNPs have 2 alleles and are adequately represented with 2-dimensional plots, one axis per allele. Multiallelic SNPs have more than 2 alleles and hence should be plotted with more than 2 dimensions to have one axis per allele. `SNPolisher` produces 3D plots for multiallelic SNPs with 2 or more alleles present in the genotype calls. *Ps_Visualization* can also produce 2D plots of all possible biallelic combinations from the SNP’s alleles and one 2D plot of all samples. A new page is started for multiallelic SNPs if the biallelic combination option is selected and there is more than one SNP in the pid list. The axes for multiallelic plots are in \log_2 space and not in MvA (size versus contrast). The user cannot change the transformation or colors for multiallelic plots.

Every multiallelic SNP plot contains a legend detailing the colors and shapes assigned to the genotype calls. These colors and shapes are dynamically assigned for each SNP, and the legend for one multiallelic SNP cannot be used with another multiallelic SNP’s plots. The colors and shapes for multiallelic plots are assigned in the same order for every plot and cannot be changed by the user. NoCalls are still assigned a grey square, and the reserved colors and shapes for AA, AB, BB, A, B, and OTV are the same as in biallelic plots.

One additional plot is created for multiallelic SNPs with 3 or more alleles in the genotype calls. A Principal Components Analysis (PCA) plot is produced which uses the first two principle components as the axes and displays the vectors for the 3 most common alleles. The combination of the intensity, calls, and PCA plots allows the user to visually determine if the genotype clusters are cleanly separated.

Genotype Order	Color	Shape
1st	hot pink (#00FF00)	inverted triangle
2nd	hot green (#FF00FF)	triangle
3rd	dark purple (#9B30FF)	circle
4th	blue (#6495ED)	square
5th	teal (#008B45)	diamond
6th	orange (#FF7F00)	inverted triangle
7th	light green (#3CB371)	diamond
8th	very dark green (#556B2F)	triangle
9th	light orange (#FDAE6B)	triangle
10th	light purple (#DBA5D3)	circle
11th	pink (#EE82EE)	diamond
12th	light tan (#EEE8AA)	inverted triangle
13th	brown (#a63603)	circle

Table 6: The colors and shapes assigned in multiallelic plots.

If there are more than 13 calls and colors and shapes need to be assigned beyond the reserved colors and shapes and the 13 multiallelic colors and shapes, the color and shape assignments are started over from the beginning of the list (i.e. the 14th multiallelic call will be assigned hot pink and inverted triangle). This situation is very rare and is not likely to occur.

There are two arguments that control the options for multiallelic plots beyond those already listed: *multiallele.pairwise* and *multiallele.order*.

Argument	Description
<i>multiallele.pairwise</i>	option controlling if pairwise biallelic combinations should be plotted
<i>multiallele.order</i>	option controlling which in order biallelic combinations should be plotted

Table 7: The arguments for multiallelic plotting in *Ps_Visualization*.

multiallele.pairwise is a logical value that controls if all biallelic pairwise combinations should be plotted or whether one plot with all samples should be plotted. The default value of **FALSE** produces one 3D plot with all samples. When *multiallele.pairwise* is set to **TRUE**, one plot is produced with all samples and then all possible biallelic combinations.

multiallele.order handles the ordering of the biallelic pairwise combinations made from the total set of alleles for multiallelic SNPs. The default value is “decreasing” and the other option is “alphabetical”. Decreasing order means that the pairwise combination with the most samples is plotted first, while alphabetical means that the pairwise combinations are plotted alphabetically (e.g. A vs B, A vs C, A vs D, B vs C, B vs D, C vs D).

If there are two or fewer unique alleles appearing in the calls for a multiallelic SNP, only one biallelic plot is produced. The alleles appearing in the calls are used for the axes, e.g. if the calls are CD, DD, and CC, the two axes are C and D (for intensity plots) or $\log_2(C)$ and $\log_2(D)$ (for calls plots). If there is only one allele, the plot uses A for the first allele and the one allele for the second allele, e.g. if the calls are DD then the two axes are A and D (for intensity plots) or $\log_2(A)$ and $\log_2(D)$ (for calls plots). If the only alleles in the calls are A, then the second allele used in the plots is B. The transformation is always \log_2 , regardless of how many alleles appear.

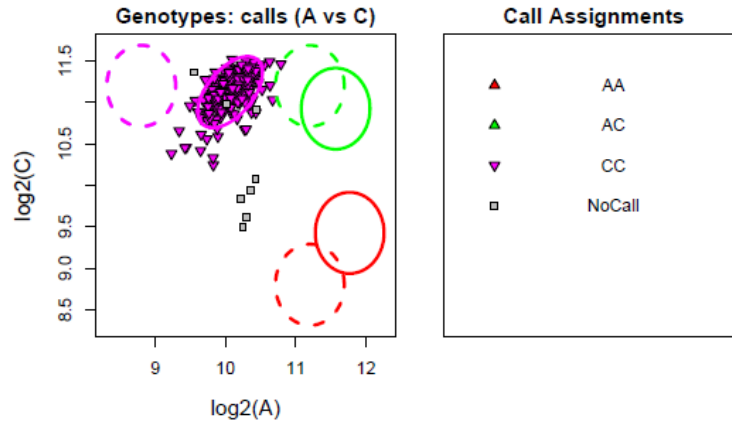


Figure 21: Plot of a multiallelic SNP with one unique allele.

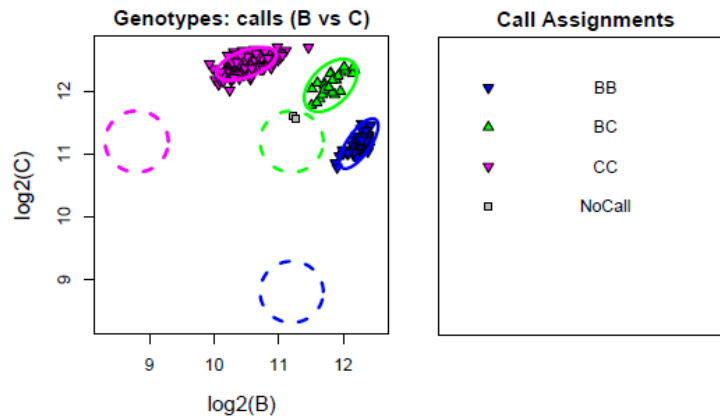


Figure 22: Plot of a multiallelic SNP with two unique alleles.

Multiallelic probesets with 3 or more unique alleles have one additional plot, the PCA plot. The intensity and calls plots have 3 axes, even when there are more than 3 alleles in the calls. The first (X) axis is the allele that occurs most often in the genotype calls, the second (Y) axis is the allele that occurs second most often in the genotype calls, and the third (Z) axis is the allele that occurs third most often in the genotype calls. While the X and Y axes are plotted the same as in a 2D plot (“going across” and “going up”), the Z axis is plotted “going backwards” through the page. As the size of the Z or $\log_2(Z)$ value increases, the shape representing a sample moves “backwards” in the Z space.

The PCA plot displays the samples plotted in the space of the first two principal components (PC1 and PC2), which are the components that account for the largest and second largest amount of variance in the data. Vectors representing the same three alleles that are the axes of the intensity and calls plots appear in the PCA plot. The combination of the intensity, calls, and PCA plots give the user a clear view of how well the clusters are separated from each other, both in location and in variation.

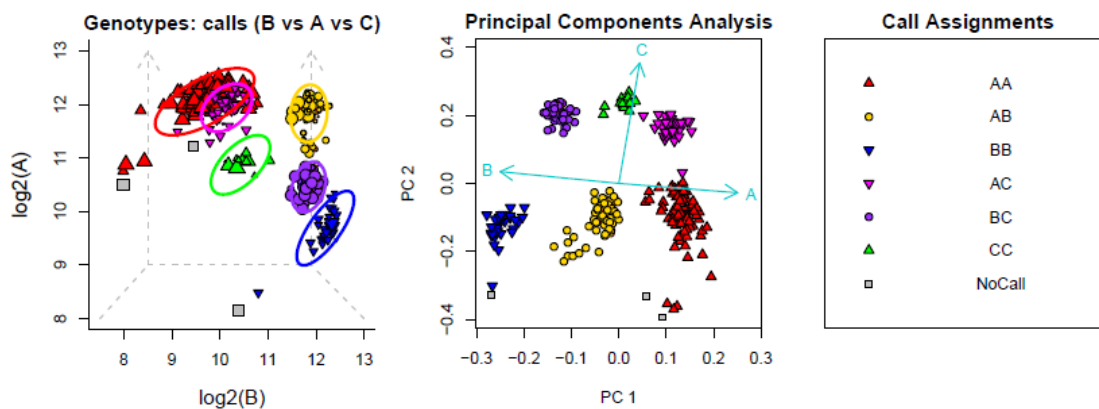


Figure 23: Plot of a multiallelic SNP with three unique alleles.

Multiallelic plotting has an additional plotting option that is not available for biallelic plotting: pairwise plots. In this option, all possible biallelic pairs are created from the three or more alleles that appear in the calls of a multiallelic probeset, and plots are produced for every biallelic pairwise combination. This allows the user to investigate how well the samples are clustered when looking only at the clusters with the two alleles present. PCA plots are not produced with the pairwise plotting option.

The first plot made for any multiallelic probeset contains all of the samples, and the two alleles used in the axes are always A and B. For an intensity plot, the axes are A and B. For a calls or confidences plot, the axes are $\log_2(A)$ and $\log_2(B)$. This is the only plot made for pairwise plots that contain all of the samples. The rest of the plots are of the biallelic pairwise combinations that have at least one biallelic calls assignment. If a SNP has calls AA, BB, AC, BC, and CC, then A versus B will be plotted and samples with AA and BB calls will appear in this plot, A versus C will be plotted and samples with AA, AC, and CC calls will appear in this plot, and B versus C will be plotted and samples with BB, BC, and CC calls will appear in this plot.

Figure 24 shows the same probeset that appears in figure 23, plotted with biallelic pairwise combinations.

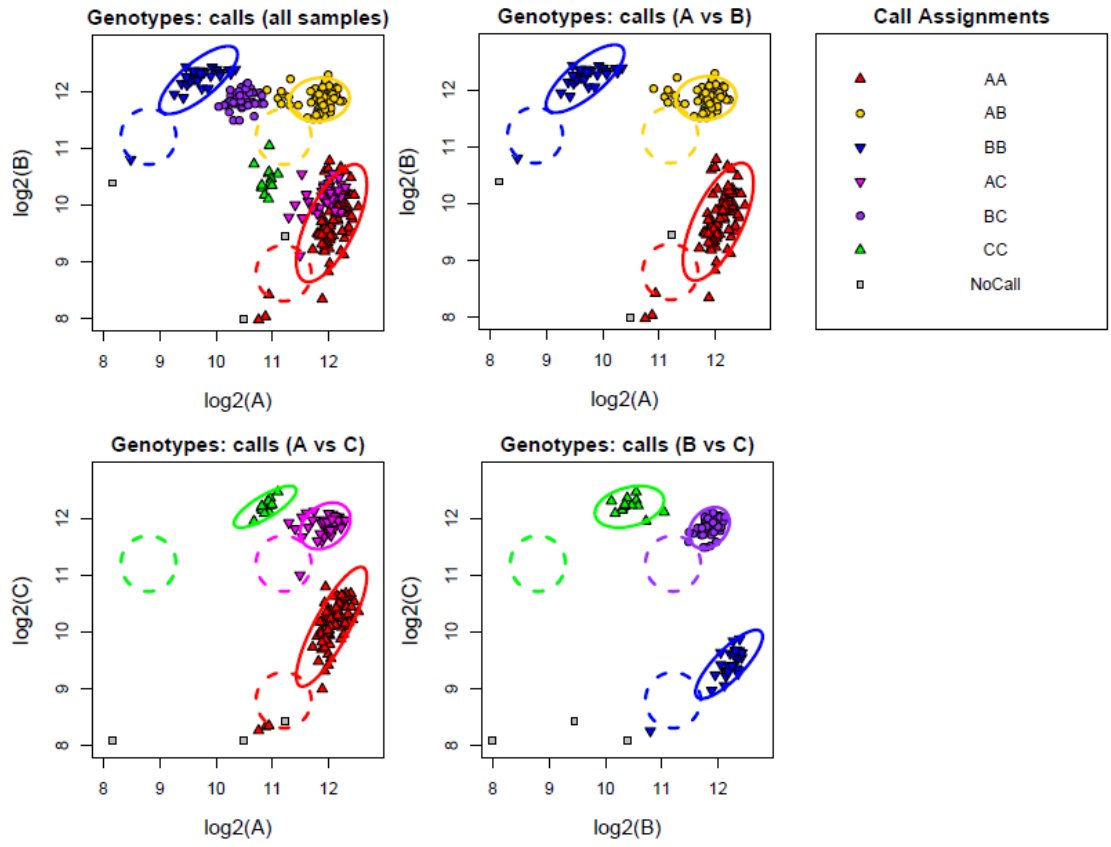


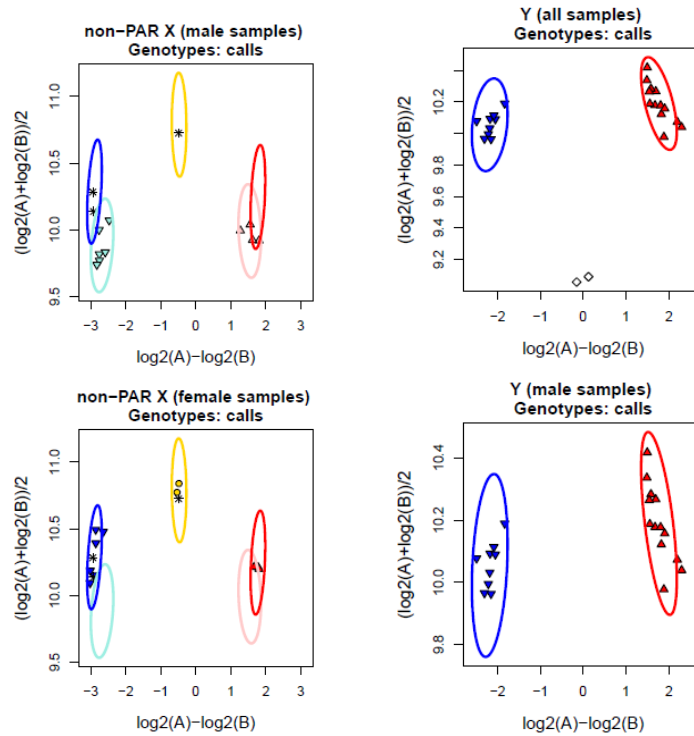
Figure 24: Pairwise plots of a multiallelic SNP with three unique alleles.

4.2.1.3 Gender-Separated Plots When a special SNPs file and a report file are provided, *Ps_Visualization* will produce gender-separated plots for non-PAR X, Y, Z (avian species only), and W (avian species only) SNPs. Non-PAR X and Z SNPs have two plots, one with only male samples and one with only female samples. Y and W SNPs have two plots, one with only male (Y) or female (W) samples and one with all samples. Samples with unknown gender are plotted as a black asterisk and appear in all plots. If there are a lot of samples with unknown gender, the option *unknownGender* can be set to **FALSE** and samples with unknown genders will not be included in the plot.

Plotting the genders separately can make it easier to visually inspect a SNP. For a non-PAR X SNP, the female samples can be diploid, haploid, or ZeroCN. The male samples should be haploid. This means that the male samples are expected to be clustered below the diploid female samples, and should be closer to $X = 0$ in the transformed data space. For a Z SNP, the male samples can be diploid, haploid, or ZeroCN. The female samples should be haploid. The female samples are expected to be clustered below the the diploid male samples, and should be closer to $X = 0$ in the transformed data space.

For a Y SNP, the male samples should be haploid and the female samples should be ZeroCN or NC. This means that the female samples are expected to be clustered below the male samples and to sit approximately at $X = 0$ in the transformed data space. For a W SNP, the female samples should be haploid and the male samples should be ZeroCN or NC. The male samples are expected to be clustered below the female samples and to sit approximately at $X = 0$ in the transformed data space.

Mitochondrial (MT) and chloroplast (CP) SNPs should have all haploid samples and there is not different behavior between genders. MT and CP SNPs are plotted in a single plot and do not need the special SNPs file or the report file.



(a) Plot of a non-PAR X SNP: the male samples plot shows that the male samples are all haploid as expected and there are several diploid samples with unknown gender, and the female samples plot shows that the females are above the males.

(b) Plot of a Y SNP: the plot with all samples shows that the female samples are below the males and centered at 0, and the male samples plot shows that the male samples are well clustered.

Figure 25: Plots of a non-PAR X SNP and a Y SNP.

4.2.1.4 Batch Plotting When there is a large number of samples, genotyping may be split into batches. This will result in multiple calls, summary, and posteriors files being produced for the same SNPs with different samples in each batch. Batch plotting can be used to investigate the behavior of samples across batches for a SNP, and visual inspection may be used to detect problems with a batch such as cluster flips or splits or unexpected allele frequencies. *Ps_Visualization* produces multiple batch plots per SNP when batch files are supplied as input.

Because batches are based on samples and have the same SNPs, files that only have SNP information do not need to be supplied multiple times: the special SNPs file and the pid file should only be supplied once. The summary, calls, and confidences files have information for the samples and there must be one summary file per batch and either one calls or one confidences file per batch. The priors and posteriors files do not need to be supplied for each batch; if a batch is missing a posteriors or priors file when *plot.posteriors* or *plot.priors* is TRUE, then *Ps_Visualization* will use default values. The value NULL should be supplied for any batch that does not have a posterior or priors file. The length of the values in *posteriorFile*, *multiallele.posteriorFile*, *priorFile*, and *multiallele.priorFile* must be the same length as the total number of batches. The reference file has the same behavior as the posteriors and priors file: any batch without a reference file should use a value of NULL instead.

If a special SNPs file is provided, a report file must be provided for every batch. If more than one special SNPs file is provided, *SNPolisher* will output an error message.

One sample file per batch may be supplied to highlight samples that appear in specific batches. If there are no samples to highlight in a batch, a value of NULL may be provided for that batch instead.

When plotting batches, *Ps_Visualization* automatically appends the batch number to the main plot title with the phrase “batch 1”, “batch 2”, etc. If a labels file is provided, that main title is used. To change the main plot title within a batch, a labels file should be supplied for each batch. If a batch does not have any titles to change, then that batch accepts the value `NULL` instead of a file name. The length of the values in the *labelsFile* object must be the same length as the total number of batches.

Most other arguments in *Ps_Visualization* are not batch-dependent and are applied across all batches.

Ps_Visualization handle the layout of batch plots automatically. A new page is started in a file at the beginning of each set of batch plots for every SNP. For biallelic SNPs, all plots are produced per batch and then the next batch is started. For example, if a user is plotting batches and requests intensity, genotype, and reference plots, then *Ps_Visualization* will produce an intensity plot, an intensity reference plot, a genotype plot, and a genotype reference plot before starting to plot the next batch. This ensures that the same plots for each batch are plotted on the same row for easy comparison.

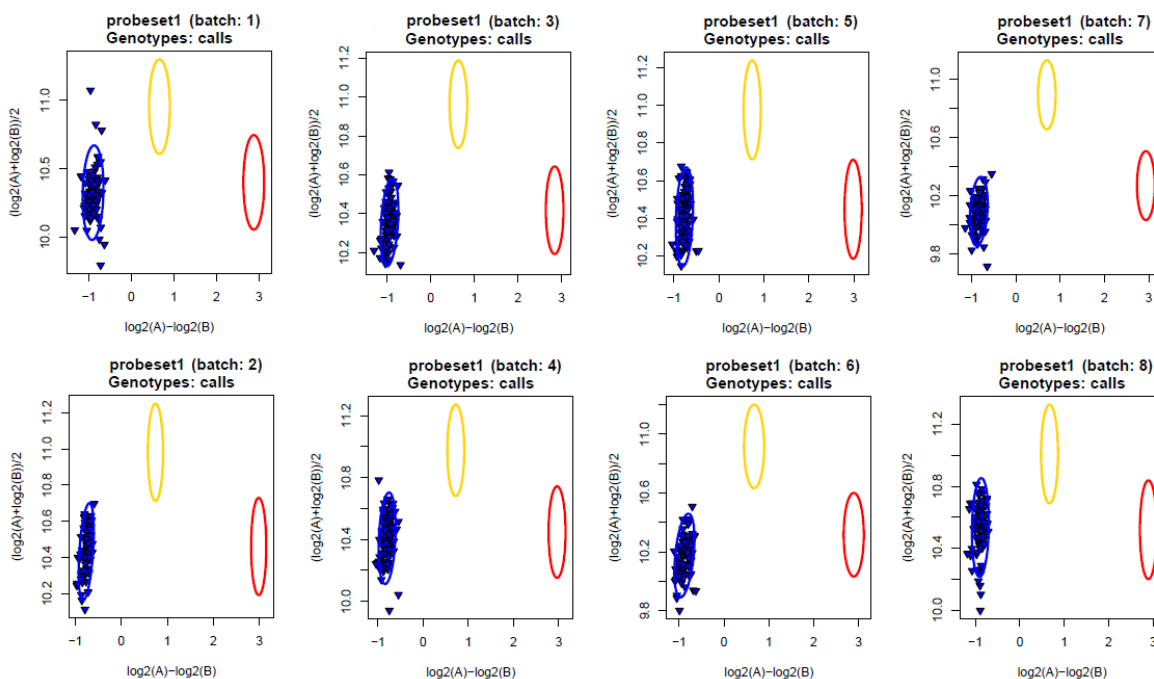


Figure 26: Batch plots for a biallelic SNP.

Batch plotting for multiallelic SNPs is similar to batch plotting for biallelic SNPs. Each type of plot is made per batch and then the next batch is plotted. Each batch is started in a new column, and each SNP is started on a new page if all plots are written to the same file. Adjusting the number of rows (*num.rows*) or columns (*num.cols*) can be helpful when plotting multiple batches. Multiallelic SNPs with six or more batches are likely to have multiple pages of plots.

Because multiallelic SNPs have dynamically assigned call codes, it is possible but unlikely that SNPs genotyped in separate batches may end up with different call codes for the same genotypes. The user may also want to plot samples that were genotyped at different times, or that are subsets of other batches that were genotyped together. *Ps_Visualization* automatically handles samples with the same genotype calls but different assigned call codes, and all genotype calls across all batches have the same color and shape assignments regardless of the assigned call codes. The legend displays the call assignments.

Figure 27 displays the intensity, calls, and PCA plots for a multiallelic SNP that has 5 batches. *num.rows* and *num.cols* did not need to be adjusted because the default values for rows (4) and columns (6) were enough for the 5 batches plus legend with 3 plots per batch. The legend displays the assigned colors and shapes for every call that appears in the batch plots. This set of batch plots displays consistency in genotypes across all 5 batches. While each batch is slightly different from each other, the sample clustering and genotype assignments are very similar. All batches show a well-genotyped SNP.

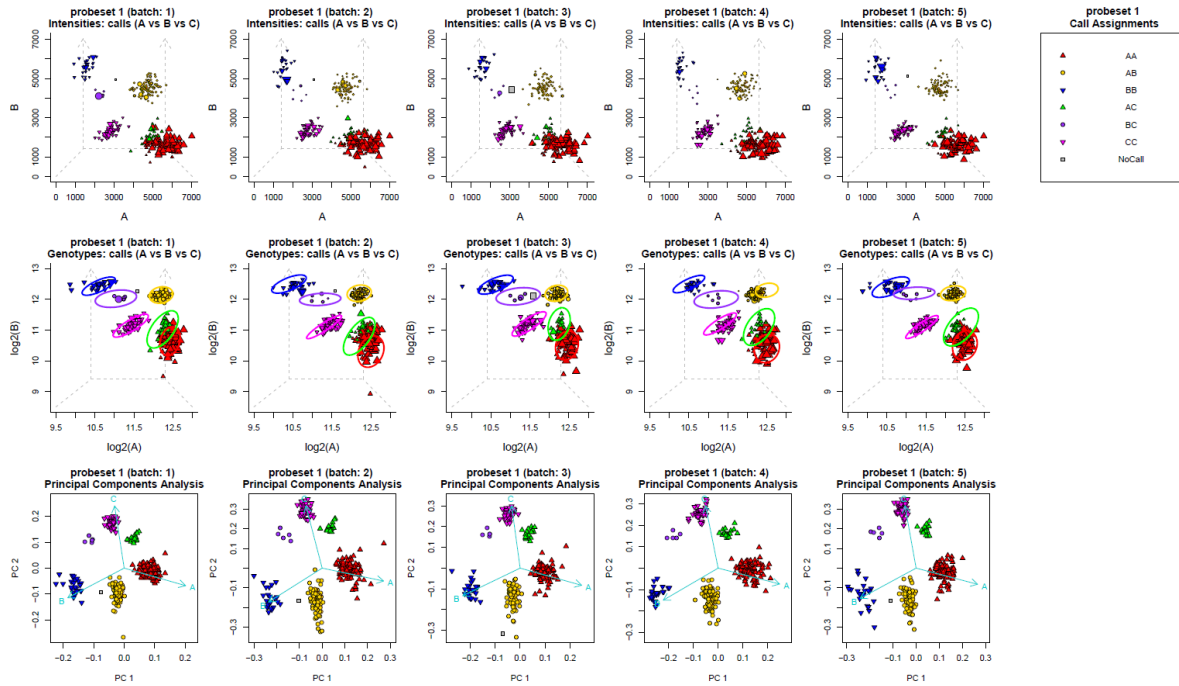


Figure 27: Batch plots for a multiallelic SNP.

Batch plotting can be used to easily identify if there are any batches with genotyping results that are considerably different from the majority of batches. Figure 28 displays the intensity, calls, and PCA plots for a multiallelic SNP that has 4 batches. Batch 2 shows considerably different cluster patterns and call assignments from batches 1, 3, and 4, and the PCA plot for batch 2 displays a very different pattern. Batch plotting lets the user perform a visual analysis across multiple batches to easily determine the consistency across batches.

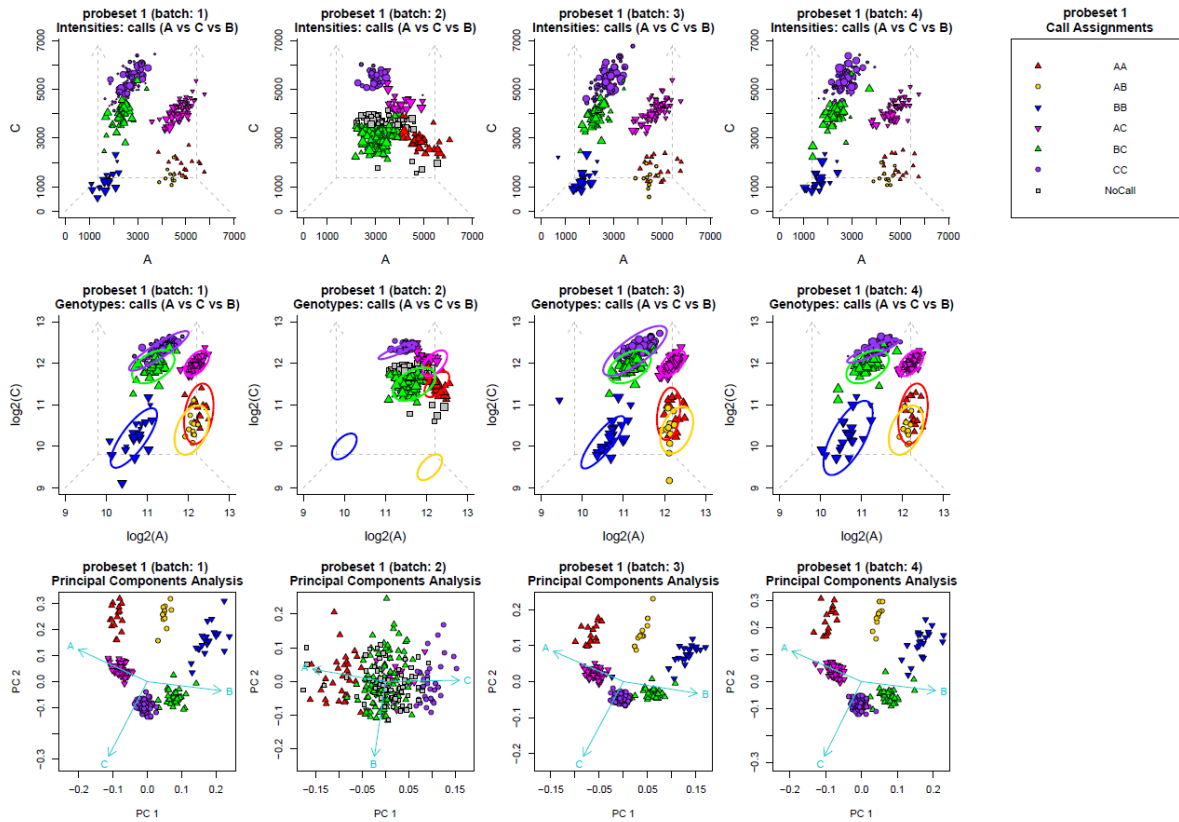


Figure 28: Batch plots for another multiallelic SNP.

Pairwise plotting may also be used in combination with batch plotting, and can be helpful when looking at the relationship between several specific clusters rather than all clusters. The layout for pairwise batch plotting is different from regular multiallelic batch plotting. The first set of plots run through all batches and shows all samples and is plotted in $\log_2(A)$ versus $\log_2(B)$ space. The rest of the plots are of all possible biallelic combinations that appear in the calls across all of the batches. Each biallelic combination is plotted for all batches before moving on to the next biallelic combination. This allows the user to compare the same biallelic combinations across all batches at the same time. When all batches have been plotted for a biallelic combination, new column is started. PCA plots are not produced with the pairwise plotting option.

If there are fewer than 10 batches, we recommended adjusting the *num.cols* option to be the total number of batches, so that all plots for a biallelic combination will appear in one column. There is always an extra column, with the legend plot. It is not unusual to adjust the *num.rows* and *num.cols* options when plotting with the pairwise and batch options.

Figure 29 shows the same SNP from figure 27 with pairwise batch plotting. The *num.cols* option was set to 5 because there are 5 batches. Intensity plots were not produced (*plot.intensity=F*) to fit all of the plots on one page. If the intensity plots had been made, there would have been 9 columns instead of 5. Again, this probe set shows consistent behavior across all batches.

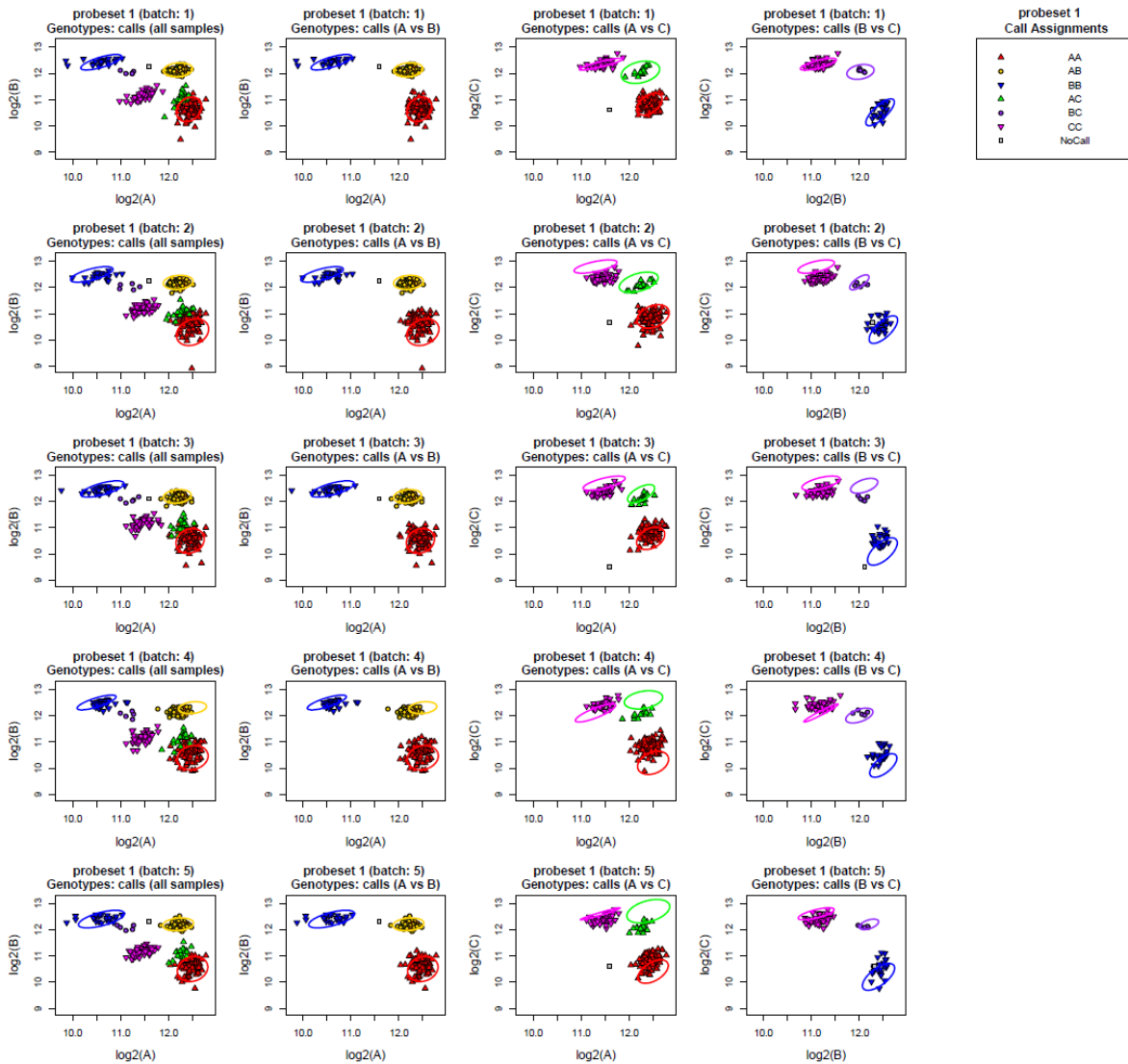


Figure 29: Pairwise batch plots for a multiallelic SNP.

Figure 30 shows the same SNP from figure 28 with pairwise batch plotting. The plots of the biallelic combinations make it clear that there are major differences in the A versus B and B versus C plots for batch 2 compared with batches 1, 3, and 4. The *num.rows* and *num.cols* options were not adjusted because all plots fit onto one page. Again, intensity plots were not produced. Note that the ranges of the plots are larger than expected because of the outlier in batch 3. The ranges of all plots are same across the batches so that the locations of the batches can be compared as well.

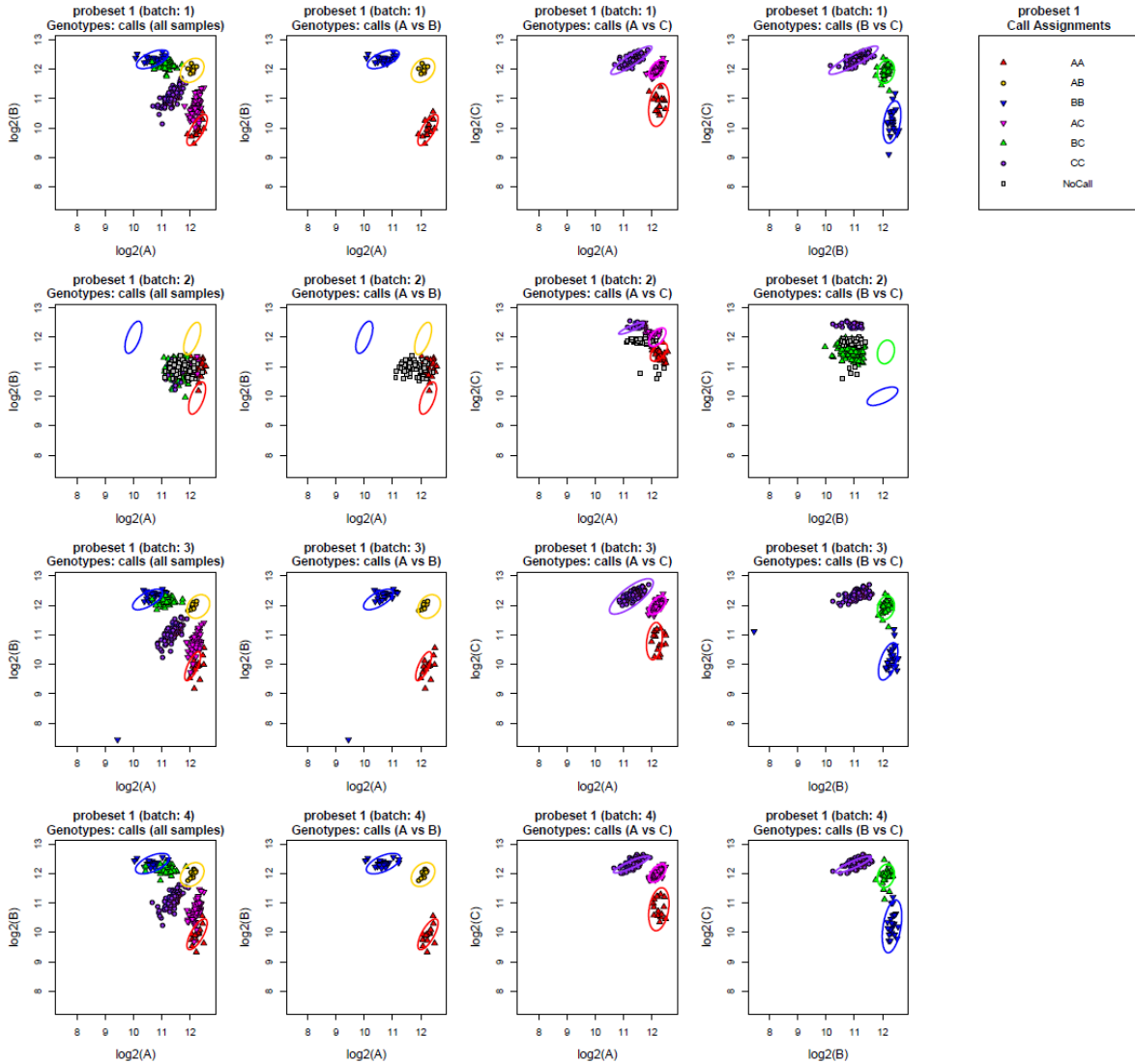


Figure 30: Pairwise batch plots for another multiallelic SNP.

4.2.2 Outputs

Ps_Visualization either produces one file containing all plots or one file per plot. The file types are PDF, PNG, or SVG. Note that PDF files are considerably larger in file size than the other two types of files, and may not always display the plots the exactly the same way across different operating systems. The user controls which directory the output files are written to so that if multiple output files are requested, all files will be written to the same directory.

The default output is to write all plots to the same file, with the default configuration of six columns per page (one SNP per column) and four rows per page (two plots per SNP: intensity calls and genotype calls). Because the summary, calls, confidences, and posteriors data must be extracted from larger files for plotting, we recommend plotting a smaller number of SNPs as it can be time consuming to generate cluster plots for thousands of SNPs (default is 5,000 SNPs maximum). To decrease plotting times, the function *Ps_Extract* may be used to write smaller files with the data for SNPs. If the SNPs have been plotted previously and the temporary directory has not been deleted, the user may supply this directory to *Ps_Visualization* and the step of extracting the data will be skipped; the smaller files in the temporary directory will be used instead. All SNPs being plotted must have a smaller file in the temporary directory. *Ps_Visualization* will ignore any files in the temporary directory that are not for the SNPs being plotted, so if a user has plotted a larger number of SNPs and wishes to make a file with only a subset of those SNPs, the same temporary directory can be used.

We recommend routinely viewing about 200 random probesets from each of the SNP categories produced by *ps-classification*.

4.3 Ps_Vis_Density

Ps_Vis_Density is a variant of *Ps_Visualization*. *Ps_Vis_Density* generates density cluster plots to visualize the distribution of samples within each cluster. Genotype call plots and reference plots may also be created to accompany the density plots. Shape and color assignments for genotype plots are handled identically to *Ps_Visualization*.

Ps_Vis_Density contains many of the same options as *Ps_Visualization*: biallelic and multiallele plotting, batch plotting, sample highlighting, and gender-separated plotting. *Ps_Vis_Density* does not make intensity plots or confidences plots. Because *Ps_Vis_Density* is a companion function to *Ps_Visualization*, we have assumed that the user is familiar with *Ps_Visualization*. Please read the in-depth description of the parameters and options in section 4.2 to get a complete description of all plotting options.

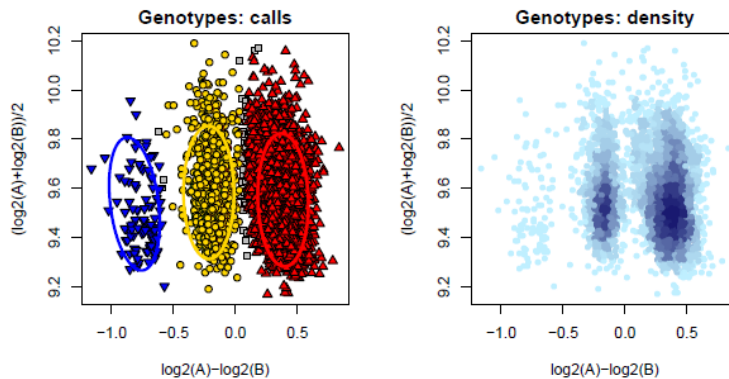


Figure 31: Density and calls plots for a biallelic SNP.

4.3.1 Inputs and Arguments

Ps_Vis_Density takes fewer possible input files than *Ps_Visualization*, and has only two required files: the pid file (*pidFile*) and the summary file (*summaryFile*). These files provide *Ps_Vis_Density* with a list of probesets to plot, the locations of each sample that was genotyped for each probeset, and data that is used to calculate the sample densities. All other input files and parameters are optional. If calls plots are requested, *Ps_Vis_Density* handles shape and colors assignments identically to *Ps_Visualization*.

`Ps_Vis_Density(pidFile,summaryFile)`

- *pidFile*: text file listing probeset IDs
- *summaryFile*: Axiom summary file (usually “AxiomGT1.summary.txt”)

The optional input files for *Ps_Vis_Density* are used for plotting calls, reference genotypes, posteriors, priors, and assigning labels for plot titles and highlighting samples. They are the same options that are listed in table 2, with the addition of the *callFile* argument, which is optional in *Ps_Vis_Density*. All of the optional inputs in table 8 behave exactly the same as in *Ps_Visualization*.

Argument	Description
callFile	Axiom calls file (usually “AxiomGT1.calls.txt”)
posteriorFile	Axiom biallelic posteriors file (usually “AxiomGT1.snp–posteriors.txt”)
multiallele.posteriorFile	Axiom multiallelic posteriors File (usually “AxiomGT1.snp–posteriors.multi.txt”)
priorFile	Axiom biallelic priors file (usually “AxiomGT1.priors.txt”)
multiallele.priorFile	Axiom multiallelic priors file (usually “AxiomGT1.priors.multi.txt”)
sampleFile	file listing the samples to be highlighted and the sample color
labelsFile	file listing the labels to be used as main plot titles instead of SNP names
specialSNPsFile	file listing the special SNPs (report file must be supplied with the special SNPs file)
reportFile	file listing the genders of the samples (special SNPs file must be supplied with the report file)
refFile	file with the reference genotype calls
sampleLookupFile	file matching up the same samples in the calls and references files that have different names
presenceAbsenceFile	file listing the presence and background allele of the subset of probesets that are plotted with the PAV transformation
temp.dir	name of the temporary directory created by <i>Ps_Visualization</i>
keep.temp.dir	flag to keep or remove the temporary directory
use.temp.dir	flag to use an already existing temporary directory

Table 8: The optional input file arguments for *Ps_Vis_Density*.

Ps_Vis_Density does not plot confidences or intensities. Density plots, genotype call plots, and genotype reference plots are the only plotting options. Any options not listed are identical to *Ps_Visualization*. Options for intensity plotting (e.g. *xlim.intensity*) do not appear at all in *Ps_Vis_Density* and will not have any effect.

Argument	Description
plot.calls	flag indicating if genotype calls should be used to assign shapes and colors in plots
plot.density	flag indicating if density plots should be made
density.levels	number of bins that sample density should be split into (default is 20)
density.col	colors assigned to samples in each density bin (default is "blue")
plotNoCalls	flag indicating if NoCalls should be plotted with genotype calls
plot.pca	flag indicating if the PCA plots should be plotted for multiallelic probesets
plot.ref	flag indicating if reference calls should be used to assign shapes and colors in plots
refNoCalls	flag indicating if NoCalls should be plotted with reference calls
plot.prior	flag indicating if prior ellipses should be plotted
plot.prior.ref	flag indicating if prior ellipses should be plotted on reference plots
plot.posterior	flag indicating if posteriors ellipses should be plotted
plot.posterior.ref	flag indicating if posterior ellipses should be plotted on reference plots
unknownGender	flag indicating if samples with unknown gender should be included in gender-separated plots
transform	which data transformation should be used when plotting genotype plots
K	variable used with several transformations
autotetraploid	flag indicating if the data is from an autotetraploid species
axis.mirror	flag for mirroring the X axis range and Y axis range (intensity plots only)
vertical.line	flag that controls if a light grey, dashed vertical line is drawn through $X = 0$ for calls plots

Table 9: The arguments for controlling plotting types in *Ps_Vis_Density*.

plot.density, *density.levels*, and *density.col* are the arguments that are found only in *Ps_Vis_Density*.

plot.density is a logical value which handles whether density plots are made. Although this value is allowed to be set to **FALSE**, this is the only plotting option which is found only in *Ps_Vis_Density*. If density plots are not requested, then *Ps_Visualization* should be used instead of *Ps_Vis_Density*.

density.levels is the number of bins that are used in calculating the sample densities. This value must be an integer between 3 and 20.

density.col can be either a one-word name of a color (red, yellow, orange, green, purple, brown, black, grey/gray, or blue) that produces density colors in those shades or it may be a set of colors in RGB or hexadecimal notation. If more than one color is supplied in *density.col*, then the number of colors must be the same as the number of bins in *density.levels*. This option can be used to set the density colors to be multiple colors instead of shades of one color. See section 4.2.1.1 for more details on selecting colors in R. The default value is "blue".

The other arguments to *Ps_Vis_Density* can be found in tables 4 and 5.

The genotype calls and reference plots from *Ps_Vis_Density* are the same as those produced by *Ps_Visualization*. The density plots only use a circle as the shape that is plotted for all samples (**pch=20**). The user cannot change this shape. If both calls and density plots are requested, then the plots have the same X and Y (and Z) limits. When all three plotting options are requested, genotype calls are plotted first, then reference plots, and then density plots. If reference plots are not requested, calls plots are made before density plots.

Gender-separated plots, batch plots, and multiallelic plots are very similar in *Ps_Vis_Density* to *Ps_Visualization*. If all three plotting options are requested, then each batch is plotted in a separate column in order to make comparison across batches easier. Gender-separated plots have two plots per plotting option, for a total of six plots if all three plotting options are requested. The only difference between multiallelic plotting in *Ps_Visualization* and *Ps_Vis_Density* is that multiallelic probesets do not have a legend supplied when only density plots are made because there are no shape and color assignments based on call codes.

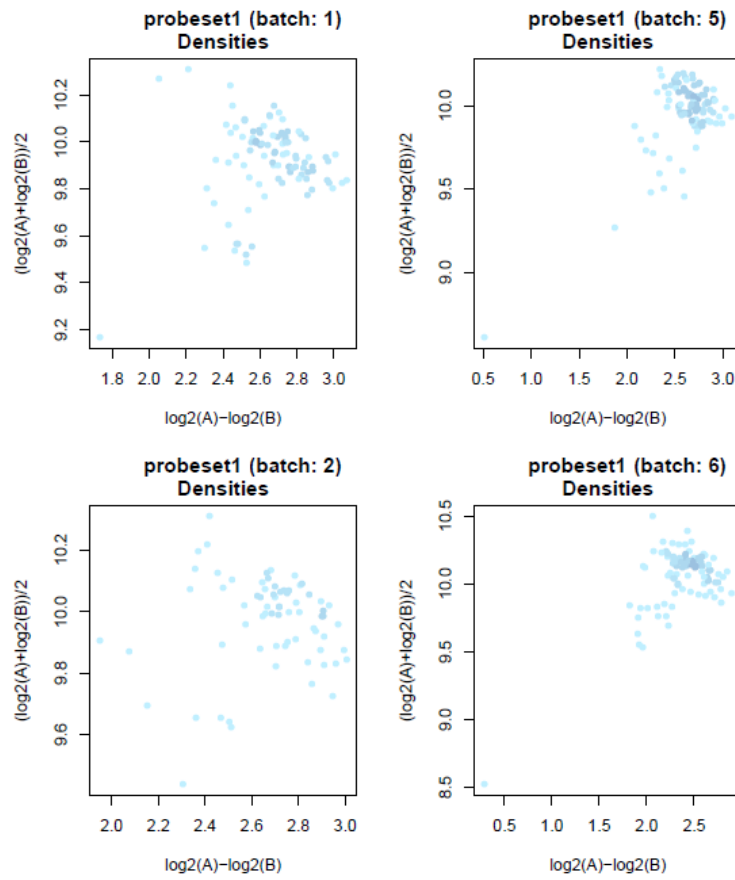


Figure 32: Density batch plots for a biallelic SNP.

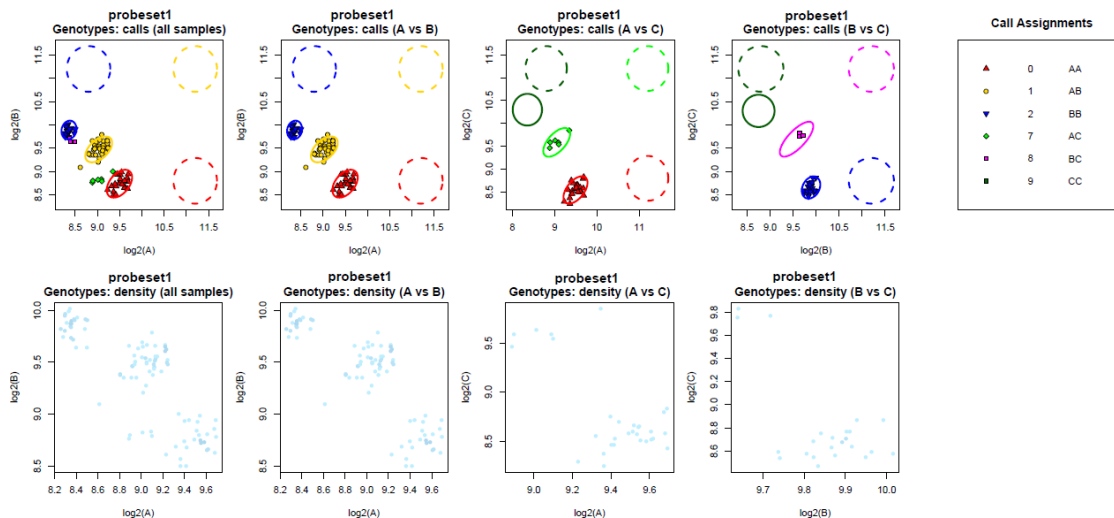


Figure 33: Pairwise calls and density plots for a multiallelic SNP.

Ps_Vis_Density and *Ps_Visualization* both use the same temporary files in the temporary directory to make plots. As a result, any SNPs that have previously been plotted in either *Ps_Visualization* or *Ps_Vis_Density* can be used with *Ps_Vis_Density*. The *use.temp.dir* flag tells *Ps_Vis_Density* to use an already extant temporary directory as long as that directory contains temporary files for all of the SNPs that will get density plots (default is `FALSE`).

4.3.1.1 Density Algorithm Density values are assigned using a simple 2D (or 3D) binning algorithm. A fine two-dimensional grid is laid over the plotting space, and the number of samples that occur within each grid square is counted. The grid counts are sorted by size and the percentiles that correspond to the number of bins are calculated. If *density.levels* is set to 10, then the percentiles are 0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, and 0.9. Grid squares with sample counts in the range between 0 and the 10th percentile (0.1) are assigned to the first bin. These samples will be colored with the first and lightest color. Grid squares with sample counts in the range between the 10th percentile (0.1) and the 20th percentile (0.2) are assigned to the second bin. These samples will be colored with the second color. This binning continues until all samples have been assigned a density color.

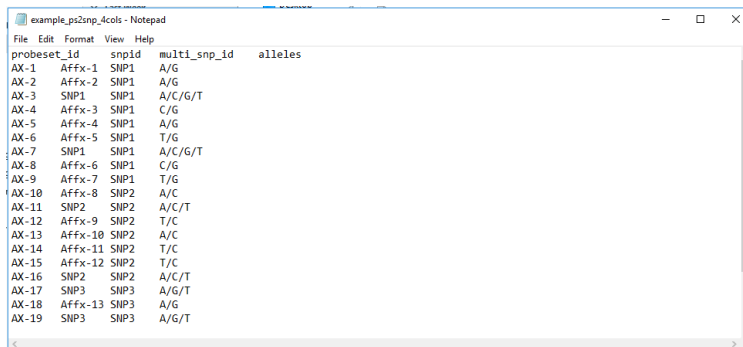
4.3.2 Outputs

Ps_Vis_Density has the exact same output options as *Ps_Visualization*. *Ps_Vis_Density* either makes one file with all plots or one file per plot. The *plot.all* option controls how many files are output. The file type is controlled by *plot.type*. PDF files are generally much larger than PNG or SVG files and do not always display exactly the same way across different operating systems. The directory that output files are written to is controlled by *output.dir*.

4.4 Ps_Vis_Multi_ID

Ps_Vis_Multi_ID is another variant of *Ps_Visualization*. *Ps_Vis_Multi_ID* generates cluster plots of all probe-sets used to produce multiallelic genotype calls for a multiallelic SNP ID. Multiallelic genotype calls are assigned by compiling genotype information from a set of biallelic and multiallelic probesets. *Ps_Vis_Multi_ID* allows the user to visually inspect and compare the set of underlying probesets for a multiallelic SNP ID.

Ps_Vis_Multi_ID contains many of the same options as *Ps_Visualization*: biallelic and multiallele plotting, batch plotting, sample highlighting, and gender-separated plotting. *Ps_Vis_Density* makes intensity, calls, and reference plots. It does not make confidences plots. Because *Ps_Vis_Multi_ID* is a companion function to *Ps_Visualization*, we have assumed that the user is familiar with *Ps_Visualization*. Please read the in-depth description of the parameters and options in section 4.2 to get a complete description of all plotting options.



```
example_ps2snp_4cols - Notepad
File Edit Format View Help
probeset_id snpid multi_snp_id alleles
AX-1 Affx-1 SNP1 A/G
AX-2 Affx-2 SNP1 A/G
AX-3 SNP1 SNP1 A/C/G/T
AX-4 Affx-3 SNP1 C/G
AX-5 Affx-4 SNP1 A/G
AX-6 Affx-5 SNP1 T/G
AX-7 SNP1 SNP1 A/C/G/T
AX-8 Affx-6 SNP1 C/G
AX-9 Affx-7 SNP1 T/G
AX-10 Affx-8 SNP2 A/C
AX-11 SNP2 SNP2 A/C/T
AX-12 Affx-9 SNP2 T/C
AX-13 Affx-10 SNP2 A/C
AX-14 Affx-11 SNP2 T/C
AX-15 Affx-12 SNP2 T/C
AX-16 SNP2 SNP2 A/C/T
AX-17 SNP3 SNP3 A/G/T
AX-18 Affx-13 SNP3 A/G
AX-19 SNP3 SNP3 A/G/T
```

Figure 34: An example of a 4-column ps2snp file.

4.4.1 Inputs and Arguments

Ps_Vis_Multi_ID takes fewer possible input files than *Ps_Visualization*, and has only two required files: the 4-column ps2snp file (*ps2snpFile*) and the summary file (*summaryFile*). These files provide *Ps_Vis_Multi_ID* with a list of SNPs to plot, the locations of each sample that was genotyped for each probeset, and the set of probesets that make up a SNP. All other input files and parameters are optional. If calls plots are requested, *Ps_Vis_Multi_ID* handles shape and colors assignments identically to *Ps_Visualization*.

`Ps_Vis_Multi_ID(snpFile,summaryFile)`

- *ps2snpFile*: text file listing SNP and probeset IDs
- *summaryFile*: Axiom summary file (usually “AxiomGT1.summary.txt”)

The 4-column ps2snp file matches up a probeset ID, SNP ID, and multiallelic SNP ID. *Ps_Vis_Multi_ID* plots all probesets together that contribute to the genotype call assignments for a multiallelic SNP ID. The main title contains the name of the multiallelic SNP ID and each probeset ID. The ps2snp file is a library file that is distributed to users, with one ps2snp file per array type. Arrays with multiallelic SNPs can have a ps2snp file with 4 columns. Arrays with only biallelic SNPs will have a ps2snp file with only 2 columns. The four column ps2snp files have columns named “probeset_id”, “snpid”, “multi_snp_id”, and “alleles”. If the second column is named “affx_snp_id”, *Ps_Vis_Multi_ID* will stop with an error and ask for the column to be renamed. The “probeset_id” and “snpid” columns are the same regardless of whether the ps2snp file has 2 or 4 columns. The “multi_snp_id” columns lists the multiallelic SNP ID that the probesets contribute call information to, and the “alleles” column lists the set of alleles that each probeset has calls for in the multiallelic SNP.

The ps2snp library file that comes with an array contains all probesets on the entire array. The user should create a smaller version of the 4-column ps2snp file to use as input to *Ps_Vis_Multi_ID*. Probesets are plotted in the order that appear in the “probeset_id” column for a multiallelic SNP, and are not sorted on the “snpid” column. The user should sort probesets in the 4-column ps2snp file into the order they want the probesets to be plotted for a multiallelic SNP.

The color and shape assignments are coordinated across the various probesets and are the same for all probesets belonging to a multiallelic SNP ID.

The optional input files for *Ps_Vis_Multi_ID* are used for plotting calls, reference genotypes, posteriors, priors, and assigning labels for plot titles and highlighting samples. They are the same options that are listed in table 2. All of the optional inputs in table 10 behave exactly the same as in *Ps_Visualization* with the exception of the labels file.

Argument	Description
callFile	Axiom calls file (usually "AxiomGT1.calls.txt")
posteriorFile	Axiom biallelic posteriors file (usually "AxiomGT1.snp-posteriors.txt")
multiallele.posteriorFile	Axiom multiallelic posteriors File (usually "AxiomGT1.snp-posteriors.multi.txt"))
priorFile	Axiom biallelic priors file (usually "AxiomGT1.priors.txt")
multiallele.priorFile	Axiom multiallelic priors file (usually "AxiomGT1.priors.multi.txt")
sampleFile	file listing the samples to be highlighted and the sample color
labelsFile	file listing the labels to be used as main plot titles instead of SNP names
specialSNPsFile	file listing the special SNPs (report file must be supplied with the special SNPs file)
reportFile	file listing the genders of the samples (special SNPs file must be supplied with the report file)
refFile	file with the reference genotype calls
sampleLookupFile	file matching up the same samples in the calls and references files that have different names
presenceAbsenceFile	file listing the presence and background allele of the subset of probesets that are plotted with the PAV transformation
temp.dir	name of the temporary directory created by <i>Ps_Visualization</i>
keep.temp.dir	flag to keep or remove the temporary directory
use.temp.dir	flag to use an already existing temporary directory

Table 10: The optional input file arguments for *Ps_Vis_Multi_ID*.

The labels file can contain new labels for the probeset IDs, multiallelic SNP IDs, or both. All of the IDs should be listed in the “probeset_id” column and all of the new labels should be listed in the “labels” column. For example, if the user wants to replace the multiallelic SNP ID *AFX-1234* with the label *SNP 1* and the probeset ID *AX-5678* with the label *probeset 1*, then both *AFX-1234* and *AX-5678* should be in the “probeset_id” column and *SNP 1* and *probeset 1* should appear correspondingly in the “labels” column.

In contrast to *Ps_Vis_Density*, *Ps_Vis_Multi_ID* does plot confidences. All plotting options are the same as in *Ps_Visualization*, and any options not listed are identical to *Ps_Visualization*.

Argument	Description
plot.intensity	flag indicating if intensity plots should be made
plot.genotype	flag indicating if genotype plots should be made
plot.calls	flag indicating if genotype calls should be used to assign shapes and colors in plots
plotNoCalls	flag indicating if NoCalls should be plotted with genotype calls
plot.ref	flag indicating if reference calls should be used to assign shapes and colors in plots
refNoCalls	flag indicating if NoCalls should be plotted with reference calls
plot.prior	flag indicating if prior ellipses should be plotted
plot.prior.ref	flag indicating if prior ellipses should be plotted on reference plots
plot.posterior	flag indicating if posteriors ellipses should be plotted
plot.posterior.ref	flag indicating if posterior ellipses should be plotted on reference plots
plot.pca	flag indicating if the PCA plots should be plotted for multiallelic probesets
transform	which data transformation should be used when plotting genotype plots
K	variable used with several transformations
autotetraploid	flag indicating if the data comes from an autotetraploid species
unknownGender	flag indicating if samples with unknown gender should be included in gender-separated plots
axis.mirror	flag for mirroring the X axis range and Y axis range (intensity plots only)
vertical.line	flag that controls if a light grey, dashed vertical line is drawn through $X = 0$ for calls plots

Table 11: The arguments for controlling plotting types in *Ps_Vis_Multi_ID*.

Gender-separated plots, batch plots, and multiallelic plots are the same in *Ps_Vis_Multi_ID* as in *Ps_Visualization*. If multiple plotting options are requested, then each batch is plotted in a separate column in order to make comparison across batches easier. Gender-separated plots have two plots per plotting option, for a total of eight plots if all plotting options are requested.

The page layouts in plots produced by *Ps_Vis_Multi_ID* may not be as clean as those produced by *Ps_Visualization*. This is because all of the probesets for a multiallelic SNP ID are plotted and only then is a new page started. If there are more probesets than can fit on one page, plotting continues straight onto the next page. The number of rows or columns may need to be adjusted to fit all of the probesets for a SNP on one page. All biallelic probesets are plotted before all multiallelic probesets.

All probesets for a SNP have the same shape and color assignments. There is one legend plot produced per multiallelic SNP if there is at least one multiallelic probeset. The legend plot does not appear when there are only biallelic probesets. Biallelic and then multiallelic probesets are plotted in the order in which they appear in the ps2snp file for each type.

Figure 35 shows the 7 probesets that make up the multiallelic SNP ID “SNP 1”. Probesets 1, 3, 4, 5, and 6 are biallelic probesets and probesets 2 and 7 are multiallelic probesets.

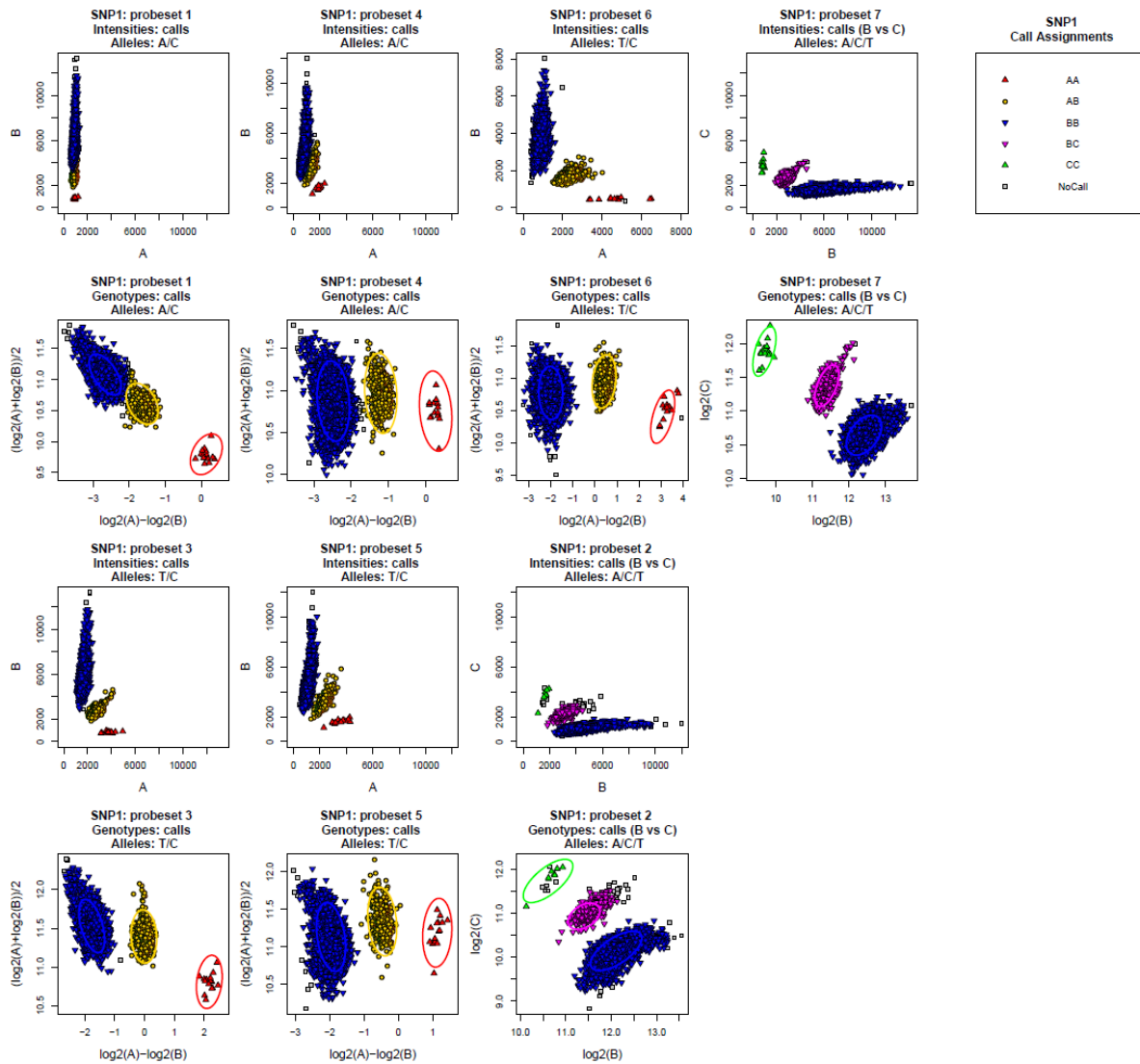


Figure 35: Probeset plots for one SNP ID.

Ps_Vis_Multi_ID and *Ps_Visualization* both use the same temporary files in the temporary directory to make plots. As a result, any SNPs that have previously been plotted in either *Ps_Visualization* or *Ps_Vis_Multi_ID* can be used with *Ps_Vis_Multi_ID*. The *use.temp.dir* flag tells *Ps_Vis_Multi_ID* to use an already extant temporary directory as long as that directory contains temporary files for all of the SNPs that will be plotted (default is **FALSE**).

4.4.2 Outputs

Ps_Vis_Multi_ID has the exact same output options as *Ps_Visualization*. *Ps_Vis_Multi_ID* either makes one file with all plots or one file per plot. The *plot.all* option controls how many files are output. The file type is controlled by *plot.type*. PDF files are generally much larger than PNG or SVG files and do not always display exactly the same way across different operating systems. The directory that output files are written to is controlled by *output.dir*.

4.5 CN_Visualization

CN_Visualization generates plots from standard AxiomCNV output files of fixed copy number genomic regions. Plots display the genotype calls, prior and posterior information, and the region details. The true copy number state may also be included. The cluster plots can help quality check genotyping in regions with variable copy number possibilities.

4.5.1 Inputs and Arguments

CN_Visualization takes four required arguments: the calls file (*CNregioncallsFile*), the priors file (*CNpriorFile*), the posteriors file (*CNposteriorFile*), and the region details file (*CNdetailsFile*). These arguments provide *CN_Visualization* with a list of regions to plot, the calls assignments for samples across the regions, and the prior and posterior distributions that were used in assigning the calls. All other input files and parameters are optional.

`CN_Visualization(CNregioncallsFile,CNpriorFile,CNposteriorFile,CNdetailsFile)`

- *CNregioncallsFile*: Axiom CN calls file (usually “AxiomCN.cnregioncalls.txt”)
- *CNpriorFile*: Axiom CN priors file (usually contains the name of the array and “cn_priors”)
- *CNposteriorFile*: Axiom CN posteriors file (usually “AxiomCN.cn_posteriors.txt”)
- *CNdetailsFile*: region details file (usually “AxiomCN.cnregions.details.txt”)

The optional arguments allow the user to supply a CN truth file, create an additional plate effects plot, and specify the name and location of the output file.

Argument	Description
CNtruthFile	Axiom CN truth file
plot.plate.effects	flag specifying whether to create the plate effects plot (default is FALSE)
output.dir	name of the output directory (default is <code>Graph_Outputs</code>)
output.File	name of output file with all plots

Table 12: The optional file arguments for *CN_Visualization*.

If the name of an output directory is not supplied, *CN_Visualization* creates a folder named “Graph_Outputs” in the working directory. All files are written to this output directory.

The default behavior for *CN_Visualization* is to create three files: plate histogram plots, batch histogram plots, and heatmap plots. The default file names are “CN_Plate_Histogram.pdf”, “CN_Batch_Histogram.pdf”, and “CN_Heatmap.pdf”. If the user supplies a file name *output.File*, all plots are written to the one file in *output.File*. The user should only supply one file name, and cannot rename the three files. If the truth file is supplied, the plate histogram plots file is named “CN_Plate_Histogram_w_Truth.pdf”. If the plate effects flag is set to TRUE, an additional file is created: the plate effects plots, with a default file name of “CN_Plate_Effects.pdf”. If the file name is supplied, this set of plots is created in the file as well.

The calls, priors, posteriors, details, and truth files may be gzipped and used as input files. R handles reading compressed gzipped files internally without needing to expand them.

The region calls file holds the information about the copy number assignment for each sample in a region. The priors and posteriors files hold the expected and observed distribution of copy number clusters across the regions. The region details file holds information on the distribution model used for each region and which type of mixture model it is. If the CN truth file is supplied, it holds the true copy number state for each sample in each region.

CN_Visualization will output a warning if the truth file is supplied without the calls file. Both a region calls file and a CN truth file are required to create plots with the truth data.

CN_Visualization uses the `ggplot2` R package [3]. If `ggplot2` is not installed, *CN_Visualization* will quit with an error message asking for it to be installed. `ggplot2` is automatically loaded when running *CN_Visualization* and only needs to be installed once.

4.5.1.1 Batch Histograms Batch histograms display the distribution of the samples' copy number states across all plates for each region. These plots allow the user a high-level view of the behavior within each region and can be used to visually inspect the overall copy number results. Figure 36 shows the histogram of 250 samples in copy number region 44. The majority of samples are copy number 1, followed by copy number 0 and copy number 2. A very small number of samples are copy number 3. This region displays a variety of copy number calls and the plot shows that the samples are spread out among the different copy number states.

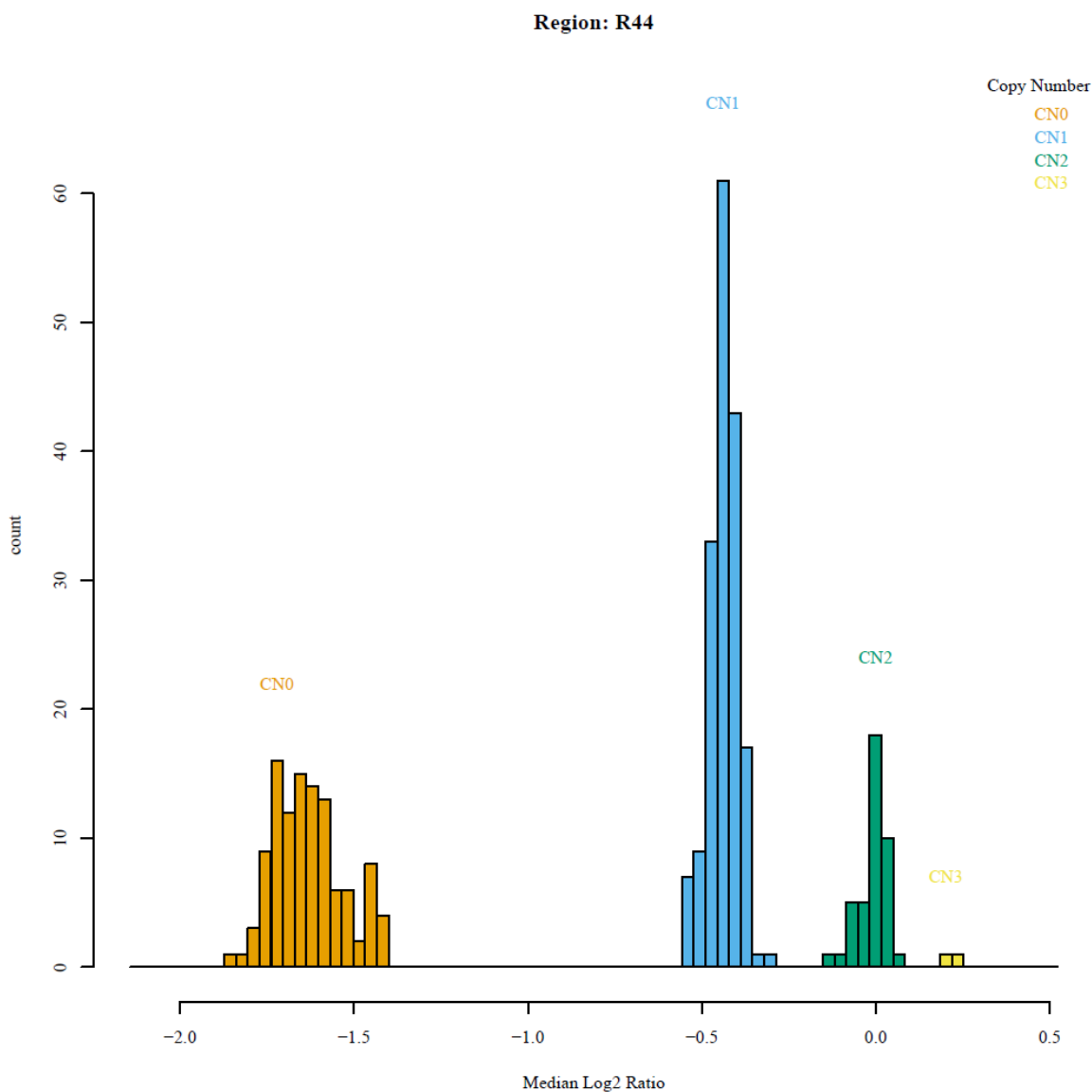


Figure 36: Batch histogram for one CN region.

4.5.1.2 Plate Effects The two types of plots for plate effects show the distribution of the models used in assigning copy number states for each plate. These plots allow the user to see how similar the models used are for each plate. Figure 37 shows the density and histogram plots for the four plates used in this example. All four plates have very similar models.

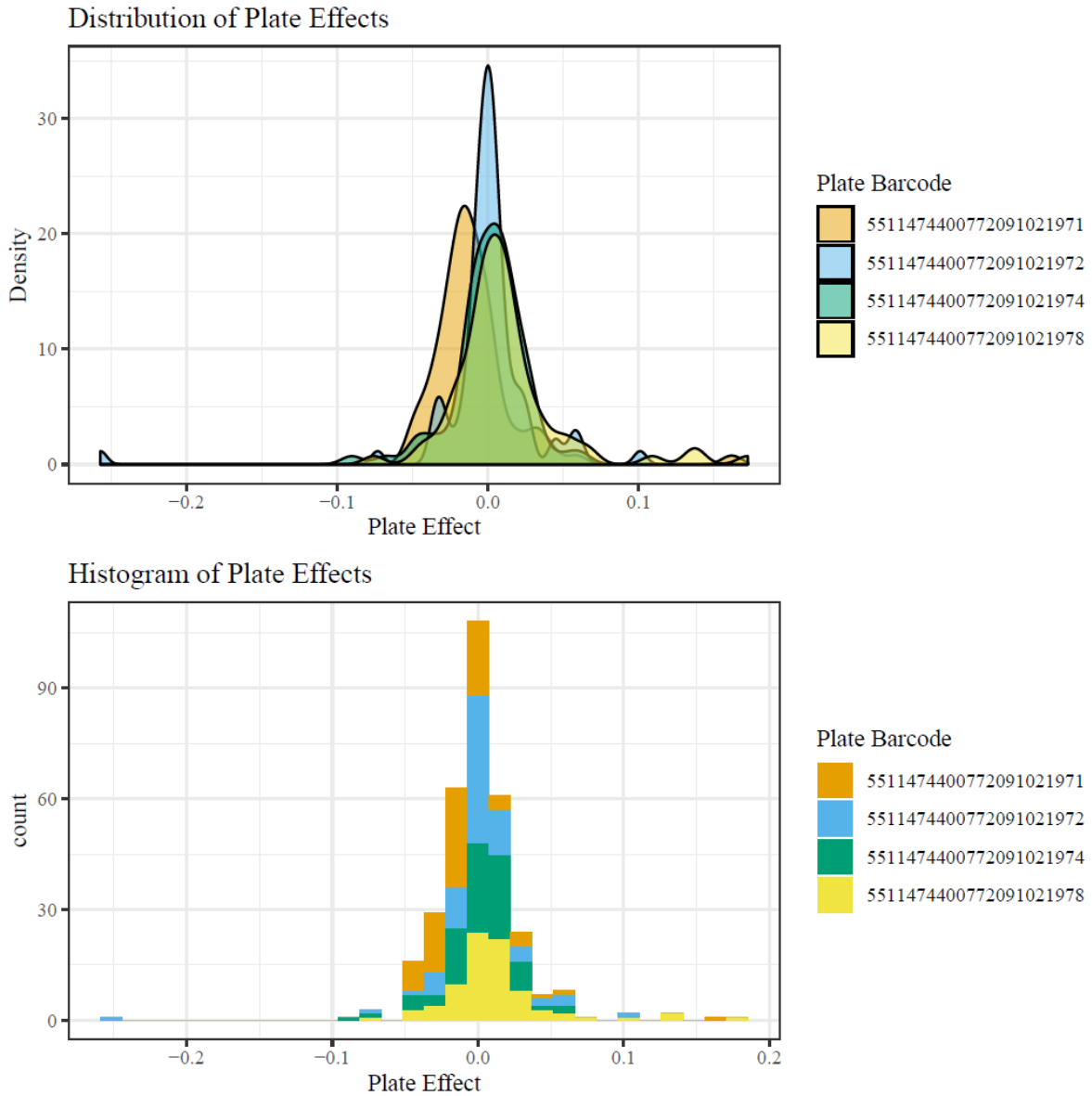


Figure 37: Plate effects for four plates.

4.5.1.3 Heatmap The heatmap plot displays the copy number assignments for all samples across all regions in the data set. The rows are the regions and columns are the samples. Copy number 2 is white, the same color as the background, so it is easy to see the non-copy number 2 trends across regions. One heatmap is produced per data set.

Figure 38 shows the heatmap for the example data set, with 82 regions and 250 samples. The widescale view of the regions and samples at the same time shows patterns across regions. For example, regions 41 to 44 show that most samples have copy number 0 or 1, and region 53 has many samples with copy number 3.



Figure 38: Heatmap for 82 regions and 250 samples.

4.5.1.4 Plate Histograms The plate histogram plots show the distribution of samples across the different plates for each region. These plots can be used to investigate if there are any plate effects, where one plate has a different distribution of copy number assignments compared to all other plates for the region. Priors and posteriors appear on each histogram. Priors are darker, dashed lines and posteriors are lighter, solid lines.

Figure 39 shows the histograms of sample copy number distribution across the four plates on 250 samples used in region 12. The overall distributions are very similar on the four plates, with most samples assigned to copy number 2 or 1, and a smaller group assigned to copy number 0. The locations of the priors (dashed lines) are approximately the same location as the samples, and the posteriors (solid lines) are very similar to the priors and are visible in the dashes of the priors. These histograms show good performance in this region.

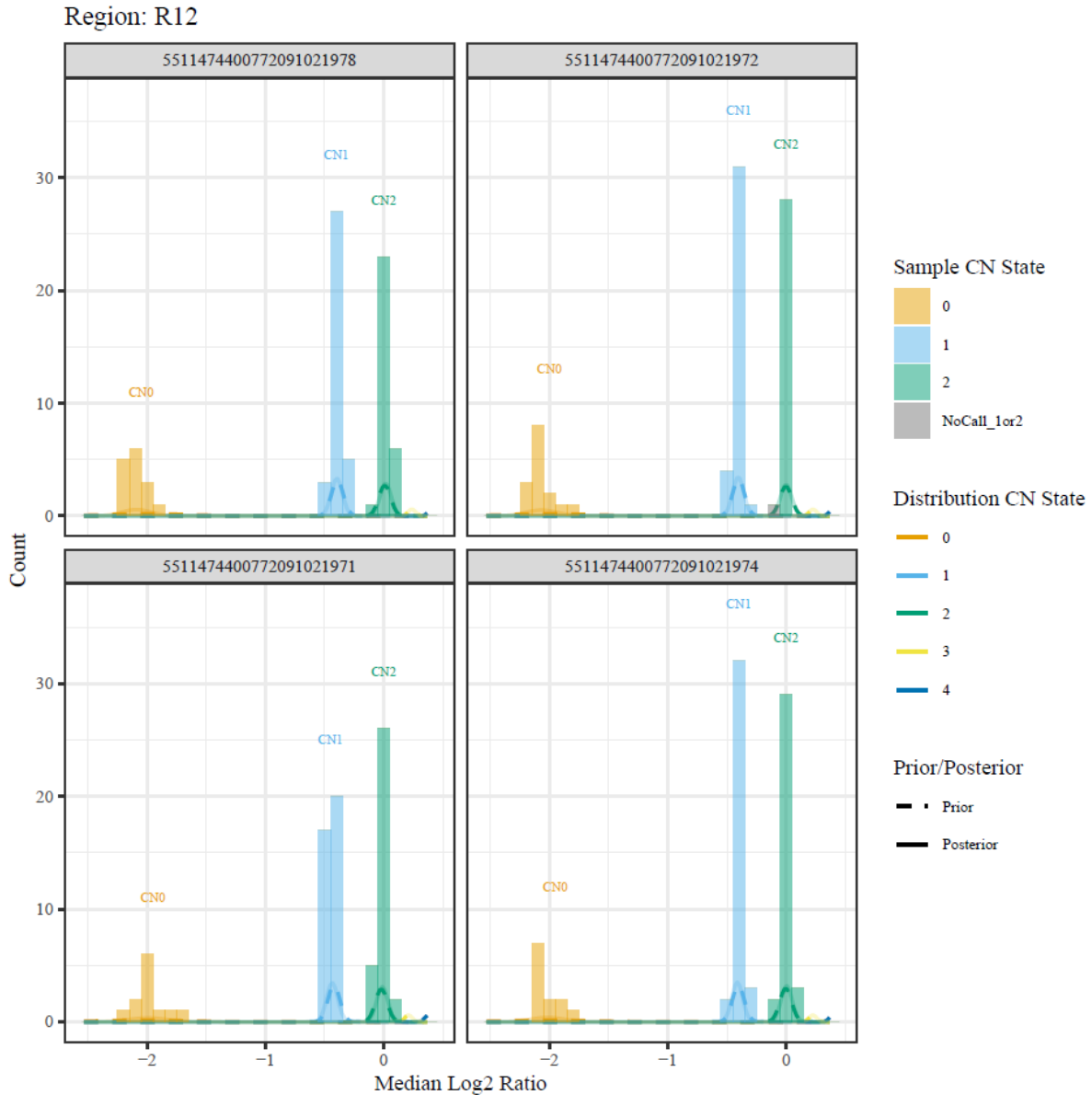


Figure 39: Histograms of 250 samples on four plates for one copy number region.

4.5.1.5 Plate Histograms with Truth When the true copy number state is known for some or all of the samples in a region, the plate histograms will be made with a second set of histograms which display the true copy number states. Regions without truth values have empty histogram plots which display only the priors. Samples without a truth value in a region with other truth values are plotted as NA (missing). Truth histograms can be used to verify how well the copy number assignment algorithm worked. If the copy number assignments match up to the truth values well in regions with truth values, then the user can have more confidence in copy number assignments in regions without truth values.

Figure 40 shows the histograms of assigned copy number and true copy number state in region 34. The distributions of the assigned copy number states are very similar to the true copy number states, showing that the assignments correctly captured the different copy number states in the samples.

Figure 41 shows what the output looks like when a region does not have true copy number states. The true plots are blank except for the priors. This combination of plots allows the user to easily compare the location of the priors to the location of the samples and their copy number assignments.

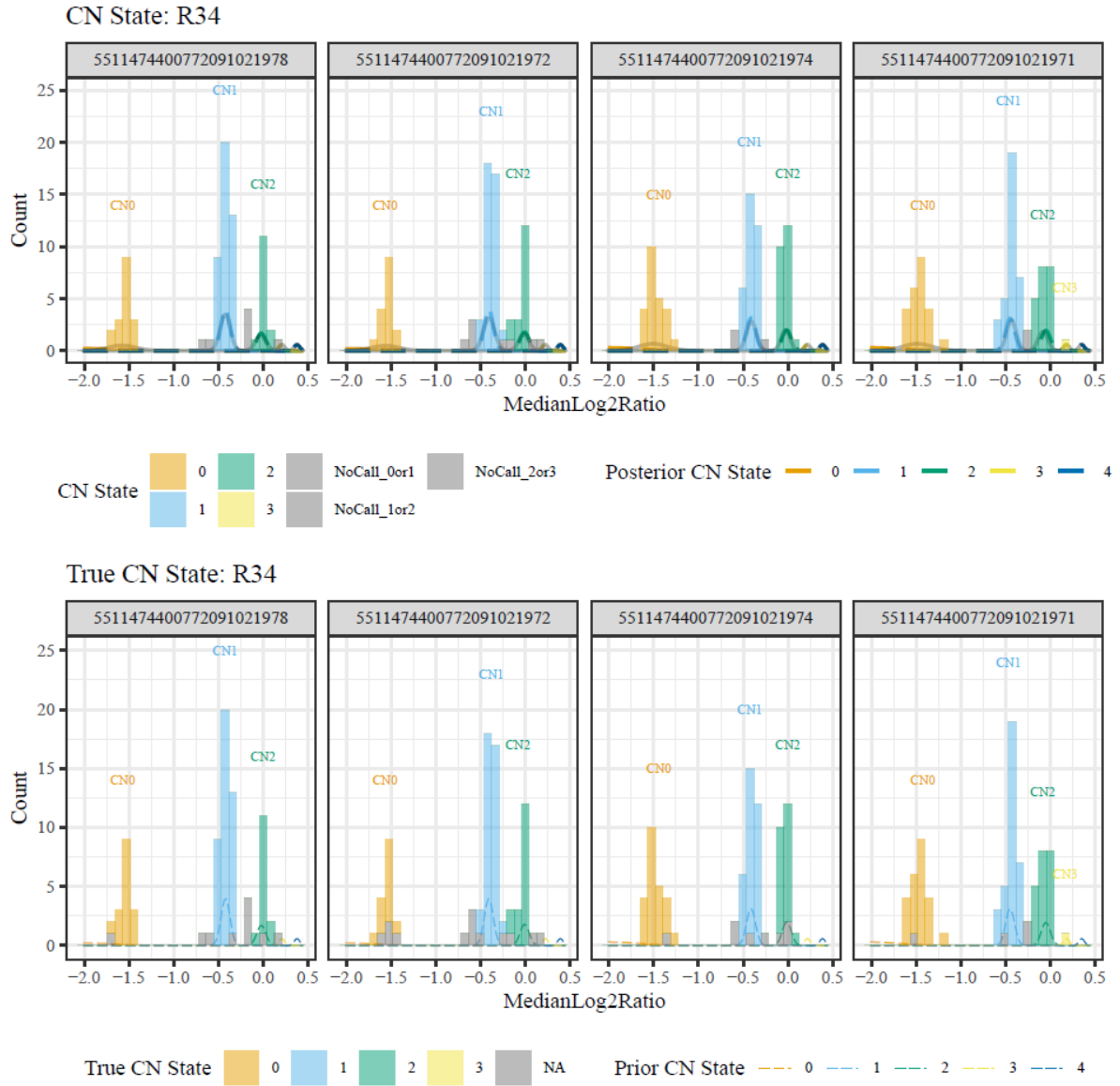


Figure 40: Histograms of 250 samples on four plates for one copy number region, with the true copy number values for the samples.

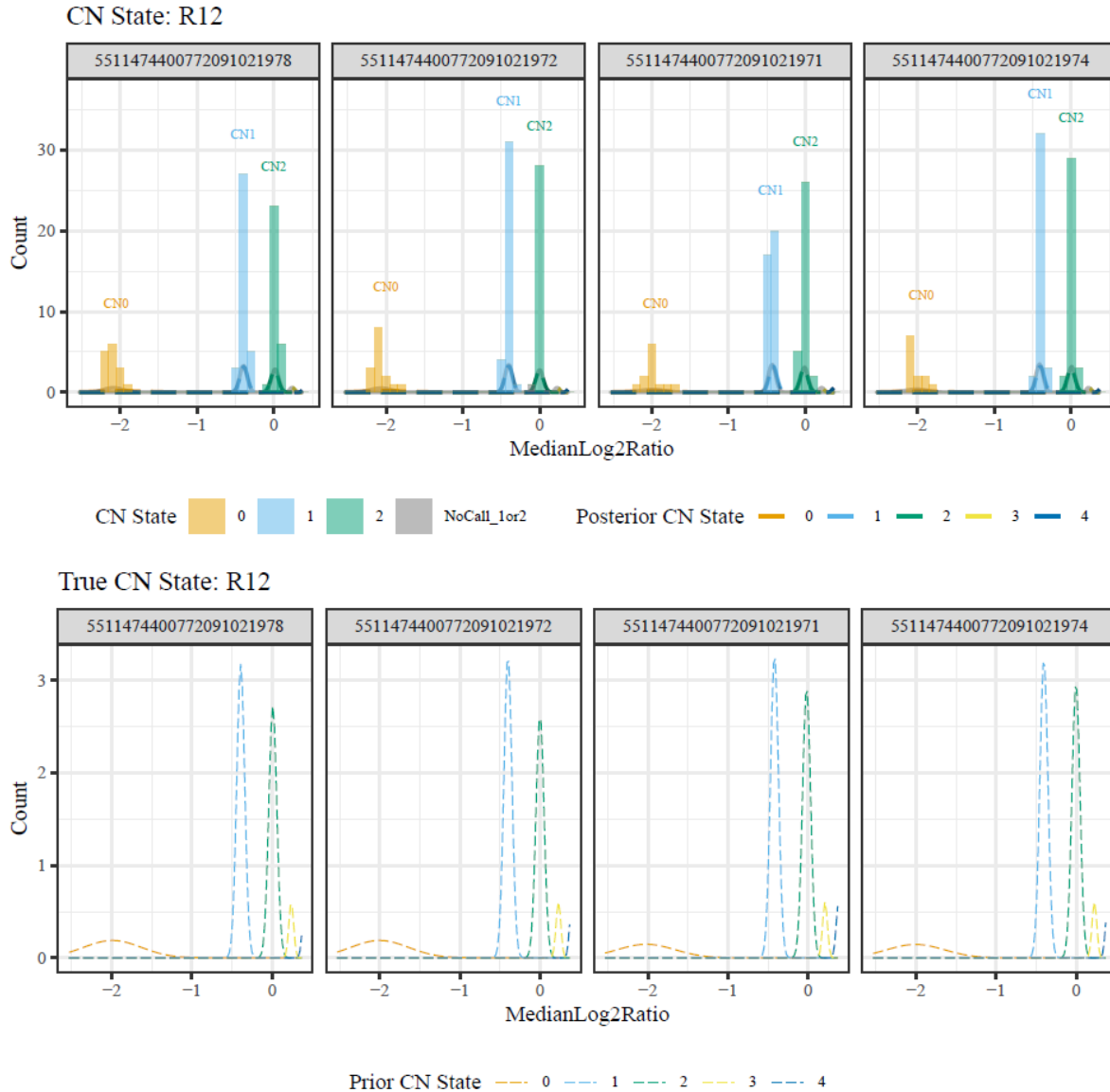


Figure 41: Histograms of 250 samples on four plates for one copy number region, without true copy number values for the samples.

4.5.2 Outputs

CN_Visualization either produces one file containing all plots or three or four files containing various plots. The file type is always PDF. The user controls which directory the output files are written to so that if multiple output files are requested, all files will be written to the same directory.

The default output is to write the plots to three different files: the batch histograms, the heatmap, and the plate histograms. If the plate effect plots are requested, either a fourth file is created or the plate effect plots are written into the one file whose name was supplied by the user.

Some of the plots can be time consuming to create, especially when there are thousands of samples. If there are a large number of samples, we recommended plotting a smaller number of regions (< 100) at one time.

4.6 Autotetraploid Functions

Autotetraploid species such as rose and potato can be genotyped on the Axiom platform. However, the workflow for assigning genotype calls is not the same as for diploid species. We suggest using the `fitTetra` [9] R package for assigning genotype calls, and then using *Ps_Visualization* on the produced calls.

`fitTetra` was developed by Dr. RE Voorips at Wageningen University's Plant Breeding section [10]. The paper describing the *fitTetra* algorithm is available at [2].

The summary file (usually "AxiomGT1.summary.txt") is required for running `fitTetra` but the calls, confidences, and posteriors produced through the normal Step 7 genotyping of the Best Practices Genotyping Analysis Workflow are not needed and will not have useful results. When genotyping autotetraploid species using APT, it is necessary to run the `apt-genotype-axiom` command with the `--summaries-only` option set to `TRUE`. This will produce only the summary file, which can be used with *fitTetra_Input* to begin the genotype calling process.

If we wanted to run the example `apt-genotype-axiom` script for Step 7 genotyping on an autotetraploid dataset, the command would be:

```
../bin/apt-genotype-axiom \  
--log-file <OUTDIR>/apt-genotype-axiom.log \  
--xml-file <ANALYSIS_FILES_DIR>/<axiom_array>_96orMore_Step2.r<#>. \  
    apt-genotype-axiom.AxiomGT1.xml \  
--analysis-files-path <ANALYSIS_FILES_DIR> \  
--out-dir <OUTDIR> \  
--summaries-only true
```

The output files produced are the summary file and a log file.

Once the summary file has been created, `fitTetra` and the *fitTetra* functions in `SNPolisher` can be run.

The steps for downloading and installing the `fitTetra` package are similar to `SNPolisher`. The only major difference is that `fitTetra` is available on CRAN. To download and install `fitTetra`, use the *install.packages* command:

```
> install.packages("fitTetra")
```

This command will tell R that the `fitTetra` package should be downloaded from the CRAN repository. R will ask the user to select a CRAN mirror. Select the mirror that is geographically closest for faster downloading.

Once the package is downloaded, it must be installed:

```
> library(fitTetra)
```

The required input file format for `fitTetra` is different than the format of the Axiom output files. Before running `fitTetra` functions, the user must run the *fitTetra_Input* function in the `SNPolisher` package. This will reformat the summary file to the correct format for `fitTetra` use.

The `fitTetra` output files are not in the required file format for use with `SNPolisher`. Once `fitTetra` has finished running, the `SNPolisher` function *fitTetra_Output* will reformat the `fitTetra` output files into the expected calls, confidences, and posteriors files for use with `SNPolisher`.

Note that runtimes for `fitTetra` increase as the number of samples increase. If there are more than 300 samples, we recommend splitting the summary file up into multiple summary files and running `fitTetra` jobs in parallel. There is a short description of the *fitTetra* algorithm in section 4.6.2, explaining why this approach works.

An example of the workflow for autotetraploid functions is given in section ??.

4.6.1 fitTetra_Input

The `fitTetra` functions take a summary file as input. Although a APT summary file has one row per SNP and as many columns as there are samples, `fitTetra` expects the input summary file to have one row per SNP and sample, so that each SNP has as many rows as there are samples. *fitTetra_Input* takes a

APT summary file and outputs a `fitTetra`-compatible summary file to be used as input with the `fitTetra` functions. When a summary file is large, this function may take up to an hour to run.

`fitTetra_Input` also has the ability to split one APT summary file into multiple `fitTetra`-compatible summary files. If the user supplies a number of markers per summary file, `fitTetra_Input` will output the reformatted summary data into multiple files, where each file has the number of markers selected by the user. For example, if the APT summary file has 1,124 markers and the user selects 100 markers per file, `fitTetra_Input` will create 12 output files where the first 11 output files have 100 markers each and the last output file has 24 markers. The summary file cannot be split on the number of samples.

4.6.1.1 Inputs and Arguments `fitTetra_Input` takes one input file (summary file). `fitTetra_Input` takes two required arguments and two optional arguments:

```
fitTetra_Input(summaryFile,output.file,pidFile,output.count)
```

- *summaryFile*: name of the summary file output by APT (default is “AxiomGT1.summary.txt”).
- *output.file*: name of the `fitTetra`-formatted output summary file. If no file name is supplied, the default used is “AxiomGT1.summary.fitTetra.txt”.
- *pidFile*: (Optional) list of probesets to convert for use in `fitTetra`
- *output.count*: (Optional) number of markers per output file

The Axiom summary file may be gzipped and used as an input file. R handles reading compressed gzipped files internally without needing to expand them.

If the user supplies a number for *output.count*, multiple output files will be created. The output files will be named using the *output.file* argument, where a number will be included in the file name to indicate which output file has been used. For example, if *output.file* was “summary.txt” and `fitTetra_Input` created three output files, the output files would be named “summary.1.txt”, “summary.2.txt”, and “summary.3.txt”.

4.6.1.2 Outputs `fitTetra_Input` outputs one (or more) text file, the `fitTetra`-formatted summary file (“AxiomGT1.summary.fitTetra.txt”). This is the input file for the `fitTetra` functions.

4.6.2 fitTetra

The `fitTetra` algorithm [2] assigns genotype calls to markers by fitting Gaussian mixture models for each marker and selecting the model that fits best. The `fitTetra` function produces a mixed model with five component distributions for the five possible genotypes (nulliplex to quadruplex) (see figure 42). Each marker’s model is fit independently from any other marker. Because of this independence, large datasets can be split up by marker and run in `fitTetra` without affecting the genotype calls assigned. `fitTetra_Input` divides summary files by markers. Note that if a dataset is split by samples instead of by marker, the results of the model fitting will be affected.

The `saveMarkerModels` function in the `fitTetra` package is designed to fit models for each marker in a dataset. It is a “convenience function” that calls the `fitTetra` function for each marker.

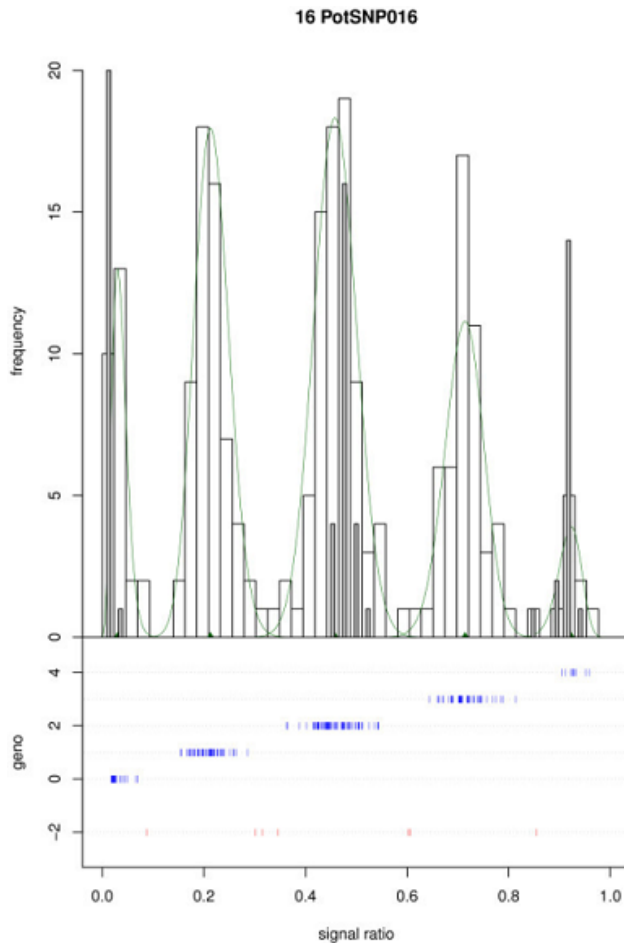


Figure 42: Standard graphical output for `fitTetra` for tetraploid potato varieties. The fitted distributions are plotted over the histogram of the signal ratios, and the assigned genotype calls are plotted in the lower part of the figure. Figure from [2].

4.6.2.1 Inputs and Arguments `saveMarkerModels` takes one input file (summary file). `saveMarkerModels` takes three required arguments and 20 optional arguments:

```
saveMarkerModels(data, modelfile, scorefile)
```

`data` is the data frame that holds the summary file. `modelfile` is the name of the file that holds the models results from `fitTetra`. `scorefile` is the name of the file that holds the scores (calls) for every sample per marker.

Of the 20 optional arguments, we suggest that the following 11 arguments are useful:

The `markers` argument lets the user specify which markers to analyze. It takes the row number that the markers are on in the dataset instead of the marker names. `try.HW` should only be set to TRUE for datasets where the user thinks that the markers may be somewhat close to Hardy-Weinberg equilibrium.

If `fitTetra` is run on a Linux system with multiple cores, we suggest using the `ncores` argument for parallel processing.

`p.threshold` is the minimum p-value that the best model must achieve to assign the fitted genotype call to a sample. `call.threshold` is the minimum fraction of samples per marker that must be assigned a genotype call for the entire marker to be accepted. If the fraction of calls assigned is smaller than `call.threshold`, then the entire marker is rejected and `saveMarkerModels` does not produce any calls for that marker. Note that any markers that are rejected will not appear in the results of `fitTetra_Output`, and the final calls, confidences, and posteriors files may have fewer markers than the original summary file.

Variable	Description	Default
markers	integer vector listing the row numbers of which markers to analyze	NA (all markers)
maxiter	maximum number of times the internal <i>nls</i> function is called	500
ncores	number of processor cores used for parallel processing (Linux only)	NA (no parallelization)
try.HW	fit models with and without a constraint on Hardy-Weinberg equilibrium	TRUE
dip.filter	whether to select models with or without a dip	1 (no dips allowed)
p.threshold	minimum p-value required to assign a genotype call to a sample	0.99
call.threshold	minimum fraction of samples to have genotypes called	0.6
logfile	name of the output log file	"" (no file)
allmodelsfile	name of the file containing all models fit per marker	"" (no file)
plot	whether to plot results for each marker	"none"
plot.type	type of plots to produce	"png"

Table 13: The suggested optional threshold arguments for *saveMarkerModels*.

plot allows the user to plot only the fitted models, all of the models tried, or nothing at all. The default value of “none” is best for data sets with more than 20 or 30 markers. If the user does want to produce plots, we suggest using “fitted” instead of “all”. *plot.type* takes “png”, “emf”, “svg”, or “pdf”. We suggest using “png” or “pdf”.

4.6.2.2 Outputs *saveMarkerModel* outputs two to four text files, and any plots if requested. The models file and the score file are always produced, and we suggest producing the log file and the allmodels file. The text files are written to the working directory and the plots are written to a subfolder created in the working directory, where the subfolder has the same name as the log file. The score file is used as the input for *fitTetra_Output*.

4.6.3 fitTetra_Output

fitTetra_Output processes output from *fitTetra* to generate *SNPolisher*-compatible input. *SNPolisher* expects cluster call data as input, not cluster distributions. *fitTetra_Output* transforms *fitTetra* output to the corresponding *SNPolisher* input.

fitTetra_Output takes the score file output by *fitTetra* and the summary file output by APT and produces a calls, confidences, posteriors and summary file, similar to those produced by Axiom. An error log is also created, which holds any lines from the score file that that are not formatted correctly. It is uncommon to have errors in the score file, but this may occur if the dataset is very large. If there are no errors, the log file will be empty. If the dataset is very large, this function may take up to an hour to run. Note that any markers that were rejected by *fitTetra* will not appear in the score file, and will not be output in the calls, confidences, and posteriors files. There may be fewer markers in these files than in the original summary file.

The new summary file contains the same summary data as the original APT summary file because *saveMarkerModel* and *fitTetra_Output* only use the summary data but do not change it. The new summary file may have fewer markers than the original APT summary file because any markers that were rejected by *fitTetra* did not appear in the score file. The set of new calls, confidences, posteriors, and summary files contain the exact same markers, and it can better to keep the new summary file with other new files for consistency.

4.6.3.1 Inputs and Arguments *fitTetra_Output* takes two input files, the score file output and the summary file. *fitTetra_Output* takes seven required arguments and seven optional arguments:

```
fitTetra_Output (scoreFile, summaryFile, output.callFile, output.confFile,  
                output.postFile, output.summaryFile, output.logfile,  
                conf.threshold, output.dir, pidFile, output.removedFile,  
                eureka, CES.k, perl.script)
```

- *scoreFile*: score file output from *fitTetra*
- *summaryFile*: APT summary file (usually “AxiomGT1.summary.txt”)
- *output.callFile*: name of the calls file produced from the score file
- *output.confFile*: name of the confidences file produced from the score file
- *output.postFile*: name of the posteriors file produced from the score file
- *output.summaryFile*: name of the new summary file produced from the score file
- *output.logFile*: name of the errors log produced from the score file
- *conf.threshold*: (Optional) threshold for the confidences from *fitTetra*
- *output.dir*: (Optional) output directory for result files. If an output directory is not supplied, a folder named “fitTetra_Output” will be created in the working directory and all output will be written to it.
- *pidFile*: (Optional) list of probeset IDs to be used in *fitTetra_Output*
- *output.removedFile*: (Optional) name of the file that the list of removed markers is written to. This file is only generated when a *pidFile* is supplied.
- *eureka*: (Optional) flag indicating if the genotyped data was produced with a Eureka array
- *CES.k*: (Optional) integer value used in the data transformation for Eureka data (default is 2).
- *perl.script*: (Optional) location of an alternate perl script for processing the data (default is the script bundled with *fitTetra_Output*)

The score and Axiom summary files may be gzipped and used as input files. R handles reading compressed gzipped files internally without needing to expand them.

Any calls from *fitTetra* with confidences greater than *conf.threshold* will be set to “No Call” (-1) in the calls file.

pidFile holds the list of marker names to be used in *fitTetra_Output*, one marker per line. *fitTetra_Output* will create calls, confidences, and posteriors for the specified probesets only. The first row of this file should always be “probeset_id”. If *pidFile* is not provided, then all probesets from the *fitTetra* score file will be used. If *pidFile* is provided and *output.removedFile* is not provided, then the list of removed probesets will be named “removed.fitTetra.ps”.

eureka tells *fitTetra_Output* that the original summary data was produced with a Eureka array, and not an Axiom array. This is necessary information and *fitTetra_Output* will not produce results for Eureka data without the *eureka* flag being set to TRUE. If *fitTetra_Output* is run on Eureka data with *eureka* as FALSE, the results in the call, confidence, posteriors, and summary will be empty. All of the files will be created and the headers will exist, but there will be nothing else. If all newly produced files are blank, the user should double-check if the data is from a Eureka array and set *eureka* to TRUE.

perl.script should only be supplied by advanced users who are familiar with the data transformations to convert *fitTetra* data to SNP`olisher`-compatible data.

4.6.3.2 Outputs *fitTetra_Output* produces five files: the calls, confidences, posteriors, summary, and log files. The calls, confidences, and posteriors are necessary for running SNP`olisher`. These files can be used either with the new summary file or with the original APT summary file (usually “AxiomGT1.summary.txt”) to run all other SNP`olisher` functions. We strongly suggest using *Ps_Visualization* to generate cluster plots for the *fitTetra*-assigned genotype calls.

4.7 Ps_Extract

Ps_Extract is a utility function that extracts the data for a supplied probeset list with an optional sample list from one or more Axiom genotyping files: calls, confidences, biallelic and multiallelic posteriors, biallelic and multiallelic priors, summary, and reference files. It can be helpful to run *Ps_Extract* before running functions that use the summary file and calls file such as *Ps_Visualization*. If a user has multiple SNP lists to make plots from a dataset with a very large summary or calls file, then running *Ps_Extract* on the combined list of SNPs will produce smaller versions of the summary and calls file that contain only the SNPs that the user will be plotting. This can save a lot of time in plotting because *Ps_Visualization*, *Ps_Vis_Density*, and *Ps_Vis_Multi_ID* will not have to search through the entire summary file to find the data for the SNPs being plotting, and instead only need to find the data in the smaller files.

If a sample file is supplied along with a probeset list, then only those samples are extracted for the file types that have data per sample: summary, calls, confidences, and references. Files that do not have data per sample only extract probesets (priors and posteriors files).

Ps_Extract also has a secondary function: it will extract the first column of a genotyping file. If a user needs to know which SNPs appear in a large calls or summary file, then running *Ps_Extract* on only the calls file or only the summary file will produce the first column of that file (i.e. the list of SNPs that appear in the file). Note that the first column of a summary file has one row per allele per SNP, so the calls file is usually a better input file to obtain the list of SNPs in a dataset. *Ps_Extract* will extract the first column of any of the input file types as long as that is the only input file.

4.7.1 Inputs and Arguments

Ps_Extract either takes a *pidFile* with the list of SNPs to extract and the files to extract them from, or a single file to extract the first column from.

```
Ps_Extract(pidFile,summaryFile,output.summary)
```

- *pidFile*: text file listing probeset IDs
- *summaryFile*: Axiom summary file (usually “AxiomGT1.summary.txt”)
- *output.summary*: the name of the extracted, smaller summary file
- *sampleFile*: text file listing sample IDs (optional)

Ps_Extract has two arguments for each file type: an input file name and an output file name. When a SNP list is provided, each input file supplied has a matching extracted output file. If an extracted, output file name is not supplied, the default is “extract_” and then the file type (e.g. “extract_calls.txt”).

Variable	Description	Default
pidFile	list of SNPs to be extracted from all files	
sampleFile	list of samples to be extracted from all files with samples	
output.pid	name of output file when first column is extracted	“pid_extract.ps”
output.dir	output directory for extracted files	“extract”
summaryFile	Axiom summary file	
output.summary	extracted summary file	“extract_summary.txt”
callFile	Axiom calls file	
output.calls	extracted calls file	“extract_calls.txt”
confidenceFile	Axiom confidences file	
output.confidences	extracted confidences file	“extract_confidences.txt”
posteriorFile	Axiom biallelic posteriors file	
output.posteriors	extracted biallelic posteriors file	“extract_posteriors.txt”
multiallele.posteriorFile	Axiom multiallelic posteriors file	
output.multiposteriors	extracted multiallelic posteriors file	“extract_multiposteriors.txt”
priorFile	Axiom biallelic priors file	
output.priors	extracted biallelic priors file	“extract_priors.txt”
multiallele.priorFile	Axiom multiallelic priors file	
output.multipriors	extracted multiallelic priors file	“extract_multipriors.txt”
refFile	genotype reference file	
output.ref	extracted reference file	“extract_references.txt”

Table 14: The input arguments for *Ps_Extract*.

4.7.2 Outputs

Ps_Extract either outputs one list of SNPs with the header of “probeset_id” or it outputs smaller versions of the input files using the SNPs listed in the *pidFile*. All extracted files are in the same format as the input files, and APT headers are output to match the input files. The user should double-check that extracted calls files, multiallelic posterior files, and multiallelic priors files have the same set of APT header comments as the original calls, multiallelic posteriors, and multiallelic priors files.

5 Vignettes

SNPolisher comes with nine example vignettes. The first vignette is a short, general introduction to the package. The second vignette is an introduction to *Ps_Visualization*. The third vignette is an introduction to *CN_Visualization*. The remaining vignettes are examples of plotting with human, wheat, and case-control data; batch plotting; plotting multiple probesets per multiallelic SNP ID with *Ps_Vis_Multi_ID*; and using *fitTetra_Input* and *fitTetra_Output* on autotetraploid data. The vignettes demonstrate how SNPolisher performs with different types of genotype data, how to use the functions, and several extra points to make running SNPolisher more convenient.

The code in the vignettes can be copied and pasted directly into an R session, either in base R or in RStudio. The example data sets are mostly included in the package and are automatically available to the user when SNPolisher is installed. The vignettes have their own example data sets which are loaded using the *data* command.

The example data sets are also bundled with the SNPolisher package in the *extdata* folder of the package. This package is installed wherever the SNPolisher package is installed on your computer. Because that location can be different for different users, the reference to the folder uses the R session's own call to the computer's operating system to identify the location of the *extdata* folder:

```
system.file("extdata", package = "SNPolisher")
```

You can copy this data to another location and use it for additional practice. The autotetraploid vignette's data is only available in the *extdata* folder as it is too large to be bundled directly into the package.

Before running any SNPolisher commands, remember to load the SNPolisher library:

```
> library("SNPolisher")
```

To see a list of all available vignettes in the SNPolisher folder, run this command:

```
browseVignettes("SNPolisher")
```

Each vignette can be called individually by its name:

```
vignette("SNPolisher-vignette", package="SNPolisher")
vignette("VIS-vignette", package="SNPolisher")
vignette("human-vignette", package="SNPolisher")
vignette("wheat-vignette", package="SNPolisher")
vignette("batch-vignette", package="SNPolisher")
vignette("casecontrol-vignette", package="SNPolisher")
vignette("multi-vignette", package="SNPolisher")
vignette("autotetraploid-vignette", package="SNPolisher")
vignette("CNV-vignette", package="SNPolisher")
```

5.1 Introductory Vignettes

There are three introductory vignettes: one for the SNPolisher package, one for *Ps_Visualization*, and one for *CN_Visualization*. The package vignette includes descriptions of all seven functions in the package, and links to the other vignettes. The introduction to *Ps_Visualization* vignette includes a small data set with eight files (pid, summary, calls, posteriors, multiallelic posteriors, multiallelic priors, special SNPs, and report files) and an example command that shows the basic structure of running *Ps_Visualization*. The introduction to *CN_Visualization* vignette includes a small data set with five files (calls, details, priors, posteriors, and truth files), a description of the five types of plots made by *CN_Visualization*, and an example command that shows the basic structure of running *CN_Visualization*.

5.2 Vignette: Human Data

The data set in this vignette contains 381 probesets genotyped on 111 human samples. There are 16 files: the six output files from the APT for Axion arrays (*AxiomGT1.calls.txt*, *AxiomGT1.confidences.txt*, *AxiomGT1.snp-posteriors.txt*, *AxiomGT1.snp-posteriors.multi.txt*, *AxiomGT1.priors.multi.txt*, and *AxiomGT1.summary.txt*); one file containing the samples and their genders (*AxiomGT1.report.txt*); one file containing a list of probeset IDs and colors for highlighting samples in plots (*samples.txt*); a list of labels to update the main titles in the plots (*labels.txt*); the PolyHighResolution probeset list (*PolyHighResolution.ps*); the special SNPS file (*example1.specialSNPs*); and five files for OTV plotting.

This example uses the default values for humans. There are 53 probesets in the PHR probeset list. The first command plots all 53 with the default values.

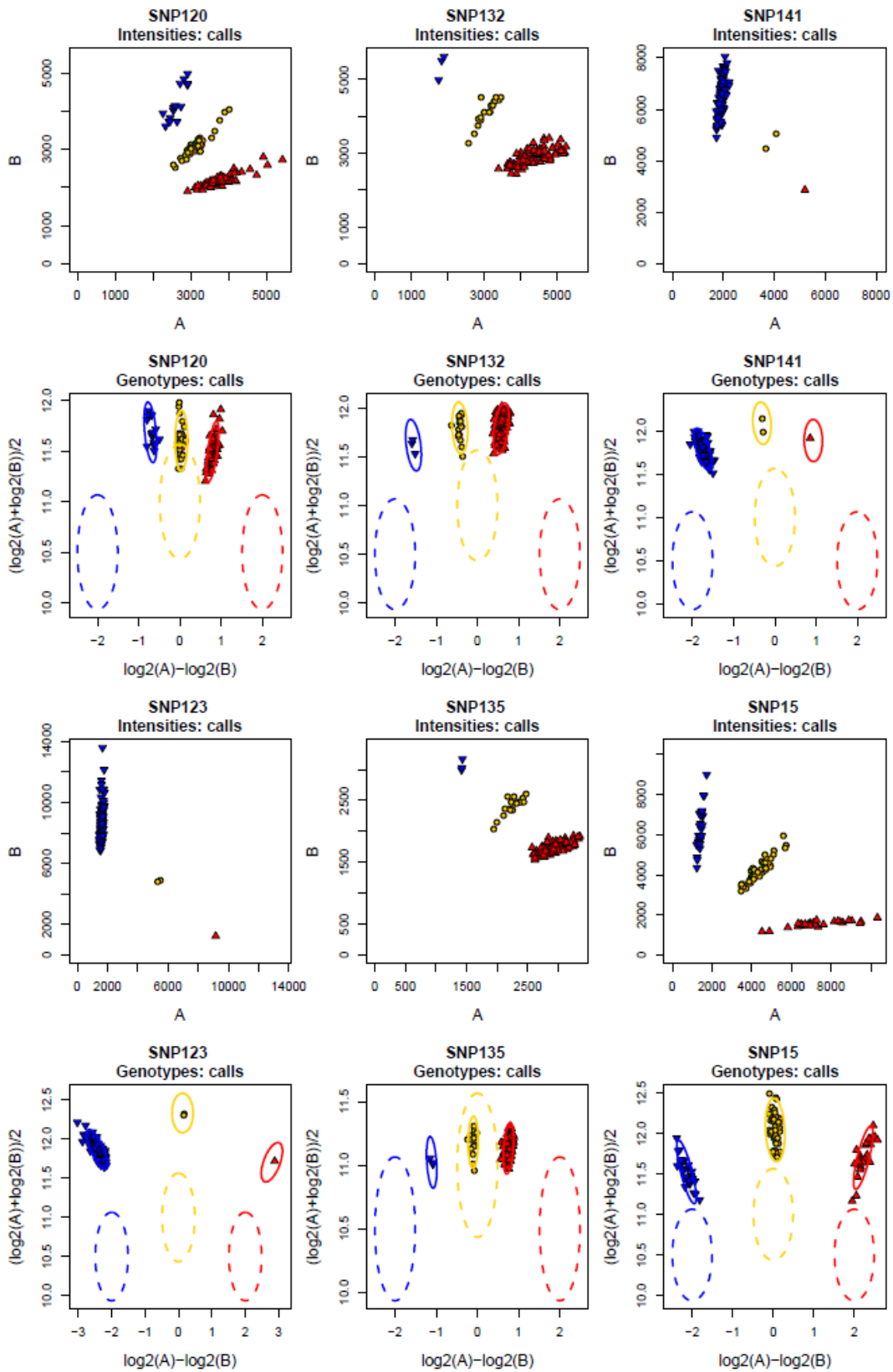


Figure 43: Standard output from *Ps_Visualization* using all default values.

In this next plot, the color of the BB cluster has been changed to dark green, the colors of 5 samples have been set to purple and 8 samples have been set to turquoise, the labels of 4 SNPs have been changed, and priors and posteriors are plotted.

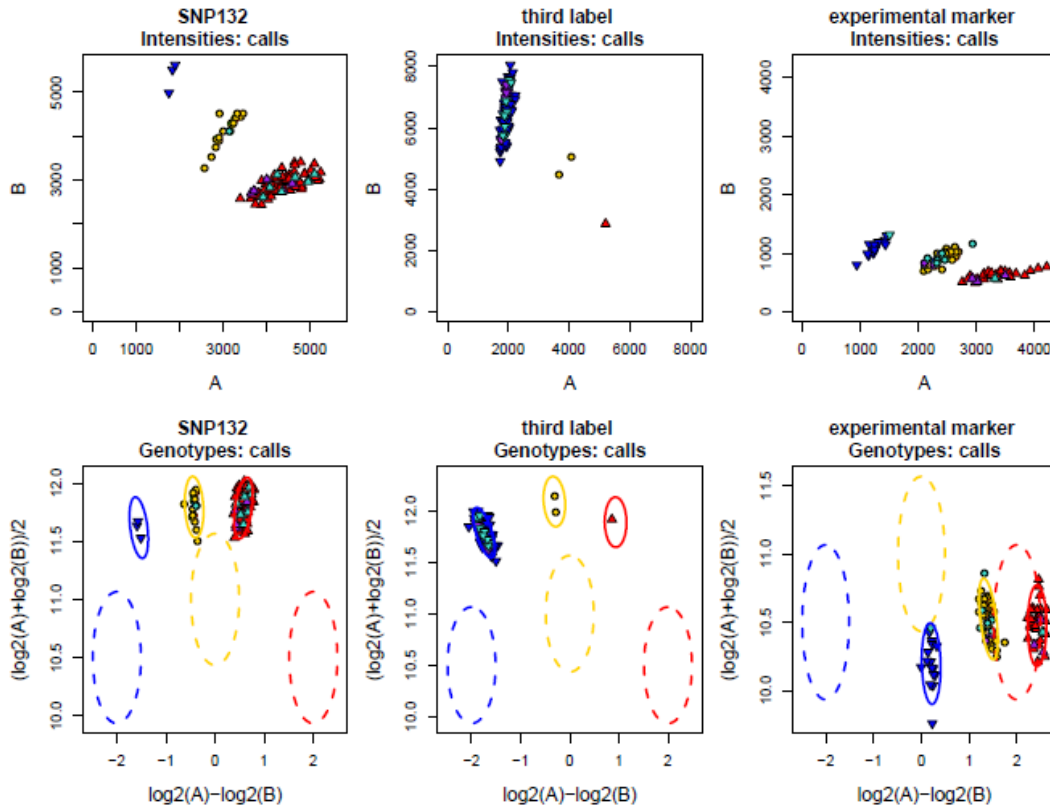


Figure 44: Updated color for the BB cluster, highlighted samples, and new labels for plots.

We can plot the same probesets and color the samples by confidence values instead of by genotype calls by changing the plot.calls argument to FALSE and the plot.conf argument to TRUE.

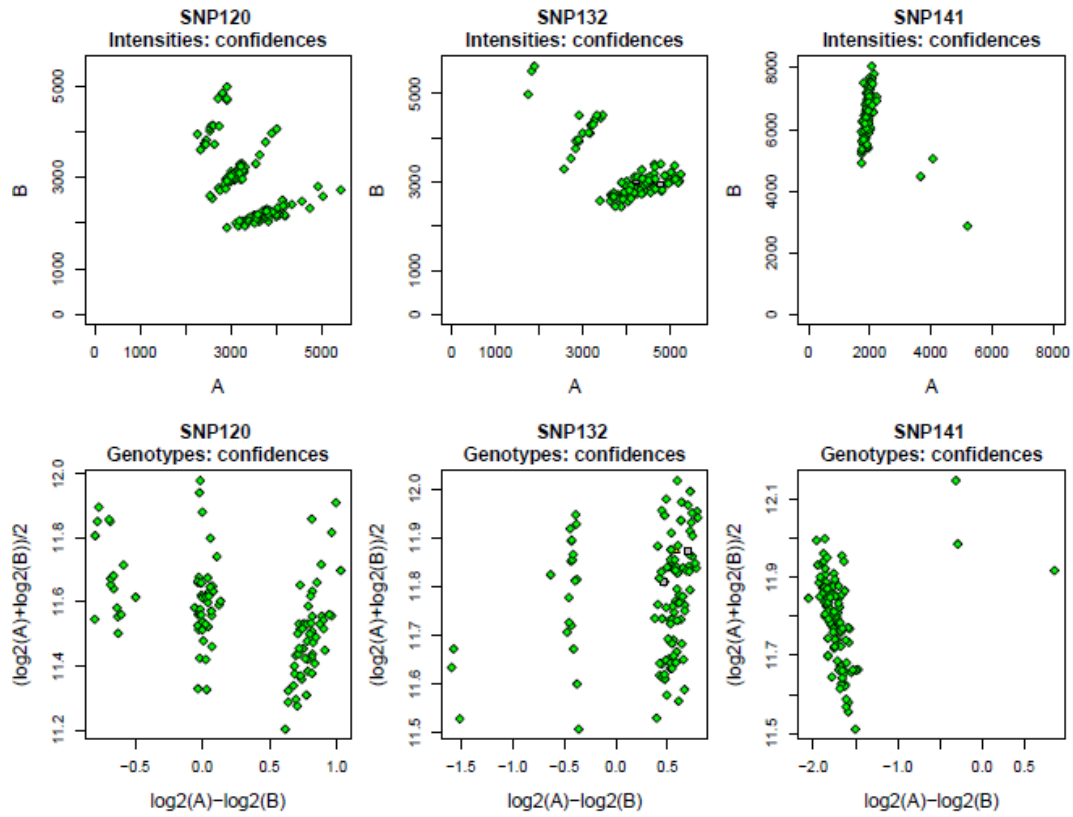


Figure 45: Standard output from *Ps_Visualization* for confidence plots.

With the new confidence cut-offs, we can see that the distribution of slightly larger confidence values are distributed evenly amongst the samples in each probeset, which indicates that there weren't any problems with cluster assignments for a particular genotype call.

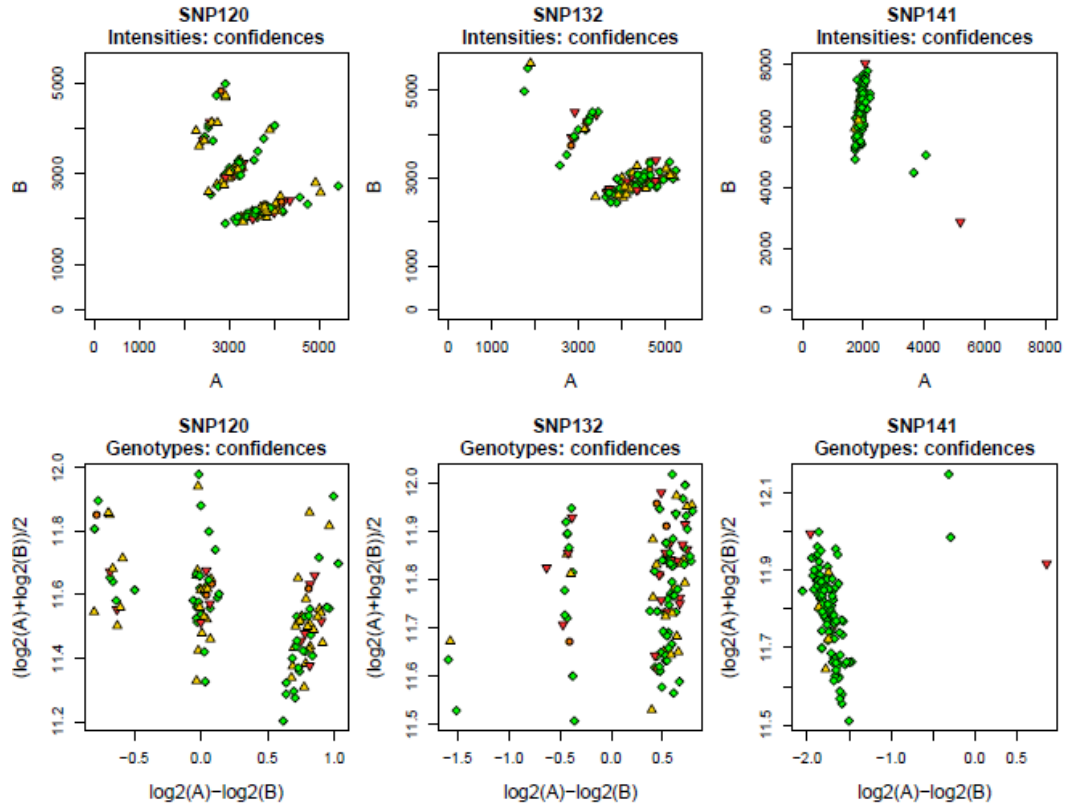


Figure 46: Confidence plots with updated confidence cut-off values.

If we are interested in comparing the confidences to their corresponding genotype calls, we can supply the calls, priors, and posteriors files and set plot.calls to TRUE.

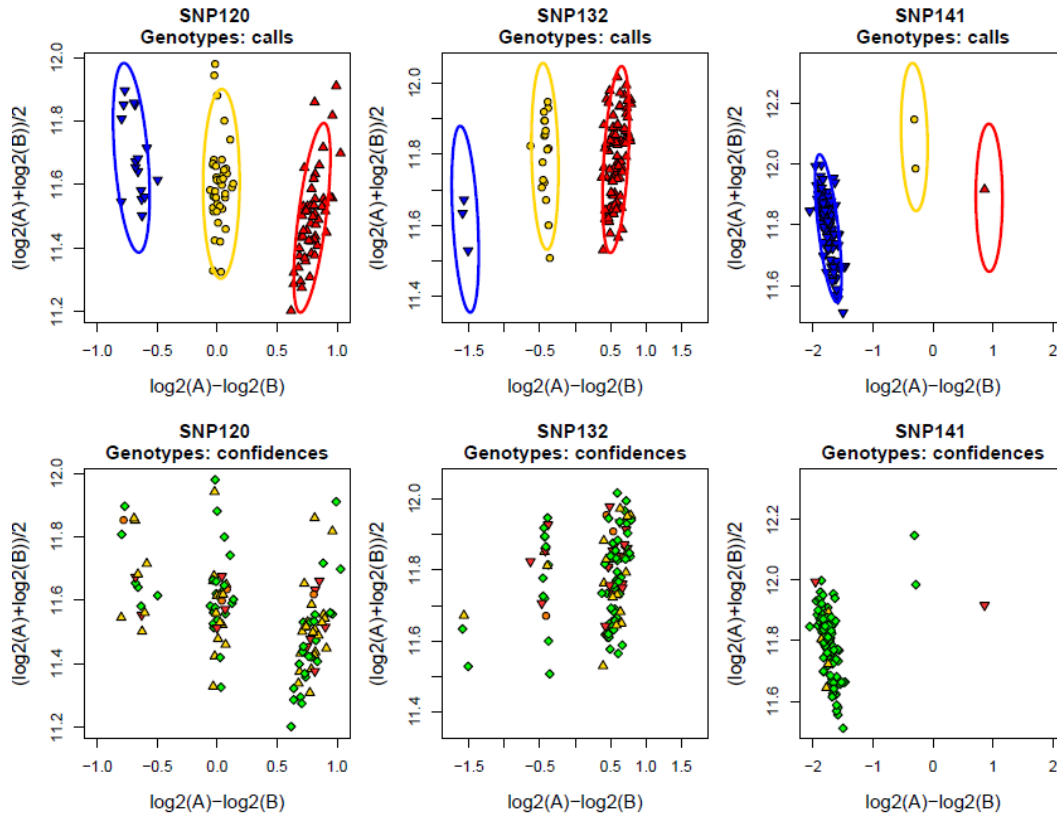


Figure 47: Calls and confidence plots with updated confidence cut-off values.

We can also plot the cluster densities for the SNPs. *Ps_Vis_Density* has nearly all of the same options as *Ps_Visualization*. *Ps_Vis_Density* makes plots of the genotype calls, densities, and reference calls. Any options that handle intensity plots are not available in *Ps_Vis_Density*.

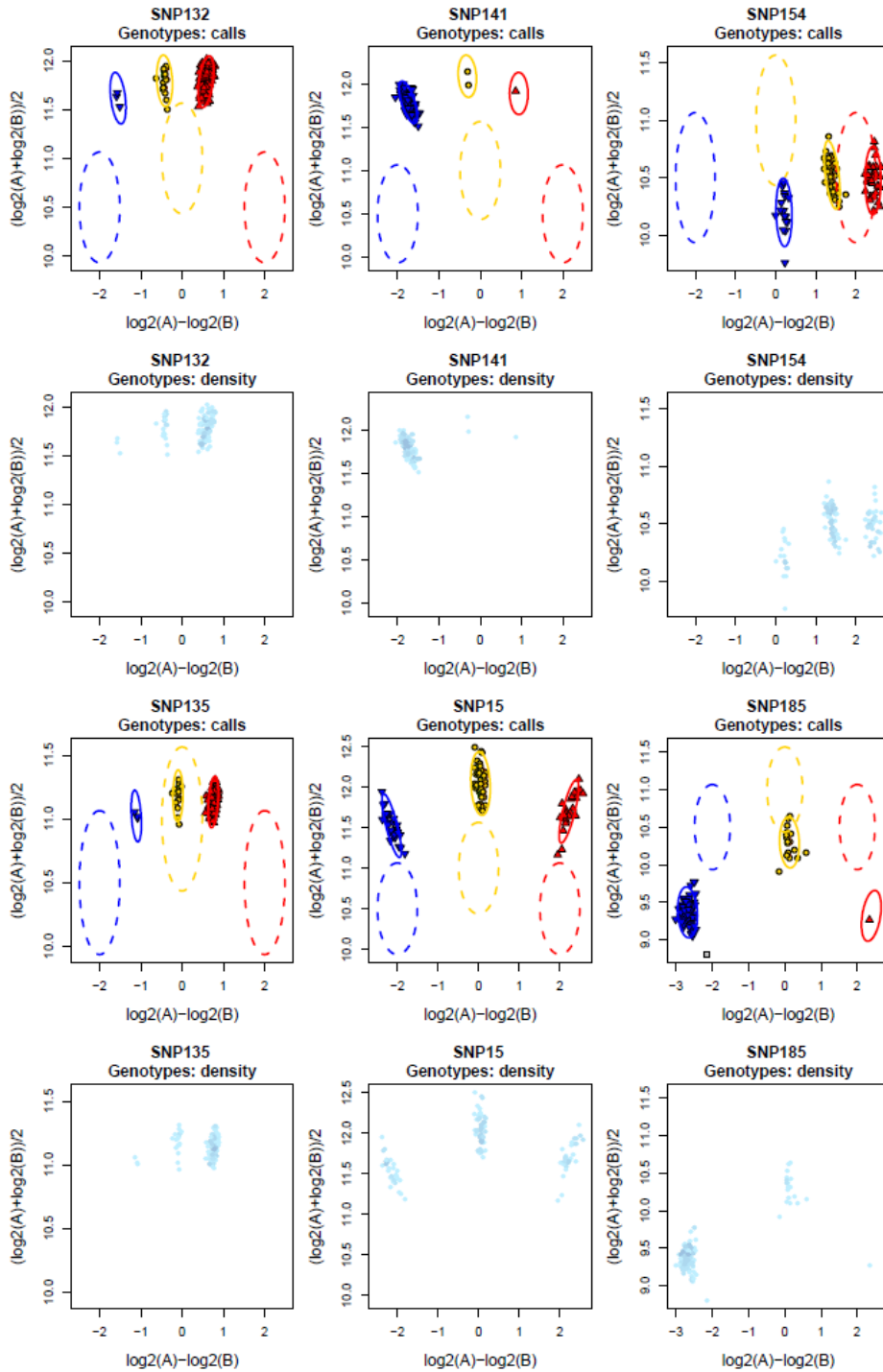


Figure 48: Standard output from *Ps_Vis_Density* using default values.

To have each plot adjusted for its clusters, we can change the options to not plot the priors and posteriors. We can also set the vertical.line option to TRUE just to demonstrate what it does.

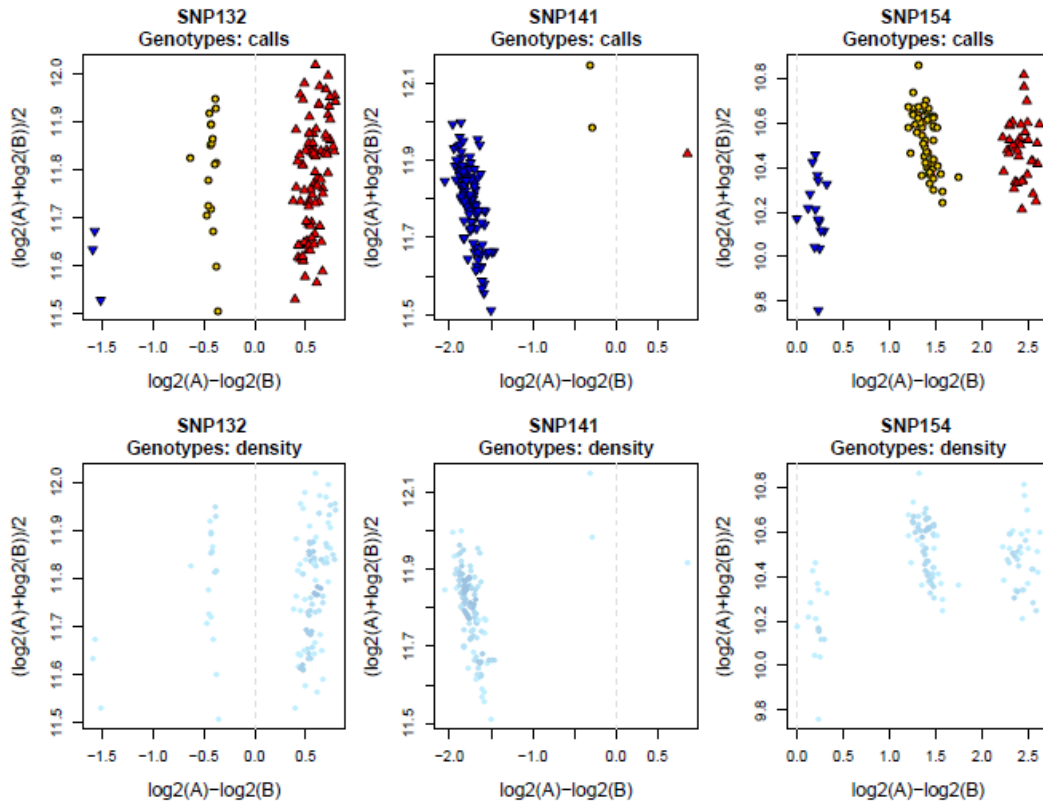


Figure 49: Output from *Ps_Vis_Density* without priors or posteriors.

We can set the densities to have fewer levels, which will produce sharper differences, and update the color to be green. The `plot.density`, `density.col`, and `density.levels` options are the only arguments that are available in *Ps_Vis_Density* and not *Ps_Visualization*.

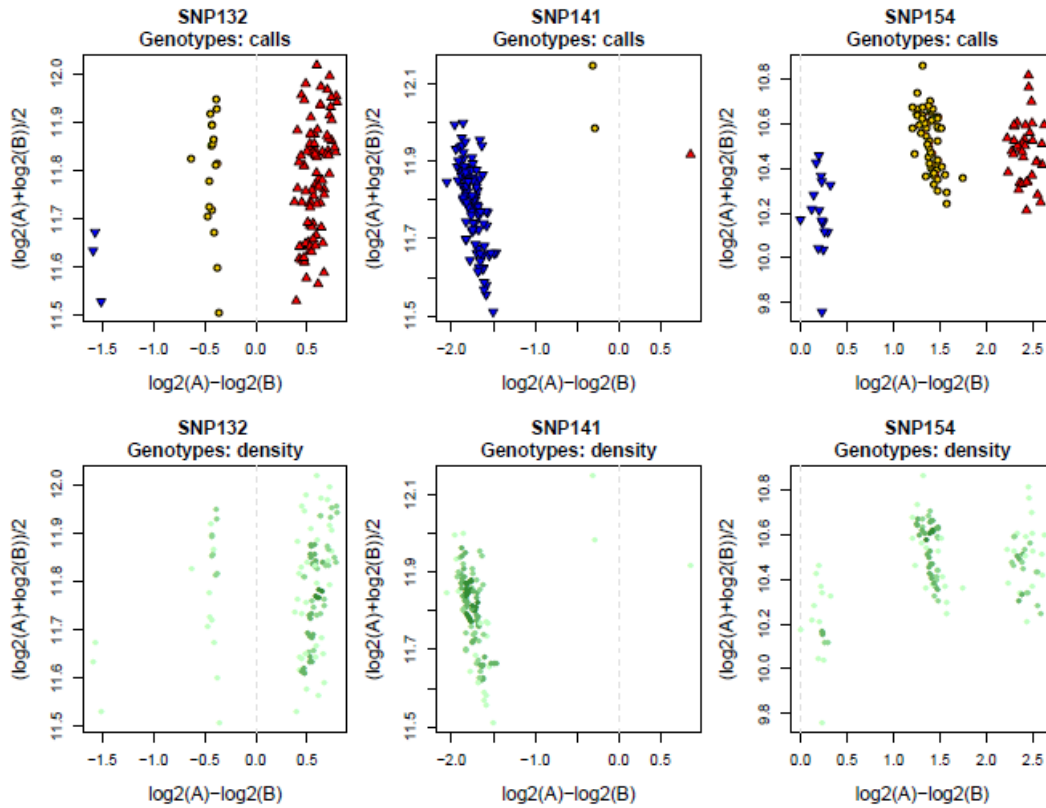


Figure 50: Output from *Ps_Vis_Density* with a smaller number of density levels and density color of green.

To plot the OTV SNPs, we can simply re-run *Ps_Visualization* using the OTV files instead of the APT Axiom files and compare the results. Plotting OTV results can help the user to decide if there truly is an OTV cluster or if something else may have happened with a SNP.

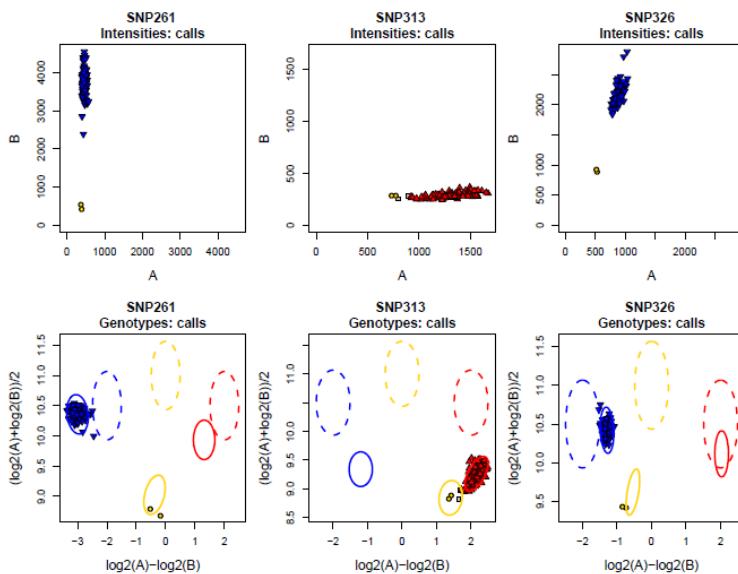


Figure 51: Output from *Ps_Visualization* with the original APT Axiom files.

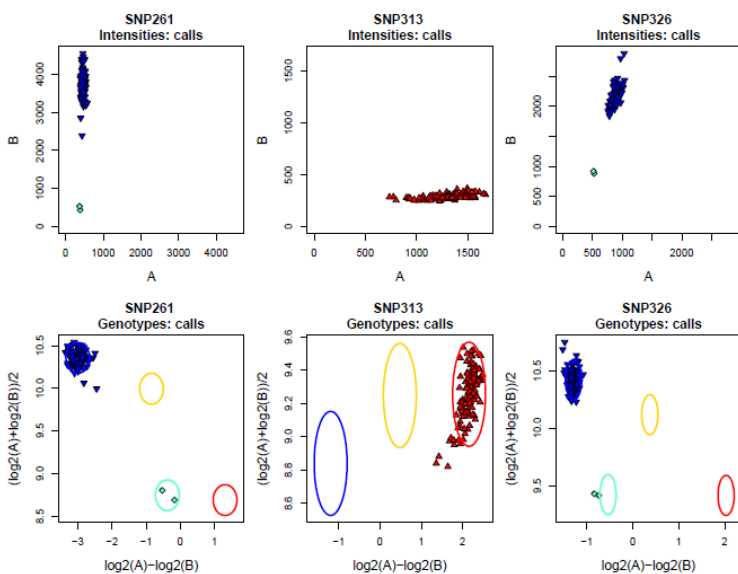


Figure 52: Output from *Ps_Visualization* with the OTV files.

5.3 Vignette: Wheat Data

This vignette uses an example data set called wheat, which has 798 probesets genotyped on 89 wheat samples. It contains 13 files: the PolyHighResolution, CallRateBelowThreshold and OffTargetVariant probeset list from *ps-classification*, and the summary, calls, confidences, and posteriors files, four files from *otv-caller*, and two files from *ps-call-adjust*.

Wheat is an allo-hexaploid species and this example will use the default values for polyploids. The Axiom files in this example do not have the APT header comments, so the assigned genotype call codes are missing from the calls file. All SNPolisher functions use the default call code assignments when the call codes are missing. In this example, the actual call code assignments match the default call assignments. Users should take care to double-check what the call assignments are if the call codes are missing from the calls file.

This example includes instructions on setting directories. The example code is more complicated than in the human vignette, so we strongly suggest trying that vignette first.

In this example, we will use the probeset list for the PolyHighResolution category: PolyHighResolution.ps. 299 of the 798 probesets were classified as PolyHighResolution. This example plots the first 6 SNPs.

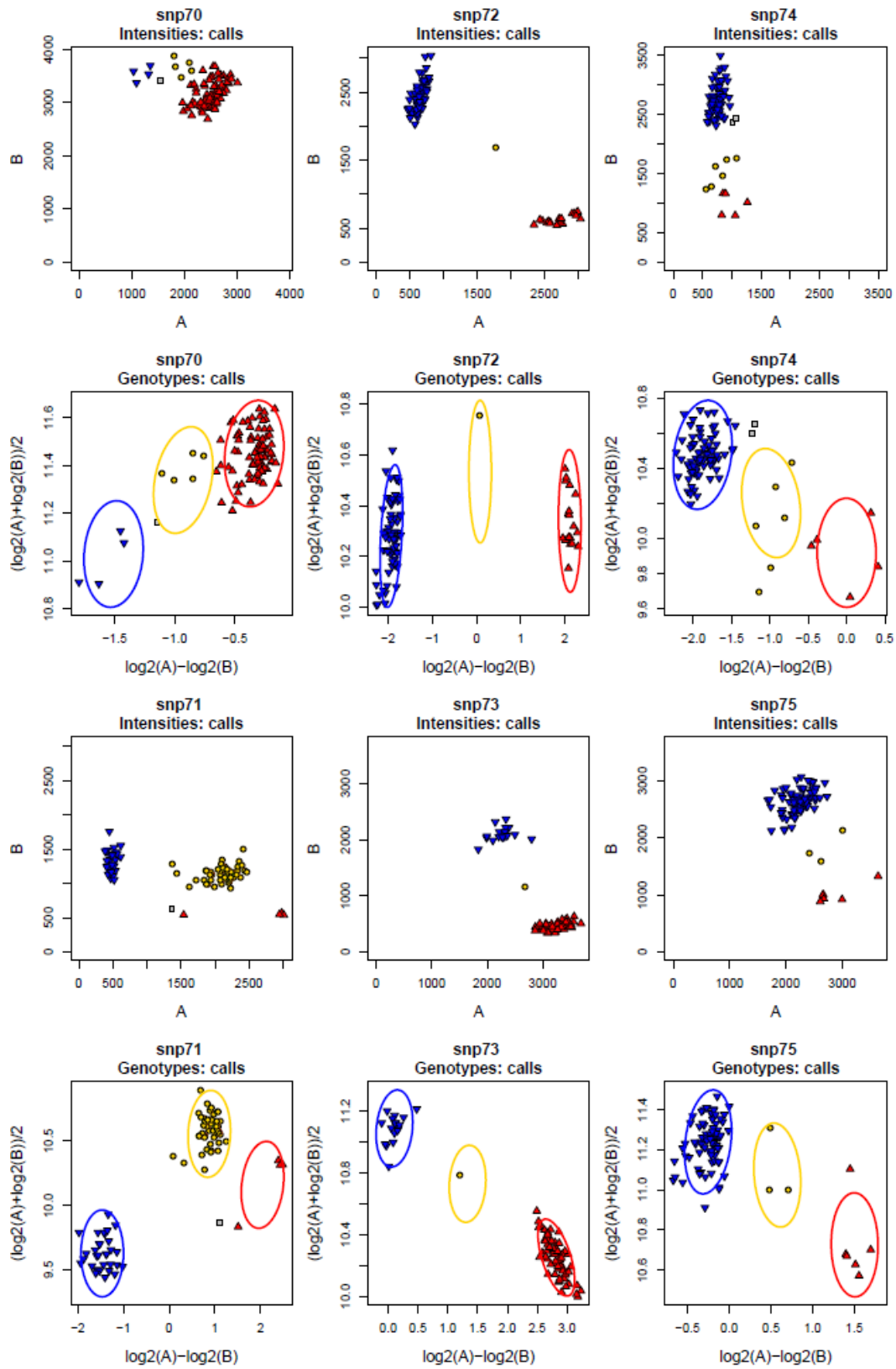


Figure 53: Genotype calls plots from *Ps_Visualization* for wheat.

The heterozygous (AB) cluster has very few samples and is hard to see as well, so we will make the color green.

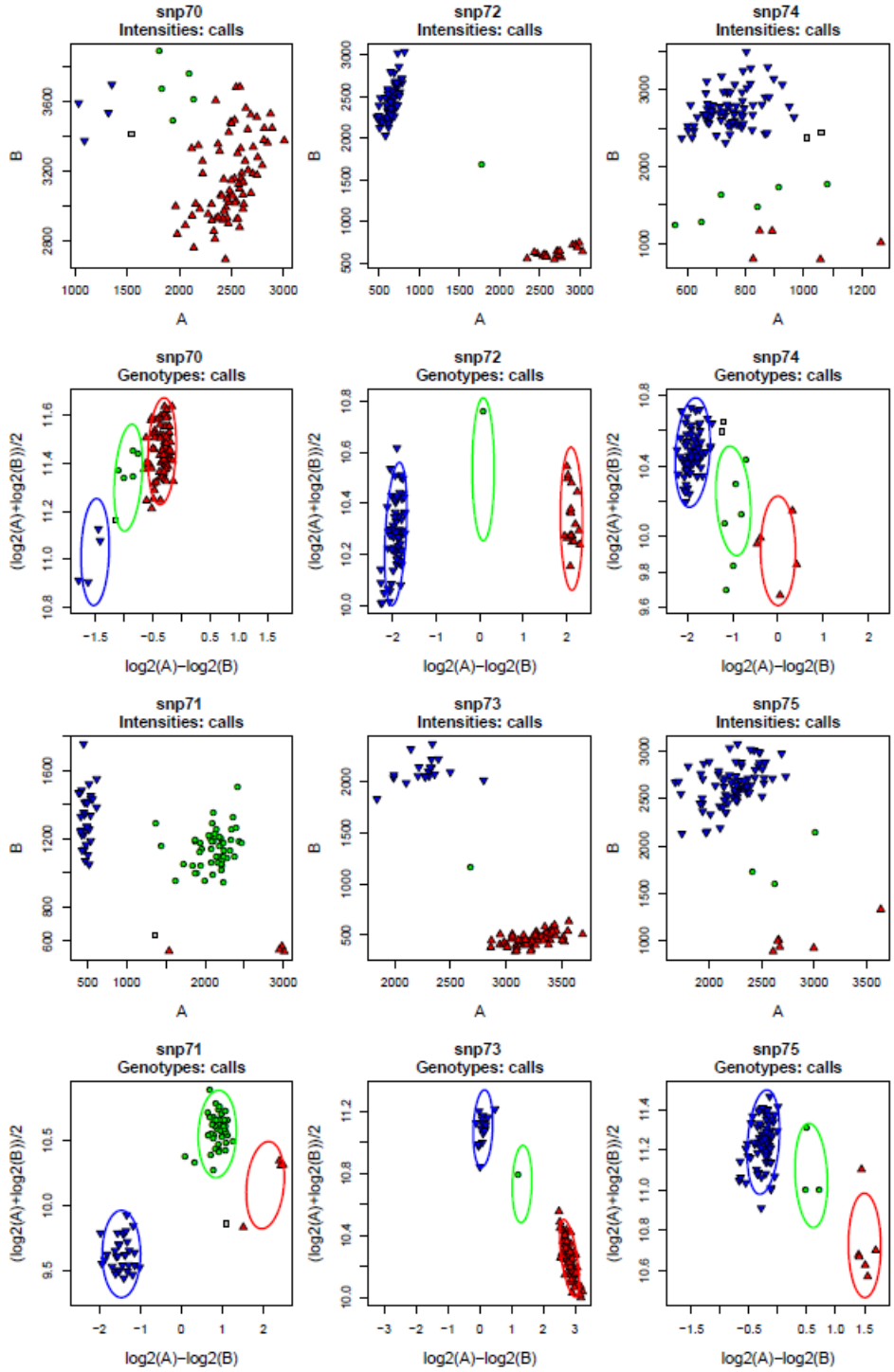


Figure 54: Genotype calls plots with AB set to green.

Because wheat is inbred, there should not be very many samples in the heterozygous cluster. The OTV plots show some samples that may be heterozygous but the intensities are too low. We should inspect these plots to confirm that they show true OTV clusters.

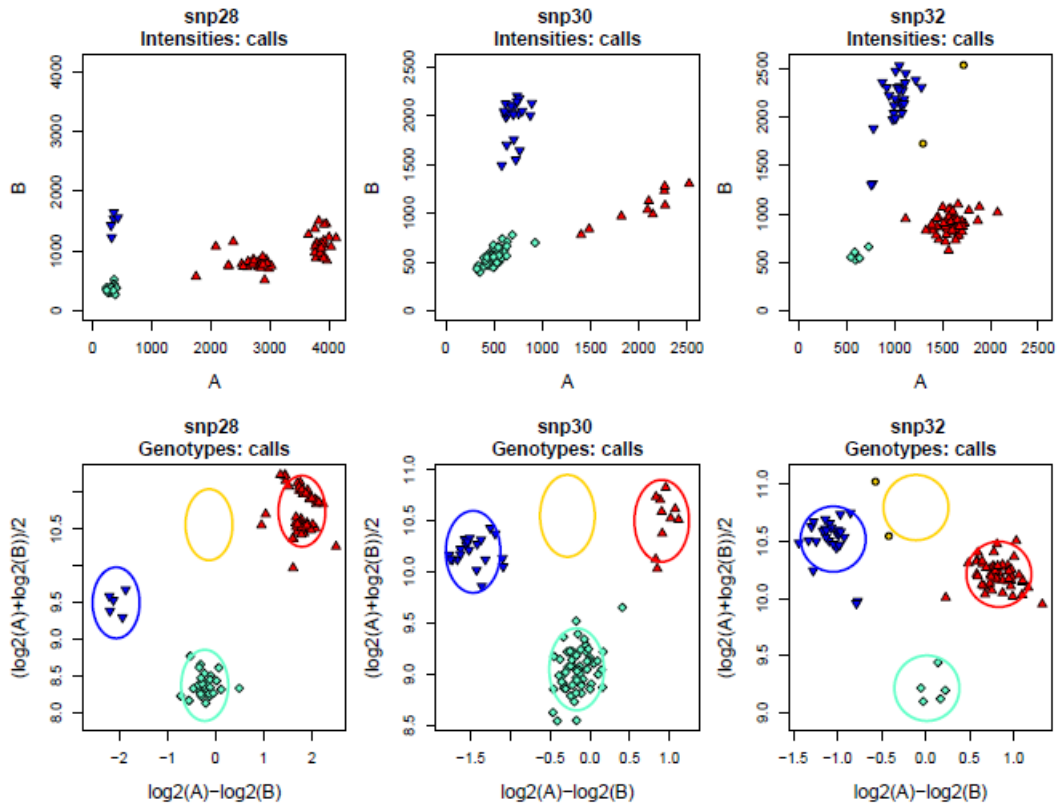


Figure 55: Plotted results from *OTV_Caller*.

Wheat is a polyploid species and inbred, so there may be a number of SNPs where some samples are close to the edge of a cluster while other samples are clearly inside of a cluster. *ps-call-adjust* can be used to manually reset the samples with a lower call rate to NoCall. In the CallRateBelowThreshold SNPs, there are 4 SNPs that may be interesting to run *ps-call-adjust* on. We can see in the confidences plots that the samples that are in the areas where the clusters overlap have confidences in the top quartile, i.e. larger than 0.02. Based on these results, a value of 0.01 or 0.02 as the confidence score threshold for adjusting the calls makes sense.

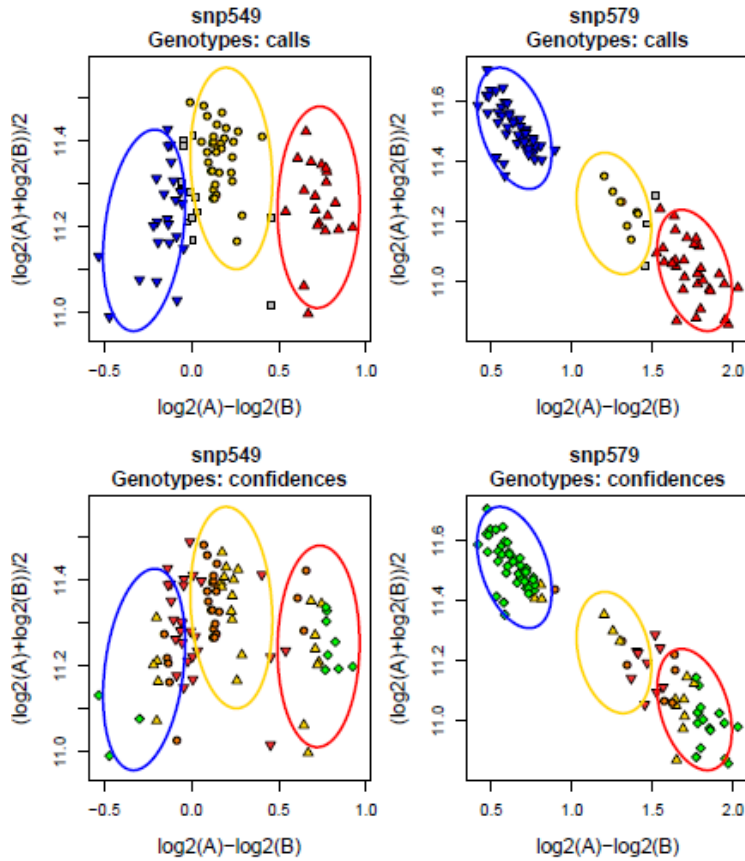
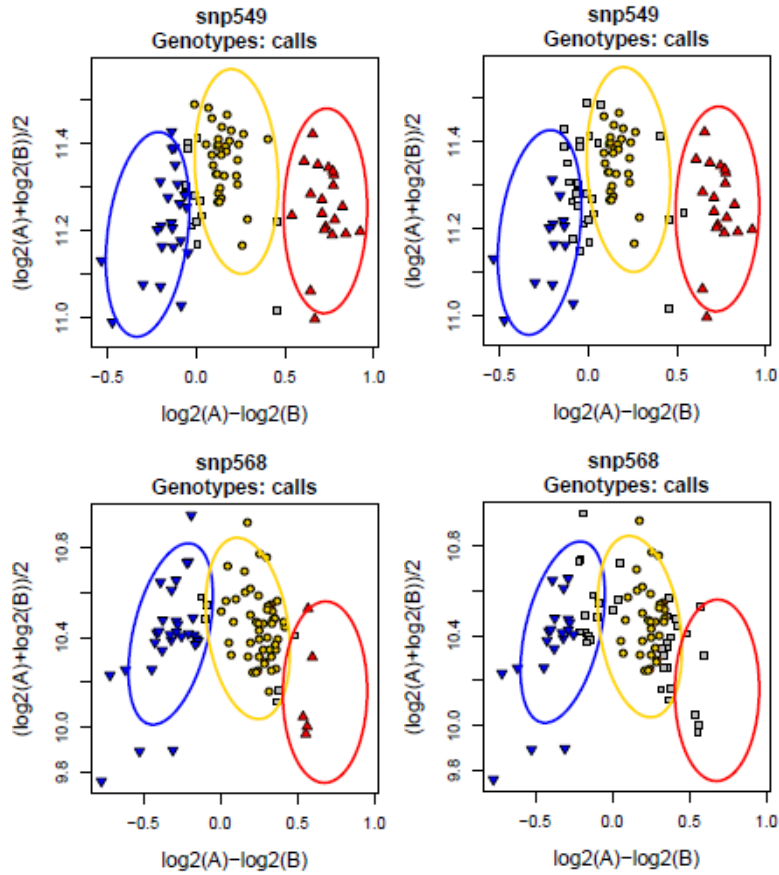


Figure 56: Calls and confidences plots for the *ps-call-adjust* SNPs.

Setting `confidences.cutoff` to 0.02 leaves some calls to close to different clusters but using 0.01 sets more calls to NoCall. This suggests running `ps-call-adjust` with a threshold value of 0.015. To compare the results, we can plot the old and new calls.



(a) CallRateBelowThreshold SNP with a `confidences.cutoff` value of 0.02. (b) CallRateBelowThreshold SNP with a `confidences.cutoff` value of 0.01.

Figure 57: Plotted results with `confidences.cutoff` values of 0.02 and 0.01.

5.4 Vignette: Case-Control Data

This vignette uses an example data set called `casecontrol`, which has 50 probesets genotyped on 999 samples. It contains 11 files: the summary, calls, confidences, and posteriors files for the case samples only and for the cases and control samples, the pid probeset list and sample highlight list, and the results of running `ps-bac` on the data.

500 of the samples were genotyped in 1 batch (controls), and the remaining 499 samples were genotyped in 5 smaller batches (cases). We need to see if there is a difference in B-allele intensities across the samples. The function `ps-bac` compares allele frequencies across batches and tests if the batch allele frequencies are consistent. We ran `ps-bac` on the case and case-control data sets with the default `-bac-intercept` value of 0.02. 26 probesets failed and 10 probesets passed the `ps-bac` frequency test. We can plot the markers to see that the cases samples have been shifted in the markers that failed. The plots will help us decide if there truly is a difference between the batches.

`Ps_Visualization` can take an optional sample file that contains the names of the samples to be highlighted and the color that those samples should be plotted with. The sample file in the `casecontrol` data set contains 100 of the case samples and the color green for each of those samples. Supplying the sample file will produce plots with the 100 case samples plotted in green.

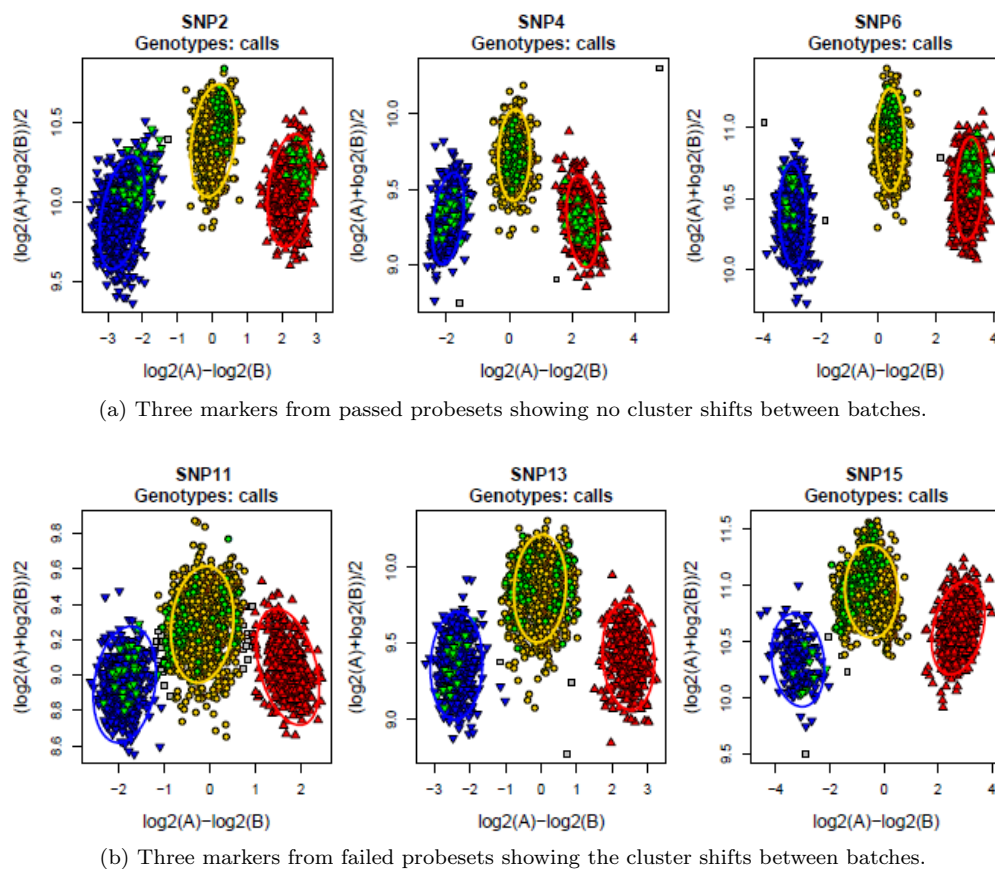


Figure 58: Markers with B-allele shifts between batches and markers with no shifts between batches.

In figure 58b, the highlighted cases samples display how the AB and AA clusters are shifted to the left compared to the rest of the samples. Specifically, the AA cluster of the highlighted samples sits in the AB cluster of the rest of the samples and is incorrectly genotyped as AB when combined with the rest of the samples. In contrast, the markers in 58a show no signs of shifting. The three highlighted clusters have the same placement as the three clusters of the rest of the samples. When all samples are combined, there is no incorrect genotyping due to batch effects.

5.5 Vignette: Batch Data

Batch plotting can be very useful for investigating the genotyping output from SNPs across multiple batches of samples. *Ps_Visualization*, *Ps_Vis_Density*, and *Ps_Vis_Multi_ID* will produce batch plots when the inputs are a list of files instead of only one file. Although this feature is helpful, the user must take care to correctly set up the lists of input files. This vignette uses an example data set which has 8 probesets genotyped on 362 diploid samples. It contains 6 files for 8 batches: the summary, calls, posteriors, and multiallelic posteriors files per batch, the samples and labels file per batch, and a probeset list ps file.

The default plotting values in *Ps_Visualization* are to plot the intensity and genotype plots, and to plot the posteriors but not the priors. If a posteriors file is not provided, default posterior values will be used with the MvA or log2 transformations. All other transformations do not use a default value and posteriors will not be plotted without a posteriors file.

The user does not need to specify if batch plots should be produced. *Ps_Visualization* uses the number of supplied files to determine if batch plots will be made or not. The old argument *batch* has been deprecated. Any input file that can be different between batches must have vectors of the same lengths as inputs (summary, calls, confidences, posteriors, priors, report) but files that do not change between batches should only have one file as an input (special SNP file, etc.). There must be one temporary directory per batch.

This data set has 8 batches. We can use `paste` to keep the command shorter. This approach works well when the different file types have similar names across all batches, e.g. the calls files have names that are variations on “calls”. Another organizational scheme is to have one folder per batch, and have files with the same names in each folder. For example, the calls file in each folder could be named “AxiomGT1.calls.txt”. In this case, setting up a list of the directories and using that list for all of the file names is the easiest approach. See 5.3 for an example of using directory names.

The number of plots produced per probeset is different for biallelic and multiallelic probesets. The default number of plots for a biallelic probeset is 2 (intensity and genotype calls) while the default number of plots for a multiallelic probeset is 3 (intensity, calls, and PCA). Multiallelic probesets have an additional plot that takes up one more column, the legend plots. The color and shape assignments for a probeset are the same across all batches, so there is only one legend plot regardless of how many batches there are.

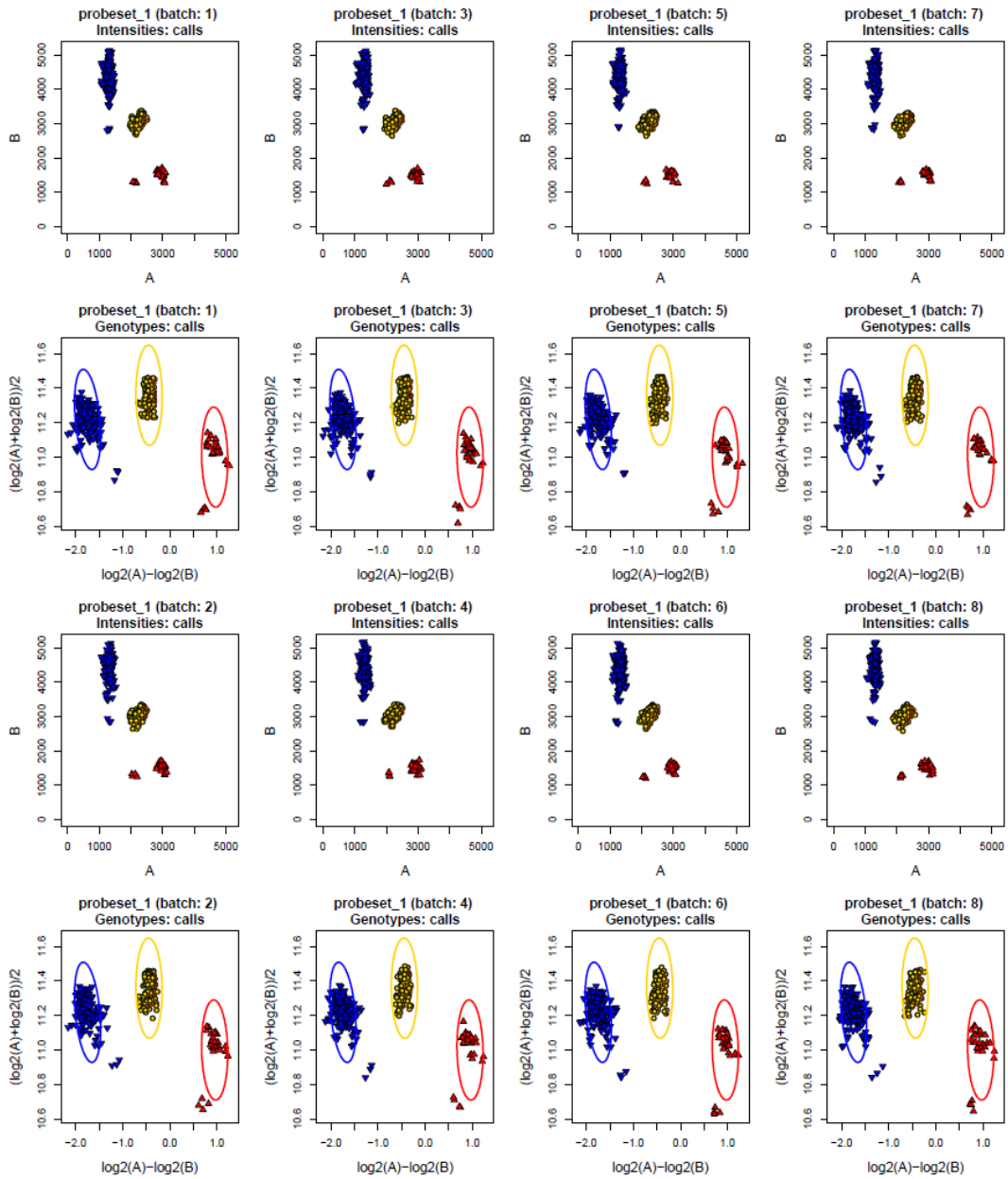


Figure 59: Standard output from *Ps_Visualization* using all default values for a biallelic probeset genotyped across 8 batches.

With 8 batches, multiallelic SNPs have their plots split across two pages. We can adjust the number of columns to have all of the plots on one page. Multiallelic SNPs have one more plot: the legend. To get all of the plots on one page, the number of columns needs to be nine, not eight.

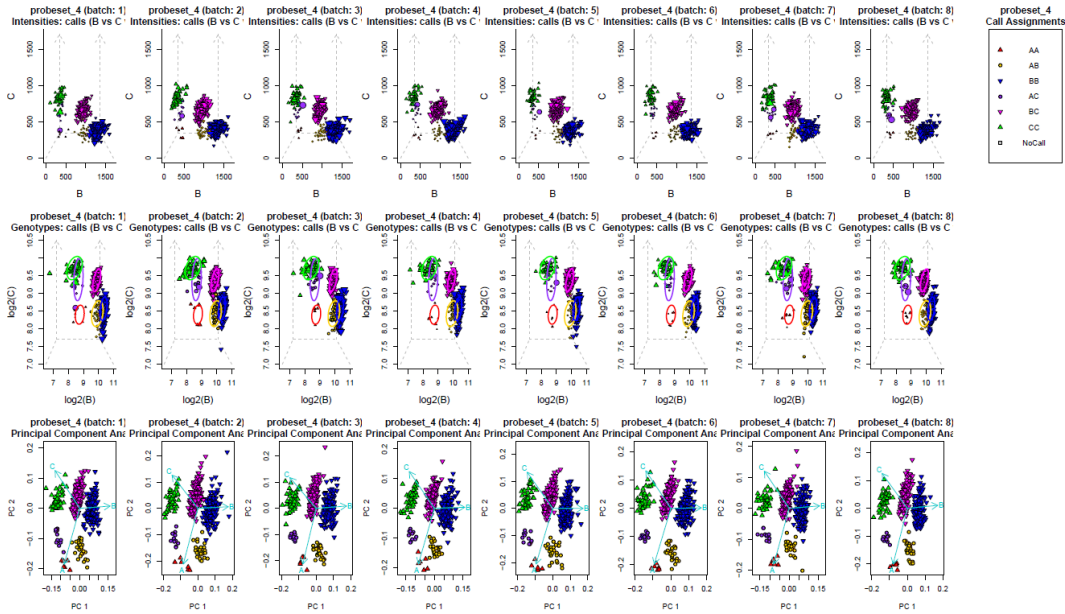


Figure 60: A multiallelic probeset genotyped across 8 batches with 9 columns.

This number of columns does not work well with the biallelic probesets in our data set. The plots are difficult to see and there is a lot of blank space on the page.

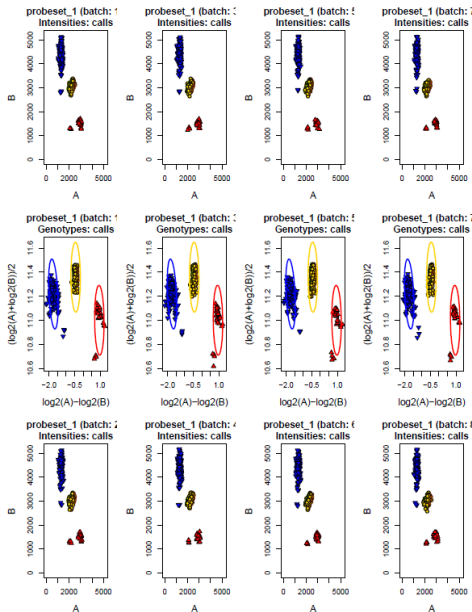


Figure 61: A biallelic probeset genotyped across 8 batches with 9 columns.

Although all of the plots for multiallelic SNPs are on one page and its easier to compare across all of the batches, the plots are pretty squished. We can adjust the number of columns to split the multiallelic plots equally onto two pages which will produce more of the multiallelic plots on the second page and keep the plots from being squished together. We also want to set the num.cols argument to 5 and not 4 for the same reason that we needed to set it to 9 instead of 8: there's one more plot with multiallelic SNPs, the legend plot.

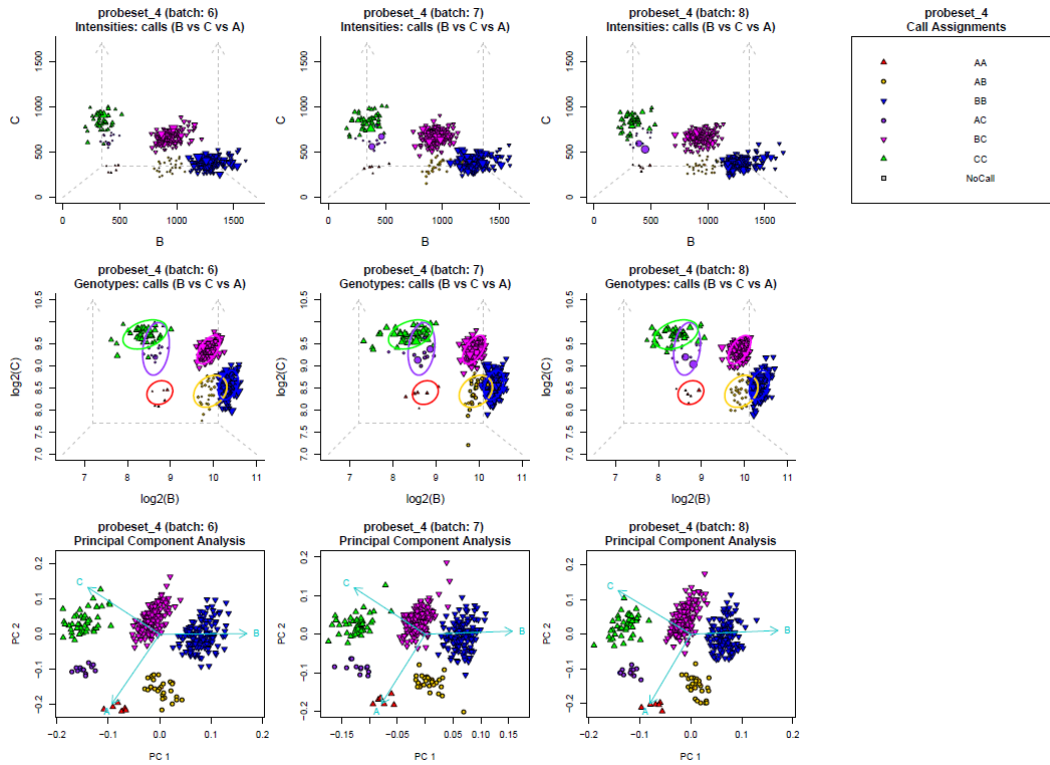


Figure 62: The second page of plots for a multiallelic probeset genotyped across 8 batches with 5 columns.

This layout works much better for the biallelic probesets as well.

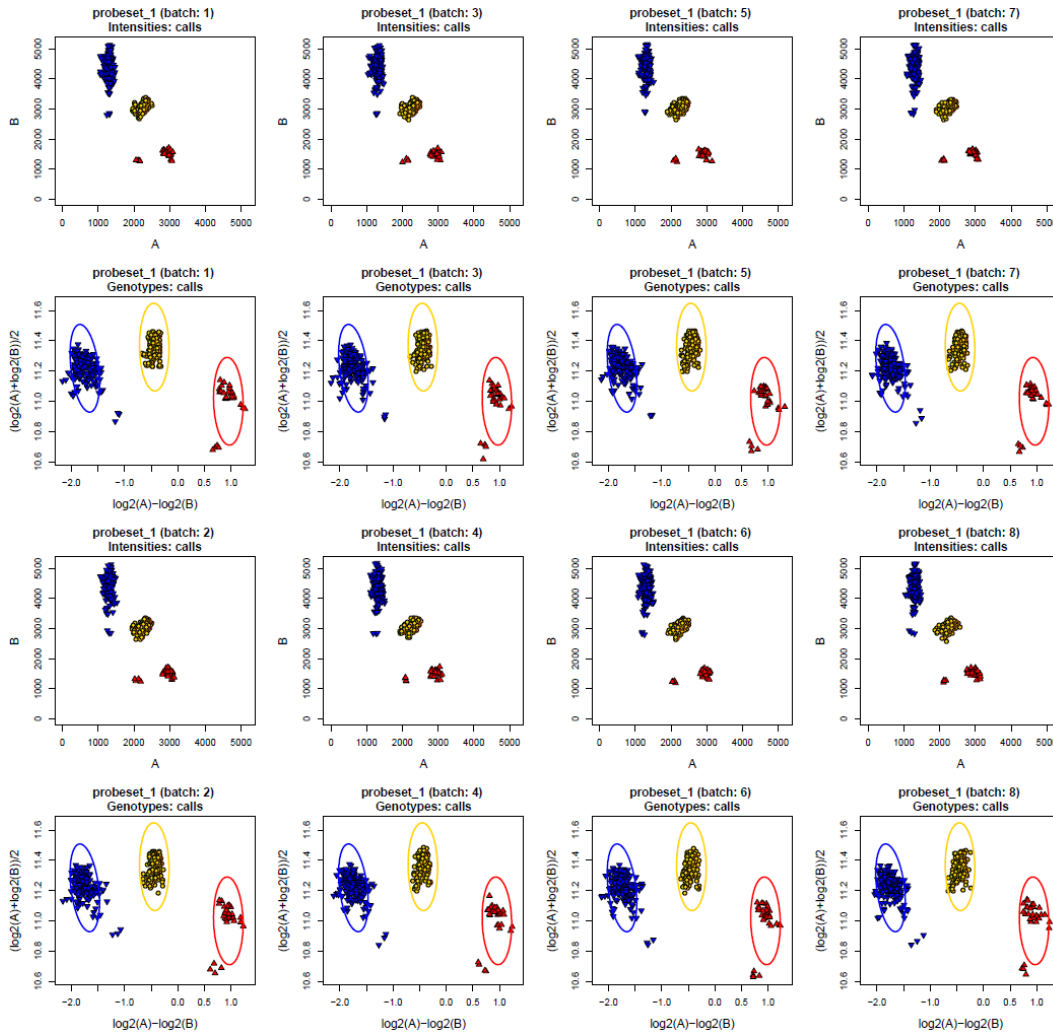


Figure 63: A biallelic probeset genotyped across 8 batches with 5 columns.

Let's say that we have a set of samples and we want to see if their behavior changes from batch to batch. We can provide sample files with the sample names and the colors we want those samples to be highlighted in, one file per batch. We are using the color "deeppink" as the highlighting color.

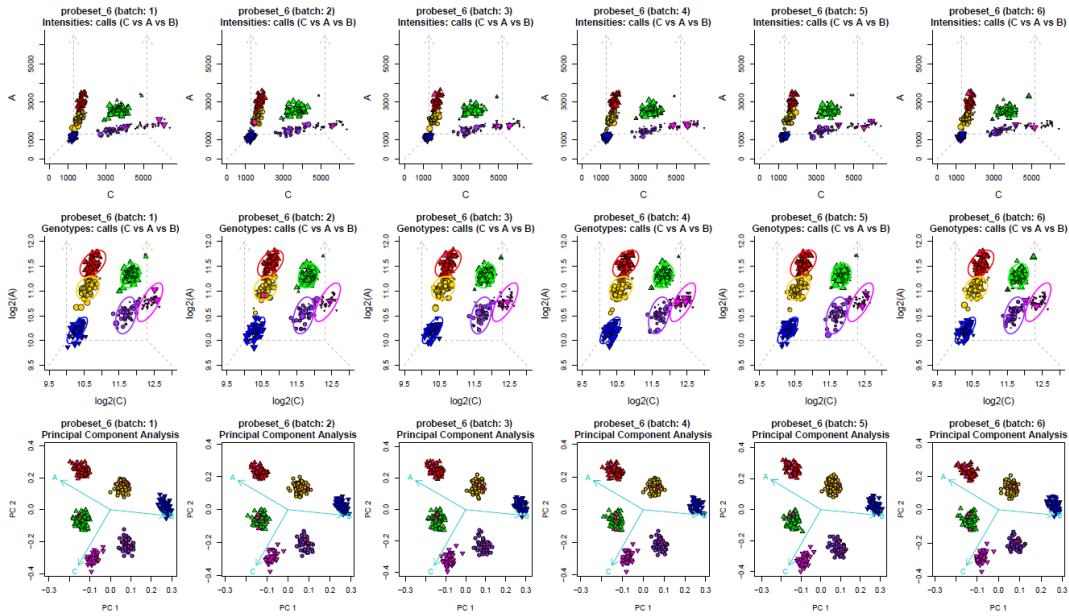


Figure 64: A multiallelic probeset with samples highlighted in deeppink.

If we don't want to highlight samples in all of the batches, we can supply NA instead of a file name for those batches. Let's say that we only want to highlight samples in batches 1, 2, 6, 7, and 8 and skip highlighting samples in batches 3, 4, and 5.

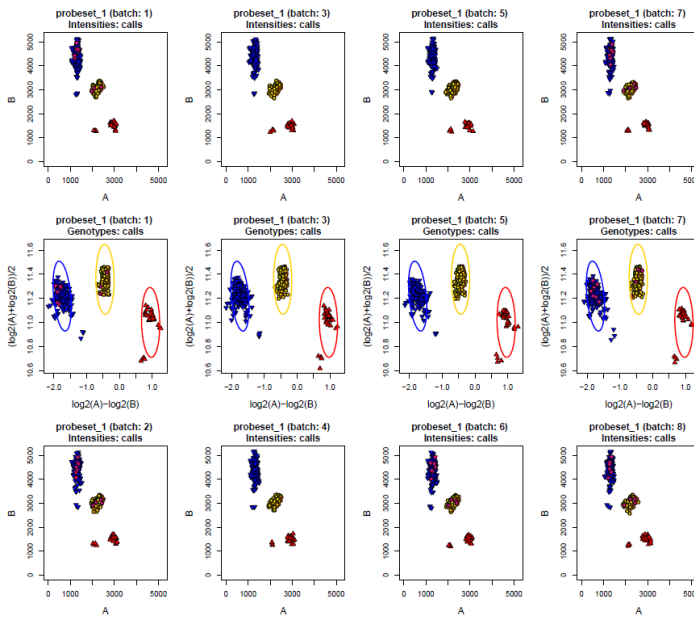


Figure 65: A biallelic probeset with samples highlighted in batches 1, 2, 6, 7, and 8.

If you want the main titles on the plots to have different labels than the probeset names, you can supply a labels file with the new titles. The labelsFile argument should be the same length as the number of batches. This lets us change the same probeset name to different labels for different batches if we want to. Let's say that something went wrong with the sample collection in batches 3 and 4, and somebody forgot to mark if the samples were buccal swabs or sweat. For those two batches, we can supply different labels ("unknown").

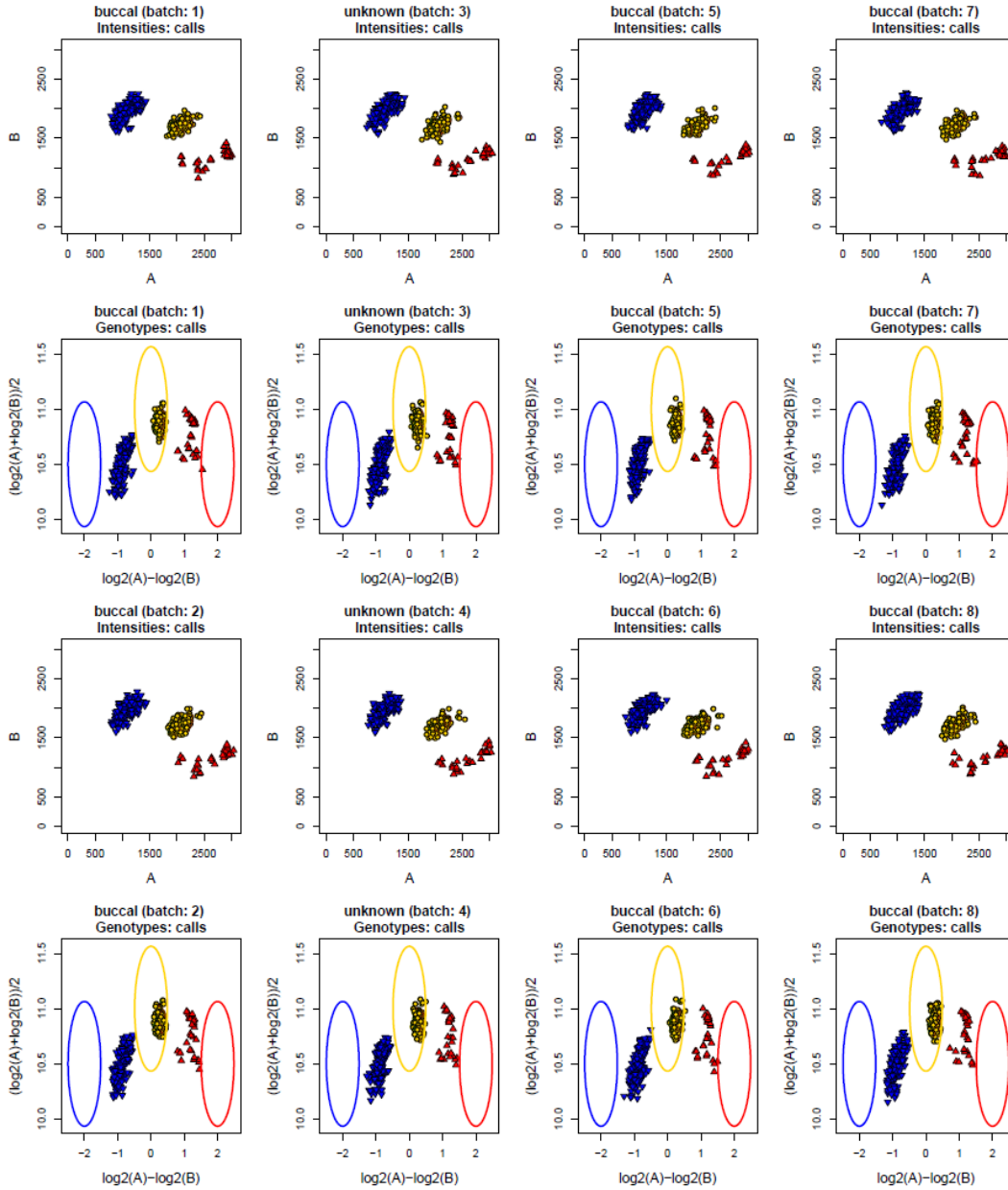


Figure 66: A biallelic probeset with different labels for different batches.

5.6 Vignette: Multiallelic Data

This vignette uses an example data set called `multi.snp`, which contains six files: the `ps2snp` file, `summary`, `calls`, `posteriors`, `multiallelic posteriors`, and `labels` files. These files are used to group the probesets by SNP and create plots. The `ps2snp` file used in `Ps_Vis_Multi_ID` must contain four columns named `probeset_id`, `snpid`, `multi_snp_id`, and `alleles`. This file is a library file. If you have a version with two columns instead of four, please contact your FAS or FBS.

This example produces plots of human data using most of the default options for `Ps_Vis_Multi_ID`: intensity and genotype plots with posteriors. There are biallelic and multiallelic probesets, and `Ps_Vis_Multi_ID` plots the biallelic probesets before the multiallelic probesets for each SNP. If there are only biallelic probesets or only multiallelic probesets, they are plotted in the order that they appear in the `ps2snp` file. The color and shape assignments are the same for all probesets in one SNP, and there is a legend plot at the end with a table of call assignments.

Similarly to `Ps_Visualization`, if a multiallelic probeset has only two unique alleles in the calls, the PCA plots are not produced. In addition to the standard plot titles produced by `Ps_Visualization`, probeset plots from `Ps_Vis_Multi_ID` include a third line that lists the alleles interrogated by the probeset. The first SNP, SNP 1, fills up all 6 columns on the first page with probeset plots, and the legend plot for SNP 1 is plotted all by itself on the second page. We can set the number of columns per page to be 7 instead of 6.

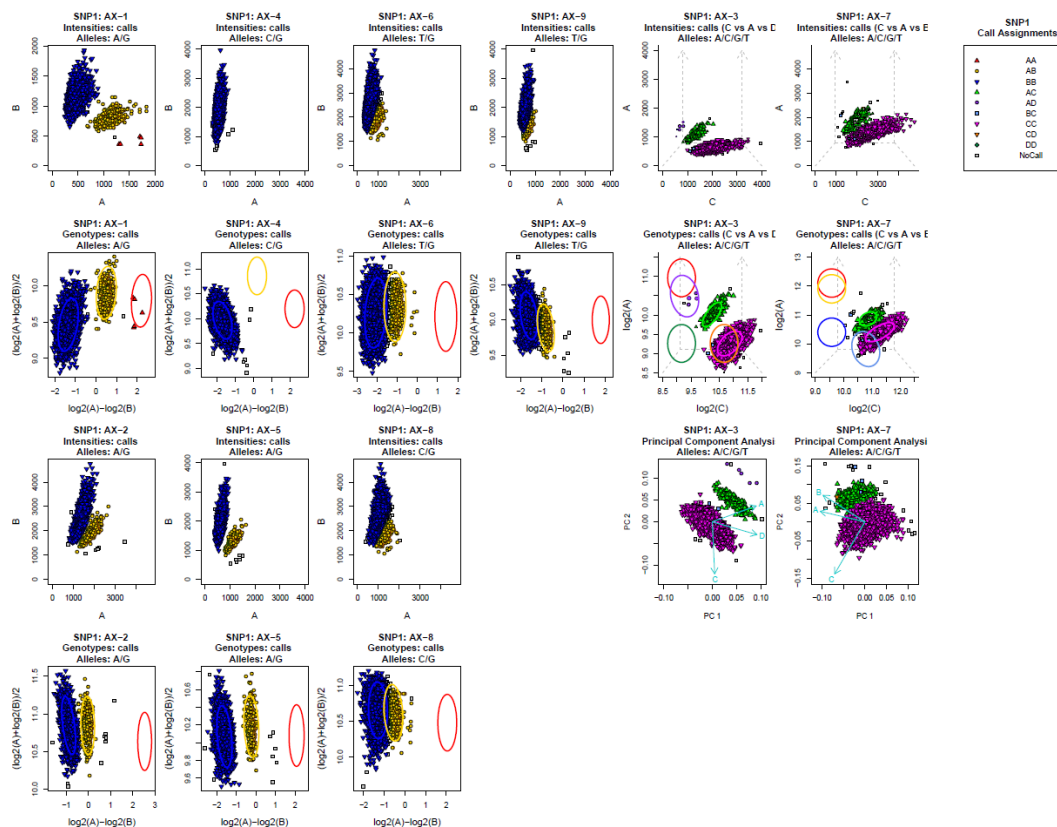


Figure 67: A multiallelic SNP made up of 9 biallelic and multiallelic probesets.

We may want to label the individual probesets contributing to a SNP with a more informative title than the probeset name, or we may want to change the SNP name itself. We can do both of these things using the `labels` file. The `labels` file should have two columns named `probeset_id` and `label`. Either probeset names or SNP names can be used in the `probeset_id` column. The desired titles for the plots should be in the `label` column.

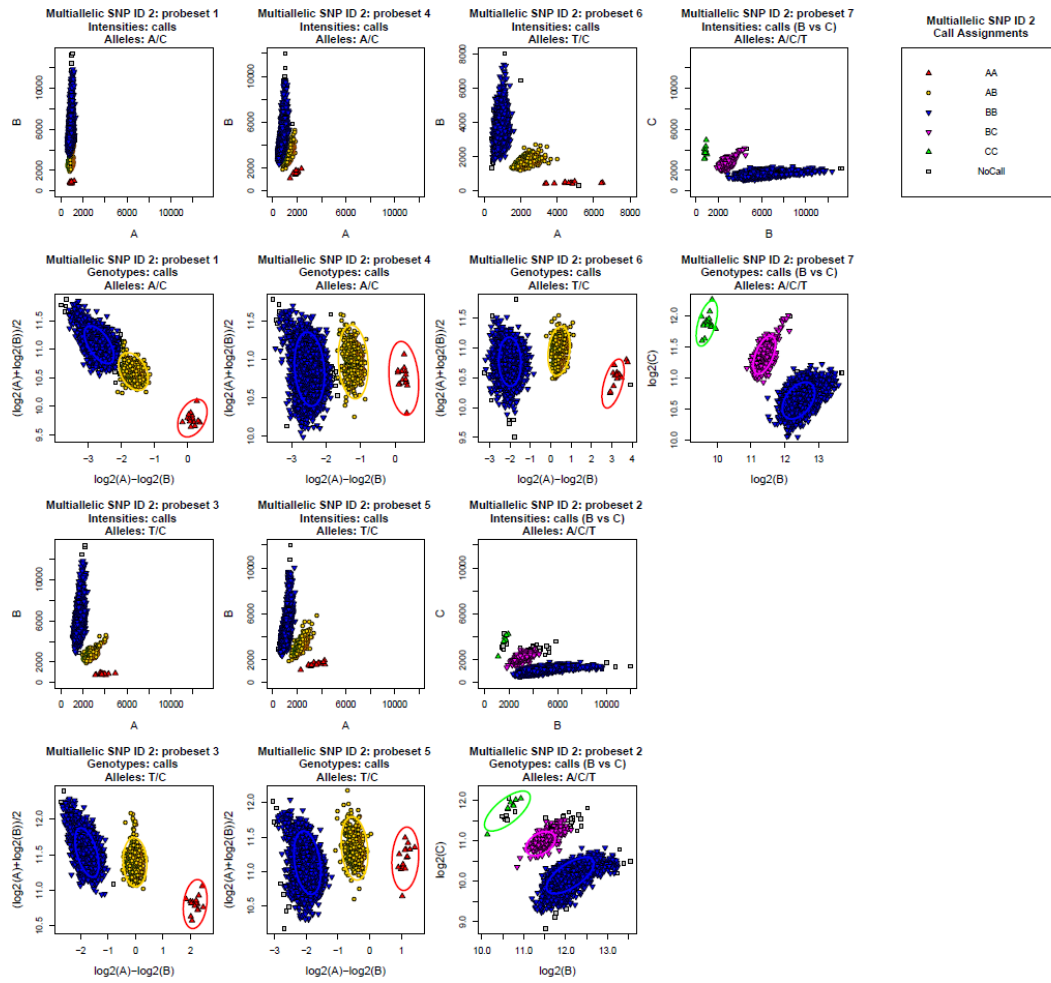


Figure 68: A multi-allelic SNP and its probesets with new titles.

5.7 Vignette: Copy Number Data

Axiom Analysis Suite (AxAS) and Array Power Tools (APT) software enable copy number variation analysis in fixed genomic regions of interest. *CN_Visualization* produces different types of plots to visually inspect these regions and display copy number changes. Four files are required for running *CN_Visualization*: a region calls file, a region details file, a priors file, and a posteriors file. This vignette uses an example data set which contains five files: the four required files and a truth file. The truth file is used to compare the assigned copy number to the expected copy number.

If the user provides a file name, all plots will be written to that one file. The default behavior of *CN_Visualization* when a file name is missing is to create three separate files, one for each type of plot. The files names are *CN_Batch_Histogram.pdf*, *CN_Heatmap.pdf*, and *CN_Plate_Histogram.pdf* (or *CN_Plate_Histogram_w_Truth.pdf*). When the plate effects argument is set to `TRUE`, a file named *CN_Plate_Effects.pdf* is created as well. In this example, we won't provide a file name and we will end up with three files. When an output directory is not supplied, *CN_Visualization* creates a directory named *Graph_Outputs* in the working directory and writes all plots to that folder.

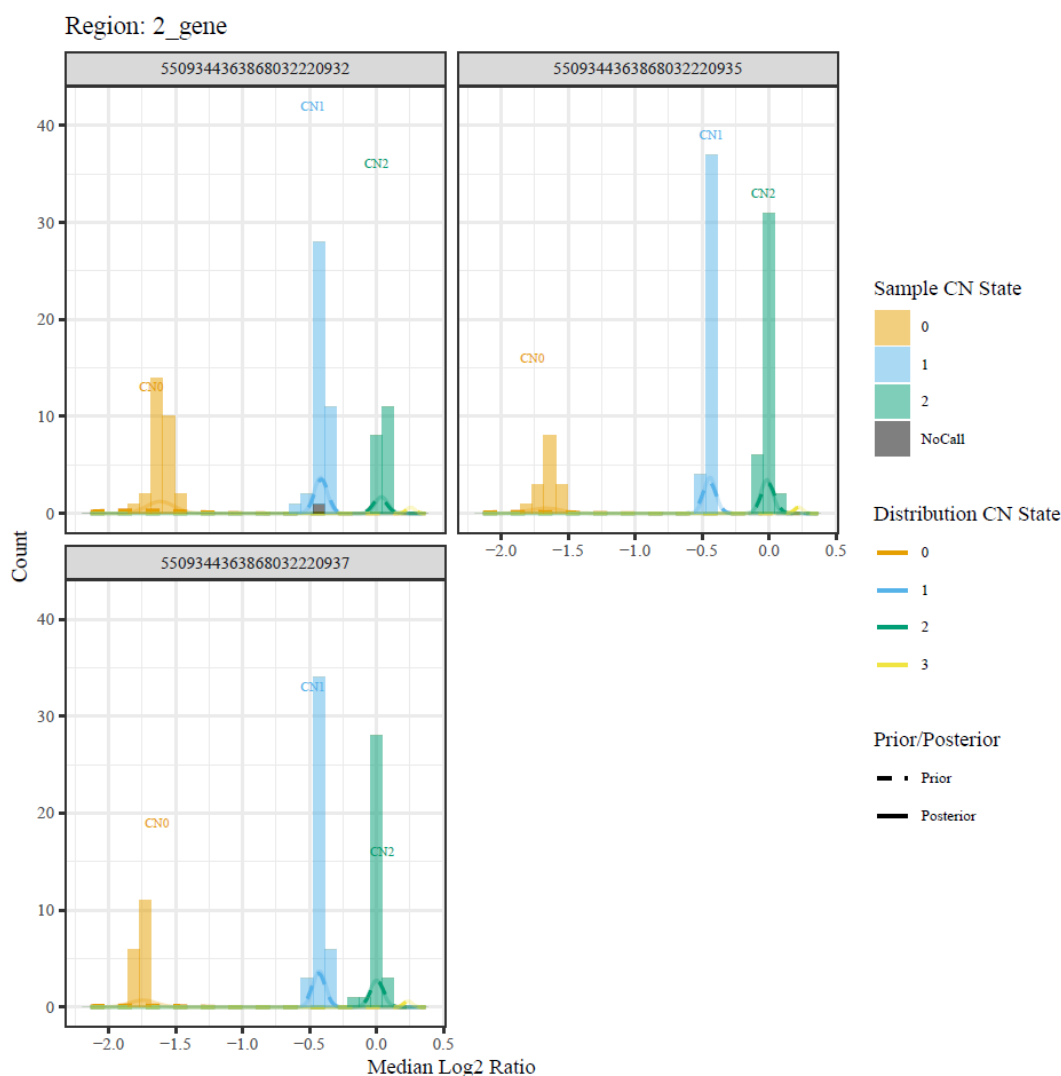


Figure 69: Plate histogram for a CN region.

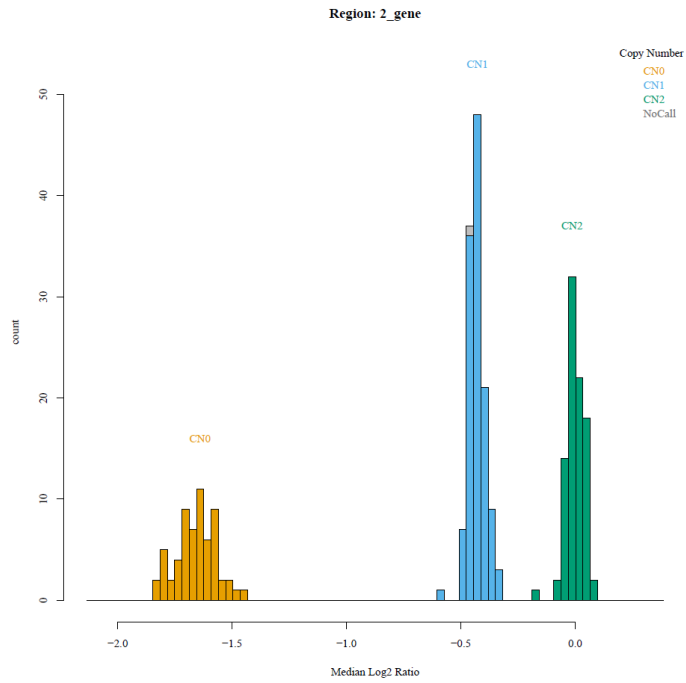


Figure 70: Batch histogram for a CN region.

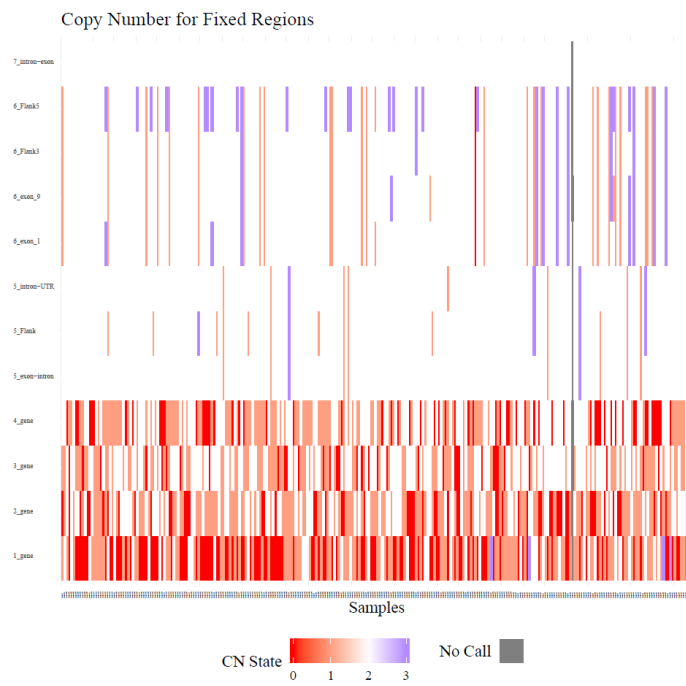


Figure 71: Heatmap of all CN regions.

5.8 Vignette: Autotetraploid Data

This vignette uses an example data set which has 57 autotetraploid markers genotyped on 96 samples. It contains 3 files: the summary file, and two probeset lists with all autotetraploid markers and the set of markers kept for plotting. When *fitTetra_Input*, *saveMarkerModels* from the *fitTetra* package, and *fitTetra_Output* have been run, we can plot the results.

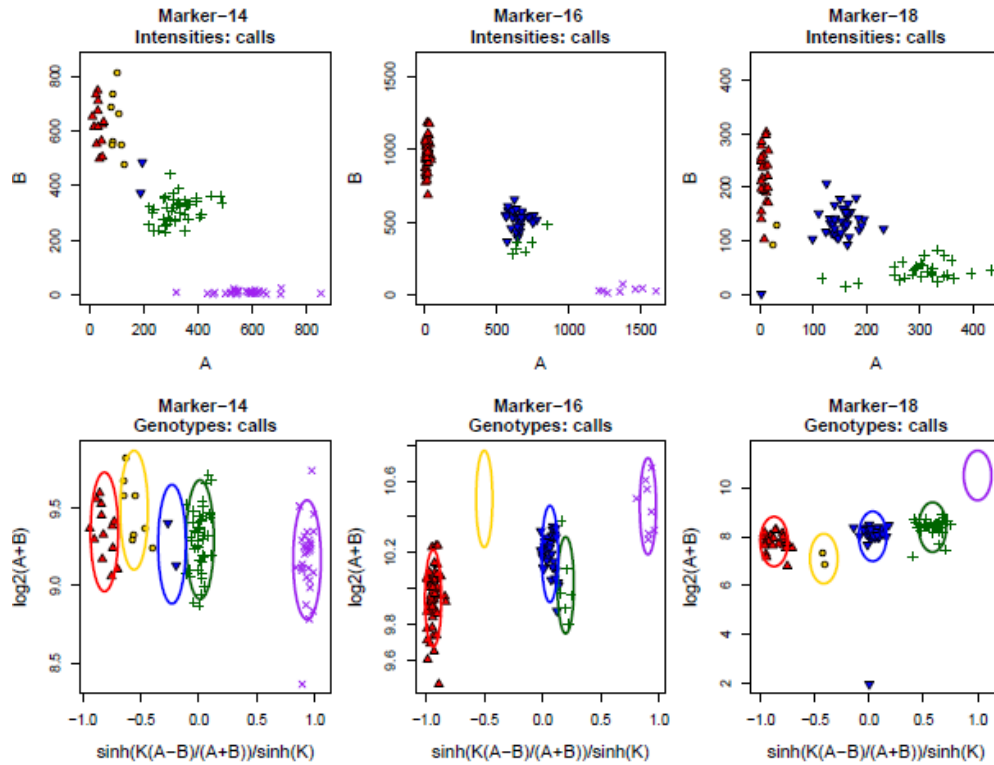


Figure 72: Standard output from *Ps_Visualization* using all default values for autotetraploid probesets.

6 References

References

- [1] **Discovery of novel variants in genotyping arrays improves genotype retention and reduces ascertainment bias.** Didion, J.P., Yang, H., Sheppard, K., Fu, C., McMillan, L., Villena, F.P., and Churchill, G.A. (2012) *BMC Genomics* 13 (2012) - p. 34
- [2] **Genotype calling in tetraploid species from bi-allelic marker data using mixture models.** Voorrips, R.E. , Gort, G. , Vosman, B. (2011) *BMC Bioinformatics* 12 (2011). - ISSN 1471-2105 - p. 11.
- [3] **ggplot2: Elegant Graphics for Data Analysis.** Wickham, H. (2016) Springer-Verlag New York. ISBN 978-3-319-24277-4, <https://ggplot2.tidyverse.org>
- [4] **Points of view: Color blindness.** Wong, B. (2011) *Nat Methods* 8, 441. <https://doi.org/10.1038/nmeth.1618>
- [5] **knitr: A General-Purpose Package for Dynamic Report Generation in R.** Xie, Y. (2020) R package version 1.28, <https://yihui.org/knitr/>
- [6] **Dynamic Documents with R and knitr, 2nd edition.** Xie, Y. (2015) Chapman and Hall/CRC, Boca Raton, Florida. ISBN 978-1498716963, <https://yihui.org/knitr/>
- [7] **knitr: A Comprehensive Tool for Reproducible Research in R.** Xie, Y. (2014) In Stodden V, Leisch F, Peng RD (eds.), *Implementing Reproducible Computational Research*. Chapman and Hall/CRC. ISBN 978-1466561595, <http://www.crcpress.com/product/isbn/9781466561595>
- [8] **Axiom® Genotyping Solution Data Analysis Guide**, https://assets.thermofisher.com/TFS-Assets/LSG/manuals/axiom_genotyping_solution_analysis_guide.pdf
- [9] <http://cran.r-project.org/web/packages/fitTetra/index.html>
- [10] <http://www.wageningenur.nl/en/Expertise-Services/Collaboration-and-partnerships/Plant-Breeding.htm>
- [11] https://en.wikipedia.org/wiki/Mahalanobis_distance
- [12] https://en.wikipedia.org/wiki/Prasanta_Chandra_Mahalanobis
- [13] https://en.wikipedia.org/wiki/Hoeffding%27s_inequality
- [14] <http://www.r-project.org>
- [15] <http://cran.r-project.org>
- [16] <http://www.rstudio.com>
- [17] http://www.cran.r-project.org/doc/manuals/R-admin.html#Add_002don-packages
- [18] <http://www.cyclismo.org/tutorial/R/types.html>
- [19] <http://www.cran.r-project.org/doc/manuals/R-intro.pdf>
- [20] http://cran.r-project.org/doc/contrib/Lam-IntroductionToR_LHL.pdf
- [21] <http://www.r-tutor.com/r-introduction>
- [22] http://www.computerworld.com/s/article/9239625/Beginner_s_guide_to_R_Introduction
- [23] <http://adv-r.had.co.nz/memory.html>
- [24] <http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf>

- [25] <http://research.stowers-institute.org/efg/R/Color/Chart/>
- [26] <http://www.perl.org>
- [27] <http://www.activestate.com/activeperl/downloads>
- [28] <http://gradstudents.wcas.northwestern.edu/~jaf502/blog/2013/03/11/installing-r-in-linux/>
- [29] http://www.cran.r-project.org/doc/manuals/R-admin.html#Installing-R-under-Unix_002dalikes
- [30] <https://fedoraproject.org/wiki/EPEL>
- [31] <http://www.7-zip.org/>
- [32] <https://plot.ly>
- [33] <https://plot.ly/r/getting-started/>
- [34] <http://get.webgl.org>

Appendix A Installation

You must have R and 64-bit perl installed for `SNPolisher` to work. Both R and perl are free, and can be downloaded from their respective websites:

<http://cran.r-project.org> [15]

<http://www.perl.org> [26]

We also recommend installing RStudio as well as R. RStudio is a free user interface for R that displays multiple windows at once (history, plots, workspace) instead of only the command line.

<http://www.rstudio.com> [16]

To run `SNPolisher`, you need to have R version 4.0 or newer and perl version 5.8.0 (64-bit) or newer. The R package `ggplot2` should be installed.

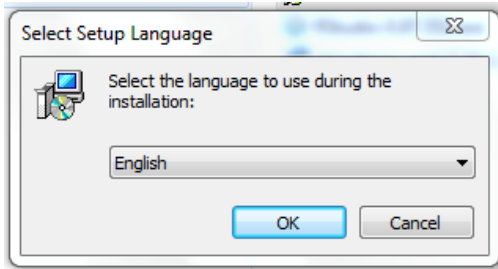
A.1 Windows

A.1.1 R

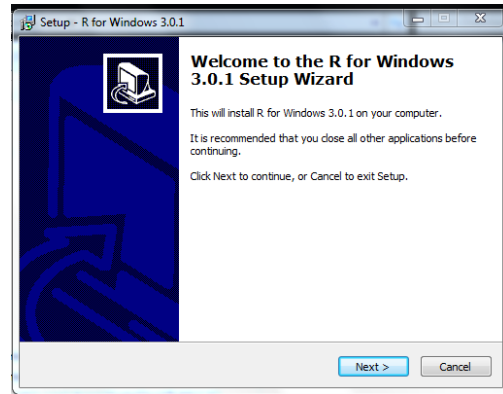
1. On the CRAN website, click on “Download R for Windows”.
2. Click on “base”.
3. Click on “Download R 3.0.1 for Windows” (or whatever version is the most current).
4. Download and save the “R-win.exe” file.
5. Double-click on the *exe* file to begin installation.
6. Select your language (English is the default). See figure 73a.
7. Begin the installation and accept the GNU public license. See figure 73b.
8. Select the installation location. The default is to install to *Program Files/R*, which we recommend you do. See figure 73c.
9. Select which components to install. The default is to install all components. If you don’t know whether to install 32 bit or 64 bit R, install both. See figure 73d.
10. Customize the startup options (create icons, etc.).
11. Choose if R should be added to the start menu.
12. Create icons on the desktop and save registry information.
13. Install R. See figure 73e.
14. Click “Finish” to complete the installation. See figure 73f.

A.1.2 R Studio

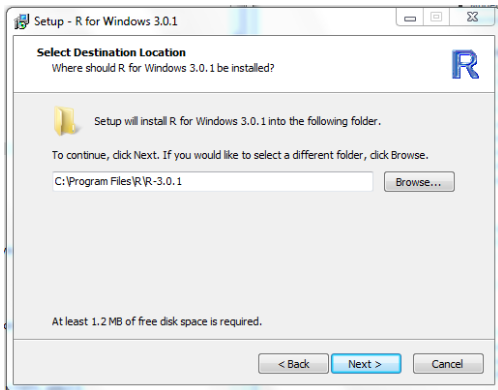
1. On the RStudio website, click on “download now”.
2. Select which version to download and click. RStudio Server will not work unless R has previously been installed on the server.
3. Download the *exe* file for your version of Windows.
4. Double-click on the *exe* file to begin installation.
5. Select the installation location. The default is to install to *Program Files/RStudio*, which we recommend you do. See figure 74a.
6. Install RStudio. See figure 74b



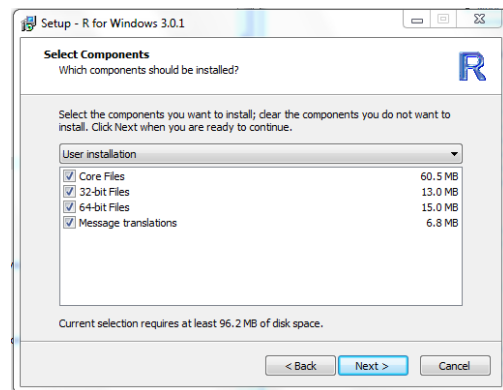
(a) Select the language



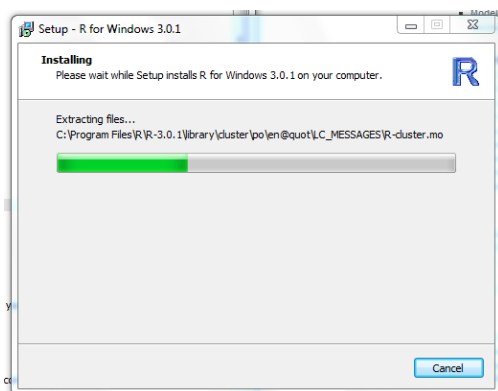
(b) Start the installation



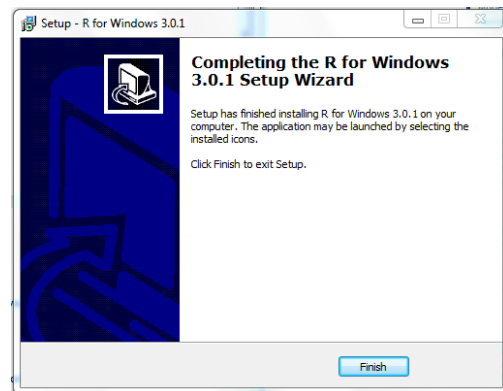
(c) Select the installation location. Default is to *Program FilesR*.



(d) Select components to install.



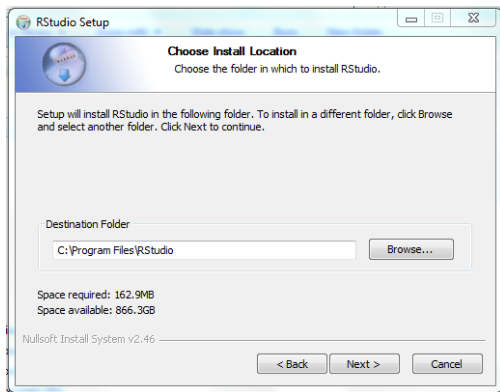
(e) The installation running should look similar to this.



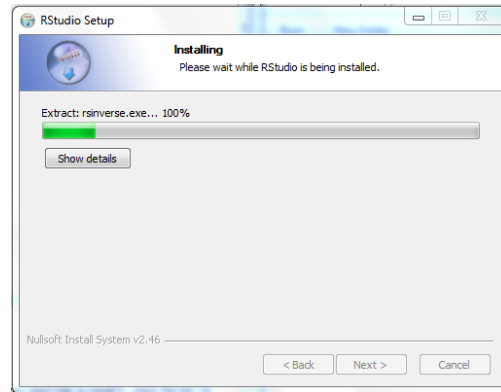
(f) Click "finish" to complete the installation.

Figure 73: Installing R for Windows.

7. Click “Finish” to complete the installation.



(a) Select the installation location. Default is to *Program Files\RStudio*.



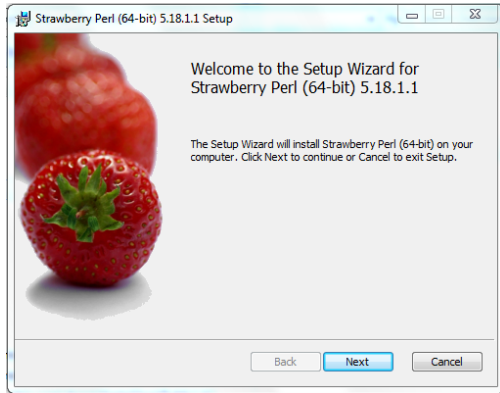
(b) The installation running should look similar to this.

Figure 74: Installing RStudio for Windows.

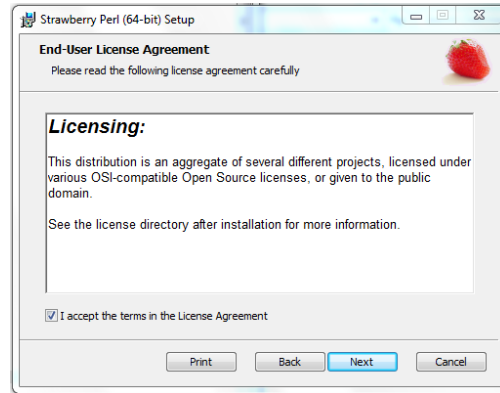
A.1.3 Perl

The example shown uses Strawberry perl. However, any version of perl for Windows will have a similar installation process. Make sure to download the 64-bit version.

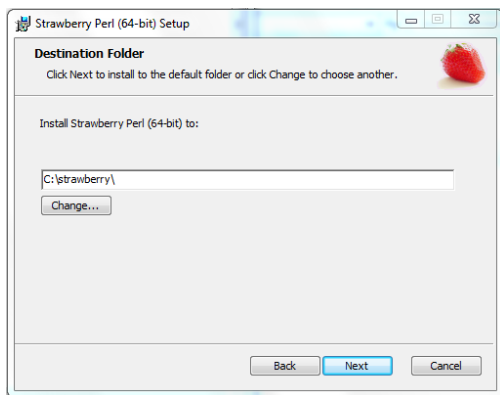
1. Download the *msi* file for perl installation.
2. Accepting the licensing agreement. See figure 75b.
3. Select the installation location. We recommend you use the default location provided by the installer. See figure 75c.
4. Install perl. See figure 75d.
5. Click “Finish” to complete the installation.



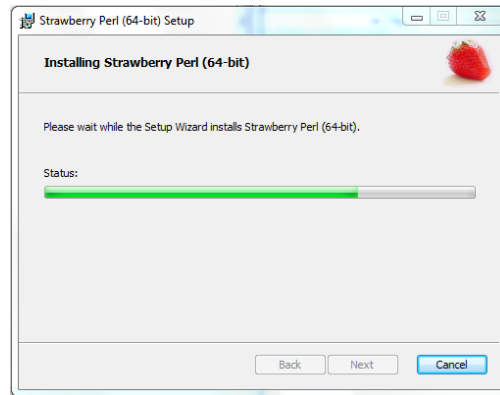
(a) Start the installation.



(b) Accept the licensing agreement.



(c) Select the installation location. Use the default location.



(d) The installation running should look similar to this.

Figure 75: Installing Perl for Windows.

A.2 Mac OS X

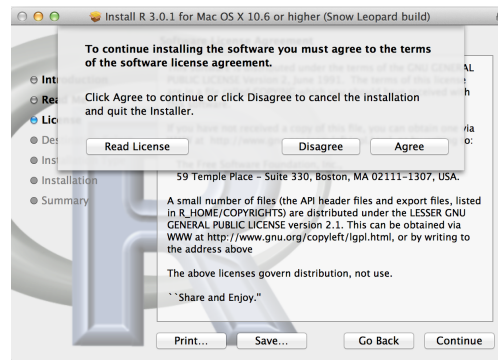
A.2.1 R

1. On the CRAN website, click on “Download R for (Mac) OS X”.
2. Click on the package link to download and save (R-3.0.1.pkg) to the designated downloads folder.
3. Double-click on the *pkg* file to begin installation. See figure 76a.
4. Click the “Continue” button to see the Introduction, Read Me, and License screens. Accept the license agreement. See figure 76b.
5. Select which users will have access to R. See figure 76c.
6. Select the installation location. We suggest that you use the default installation location (Applications). See figure 76d.
7. Enter your admin password to grant permission to install. See figure 76e.
8. Click “Close” to complete the installation. See figure 76f.
9. (If you are not using RStudio) Open the Applications folder, find the R icon, and drag to the dock to create a shortcut to R.

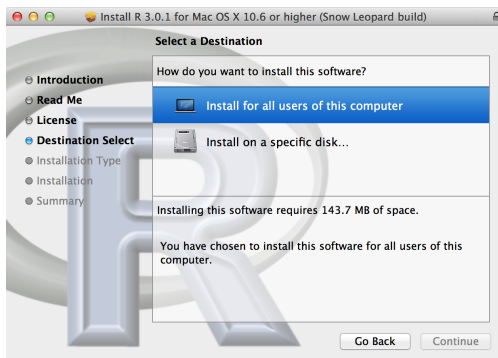
10. (If you are not using RStudio) The first time you use R, you will be asked if you want to open a file downloaded from the internet. Click on “Yes”.



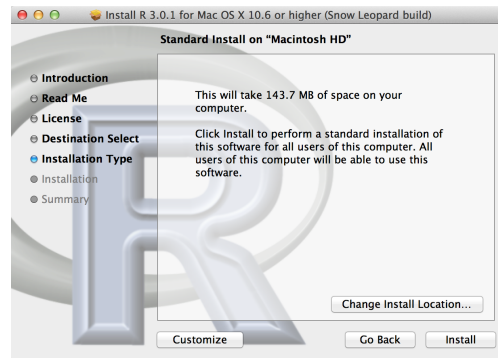
(a) Start the installation



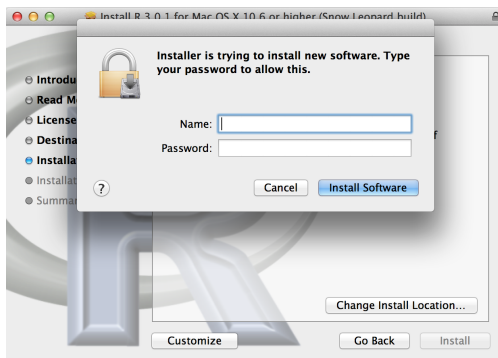
(b) Accept the license agreement



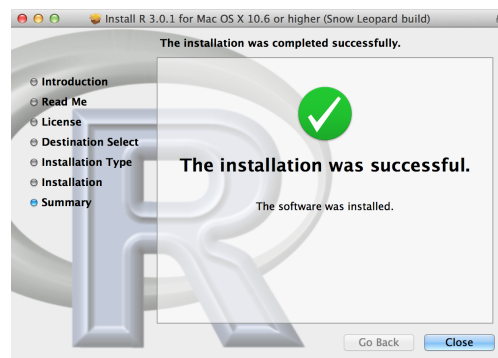
(c) Select the users.



(d) Installation.



(e) Input your admin password.



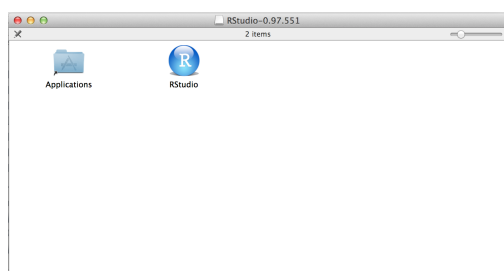
(f) Click “Close” to complete the installation.

Figure 76: Installing R for OS X.

A.2.2 R Studio

1. On the RStudio website, click on “download now”.
2. Select which version to download and click. RStudio Server will not work unless R has previously been installed on the server. The release for Mac OS X 10.6+ is under “All Platforms” if it does not display as “Recommended For Your System”.
3. Download the *dmg* file.

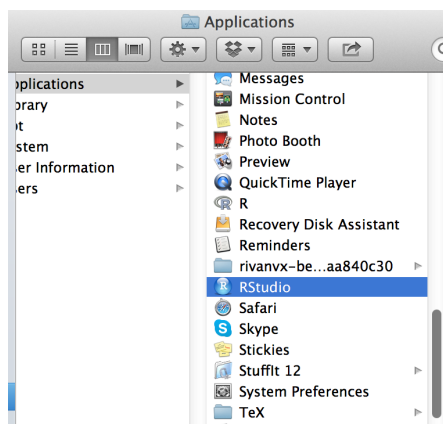
4. Double-click on the *dmg* file to begin installation. The mounted drive will display two icons: Applications folder and RStudio. Drag the RStudio icon into the Applications folder icon. RStudio will be copied to your Applications folder. See figure 77a.
5. Click the red button in the upper left-hand corner to close the mounted drive folder. The drive will be visible in a folder window under the “Devices” menu and as an icon on the desktop. Either drag the drive icon to the trash or click on the eject button in the devices menu to unmount the drive. The disk icon will not be visible on the desktop once the drive is unmounted. See figure 77b.
6. Open the Applications folder, find the RStudio icon, and drag to the dock to create a shortcut to RStudio. See figure 77c.
7. The first time you use RStudio, you will be asked if you want to open a file downloaded from the internet. Click on “Yes”. See figure 77d.



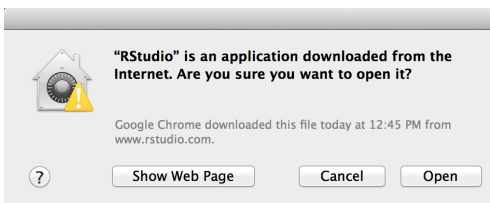
(a) The two icons in the mounted drive.



(b) The mounted drive on the desktop.



(c) RStudio icon in the Applications folder.

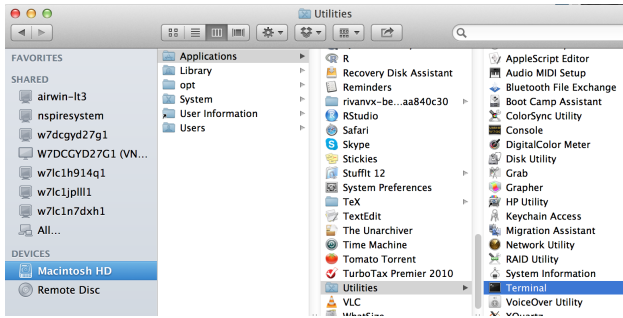


(d) Opening RStudio for the first time.

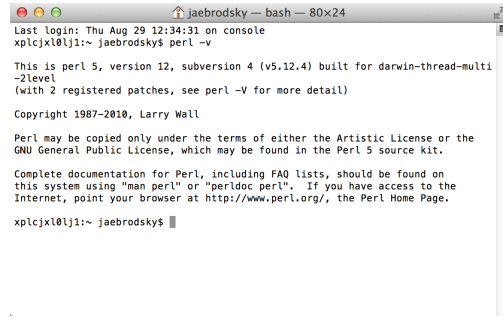
Figure 77: Installing RStudio for OS X.

A.2.3 Perl

Perl is already installed on OS X. This version should be sufficient to run *SNPolisher*. If you want to know what version of perl is installed, go to Application → Utilities → Terminal (see figure 78a). Open a terminal window and type `perl -v` at the prompt (see figure 78b). If you want to install the newest version of perl, perl.org has the binaries for installing Activeperl (<http://www.perl.org/get.html>) [26].



(a) Open a terminal window.



(b) Type `perl -v` to see which version of perl is installed.

Figure 78: Installing Perl for OS X.

A.3 Linux

A.3.1 R

See these pages for more details: [28] and [29].

Installing R is similar for Debian and RedHat distributions of Linux. These instructions assume that you use a package manager such as apt or yum.

CRAN maintains detailed instructions for installing R on its website. Click on “Download R for Linux” and then select the distribution folder. The Debian and Ubuntu folders each have a comprehensive README file but the RedHat folder does not.

For Debian and Ubuntu, you need to add the CRAN repository of packages to `/etc/apt/sources.list`.

Ubuntu To manually edit `sources.list`, open a terminal window and type `sudo nano /etc/apt/sources.list`. To install R, open a terminal window and type

```
# Grabs your version of Ubuntu as a BASH variable
CODENAME=$(grep CODENAME /etc/lsb-release | cut -c 18-)

# Appends the CRAN repository to your sources.list file
sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/ubuntu $CODENAME"
>> /etc/apt/sources.list'

# Adds the CRAN GPG key, which is used to sign the R packages for security.
sudo apt-key adv --keyserver keyserver.ubuntu.com --recv-keys E084DAB9

sudo apt-get update
sudo apt-get install r-base r-dev
```

Debian Just like Ubuntu, you need to update the `sources.list` file to include the CRAN repository:

```
# Appends the CRAN repository to your sources.list file
sudo sh -c 'echo "deb http://cran.rstudio.com/bin/linux/debian lenny-cran/"
>> /etc/apt/sources.list'

# Adds the CRAN GPG key, which is used to sign the R packages for security.
sudo apt-key adv --keyserver subkeys.pgp.net --recv-key 381BA480
sudo apt-get update
sudo apt-get install r-base r-base-dev
```

RedHat For RedHat EL5 and EL6, you need to add the Extra Packages for Enterprise Linux (EPEL) repository instead of the CRAN repository. You can read more about EPEL at <https://fedoraproject.org/wiki/EPEL> [30]. Open a terminal window and type


```
# For El5 or CentOS 5
su -c 'rpm -Uvh http://download.fedoraproject.org/pub/epel/5/i386/
    epel-release-5-4.noarch.rpm'
sudo yum update
sudo yum install R
```

```
# For El6 or CentOS 6
su -c 'rpm -Uvh http://download.fedoraproject.org/pub/epel/6/i386/
    epel-release-6-8.noarch.rpm'
sudo yum update
sudo yum install R
```

You can search for more R packages by opening a terminal window and typing `yum list R-`.

Fedora The newer versions of Fedora have the several most recent R versions in their repositories. Open a terminal window and type

```
sudo yum update
sudo yum install R
```

A.3.2 RStudio

To install RStudio, you need the link to the latest RStudio package. RStudio supports Debian-based and RedHat-based distributions. On the RStudio webpage (<http://www.rstudio.com>), click on “Download now”, select either desktop or server, and locate your distribution. In this example, the Debian/Ubuntu 64 bit release has the location <http://download1.rstudio.org/rstudio-0.97.551-amd64.deb>. Once you have determined the location of the package you need, open a terminal window and type

Debian/Ubuntu

```
sudo apt-get install
    http://download1.rstudio.org/rstudio-0.97.551-amd64.deb
```

RedHat/Fedora

```
sudo yum install
    http://download1.rstudio.org/rstudio-0.97.551-x86_64.rpm
```

A.3.3 Perl

Perl is already installed on Linux distributions. This version should be sufficient to run **SNPolisher** If you want to install the newest version of perl, perl.org (<http://www.perl.org/get.html> [26]) has links to the Activeperl website for downloading the newest binary files (<http://www.activestate.com/activeperl/downloads> [27]).

Appendix B R Basics for SNPolisher

There are many good introductions to R available on the internet. We recommend several official tutorials available through CRAN (see [19] and [20]), as well as the R Tutorial website's introduction ([21]) and ComputerWorld's introduction ([22]). This section very briefly describes some useful information for running SNPolisher in R, but it does not go in depth on how to use R. We strongly recommend that SNPolisher users who have never used R before spend an hour or two learning the basics of R. This will greatly improve the SNPolisher experience.

In R, commands are typed at the prompt (`>`) and are called *functions*. The inputs for a function are called *arguments*. Functions in R are passed arguments, written inside parentheses. This tells R to run the function using those arguments. The function *exp* is exponentiation, so if wanted to know e^0 or e^1 we would type:

```
> exp(0)
[1] 1
> exp(1)
[1] 2.718282
```

Sometimes we may need more information about a function. Say we want to know what the sine of 30 degrees is, so at the prompt we would type:

```
> sin(30)
[1] -0.9880316
```

But the sine of 30 degrees is $\frac{1}{2}$. It looks like the *sin* function expected radians instead of degrees. How can we tell it that we want to input degrees?

All R functions come with help files, which are accessible from the prompt by typing either `help` or `?` with the function name. In our case, we can type `help(sin)` or `?sin`. R will bring up a webpage with the help file (in RStudio, it is displayed in the help window). The help files are stored locally in a function's library so they will display even if there isn't an internet connection.

```
> help(sin)
```

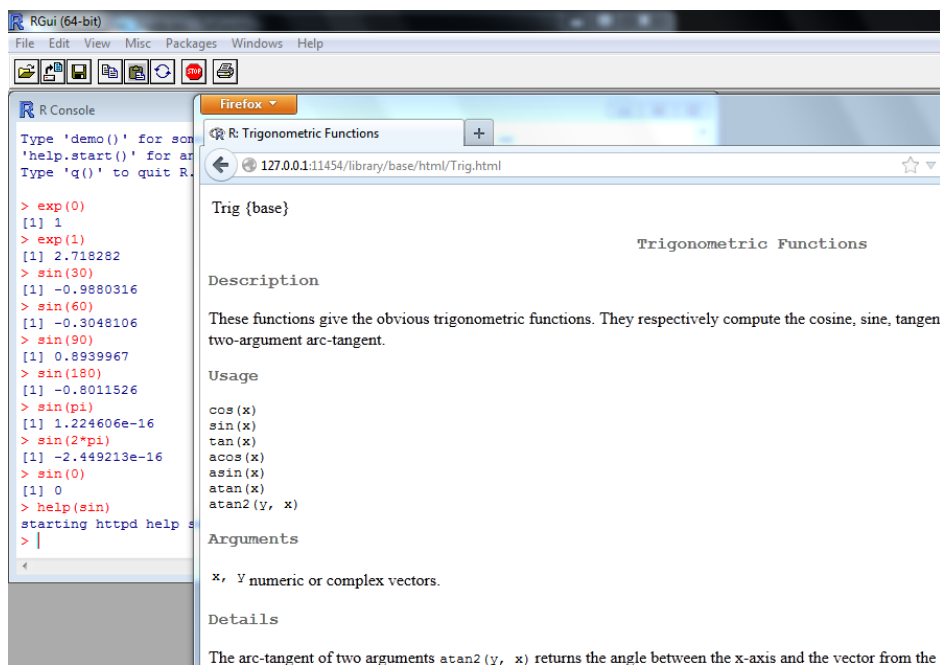


Figure 79: R help file for sine.

Help files have a short description of what the function does (*Description*), how the function must be typed at the prompt (*Usage*), how many and what type of inputs must be used with the function (*Arguments*), and any other information and references that may be helpful (*Details*, *S4 methods*, *References*, etc.).

The help command for *sin* brings up one help file for all of the basic trigonometric functions. It tells us that input must be in radians:

```
Angles are in radians, not degrees (i.e., a right angle is pi/2)
```

Now we know that we must ask for sine of $\frac{\pi}{6}$ instead of 30 degrees:

```
> sin(pi/6)
[1] 0.5
```

The help file also tells us what the arguments should be. The functions all take an argument called *x* (`cos(x)`, `sin(x)`, etc.) and the function *atan2* takes two arguments, *x* and *y* (`atan2(y,x)`). The help file shows that these arguments must be real or complex numbers and must be in the form of a vector:

```
x, y numeric or complex vectors.
```

R has several different data types. The most commonly used are scalars, vectors, matrices, data frames, and lists. This help file tells us that the trigonometric functions take vectors, which are $1 \times n$ or $n \times 1$ matrices. *SNPolisher* takes external files as inputs, so you do not need to be familiar with R data types to use it. You can read more about R data types in [18].

In several *SNPolisher* functions, data files are read into R using *read.table*. This is a generic, all-purpose command for loading files that may have row and/or column names. *read.table* is designed to handle data that is in a table format, and it produces data frames when the data is read into R. The user can specify if there is a first line that holds the names of the columns (`header=T`) or not (`header=F`), and what the separating character is between columns (`sep=" "`).

A data frame can be thought of as a 2-dimensional matrix that stores most types of data (similar to an excel worksheet). It is easy to call smaller portions of a data frame instead of the entire data frame. For example, we could read in a performance file output from *ps-classification* as a data frame and then look at the first five rows:

```
> ps.performance <- read.table("Ps.performance.txt",header=T)
> ps.performance[1:5,]
```

When calling data from a data frame, square brackets are used to indicate that the number of a row and the number of a column are going to be specified. Rows are written first and then columns: `data[row,column]`. If no rows or columns are given, then all rows or columns are returned: `data[row,]` returns that one row across all columns.

A user can also select multiple rows and columns in combinations. The colon operator (`:`) means “to”, so `1:5` means “1 to 5”. To specify rows or columns that are not directly adjacent, the concatenate operator (`c`) can be used to create a vector of numbers: `c(1,3,5)` produces `[1] 1 3 5`. In the previous example, we wanted to see the first five rows of the performance file. Now let’s say that we also want to see columns 1, 2, 16, and 17:

```
> ps.performance[1:5,c(1:2,16:17)]
```

