Analysis with MutComFocal

Vladimir Trifonov

September 9, 2013

Load the library

```
> library(mutcomfocal)
> options(stringsAsFactors=FALSE)
```

Read the data files

```
> cnv<-read.delim("cnv_data", header=TRUE, sep="\t")
> mut<-read.delim("mut_data", header=TRUE, sep="\t")
```

They should look like this

```
> head(cnv)
```

```
        ID chrom loc.start   loc.end seg.mean
1 17_DLBCL     6     94649    505558    0.326
2 17_DLBCL     6  32689804  32974073    0.458
3 17_DLBCL     8     21242 146268947    0.324
4 17_DLBCL     9  37293477  37667675    0.444
5 17_DLBCL    10  80817120  91427131   -0.348
6 17_DLBCL    10  98154050  99737051   -0.253
```

```
> head(mut)
```

```
    sample gene.chr
1 33_DLBCL ABCA12@2
2 38_DLBCL ABCA12@2
3 45_DLBCL ABCA12@2
4 10_DLBCL ABCA13@7
5  1_DLBCL ABCA13@7
6 25_DLBCL ABCA13@7
```

Calculate the copy number change

```
> cnv$cnc<-2*2^cnv[,5]-2
```

Note that this assumes that there is no chr X and/or Y. If such are present the calculation should take care of this, but keep in mind that in this case it is probably not a good idea to mix male and female data.

Load the reference for the CNV data

```
> hg18<-ref("human/hg18")
```

At the moment there is only "human/hg18". The reference is of class `mcf_ref` and a descendant of `list`.

```
> class(hg18)
```

```
[1] "mcf_ref" "list"
```

It has two important members: `gene` and `chr`. `gene` contains information about the genes included in the reference

```
> head(hg18$gene)
```

```
   gene chr    begin      end
1  A1BG  19 63549983 63556677
2  A1CF  10 52236330 52315441
3 A2BP1  16  6009132  7703341
4 A2LD1  13 99981810 99983998
5   A2M  12  9111570  9159825
6 A2ML1  12  8866416  8920644
```

and `chr` contains information about the chromosome of the reference

```
> head(hg18$chr)
```

```
        end num_genes idx name centr_begin centr_end
1 247179968      2224   1    1   121236957 123476957
2 242751142      1329   2    2    91689898  94689898
3 199392125      1138   3    3    90587544  93487544
4 191247457       829   4    4    49354874  52354874
5 180727832       958   5    5    46441398  49441398
6 170735673      1114   6    6    58938125  61938125
```

Convert the CNV/mut data into lesions data

```
> data<-lesions(hg18, cnv, mut)
```

```
1
10
11
12
13
14
15
16
17
18
19
2
20
21
22
3
4
5
6
7
8
9
```

The lesions data is of class `mcf_lesions` and a descendant of `data.frame`

```
> class(data)
```

```
[1] "mcf_lesions" "data.frame"
```

```
> head(data)
```

```
  type         d   sample  gene id
1    1 0.9451777 10_DLBCL 14059  1
2    1 0.7307291 27_DLBCL 14059 53
3    1 0.5897020 28_DLBCL 14059 63
```

```
4    1 0.9451777 10_DLBCL 10288  1
5    1 0.7307291 27_DLBCL 10288 53
6    1 0.5897020 28_DLBCL 10288 63
```

`type` is 0 for amplifications, 1 for deletions and 2 for mutations. A particular lesion, e.g. with `id`=17, looks like this

```
> data.frame(subset(data, id==17), hg18$gene[subset(data, id==17)$gene, ])
```

```
     type        d  sample  gene id gene.1 chr     begin       end
9325    0 1.661665 3_DLBCL 17672 17    SHE   1 152718582 152741213
9343    0 1.661665 3_DLBCL 19813 17 TDRD10   1 152741318 152787247
9361    0 1.661665 3_DLBCL 21096 17 UBE2Q1   1 152787674 152797744
9379    0 1.661665 3_DLBCL  3856 17 CHRNB2   1 152806880 152818977
9397    0 1.661665 3_DLBCL   295 17   ADAR   1 152821157 152867061
9415    0 1.661665 3_DLBCL  9355 17  KCNN3   1 152946536 153109378
9433    0 1.661665 3_DLBCL 15219 17   PMVK   1 153163831 153176108
9451    0 1.661665 3_DLBCL 14526 17 PBXIP1   1 153183179 153195191
9469    0 1.661665 3_DLBCL 16065 17  PYGO2   1 153196125 153200882
9487    0 1.661665 3_DLBCL 17666 17   SHC1   1 153201397 153213583
9505    0 1.661665 3_DLBCL  3920 17  CKS1B   1 153213741 153218348
9523    0 1.661665 3_DLBCL  6786 17  FLAD1   1 153222440 153232211
9541    0 1.661665 3_DLBCL 10060 17  LENEP   1 153232685 153233415
```

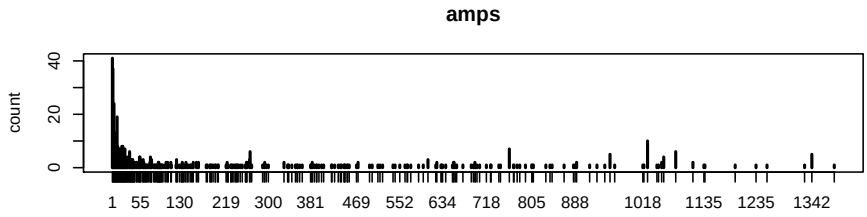
We can calculate the distribution of the numbers of genes per lesion like this

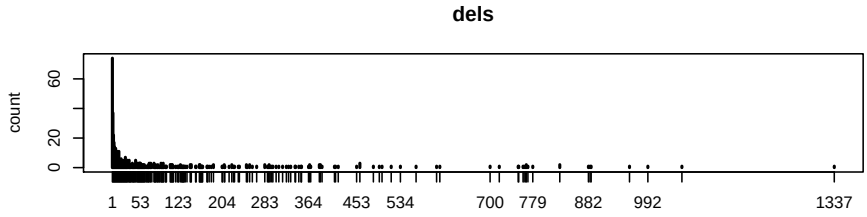
```
> x1<-data.frame(unique(data[,c("id", "sample", "type")]))
> x1<-x1[order(x1$id),]
> x1$num_genes<-as.matrix(table(data$id))
> head(x1)
```

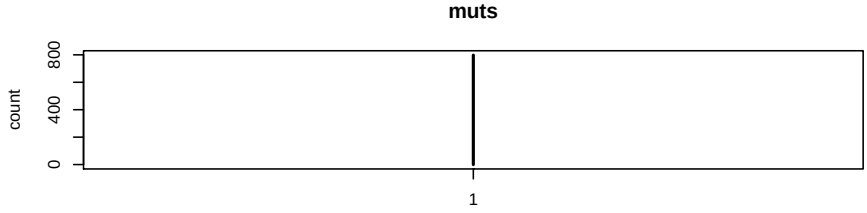
```
     id   sample type num_genes
1     1 10_DLBCL    1       113
59    2  5_DLBCL    1       217
5627  3 49_DLBCL    0       104
5628  4 18_DLBCL    0      1054
5725  5 36_DLBCL    0       514
5739  6 28_DLBCL    0      1045
```

```
> x2<-by(x1, x1$type, function (x) table(x$num_genes))
```

```
> par(mfrow=c(3, 1))
> plot(x2$"0", main="amps", ylab="count")
> plot(x2$"1", main="dels", ylab="count")
> plot(x2$"2", main="muts", ylab="count")
```







We are ready to compute the MutComFocal scores corresponding to the lesions data

```
> mcf<-score(data)
```

```
1
```

`mcf` is of class `mcf_score` and a descendant of `data.frame`. It has the following columns

```
> colnames(mcf)
```

```
 [1] "idx"        "amp_recurr" "amp_focal"  "amp_mut"    "amp_comb"
 [6] "del_recurr" "del_focal"  "del_mut"    "del_comb"   "mut"
[11] "all"
```

`idx` is the index of the gene and can be used in combination with the reference to get more information about the gene.

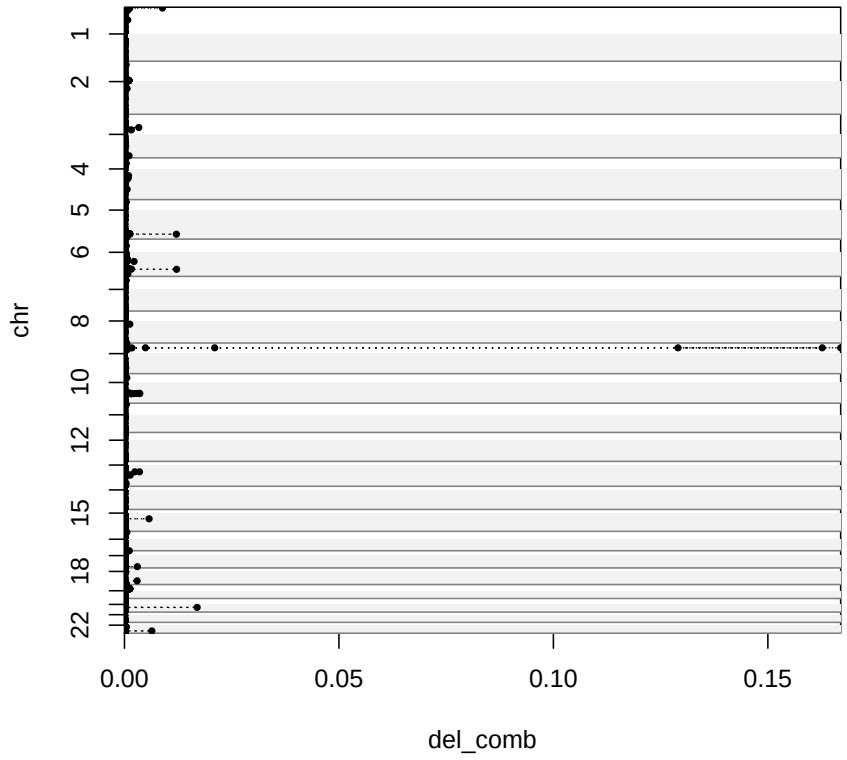
```
> head(data.frame(hg18$gene[mcf$idx,], mcf[,c("amp_comb", "del_comb")]))
```

```
   gene chr    begin      end     amp_comb     del_comb
1  A1BG  19 63549983 63556677 2.084621e-09 0.000000e+00
2  A1CF  10 52236330 52315441 9.887932e-07 2.683236e-08
3 A2BP1  16  6009132  7703341 4.810452e-07 1.209720e-06
4 A2LD1  13 99981810 99983998 4.293735e-06 1.193850e-05
5   A2M  12  9111570  9159825 4.268823e-06 3.943025e-08
6 A2ML1  12  8866416  8920644 4.268823e-06 3.943025e-08
```

One can use the `plot` function to explore visually the MutComFocal scores.

```
> mcf_black<-as.score(data.frame(mcf[,c("idx", "del_comb")], color="black"),
+                     hg18)
```

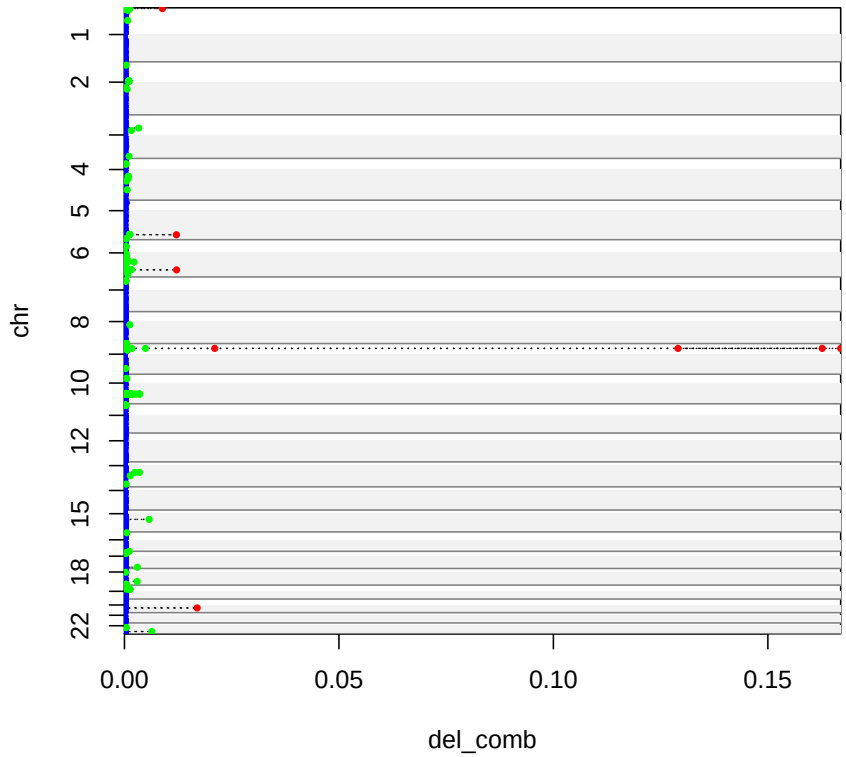
```
> plot(mcf_black)
```



Note that here we used `as.score` to convert a data frame into an object of class `mcf_score`. We do this because the `plot` function needs an object of that class to produce the correct plots. Besides the data frame to be converted, `as.score` takes an additional argument, `hg18` in the example above, which is the reference to be associated with the new `mcf_score` object.

The `plot` function can be customized a lot. For example, if we want to color the genes by their tier we can use the following code to color the genes with three colors: one for the 1st tier, one for the 2nd, and one for the rest

```
> mcf_color<-color_by_tier(mcf[,c("idx", "del_comb")], 2)
> head(mcf_color)
  idx     del_comb     color
1   1 0.000000e+00 #0000FFFF
2   2 2.683236e-08 #0000FFFF
3   3 1.209720e-06 #0000FFFF
4   4 1.193850e-05 #0000FFFF
5   5 3.943025e-08 #0000FFFF
6   6 3.943025e-08 #0000FFFF
> plot(mcf_color)
```



The first argument to `plot` is an `mcf_score` object with three columns: the index of the gene, the score to be plotted, and the color for the gene.

We can also mark some of the genes on the plot. For example if we want to mark the leaders of the top 25 regions of the `del_comb` score, first we use the regions function

```
> regs<-regions(mcf[,c("idx", "del_comb")], 25)
```

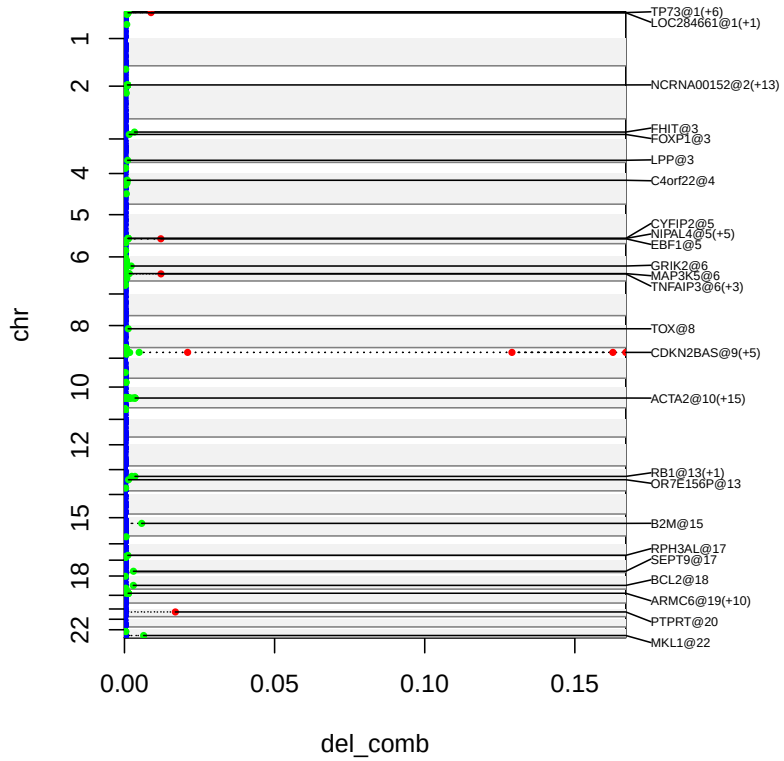
```
> head(regs)
```

```
    idx         leader       score chr     begin       end
1  3611 CDKN2BAS@9(+5) 0.166975775   9  21792634  22442472
2 16008       PTPRT@20 0.016946710  20  40134805  41251971
3 20389  TNFAIP3@6(+3) 0.012150709   6 137506649 138246142
4  5585         EBF1@5 0.012108909   5 158055500 158459366
5 20522     TP73@1(+6) 0.008848885   1   2975603   3640327
6 12408        MKL1@22 0.006382094  22  39136237  39362636
                                          genes
1     CDKN2BAS,CDKN2A,CDKN2B,C9orf53,MTAP,DMRTA1
2                                          PTPRT
3                  TNFAIP3,IL22RA2,IFNGR1,OLIG3
4                                          EBF1
5 TP73,PRDM16,ARHGEF16,MEGF6,MIR551A,TPRG1L,WDR8
6                                          MKL1
```

and then

```
> regs_color<-data.frame(regs[,c("idx", "leader")], color="black")
```

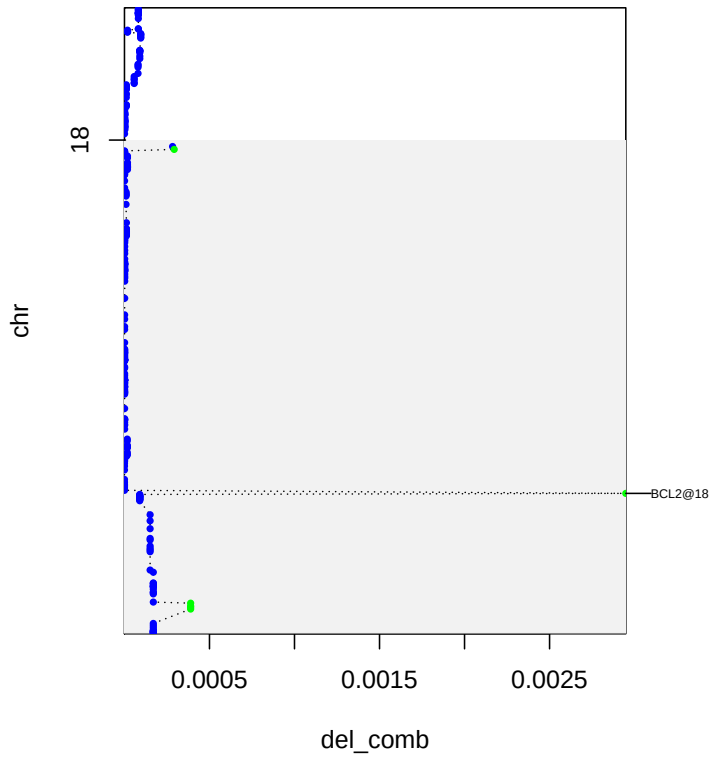
```
> plot(mcf_color, regs_color)
```



The second, optional, argument to `plot` is a data frame with three columns: the index of the gene to be labeled, the label to be used, and the color for the label.

We can focus only on some chromosomes

```
> chr18<-subset(mcf_color, hg18$gene$chr[idx]=="18")
> plot(chr18, regs_color)
```

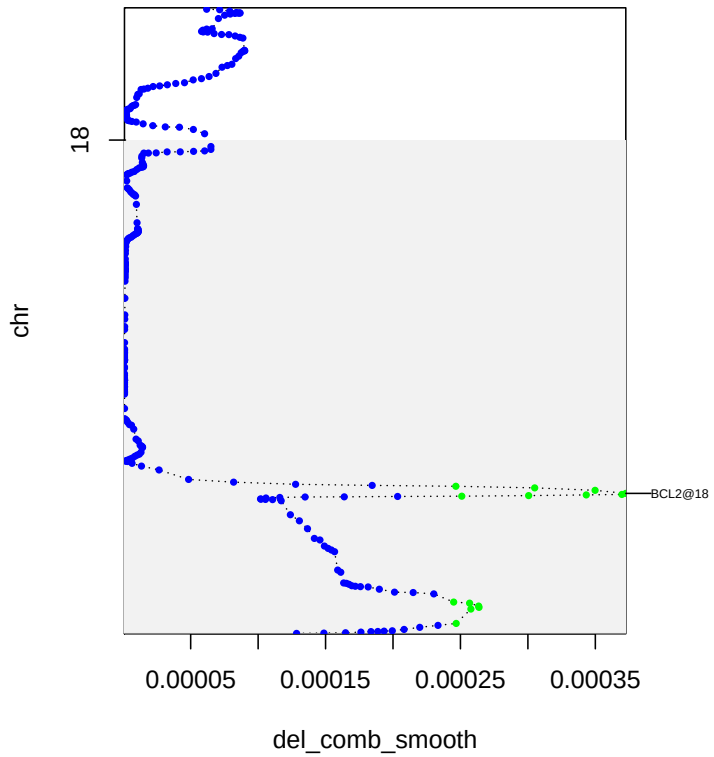


The function `smooth` can be used to smoothen a particular score. To smoothen the `del_comb` score with a window of 5 genes around each gene we use

```
> mcf_smooth<-smooth(mcf[,c("idx", "del_comb")], 5)
> head(mcf_smooth)
  idx     del_comb del_comb_smooth
1   1 0.000000e+00    0.000000e+00
2   2 2.683236e-08    2.617715e-08
3   3 1.209720e-06    1.028200e-06
4   4 1.193850e-05    3.663999e-05
5   5 3.943025e-08    3.943025e-08
6   6 3.943025e-08    3.943025e-08
```

To see what chr18 looks like after the smoothening we do

```
> mcf_smooth_color<-color_by_tier(mcf_smooth[, c("idx", "del_comb_smooth")], 2)
> chr18_smooth_color<-subset(mcf_smooth_color, hg18$gene$chr[idx]=="18")
> plot(chr18_smooth_color, regs_color)
```

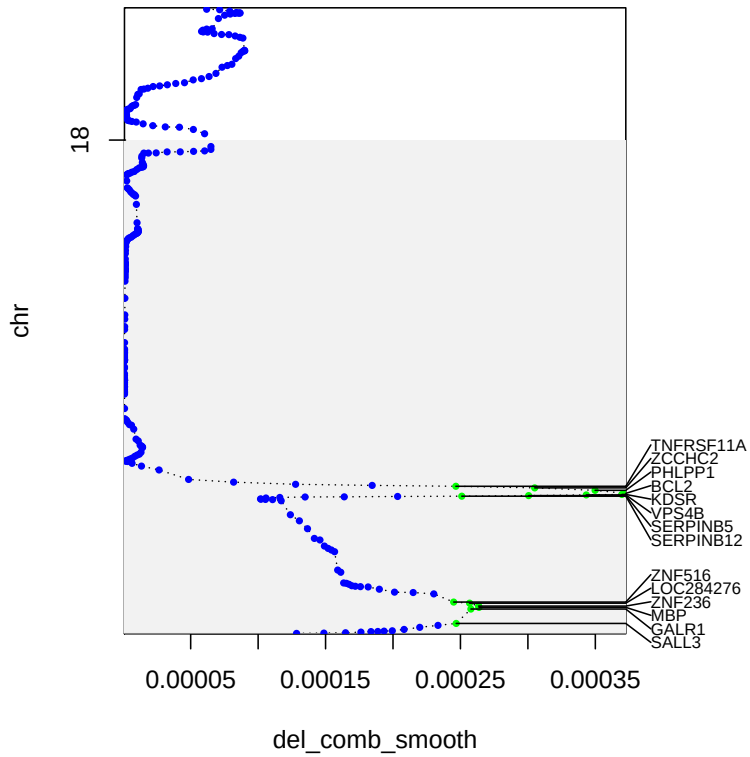


We can see what are the genes around BCL2 like this

```
> mcf_smooth_tier<-trfr(mcf_smooth[, c("idx", "del_comb_smooth")])
> mcf_smooth_tier<-mcf_smooth_tier[,c("idx", "del_comb_smooth_tier")]

> chr18_idx<-subset(mcf_smooth_tier,
+                   hg18$gene$chr[idx]=="18" & abs(del_comb_smooth_tier)<=2)$idx
> chr18_labels<-data.frame(idx=chr18_idx,
+                          gene=hg18$gene$gene[chr18_idx],
+                          color="black")

> plot(chr18_smooth_color, chr18_labels, text_cex=0.7, text_dist=1.3/60)
```



Here we used the function `trfr` which computes the tier, rank, fdr, and regions at a particular gene for a given score.

```
> mcf_perm<-score(data, perm=1)
```

```
1
1
```

```
> mcf_del_comb_trfr<-trfr(mcf[,c("idx", "del_comb")], mcf_perm)
```

```
> head(mcf_del_comb_trfr)
```

```
  idx     del_comb del_comb_tier del_comb_rank del_comb_fdr del_comb_regions
1   1 0.000000e+00           -17         20824           1               22
2   2 2.683236e-08           -11         13232           1               82
3   3 1.209720e-06            -7          7640           1              160
4   4 1.193850e-05            -5          3171           1              148
5   5 3.943025e-08           -11         13027           1               84
6   6 3.943025e-08           -11         13027           1               84
```

`tier` is computed by an interative procedure using the entropy of the score. The sign of the `tier` of a gene is negative if there is a close by peak with higher tier. `regions` for a gene denotes the number of contiguous sequences of genes with score higher than that gene. The `fdr` is computed with respected to the scores of a permuted dataset of lesions (this is done by the call to `score` above).

A labeling of the peaks is produced by the `peaks` function

```
> pks<-peaks(mcf[,c("idx", "del_comb")])
> head(pks)
```

```
    idx     del_comb tier chr    begin       end          name
56   56 8.228392e-05    3  16 46674384  47201621 ABCC12@16(+4)
202 202 3.550557e-03    2  10 90684812  90741127      ACTA2@10
239 239 2.152004e-05    4   2 54195913  54385939       ACYP2@2
263 263 1.505884e-05    5   8 39291338  39993067  ADAM5P@8(+5)
373 373 1.044163e-04    3   2 99530147 100125469       AFF3@2
437 437 3.237439e-04    2   7 17304800  19003517     AHR@7(+3)
                                    genes
56       ABCC12,ABCC11,LONP2,SIAH1,N4BP1
202                                 ACTA2
239                                 ACYP2
263 ADAM5P,ADAM3A,ADAM18,ADAM2,IDO1,IDO2
373                                  AFF3
437              AHR,SNX13,PRPS1L1,HDAC9
```

This can be used in a plot in the following way

```
> pks_labels<-data.frame(subset(pks, tier<=3)[,c("idx", "name")],
+                        color="black")
```

```
> plot(mcf_color[hg18$gene$chr[mcf_color$idx] %in% c("10", "18"),],
+      pks_labels)
```