LowMACA: Low frequency Mutation Analysis via Consensus Alignment

Giorgio Melloni , Stefano de Pretis

April 16, 2015

Contents

1 Introduction

The *LowMACA* package is a simple suite of tools to investigate and analyze the profile of the somatic mutations provided by the cBioPortal (via the *cgdsr*). *LowMACA* evaluates the functional impact of somatic mutations by statistically assessing the number of alterations that accumulates on the same residue (or residue that are conserved in Pfam domains). For example, the known driver mutations G12,G13 and Q61 in KRAS can be found on the corresponding residues of other proteins in the RAS family (PF00071) like NRAS and HRAS, but also in less frequently mutated genes like RRAS and RRAS2. The corresponding residues are identified via multiple sequence alignment. Thanks to this approach the user can identify new driver mutations that occur at low frequency at single protein level but emerge at Pfam level. In addition, the impact of known driver mutations can be transferred to other proteins that share a high degree of sequence similarity (like in the RAS family example).

You can conduct an hypothesis driven exploratory analysis using our package simply providing a set of genes and/or pfam domains of your interest. The user is able to choose the kind of tumor and the type of mutations (like missense, nonsense, frameshift etc.). The data are directly downloaded from the largest cancer sequencing projects and aggregated by LowMACA to evaluate the possible functional impact of somatic mutations by spotting the most conserved variations in the cohort of cancer samples. By connecting several proteins that share sequence similarity via consensus alignment, this package is able to statistically assessing the occurrence of mutations on the same residue and ultimately see:

- where mutations fall and what are the involved domains
- what is the frequency of the aberrations and what is the more represented tumor type
- if and where the mutations tend to clusterize
- what is the degree of conservation of the mutated residues
- if there are new driver genes and in particular, driver mutations

2 System Requirements

LowMACA relies on two external resources to work properly.

- Clustal Omega, our trusted aligner (http://www.clustal.org/omega/)
- Ghostscript, a postscript interpreter needed to draw logo plots (http://www.ghostscript.com/)

Clustal Omega is a fast aligner that can be downloaded from the link above. For both Unix and Windows users, remember to have ”clustalo” in your PATH variable. In case you cannot set ”clustalo” in the PATH, you can always set the clustalo command from inside R, after creating a LowMACA object:

```r
#Given a LowMACA object 'lm'
lm <- newLowMACA(genes=c("TP53" , "TP63" , "TP73"))
lmParams(lm)$clustal_cmd <- "/your/path/to/clustalo"
```

If you cannot install clustalomega, we provide a wrapper around EBI web service (http://www.ebi.ac.uk/Tools/web
You just need to set your email as explained in section setup, but you have a limit of 2000 input sequences and perl must be installed.

Ghostscript is needed to draw logo plots and it is used by the library motifStack. Download it and install it from the link above. For Windows users only, don't forget to set this R environment command:

```r
# Needed only on Windows - run once per R session
# Adjust the path to match your installation of Ghostscript
if(Sys.info()['sysname']=="Windows")
  Sys.setenv(R_GSCMD = '"C:/Program Files/gs/gs9.15/bin/gswin64c.exe"')
```

More details can be found here http://pgfe.umassmed.edu/BioconductorGallery/docs/motifStack/motifStack.html
LowMACA needs an internet connection to retrieve mutation data from cBioPortal, draw the Protter-style plot (http://wlab.ethz.ch/protter/start/) and use the web service of clustalomega (http://www.ebi.ac.uk/Tools/webse

3 Create a LowMACA object

First of all, we have to define our target genes or pfam domains that we wish to analyze.

3.1 Find your target family of proteins or pfam you want to know more about

```
library(LowMACA)
#User Input
Genes <- c("ADNP","ALX1","ALX4","ARGFX","CDX4","CRX"
        ,"CUX1","CUX2","DBX2","DLX5","DMBX1","DRGX"
        ,"DUXA","ESX1","EVX2","HDX","HLX","HNF1A"
          ,"HOXA1","HOXA2","HOXA3","HOXA5","HOXB1","HOXB3"
                          ,"HOXD3","ISL1","ISX","LHX8")
Pfam <- "PF00046"
```

```
#Construct the object
lm <- newLowMACA(genes=Genes, pfam=Pfam)

## All Gene Symbols correct!

str(lm , max.level=3)

## Formal class 'LowMACA' [package "LowMACA"] with 4 slots
##   ..@ arguments:List of 6
##   .. ..$ genes      : NULL
##   .. ..$ pfam       : chr "PF00046"
##   .. ..$ input      :'data.frame': 28 obs. of  7 variables:
##   .. ..$ mode       : chr "pfam"
##   .. ..$ params     :List of 5
##   .. ..$ parallelize:List of 2
##   ..@ alignment: list()
##   ..@ mutations: list()
##   ..@ entropy  : list()
```

Now we have created a *LowMACA* object. In this case, we want to analyze the homeodomain fold pfam (PF00046), considering 28 genes that belong to this clan. If we don't specify the pfam parameter, *LowMACA* proceeds to analyze the entire proteins passed by the genes parameter (we map only canonical proteins, one per gene). Vice versa, if we don't specify the genes parameter, *LowMACA* looks for all the proteins that contain the specified pfam (or pfams if you wish) and analyze just the portion of the protein assigned to the domain.

3.2 Change default parameters

A LowMACA object is composed by four slots. The first slot is `arguments` and is filled at the very creation of the object. It contains information as Uniprot name for the proteins associated to the genes, the amino acid sequences, start and end of the selected domains and many default parameters that can be change to start the analysis.

```
#See default parameters
lmParams(lm)

## $mutation_type
## [1] "missense"
##
```

```
## $tumor_type
## [1] "all"
##
## $min_mutation_number
## [1] 1
##
## $density_bw
## [1] 0
##
## $clustal_cmd
## [1] "clustalo"
```

```
#Change some parameters
#Accept sequences even with no mutations
lmParams(lm)$min_mutation_number <- 0
#Changing selected tumor types
#Check the available tumor types in cBioPortal
available_tumor_types <- showTumorType()
head(available_tumor_types)
```

```
##                                          Adrenocortical Carcinoma
##                                                             "acc"
##                                          Adenoid Cystic Carcinoma
##                                                            "acyc"
##                       Bladder Cancer|Bladder Urothelial Carcinoma
##                                                            "blca"
##       Breast Invasive Carcinoma|Breast cancer patient xenografts
##                                                            "brca"
##                   Cancer Cell Line Encyclopedia|NCI-60 Cell Lines
##                                                        "cellline"
## Cervical Squamous Cell Carcinoma and Endocervical Adenocarcinoma
##                                                            "cesc"
```

```
#Select melanoma, stomach adenocarcinoma, uterine cancer, lung adenocarcinoma,
#lung squamous cell carcinoma, colon rectum adenocarcinoma and breast cancer
lmParams(lm)$tumor_type <- c("skcm" , "stad" , "ucec" , "luad" , "lusc" , "coadread" , "brca")
```

4 Setup

4.1 Align sequences

```
lm <- alignSequences(lm)

## Aligning sequences...
```

This method is basically self explained. It aligns the sequences in the object. If you didn't install clustalomega yet, you can use the web service of clustalomega that we wrapped in our R package. The limit is set to 2000 sequences and it is obviously slower than a local installation. Rememeber to put your own email in the mail command to activate this option since is required by the EBI server. In case of errors, they will be sent there.

```
lm <- alignSequences(lm , mail="lowmaca@gmail.com")
```

```
str(lmAlignment(lm) , max.level=2 , vec.len=2)

## List of 4
##  $ ALIGNMENT:'data.frame': 1708 obs. of  8 variables:
##   ..$ domainID      : Factor w/ 28 levels "ARGFX|PF00046|503582|79;135",..: 1 1 1 1 1 ...
##   ..$ Gene_Symbol   : Factor w/ 27 levels "ADNP","ALX1",..: 4 4 4 4 4 ...
##   ..$ Pfam          : Factor w/ 1 level "PF00046": 1 1 1 1 1 ...
##   ..$ Entrez        : Factor w/ 27 levels "1046","127343",..: 21 21 21 21 21 ...
##   ..$ Envelope_Start: num [1:1708] 79 79 79 79 79 ...
##   ..$ Envelope_End  : num [1:1708] 135 135 135 135 135 ...
##   ..$ Align         : int [1:1708] 1 2 3 4 5 ...
##   ..$ Ref           : num [1:1708] 79 80 81 82 83 ...
##  $ SCORE    :List of 2
##   ..$ DIST_MAT     : num [1:28, 1:28] NA 36.4 ...
##   .. ..- attr(*, "dimnames")=List of 2
##   ..$ SUMMARY_SCORE:'data.frame': 28 obs. of  4 variables:
##  $ CLUSTAL  :Formal class 'AAMultipleAlignment' [package "Biostrings"] with 3 slots
##  $ df       :'data.frame': 61 obs. of  2 variables:
##   ..$ consensus   : Factor w/ 18 levels "A","D","E","F",..: 13 13 1 13 15 ...
##   ..$ conservation: num [1:61] 0.326 0.472 ...
```

- ALIGNMENT: mapping from original position to the position in the consensus
- SCORE: some score of distance between the sequences
- CLUSTAL: an object of class `AAMultipleAlignment` as provided by Biostrings R package
- df: Consesus sequence and conservation Trident Score at every position

4.2 Get Mutations and Map Mutations

```
lm <- getMutations(lm)

## Getting mutations from cancers studies...
## Impossible to retrive mutations from coadread_mskcc study
## Impossible to retrive mutations from luad_tsp study

lm <- mapMutations(lm)
```

These commands produce a change in the slot mutation and provide the results from R cgdsr package.

```
str(lmMutations(lm) , vec.len=3 , max.level=1)
```

```
## List of 3
##  $ data   :'data.frame': 1389 obs. of  8 variables:
##  $ freq   :'data.frame': 7 obs. of  29 variables:
##  $ aligned: num [1:28, 1:61] 0 0 1 0 0 1 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
```

- data: provide the mutations selected from the cBioPortal divided by gene and patient/tumor type
- freq: a table containing the absolute number of mutated patients by gene and tumor type (this is useful to explore the mutational landscape of your genes in the different tumor types)
- aligned: a matrix of m rows, proteins or pfam, and n columns, consensus positions derived from the mapping of the mutations from the original positions to the new consensus

If we want to check what are the most represented genes in terms of number of mutations divided by tumor type, we can simply run:

```
myMutationFreqs <- lmMutations(lm)$freq
#Showing the first genes
myMutationFreqs[ , 1:10]

##    StudyID Total_Sequenced_Samples ADNP ALX1 ALX4 ARGFX CDX4 CRX CUX1 CUX2
## 1     brca                    1262   11    2    2     3    2   3    3    7
## 2 coadread                     296   11    4    4     5    0   4   11    9
## 3     luad                     513    6   10   10     9   11   9   21   19
## 4     lusc                     178    2    6    2     0    3   3    8    3
## 5     skcm                     559   10    3    6    19   24  13   32   59
## 6     stad                     438   11    7   10     2    7   9   21   21
## 7     ucec                     248   11    8    6     2    9   4   21    7
```

This can be useful for a stratified analysis in the future.

4.3 Whole setup

If you don't want to provide your own mutation data you can use directly the command setup to launch alignSequences, getMutations and mapMutations at once

```
#Local Installation of clustalo
lm <- setup(lm)
#Web Service
lm <- setup(lm , mail="lowmaca@gmail.com")
```

If you have your own data, you can use the getMutations method as `lm <- getMutations(lm , repos=myOwnData)`. myOwnData is a data.frame with the following columns:

```
str(lmMutations(lm)$data , vec.len=1)

## 'data.frame': 1389 obs. of  8 variables:
##  $ Entrez              : int  3199 3213 ...
##  $ Gene_Symbol         : chr  "HOXA2" ...
##  $ Amino_Acid_Letter  : chr  "L" ...
##  $ Amino_Acid_Position: num  298 123 ...
##  $ Amino_Acid_Change  : chr  "L298M" ...
##  $ Mutation_Type      : chr  "Missense_Mutation" ...
##  $ Sample             : chr  "SA236" ...
##  $ Tumor_Type         : chr  "brca" ...
```

Alternatively, setup accepts the parameter repos too.

5 Statistics

In this step we calculate the general statistics for the entire consensus profile

```
lm <- entropy(lm)

## Making uniform model...
## Assigned bandwidth:  0

#Global Score
str(lmEntropy(lm))

## List of 5
##  $ bw       : num 0
##  $ uniform  :function (nmut)
##  $ absval   : num 3.6
##  $ log10pval: num -11.3
##  $ pvalue   : num 4.68e-12

#Per position score
head(lmAlignment(lm)$df)

##   consensus conservation       mean        lTsh        uTsh    profile
## 1         R    0.3264101 0.01510504 0.005401724 0.02876578 0.02941176
## 2         R    0.4718685 0.01727311 0.005826839 0.03364113 0.03781513
## 3         A    0.1564765 0.01691597 0.006026568 0.03226212 0.01260504
## 4         R    0.8827896 0.01674370 0.005812086 0.03225727 0.16386555
## 5         T    0.6524318 0.01740756 0.006131861 0.03334664 0.01680672
## 6         A    0.3221084 0.01717647 0.006089608 0.03282145 0.01260504
##         pvalue       qvalue misalnFreq
## 1 4.434780e-02 4.286742e-01          0
## 2 2.604874e-02 3.711945e-01          0
## 3 6.599899e-01 1.000000e+00          0
## 4 5.774242e-14 3.291318e-12          0
## 5 4.630216e-01 1.000000e+00          0
## 6 6.693102e-01 1.000000e+00          0
```

With the method entropy, we calculate the entropy score and a pvalue against the null hypothesis that the mutations are distributed uniformly. In addition, a test is performed for every position of the consensus and the output is reported in the slot `alignment`. The position 4 has a conservation score of 0.88 (highly conserved) and the corrected pvalue is significant (qvalue below 0.01). There are signs of positive selection for the position 4. To retrieve the original mutations that generated that cluster, we can use the function lfm

```
significant_muts <- lfm(lm)
#Display original mutations that formed significant clusters (column Multiple_Aln_pos)
head(significant_muts)

##   Gene_Symbol Amino_Acid_Position Amino_Acid_Change          Sample
## 1        ALX4                 218             R218Q TCGA-AA-3949-01
## 2        CDX4                 177             R177C TCGA-D3-A2JO-06
## 3        CDX4                 177             R177C TCGA-AP-A0LM-01
## 4        CDX4                 177             R177C   MEL-Ma-Mel-85
## 5        CUX1                1248            R1248W TCGA-ER-A193-06
## 6        CUX1                1248            R1248W TCGA-BG-A18B-01
##   Tumor_Type Envelope_Start Envelope_End Multiple_Aln_pos       metric
## 1   coadread            215          271                4 3.291318e-12
## 2       skcm            174          230                4 3.291318e-12
## 3       ucec            174          230                4 3.291318e-12
## 4       skcm            174          230                4 3.291318e-12
## 5       skcm           1245         1301                4 3.291318e-12
## 6       ucec           1245         1301                4 3.291318e-12
##   Entrez  Entry    UNIPROT Chromosome                       Protein.name
## 1  60529 Q9H161 ALX4_HUMAN    11p11.2 Homeobox protein aristaless-like 4
## 2   1046 O14627 CDX4_HUMAN     Xq13.2             Homeobox protein CDX-4
## 3   1046 O14627 CDX4_HUMAN     Xq13.2             Homeobox protein CDX-4
## 4   1046 O14627 CDX4_HUMAN     Xq13.2             Homeobox protein CDX-4
## 5   1523 P39880 CUX1_HUMAN     7q22.1        Homeobox protein cut-like 1
## 6   1523 P39880 CUX1_HUMAN     7q22.1        Homeobox protein cut-like 1

#What are the genes mutated in position 4 in the consensus?
cluster_4_genes <- significant_muts[ significant_muts$Multiple_Aln_pos==4 , 'Gene_Symbol']
```

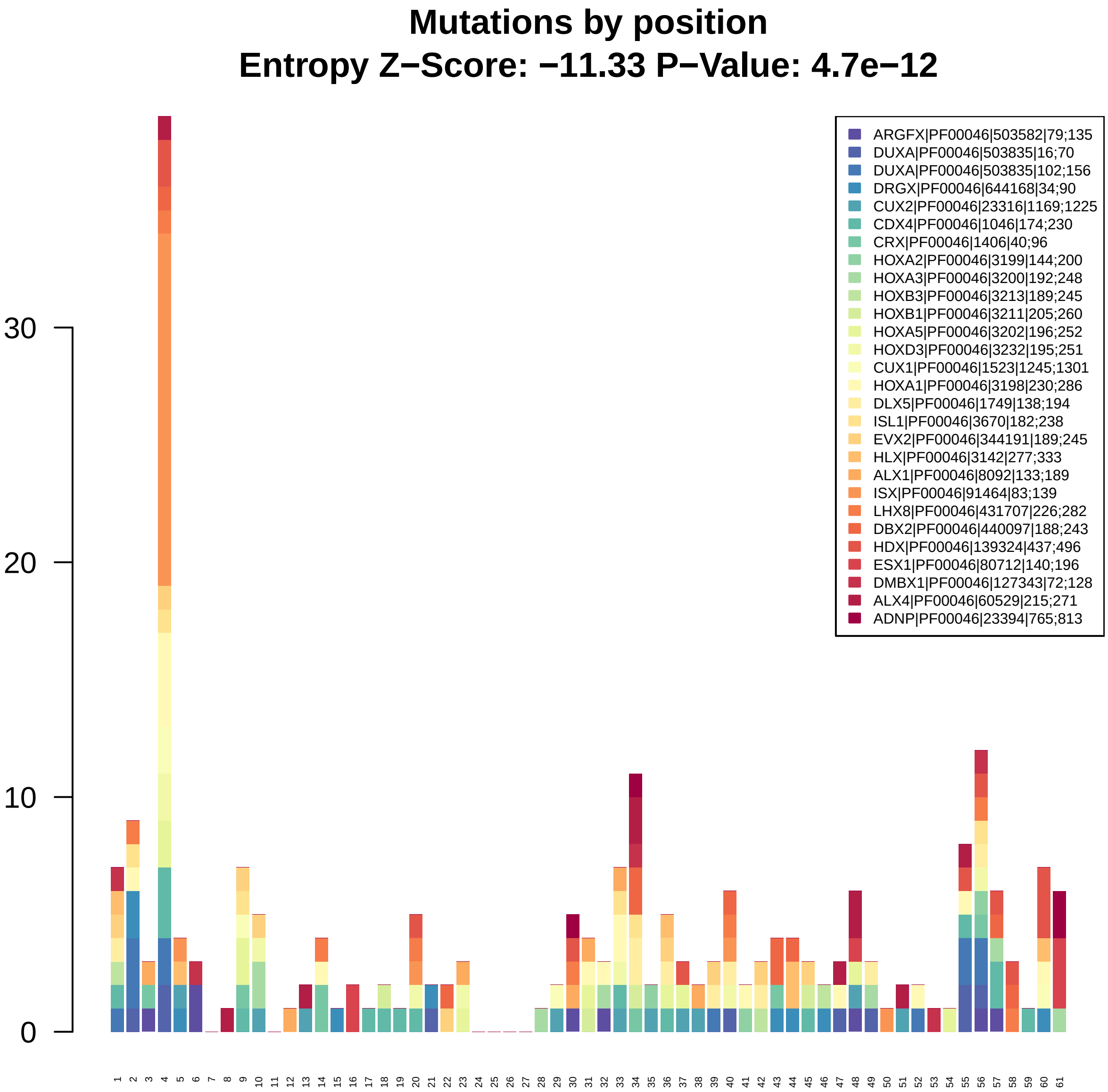
```
sort(table(cluster_4_genes))

## cluster_4_genes
##  ALX4  DBX2  EVX2  ISL1  LHX8  CUX1   HDX HOXA5 HOXD3  CDX4  DUXA HOXA1
##     1     1     1     1     1     2     2     2     2     3     4     4
##   ISX
##    15
```

The position 4 accounts for mutations in 13 different genes. The most represented one is ISX (ISX_HUMAN, Intestine-specific homeobox protein).

6 Plot

6.1 Consensus Bar Plot

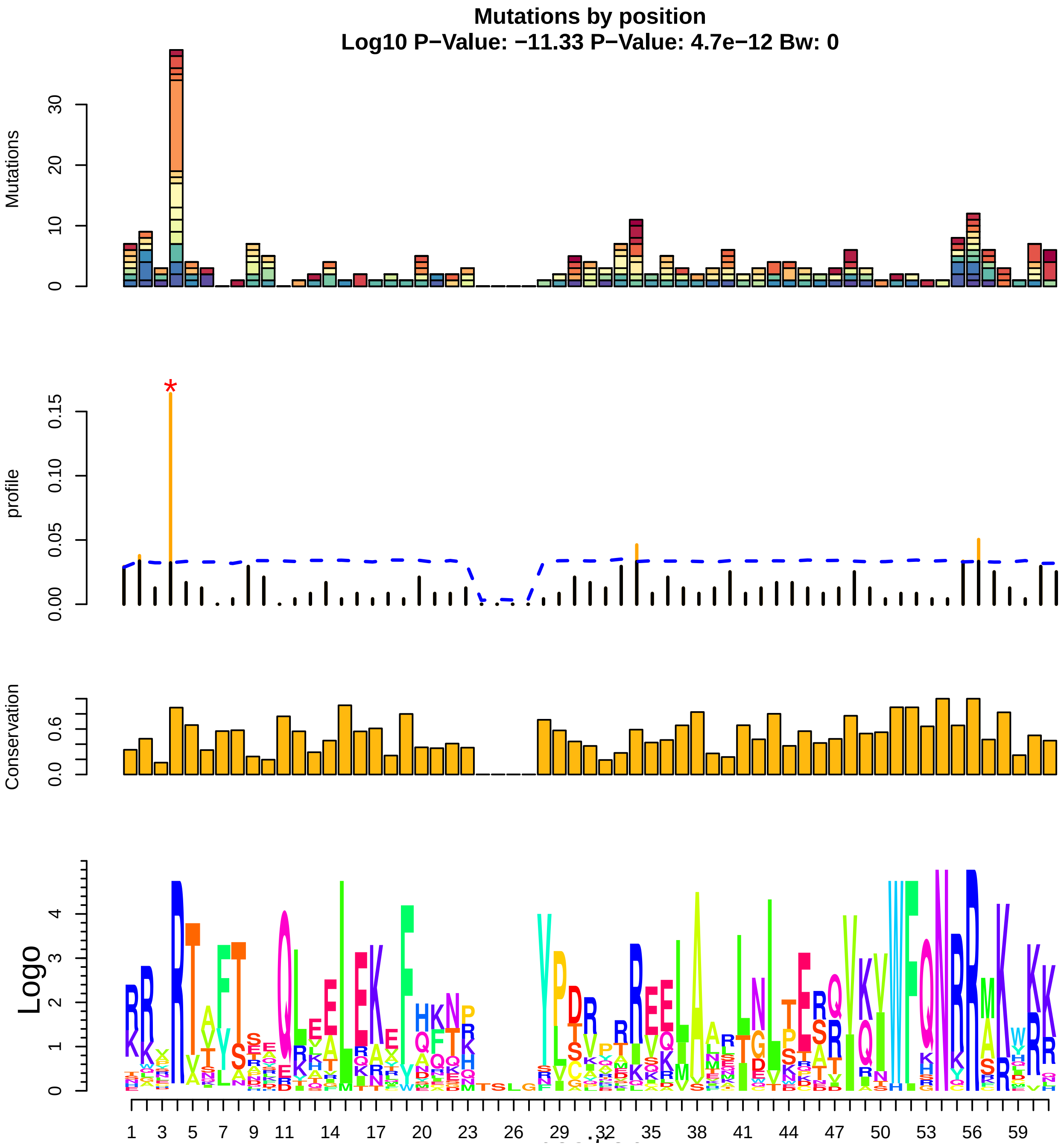
```
bpAll(lm)
```



This barplot shows all the mutations reported on the consensus sequence divided by protein/pfam domain

6.2 LowMACA comprehensive Plot

```
lmPlot(lm)
```



This four layer plot encompasses:

- The bar plot visualize before
- The distribution of mutations against the null hypothesis (blue line) with orange bars representing a pvalue below 0.05 for that position and a red star for qvalue below 0.05
- The Trident score distribution
- The logo plot representing the consensus sequence

6.3 Protter plot

```
#This plot is saved as a png image
protter(lm , filename="homeobox.png")
```

```
## [1] TRUE
```

A request to the Protter server is sent and a png file is downloaded with the possible sequence structure of the protein and the significant positions colored in orange and red

7 Summary

Copy and paste on your R console and perform the entire analysis by yourself. You need Ghostscript to see all the plots.

```
library(LowMACA)
Genes <- c("ADNP","ALX1","ALX4","ARGFX","CDX4","CRX"
                        ,"CUX1","CUX2","DBX2","DLX5","DMBX1","DRGX"
                        ,"DUXA","ESX1","EVX2","HDX","HLX","HNF1A"
                        ,"HOXA1","HOXA2","HOXA3","HOXA5","HOXB1","HOXB3"
                        ,"HOXD3","ISL1","ISX","LHX8")
Pfam <- "PF00046"
lm <- newLowMACA(genes=Genes , pfam=Pfam)
lmParams(lm)$tumor_type <- c("skcm" , "stad" , "ucec" , "luad" , "lusc" , "coadread" , "brca")
lmParams(lm)$min_mutation_number <- 0
lm <- setup(lm , mail="lowmaca@gmail.com")
lm <- entropy(lm)
lfm(lm)
bpAll(lm)
lmPlot(lm)
protter(lm)
```

8 Session Information

```
sessionInfo()

## R version 3.2.0 (2015-04-16)
## Platform: x86_64-unknown-linux-gnu (64-bit)
## Running under: Ubuntu 14.04.2 LTS
##
## locale:
##  [1] LC_CTYPE=en_US.UTF-8       LC_NUMERIC=C
##  [3] LC_TIME=en_US.UTF-8        LC_COLLATE=C
##  [5] LC_MONETARY=en_US.UTF-8    LC_MESSAGES=en_US.UTF-8
##  [7] LC_PAPER=en_US.UTF-8       LC_NAME=C
##  [9] LC_ADDRESS=C               LC_TELEPHONE=C
## [11] LC_MEASUREMENT=en_US.UTF-8 LC_IDENTIFICATION=C
##
## attached base packages:
##  [1] grid      stats4    parallel  stats     graphics  grDevices utils
##  [8] datasets  methods   base
##
## other attached packages:
##  [1] LowMACA_1.0.0       motifStack_1.12.0   ade4_1.7-2
##  [4] MotIV_1.24.0        grImport_0.9-0      XML_3.98-1.1
##  [7] Biostrings_2.36.0   XVector_0.8.0       IRanges_2.2.0
## [10] S4Vectors_0.6.0     BiocGenerics_0.14.0
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.11.5              knitr_1.9
##  [3] GenomicRanges_1.20.0     zlibbioc_1.14.0
##  [5] GenomicAlignments_1.4.0  BiocParallel_1.2.0
##  [7] BSgenome_1.36.0          lattice_0.20-31
##  [9] plyr_1.8.1               stringr_0.6.2
## [11] highr_0.4.1              GenomeInfoDb_1.4.0
## [13] tools_3.2.0              data.table_1.9.4
## [15] R.oo_1.19.0              seqLogo_1.34.0
## [17] lambda.r_1.1.7           futile.logger_1.4
## [19] LowMACAAnnotation_0.99.3 reshape2_1.4.1
## [21] RColorBrewer_1.1-2       cgdsr_1.1.33
## [23] rtracklayer_1.28.0       formatR_1.1
## [25] futile.options_1.0.0     bitops_1.0-6
## [27] RCurl_1.95-4.5           rGADEM_2.16.0
## [29] evaluate_0.6             R.methodsS3_1.7.0
## [31] Rsamtools_1.20.0         chron_2.3-45
```