facopy: Feature-based association and gene-set enrichment for copy number alteration analysis in cancer

David Mosen-Ansorena[1]

October 17, 2016

[1]Genome Analysis Platform
CIC bioGUNE and CIBERehd
dmosen.gn@cicbiogune.es

Contents

1 Introduction

This document guides the reader through the use of the **facopy** package. The main aim of the package is to provide fine-tuned copy number alteration (CNA) association modeling. Association is measured directly at the genomic features of interest and, in the case of genes, downstream gene-set enrichment analysis can be performed thanks to novel internal processing of the data. The software opens a way to systematically scrutinize the differences in CNA distribution across tumoral phenotypes, such as those that relate to tumor type, location and progression. Currently, the output format from 12 different methods that analyze data from whole-genome/exome sequencing and SNP microarrays, is supported. Overall, the package was conceived to be user-friendly, provide visual assessment at each step and keep a simple and limited parameterization.

1.1 Example data

The package bundles example data from 20 colorectal cancer samples. The DNA samples were hybridized to SNP microarrays. Then the data was GC-corrected, preprocessed with TumorBoost (**?**) and analyzed with Genome Alteration Print (GAP) (**?**). See **facopy**'s paper for more details. For the sake of space and speed, only copy number calls longer than one megabase are included. Although the example data does not include the sexual chromosomes, the package is able to process them.

2 Input

An example analysis on the previously described data follows. First, if you have not already, install the **facopy** package through the `biocLite()` function, and then load it. Use the devel version of Bioconductor to get the package seamlessly.

```
> source("http://www.bioconductor.org/biocLite.R")
> biocLite("facopy")

> library(facopy)
```

The initial step of a **facopy** analysis is to load the data from your input files and from **facopy**'s companion annotation package:

Copy number calls Copy number calls are the output files generated by CNA detection tools that analyze data from HTS (whole genome, exome sequencing) or SNP-microarrays (aCGH, SNP-based). As of now, the output format from the following methods is supported: seqCNA (**?**), CNAnorm (**?**), FREEC (**?**), OncoSNP-SEQ (**?**), Patchwork (**?**), TITAN (**?**), EXCAVATOR (**?**), ExomeCNV (**?**), GAP (**?**), OncoSNP (**?**) and FastCall (**?**).

Phenotypes Phenotypic information takes the form of one variable per phenotype. A variable can be continuous or categorical (discrete), and does not need to be defined for all the samples.

Genomic features **facopy** is able to analyze data from Homo sapiens genome builds hg18 and hg19, and Mus musculus build mm8. For this latter, genes and microRNAs retrieved from BioMart's Ensembl database (**?**) can be used as features at which to perform the association. The following sets of features taken from CaSNP (**?**) can be used on human data as well: large intergenic non-coding RNAs (lincRNAs), tumor suppressor genes and oncogenes. Note nevertheless that any other set of genomic features may be imported to the analysis from an external source.

Call `getFacopyInfo()` to get a list of possible combinations for supported input formats, variable types, genomic features and more:

```
> getFacopyInfo()

Variable types:
 -  categorical (aliases: categorical, discrete, qualitative, enumerative)
 -  quantitative (aliases: quantitative, continuous)
Alteration combinations:
 -  amplifications
 -  deletions
 -  loh
 -  cnas
 -  any (aliases: any, all)
```

```
 -  onlygain
 -  someloss
Input data:
 -  seqcna
 -  cnanorm
 -  patchwork
 -  freec
 -  oncosnp
 -  oncosnp-seq
 -  gap
 -  exomecnv
 -  titan
 -  fastcall
 -  excavator
Genome builds:
 -  hg18
 -  hg19
 -  mm8
Genomic features:
 -  hg18 feature (bundled: cancergene, ensembl, lincRNA, mirnas, oncogene, tumorsupressor)
 -  hg19 feature (bundled: cancergene, ensembl, lincRNA, mirnas, oncogene, tumorsupressor)
 -  mm8 feature (bundled: ensembl, mirnas)
External data sets:
 -  hg18 db (available: gsk_bladder, gsk_blood, gsk_bone, gsk_brain, gsk_breast, gsk_cervix
 -  hg19 db (available: gsk_bladder, gsk_blood, gsk_bone, gsk_brain, gsk_breast, gsk_cervix
 -  mm8 db
```

Starting with the copy number calls, the `readCNData()` function takes as parameters, at least, the following two: the folder with the copy number data and the name of the tool used to generate it. Check `?readCNData` to learn more about the formats that **facopy** understands as input. Here are some examples:

```
> # myCalls = readCNData("~/myFolder/", "seqcna")
> # myCalls = readCNData("~/myFolder/", "gap", pfbFilename="~/myPfb.pfb")
> # myCalls = readCNData("~/myFolder/", "cnanorm", window=50000)
```

For this vignette, the output of `readCNData()` over the example data has already been left in the `myCalls` object. Hence, we will just load it into memory through the `data()` function, like so:

```
> data(myCalls)
```

The next step is to attach the phenotypic information to the `myCalls` object. Either if you have it in a file (with headers) or in a data frame you previously created, just call the `addVariables()` function. You also need to specify, at least, the types of the variables (continuous or categorical). In our case, the `age` and `stage` variables are continuous and categorical, respectively. We will leave the combined data in the `myStudy` object.

```
> data(myVariables)
> class(myVariables)

[1] "data.frame"

> head(myVariables)

   code age stage
1   101  66     3
2   103  83     4
6   119  71     2
10  131  82     4
13  137  53     4
15  141  66     3

> myStudy = addVariables(myCalls, myVariables, c("continuous","categorical"))
```

To complete our input, we will select the genomic features of our interest and the genome build, which should coincide with the one used during the CNA detection. Typically, you will want to select the whole set of genes (`ensembl`), but there are interesting studies highlighting the relevance of miRNAs (`mirna`) and lincRNAs (`lincRNA`) in cancer (**?**). Let me stress that, in order to perform the subsequent gene-set enrichment analysis, it is necessary to indicate some kind of gene collection (`ensembl`, `oncogene`, `tumorsuppressor`, `cancergene`).

```
> myStudy = addFeatures(myStudy, "oncogene", "hg18")
> summary(myStudy)

    Length      Class       Mode
         1 facopyInfo         S4
```

By default, for each genomic feature, only overlapping copy number calls are considered. However, additional upstream and downstream flanks can be set. This might render useful, for instance, in order to include regions upstream of every gene, thus accounting for nearby regulatory elements.

3 The alterations

Two factors are used in **facopy** to discriminate copy number alterations: (1) whether the copy number is lower, equal or greater than 2 (diploid genomes are assumed) and the presence of loss of heterozygosity (LOH). The following relevant combinations of alterations are defined in **facopy**:

`amplifications` All amplifications (CN>2).

`deletions` All deletions (CN<2).

`loh` All loss of heterozygosity (LOH), regardless of copy number.

`cnas` All copy number alterations (CN<>2).

`any` Any kind of alteration.

`all` An alias for any kind of alteration, same as `any`.

`onlygain` Only non-LOH amplifications.

`someloss` All deletions plus LOH alterations.

You might also use capital letters anywhere in the names and abbreviations (e.g. `Amp` for `amplifications`). Where relevant, these modified names will show up in the graphical output.

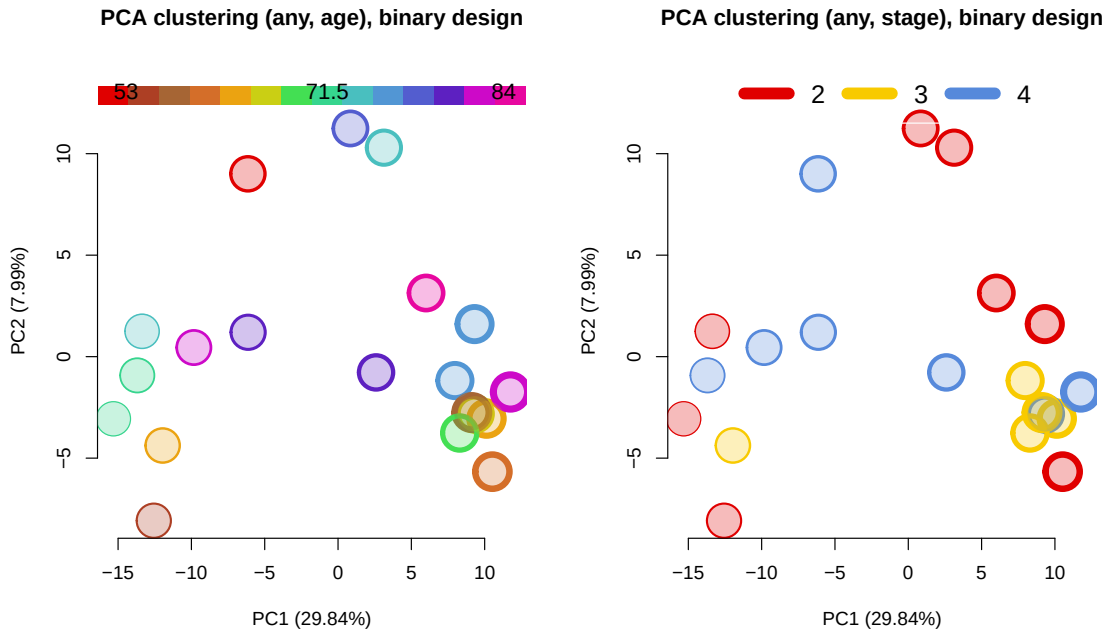
Depending on the combination of alterations, different designs are available in some of **facopy**'s functions. The simplest design is `binary`: an alteration exists or it does not. The `versus` design, for CNAs, assigns a value of -1, 0 or 1 depending on whether a deletion, no copy number change or an amplification exists for a given feature, respectively. The `vlog` design, for all (any) alterations, assigns a value of -1, 0 or 1 depending on whether a deletion or LOH, no copy number change or an amplification without LOH exists.

4 Analysis

One way to start looking at the data is to plot the results of a Principal Component Analysis (PCA), showing the similarity among the samples. By default, `plotPCA()` uses a `binary` design. In the following call, we tell it to use all alterations and to color the points (representing the samples) based on their phenotypes. By default, the thickness of the point rims represents the rank, among all samples, in the amount of total altered base pairs, where a thinner rim indicates fewer altered based pairs. Visual assessment discards correlation between amount of total alterations and the phenotypes.

```
> par(mfrow=c(1,2))
> pca1 = plotPCA(myStudy, "any", "age")
> pca2 = plotPCA(myStudy, "any", "stage")
> head(pca2$eig, 2)
```

```
       eigenvalue percentage of variance cumulative percentage of variance
comp 1   93.40288              29.841176                           29.84118
comp 2   24.99629               7.986036                           37.82721
```



4.1 Recurrence assessment

facopy offers both graphical and text output to help assessing alteration recurrence.

4.1.1 Summary tables

Text output consists of three summaries that provide an overview of the variables in the study and the alterations in the samples. The following functions generate tables that focus on summarizing or testing a specific aspect of the data, and can write the output to a selected file:

`variableSummary()` Shows per-alteration type differences among the values in each variable.

`alterationSummary()` Breaks down alteration frequencies by chromosome arm and two alteration classifications: deletion, normal copy number or amplification, and presenting LOH or not.

`variableCor()` Performs appropriate statistical tests that measure correlations between pairs of variables.

```
> variableSummary(myStudy)
```

```
      Variable Alteration      Value A Mean altered A Value B Mean altered B
 [1,] "age"    "Deletion"      NA      NA             NA      NA
 [2,] "age"    "Amplification" NA      NA             NA      NA
 [3,] "age"    "LOH"           NA      NA             NA      NA
```

```
 [4,] "stage"  "Deletion"      "2"     "43951088"     "3"     "72554076"
 [5,] "stage"  "Amplification" "2"     "887697727"    "3"     "1096143560"
 [6,] "stage"  "LOH"           "2"     "237126546"    "3"     "411355155"
 [7,] "stage"  "Deletion"      "2"     "43951088"     "4"     "76345421"
 [8,] "stage"  "Amplification" "2"     "887697727"    "4"     "829109065"
 [9,] "stage"  "LOH"           "2"     "237126546"    "4"     "329992090"
[10,] "stage"  "Deletion"      "3"     "72554076"     "4"     "76345421"
[11,] "stage"  "Amplification" "3"     "1096143560"   "4"     "829109065"
[12,] "stage"  "LOH"           "3"     "411355155"    "4"     "329992090"
      p value  tau
 [1,] "0.844"  "-0.033"
 [2,] "0.649"  "0.0749"
 [3,] "0.435"  "-0.128"
 [4,] "0.606"  NA
 [5,] "0.943"  NA
 [6,] "0.0295" NA
 [7,] "0.683"  NA
 [8,] "0.867"  NA
 [9,] "0.397"  NA
[10,] "0.87"   NA
[11,] "0.343"  NA
[12,] "0.048"  NA
```

```
> head(alterationSummary(myStudy))
```

```
    Arm    Alteration Altered length Arm length Alteration frequency
49   1p      Deletion        4328846  124300000              0.03480
50   1p  NormalPloidy       52796822  124300000              0.42500
51   1p Amplification       27230411  124300000              0.21900
331  1p         NoLOH       63706784  124300000              0.51300
341  1p           LOH       20649296  124300000              0.16600
52   1q      Deletion         756234  122949719              0.00615
```

```
> variableCor(myStudy)
```

```
     Variable A Variable B Fisher Chisquare Welch_oneway Kruskal-Wallis
[1,] "age"      "stage"    NA     NA        "0.352"      "0.287"
     Kendall_p Kendall_tau
[1,] NA        NA
```

You can actually call these capitulating functions before you attach the genomic feature annotation. Also, alternatively, you might call the `variableSummary()`, which collects the three tables and can output them to a folder you indicate:

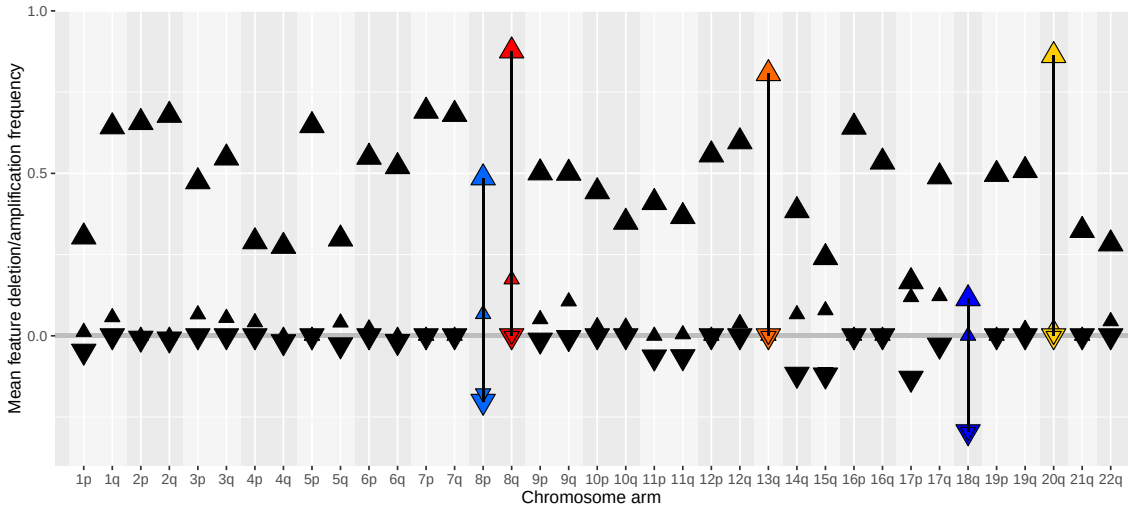
```
> myCallsPreview = preview(myStudy)
> sapply(myCallsPreview, head, 1)
```

```
$byVar
     Variable Alteration Value A Mean altered A Value B Mean altered B p value
[1,] "age"    "Deletion" NA      NA             NA      NA             "0.844"
     tau
[1,] "-0.033"

$byAlt
   Arm Alteration Altered length Arm length Alteration frequency
49  1p   Deletion        4328846  124300000               0.0348

$varCor
     Variable A Variable B Fisher Chisquare Welch_oneway Kruskal-Wallis
[1,] "age"      "stage"    NA     NA        "0.352"      "0.287"
     Kendall_p Kendall_tau
[1,] NA        NA
```

4.1.2 Visualization

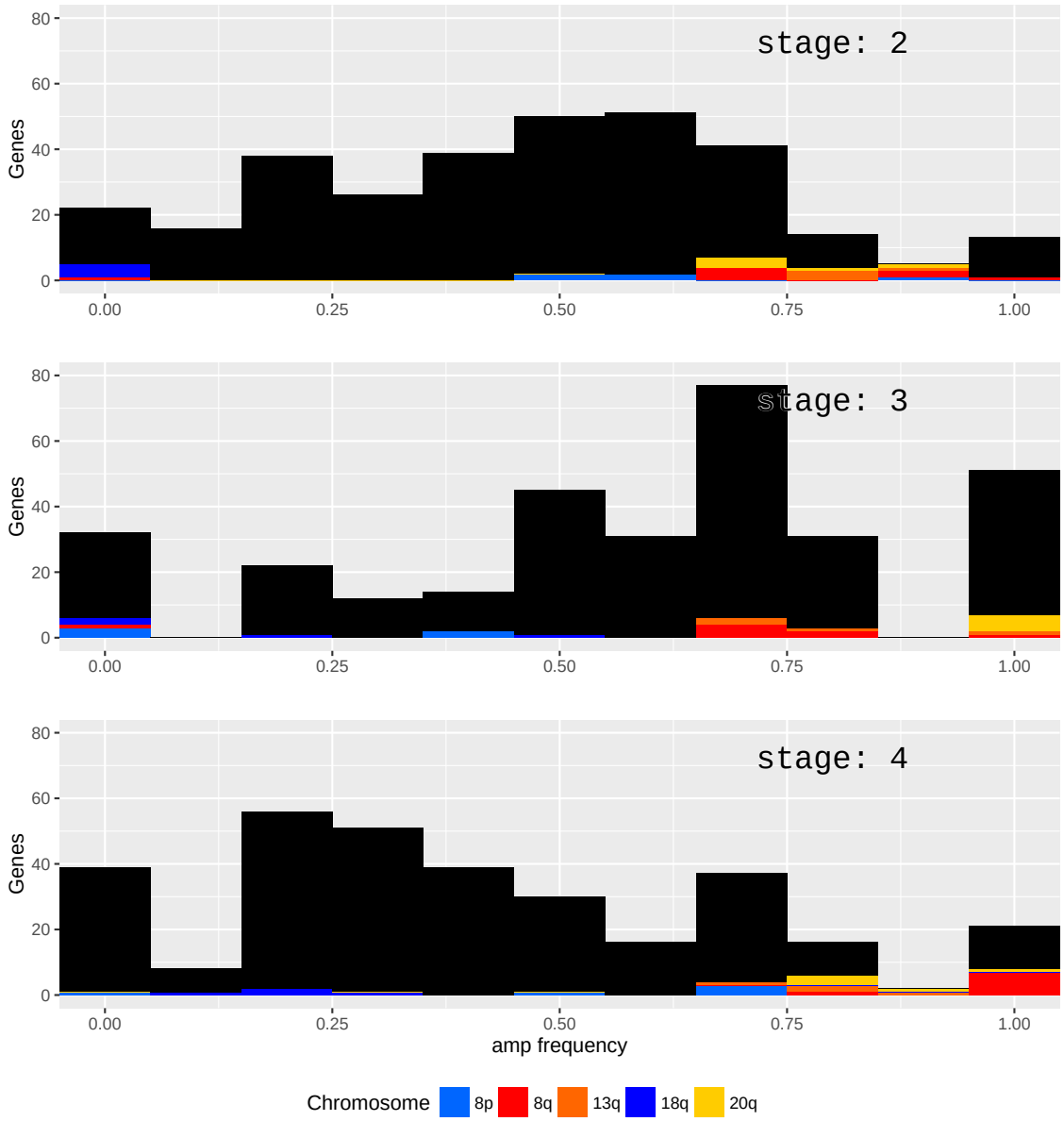
The plot generated by `plotBar()` provides an overview of chromosome arm-wise frequencies of CNA and LOH combinations. It displays, for each chromosome arm, four triangle-shaped points. The two points above the abscissa indicate amplification frequencies and the two points below it, deletion frequencies. The two big points specify the overall amplification and deletion frequencies in the arm, while the small points specify somatic LOH frequencies within either copy number alteration type. Frequencies can be calculated as the median incidence of either altered features or base pairs in a given arm across all samples in the dataset. Chromosome arms of interest can be highlighted.

```
> myArms = c("8q","13q","20q","8p","18q")
> myColors = c(rainbow(15)[1:3], rainbow(15)[10:11])
> plotBar(myStudy, TRUE, myArms, myColors, ylim=c(-0.4,1))
```



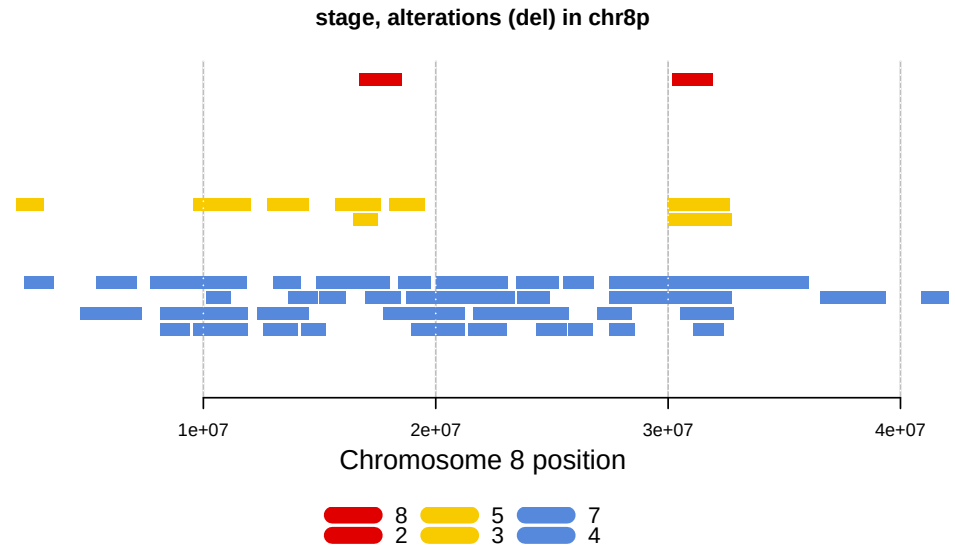
Data of such complexity, with multiple variables, genes, alteration types and chromosome arms can be broken down in many ways in order to investigate different aspects. `plotHist()` outputs individual stacked histograms with alteration frequencies for each of the values in a categorical variable. In this example, we choose to display amplifications broken down by tumor stage. Each bar in the histograms represents the number of features (here, genes) with an altered frequency that falls within a specific range. Chromosome arms of interest can be differentially colored here as well. Notice that the bin size (width of each value range) and the maximum value in the Y axis are required.

```
> plotHist(myStudy, "amp", "stage", myArms, myColors, bin=0.1, ymax=80)
```



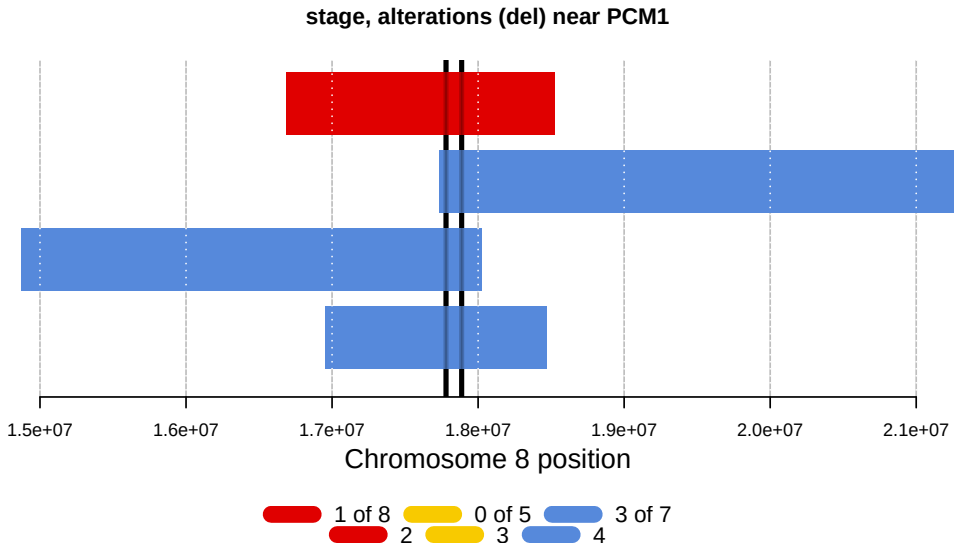
Another possibility, which deepens more into the data, is to look at a specific arm within each sample, and color a set of alterations based on the phenotypic value. Given the histograms, there seemed to be no relevant differences in main amplifications based on tumor stage. Therefore, now, let us focus on chromosome arm 8p, which looked pretty deleted overall in the plot bar, and the deletions within it. Each sample's deletions are all depicted within the same line:

```
> plotZoom(myStudy, "arm", "8p", "del", "stage")
```



Arm 8p is quite more deleted in stage IV (4) samples. Although we do not yet have association results for the genes in this arm, we can, for the sake of providing an example, zoom into gene *PCM1*, which we will see to be associated to tumor stage.

```
> plotZoom(myStudy, "feat", "PCM1", "del", "stage")
```



4.2 Association analysis

facopy implements a flexible system for the definition of association models between genomic copy number and phenotypes. The association is assessed at every genomic feature by considering for each test only those alterations that overlap with the corresponding feature.

4.2.1 Association models

In the simplest scenario, a single phenotype is measured against the presence of a certain combination of alterations overlapping each feature. The meaning of the phenotype will prompt to simply assess the independence from the copy number or to model the phenotype as either a cause (predictor) or consequence (response) of the copy number differences. Apart from just measuring the presence or absence of a combination of alterations, other designs are possible (see above).
Discrete variables can be turned into ordinal variables, while values from continuous variables can be classified into intervals and thus be considered ordinal as well.
More complex models are possible by defining variable interactions and strata, see `?formula`. In addition, null models can also be specified so that, for example, the effect of considering an additional variable in a model can be statistically measured. Finally, advanced users of facopy can fully specify their association models in order to define their own statistical tests and interactions between phenotypes and copy number.
For a detailed description of the parameters in the `facopy()` function that allow the aforementioned characteristics, see `?facopy`.

4.2.2 Examples

In these examples, we will see that `facopy()` outputs both text and graphical results.
Example 1 shows how deletions in gene *PCM1* are indeed significantly associated with tumor stage in our samples. The call to the function is pretty straightforward, so let us take a look at a slightly more advanced use case.
By default, the function establishes that the phenotype is a consequence of the copy number differences. In example 2, we indicate that our model establishes our variables to predict the copy number. Specifically, we measure the impact of the interaction between our two variables on the copy number.
If you want to visualize the results of the association, set the `plot` parameter to `TRUE` (example 3). In such case, you might overlay data from an external database. Namely, the frequencies of the relevant alterations in the desired data set will be displayed as translucent black bars. `getFacopyInfo()` lists, among other things, the collection of available data sets, which were gathered from the Cancer Genome WorkBench (**?**).

```
> genes = facopy(myStudy, "del", "stage") #1
> head(genes[order(genes$p_value),])
```

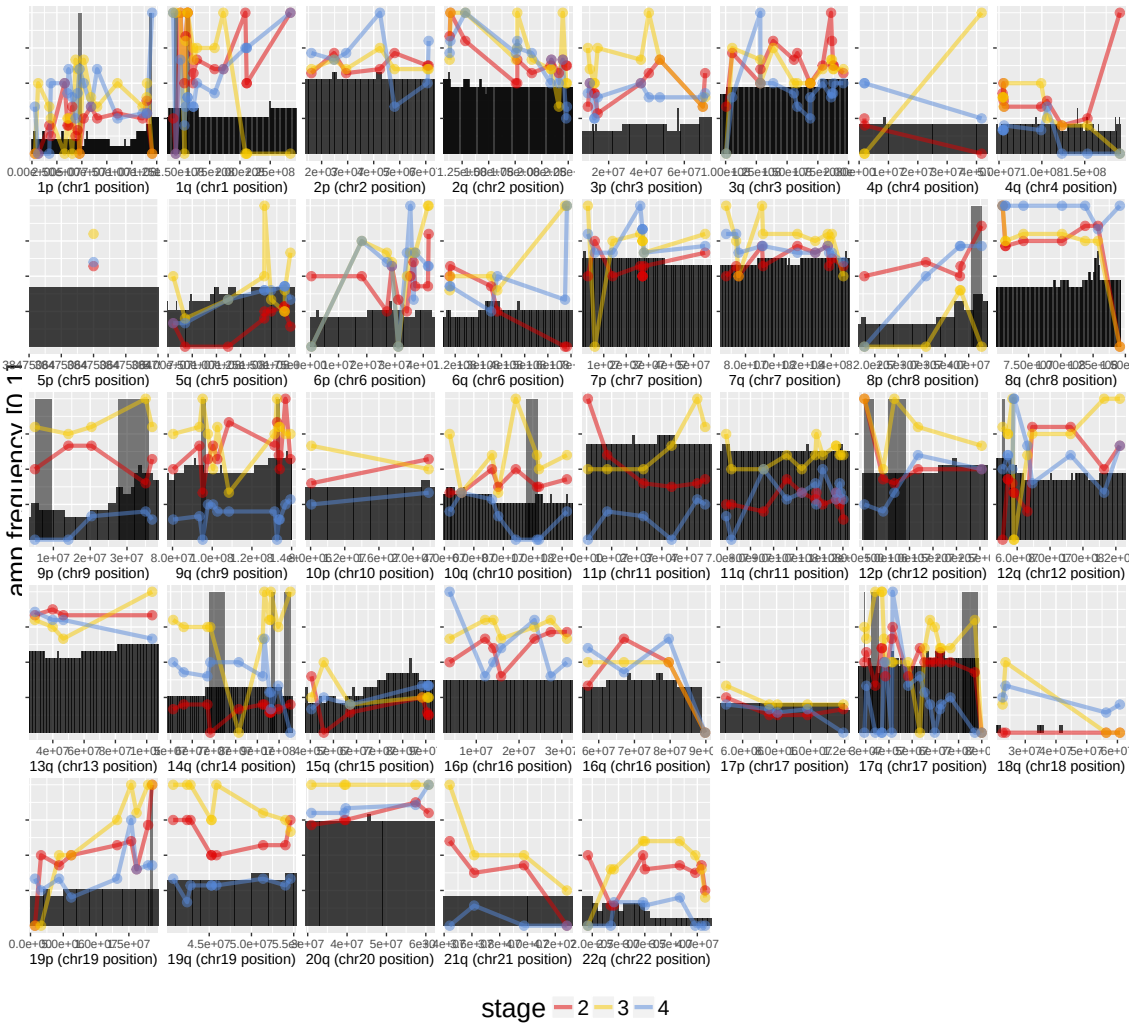
```
    feature p_value chr_q_arm    bp_st    bp_en
279    BCL2  0.0089       18q 60790578 60986613
142    PCM1  0.0370        8p 17780365 17887457
226    TSHR  0.0563       14q 81421868 81612646
```

```
253     TP53  0.0762       17p  7571719  7590863
255     GAS7  0.0762       17p  9813925 10101868
254     PER1  0.0989       17p  8043787  8055753
```

```
> genes = facopy(myStudy, "amp", "stage*age", modelPart="predictor") #2
> head(genes[order(genes$p_value),])
```

```
    feature p_value chr_q_arm     bp_st     bp_en
175     RET  0.0038       10q  43572516  43625797
166    TAL2  0.0143        9q 108424777 108425104
290   CD79A  0.0147       19q  42381189  42385439
54     PAX3  0.0148        2q 223064606 223163700
173   GATA3  0.0189       10p    8096666   8117164
167     SET  0.0206        9q 131445933 131458675
```

```
> genes = facopy(myStudy, "amp", "stage", plot=TRUE, db="gsk_colon") #3
```



4.3 Gene-set enrichment analysis

Gene-set enrichment is a statistical analysis in which biologically relevant sets of related genes (e.g. pathways) are examined for enrichment in genes found during the association. **facopy** performs such analysis over a range of gene-set collections from MSigDB (**?**) that include Gene Ontology (GO), Reactome, KEGG, miRNA and transcription factor targets, and more. Notice that gene-set enrichment analysis is only available if genes or a subset of genes, such as those that are cancer-related, were selected as features.

Before the analysis is run, genes with little influence on gene expression are disregaded, as they are not expected to present relevant mutations. This is achieved by calculating the correlation coefficient (R2) between copy number and expression. You might use your own expression data (tab-delimited with headers) or tell **facopy** to calculate the correlation in a dataset from the cBio project (**?**):

```
> # eCor = calculateCor(myStudy, "~/myExprData.txt")
> # eCor = calculateCor(myStudy, "mrna_merged_median_Zscores", "coadread_tcga_pub")
```

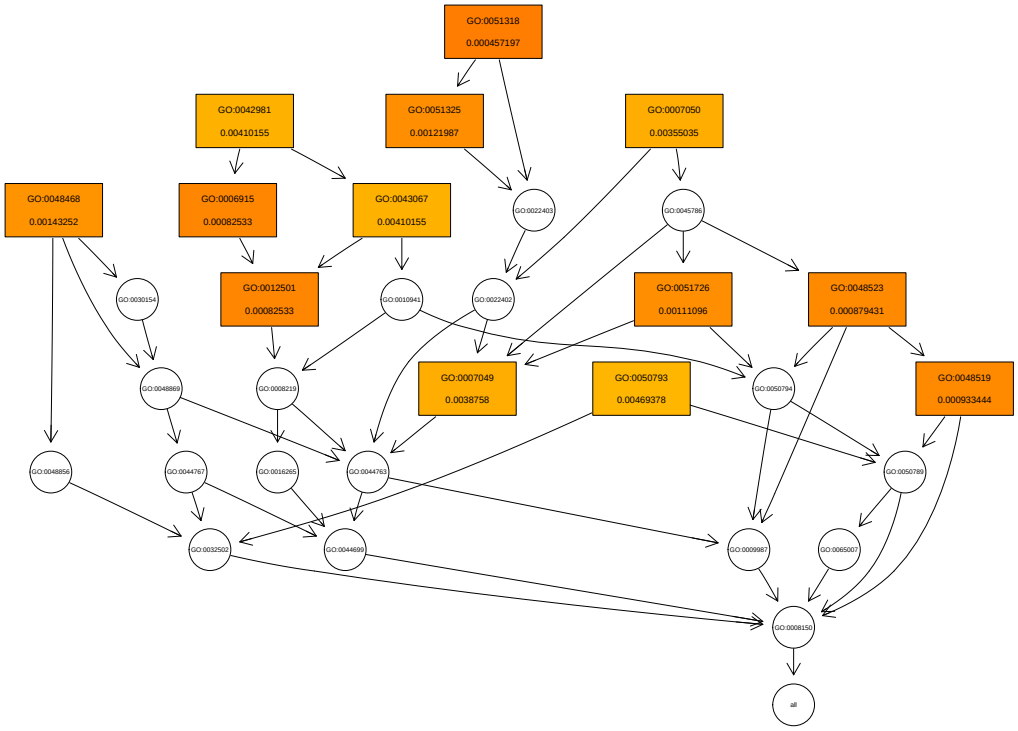
The `facopyEnrichment()` function uses the correlation coefficients to keep only those genes with values above a given threshold.

```
> # facopyEnrichment(myStudy, genes, eCor, "~/myFolder/stageAmpEnrichment")
```

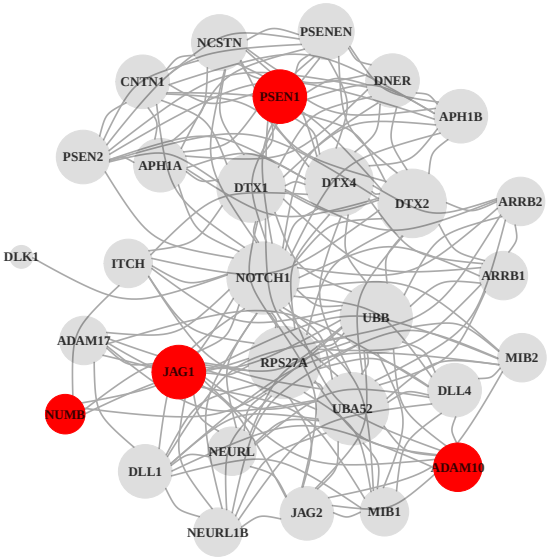
Another issue with gene-set enrichment analysis is that it assumes independent association results for each gene. However, this is not the case in cancer copy number analysis. Within a significant cluster, most likely, many of the gene alterations are merely what are known as passengers, in contrast to driver mutations, which do actually have an influence on the phenotype.
GRAS (Score column in the enrichment tables, see below) is a two-valued score that involves the examination of gene distribution across chromosome arms, in an attempt to assess the possibility that enrichment derives from the presence of many significant genes in few arm-wide alterations. One should not discard, however, that significantly lower counts are due to co-location of related genes, especially for small numbers of genes. GRAS provides a value that indicates the amount of chromosome arms to which the genes in the enrichment belong and another one for the significance of the enrichment.

Category	Size	ExpCount	Count	OddsR	Pvalue	PvalueAdj	Score	Genes
Class A/1	6	0.197	2	14.91	0.0147	1	2_0.807	...
Apoptosis	65	2.13	6	3.082	0.0190	1	5_0.044	...
Zinc transp.	7	0.230	2	11.92	0.0202	1	2_0.718	...

facopy also produces graphical output for the three gene ontologies (Biological Process, Molecular Function and Cellular Component) and for the canonical pathway collections (Reactome, KEGG and BioCarta). In the case of the former, graph trees with enriched terms are produced, whereas, for each enriched pathway, a graph with gene relationships is generated. Only those gene sets with an enrichment significance below the corresponding threshold in the call `facopyEnrichment()` will be included in the grahical output.
All the output from the gene-set enrichment analysis is saved to the indicated folder in the call to `facopyEnrichment()`.



Activated NOTCH1 Transmits Signal to the Nucleus
p−value: 0.000711663



```
> sessionInfo()

R version 3.3.1 (2016-06-21)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: OS X 10.9.5 (Mavericks)

locale:
[1] C/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

attached base packages:
[1] grid      stats     graphics  grDevices utils     datasets  methods
[8] base

other attached packages:
[1] facopy_1.8.0         facopy.annot_0.107.0 gridExtra_2.2.1
[4] ggplot2_2.1.0        coin_1.1-2           survival_2.39-5
[7] cgdsr_1.2.5

loaded via a namespace (and not attached):
 [1] Biobase_2.34.0          splines_3.3.1           assertthat_0.1
 [4] DO.db_2.9               stats4_3.3.1            RBGL_1.50.0
 [7] Category_2.40.0         RSQLite_1.0.0           lattice_0.20-34
[10] digest_0.6.10           chron_2.3-47            qvalue_2.6.0
[13] colorspace_1.2-7        sandwich_2.3-4          Matrix_1.2-7.1
[16] R.oo_1.20.0             plyr_1.8.4              GSEABase_1.36.0
[19] FactoMineR_1.33         XML_3.98-1.4            genefilter_1.56.0
[22] xtable_1.8-2            GO.db_3.4.0             mvtnorm_1.0-5
[25] scales_0.4.0            BiocParallel_1.8.0      tibble_1.2
[28] annotate_1.52.0         IRanges_2.8.0           TH.data_1.0-7
[31] nnet_7.3-12             BiocGenerics_0.20.0     magrittr_1.5
[34] DOSE_3.0.0              R.methodsS3_1.7.1       MASS_7.3-45
[37] GOstats_2.40.0          graph_1.52.0            tools_3.3.1
[40] data.table_1.9.6        multcomp_1.4-6          stringr_1.1.0
[43] S4Vectors_0.12.0        munsell_0.4.3           cluster_2.0.5
[46] AnnotationDbi_1.36.0    flashClust_1.01-2       RCurl_1.95-4.8
[49] rappdirs_0.3.1          AnnotationForge_1.16.0  leaps_2.9
[52] igraph_1.0.1            bitops_1.0-6            labeling_0.3
[55] gtable_0.2.0            codetools_0.2-15        DBI_0.5-1
[58] reshape2_1.4.1          R6_2.2.0                zoo_1.7-13
[61] knitr_1.14              dplyr_0.5.0             graphite_1.20.0
[64] fastmatch_1.0-4         fgsea_1.0.0             Rgraphviz_2.18.0
[67] modeltools_0.2-21       GOSemSim_2.0.0          stringi_1.1.2
[70] parallel_3.3.1          Rcpp_0.12.7             scatterplot3d_0.3-37
```