How to use the Musette package

Jennifer Wong, Thomas Picchetti, Francois Radvanyi and Etienne Birmele

19/02/2024

Installation

It is possible to install Musette from the GitHub repository using the devtools package.

```
library(devtools)
install_github("ebirmele/Musette")
```

It can now be loaded for use

```
library(musette)
```

```
## Warning: package 'igraph' was built under R version 4.2.3
```

Algorithm description

General aim

The user chooses among the data a subset of tumors which are called 'red' and another, disjoint subset of 'blue' tumors. A good solution S is a set of alterations $(A_1, A_2, ...A_n)$ such that :

1. many of the red tumors are affected by at least one of $A_1, \dots, A_n$.
2. few blue tumors are affected by any of $A_1, \dots, A_n$.
3. the number n of alterations in S is kept small.

A score is defined for any set S of alterations by

$$c(S) = -\frac{d_R(S)}{N_R} \log p_h(S)$$

where $p_h(S)$ denotes the hypergeometric p-value, $d_R(S)$ the number of red samples hit by S and N_R the number of red samples. The score takes both into account the goals of hitting essentially red samples (through the hypergeometric score) and as a high number of them (through $\frac{d_R(S)}{N_R}$).

The aim of the algorithm is to enumerate the best alteration sets.

Preprocessing

A data frame is constructed containing information about the alterations. Some are mandatory (name, number of blue/red neighbours), others are optional (alteration type, gene, chromosome location, etc).

For some alteration types (e.g. deletions and amplifications) which often act on a whole chromosome segment, we define two notions of domination in order to concentrate the whole segment into one alteration.

Consider two alterations A and B which are close enough on the genome, and such that A concerns more tumors than B, including a sufficient percentage (blind_percent) of the tumors touched by B. Then A is said to *dominate* B in the color-blind way. Blind domination is computed for alterations of the same type, for the types listed in 'blind_domination_step'. The set of considered alterations is reduced to a set of blind-leader alterations, that is a set such each alteration A is dominated by a some alteration, which is dominated by some alteration, ... in a chain leading to some blind-leader alteration. Only the leaders are kept to run the main algorithm.

If A and B are close enough, A has a better score than B, touching a sufficient fraction of red tumors touched by B, and if a sufficient fraction of blue tumors touched by A are also touched by B, then A dominates B in the color-aware way. The number of considered alterations is again reduced, in a similar manner to the blind domination, to a set of color_leaders. Only those are kept for the remaining steps of the analysis. In order to be able to get back to the original alterations, a list is created for every color-leader gene A, containing the list of the alterations which color_leader is A.

Enumeration algorithm

A tree of solutions (sets of alterations) is generated from a root which is the empty set. Every other solution S is the child of a solution S' having one less alteration. The child S is constructed and generated only if its score $c(S)$ is significantly better than its parents'. This is evaluated by computing a 'step score', which can be defined as :

- original mode: the probability of getting a better score than $c(S)$ by adding a random alteration to S'.
- best-first mode: the probability of getting a better score than $c(S)$ by replacing the 'worst' alteration in S by a random alteration (here 'worst' is in terms of the score of the individual alteration)
- strict mode: the highest of the p_i, where p_i is the probability of getting a better score than $c(S)$ by replacing alteration number i in S by a random alteration. The solution S is added to the tree only if this step-score is below a certain threshold. The tree of solutions is grown by gradually raising this threshold.

Outcome

The solutions are displayed in a data frame which columns include the alterations included each solution, its specificity and sensitivity in terms of covering of the red tumors or the fact that it can be extended or not to a better solution.

An example

Data and parameter preparation

We consider the *tcga_bladder* data present in the package, which was downloaded from the TCGA database.

It contains an object *matrices*, which is composed of three booelan matrices $matricesmuta*, *matricesdele$ and *matrices$ampli* which respectively indicate which mutations (among 16305), deletions (among 15063) and amplifications (among 20695) occur in 388 samples.

It also contains information on the chromosome and position of those alterations, as well as the known pathways for the corresponding genes.

```r
library(ComplexHeatmap)
data("tcga_bladder",package="musette")
```

The first (mandatory) step to define which sub-family of samples one wants to characterize. By analogy with the summary Figure, those samples are called *reds* whereas all samples are called *blues*.

Let us here define the reds as the *basal* samples. The aim of that run will thus be to identify alteration sets characterizing basal samples with respect to all the other types.

```r
reds= (groups == 'basal')
names(reds)=names(groups)
blues=!reds
names(blues)=names(groups)
```

Some other parameters to set are the number of solutions to generate and the mode of alteration sets generation (see [] for a description of the different choices)

```
bound=100  #number of solutions to generate
stepmode="strict"
```

Two pre-processing steps of domination can be run in order to decrease the number of considered alterations and thus the size of the combination space to explore.

The blind domination merges two alterations if they share mainly the same neighborhood. It requires to specify the type of alterations it concerns, as well as the minimum percentage of common neighbors and maximal distance in the genome to define a domination between two alterations.

Consider two deletions d_1 and d_2, close on the genome and such that in at least 90 of the cases where d_2 is effective, d_1 is also. Than d_2 can be considered as a side-effect of d_1 as deletions biologically concern whole regions. From an combinatorial point of view, d_2 is than suppressed from the instance and hidden behind d_1 (the list of alterations hidden behind the master alterations is kept in memory for final biological analysis)

```
blind_domination_step=c("ampli","dele") # alteration types  to be considered for the "blind" domination
blind_percent=90  #  required percentage for the "blind" domination step
blind_distance=5000000 # maximal distance at which an alteration can blind-dominate another one
```

The color domination is identical but by with a possibility of distinction between the blue and red percentages.

```
color_domination_step=c("ampli","dele") # same parameters for the "colored" domination.
red_percent=80  # two percentages (red and blue) have to be defined
blue_percent=80
color_distance=5000000
```

musette algorithm

The main algorithm can now be run. The messages given by the code are printed here to show the preprocessing steps, and the evolution of the step-score with the corresponding number of solutions. In this exemple, the stopping criterion is the discovery of the best 100 solutions.

```
ll=do.musette(matrices=matrices, reds=reds, blind_domination_step = blind_domination_step,
              color_domination_step = color_domination_step, blind_distance=blind_distance,
              color_distance=color_distance, blind_percent=blind_percent, red_percent=red_percent,
              blue_percent=blue_percent, chromosome = chromosome, longname=longname,
              pathways=pathways,position=position, bound=bound)
```

```
## [1] "computing full graph..."
## [1] "done."
## [1] "computing blind domination for  ampli"
## [1] "done."
## [1] "computing blind leaders for ampli"
## [1] "done."
## [1] "computing blind domination for  dele"
## [1] "done."
## [1] "computing blind leaders for dele"
## [1] "done."
## [1] "computing color-aware domination for  ampli"
## [1] "done."
## [1] "computing color-aware leaders for ampli"
## [1] "done."
## [1] "computing color-aware domination for  dele"
## [1] "done."
## [1] "computing color-aware leaders for dele"
## [1] "done."
## [1] "computing attributes for pseudo_alterations..."
## [1] "done."
## threshold <- 0.000000, Nodes : 1
## threshold <- 0.000000, Nodes : 2
## threshold <- 0.000000, Nodes : 4
## threshold <- 0.000001, Nodes : 5
## threshold <- 0.000011, Nodes : 6
## threshold <- 0.000018, Nodes : 7
## threshold <- 0.000018, Nodes : 8
## threshold <- 0.000031, Nodes : 9
## threshold <- 0.000035, Nodes : 10
## threshold <- 0.000036, Nodes : 12
## threshold <- 0.000048, Nodes : 13
## threshold <- 0.000069, Nodes : 14
## threshold <- 0.000073, Nodes : 15
## threshold <- 0.000084, Nodes : 16
## threshold <- 0.000086, Nodes : 17
## threshold <- 0.000103, Nodes : 18
## threshold <- 0.000124, Nodes : 19
## threshold <- 0.000137, Nodes : 20
## threshold <- 0.000145, Nodes : 22
## threshold <- 0.000147, Nodes : 23
## threshold <- 0.000153, Nodes : 24
## threshold <- 0.000163, Nodes : 26
## threshold <- 0.000163, Nodes : 27
## threshold <- 0.000171, Nodes : 28
## threshold <- 0.000174, Nodes : 29
## threshold <- 0.000175, Nodes : 30
## threshold <- 0.000181, Nodes : 31
## threshold <- 0.000184, Nodes : 32
## threshold <- 0.000188, Nodes : 34
## threshold <- 0.000192, Nodes : 35
## threshold <- 0.000199, Nodes : 37
## threshold <- 0.000201, Nodes : 40
## threshold <- 0.000209, Nodes : 41
## threshold <- 0.000212, Nodes : 43
## threshold <- 0.000212, Nodes : 44
## threshold <- 0.000216, Nodes : 45
## threshold <- 0.000222, Nodes : 48
## threshold <- 0.000223, Nodes : 49
## threshold <- 0.000230, Nodes : 50
## threshold <- 0.000232, Nodes : 52
## threshold <- 0.000237, Nodes : 53
## threshold <- 0.000238, Nodes : 54
## threshold <- 0.000239, Nodes : 56
## threshold <- 0.000240, Nodes : 57
## threshold <- 0.000253, Nodes : 58
## threshold <- 0.000269, Nodes : 59
## threshold <- 0.000270, Nodes : 60
## threshold <- 0.000270, Nodes : 63
## threshold <- 0.000276, Nodes : 64
## threshold <- 0.000280, Nodes : 65
## threshold <- 0.000291, Nodes : 68
## threshold <- 0.000292, Nodes : 71
## threshold <- 0.000295, Nodes : 74
## threshold <- 0.000296, Nodes : 75
## threshold <- 0.000305, Nodes : 87
## threshold <- 0.000306, Nodes : 88
## threshold <- 0.000307, Nodes : 91
## threshold <- 0.000307, Nodes : 95
## threshold <- 0.000309, Nodes : 96
## threshold <- 0.000310, Nodes : 97
## threshold <- 0.000310, Nodes : 98
## threshold <- 0.000312, Nodes : 99
## threshold <- 0.000315, Nodes : 100
## Size bound reached
```

Note that the *chromosome*, *longname* and *pathways* arguments are not mandatory.

The output contains two objects, best explored with the *View()* function:

```
solutions=ll$solutions
#View(solutions)
alterations=ll$alterations
#View(alterations)
```

The solution array lists the explored alteration sets with the following items for each:

- its number of red and blue neighbors and associated score
- the stepscore needed to explore it
- its sensitivity and specificity to discriminate red samples
- the list of all alterations, including those hidden behind the selected ones

```
head(solutions)
```

```
##     alterations reds blues    score   step_score    threshold size red.total
## 63 muta_RB1....  119    67 24.93010 2.173537e-04 2.704856e-04    6       167
## 52 muta_RB1....  100    44 21.10295 2.294950e-04 2.315727e-04    5       167
## 62 muta_RB1....  111    62 20.68118 1.685246e-04 2.704856e-04    5       167
## 95 muta_RB1....   99    44 20.34539 2.196092e-04 3.074502e-04    6       167
## 44 muta_TP5....  139   115 19.63485 2.123311e-04 2.123311e-04    5       167
## 12 muta_RB1....   92    37 18.78561 1.996783e-05 3.582033e-05    4       167
##    blue.total sensitivity specificity parent  leaf childrenThreshold
## 63        221   0.7125749   0.6968326     62  TRUE      0.0003729691
## 52        221   0.5988024   0.8009050     51  TRUE      0.0007938239
## 62        221   0.6646707   0.7194570     61 FALSE      0.0002173537
## 95        221   0.5928144   0.8009050     94  TRUE      0.0015436092
## 44        221   0.8323353   0.4796380     39  TRUE      0.0006424077
## 12        221   0.5508982   0.8325792     11  TRUE      0.0006118433
##       all_genes     all_loci allred allblue
## 63 muta_RB1.... 13, 2, 1....    167     221
## 52 muta_RB1.... 13, 2, 17, 5    167     221
## 62 muta_RB1.... 13, 2, 1....    167     221
## 95 muta_RB1.... 13, 2, 1....    167     221
## 44 muta_TP5.... 17, 8, 1....    167     221
## 12 muta_RB1.... 13, 2, 3, 17    167     221
```

The alterations array lists all aterations with the following items:

- the alterations hidden behind them or the alterations it is hidden behind
- the pathways it belongs to

- its full name (*longname*), chromosome and position

```
head(alterations)
```

```
##                     name type   gene chromosome    position     longname
## muta_BPIFC   muta_BPIFC muta  BPIFC         22  32439165.5 BPI fold....
## muta_MAGEA3 muta_MAGEA3 muta MAGEA3         23 152700549.5 MAGE fam....
## muta_DRP2     muta_DRP2 muta   DRP2         23   101242133 dystroph....
## muta_GCSAML muta_GCSAML muta GCSAML          1   247542374 germinal....
## muta_ETV5     muta_ETV5 muta   ETV5          3   186078313 ets vari....
## muta_PI3       muta_PI3 muta    PI3         20    45175710 peptidas....
##             pathways   neighbours redneighbours blueneighbours nbredneighbours
## muta_BPIFC           TCGA.DK.....  TCGA.DK.....   TCGA.DK.....               3
## muta_MAGEA3          TCGA.DK.....  TCGA.DK.....   TCGA.UY.....               1
## muta_DRP2            TCGA.BT.....  TCGA.BT.....   TCGA.C4.....               2
## muta_GCSAML          TCGA.DK.....  TCGA.DK.....                              1
## muta_ETV5            TCGA.4Z.....  TCGA.DK.....   TCGA.4Z.....               2
## muta_PI3             TCGA.DK.....  TCGA.DK.....                              1
##             nbblueneighbours     hyper       score stepscore blind_dominators
## muta_BPIFC                 3 0.6559486 0.011783507        -1
## muta_MAGEA3                1 0.3912613 0.002342882        -1
## muta_DRP2                  4 0.2069502 0.002478445        -1
## muta_GCSAML                0 0.8430115 0.005047973        -1
## muta_ETV5                  2 0.5495109 0.006580969        -1
## muta_PI3                   0 0.8430115 0.005047973        -1
##             blind_dominated blind_leader  blind_repr color_dominated
## muta_BPIFC                    muta_BPIFC  muta_BPIFC
## muta_MAGEA3                  muta_MAGEA3 muta_MAGEA3
## muta_DRP2                      muta_DRP2   muta_DRP2
## muta_GCSAML                  muta_GCSAML muta_GCSAML
## muta_ETV5                      muta_ETV5   muta_ETV5
## muta_PI3                        muta_PI3    muta_PI3
##             color_dominators color_leader  color_repr   followers merged_from
## muta_BPIFC                     muta_BPIFC  muta_BPIFC  muta_BPIFC
## muta_MAGEA3                   muta_MAGEA3 muta_MAGEA3 muta_MAGEA3
## muta_DRP2                       muta_DRP2   muta_DRP2   muta_DRP2
## muta_GCSAML                   muta_GCSAML muta_GCSAML muta_GCSAML
## muta_ETV5                       muta_ETV5   muta_ETV5   muta_ETV5
## muta_PI3                         muta_PI3    muta_PI3    muta_PI3
##             merged_in
## muta_BPIFC
## muta_MAGEA3
## muta_DRP2
## muta_GCSAML  pseudo_4
## muta_ETV5
## muta_PI3     pseudo_4
```

Post-treatment

Graph Visualization

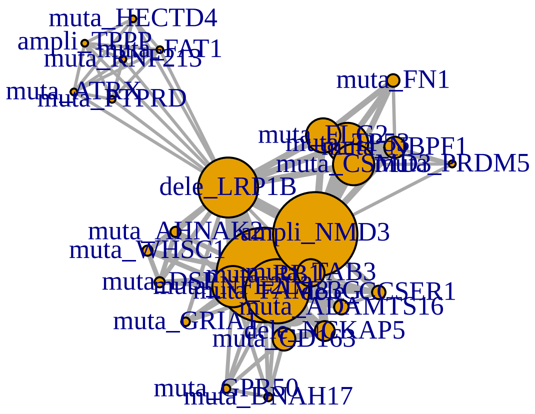
It is possible to visualize the graph of all alterations present in the top solutions. The node sizes are proportional to the sum of scores of the solutions an alteration belongs to, the edge width are proportional to the sum of scores of the solutions a pair has in common. The notion of alteration set is lost in that representation but it allows a quick glance into the results in terms of main alterations.

The plot is interactive, meaning that the nodes can be moved for a better visualization and some information on the alterations is given when pointing on them. The following code generates a plot using the 20 best alteration sets of the example:

```
g <- influenceGraph(ll,20,TRUE)
visIgraph(g)
```

As the interactive plot cannot be drawn in the pdf document, let us plot the influence graph as a basic igraph object:

```
g <- influenceGraph(ll,20,TRUE)
plot(g)
```



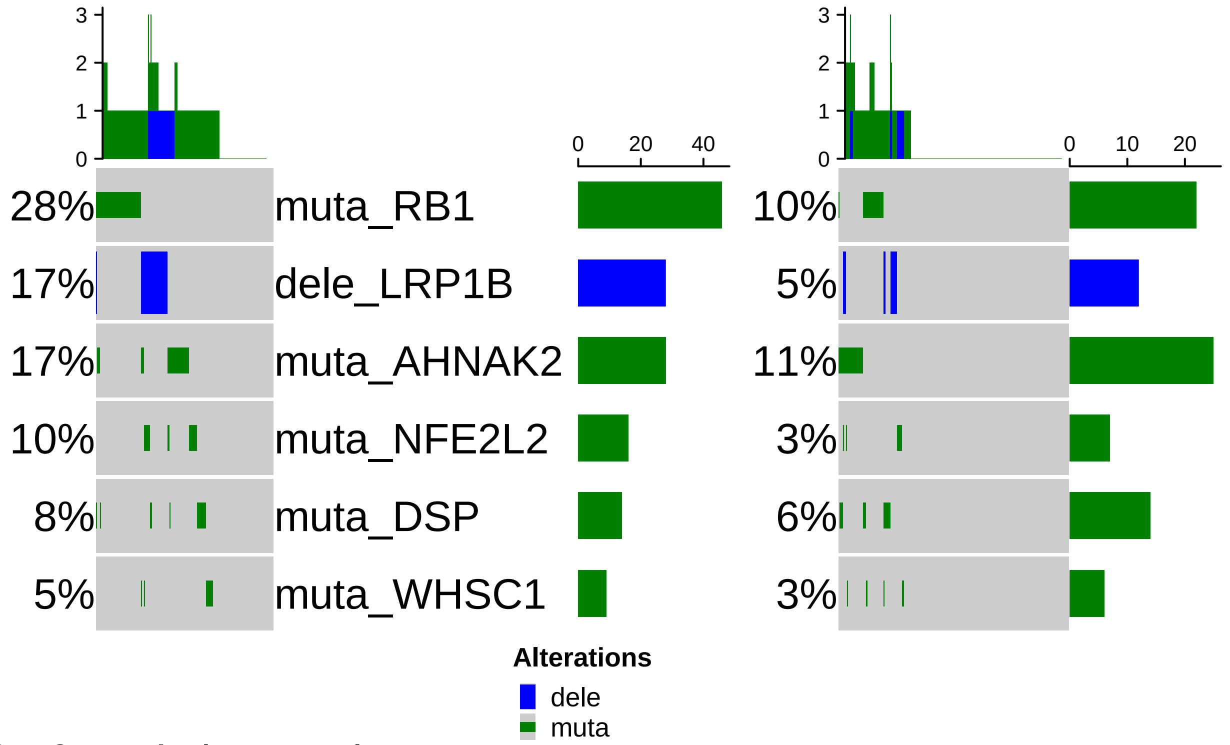
Other useful plots are the oncoplots generated with the *oncoPrint* function of the *ComplexHeatmap* package. Each such plot corresponds to a solution set and gives a readable summary of the considered alterations and the red and blue samples hit by each alteration.

The following code shows such plots for the 3 best solutions in terms of score.

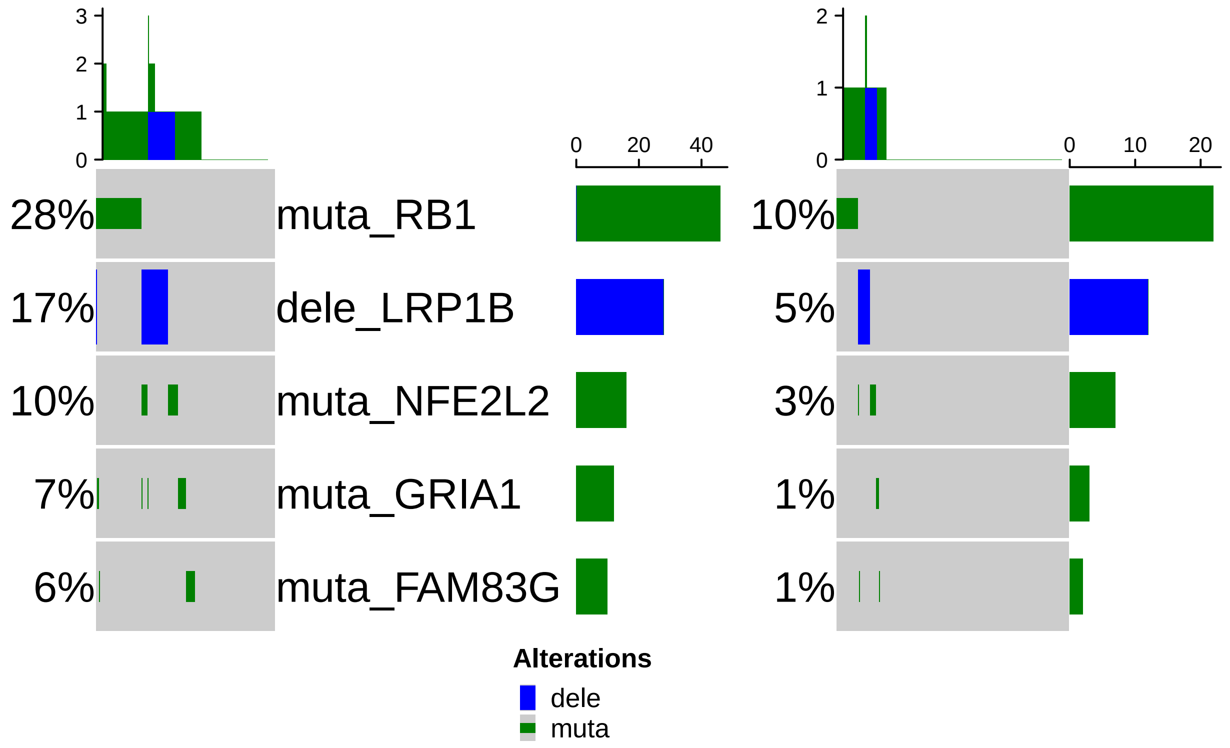
```
indices <- rownames(solutions)[1:3]
```

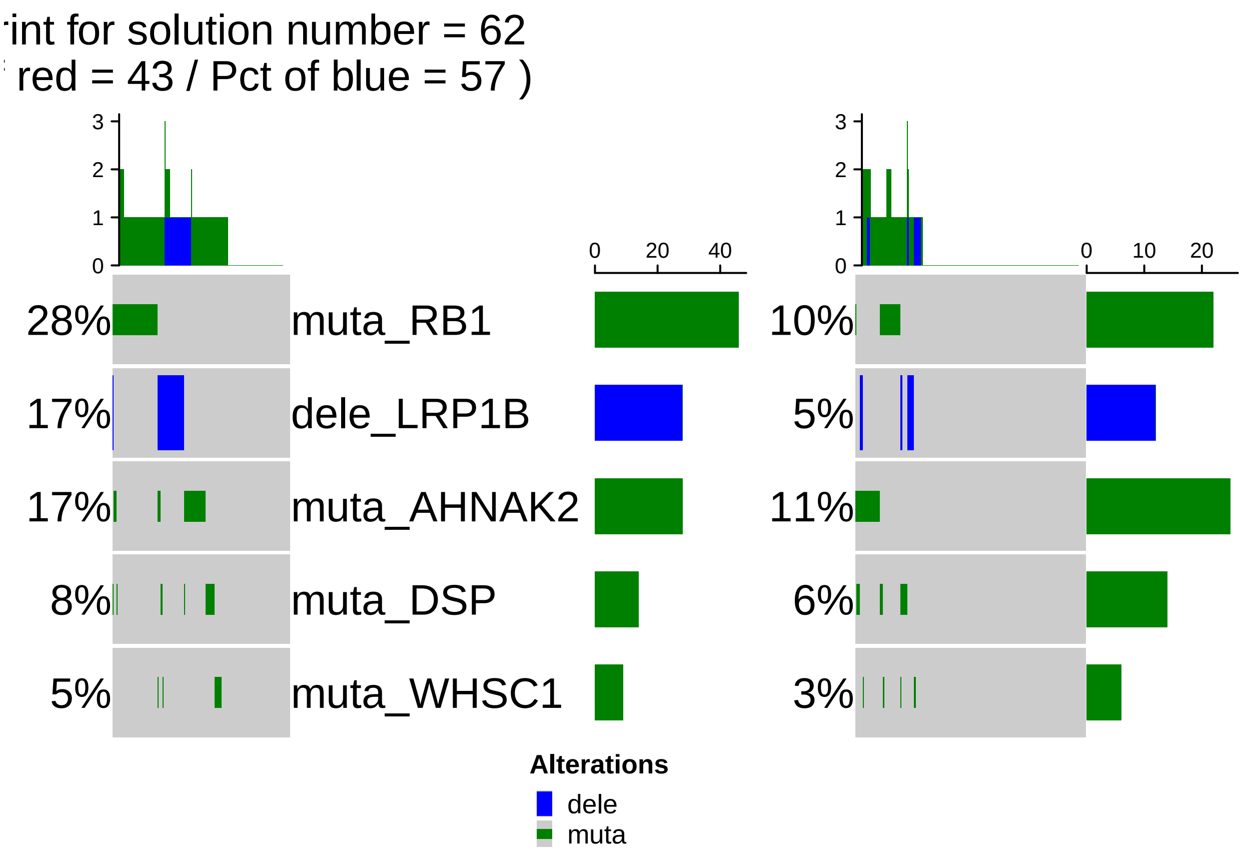
```
for (index in indices){
   solution.oncomatrix(ll,index,reds)
}
```

int for solution number = 63
red = 43 / Pct of blue = 57)



int for solution number = 52
red = 43 / Pct of blue = 57)





Pathway enrichment of solutions

It may be interesting from an interpretation point of view which pathways appear to contain several altered genes in the solutions. The *sharedPathway* function generates a dataframe with one line per (solution,pathway) couple for which the pathway appears at least twice in the list of concerned genes. This dataframe also contains the list of all concerned genes and their red and blue neighbors.

```
spdf = sharedPathways(solutions,alterations)
```

```
## [1] "computing shared pathway informations..."
```

```
#View(spdf)
```

```
head(spdf)
```

```
##   sol  alterations                                         pathway         gene
## 1  44 muta_TP5....                               Metabolic pathways ampli_SL....
## 2  12 muta_RB1....                               Metabolic pathways dele_KYN....
## 3  29 muta_RB1....                               Metabolic pathways ampli_SL....
## 4  29 muta_RB1....                                       Cell cycle muta_RB1....
## 5  29 muta_RB1....          Cytokine-cytokine receptor interaction ampli_IL....
## 6  28 muta_RB1.... Protein processing in endoplasmic reticulum ampli_SS....
##         leader redneighbours blueneighbours nbredneighbours nbblueneighbours
## 1 ampli_NM....   TCGA.CU.....   TCGA.2F.....              12                4
## 2 dele_LRP....   TCGA.CU.....   TCGA.FD.....              12                4
## 3 ampli_NM....   TCGA.CU.....   TCGA.2F.....              13                3
## 4 muta_RB1....   TCGA.4Z.....   TCGA.5N.....              47               22
## 5 ampli_NM....   TCGA.CU.....   TCGA.2F.....              13                3
## 6 ampli_NM....  TCGA.CU.....   TCGA.2F.....                24                10
```

Export in csv files

The dataframes containing the alterations, the solutions or the shared pathways have a format which is not compatible with the *write.csv* function. To obtain correctly files csv files, use the *musette2csv* function before using it.

```
# Data export to csv files
csv_sol=musette2csv(sol)
#write.csv(csv_sol,file="mysolutions.csv")
```