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[*]

July 20, 2015

Abstract. Genotype-level *cancer progression models* describe the ordering of accumulating mutations, e.g., somatic mutations / copy number variations, during cancer development. These graphical models help understand the "causal structure" involving events promoting cancer progression, possibly predicting complex patterns characterising genomic progression of a cancer. Reconstructed models can be used to better characterise genotype-phenotype relation, and suggest novel targets for therapy design.

tronco (tr*anslational* onco*logy*) is a r package aimed at collecting state-of-the-art algorithms to infer *progression models* from *cross-sectional* data, i.e., data collected from independent patients which does not necessarily incorporate any evident temporal information. These algorithms require a binary input matrix where: (i) each row represents a patient genome, (ii) each column an event relevant to the progression (a priori selected) and a $0/1$ value models the absence/presence of a certain mutation in a certain patient.

The current first version of tronco implements the caprese algorithm (ca*ncer* pr*ogression* e*xtraction with* s*ingle* e*dges*) to infer possible progression models arranged as *trees*; cfr.

- *Inferring tree causal models of cancer progression with probability raising*, L. Olde Loohuis, G. Caravagna, A. Graudenzi, D. Ramazzotti, G. Mauri, M. Antoniotti and B. Mishra. PLoS One, *to appear*.

This vignette shows how to use tronco to infer a tree model of ovarian cancer progression from CGH data of copy number alterations (classified as gains or losses over chromosome's arms). The dataset used is available in the SKY/M-FISH database. The reference manual for tronco is available in the package.

Requirements: You must have `rgraphviz` installed to use the package, see `Bioconductor.org`.

Event selection

First, load tronco in your R console and the example *"dataset"*.

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

[*] 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.

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

```
> show(data)
```

```
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
```

These are all the events it contains

```
> as.events(data)
```

```
         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(data)
```

```
 [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"
```

These are `SETBP1` alterations across input samples

```
> as.gene(data, 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
```

These are the genes for which we found a literature supporting the patterns that we include below. References are in the main CAPRI paper.

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

Regardless the distinct types of mutations that we included, we want to select only genes altered in 5% of the cases. Thus we first transform data in ”Alteration” (collapsing all event types for the same gene), and then we use select only those events

```
> alterations = events.selection(as.alterations(data), 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).
[1] TRUE
Selected 7 events.

Selected 7 events, returning.
```

We visualize the selected genes. This plot has no title since name annotation is not copied by `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

Then to reconstruct the aCML model we select from `data` **which have been selected in** `alteration` **- via** `as.genes(alterations)` **or that are part of the prior in** `gene.hypotheses`**. We use** `filter.in.names` **to force selection of all the events involving those genes from** `data`

```
> hypo = events.selection(data, 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 show selected data and we annotate genes in `gene.hypotheses` **to identify them. Samples 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 add the hypotheses that are described in CAPRI's manuscript

Add hypotheses of hard exclusivity (XOR) for NRAS/KRAS events (Mutation). The hypothesis is tested against all other dataset events

```
> 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 commented because it gives errors.

```
> ### do not run
> # hypo = hypothesis.add(hypo, 'NRAS or KRAS',  OR('NRAS', 'KRAS'))
> ###
```

For the sake to better highlight the perfect (hard) exclusivity between NRAS/KRAS mutations one can visualize 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 includes other events. Again, we can include only the hard exclusivity pattern

```
> hypo = hypothesis.add(hypo, 'SF3B1 xor ASXL1', XOR('SF3B1', OR('ASXL1')), '*')
> ### do not run
> # hypo = hypothesis.add(hypo, 'SF3B1 or ASXL1', OR('SF3B1', OR('ASXL1')), '*')
> ###
```

We now do the same for TET2 and IDH2. In this case 3 events for TET2 are present, which are "Ins/Del", "Missense point" and "Nonsense point". For this reason, since we are not specifying a subset of such events all TET2 alterations are used. Since these show a perfect hard exclusivity trend these will be included in XOR.

```
> 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'), '*')
> ### do not run
> # hypo = hypothesis.add(hypo,  'TET2 or IDH2', OR('TET2', 'IDH2'), '*'))
> ###
```

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

```
*** Events selection: #events=20, #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**

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 dataset input to CAPRI is 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): 9.1
Setting automatic samples font half of row font: 4.5
```

Figure 5: **Oncoprint output**

Model reconstruction

We execute CAPRI with its default parameter: we use both AIC/BIC regularizators, Hill-climbing exhaustive bootstrap (100 replicates for Wilcoxon testing), p-value 0.05 and we set seed

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

```
*** Checking input events.
*** Inferring a progression model with the following settings.
        Dataset size: n = 64, m = 25.
        Algorithm: CAPRI with "bic" regularization and "hc" likelihood-fit strategy.
        Random seed: 12345.
        Bootstrap iterations (Wilcoxon): 100.
```

```
                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 42 edges out of 79 (53%)
*** Performing likelihood-fit with regularization bic.
The reconstruction has been successfully completed in 00h:00m:11s
```

We can plot the reconstructed model. We set some parameters to get a fancy plot; confidence is shown as temporal priority and probability raising (selective advantage scores) and hypergeometric testing (goodness of input data).

```
> 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.
```

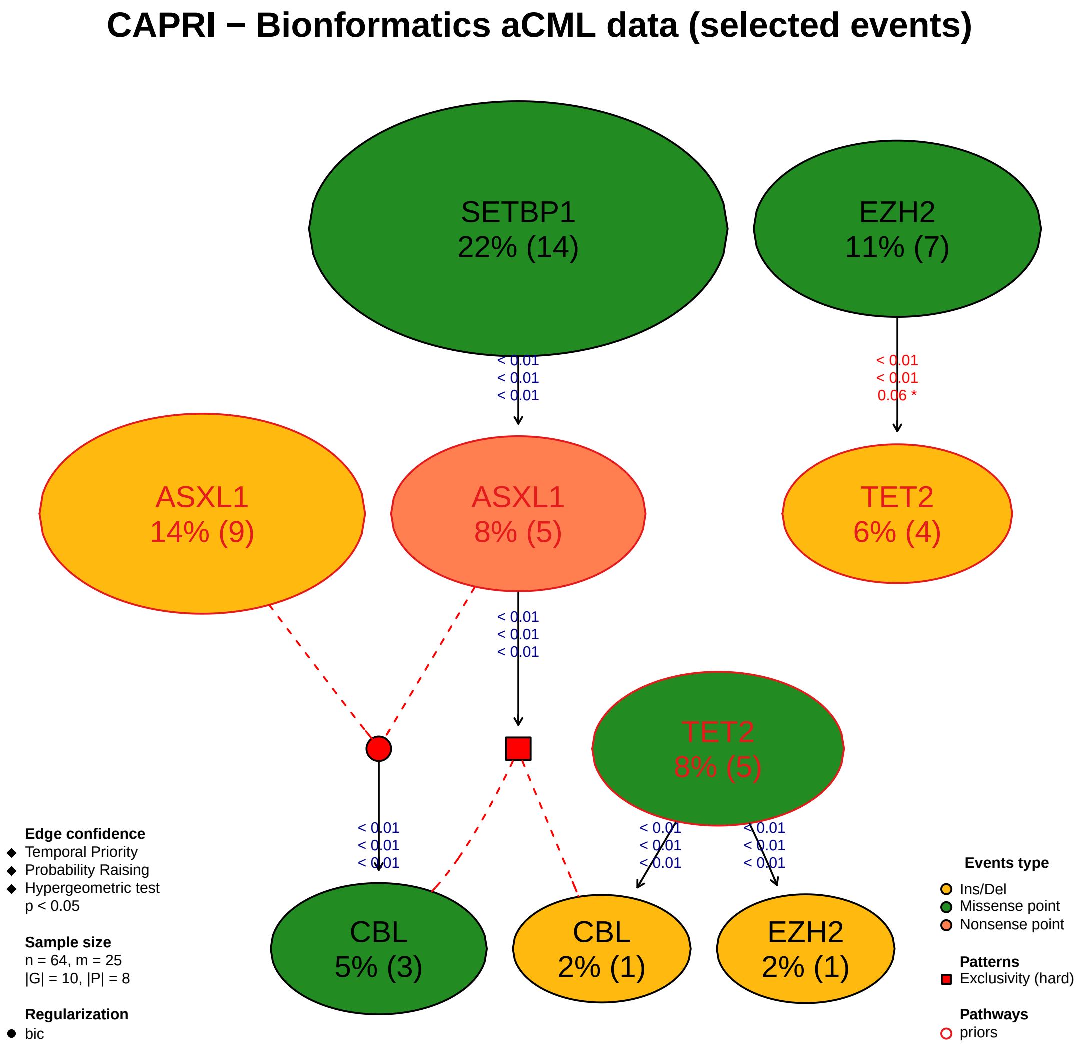

Figure 6: **aCML Reconstructed model** Pre bootstrap.

Bootstrapping data

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

Executing now the bootstrap procedure, this may take a long time...
Expected completion in approx. 00h:00m:21s
*** Using 3 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.
```

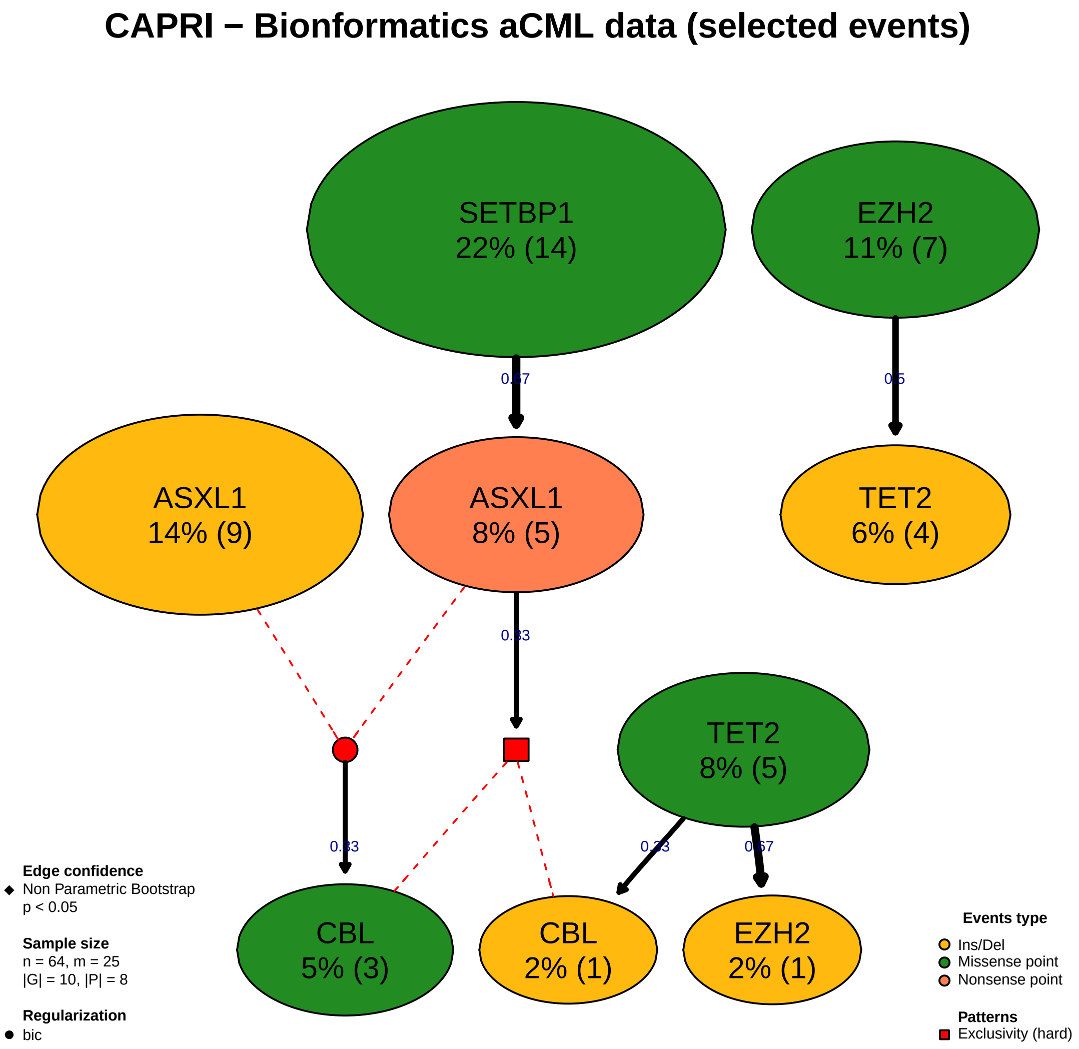

Figure 7: **aCML Reconstructed model** After bootstrap.