Using the TRONCO package

Marco Antoniotti[*] Giulio Caravagna[*] Luca De Sano[*] Alex Graudenzi[*]
Ilya Korsunsky[†] Mattia Longoni[*] Loes Olde Loohuis[‡] Giancarlo Mauri[*]
Bud Mishra[†] Daniele Ramazzotti[*]

August 4, 2015

Abstract. Genotype-level *cancer progression models* describe the temporal ordering in which genomic alterations such as somatic mutations and copy number alterations tend to fixate and accumulate during cancer formation and progression. These graphical models can describe trends of *natural selection* across a population of independent tumour samples (cross-sectional data), or reconstruct the clonal evolution in a single patient's tumour (multi-region or single-cell data). In terms of application, such models can be used to better elucidate genotype-phenotype relation, predict cancer hallmarks and outcome of personalised treatment as well as suggest novel targets for therapy design.

TRONCO (TR*anslational* ONCO*logy*) is a R package which collects algorithms to infer progression models from Bernoulli 0/1 profiles of genomic alterations across a tumor sample.
Such profiles are usually visualized as a binary input matrix where each row represents a patientâĂŹs sample (e.g., the result of a sequenced tumor biopsy), and each column an event relevant to the progression (a certain type of somatic mutation, a focal or higher-level chromosomal copy number alteration, etc.); a 0/1 value models the absence/presence of that alteration in the sample.
In this version of TRONCO such profiles can be readily imported by boolean matrices and MAF or GISTIC files. The package provides various functions to editing, visualize and subset such data, as well as functions to query the cBioPortal for cancer genomics.
In the current version, TRONCO provides parallel implementations of CAPRESE [PLoS ONE 9(12): e115570] and CAPRI [Bioinformatics, doi:10.1093/bioinformatics/btv296] algorithms to infer progression models arranged as trees or general direct acyclic graphs. Bootstrap procedures to assess the non-prametric and statistical confidence of the inferred models are also provided. The package comes with example data available, which include the dataset of Atypical Chronic Myeloid Leukemia samples by Piazza et al., Nat. Genet., 45 (2013).

Requirements: You need to have installed the R package `rgraphviz`, see `Bioconductor.org`.

[*]Dipartimento di Informatica Sistemistica e Comunicazione, Universitá degli Studi Milano Bicocca Milano, Italy.
[†]Courant Institute of Mathematical Sciences, New York University, New York, USA.
[‡]Center for Neurobehavioral Genetics, University of California, Los Angeles, USA.

Events selection

We will start by loading TRONCO in the R console along with an example *"dataset"* that comes with the package.

```
> library(TRONCO)
> data(aCML)
> hide.progress.bar <<- TRUE
```

We then use the function `show` to get a short summary of the aCML dataset.

```
> show(aCML)
```

```
Description: CAPRI - Bionformatics aCML data.
Dataset: n=64, m=31, |G|=23.
Events (types): Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point.
Colors (plot): darkgoldenrod1, forestgreen, cornflowerblue, coral.
Events (10 shown):
         gene 4 : Ins/Del TET2
         gene 5 : Ins/Del EZH2
         gene 6 : Ins/Del CBL
         gene 7 : Ins/Del ASXL1
         gene 29 : Missense point SETBP1
         gene 30 : Missense point NRAS
         gene 31 : Missense point KRAS
         gene 32 : Missense point TET2
         gene 33 : Missense point EZH2
         gene 34 : Missense point CBL
Genotypes (10 shown):
          gene 4 gene 5 gene 6 gene 7 gene 29 gene 30 gene 31 gene 32 gene 33 gene 34
patient 1      0      0      0      0       1       0       0       0       0       0
patient 2      0      0      0      0       1       0       0       0       0       1
patient 3      0      0      0      0       1       1       0       0       0       0
patient 4      0      0      0      0       1       0       0       0       0       1
patient 5      0      0      0      0       1       0       0       0       0       0
patient 6      0      0      0      0       1       0       0       0       0       0
```

With the function `as.events`, we check the events in the dataset.

```
> as.events(aCML)
```

```
         type             event
gene 4   "Ins/Del"        "TET2"
gene 5   "Ins/Del"        "EZH2"
gene 6   "Ins/Del"        "CBL"
gene 7   "Ins/Del"        "ASXL1"
gene 29  "Missense point"   "SETBP1"
gene 30  "Missense point"   "NRAS"
gene 31  "Missense point"   "KRAS"
gene 32  "Missense point"   "TET2"
gene 33  "Missense point"   "EZH2"
gene 34  "Missense point"   "CBL"
gene 36  "Missense point"   "IDH2"
gene 39  "Missense point"   "SUZ12"
gene 40  "Missense point"   "SF3B1"
gene 44  "Missense point"   "JARID2"
gene 47  "Missense point"   "EED"
gene 48  "Missense point"   "DNMT3A"
gene 49  "Missense point"   "CEBPA"
gene 50  "Missense point"   "EPHB3"
gene 51  "Missense point"   "ETNK1"
gene 52  "Missense point"   "GATA2"
gene 53  "Missense point"   "IRAK4"
gene 54  "Missense point"   "MTA2"
gene 55  "Missense point"   "CSF3R"
gene 56  "Missense point"   "KIT"
gene 66  "Nonsense Ins/Del" "WT1"
gene 69  "Nonsense Ins/Del" "RUNX1"
gene 77  "Nonsense Ins/Del" "CEBPA"
gene 88  "Nonsense point"   "TET2"
gene 89  "Nonsense point"   "EZH2"
gene 91  "Nonsense point"   "ASXL1"
gene 111 "Nonsense point"   "CSF3R"
```

Which account for alterations in the following genes.

```
> as.genes(aCML)

 [1] "TET2"   "EZH2"   "CBL"    "ASXL1"  "SETBP1" "NRAS"   "KRAS"   "IDH2"   "SUZ12"
[10] "SF3B1"  "JARID2" "EED"    "DNMT3A" "CEBPA"  "EPHB3"  "ETNK1"  "GATA2"  "IRAK4"
[19] "MTA2"   "CSF3R"  "KIT"    "WT1"    "RUNX1"
```

Now we take a look at the alterations of the gene `SETBP1` **across the samples.**

```
> as.gene(aCML, genes='SETBP1')

           Missense point SETBP1
patient 1                      1
patient 2                      1
patient 3                      1
patient 4                      1
patient 5                      1
patient 6                      1
patient 7                      1
patient 8                      1
patient 9                      1
patient 10                     1
patient 11                     1
patient 12                     1
patient 13                     1
patient 14                     1
patient 15                     0
patient 16                     0
patient 17                     0
patient 18                     0
patient 19                     0
patient 20                     0
patient 21                     0
patient 22                     0
patient 23                     0
patient 24                     0
patient 25                     0
patient 26                     0
patient 27                     0
patient 28                     0
patient 29                     0
patient 30                     0
patient 31                     0
patient 32                     0
patient 33                     0
patient 34                     0
patient 35                     0
patient 36                     0
patient 37                     0
patient 38                     0
patient 39                     0
patient 40                     0
patient 41                     0
patient 42                     0
patient 43                     0
patient 44                     0
patient 45                     0
patient 46                     0
patient 47                     0
patient 48                     0
patient 49                     0
patient 50                     0
patient 51                     0
patient 52                     0
patient 53                     0
patient 54                     0
patient 55                     0
patient 56                     0
patient 57                     0
patient 58                     0
patient 59                     0
patient 60                     0
patient 61                     0
patient 62                     0
patient 63                     0
patient 64                     0
```

We consider a subset of all the genes in the dataset to be involved in patters based on the support we found in the literature. See the main CAPRI paper for the reference.

```
> gene.hypotheses = c('KRAS', 'NRAS', 'IDH1', 'IDH2', 'TET2', 'SF3B1', 'ASXL1')
```

Regardless from which types of mutations we include, we select only the genes which appear alterated at least in the 5% of the patients. Thus, we first transform the dataset into "Alteration" (i.e., by collapsing all the event types for the same gene), and then we use only the select events from the original dataset.

```
> alterations = events.selection(as.alterations(aCML), filter.freq = .05)
```

```
*** Aggregating events of type(s) {Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point}
in a unique event with label "Alteration".
Dropping event types Ins/Del, Missense point, Nonsense Ins/Del, Nonsense point for 23 genes.
*** Binding events for 2 datasets.
*** Events selection: #events=23, #types=1 Filters freq|in|out = {TRUE, FALSE, FALSE}
Minimum event frequency: 0.05 (3 alterations out of 64 samples).
Selected 7 events.

Selected 7 events, returning.
```

We display a plot of the selected genes. Note that this plot has no title as name annotation is not copied by the function `events.selection`**.**

```
> dummy = oncoprint(alterations)
```

```
*** Oncoprint for ""
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Setting automatic row font (exponential scaling): 13
```

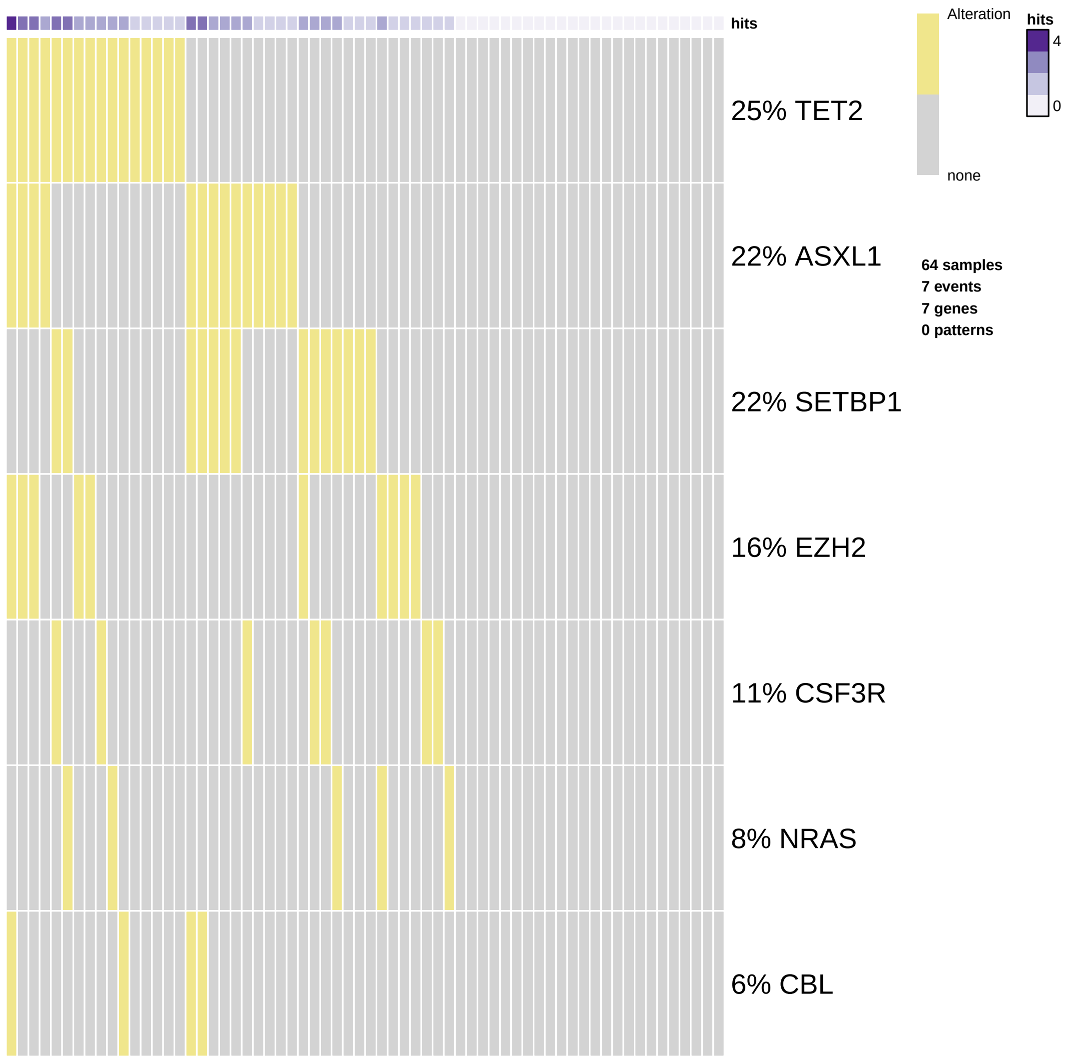

Figure 1: **Oncoprint output**

Adding Hypotheses

We now create the `dataset` to be used for the inference of the progression model. We consider the original dataset and from it we select all the genes whose mutations are occurring at least the 5% of the times together with any gene involved in an hypothesis. To do so, we use the function `filter.in.names` as shown below.

```
> hypo = events.selection(aCML, filter.in.names=c(as.genes(alterations), gene.hypotheses))
```

```
*** Events selection: #events=31, #types=4 Filters freq|in|out = {FALSE, TRUE, FALSE}
[filter.in] Genes hold: TET2, EZH2, CBL, ASXL1, SETBP1 ...  [10/14 found].
Selected 17 events, returning.
```

```
> hypo = annotate.description(hypo, 'CAPRI - Bionformatics aCML data (selected events)')
```

We display a new oncoprint with this dataset where we annotate the genes in `gene.hypotheses` **in order to identify them. The sample names are also shown.**

```
> dummy = oncoprint(hypo,  gene.annot = list(priors= gene.hypotheses), sample.id = T)
```

```
*** Oncoprint for "CAPRI - Bionformatics aCML data (selected events)"
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Annotating genes with RColorBrewer color palette Set1 .
Setting automatic row font (exponential scaling): 10.7
Setting automatic samples font half of row font: 5.3
```

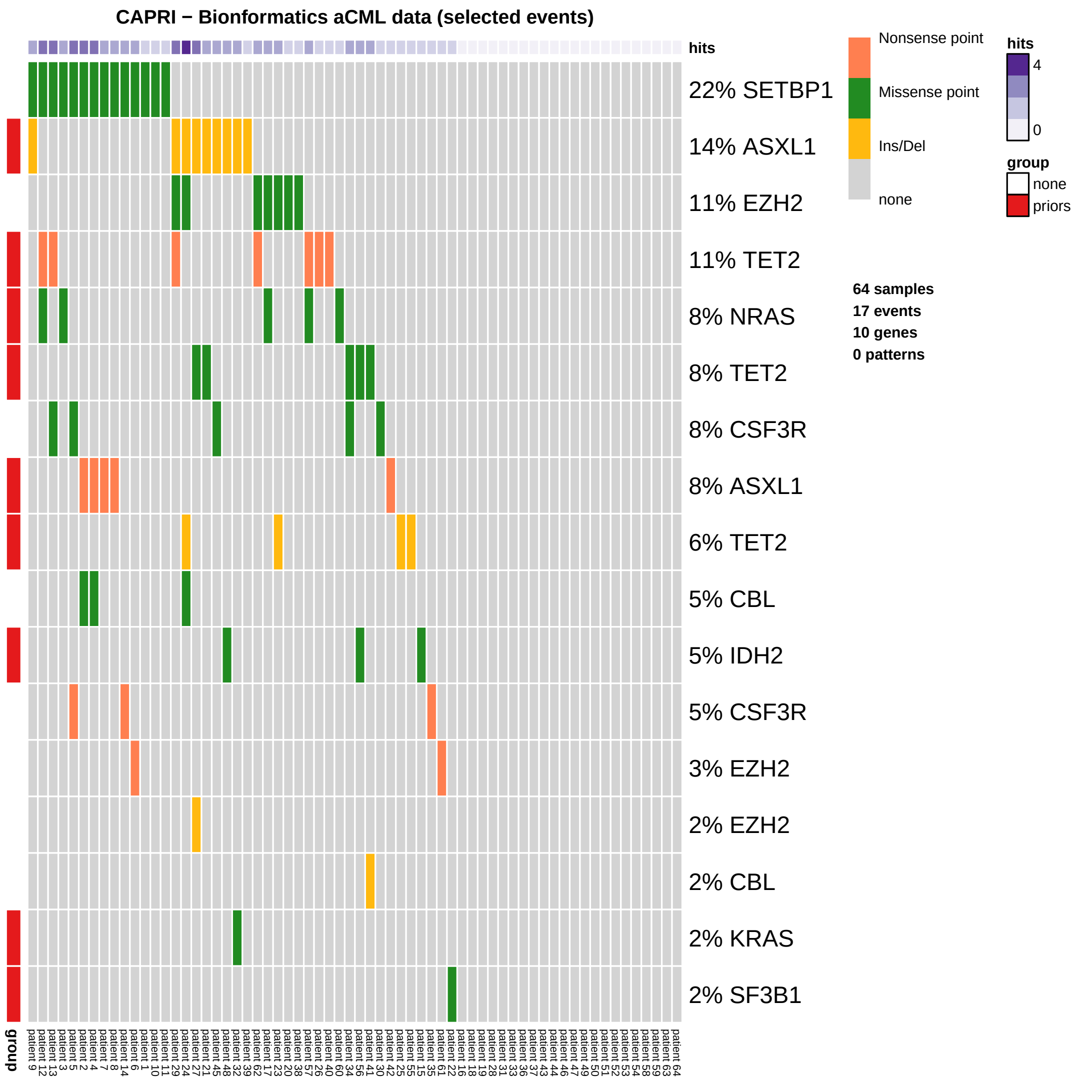

Figure 2: **Oncoprint output**

We now also add the hypotheses that are described in CAPRI's manuscript. Hypothesis of hard exclusivity (XOR) for NRAS/KRAS events (Mutation). This hypothesis is tested against all the events in the dataset.

```
> hypo = hypothesis.add(hypo, 'NRAS xor KRAS', XOR('NRAS', 'KRAS'))
```

Here we try to include also a soft exclusivity (OR) pattern but, since its *"signature"* is the same of the hard one, it will not be included. The code below is expected to rise an error.

```
> hypo = hypothesis.add(hypo, 'NRAS or KRAS',  OR('NRAS', 'KRAS'))
```

For the sake of better highlighting the perfect (hard) exclusivity among NRAS/KRAS mutations, one can have a further look at their alterations.

```
> dummy = oncoprint(events.selection(hypo, filter.in.names = c('KRAS', 'NRAS')))
```

```
*** Events selection: #events=18, #types=4 Filters freq|in|out = {FALSE, TRUE, FALSE}
[filter.in] Genes hold: KRAS, NRAS ...  [2/2 found].
Selected 2 events, returning.
*** Oncoprint for ""
with attributes: stage=FALSE, hits=TRUE
```

```
Sorting samples ordering to enhance exclusivity patterns.
Setting automatic row font (exponential scaling): 14.4
```

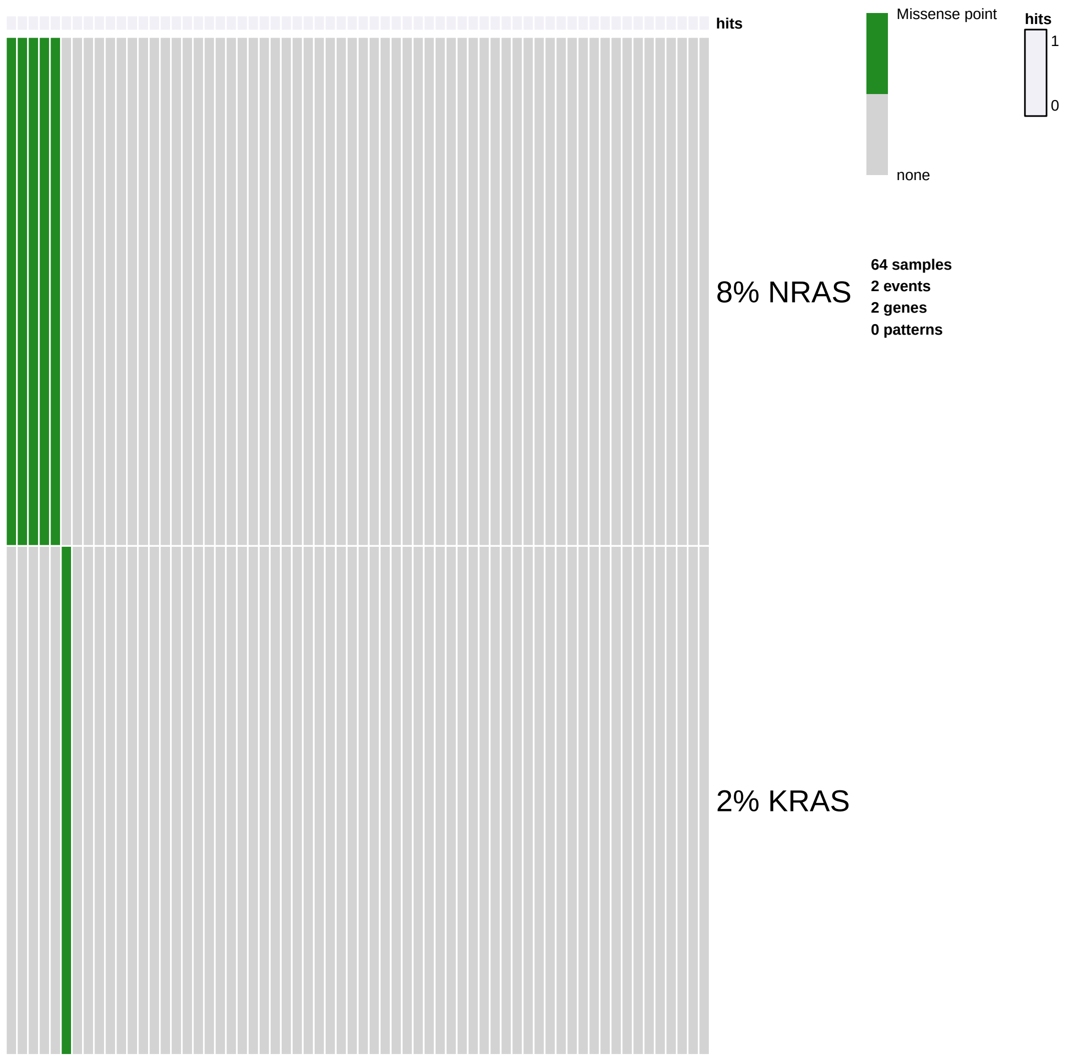

Figure 3: **Oncoprint output**

This is as above, but now it includes other hypotheses. For the same reason as before, we will include only the hard exclusivity pattern.

```
> hypo = hypothesis.add(hypo, 'SF3B1 xor ASXL1', XOR('SF3B1', OR('ASXL1')), '*')
```

```
> hypo = hypothesis.add(hypo, 'SF3B1 or ASXL1', OR('SF3B1', OR('ASXL1')), '*')
```

Finally, we now do the same for the genes TET2 and IDH2. In this case 3 events for the gene TET2 are present, that is *"Ins/Del"*, *"Missense point"* and *"Nonsense point"*. For this reason, since we are not specifying any subset of such events to be considered, all TET2 alterations are used. The patter among these events will be added both in XOR and in OR.

```
> as.events(hypo, genes = 'TET2')
```

```
        type             event
gene 4  "Ins/Del"        "TET2"
gene 32 "Missense point" "TET2"
gene 88 "Nonsense point" "TET2"
```

```
> hypo = hypothesis.add(hypo, 'TET2 xor IDH2', XOR('TET2', 'IDH2'), '*')
> hypo = hypothesis.add(hypo,  'TET2 or IDH2', OR('TET2', 'IDH2'), '*')
```

```
> dummy = oncoprint(events.selection(hypo, filter.in.names = c('TET2', 'IDH2')))
```

```
*** Events selection: #events=21, #types=4 Filters freq|in|out = {FALSE, TRUE, FALSE}
[filter.in] Genes hold: TET2, IDH2 ...  [2/2 found].
Selected 4 events, returning.
*** Oncoprint for ""
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Setting automatic row font (exponential scaling): 13.8
```

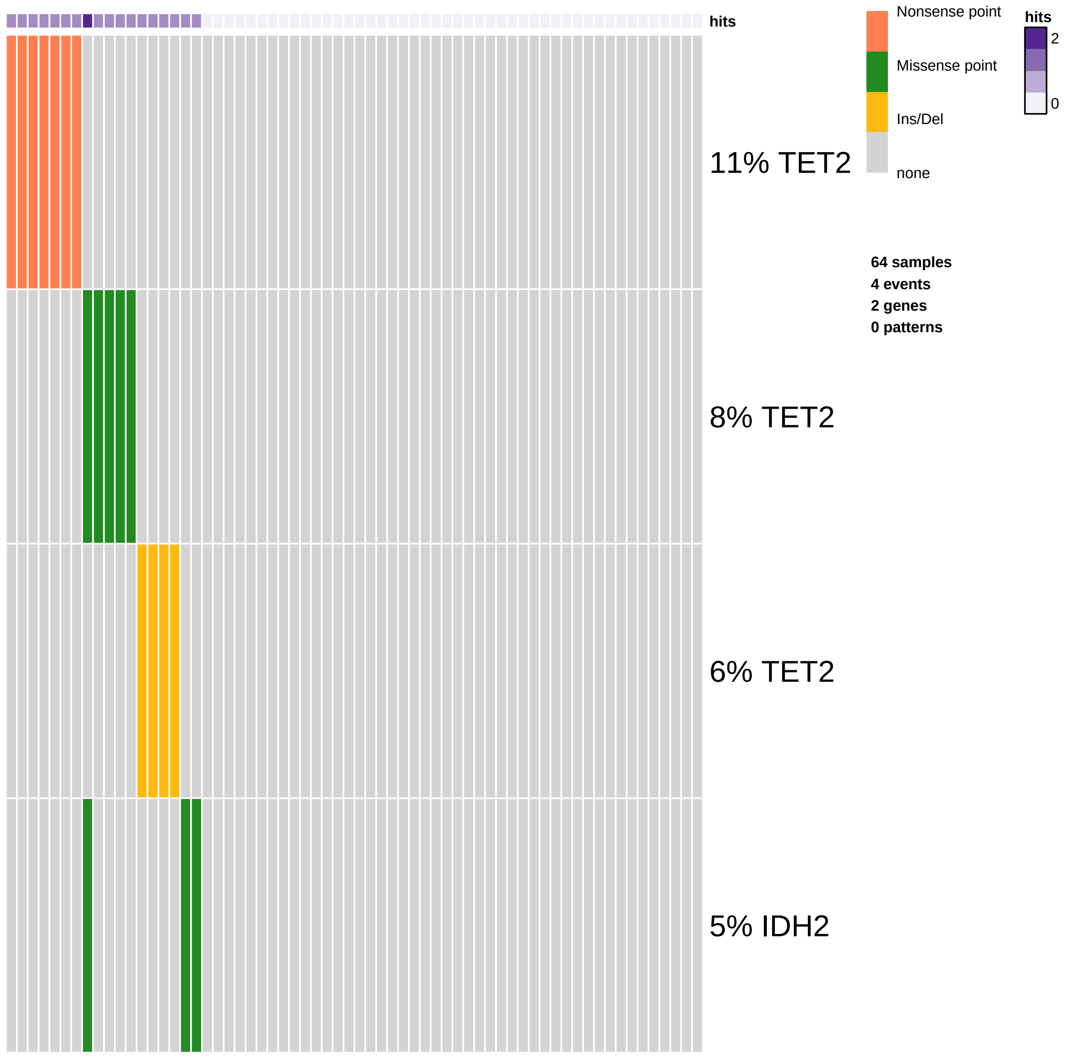

Figure 4: **Oncoprint output**

We now add possible homologous events. For every gene that has more than one event associated we also add a soft exclusivity pattern for its events.

```
> hypo = hypothesis.add.homologous(hypo)
```

```
*** Adding hypotheses for Homologous Patterns
 Genes: TET2, EZH2, CBL, ASXL1, CSF3R
 Function: OR
 Cause: *
 Effect: *
Hypothesis created for all possible gene patterns.
```

The final dataset that will be given as input to CAPRI is now shown.

```
> dummy = oncoprint(hypo,  gene.annot = list(priors= gene.hypotheses), sample.id = T)
```

```
*** Oncoprint for "CAPRI - Bioinformatics aCML data (selected events)"
with attributes: stage=FALSE, hits=TRUE
Sorting samples ordering to enhance exclusivity patterns.
Annotating genes with RColorBrewer color palette Set1 .
Setting automatic row font (exponential scaling): 8.9
Setting automatic samples font half of row font: 4.5
```

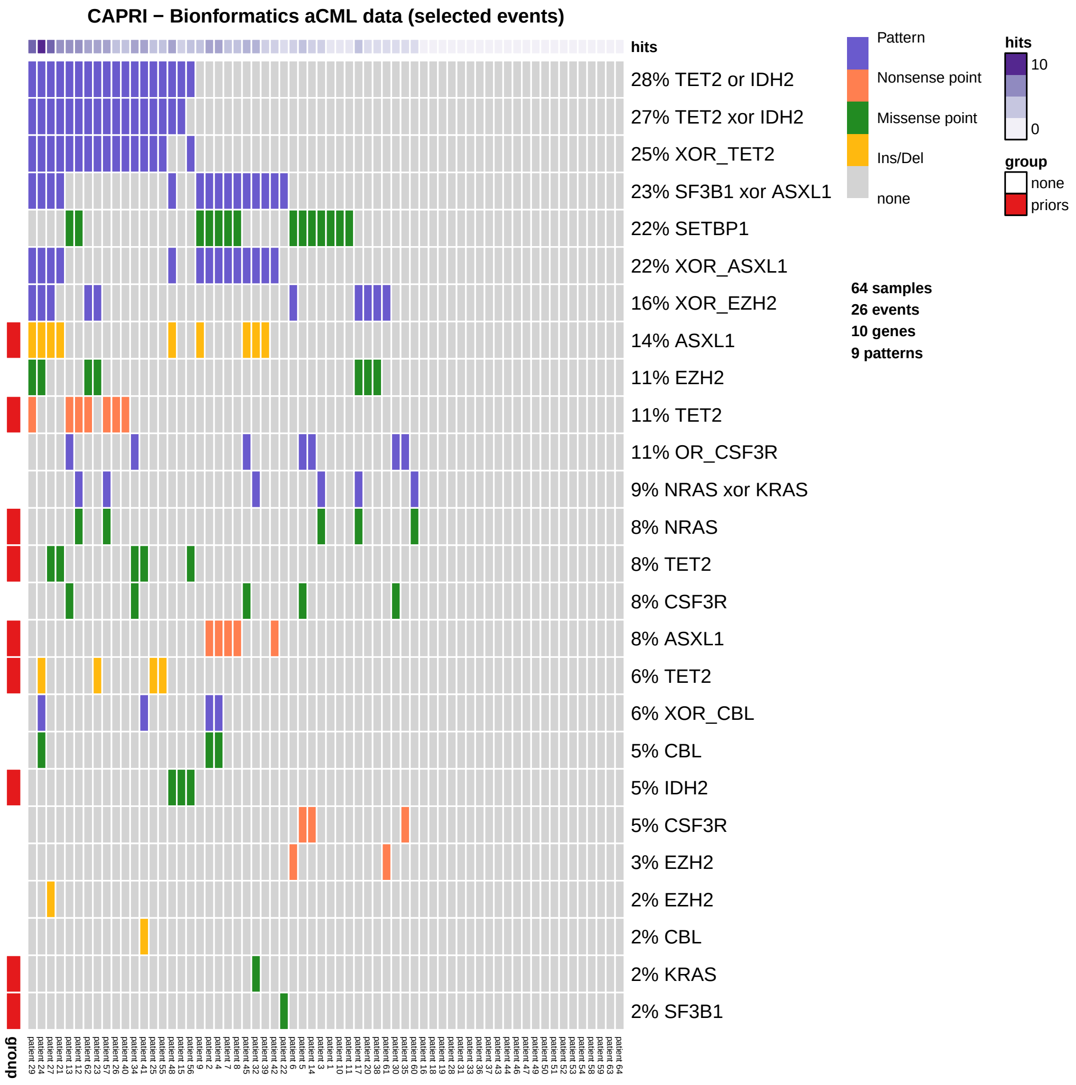

Figure 5: **Oncoprint output**

Model reconstruction

We run the inference of the model by CAPRI with its default parameter: we use both AIC/BIC as regularizators, Hill-climbing as solution heuristic search and exhaustive bootstrap (nboot replicates or more for Wilcoxon testing, i.e., I can perform more then nboot iteration if I reject samples), p-value set at 0.05. We set the seed for the sake of reproducibility.

```
> model = tronco.capri(hypo, boot.seed = 12345, regularization='bic', nboot=6)
```

```
*** Checking input events.
*** Inferring a progression model with the following settings.
        Dataset size: n = 64, m = 26.
        Algorithm: CAPRI with "bic" regularization and "hc" likelihood-fit strategy.
        Random seed: 12345.
        Bootstrap iterations (Wilcoxon): 6.
                exhaustive bootstrap: TRUE.
                p-value: 0.05.
                minimum bootstrapped scores: 3.
*** Bootstraping selective advantage scores (prima facie).
        Evaluating "temporal priority" (Wilcoxon, p-value 0.05)
        Evaluating "probability raising" (Wilcoxon, p-value 0.05)
*** Loop detection found loops to break.
        Removed 36 edges out of 70 (51%)
*** Performing likelihood-fit with regularization bic.
The reconstruction has been successfully completed in 00h:00m:02s
```

We can plot the reconstructed model. We set some parameters to get a fancy plot; confidence is shown both in terms of temporal priority and probability raising (selective advantage scores) and hypergeometric testing (statistical relevance of the dataset of input).

```
> tronco.plot(model,
+   fontsize = 13,
+   scale.nodes = .6,
+   confidence = c('tp', 'pr', 'hg'),
+   height.logic = 0.25,
+   legend.cex = .5,
+   pathways =  list(priors= gene.hypotheses))
```

```
*** Expanding hypotheses syntax as graph nodes:
*** Rendering graphics
Nodes with no incoming/outgoing edges will not be displayed.
Annotating nodes with pathway information.
Annotating pathways with RColorBrewer color palette Set1 .
Set automatic fontsize for edge labels: 6.5
Adding confidence information: tp, pr, hg
RGraphviz object prepared.
Plotting graph and adding legends.
```

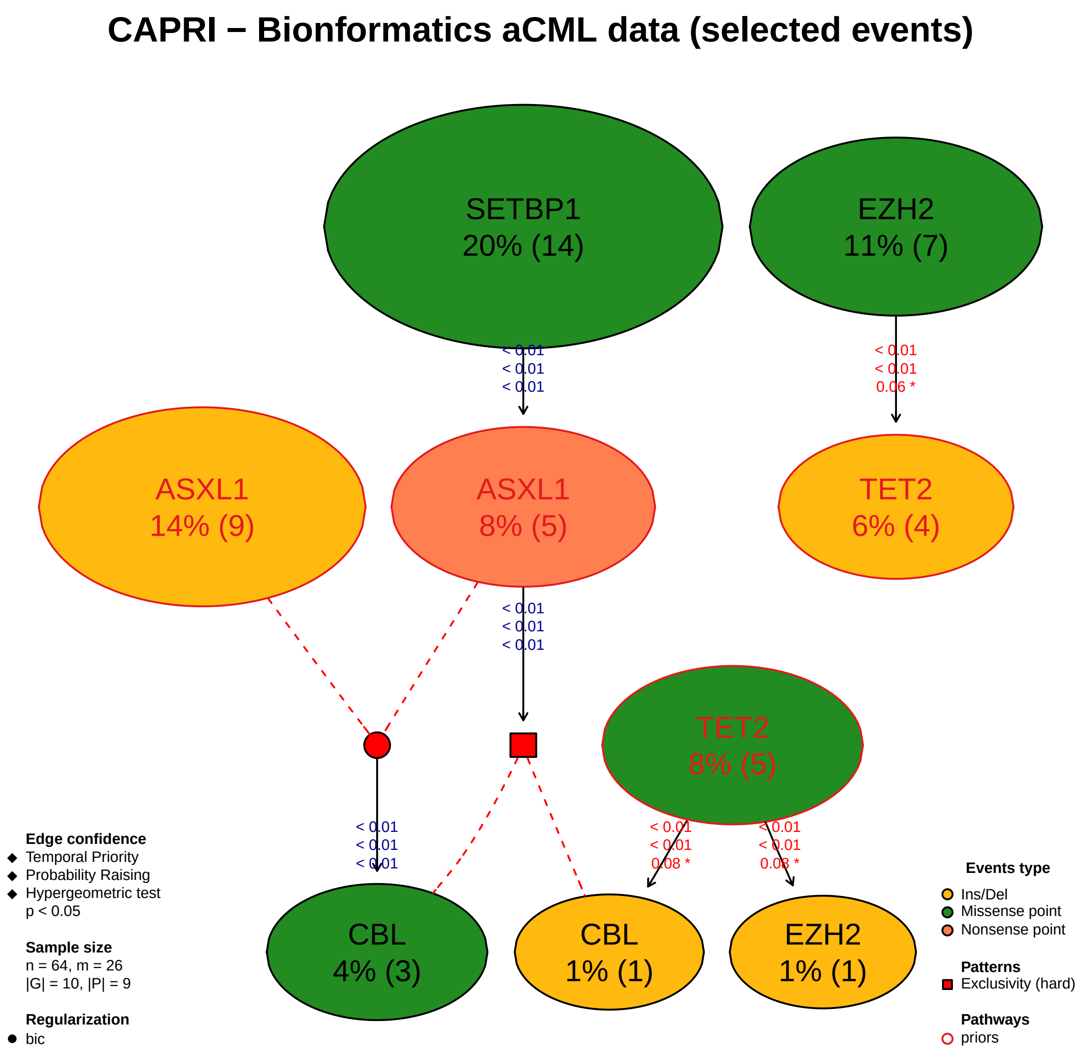

Figure 6: **aCML Reconstructed model** Pre bootstrap.

Bootstrapping data

Finally, we perform non-parametric bootstrap as a further estimation of the confidence in the results.

```
> model.boot = tronco.bootstrap(model, nboot=6)
```

```
Executing now the bootstrap procedure, this may take a long time...
Expected completion in approx. 00h:00m:01s
*** Using 7 cores via "parallel"

*** Reducing results

Performed non-parametric bootstrap with 6 resampling and 0.05 as pvalue
for the statistical tests.
```

```
> tronco.plot(model.boot,
+             fontsize = 13,
+             scale.nodes = .6,
+             confidence=c('npb'),
+             height.logic = 0.25,
+             legend.cex = .5)

*** Expanding hypotheses syntax as graph nodes:
*** Rendering graphics
Nodes with no incoming/outgoing edges will not be displayed.
Set automatic fontsize for edge labels: 6.5
Adding confidence information: npb
RGraphviz object prepared.
Plotting graph and adding legends.
```

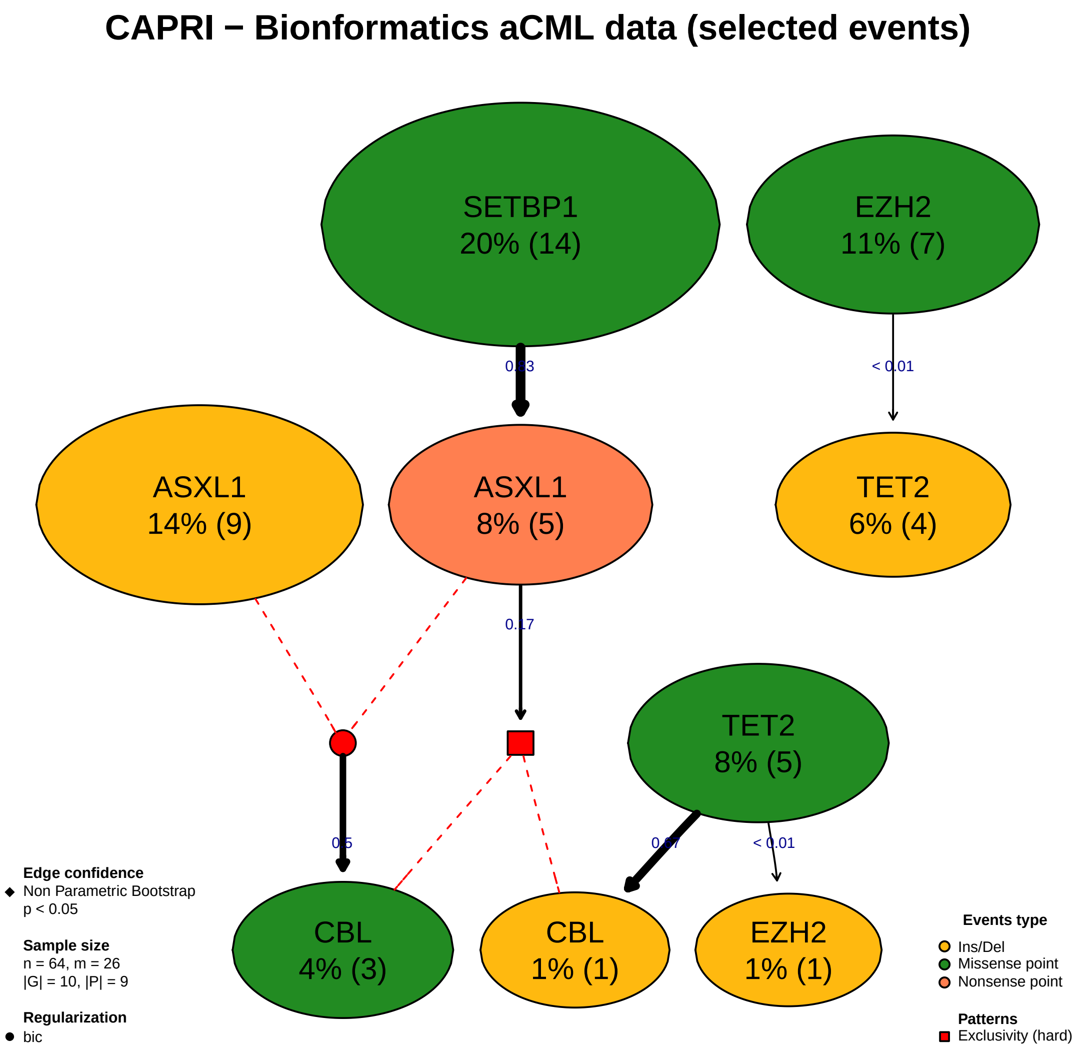

Figure 7: **aCML Reconstructed model** After bootstrap.