maftools : Summarize, Analyze and Visualize MAF files.

2016-03-28

Introduction.

With advances in Cancer Genomics, Mutation Annotation Format (MAF) is being widley accepted and used to store somatic variants detected. The Cancer Genome Atlas Project has seqenced over 30 different cancers with sample size of each cancer type being over 200. Resulting data consisting of somatic variants is stored in the form of Mutation Annotation Format. This package attempts to summarize, analyze, annotate and visualize MAF files in an efficient manner from either TCGA sources or any in-house studies as long as the data is in MAF format.

MAF files contain many fields ranging from chromosome names to cosmic annotations. However most of the analysis in maftools uses following fields.

- Mandatoty fields: **Hugo_Symbol, Chromosome, Start_Position, End_position, Variant_Classification, Variant_Type and Tumor_Sample_Barcode**.
- Recommended optional fields: non MAF specific fields containing vaf and amino acid change information.

Complete specifiifcation of MAF files can be found on NCI TCGA page.

This vignette demonstrates the usage and application of maftools on an example MAF file from TCGA LAML cohort[1].

Installation

```
#Install Bioconductor dependencies.
source("http://bioconductor.org/biocLite.R")
biocLite("ComplexHeatmap")
biocLite("VariantAnnotation")
biocLite("Biostrings")

#Install maftools from github repository.
library("devtools")
install_github(repo = "PoisonAlien/maftools")
```

Reading maf files.

`read.maf` reads MAF files, summarizes it in various ways and stores it as an MAF object.

```
suppressWarnings(require(maftools))
#read TCGA maf file for LAML
laml.maf = system.file('extdata', 'tcga_laml.maf.gz', package = 'maftools')
laml = read.maf(maf = laml.maf, removeSilent = T, useAll = F)
```

Summarized MAF file is stored as an MAF object. MAF object contains main maf file, summarized data and oncomatrix which is useful to plot oncoplots (aka waterfall plots). There are accessor methods to access the useful slots from MAF object. However, all slots can be accessed using `@`, just like most of S4 objects.

```
#Typing laml shows basic summary of MAF file.
laml
```

```
## An object of class  MAF
##                    ID          summary       Mean Median
##  1:        NCBI_Build               37         NA     NA
##  2:            Center genome.wustl.edu         NA     NA
##  3:           Samples              192         NA     NA
##  4:   Frame_Shift_Del               52 0.27083333      0
##  5:   Frame_Shift_Ins               91 0.47395833      0
##  6:      In_Frame_Del               10 0.05208333      0
##  7:      In_Frame_Ins               42 0.21875000      0
##  8: Missense_Mutation             1342 6.98958333      7
##  9: Nonsense_Mutation              103 0.53645833      0
## 10:       Splice_Site               92 0.47916667      0
## 11:              total             1732 9.02083333      9
```

```
#Shows sample summry.
getSampleSummary(laml)
```

```
##      Tumor_Sample_Barcode Frame_Shift_Del Frame_Shift_Ins In_Frame_Del
##   1:         TCGA.AB.3009               0               5            0
##   2:         TCGA.AB.2807               1               0            1
##   3:         TCGA.AB.2959               0               0            0
##   4:         TCGA.AB.3002               0               0            0
##   5:         TCGA.AB.2849               0               1            0
##  ---
## 188:         TCGA.AB.2933               0               0            0
## 189:         TCGA.AB.2942               0               0            0
## 190:         TCGA.AB.2946               0               0            0
## 191:         TCGA.AB.2954               0               0            0
## 192:         TCGA.AB.2982               0               0            0
##      In_Frame_Ins Missense_Mutation Nonsense_Mutation Splice_Site total
##   1:            1                25                 2           1    34
##   2:            0                16                 3           4    25
##   3:            0                22                 0           1    23
##   4:            0                15                 1           5    21
##   5:            0                16                 1           2    20
##  ---
## 188:            0                 1                 0           0     1
## 189:            1                 0                 0           0     1
## 190:            0                 1                 0           0     1
## 191:            0                 1                 0           0     1
## 192:            0                 1                 0           0     1
```

```
#Shows frequently mutated genes.
getGeneSummary(laml)
```

```
##       Hugo_Symbol Frame_Shift_Del Frame_Shift_Ins In_Frame_Del
##    1:      DNMT3A               4               0            0
##    2:        FLT3               0               0            1
##    3:        NPM1               0              33            0
##    4:        TET2              10               4            0
##    5:        IDH2               0               0            0
##   ---
## 1237:      ZNF689               0               0            0
```

```
## 1238:      ZNF75D               0                0           0
## 1239:      ZNF827               1                0           0
## 1240:       ZNF99               0                0           0
## 1241:        ZPBP               0                0           0
##       In_Frame_Ins Missense_Mutation Nonsense_Mutation Splice_Site total
##    1:            0                39                 5           6    54
##    2:           33                15                 0           3    52
##    3:            0                 1                 0           0    34
##    4:            0                 4                 8           1    27
##    5:            0                20                 0           0    20
##   ---
## 1237:            0                 1                 0           0     1
## 1238:            0                 1                 0           0     1
## 1239:            0                 0                 0           0     1
## 1240:            0                 1                 0           0     1
## 1241:            0                 1                 0           0     1
##       MutatedSamples
##    1:             48
##    2:             52
##    3:             33
##    4:             17
##    5:             20
##   ---
## 1237:              1
## 1238:              1
## 1239:              1
## 1240:              1
## 1241:              1
```

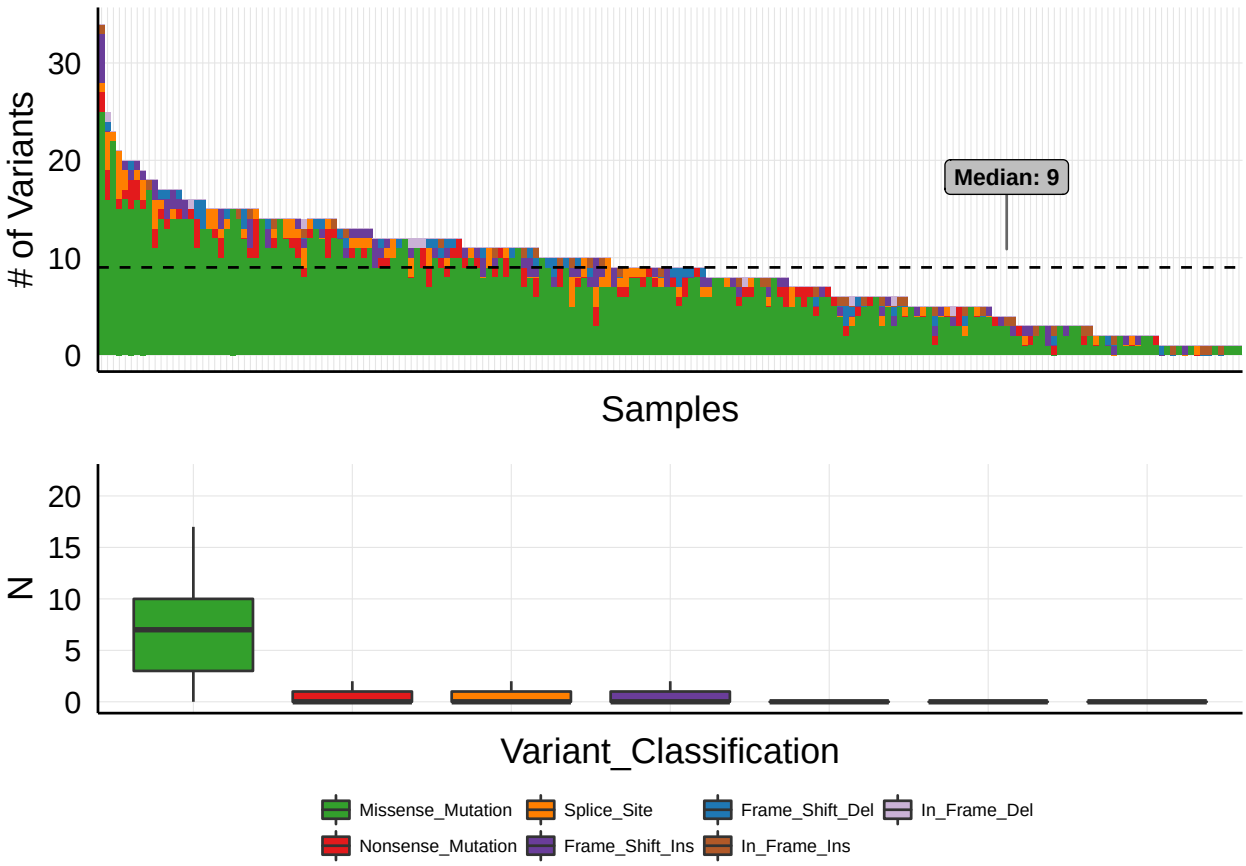
```
#Writes maf summary to an output file with basename laml.
write.mafSummary(maf = laml, basename = 'laml')
```

Plotting MAF summary.

We can use `plotmafSummary` to plot the summary of the maf file, which displays number of variants in each sample as a stacked barplot and variant types as a boxplot summarized by Variant_Classification. We can add either mean or median line to the stacked barplot to displat average/median number of variants across the cohort.

```
plotmafSummary(maf = laml, rmOutlier = T, addStat = 'median')
```

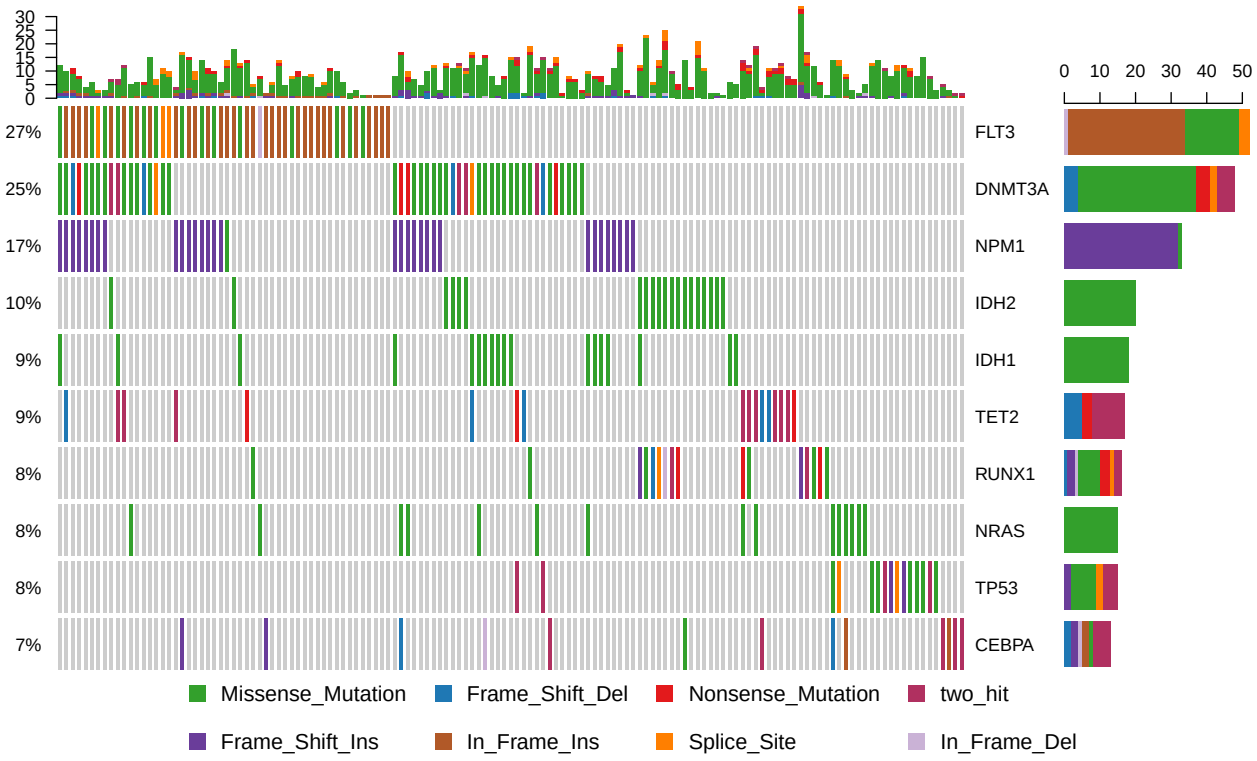
```
## Warning: Removed 1 rows containing non-finite values (stat_boxplot).
```



Oncoplots

Bettter representaion of maf file can be shown as oncoplots, also known as waterfall plots. Oncoplot function uses ComplexHeatmap to draw oncoplots. Side barplot and top barplots can be controlled by `drawRowBar` and `drawColBar` arguments respectivelly.

```
#We will draw oncoplots for top ten mutated genes. (Removing non-mutated samples from top ten genes for better visu
oncoplot(maf = laml, top = 10, removeNonMutated = T)
```

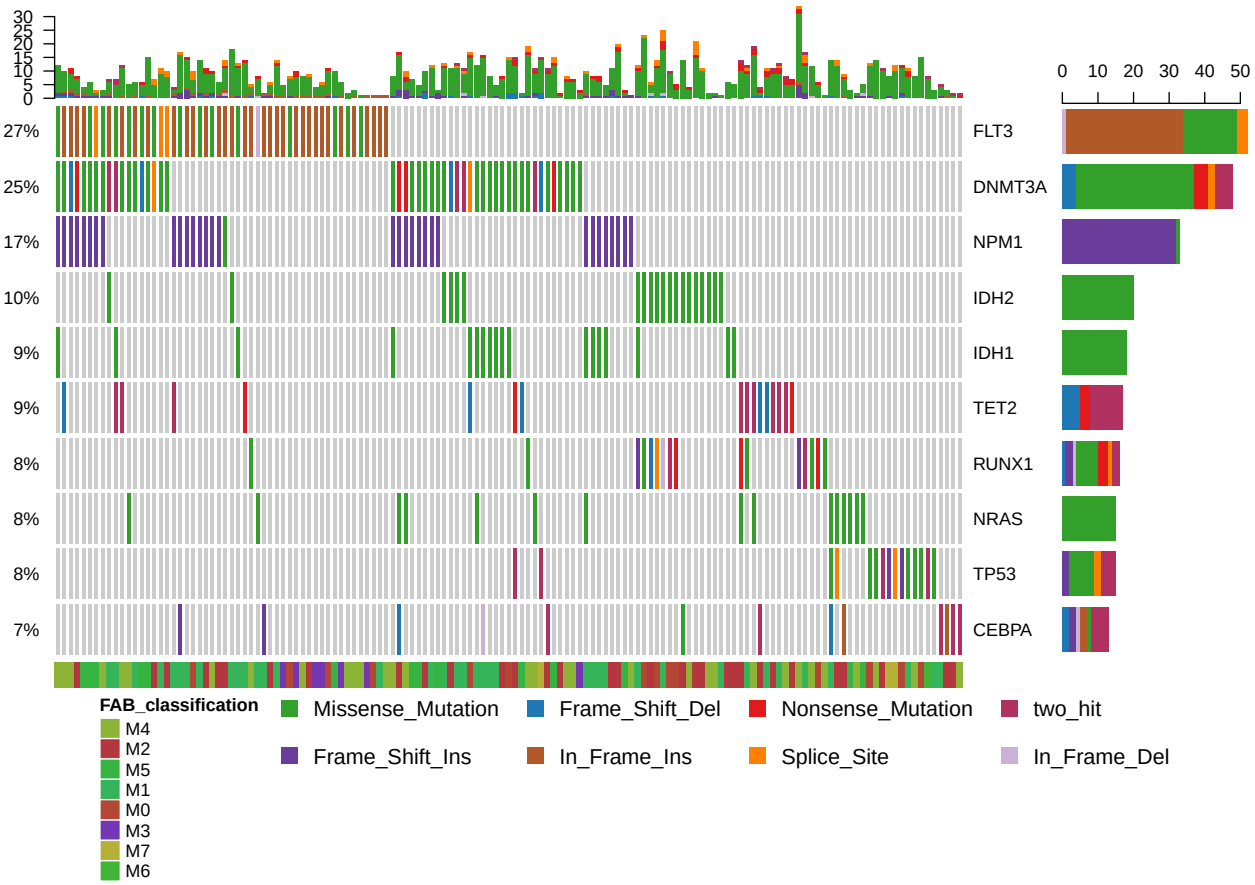


It is often the case that we include meta data to show sample characterstics such as gender, treatment, etc. We can include such meta data by passing them to `annotation` argument of `oncoplot`.

```
#Read FAB classification of TCGA LAML barcodes.
laml.fab.anno = system.file('extdata', 'tcga_laml_fab_annotation.txt', package = 'maftools')
laml.fab.anno = read.delim(laml.fab.anno, sep = '\t')
head(laml.fab.anno)
```

```
##   Tumor_Sample_Barcode FAB_classification
## 1         TCGA-AB-2802                 M4
## 2         TCGA-AB-2803                 M3
## 3         TCGA-AB-2804                 M3
## 4         TCGA-AB-2805                 M0
## 5         TCGA-AB-2806                 M1
## 6         TCGA-AB-2807                 M1
```

```
#We will plot same top ten mutated genes with FAB classification as annotation.
oncoplot(maf = laml, top = 10, annotation = laml.fab.anno, removeNonMutated = T)
```



Mutual exclusivity.

Many disease causing genes in cancer show strong exlcusiveness in their mutation pattern. Such mutually exlcuive set of genes can be detected using `mutExclusive` function which performs an exact test to detect such significant pair of genes. `mutExclusive` uses `comet_exact_test` on a given set of genes to calculate significance value. Please cite CoMET article if you use this function[2].

```
#We will run mutExclusive on top 10 mutated genes.
laml.mut.excl = mutExclusive(maf = laml, top = 10)
head(laml.mut.excl)
```

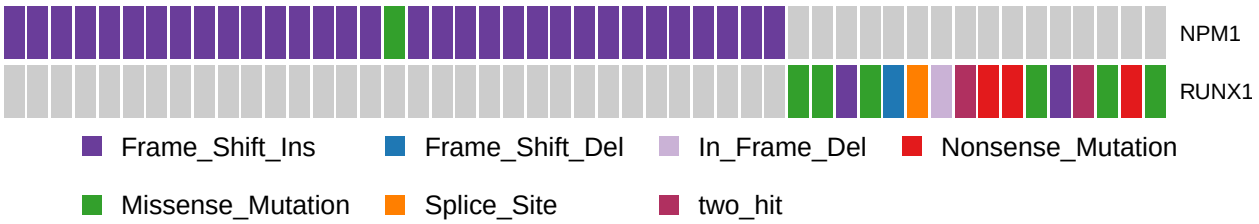
```
##   n.00 n.01 n.10 n.11  gene1 gene2               pval
## 1  125   15   52    0   FLT3  TP53 0.00352781328800287
## 2  139   20   33    0   NPM1  IDH2  0.0092131137152039
## 3  143   16   33    0   NPM1 RUNX1  0.0213098782919558
## 4  125   15   51    1   FLT3 RUNX1   0.021565502698412
## 5  144   15   33    0   NPM1  TP53  0.0261933920671912
## 6  129   15   47    1 DNMT3A RUNX1  0.0319088921944485
```

Oncoprint

We can visualize any set of genes using `oncoprint` function, which draws mutations in each sample similar to OncoPrinter tool on cBioPortal. For example in above `mutExclusive` analysis, we can see many genes show exclusiveness. For example NPM1 and RUNX1 show a strong exclusiveness with a p-value of 0.02. We can draw this pair of genes to

show this exclusiveness using `oncoprint`. `oncoprint` can be used to draw any number of genes using `top` or `genes` arguments.

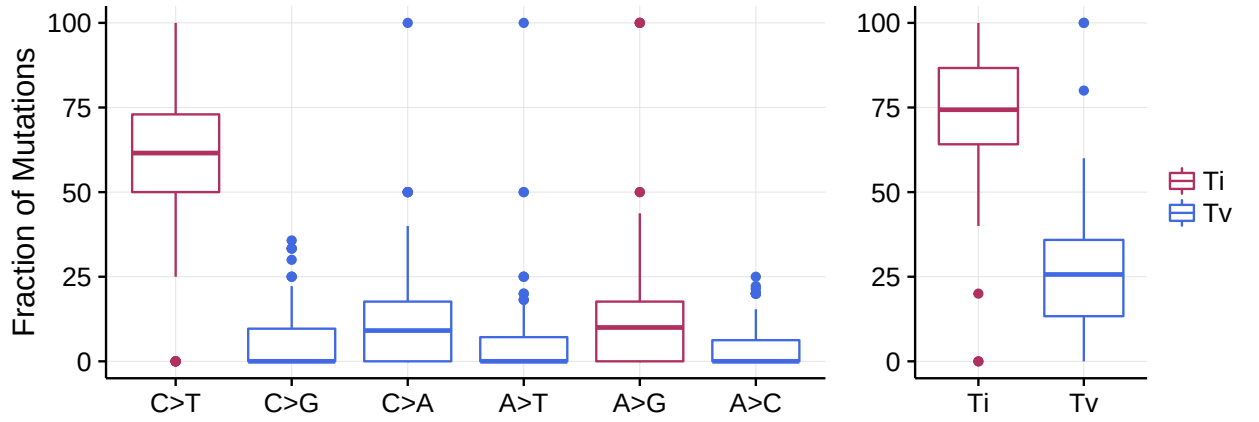
```
oncoprint(maf = laml, genes = c('NPM1', 'RUNX1'), removeNonMutated = T, showTumorSampleBarcodes = F)
```

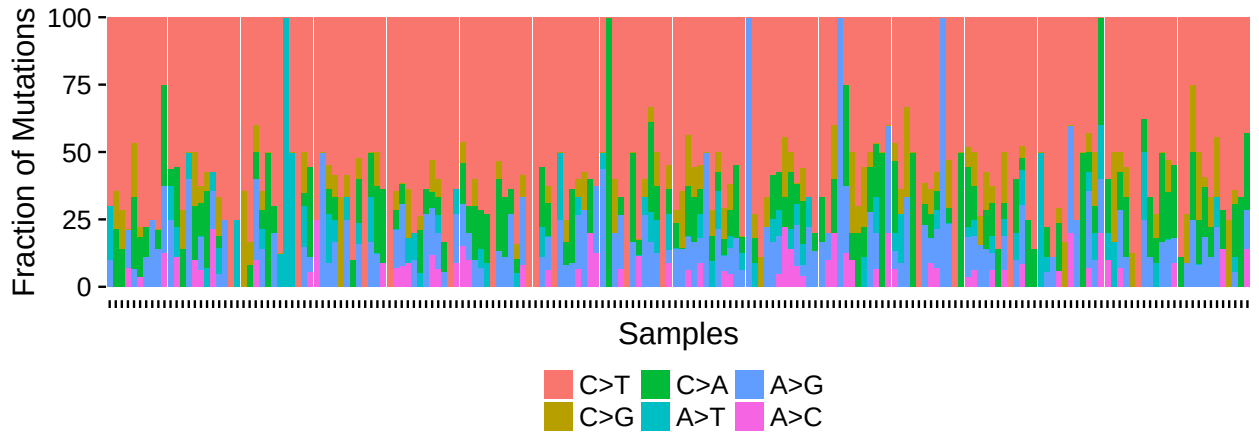


Transition and Transversions.

`titv` function classifies SNPs into Transitions and Transversions and returns a list of summarized tables in various ways. Summarized data can also be visulaized as a boxplot showing overall distribution of six different conversions and as a stacked barplot showing fraction of conversions in each sample.

```
laml.titv = titv(maf = laml, plot = F, useSyn = T)
#plot titv summary
plotTiTv(res = laml.titv)
```





Lollipop plots for amino acid changes.

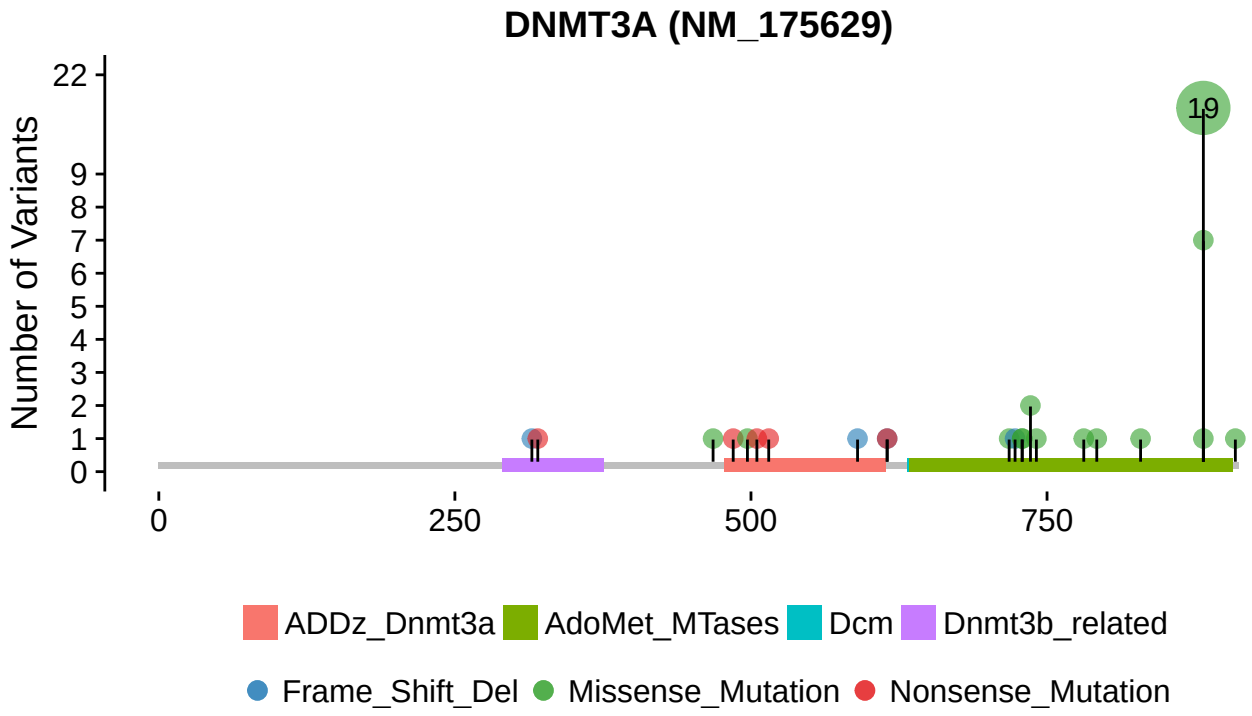
Lollipop plots are simple and most effective way showing mutation spots on protein structure. Many oncogenes have a preferential sites which are mutated more often than any other locus, which are considered to be mutational hotspots. We can draw such figures using the function `lollipopPlot`. This fuction requires us to have amino acid changes information in the maf file. However MAF files have no clear guidelines on naming the field for amino acid changes, with many different studies having different field (or column) names for amino acid changes. By default, `lollipopPlot` looks for column `AAChange`, and if its not found in the MAF file, it prints all availble fields with a warning message. For below example, MAF file contains amino acid changes under a field/column name 'Protein_Change'. We will manually specify this using argumnet `AACol`. This function also returns the plot as ggplot object, which user can later modify if needed.

```
#Lets plot lollipop plot for DNMT3A, which is one of the most frequent mutated gene Leukemia.
dnmt3a.lpop = lollipopPlot(maf = laml, gene = 'DNMT3A', AACol = 'Protein_Change')
```

```
## 3 transcripts available. Use arguments refSeqID or proteinID to manually specify tx name.
```

```
##       HGNC refseq.ID protein.ID aa.length Start End          Label
##  1: DNMT3A NM_175629  NP_783328       912   290 376 Dnmt3b_related
##  2: DNMT3A NM_175629  NP_783328       912   478 614    ADDz_Dnmt3a
##  3: DNMT3A NM_175629  NP_783328       912   632 795            Dcm
##  4: DNMT3A NM_175629  NP_783328       912   634 907  AdoMet_MTases
##  5: DNMT3A NM_022552  NP_072046       912   290 376 Dnmt3b_related
##  6: DNMT3A NM_022552  NP_072046       912   478 614    ADDz_Dnmt3a
##  7: DNMT3A NM_022552  NP_072046       912   632 795            Dcm
##  8: DNMT3A NM_022552  NP_072046       912   634 907  AdoMet_MTases
##  9: DNMT3A NM_153759  NP_715640       723   101 187 Dnmt3b_related
## 10: DNMT3A NM_153759  NP_715640       723   289 425    ADDz_Dnmt3a
## 11: DNMT3A NM_153759  NP_715640       723   443 606            Dcm
## 12: DNMT3A NM_153759  NP_715640       723   445 718  AdoMet_MTases
```

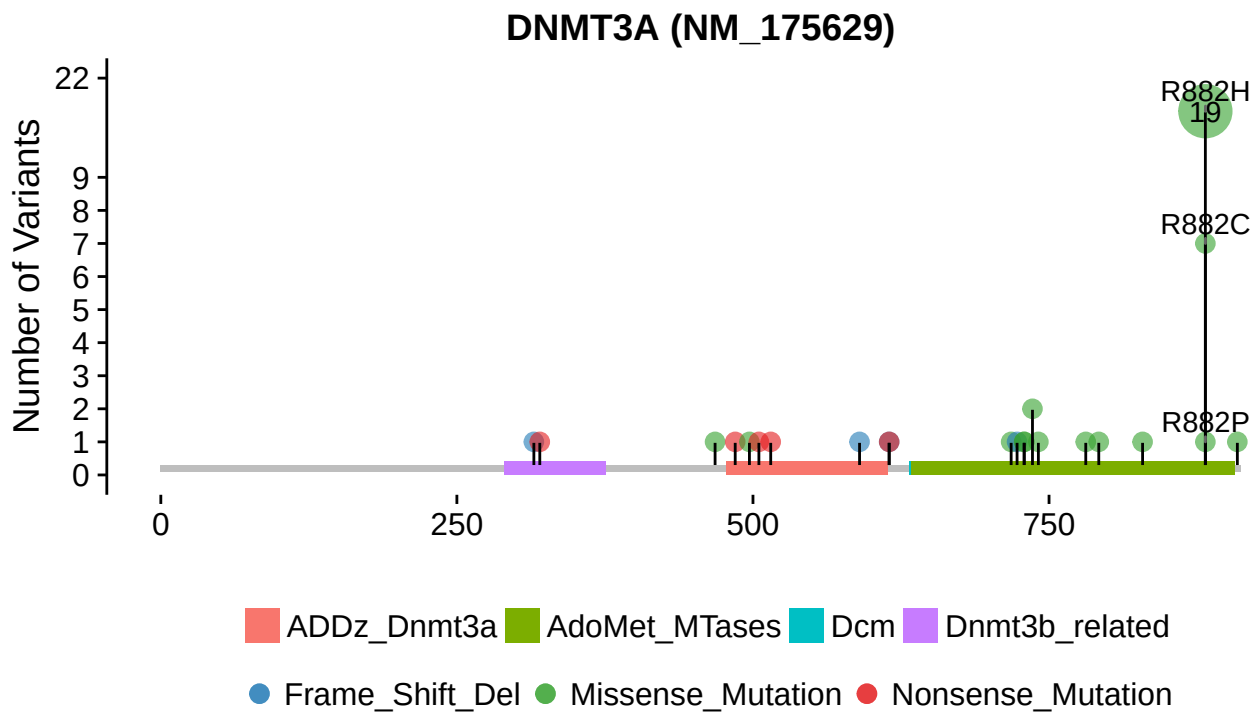
```
## Using longer transcript NM_175629 for now.
```



Note that `lollipopPlot` warns user on availability of diferent transcripts for the given gene. If we know the transcript id before hand, we can specify it as `refSeqID` or `proteinID`. By default lollipopPlot uses the longer isoform.

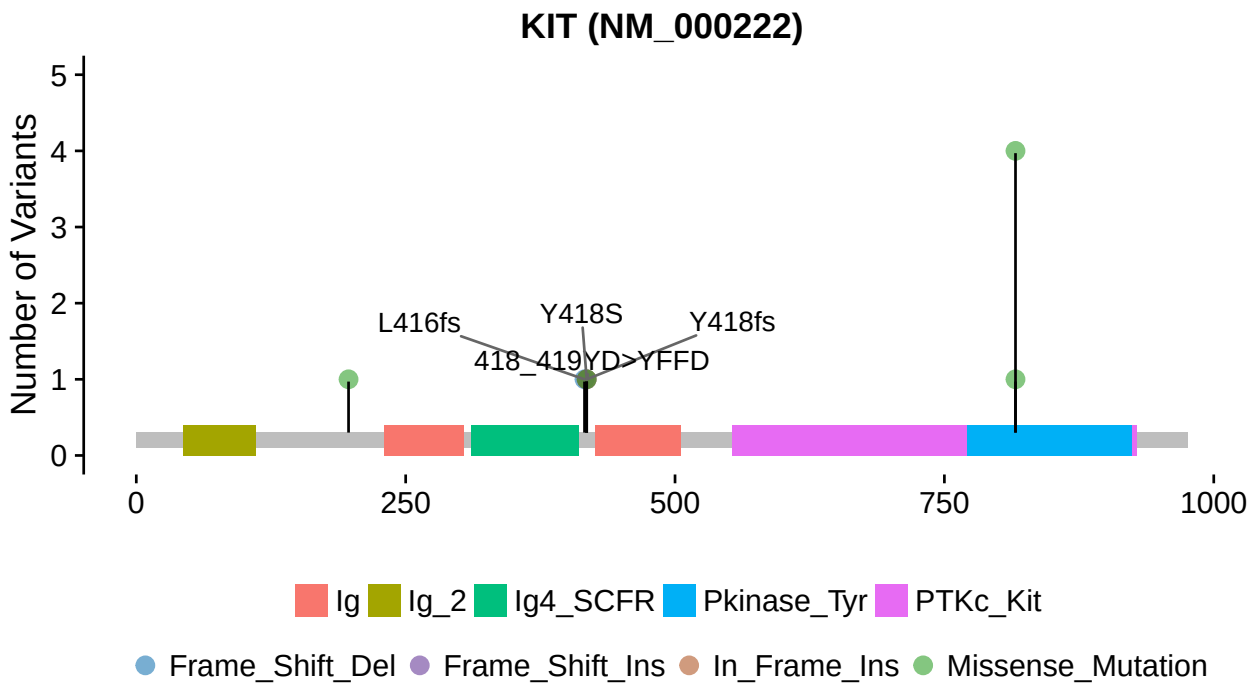
We can also label points on the `lollipopPlot` using argument `labelPos`. If `labelPos` is set to 'all', all of the points are highlighted, but it will make plot cluttery.

```
#Lets highlight pos 882 which is one of the hotspot.
dnmt3a.lpop = lollipopPlot(maf = laml, gene = 'DNMT3A', AACol = 'Protein_Change', labelPos = 882, refSeqID = 'NM_17
```

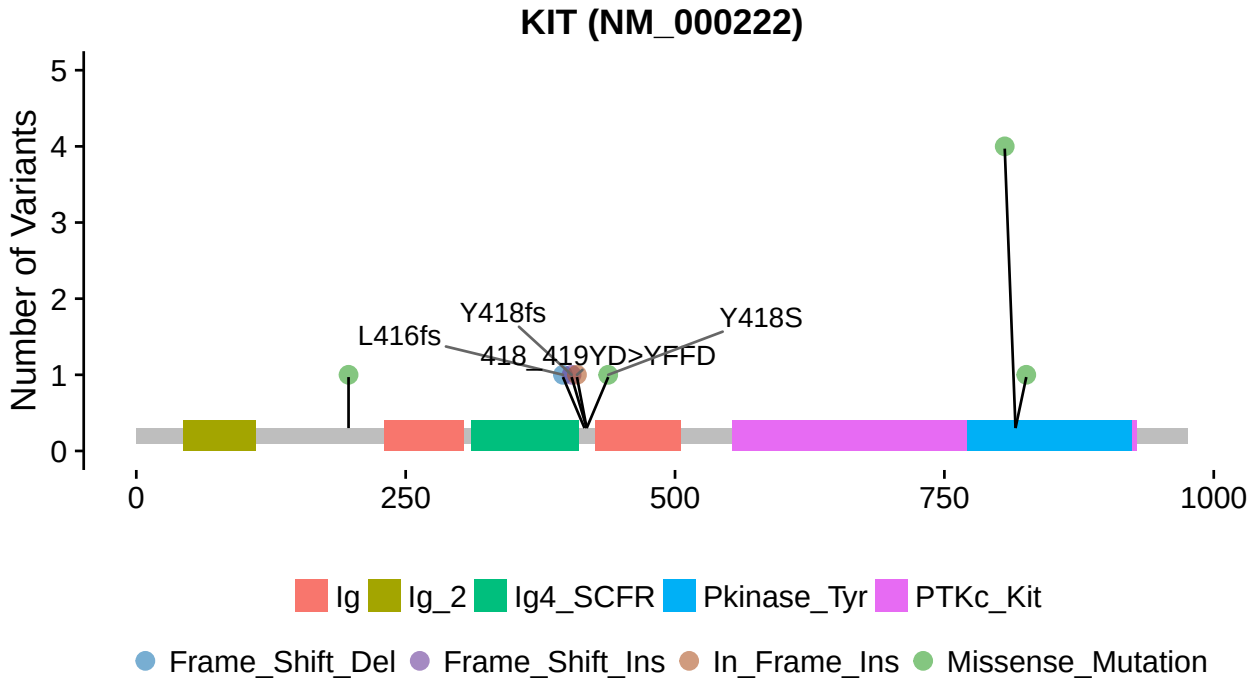


Sometimes, many mutations are clustered within a range of few amino acid positons. In that case we can use `repel` option which tries to repel points.

```
#Lets mutations on KIT gene, without repel option.
kit.lpop = lollipopPlot(maf = laml, gene = 'KIT', AACol = 'Protein_Change', labelPos = c(416, 418), refSeqID = 'NM_
```




```
#Same plot with repel=T
kit.lpop = lollipopPlot(maf = laml, gene = 'KIT', AACol = 'Protein_Change', labelPos = c(416, 418), refSeqID = 'NM_
```



Detecting cancer driver genes based on positional clustering.

maftools has a function `oncodrive` which identifies cancer genes (driver) from a given MAF. `oncodrive` is a based on algorithm oncodriveCLUST which was originally implemented in Python. Concept is based on the fact that most of

the variants in cancer causing genes are enriched at few specific loci (aka hotspots). This method takes advantage of such positions to identify cancer genes. If you use this function, please cite OncodriveCLUST article[3].

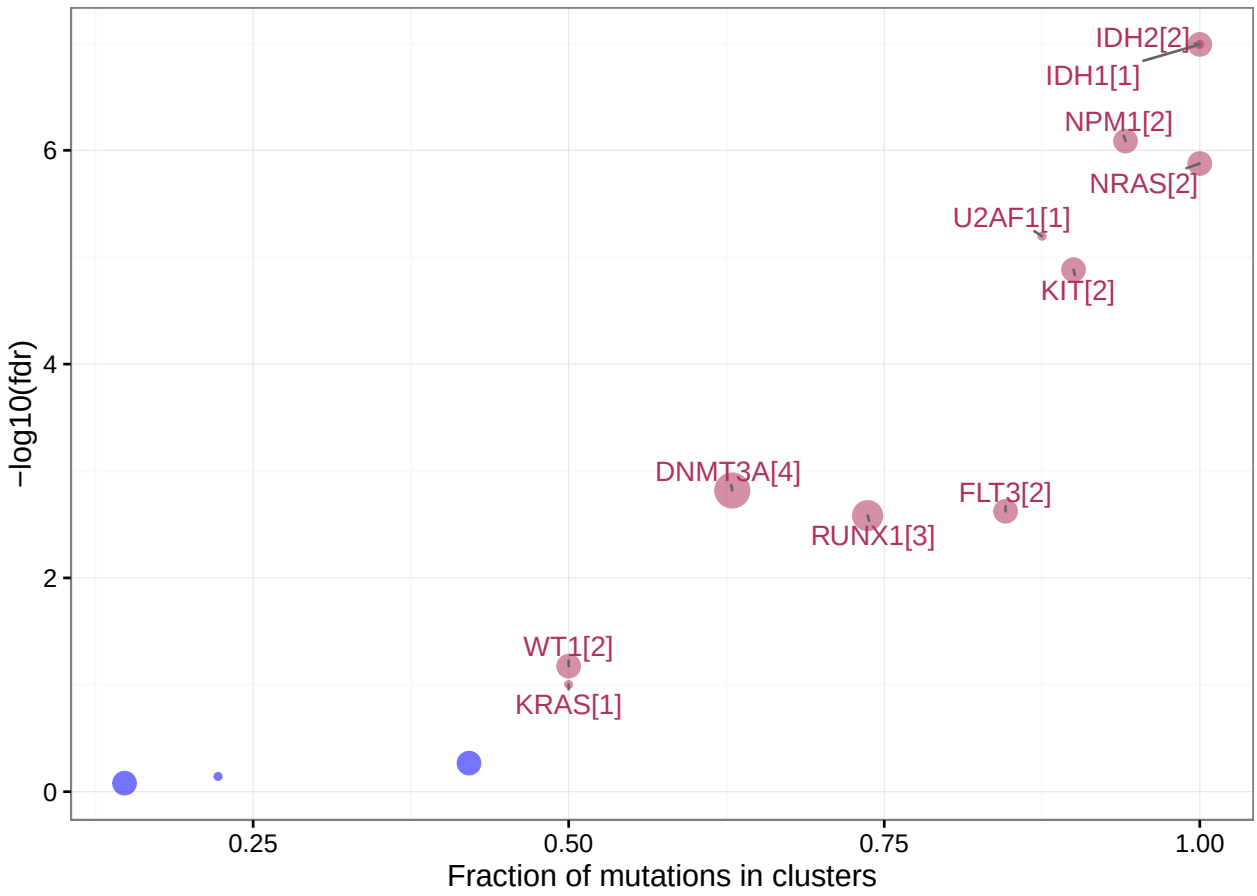
```
laml.sig = oncodrive(maf = laml, AACol = 'Protein_Change', minMut = 5, pvalMethod = 'zscore')
```

We can plot the results using `plotOncodrive`.

```
head(laml.sig)
```

```
##    Hugo_Symbol Frame_Shift_Del Frame_Shift_Ins In_Frame_Del In_Frame_Ins
## 1:        IDH1               0               0            0            0
## 2:        IDH2               0               0            0            0
## 3:        NPM1               0              33            0            0
## 4:        NRAS               0               0            0            0
## 5:       U2AF1               0               0            0            0
## 6:         KIT               1               1            0            1
##    Missense_Mutation Nonsense_Mutation Splice_Site total MutatedSamples
## 1:                18                 0           0    18             18
## 2:                20                 0           0    20             20
## 3:                 1                 0           0    34             33
## 4:                15                 0           0    15             15
## 5:                 8                 0           0     8              8
## 6:                 7                 0           0    10              8
##    clusters muts_in_clusters clusterScores protLen   zscore         pval
## 1:        1               18     1.0000000     414 5.546154 1.460110e-08
## 2:        2               20     1.0000000     452 5.546154 1.460110e-08
## 3:        2               32     0.9411765     294 5.093665 1.756034e-07
## 4:        2               15     0.9218951     189 4.945347 3.800413e-07
## 5:        1                7     0.8750000     240 4.584615 2.274114e-06
## 6:        2                9     0.8500000     976 4.392308 5.607691e-06
##             fdr fract_muts_in_clusters
## 1: 1.022077e-07              1.0000000
## 2: 1.022077e-07              1.0000000
## 3: 8.194826e-07              0.9411765
## 4: 1.330144e-06              1.0000000
## 5: 6.367520e-06              0.8750000
## 6: 1.308461e-05              0.9000000
```

```
plotOncodrive(res = laml.sig, fdrCutOff = 0.1, useFraction = T)
```

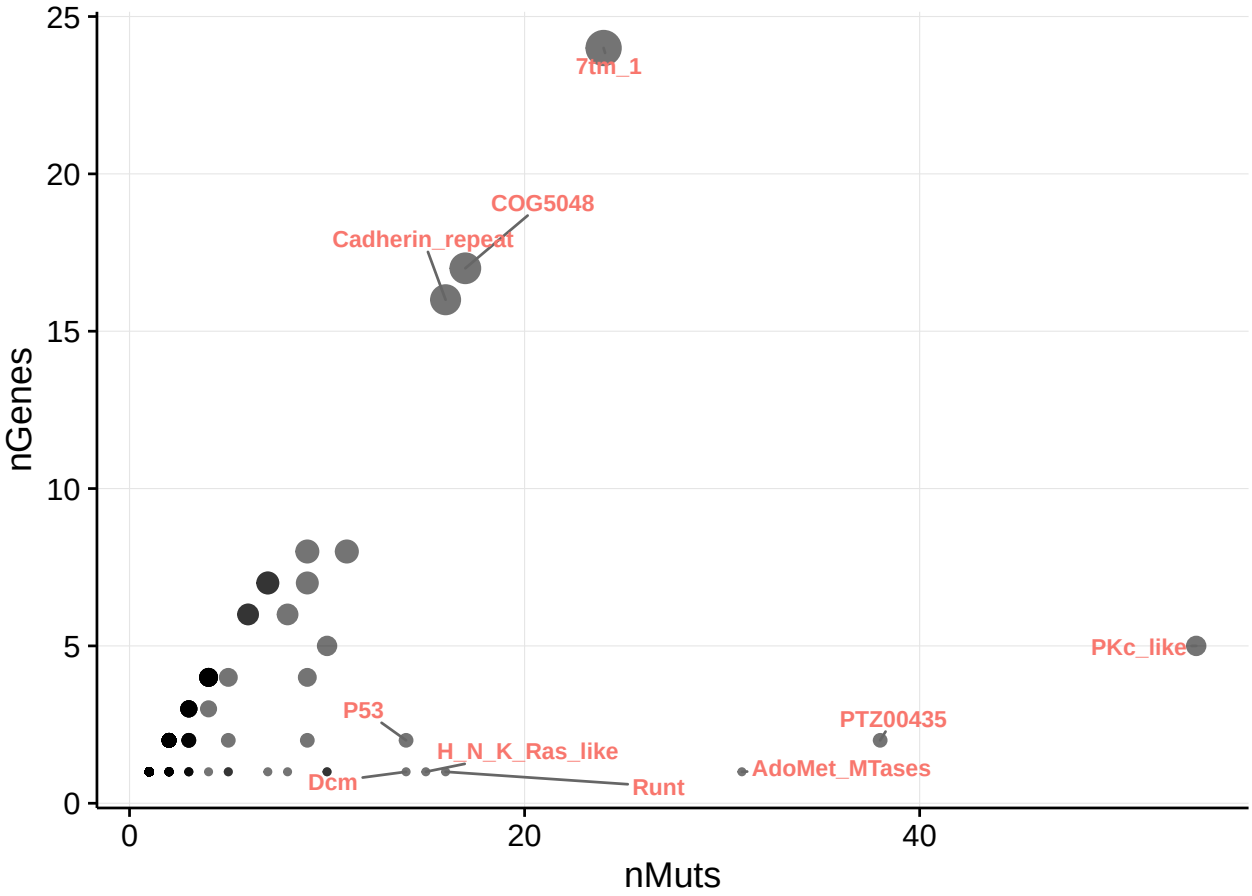


`plotOncodrive` plots the results as scatter plot with size of the points proportional to the number of clusters found in the gene. X-axis shows number of mutations (or fraction of mutations) observed in these clusters. In the above example, IDH1 has a single cluster and all of the 18 mutations are accumulated within that cluster, giving it a cluster score of one. For details on oncodrive algorithm, please refer to OncodriveCLUST article[3].

Adding pfam domains.

maftools comes with the function `pfamDomains`, which adds pfam domain information to the amino acid changes. `pfamDomain` also summarizes amino acid changes accoriding to the domains that are affected. This serves the puposes of knowing what domain in given cancer cohort, is most frequently affected. This function is inspired from Pfam annotation modulce from MuSic tool[4].

```
laml.pfam = pfamDomains(maf = laml, AACol = 'Protein_Change', top = 10)
```




```
#Protein summary (Printing first 7 columns for display convenience)
laml.pfam$proteinSummary[,1:7, with = F]
```

```
##         HGNC AAPos Variant_Classification  N total  fraction   DomainLabel
##    1: DNMT3A   882      Missense_Mutation 27    54 0.5000000 AdoMet_MTases
##    2:   IDH1   132      Missense_Mutation 18    18 1.0000000      PTZ00435
##    3:   IDH2   140      Missense_Mutation 17    20 0.8500000      PTZ00435
##    4:   FLT3   835      Missense_Mutation 14    52 0.2692308      PKc_like
##    5:   FLT3   599           In_Frame_Ins 10    52 0.1923077      PKc_like
##   ---
## 1470: ZNF646   875      Missense_Mutation  1     1 1.0000000            NA
## 1471: ZNF687   554      Missense_Mutation  1     2 0.5000000            NA
## 1472: ZNF687   363      Missense_Mutation  1     2 0.5000000            NA
## 1473: ZNF75D     5      Missense_Mutation  1     1 1.0000000            NA
## 1474: ZNF827   427        Frame_Shift_Del  1     1 1.0000000            NA
```

```
#Domain summary (Printing first 3 columns for display convenience)
laml.pfam$domainSummary[,1:3, with = F]
```

```
##        DomainLabel nMuts nGenes
##   1:      PKc_like    54      5
##   2:      PTZ00435    38      2
##   3: AdoMet_MTases    31      1
##   4:         7tm_1    24     24
##   5:       COG5048    17     17
```

```
##  ---
## 473:    ribokinase     1      1
## 474:   rim_protein     1      1
## 475: sigpep_I_bact     1      1
## 476:           trp     1      1
## 477:        zf-BED     1      1
```

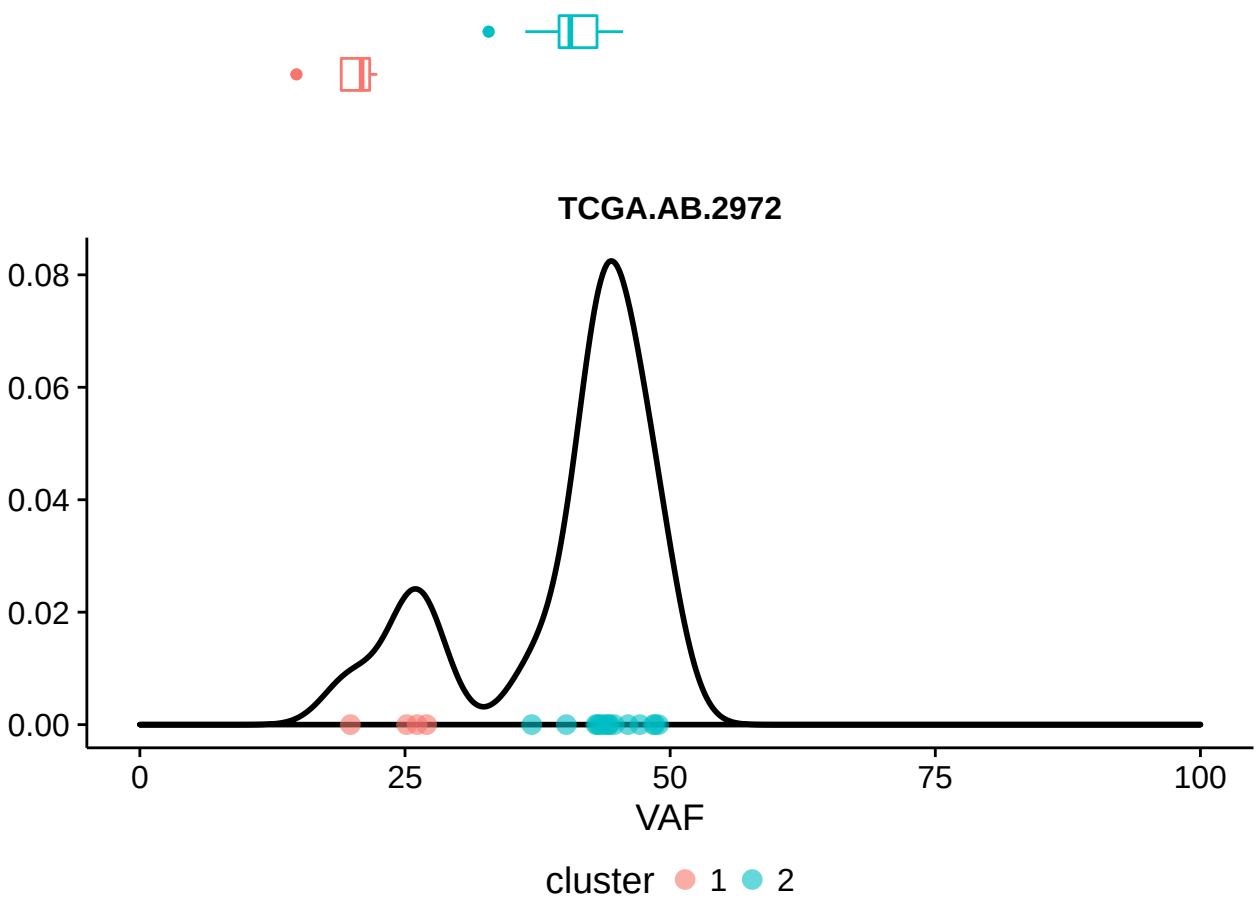
Above plot and results shows AdoMet_MTases domain is frequently mutated, but number genes with this domain is just one (DNMT3A) compared to other domains such as 7tm_1 domain, which is mutated across 24 different genes. This shows the importance of mutations in methyl transfer domains Leukemia.

Tumor heterogeneity and MATH scores.

Heterogeneity in tumor samples.

Tumors are generally heterogenous i.e, consist of multiple clones. This heterogenity can be inferred by clustering variant allele frequencies. `inferHeterogeneity` function uses vaf information to cluster variants (using `mclust`), to infer clonality. By default, `inferHeterogeneity` function looks for column *t_vaf* containing vaf information. However, if the field name is different from *t_vaf*, we can manually specify it using argument `vafCol`. For example, in this case study vaf is stored under the field name *i_TumorVAF_WU*. Although mlcust performs fairly well, it is recommended to try SciClone which does better job at clustering and density estimation[5].

```
#We will run this for sample TCGA.AB.2972
inferHeterogeneity(maf = laml, tsb = 'TCGA.AB.2972', vafCol = 'i_TumorVAF_WU')
```



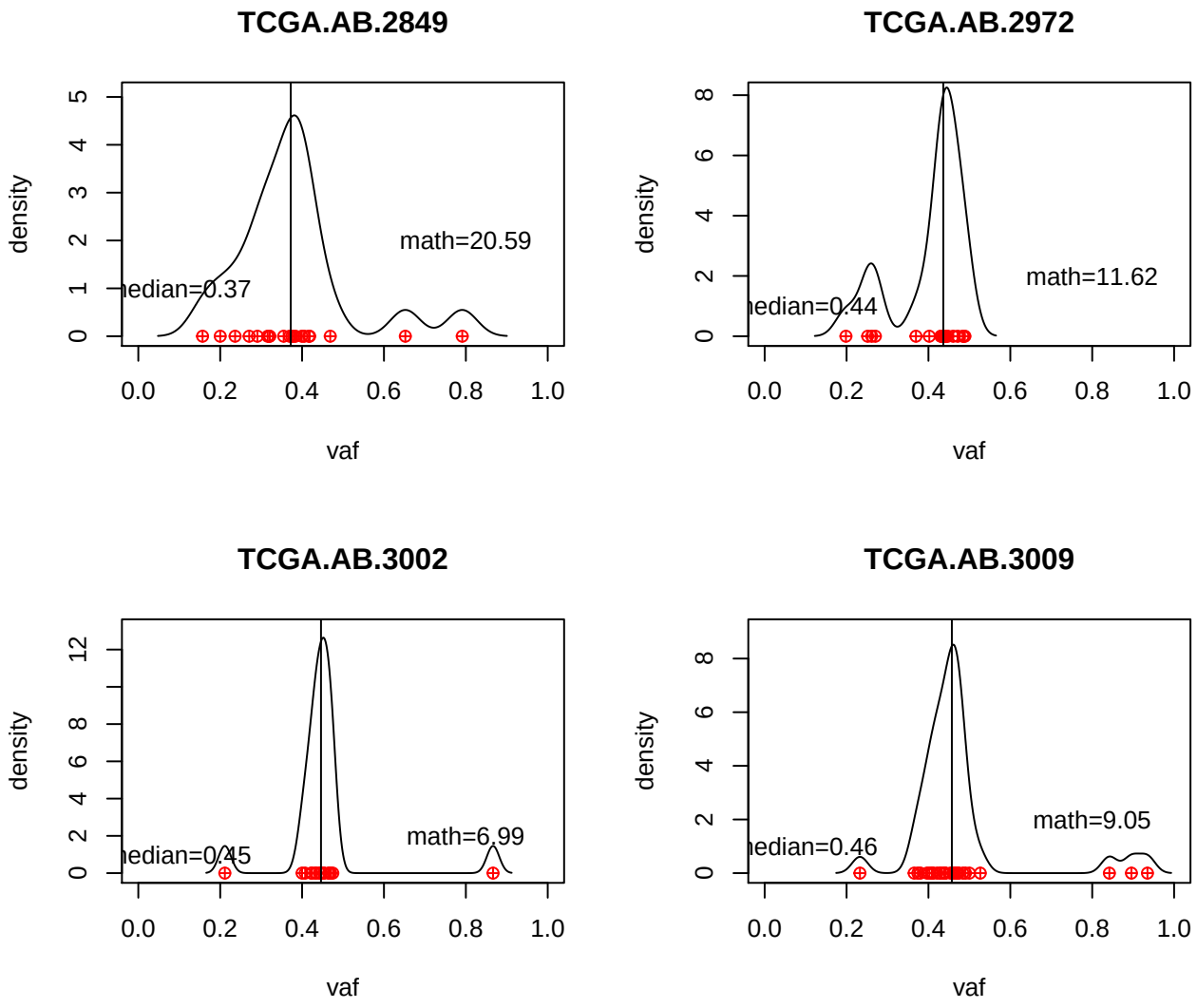
```
## $clusterData
##    Hugo_Symbol Tumor_Sample_Barcode t_vaf cluster
## 1         ASTL         TCGA.AB.2972 36.95       2
## 2       ATP1B4         TCGA.AB.2972 43.00       2
## 3    C10orf118         TCGA.AB.2972 48.43       2
## 4        DNAH3         TCGA.AB.2972 47.15       2
## 5        DNAH5         TCGA.AB.2972 44.73       2
## 6        DOCK2         TCGA.AB.2972 40.21       2
## 7        FANCI         TCGA.AB.2972 43.95       2
## 8        HMCN1         TCGA.AB.2972 48.58       2
## 9     KIAA0240         TCGA.AB.2972 43.63       2
## 10      LARP4B         TCGA.AB.2972 27.04       1
## 11       MORC3         TCGA.AB.2972 44.25       2
## 13      PTPN11         TCGA.AB.2972 25.16       1
## 14       RIMS1         TCGA.AB.2972 19.85       1
## 15      RNASEN         TCGA.AB.2972 44.22       2
## 16       SFRS6         TCGA.AB.2972 26.14       1
## 17       STAG2         TCGA.AB.2972 46.03       2
## 18       TUFT1         TCGA.AB.2972 43.28       2
## 19      ZC3H18         TCGA.AB.2972 43.15       2
## 20       ZNF43         TCGA.AB.2972 48.91       2
##
## $clusterMeans
##   Tumor_Sample_Barcode cluster  meanVaf
## 1         TCGA.AB.2972       1 24.54750
## 2         TCGA.AB.2972       2 44.43133
```

Above figure shows clear seperation of two clones clustered at mean variant allele frequencies of ~45% (major clone) and another minor clone at variant allele frequency of ~25%.

MATH (Mutant-Allele Tumor Heterogeneity) scores to infer extent of heterogeneity.

Although clustering of variant allele frequencies gives us a fair idea on heterogeneity, it is also possible to measure the extent of heterogeneity in terms of a numerical value. MATH score is a simple quantitative measure of intra-tumor heterogeneity, which calculates the width of the vaf distribution. Higher MATH scores are found to be associated with poor outcome. MATH score can also be used a proxy variable for survival analysis[6].

```
#we will specify for random 4 patients.
laml.math = math.score(maf = laml, vafCol = 'i_TumorVAF_WU',
                       sampleName = c('TCGA.AB.3009', 'TCGA.AB.2849', 'TCGA.AB.3002', 'TCGA.AB.2972'))
```




```
print(laml.math)
```

```
##    Tumor_Sample_Barcode      MATH MedianAbsoluteDeviation Frame_Shift_Del
## 1:         TCGA.AB.2849 20.588348                5.170000               0
## 2:         TCGA.AB.2972 11.621572                3.420000               0
## 3:         TCGA.AB.3002  6.991207                2.104062               0
## 4:         TCGA.AB.3009  9.045040                2.789054               0
##    Frame_Shift_Ins In_Frame_Del In_Frame_Ins Missense_Mutation
## 1:               1            0            0                16
## 2:               1            0            0                16
## 3:               0            0            0                15
## 4:               5            0            1                25
##    Nonsense_Mutation Splice_Site total
## 1:                 1           2    20
## 2:                 2           1    20
## 3:                 1           5    21
## 4:                 2           1    34
```

From the above results, sample TCGA.AB.2849 has highest of MATH score (20.58) compared to rest of the three samples. It is also evident from the density plot, that vaf distribution is wider for this sample, whereas rest of three samples have sharp peaks with relatively low MATH scores, suggesting more homogeneity and lesser heterogeneity.

Mutational Signatures.

Every cancer, as it progresses leaves a signature characterised by specific pattern of nucleotide substitutions. Alexandrov et.al have shown such mutational signatures, derived from over 7000 cancer samples. Such signatures can be extracted by decomposiong matrix of nucleotide substitutions, classified into 96 substitution classes based on immediate bases sorrouding the mutated base. Extracted signatures can also be compared to those 21 validated signatures.

`extractSignatures` uses non-negative matrix factorization to decompose nx96 dimesion matrix into r signatures[7]. By default function runs nmf on 6 ranks and chooses the best possible value based on maximum cophenetic-correlation coefficients. It is also possible to manually specify r. Once decomposed, signatures are compared against known 21 signatures derived from Alexandrov et.al, and correlation coeeficient is calculated to identify best match [8].

NOTE: Eventhough reading fasta and extracting bases is fairly fast, it is a memory consuming process as it occupies ~3gb of memory while running.

```
#First we extract adjacent bases to the mutated locus and clssify them into 96 substitution classes.
suppressPackageStartupMessages(require("VariantAnnotation", quietly = T))
laml.tnm = trinucleotideMatrix(maf = laml, ref_genome = '~/NGS/gatk_ref/hg19.fa', prefix = 'chr', add = T, ignoreCh
```

```
## reading fasta (this might take a while)..
```

```
## Extracting adjacent bases..
```

```
## matrix of dimension 187x96
```

```
#Run main function with maximum 6 signatures.
laml.sign = extractSignatures(mat = laml.tnm, nTry = 6)
```

```
## Warning : Found zero mutations for conversions A[T>G]C
```

```
## Estimating best rank..
```

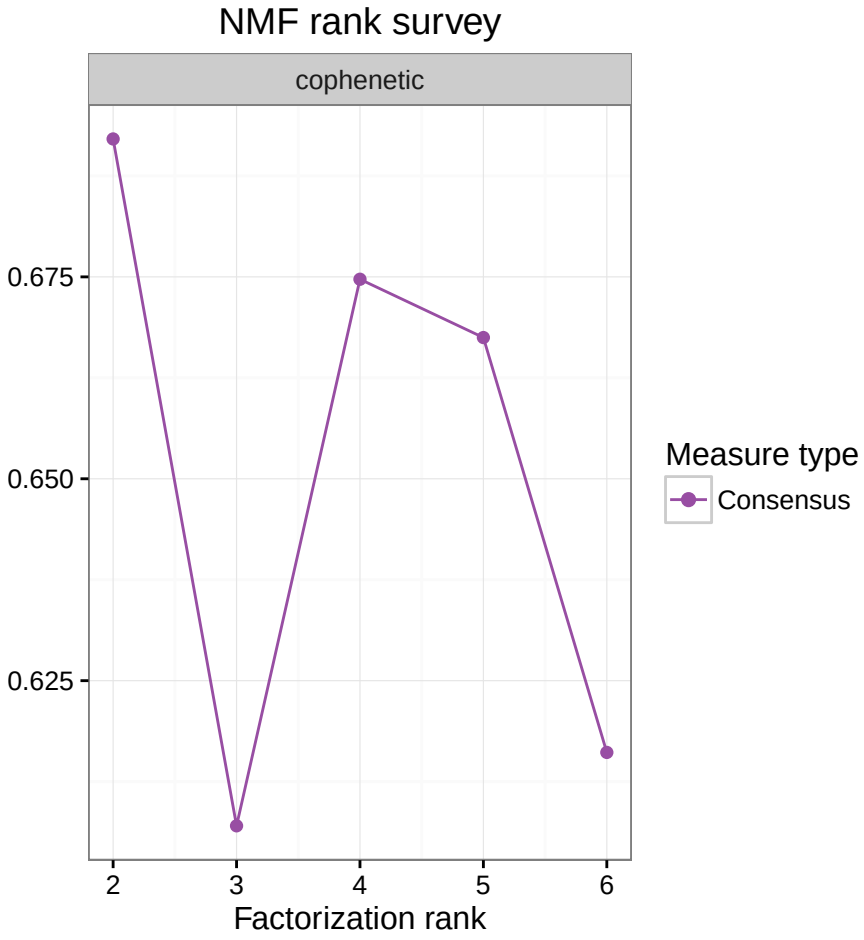
```
##   method   seed rng metric rank sparseness.basis sparseness.coef      rss
## 2 brunet random   1     KL    2        0.6067822       0.5758488 1729.158
## 3 brunet random   5     KL    3        0.6504446       0.6509778 1663.661
## 4 brunet random   3     KL    4        0.7220199       0.5917800 1589.476
## 5 brunet random   2     KL    5        0.7513203       0.6223095 1547.254
## 6 brunet random   4     KL    6        0.7566506       0.6293098 1494.899
##        evar silhouette.coef silhouette.basis residuals niter   cpu cpu.all
## 2 0.3741443       1.0000000        1.0000000  2844.238  2000 2.407  71.843
## 3 0.3978505       0.7600218        0.7678822  2669.199  1600 2.234  88.823
## 4 0.4247011       0.5787328        0.7764507  2503.733  2000 3.031  82.138
## 5 0.4399832       0.5423037        0.8159478  2373.979  2000 3.354 121.824
## 6 0.4589326       0.5002800        0.7840189  2264.855  2000 3.649 131.797
##   nrun cophenetic dispersion silhouette.consensus
## 2   10  0.6920776  0.1851160            0.3511645
## 3   10  0.6070270  0.2648117            0.1947465
## 4   10  0.6747046  0.4101130            0.2016363
## 5   10  0.6674872  0.4891841            0.1763987
## 6   10  0.6161154  0.5455175            0.1549031
```

```
## Using 2 as a best-fit rank based on maximum cophenetic correlation coefficient.
```

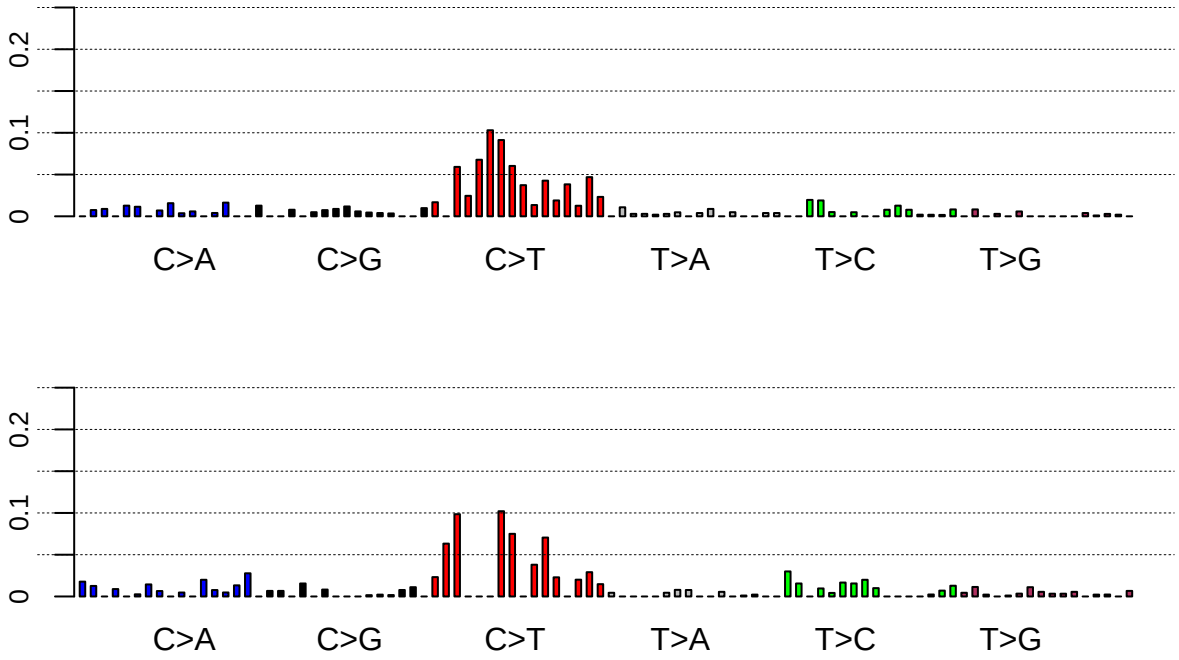
```
## Comparing against experimentally validated 21 signatures.. (See Alexandrov et.al Nature 2013 for details.)
```

```
## Found Signature_1 most similar to validated Signature_1B. Correlation coeff: 0.680267220446278
```

```
## Found Signature_2 most similar to validated Signature_1A. Correlation coeff: 0.789994248568557
```




```
plotSignatures(laml.sign)
```



`extractSignatures` gives a warning that no mutations are found for class A[T>G]C conversions. This is possible when the number of samples are low or in tumors with low mutation rate, such as in this case of Leukemia. In this scenario, a small positive value is added to avoid computational difficulties. It also prints other statistics for range of values that was tried, and chooses the rank with highest cophenetic metric (for above example r=2). The above stats should give an estimate of range best possible r values and in case the chosen r is overestimating, it is also possible to be re-run `extractSignatures` by manually specifying r using argument `n`.

Once decomposed, signatures are compared against known and validated signatures from Sanger[8]. In the above exaple, 2 signatures are derived. One of the signatures is most similar to validated signature_1A with a high correlation coefficient. Signature_1A is a result of elevated rate of spontaneous deamination of 5-methyl-cytosine, which results in C>T transitions and which predominantly occurs at NpCpG trinucleotide which is a most common process in AML[8,9]. .

Annotating variants using Oncotator.

We can also annotate variants using oncotator API[10]. `oncotate` function quires oncotator web api to annotate given set of variants and converts them into MAF format. Input should be a five column file with chr, start, end, ref_allele, alt_allele. However, it can conatain other information such as sample names (Tumor_Sample_Barcode), read counts, vaf information and so on, but only first five columns will be used, rest of the columns will be attached at the end of the table.

```
var.file = system.file('extdata', 'variants.tsv', package = 'maftools')
#This is what input looks like
var = read.delim(var.file, sep = '\t')
head(var)
```

```
##   chromsome    start      end     ref alt Tumor_Sample_Barcode
## 1      chr4 55589774 55589774       A   G               fake_1
## 2      chr4 55599321 55599321       A   T               fake_2
## 3      chr4 55599332 55599332       G   T               fake_3
## 4      chr4 55599320 55599320       G   T               fake_4
## 5     chr15 41961117 41961123 TGGCTAA   -               fake_4
## 6      chr4 55599320 55599320       G   T               fake_5
```

```
#Annotate
var.maf = oncotate(maflite = var.file, header = T)
```

```
#Results from oncotate. First 20 columns.
var.maf[1:10, 1:20, with =F]
```

NOTE: This is quite time consuming if input is big.

Adding read counts to maf file.

`addReadCounts` is wrapper script for bam-readcount programme, which takes MAF file as an input and adds read counts from the corresponding bam file[11]. `addReadCounts` assumes bam-readcount is installed and is under path.

Other useful functions.

maftools has few other functions such as `plotVaf` and `genesToBarcodes` which helps to plot vaf distributions and maps samples where a given genes are mutated respectively.

References

1. Cancer Genome Atlas Research, N., Genomic and epigenomic landscapes of adult de novo acute myeloid leukemia. N Engl J Med, 2013. 368(22): p. 2059-74.
2. Leiserson, M.D., et al., CoMEt: a statistical approach to identify combinations of mutually exclusive alterations in cancer. Genome Biol, 2015. 16: p. 160.
3. Tamborero, D., A. Gonzalez-Perez, and N. Lopez-Bigas, OncodriveCLUST: exploiting the positional clustering of somatic mutations to identify cancer genes. Bioinformatics, 2013. 29(18): p. 2238-44.
4. Dees, N.D., et al., MuSiC: identifying mutational significance in cancer genomes. Genome Res, 2012. 22(8): p. 1589-98.
5. Miller, C.A., et al., SciClone: inferring clonal architecture and tracking the spatial and temporal patterns of tumor evolution. PLoS Comput Biol, 2014. 10(8): p. e1003665.
6. Mroz, E.A., et al., Intra-tumor genetic heterogeneity and mortality in head and neck cancer: analysis of data from the Cancer Genome Atlas. PLoS Med, 2015. 12(2): p. e1001786.
7. Gaujoux, R. and C. Seoighe, A flexible R package for nonnegative matrix factorization. BMC Bioinformatics, 2010. 11: p. 367.
8. Alexandrov, L.B., et al., Signatures of mutational processes in human cancer. Nature, 2013. 500(7463): p. 415-21.
9. Welch, J.S., et al., The origin and evolution of mutations in acute myeloid leukemia. Cell, 2012. 150(2): p. 264-78.
10. Ramos, A.H., et al., Oncotator: cancer variant annotation tool. Hum Mutat, 2015. 36(4): p. E2423-9.
11. bam-readcount: https://github.com/genome/bam-readcount